Implementation and Evaluation of an IVR Rendering Platform for Data Collection in the Developing World

by

Adam Lerer

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

Author .....
Department of Electrical Engineering and Computer Science
July 6, 2010

Certified by .....
Saman Amarasinghe
Professor
Thesis Supervisor

Accepted by .....
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Implementation and Evaluation of an IVR Rendering Platform for Data Collection in the Developing World

by

Adam Lerer

Submitted to the
Department of Electrical Engineering and Computer Science

July 6, 2010

in partial fulfillment of the requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Due to the rapid spread of mobile phones and coverage in the developing world, mobile phones are being increasingly used as a technology platform for developing-world applications. Data collection is one such application area, and a variety of software has been written to enable data collection over mobile phones. However, reaching the vast majority of mobile phone users without access to specialized software requires a data collection strategy that operates over IVR or SMS. We have developed ODK Voice, an IVR platform for delivering data collection protocols based on the XForms standard and targeted at users in the developing world.

User testing of ODK Voice was performed both in controlled scenario experiments, and in a real-world deployment in Uganda. In controlled experiments in the United States, users were able to complete a complex survey with high accuracy. However, in a real-world deployment with teachers in rural Uganda lacking training or IVR experience, a number of significant interface modifications were required in order to achieve high success rates. The task success rate increased from near 0% to over 75% based on changes in interface design. Notably, most participants were not able to use a touchtone or touchtone-voice hybrid interface without prior training. A set of design recommendations is proposed based on the performance of users in Uganda on several interface iterations.

Thesis Supervisor: Saman Amarasinghe
Title: Professor

Acknowledgments

I would like to thank Professor Saman Amarasinghe and Bill Thies for their mentorship and guidance throughout the entirety of this project; Professor Gaetano Borriello, Yaw Anokwa, Carl Hartung, Waylon Brunette, and the rest of the ODK team, for their support and collaboration; Neil Lesh, for shaping the initial direction of this project; Molly Ward and the Project WET team for helping to get ODK Voice deployed in Uganda; as well as the JavaRosa team, Rob Miller, and Latif Alam.

Contents

List of Figures

Chapter 1

Introduction

1.1 Mobile Data Collection in the Developing World

In the past several years, there has been a growing adoption of mobile phone technology as a tool for solving a variety of challenges in international development, including health delivery, disaster management, microbanking, sanitation, and education. This new focus on technology is a result of the explosive growth of mobile phone usage and coverage throughout the developing world. As of 2008, there were 4.1 billion worldwide mobile phone subscribers, compared to just 1.5 billion internet users [35]. 60% of these mobile phone users live in the developing world [34]. Millions of people, many of whom have never used a computer and earn only a couple dollars a day, now own their own mobile phone; this trend is revolutionizing the communication patterns of the poor, and enabling a wide range of potential technological solutions that were not possible a decade ago.

One important use of mobile technology in the developing world - particularly but not exclusively for health applications - is data collection. Collecting data in the developing world presents a number of unique challenges: a diffuse rural population, low literacy and education, and a lack of financial resources. Current data

collection practices are primarily paper-based, leading to inneficiencies such as slow turn-around/aggregation time, non-standardized data, and lack of data security, validation and integrity [4]. For example, in much of the rural developing world, the first line of health services are provided by lightly trained, mobile 'community health workers', or CHWs. CHWs travel to individual villages and homes, providing basic health education, diagnosis, and triage. Data collection is performed using (often ad hoc) paper forms, such as the one shown in Figure 1-1. It can take weeks for these forms to reach a central location where the data can be aggregated or viewed by a doctor, there is no data validation or standardization, and forms can be lost or damaged. Mobile phones have the potential to be faster and less error-prone, and open up a wide range of possibilities for more intelligent data protocols and integration as well as the ability to collect audio, image, and barcode data [4].

While organizations have in the past developed mobile software to collect data for particular applications, these projects were unable to interoperate. Recently, the OpenROSA consortium was created to bring together a number of organizations in order to promote open standards-based data collection for public health in the developing world [18]. OpenROSA has chosen XForms as their form specification standard, augmented by additional

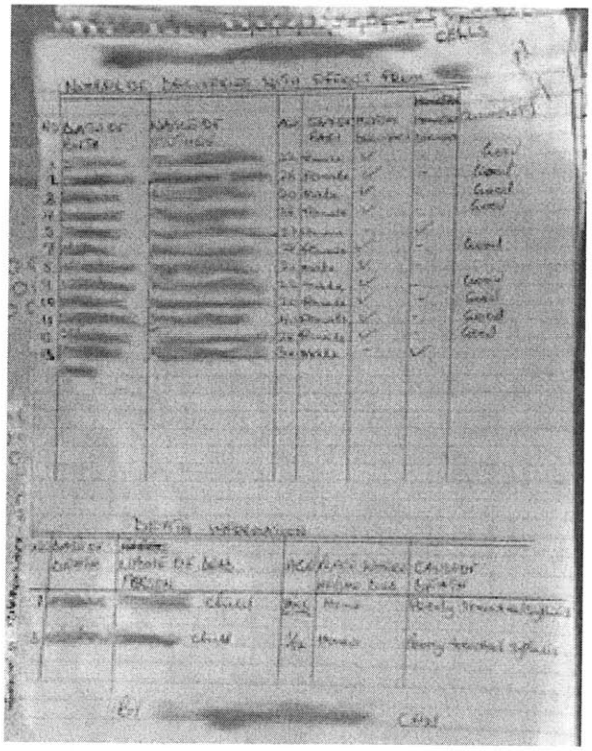

Figure 1-1: Community health workers collect data using paper forms that are non-standardized, error-prone, and slow to process.

community standards as necessary. XForms is the W3C specification for next-generation forms, and includes support for a variety of data types and UI widgets as well as complex branching and constraint logic. As a result of the OpenROSA effort, a number of applications have been developed for rendering XForms on mobile devices, including JavaRosa, EpiSurveyor, and Open Data Kit, as well as for aggregating and analyzing collected XForms data [4, 18]. A number of organizations (e.g. Dimagi, D-Tree, AMPATH) are now using these applications to collect standardized patient data in the field, execute decision protocols, and aggregate and analyze data in ways that were not possible using paper-based methods.

ODK Voice is part of the Open Data Kit project, which is developing a suite of OpenROSA-compliant tools for collecting and aggregating data over mobile phones. ODK was originally designed specifically for community health workers, but has now spread to use in a number of applications ranging from forest management to census taking [16].

ODK Voice is an ODK-integrated platform for collecting data using automated voice dialogues (IVR) over phone calls.

1.2 Data Collection Over Voice

Although the use of mobile phones as a platform has drastically lowered the bar for automating data collection, existing solutions nevertheless require access to particular mobile phones running particular software. Thus, data collection requires a health worker to visit each patient and collect data; there is no way for patients to report data for themselves on their mobile phone. The same principle applies for other potential application areas, such as disaster reporting, environmental monitoring and microfinance.

Expanding the reach of mobile data collection to all mobile phone users requires the use of either voice or SMS channels, since these capabilities are available on nearly all mobile phones. Of these, only voice is suitable for answering a series of questions (although SMS could play an ancillary role in collection). Therefore, an interactive voice response (IVR) platform for rendering XForms is the natural choice for expanding the reach of XForms-based data collection beyond customized smartphones and PDAs.

In addition to expanded reach, voice-based data collection has several additional advantages:

1. Using voice-based communication circumvents the serious incentive problems in more common, SMS-based ICTD programs (e.g. those using FrontlineSMS [1]). Since sending SMS costs respondents money, there must be additional incentives for them to respond; on the other hand, a phone call initiated by the survey application does not incur a cost to the respondent. In fact, we were approached by a partner organization specifically because their respondents were not willing to pay SMS fees to respond to their SMS-based data collection program.
2. An audio interface is particularly suitable for illiterate and semi-literate users, who are unable to use text-based (or SMS-based) interfaces [21].
3. There is preliminary evidence that data collected over voice in resource-poor areas is significantly more accurate than data collected by either SMS or custom mobile applications. One study in Gujarat found that voice entry using human voice operators had 0.45% error rates, while SMS and mobile data entry had error rates over 4% [25]. See Section 2.3.
4. A number of studies have shown that data collection through an automated voice system is significantly more effective at obtaining sensitive information than a live interviewer [22, 33]. This is extremely important in public health applications, where organizations are collecting information such as HIV status

from respondents.

There are many potential applications for ODK Voice, including:

- An automated helpline run by a community health organization. Patients could call in to receive health advice, hear instructions on taking their medication, or report an illness. Health and medication information could be provided based on built-in form decision logic.
- Automated check-in or reminders for regular medication adherence (e.g. direct observation therapy).
- Collecting feedback on development initiatives undertaken in regions where data cannot be collected in person (see Chapter 5).
- Disaster management applications such as family tracing and reunification (FTR), if cellular networks are available.
- Mobile citizen reporting of crime, corruption, etc.

1.3 Challenges for Voice Interfaces for Resource-Poor Regions

Creating a voice-based data collection platform for use in resource-poor regions poses both technical and usability challenges. ODK Voice is both an attempt to address some of these challenges and to evaluate the types of voice interfaces and interactions that are effective in these particularly difficult conditions. This section briefly describes some challenges and how we have addressed them.

1.3.1 Usability

The most serious challenge for any automated voice system in a developing world setting is usability. Even in the best of circumstances, most voice interfaces present

usability challenges such as the conventions of spoken language, limitations of speech recognition, limitations of human cognition and working memory, and differences between users [32] These usability problems are exacerbated by a user population who lacks experience using voice interfaces or even other automated interfaces, and who often have a low level of education and literacy [21]. Several projects have had success with voice interfaces in resource poor environments, but they have all provided training to their users in one form or another. We believe that interfaces that assume prior training or IVR experience are less scalable and inappropriate for many of the exact situations for which IVR interfaces are particularly compelling: namely, situations in which average citizens, rather than trained users, interact with the system. Our goal is to maximize voice interface success with limited or no user training.

In order to make ODK Voice more learnable for new users, we traded efficiency for learnability in a number of initial design decisions, such as detailed instructions, explicit response confirmation, and a minimal number of options. More importantly, an iterative design process involving a cycle of live user testing in Uganda and interface redesign was undertaken to improve interface usability (see Chapter 5). This design process improved task success rate from 5% (already after several iterations) to over 75%.

1.3.2 Language and Literacy

Issues of language and literacy create challenges for voice interface design, especially in the developing world. Many developing countries have dozens of local languages and dialects, almost none of which are supported by available text-to-speech (TTS) and automatic speech recognition (ASR) packages. Even within a single region, different participants may have widely varying abilities to understand and speak different languages.

ODK Voice is designed to handle the language and literacy challenges present in the developing world. ODK Voice is fully localizable and designed to be language-agnostic. This requires us to rely on recorded prompts and keypad (DTMF) input more heavily instead of TTS and ASR. We also enable the creation of multilingual surveys and allow participants to switch languages/dialects within the survey. Finally, we attempt to make questions robust to language and literacy problems (e.g. correcting spelling errors in string entry, requiring answer confirmation).

1.3.3 Resources and Infrastructure

An obvious constraint on any technology platform developed for resource-poor regions is the lack of financial resources and technical infrastructure in these regions. Most organizations operating in these regions have much tighter financial constraints than equivalent organizations elsewhere, and the financial incentives for automation are lessened since labor costs are much lower in these regions. Furthermore, participants are less willing to use automated systems that may incur cell phone charges. Automated systems are also hampered by infrastructural challenges such as power and internet service outages, and cellular network unreliability.

ODK Voice attempts to minimize resource requirements for organizations and respondents. ODK Voice is standards-based and can be configured on a completely free and open source software stack (see Chapter 4). The only operating expenses for an ODK Voice instance are (a) purchase and maintenance of a server, and (b) cellular network usage charges. This can be significantly cheaper than a software-based data collection methodology, which entails hiring data collectors and providing them with mobile devices.

Even more importantly, ODK Voice can be hosted anywhere (e.g. in the cloud) and can connect to regional cell networks through a voice-over-IP (VoIP) provider. Therefore, organizations can conduct surveys with no on-the-ground technical exper-

tise, and they will not be affected by infrastructural outages.

To avoid cell phone charges for survey respondents, outgoing calls can be scheduled from the application to respondents, since cell subscribers are not charged for incoming calls in most of the developing world. However, we learned that many of the usability challenges of a voice interface are even more severe when the calls are initiated by the application, so there is a tradeoff between incentivization and usability.

1.3.4 End User Programming

Most of the automated voice applications in use today are designed by specialists in voice interface design. Voice interfaces are well-studied, and there is a significant body of literature devoted to the technical and usability issues of creating voice applications [7, 23, 32]. However, ODK Voice is targeted at small, community-based organizations who do not employ voice interface specialists and may not even employ dedicated technology staff.

To facilitate end-user programming, ODK Voice is designed to require minimal technical expertise. There are two components to deploying ODK Voice: (a) setting up the server, and (b) creating and configuring surveys on the server. Setting up the server requires installing various software and interfacing with a VoIP provider, but this is a one-time operation and could be mostly packaged inside a virtual machine. Surveys are specified as XForms, so the survey creation process is aided by a large community of XForms designers and a number of XForms design tools, such as ODK Build [2]. Once an XForm is uploaded, the only additional step required is prompt recording, which can be done over the phone.

The issue of voice interface expertise is only partially solved by ODK Voice. The system is designed so that the interfaces it creates from standard, non-voice-specific XForms are as usable as possible; additional customization can be provided by voice-

specific rendering attributes added to the form, as well as by customization of the software itself. However, there is no replacement for UI design expertise, and for large organizations/deployments ODK Voice should be thought of as a piloting/prototyping tool that could lead to a more comprehensively designed and tested system.

1.4 Evaluation of ODK Voice in Uganda

The ODK Voice interface was deployed in Uganda to collect feedback from teachers on a water sanitation and hygiene training program delivered in July and August 2009. The conditions of this deployment were different than previous studies in that not only was the deployment in rural Uganda and the teachers lacked previous IVR experience, but the teachers received no training or assistance in using the interface and were called at a random time outside of a controlled environment. We consider these conditions to be more realistic for scalability of an end-user voice application. The goal of this deployment was to determine the coniditions and types of interaction that would be successful in collecting data under these conditions.

The survey interface underwent a number of iterations over a period of several months, each of which was tested on real participants in Uganda. Success rates improved dramatically between the initial and final versions of the interface, from 5% to over 75% (see Chapter 5). A summary of design recommendations based on this work is provided in Chapter 7.

1.5 Chapter Outline

Chapter 1 provides background and motivation for the problems being addressed and a summary of the important features of ODK Voice. Chapter 2 describes recent work related to data collection in the developing world, voice interfaces, and evaluation

of mobile interfaces in the developing world. Chapter 3 contains the methodology and results of early-stage research that informed our development of ODK Voice. Chapter 4 documents the features, software architecture, and novel technical aspects of ODK Voice. Chapter 5 describes a deployment of ODK Voice in rural Uganda, and the results of a number of iterations of the ODK Voice interface tested on users in Uganda. Chapter 6 documents the results of preliminary work characterizing the accuracy of ODK Voice. Chapter 7 contains a summary of the results of this work and concluding remarks. Chapter 8 suggests extensions to ODK Voice and future research on voice interfaces in the developing world.

Chapter 2

Related Work

2.1 Mobile Data Collection

The ODK project is part of a consortium of data collection organizations called OpenRosa that is promoting XForms as a standard for data collection in the developing world. The goal of the OpenROSA community is to create interoperable, standards-based data collection tools instead of the silo-ed, proprietary medical systems that have plagued current medical practice [18]. Other OpenROSA XForms-based data collection tools, such as JavaRosa and EpiSurveyor, have shown initial promise in improving the speed and accuracy of data collection by health workers, although there is not yet quantitative data to support these claims [18, 4]. The Open Data Kit project has worked with a number of community health organizations, including Grameen AppLab and AMPATH, to improve the data collection and management process of community health workers in Africa.

There have been a number of previous studies evaluating technology-assisted data collection by rural health workers. Reported error rates on mobile phones and PDAs range from less than 1% [10, 9] to 5% [25] after several hours of training. Accuracy is highly training-dependent however: a study where users had only 2-3 minutes of

training reported an error rate of 14%. Several other studies [18, 4, 28, 8, 5] have reported qualitative improvements but have not quantitative characterized speed or accuracy improvements.

2.2 Speech Interfaces

There is a large body of work on speech interfaces in the developed world. Commercial interfaces tend to focus on simple task completion, especially for call center operation. Several authors have provided guidelines for creating usable voice interfaces (e.g. [7, 23, 32]), with many ideas drawn from the field of computer-human interaction, such as the iterative design process, rapid prototyping, and heuristic and user evaluation techniques. Academic projects such as MIT's Jupiter weather information system [11] and Carnegie Mellon's Communicator travel information interface [27], as well as some modern commercial interfaces, allow for conversational, user-directed call flow, accepting a wide range of user utterances. However, these systems target neither the needs [30] nor the usability challenges of people in resource- and literacy-poor regions, since these users are generally not considered economically valuable consumers of commercial services.

A number of previous studies have designed and evaluated voice interfaces in the developing world for applications such as health reference [29, 31], microbanking [21], real-time information dissemination [15, 24, 26], citizen reporting [19], and data collection [25]. Berkeley's Tamil Market project [26] was the first speech interface that was rigorously evaluated with low-literacy users in the developing world. The Tamil Market project provided a speech interface for daily crop prices and information, and speech input was restricted to mostly yes or no answers. Developers performed user studies and interviews, and recorded target users; recordings were used to tune the acoustic models for each village. By limiting the vocabulary, an error rate of less

than 2% was achieved with a small training sample. The study suggests that there are differences in task success between literate and illiterate users, but the sample sizes were too small to be conclusive.

Several subsequent studies have compared touchtone and voice input modalities for voice interfaces for illiterate or semi-literate users. Patel et al. designed Avaaj Otalo, a phone-based angricultural information system in Gujarat [24]. Avaaj Otalo consisted of an announcement board, an archive of agricultural radio programs, and a question-answering service. Patel et al. used Avaaj Otalo to compare success rates for touchtone and ASR input. Rather than using a trained language model, Patel converted the Gujarati words to an English phoneme set. Patel found that subjects with less than an eighth grade education performed significantly better using touchtone input than speech recognition. The OpenPhone team in South Africa developed a health helpline and also found that 'most of the low literacy caregivers in the study...preferred the touchtone system' [20].

Sherwani et al. have developed Healthline, a speech interface for low literacy health workers in Pakistan [31]. Sherwani used a hybrid lexical/statistical speech recognition system, using experts to encode Urdu pronounciations in terms of English phonemes with wildcards, and sample utterances to fill in the wild cards with their most accurate English pronounciations. A 'field kit' consisting of a mobile telephony server was also developed to perform rapid testing and prototyping of voice interfaces in the field. Sherwani found that task success in a speech interface was significantly higher than a touchtone interface (86% versus 61%). This contradicts the results of [24] and [20]. Literate participants had significantly higher task success rates than low-literate participants, and proper training procedures were essential for task success. Finally, this work emphasized the importance and difficulty of supporting local languages and dialects, as many of their participants spoke rare regional dialects, some of which did not even have a written form.

Finally, two notable example of systems being developed for end-user voice application development in the developing world are Freedom Fone [15] and AudioWiki [19]. Freedom Fone is an easy-to-use telephony platform primarily for information access (voice menus). AudioWiki is a speech/touchtone message board or 'wiki' where participants can listen to and contribute recorded audio information and moderate user contributions. AudioWiki has been deployed in rural India for citizen reporting.

2.3 Comparison of Voice to Other Interfaces

A recent study involving health workers in Gujarat compared data collection accuracy using three different mobile data collection methodologies: a mobile phone electronic forms interface, an SMS data encoding scheme, and transcription via a live voice operator [25]. Thirteen community health workers and hospital staff participated, with education ranging from 10 years to a B.A. degree. Participants were trained for 1-8 hours until they could successfully enter numeric and multiple-choice patient data using any of the three methodologies. Participants were then evaluated in a within-subjects experiment collecting data on fake patients, and the accuracy of the different collection methodologies were compared. The observed error rates for electronic forms and SMS were 4.2% and 4.5%, respectively, while the error rate for a live voice operator was 0.45%. The drastically lower error rate for voice operator transciption suggests that either (a) data collection over voice is more accurate than through a graphical interface, or (b) the dynamics of human interaction with an operator improve data collection accuracy. Evaluation of an automated voice interface in a similar experiment would be required to determine which of these factors plays a role in increased data accuracy.

Indrani et al. performed a usability study on banking UIs for money transfer for low-literate users in India [21]. The ability of users to complete transactions was

evaluated on a text UI, a 'rich' audiovisual UI, and a voice UI. Task completion rates were 0% for the text UI, 100% for the rich UI, and 72% for the voice UI. However, the voice UI took less than half the time, and the authors reported greater accuracy with voice. Indrani reported that users were hesitant to press buttons on the phone in the rich UI and preferred a conversational interaction (many of these users had no or limited experience using mobile phones). Indrani also found that for subjects without previous voice UI experience, women took significantly longer but had a higher probability of completing the task; qualitatively, women were 'more patient, attentive and slower when interacting with each of the functions in the voice-based UI', although this observation was based on a sample size of only 6. These gender differences are echoed in our results from Uganda, described in Chapter 5. Of the participants without voice UI experience, the success rate was 55%. Participants who were not successful reported being confused by the inflexible system responses: 'subjects would keep saying "What Sir"?,"Yes Sir", "Can't understand what you are saying, Sir", thinking it was a real person' [21].

All evaluations of low-literate users have cited effective training as crucial for task success with speech interfaces. The Healthline team found that human-guided tutorials were most effective for training users to use their interface[29]. Patnaik et al. trained CHWs for several hours until they could complete a survey successfully before evaluating the system[25]. We are interested in exploring what success rate and data quality can be achieved by users with little (in the case of CHW patients) or no training (in the case of survey respondents), and with the distractions and call quality of a realistic, rather than controlled environment. Evaluation under these conditions is a central component of our work.

Chapter 3

Formative Experiments and Design

This chapter describes the initial design process that informed ODK Voice. Section 3.1 describes a prototyping tool that was built and used to test various early design choices. Secion 3.2 explains some of the important design choices that were made in ODK Voice as a result of this prototyping, as well as the literature and our own ideas.

3.1 Wizard of Oz Experiment

During the initial design phase of ODK Voice, a number of important design alternatives needed to be quickly prototyped and tested with target users. To rapidly iterate on these early designs, a simple application for creating 'Wizard of Oz' prototypes was constructed. 'Wizard of Oz' prototypes are prototypes in which the backend functionality is replaced by a human experimenter in such a way that the user does not realize that he is not interacting with a real automated system. The Wizard of Oz prototyping technique was actually first used to prototype a speech transcription interface (a 'listening typewriter') in 1982 [12], although recently Wizard of Oz prototyping has become a common technique in modern user interface design practice.

Existing Wizard of Oz prototyping tools, e.g. [17] are designed to replace the speech recognition component of voice interfaces with a human operator. Since we

planned for ODK Voice to use touchtone input instead of speech recognition, the Wizard of Oz prototype was designed to accept touchtone, rather than speech, input. We also believed it was important that the participant conduct the survey on a mobile phone, not a computer interface, and not be aware that there was a human operator 'behind the curtain', which would change user behavior.

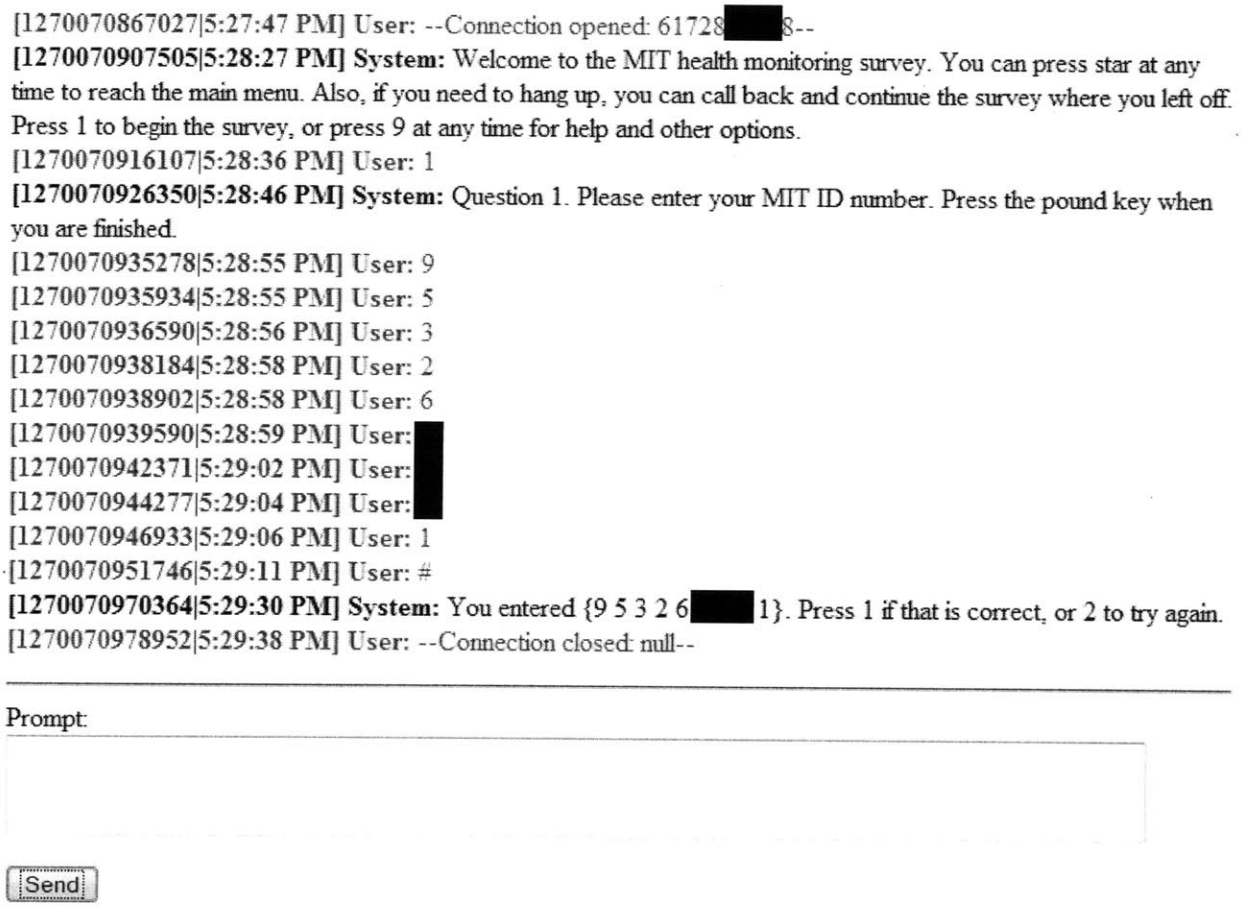

Figure 3-1: A screenshot of the 'control console' for VoiceSim, our Wizard of Oz prototype of ODK Voice. The experimenter copies and pastes prompts from a script which are played to the user via text-to-speech, and user touchtone responses are displayed. This screenshot shows the beginning of a survey call being delivered by an experimenter.

Our Wizard of Oz prototyping tool, called VoiceSim, allows an experimenter to convincingly simulate a touchtone voice interface for experiments. VoiceSim consists of a web-based 'control console' for the experimenter that behaves similarly to

a chat window; the experimenter can enter (copy and paste) text from a script into the chat window, which is played over the participant's mobile phone using text-to-speech. The user touchtone input appears on the chat window in response. No audio is exchanged between the participant and the experimenter, to mimic the constraints of a real touchtone-only audio interface. The control console from the middle of a sample experiment is shown in Figure 3-1. VoiceSim was built using Tomcat Java servlets generating VoiceXML, which was rendered using the Voxeo Prophecy VoiceXML client. This architecture is very similar to that used for ODK Voice, as described in Chaper 4.

We tested the Wizard of Oz prototypes with users in the MIT community. These users were not the target users of the ODK Voice system, but they represented an 'upper bound' on the abilities of our target users; at this early stage, our interface was far too poor to have any success with the target population. We initially planned to use the Wizard of Oz interface with our target population, but we eventually felt that it was too low-fidelity too be useful for testing on our target population. In particular, the difficulty of understanding the text-to-speech prompts, and the long delay between the user pressing buttons and hearing a response prompt (due to the experimenter having to cut and paste text into the chat window), made it unsuitable for a population with poor English literacy and a lack of understanding of IVR systems [1].

The test survey we used for our VoiceSim experiments was an 'MIT Health Survey', in which we asked users a number of questions of different types about their demographics and health:

1. Please enter your MIT ID number. *[numeric]*
2. Please enter your last name. *[string]*
3. How many hours per week do you spend watching TV? *[single-select]*

[1] Other types of Wizard of Oz voice interface prototyping *can* be used succesfully with low-literacy users; see e.g. [25].

4. Which of the following activities to you take part in at least once a week? *[multi-select a]*
5. Which of the following activities have you engaged in within the last month? *[multi-select b]*
6. When did you or will you graduate from college? *[date]*
7. What did you eat at your last meal? *[audio]*

We recorded transcripts from these survey calls, and interviewed the participants after the call.

We identified a number of design considerations and usability issues from our VoiceSim experiment. Below is a list of some of our important observations. These observations informed our design of ODK Voice.

- We tested two different methods for multi-select question types. In method 1, the user is presented with a list of options with associated touchtone numbers, and asked to press all numbers that apply. In method 2, the user is asked to press 1 for yes and 2 for now for each of the options. In both methods, the chosen options are repeated after the question is answered, and the user is asked for confirmation. Users were both more successful and reported greater satisfaction using method 2.
- We offered users the choice of answering the string entry question either interactively over the phone call or via an out-of-band text message sent from their phone. All users chose to answer the question interactively (although one user reported that she would have chosen to answer it via text message if the question had required a longer answer). Users' reasons for not answering via text message included that they didn't want to pay for it, and that they didn't know how to do voice and SMS simultaneously. Our interactive string entry approach was similar to text message input: the user pressed a touchtone key the appropriate number of times for the letter followed by the pound key. To

finish entry, the user pressed the pound key twice. As expected, this entry method was very inaccurate, and users reported dissatisfaction using it. Even though this technique might be appropriate for SMS, where the user receives immediate feedback, it is very hard to use in an IVR context. This inspired us to use a 1-digit-per-letter approach that required the system to infer the word from the entered digits, as is performed on more modern phones. (see Section 4.4 for more details).

- Users reported a desire to know 'where they were' in the survey, i.e. what question they were on and what the total number of questions was.
- The prototype had both immediate confirmation of answers and an option to go back and correct answers. Users were annoyed by the confirmation dialogues, but reported that they would be able to use them to correct their answers. On the other hand, users reported that they did not know how to go back and correct their answers, and probably wouldn't do so even if they made a mistake and were able to. As a result, ODK Voice emphasized immediate confirmation and retry instead of providing the ability to navigate a survey and correct earlier answers.
- As expected, it was very important that the number of options be kept to a minimum, and that their functionality be consistent throughout the survey. Users were told that they could press the star key to reach a main menu, but none of the users took advantage of this option and some said they were confused about what this main menu would provide. Users were also very confused when one of the questions used the pound key to continue, when all the other questions had used the 1 button. Therefore, we made ODK Voice as directed as possible; there were no universal options to navigate the survey or reach a main menu, although the users could press the star key at any time to repeat the current prompt.

3.2 Design Decisions

Our early design decisions for ODK Voice were based on both the results of our Wizard of Oz simulation and recommendations and results from the literature.

Perhaps the most important design decision - and certainly the most quantitatively studied in a developing-world context - is touchtone vs voice as an input modality. As discussed in Chapter 2, the majority of previous literature suggests that at least for trained developing-world users, touchtone input is more effective than voice input, although some recent studies conclude the opposite. The studies where voice input *was* effective, however, emphasized the importance of well-designed voice recognition systems trained on local users [31]. The goal of ODK Voice is to allow end-user programming of voice interfaces *without* requiring voice experts and extensive training. Furthermore, data collection - especially for input types such as numbers and dates - poses a far more difficult speech recognition challenge than the simple 2- and 3-word navigation grammars used in [31] and [24].

Feedback and input confirmation is one of the most important aspects of voice interfaces, and it is mentioned prominently in the literature on voice interfaces [2]. The simpliest approach to input confirmation is to repeat each response and ask the user whether it is correct (e.g. 'Press 1 if that is correct, or 2 to try again'). Alternatively, the system can confirm responses implicitly; in this approach, the system repeats the response, and moves forward without user confirmation. The system must then provide a way for users to go back and change previous (incorrect) answers. This could be accomplished, for example, by a navigation menu within the data collection interface. Finally, some types of input (e.g. boolean or recorded audio) may not need confirmation whatsoever.

ODK Voice asks for explicit confirmation for each survey question. There were

[2]More generally, effective error correction and the ability to 'undo' user actions is considered one of the most important elements of user interface design.

several reasons for this decision. Since we intend ODK Voice to be used for medical applications, data accuracy is very important. Explicit confirmation is most reliable, and in fact respondents from the VoiceSim experiment said that they would not try to go back and correct answers even if they realized they were incorrect. Explicit confirmation is also considered more appropriate for users who may have a difficult time using voice interfaces. Oberle writes,

> Since explicit confirmation doesnt place high demands on the callers ability to interact with the dialogue system, this kind of approach seems to be particularly suitable for callers with a limited ability to interact (e.g. elderly people, non-native speakers or children) or for those speech applications that will not be used very often and where callers will not be very familiar with comparable systems. [23]

Although some of the MIT VoiceSim participants found explicit confirmation to be inefficient, it is better targeted to a population with low literacy and limited interaction with voice systems. Finally, implicit confirmation (and navigation in general) is difficult because for some question types (e.g. numeric input) all the number keys are already being used, and developing-world mobile phone users often do not know what pound and star keys are.

Finally, ODK Voice is designed to allow for greater control over rendering from within an XForm, while providing default behavior that is suitable for 'average' deployments. This is accomplished using 'attributes' or 'hints' attached to a form or a specific question. A list of attributes is provided in Section 4.3.

Table 3.1 lists each of the XForms control types we chose to implement in ODK Voice, and a brief description of the approach to rendering this control type in a voice UI.

This chapter described a Wizard of Oz voice interface prototype we built and

Control Type	Rendering
<input type=int>	Enter integer over DTMF.
<select1>	"Select 1 for <item1>, select 2 for <item2>, ..." If there are more items than DTMF keys, we would need to split the items between multiple dialogs, e.g. "Press 9 for more options."; however, form designers should avoid large <select1>s.
<select>	Multiple yes-no dialogues, e.g. "Which symptoms do you have? Press 1 for yes, 2 for no. Fever (wait for response). Chills (wait for response)..." is more effective than a single dialogue, e.g. "Press the keys for all options that apply:..."
<input type=string>	Entering a string over multi-touch DTMF (i.e. 222 codes 'C') appears very difficult over IVR, because there is no feedback. Furthermore, it does not support non-Latin character sets. Instead, use single-touch DTMF (i.e. 733 codes 'RED') and use a dictionary to figure out what word was typed. The design should be robust to spelling mistakes, and should figure out over time which are the most likely words for a particular question. Alternatively, strings can be entered out-of-band over SMS.
<upload type=audio>	Ask the question, return the response as a wmv file. This is the only question type that uses an audio response from the user.
<input type=date>	Three separate dialogues: year, then month, then day. However, this is not well suited to peoples memory (e.g. it happened last Friday). We do not know of an audio analogue of the graphical 'date chooser' widget.
<bind jr:preload=xxx>	JavaRosa uses the preload tag to automatically fill in (or set defaults for) fields in a form based on system-level properties. We will require preloads for caller ID, session ID, and start/end timestamps.

Table 3.1: Initial design decisions for rendering each of the XForms control types over voice.

used to test early design decisions. This prototype helped select among alternatives for multi-select and string entry widgets, pare down necessary UI actions, and test different response confirmation strategies. In the initial design of ODK Voice, we decided on primarily touchtone input, explicit confirmation, and a system of voice rendering attributes which allowed for rendering customization.

Chapter 4

ODK Voice System

ODK Voice is a platform for rendering XForms through automated telephone calls. XForms is a W3C form specification standard; a sample XForm is provided in Appendix A. This chapter contains a description of ODK Voice features, a breakdown of the system architecture and infrastructure, and provides additional details about voice-specific rendering attributes and about string entry over voice.

4.1 Features

4.1.1 Control Types

ODK Voice provides support for most of the XForm control (question) types provided by JavaRosa, including multiple choice, numeric, date, string, and audio recording questions (see Table 3.1). There is also a read-only (no input) control type to provide information during surveys, especially for branching decision protocol applications.

4.1.2 Other XForm Support

ODK Voice is powered by the JavaRosa XForms engine, and inherits much of the XForms feature support present in JavaRosa[14]. ODK Voice supports the following

XForms features:

Constraints Constraints on input data can be specified in an XForm and enforced by ODK Voice. A wide variety of constraints can be encoded in XForms, such as:

- Marking a question as 'required'; the survey cannot continue until this question is answered.
- Enforcing valid ranges for numeric input.
- Regular expression matching of input.
- Enforcing complex relationships between the responses to different questions (e.g. answering 'male' in one question and 'pregnant' in another is disallowed).

Branching A form can include branching logic based on answers to questions (or more complex expressions). This allows different questions to be asked based on previous answers. Forms can make extensive use of XForms branching logic to provide entire decision support protocols, such as asking questions about patient symptoms to arrive at a diagnosis.

Repeats In some cases, a group of questions needs to be asked multiple times; for example, if a question asks 'How many children do you have?' and the response is '3', a repeat can be used to ask a group of questions 3 times (once about each child). Repeats provide this functionality.

See [2] for sample XForms utilizing these features.

4.1.3 Form and Prompt Management

ODK Voice provides a web interface for uploading forms and recording form prompts. The form management interface is basic, allows for multiple forms, and provides JavaRosa validation of forms on upload.

The prompt management interface allows administrators to easily record form prompts without experience or expensive software or hardware. ODK Voice scans uploaded forms and automatically identifies the set of audio prompts that will be needed to render them. The administrator can then record these prompts using either a mobile phone or recording software/hardware on their computer.

To record prompts over the phone, the administrator calls the ODK Voice phone number while viewing the prompt recording web page. By entering a touchtone code, the administrator can enter the prompt recording mode on the phone. When it is time for the administrator to record a prompt, that prompt appears in a box on the web page (see Figure 4-1), and the administrator speaks that prompt over the phone.

The administrator may also upload recorded audio in WAV format instead of recording over the phone. This is helpful for higher-quality audio as well as inserting sound effects or 'earcons'.

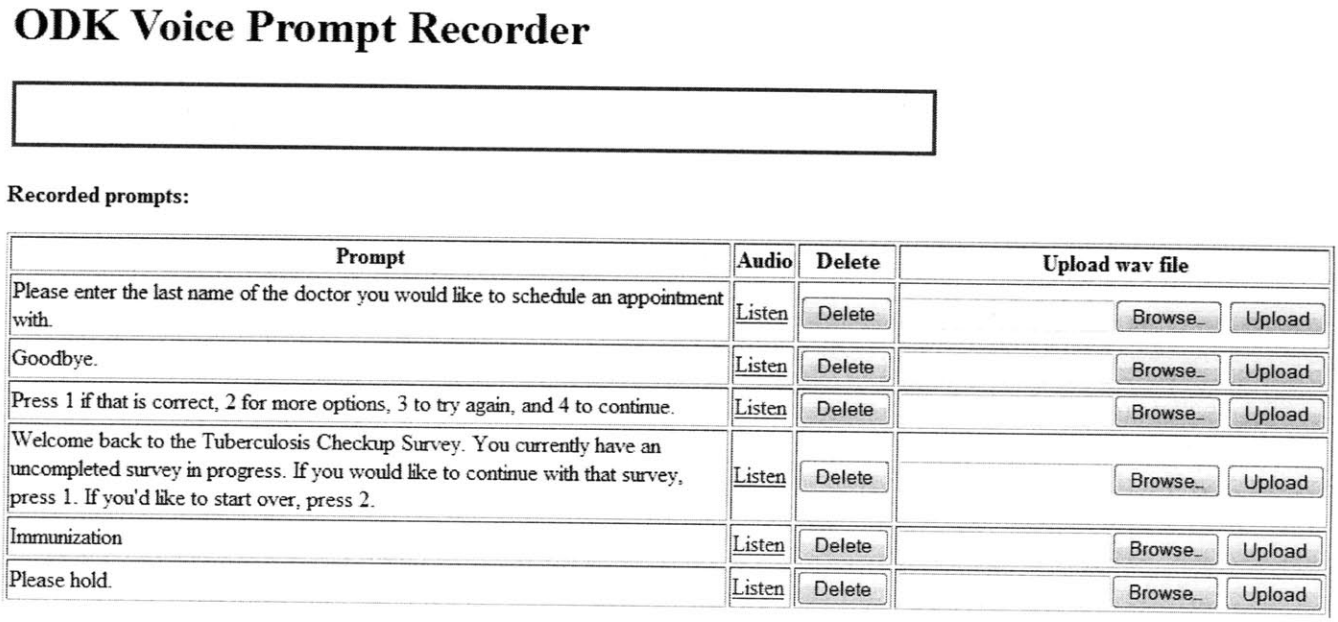

Figure 4-1: A screenshot of the ODK Voice web interface for prompt recording. Prompts can be recorded over the phone, or by uploading WAV files. When the administrator calls in, the prompt text to be recorded appears in the red box.

4.1.4 Multi-Lingual Support

In many regions of the developing world, several distinct regional languages or dialects may be spoken in addition to national languages. Sherwani et al. reported that less than half of users of the Tamil Market project spoke Urdu (the national language); others spoke dialects of regional languages [31]. Therefore, ODK Voice allows form designers to write and record survey prompts in multiple languages (following the XForms specification) and allows users to select from available languages from within the voice interface.

There are a couple cases where multi-lingual support is not possible because the range of possible prompts is too large to record. Specifically, playing back integer responses requires text-to-speech (one cannot record every integer), and playing back string responses for large dictionaries (more than a few thousand words) is generally infeasible without text-to-speech. Other datatypes are fully multilingual.

4.1.5 Survey Resume

ODK Voice allows users to hang up in the middle of a survey and call back and resume it later. This is accomplished via a session manager with sessions indexed by caller ID. Users have the option of either continuing the survey where they left off or starting the survey from the beginning.

4.1.6 Outgoing Call Management

ODK Voice supports both inbound and outbound calling. A web interface is provided to manage bulk outbound calling and to provide a basic scheduling system so that (a) calls can be scheduled in advance and delivered during appropriate hours and when the system has free phone lines, and (b) failed calls can be automatically retried under certain specified conditions.

The web interface for managing outbound calls is shown in Figure 4-2. The administrator copies a list of phone numbers into the web form, and can choose to either deliver them immediately, or schedule them for a time period in the future. If a call fails[1], it can be retried after a specified time interval. A call queue provides status and delivery information about scheduled calls, which allows administrators to differentiate between pending calls, in-progress calls, completed surveys, partially completed surveys, calls sent to voicemail, and failed calls.

For the Project WET deployment (see Chapter 5), calls were scheduled to be delivered between 10AM and 6PM local time, with retries every two hours. This proved most successful, because participants were usually able to pick up at some time during the work day, and they were not disrupted at late hours.

4.1.7 Integration with XForms Design, Aggregation & Analysis Platforms

The Open Data Kit project is envisioned as a collection of modular tools for XForm design, rendering, aggregation and analysis that can be easily integrated and interchanged. ODK Voice is an XForm rendering tool, and as such must integrate with XForm design tools on one end and aggregation and analysis tools on the other.

ODK Voice stores all survey data locally, but is also designed to send it via HTTP to an aggregation server such as ODK Aggregate [2], which may be physically colocated with ODK Voice or remote. ODK Voice implements a robust upload scheduler with exponential backoff retransmission to ensure that all submissions are successfully uploaded to the aggregation server.

Using ODK Aggregate as a data backend, results can be viewed on the web and exported to CSV, KML, or integrated with Google Spreadsheets. ODK Aggregate

[1] An uncompleted survey does *not* trigger a retry, as this would not be ethical towards the participants. Only unanswered calls are retried.

ODK Voice Outbound Call Control

Enter the phone numbers, one per line:

Send now ☐
or Schedule calls between _____ and _____ hours from now.
If calls fail, retry every _____ hours.
Schedule

Call Queue

Phone Number	Status	Retry	Delete	Time Added	Delivery Information
+256775[redacted]	COMPLETE	Retry	Delete	5/24/10 3:58 PM EDT	COMPLETE after 2 attempts between 5/25/10 3:28 AM and 5/25/10 12:28 PM
+256774[redacted]	NOT_COMPLETED	Retry	Delete	5/24/10 3:58 PM EDT	NOT_COMPLETED after 1 attempts between 5/25/10 3:28 AM and 5/25/10 12:28 PM

Figure 4-2: A screenshot of the ODK Voice web interface for scheduling outbound calls. A list of phone numbers is scheduled either immediately or in the future, and call status are displayed in a call queue.

can also be used as a common backend to receive results from the same XForms rendered on both ODK Voice and on mobile phone or PDA software such as ODK Collect or JavaRosa. Finally, XForms can also be submitted to other backends, such as an XForms OpenMRS module currently in development.

4.2 System Architecture

4.2.1 External Infrastructure

ODK Voice consists of a Java web server that processes Xforms using the JavaRosa core, and renders them as IVR dialogues using VoiceXML markup. The VoiceXML markup is interpreted by an external VoiceXML client[2], which can either be running on the same server as ODK Voice (ideal) or located remotely. The ODK Voice application also includes HTML servlets for administrative interface such as form, prompt, and outbound call management.

We chose to encode voice dialogues as VoiceXML instead of interfacing directly with a particular platform such as Asterisk because it is in line with the ODK/JavaRosa mission of supporting open standards, and because it provided flexibility for ODK Voice to run on a variety of hosted or standalone platforms[3]. However, we have found a number of limitations of VoiceXML that would be important to consider for those planning to use VoiceXML for UI research in the future. First, although VoiceXML is supported by a number of platforms, these platforms are almost all proprietary, and the open source offerings (such as VoiceGlue) provide very limited functionality. Although the proprietary providers offer free versions and licenses and a surprising level of free support, one is ultimately at the mercy of these providers for resolving

[2]Sometimes also called a VoiceXML 'interpreter' or 'browser'.

[3]Voxeo and TellMe both provide proprietary hosted and standalone VoiceXML clients. VoiceGlue is an open source VoiceXML plugin for Asterisk, and VXI* is a VoiceXML plugin for Asterisk from i6Net. In our work, we used the Voxeo platform, which is provided for free for up to 2 ports per instance. See `http://www.voxeo.com`.

problems and supporting one's feature needs. Second, we found that a dialogue-level language such as VoiceXML lacked the flexibility to accomodate lower-level changes and features that we found interesting from a research standpoint. For example, it was not possible to control exactly what the system considers silence vs. input, or keep track of user input while continuing to play a prompt, or respond adaptively to a user's pauses during an audio recording. VoiceXML follows a model similar to the web, in which the client 'browses' VoiceXML documents following links and sending requests to the server. The server can only respond to requests and never receives the actual audio stream to interpret arbitrarily.

The VoiceXML client must connect to the POTS/mobile network via a gateway. One inexpensive gateway option is a GSM modem, which can cost less than $100USD (more advanced GSM gateway hardware can also be used). The advantage of a GSM modem is that it can be very cost-efficient when used in-country, since mobile minutes are very cheap. The disadvantage is that the server must be set up in-country, where it may require maintenance and suffer from power and service outages.

Alternatively, the VoiceXML client can connect to a SIP gateway. There are many SIP providers that can provide inbound and/or outbound dialing to telephone networks worldwide at relatively low rates. The advantage of using a SIP gateway is that it doesn't require any dedicated hardware, and doesn't require setting up a physical server in-country. For example, we delivered surveys to Uganda for Project WET (see Chapter 5) from a server located in Boston. The disadvantage is that the (long distance) cost per minute is significantly higher than using a GSM modem for many countries.

ODK Voice sends completed survey data to ODK Aggregate via HTTP POST, as described in Section 4.1.7.

Figure 4-3 illustrates the ODK Voice infrastructure and interfaces described in this section

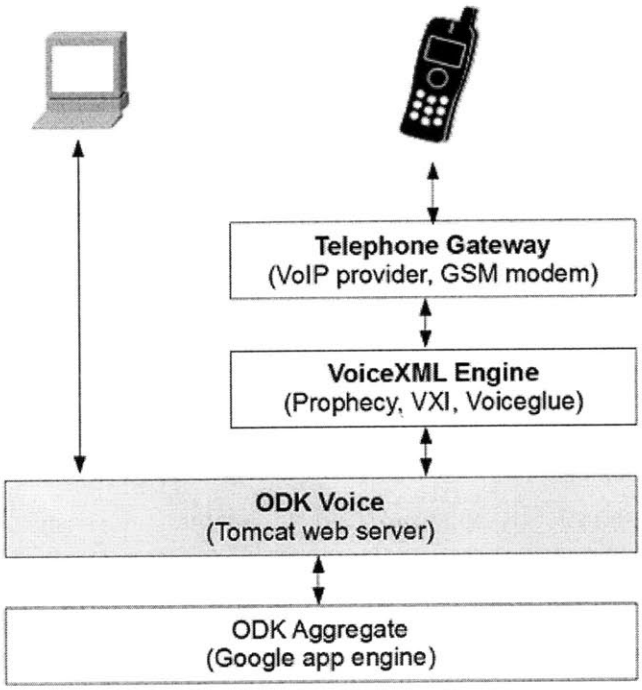

Figure 4-3: A diagram of the hardware/software infrastructure that ODK Voice depends on. ODK Voice uses VoiceXML to specify the audio dialogues that are played by a VoiceXML engine and transmitted via a telephone gateway to the cellular network. Collected data is sent to ODK Aggregate for viewing.

4.2.2 Internal Architecture

Figure 4-4 shows the dependency relationships between the ODK Voice classes.

`FormVxmlRenderer` is the top-level class for rendering VoiceXML dialogues. When a new IVR session is started, the server creates a new `FormHandler` object that contains a data model for the Xform, keeps track of the user's position in the form, and stores completed data from the session. The server stores the `FormHandler` in a `VoiceSessionManager`, and the session token is sent in each VoiceXML dialogue so that this session state can be maintained on the server. Most of the Xform parsing will be delegated to the JavaRosa core; translation of individual controls (i.e. questions) to VoiceXML will be handled with a subclass of `Widget` specific to the particular question type.

Each question is rendered as its own VoiceXML document; moving to a new question requires querying the VoiceXML server for the new question (along with submitting the data, if any, for the previous question). This allows the VoiceXML server to respond dynamically to submitted data; this is necessary in order to support features such as data constraints, branching, and repeats. Completed surveys are saved and forwarded to an XForms data aggregation backend, such as ODK Aggregate [2].

Each widget class keeps track of the prompts necessary for questions of that type, so ODK Voice can determine the prompts needed for recording by iterating through the survey and requesting a prompt list from each `Widget`.

The `OutboundCallScheduler` and `UploadScheduler` are daemon threads that handle sending outbound calls and uploading completed surveys, respectively.

Data persistence of forms, prompts, survey data, and outbound calls is accomplished with a mySQL database. Each VoiceXML HTTP request is also stored in the database so 'research' queries such as "How long did question X take on average" can be answered retrospectively.

Localization is accomplished using Java `ResourceBundle`s associated with XForms language identifiers.

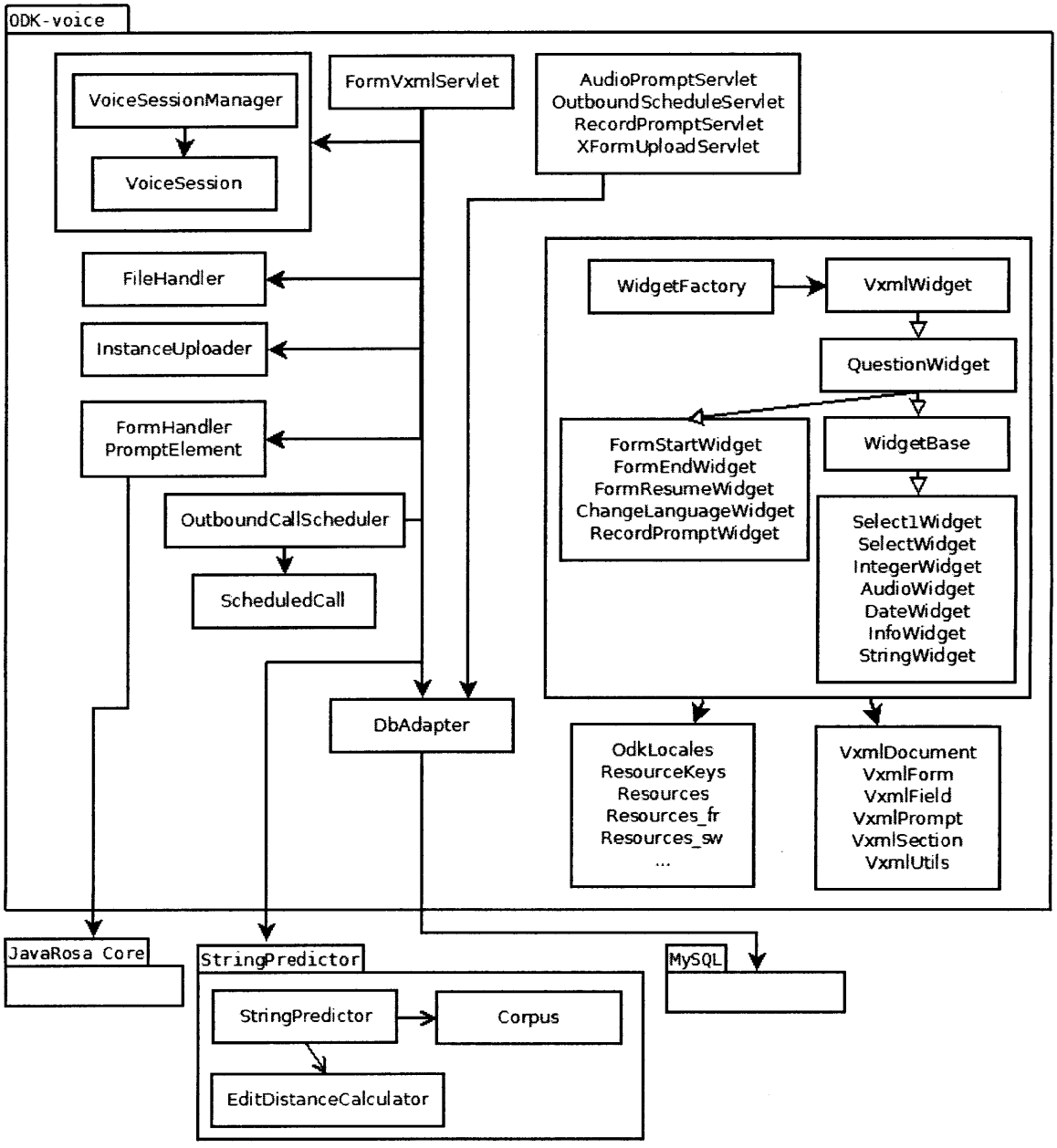

Figure 4-4: Module dependency diagram for the basic components of the ODK Voice Java web server.

4.3 Form and Question Attributes

Form designers who want greater control over ODK Voice's rendering can annotate their form with rendering attributes or 'hints' at either the question or form scope. Rendering hints are used to provide greater rendering flexibility while preserving cross-compatibility with standard XForms. Rendering hints are also a simple way for future developers to add new rendering functionality to ODK Voice while maintaining compatability, by adding functionality that is activated by particular rendering hints. This functionality can then be pushed back into the ODK Voice core without changing default behavior.

Table 4.3 contains a list of attributes currently implemented in ODK Voice.

Attribute	**Scope**	**Function**
digits	Q	For numeric question, play back the response as digits (e.g. two five) instead of a number (e.g. twenty-five).
skipInstructions	Q/F	Skip generic question instructions (custom instructions should be included in the question prompt).
skipQuestionCount	Q/F	Skip saying 'Question 1 of 3' at the beginning of a question.
repeatQuestionOption	Q/F	Remind the user that they can press star to repeat the current question.
skipConfirmation	Q/F	Skip the confirmation step for a question.
customIntroPrompts	F	Replace the generic form intro prompts with custom prompts.
resumeDisabled	F	Disable the ability to call back and resume a form (see 4.1.5).
maxTime	Q	For audio question, set the maximum record time.
stringCorpus	Q	For string question, specify the corpus (see 4.4).
forceQuiet	Q/F	Prompt the user to stop talking if they interrupt the instructions, and offer to call back if the connection remains noisy.

Table 4.1: List of XForm attributes that specify custom rendering options in ODK Voice. Q is question-scope, and F is form-scope.

Unfortunately, JavaRosa does not currently expose custom XML attributes, although this feature under discussion. Therefore, as a temporary solution, <hint> tags are used to store rendering attributes.

4.4 Adaptive String Widget

String input is particularly challenging to achieve in an audio interface such as ODK Voice. On the one hand, mobile phones accomplish string entry (e.g. for text messaging) using touchtone input, but this is heavily dependent on immediate feedback from the phone's graphical UI. On the other hand, existing voice UIs rarely require string input, and usually have a small custom vocabulary (e.g. cities) when they do.

String input is probably a poor choice for ODK Voice's low-literacy target population. Nevertheless, if only from a compatibility standpoint, it would be ideal for ODK Voice to be able to achieve string input, since existing JavaRosa XForms make use of string input, and there are certain types of data that are hard to enter except as strings. However, our initial usability tests (see Chapter 3) suggested that string entry is extremely difficult even for experienced users, and users did not want to enter strings through a separate text message.

In order to create a usable string widget, string entry was restricted to words (or sets of words) from a predefined corpus. This corpus can be generic (e.g. a dictionary) or specific to the question (e.g. a list of cities or patient names). This restriction had two advantages: first, the user can press one key per letter (e.g. 733 for 'RED') and the system can infer the word; second, the system can correct spelling errors by looking for a 'closest match' (this is especially important in low-literacy populations). Of course, even without spelling mistakes there is a possibility of collisions (since more than one word can have the same key code); we disambiguate between these choices by asking the user for confirmation for each of the most likely words in order of likelihood.

The likelihood of word W_i being the intended word, given an observed key sequence O is, by Bayes rule:

$$P(W_i|O) = \frac{P(O|W_i)P(W_i)}{P(O)} \tag{4.1}$$

$P(O|W_i)$, or the probability of observing a key sequence O for a given intended word W_i, is calculated using the 'edit distance' or 'Levenshtein distance' between the expected and observed key sequences. Edit distance is a generalization of Hamming distance that allows for insertions and deletions in addition to substitutions; the edit distance between s and t is the minimum number of insertions, deletions, or substitutions required to get from s to t.

Edit distance can be calculated using the Wagner-Fisher algorithm in $O(mn)$ time, where m and n are the lengths of the two strings s and t being compared [13]. Essentially, an $m \times n$ matrix is constructed, with each entry M_{ij} corresponding to the edit distance between $s_{1...i}$ and $s_{1...j}$. The first row and column of the matrix are initialized with $M_{1,i} = M_{i,1} = i$, and the rest of the matrix can be calculated with the following rule:

$$M_{i,j} = \begin{cases} M_{i-1,j-1} & \text{if } s_i = t_j \\ min(M_{i-1,j-1}, M_{i-1,j}, M_{i,j-1}) + 1 & \text{otherwise} \end{cases} \tag{4.2}$$

$P(O|W_i)$ is then computed as r^d, where d is the edit distance with the word and r is the estimated probability of a single error (assuming independent errors).

$P(W_i)$ is the prior probability for a word in the corpus. Since we don't know $P(W_i)$, a prior estimate $P'(W_i)$ is used. We start with uniform priors, $\forall i, P'(W_i) = \epsilon$, and adjust the priors in response to user behavior: a word's prior increases additively each time it is chosen by any user. As the number of responses becomes large (i.e. $P'(W_i) >> \epsilon$), these dynamic priors will approach the relative frequency of the word being chosen, i.e. $P'(W_i) \approx P(W_i)$. In effect, words that have been chosen more often in the past are suggested before less frequently chosen alternatives. An upshot of this approach is that as long as there are enough users, the form designer does not need a

custom corpus; the form designer can simply use a superset corpus (e.g. a dictionary) and the system will learn to only suggest words from the 'true' underlying corpus.

$P(O)$ is a normalizing factor and is independent of W_i, so it can be ignored.

In summary, when a key sequence is entered, ODK Voice calculates $P(W_i|O)$ for each word in the corpus, and returns a list of the most likely words. The user hears each of these words sequentially and chooses the correct one. Finally, the chosen word's prior is increased based on the user's choice. The runtime of this algorithm of $O(n^2d)$, where n is word length and d is dictionary size.

A graphical interface for the string entry functionality was built to evaluate its success with users; this interface is shown in Figure 4-5. The functionality was then incorporated in the audio interface.

String entry was evaluated as part of a scenario study in the United States (see Section 6), and had a 92% success rate when entering names from a 25,000 word English dictionary.

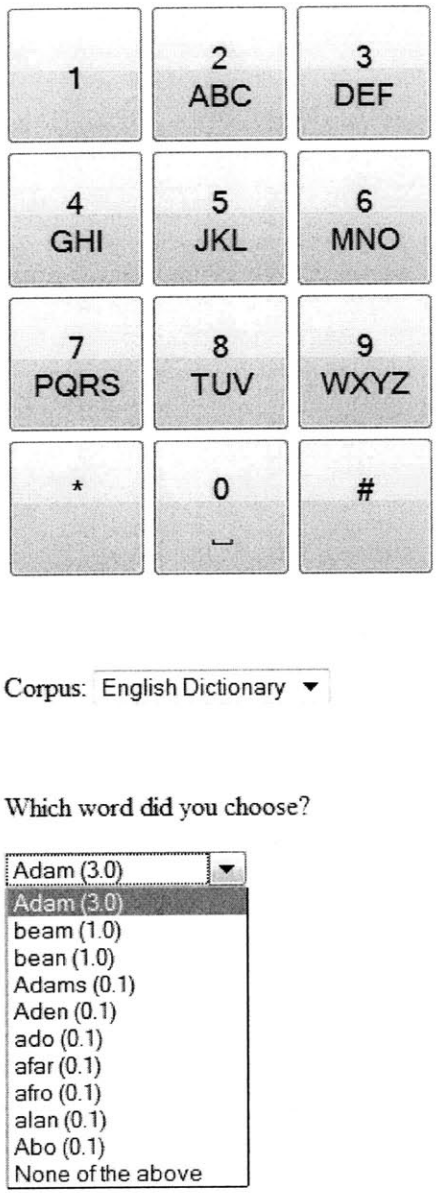

Figure 4-5: A test UI for the adaptive string widget.

Chapter 5

Project WET Deployment in Uganda

This chapter describes a pilot study of ODK Voice in rural northern Uganda undertaken in collaboration with the Project WET organization. Feedback was collected from rural Ugandan teachers about a training conducted by Project WET. The survey interface was created through an interative design process involving testing on Project WET employees, volunteers in Uganda, and teachers from the target population. Success rates and qualitative observations of different interface designs are presented, along with lessons learned from the iterative design process.

5.1 Background

Project WET is a non-profit organization whose mission is to "reach children, parents, educators, and communities of the world with water education." [3]. Project WET conducted a teacher training program throughout rural Northern Uganda in July and August 2009. Teachers were trained and given materials to teach students proper sanitation and personal hygiene practices.

The Project WET organizers were interested in obtaining feedback from par-

Figure 5-1: A Project WET teacher training in Uganda.

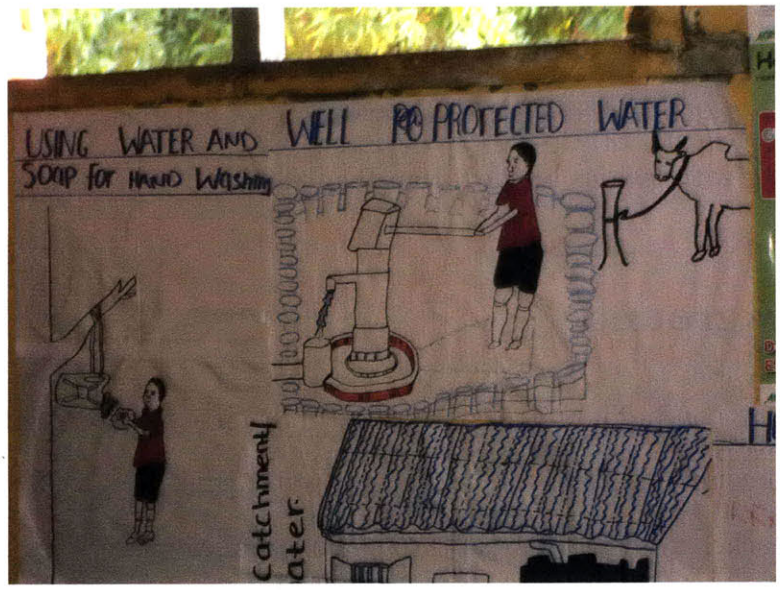

Figure 5-2: Project WET materials displayed in a school.

ticipating teachers about if and how they had used the Project WET materials in their communities. The teachers were located throughout rural Uganda, but approximately 250 of the 524 teachers provided mobile phone numbers at which they could be reached. Project WET originally planned to collect feedback with an SMS campaign. However, teachers were not willing to pay for SMS usage to provide feedback. Calling teachers with an automated voice survey circumvented this problem because mobile phone users are not charged for received calls in Uganda (and most other countries). Furthermore, a voice survey is capable of collecting more detailed information than could be sent in a 160-character SMS message.

Text messages were sent to teachers 24 hours in advance advising them that they would receive a survey call. The text of the message was:

> Hello! This is Project WET in the USA. Please expect a recorded survey call on Saturday. Help us by answering the questions. Thank you!

All survey calls were recorded and the audio transcripts were reviewed to improve the survey design.

Calls were scheduled by ODK Voice to be sent between the hours of 10:30 AM and 7:30 PM local time. If calls failed or were not answered, they were rescheduled up to 4 additional times at 2 hour intervals. Calls that were answered but not completed were not retried.

The study was conducted as an iterative design and evaluation process. Early versions of the interface were tested with members of the ODK and Project WET teams. Next, volunteers in rural Uganda tested the survey and spoke with us in interviews. Finally, several iterations were performed with the actual survey cohort. At each iteration, transcriptions of survey calls were observed in order to inform the next iteration of the interface. Task success rates were not calculated for early iterations because they were so unsuccessful that we chose not to test on a large sample size. In later iterations, calls were qualitatively characterized (see Table 5.1)

and task success rates calculated.

5.2 Touchtone Interface

The first version of the survey contained audio, boolean (single select), and numeric questions. The first question recorded the teacher's name and school. The second question asked whether the teacher had used the materials. If the teacher selected yes, the survey asked what results the teacher had seen (audio) and how many children had been reached (integer). If the teacher selected no, the survey asked why the materials had not been used (audio) and whether the teacher planned to use them in the future (boolean). Each question required explicit confirmation to continue, and the participant could press the star key at any time to repeat the current question.

One thing that became clear from the initial testing was the importance of the text message 'warning'. Each of the Ugandans interviewed cited the importance of the text message to 'prepare' them for the call. Participants who were sent a call without receiving a text message warning were confused and would hang up after a few seconds of failed attempts to start a conversation with the recorded voice.

Although having an IVR system call participants - rather than having participants initiate the call - is financially advantageous, we found that it introduced additional usability problems, which were only partially offset by the use of text message warnings. First, participants were often in an environment not conducive to a successful IVR interaction. These environmental factors included loud background noise, external distractions such as conversations with third parties, and intermittent call quality. As a result, participants often missed parts of the instructions and needed them to be repeated.

In most cases, participants also did not immediately realize the nature of the IVR calls; we found that no matter how we began the survey dialogue, participants

repeatedly said "Hello, hello? Who is this?", trying to establish a conversation, and thus missing the instructions. To overcome this problem, a 2-3 second 'chime' sound was placed at the beginning of the survey, followed by the instructions "This is a recorded survey call from Project WET. You are not talking to a person". Hearing the chime sound followed caused most participants to stop talking and listen for long enough that they heard the initial instructions.

The most serious usability problems with the initial Project WET survey involved understanding how and when to use the touchtone keys. In our initial interviews with participants in Uganda, we received feedback such as "It was very good, but the keys were very hard. It would be better if you could get rid of the keys", and "Pressing the buttons did not work for me." Many participants did not press any keys or did so only with significant prompting, and most participants who did press keys made a number of mistakes throughout the interaction. This observation is in line with Indrani et al., who found that subjects responded well to an ASR-based voice UI prototype but "required significant prompting and encouragement [from the experimenter] to press any key" in a graphical mobile interface [21]. Combining touchtone and audio input made matters even worse: once participants learned that they were supposed to enter information on the keypad, they often did not say anything during audio input questions. We speculate that difficulties with hearing and/or comprehending the touchtone instructions, the added difficulty of moving the phone from one's ear and finding a button to press (possibly missing further instructions), unfamiliarity with using the keypad during a phone call, and confusing the system with a live speaker, all may have contributed to the failure of a touchtone interface.

Several features were added to improve use of the touchtone keys. The instructions were elaborated to say "Some questions will have you press the number buttons on the phone. Other questions will have you say your answers.", and each prompt explicitly said either "Please press the X button on your phone" or "Please SAY your

answer after the beep." Prompts were also repeated again when users did not respond appropriately. Finally, the number of touchtone inputs was successively reduced in each iteration.

Despite these improvements, only 1 of the 20 participants receiving a touchtone survey completed it successfully, with another 3 achieving success on some of the questions [1]. 55% of the participants receiving a touchtone survey did not even succeed in pressing the 1 button to begin the survey, even when they were told to "Please press the 1 button on your phone to begin the survey." Instead, they said "Yes" or "1" or "Yes, I am ready" or simply hung up after hearing the instructions. In the cases where calls were at least carried out to completion (successfully or unsuccessfully), they usually took 6-8 minutes for 4 questions because participants had to repeat questions multiple times until they could answer them correctly. This may have been a useful learning experience for participants, but was almost certainly also a frustrating one. Finally, considering the low success rate, one must question the validity of the touchtone responses, even though these responses were confirmed by users in a follow-up dialogue.

These results suggest that without at least some initial training, a touchtone interface is infeasible for this target population.

5.3 Voice-Only Interface

Based on the results of the touchtone survey evaluation, the survey was completely redesigned to require no touchtone input. The voice-only survey had three questions with recorded audio responses. The initial instructions were:

> [Intro Music] This is a recorded call from Project WET. During this call you will not be talking to a person. You are receiving this call because you were part of a Project WET training in July and August 2009. We

[1] Call classification was based on listening to recorded transcripts of the calls

want to learn how you have used the Project WET materials since this training. The call will take about 5 minutes. Although you are not talking to a person, this survey will record you SAYING your answers. When it is your turn to say your answer, you will hear a beep that sounds like this: [beep] Wait until you hear this sound and then slowly and clearly say your answer. When you stop talking, the survey will continue. Let's begin.

The survey question prompts were:

1. After you hear the beep, please say your name and the name of the school where you work. When you stop talking, the survey will continue.
2. In a moment, you will hear another beep. After the beep, please say 'yes' if you have used the Project WET materials since the training, and 'no' if you have not used the materials. When you stop talking, the survey will continue.
3. In a moment, you will hear another beep. After this beep, please explain how you have used the Project WET materials, or why you haven't used them.

After each question, a beep was played, and the user response was recorded. The recording period ended after a certain period of silence. If the user did not say anything during the recording period, the question was played again; otherwise, the next question was asked. The timeout period had to be carefully tuned: too short a timeout caused the system to stop recording while the user was still responding; too long a timeout caused users to get confused and start talking again. A 3 second timeout was found to be optimal. Absolute timeouts for each question were also included to prevent user rambling or background noise to cause the recording to continue indefinitely.

Of 70 participants who received this version of the survey, 13 completed the survey successfully, 8 achieved partial success (some questions answered successfully), 19 could not complete the survey successfully, 22 hung up in the initial instruction period [2], 6 could not complete the survey because of environmental factors[3], and 2 calls were answered by the wrong person[4]. The overall complete and partial success rates are

[2]It is not clear why the users hung up in these cases, but many were likely unavailable to take the call, since calls were delivered during working hours.

[3]This category consisted of cases in which the participant said 'Please call me back another time' or in which there was very loud noise which made it impossible to hear.

[4]An additional 17 (excluded) participants did not pick up the phone when called, or had incorrect phone numbers.

19% and 11% (30% total). If we exclude the calls that failed due to factors external to the interface (hang-ups, environmental factors, wrong person), the complete and partial success rates are 33% and 20% (52% total). This success rate is several times higher than that of the touchtone interface. Figure 5-3 shows a breakdown of the call outcomes. Average time to complete the survey was 2 minutes, 55 seconds.

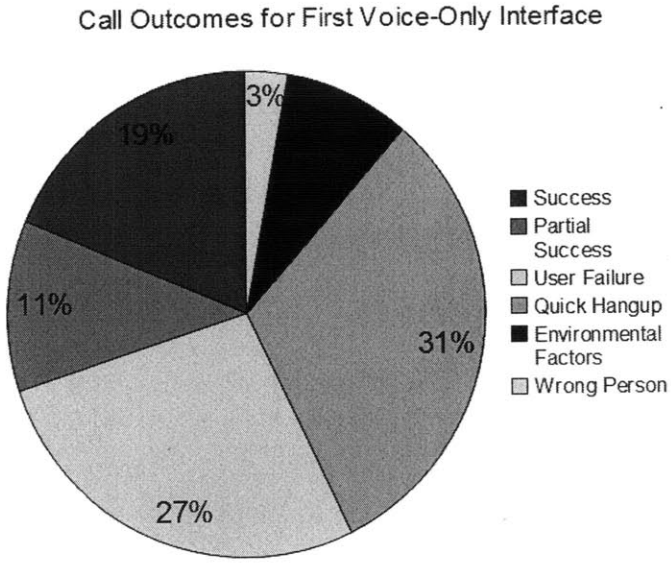

Figure 5-3: Pie chart showing call outcomes for the first voice-only Project WET interface.

One common characteristic of many of the failed calls was that participants did not stay quiet and hear the instructions, either because (a) they did not realize the prompt was automated and were trying to establish a conversation, (b) they were engaged in a conversation with a third party, or (c) there was too much background noise or poor connection quality. The interface was modified to 'force' the user to be quiet during the instructions. During the instructions, if talking above a certain threshold was detected, a prompt would be played asking the user to be quiet and listen to the instructions, which would then be repeated. If talking (or loud background noise) continued, the user would be informed that it was too loud and that he/she would be called back later.

Based on a small set of test calls, this 'force quiet' feature actually *reduced* the usability of the interface. We later realized from directly observing Ugandans using mobile phones that they often say "Hello?" or "Okay" habitually during calls to check the connection; asking users to stop talking did not always stop them from making these remarks, and users were 'scolded' for making these affirmatory comments during the instructions. Even more problematic was the fact that once users were 'scolded' once or twice for talking out of turn, they were very hesitant to talk even when prompted; in most cases where the interface *did* force the user to be quiet, the user remained silent for the rest of the interaction.

In the final version of the survey, the initial instructions and question prompts were further shortened, simplified, and made more conversational. Instead of forcing users to understand our metaphor of 'talk after the beep', the dialogue was constructed to emulate natural turn-taking behavior. Additionally, we received feedback from participants that the American accent was difficult to understand, so this version was recorded by a native Ugandan speaking slowly and deliberately in a Ugandan accent.

> [Intro Music] This is a recorded call from Project WET. You are not talking to a real person. This call will record your answers to three questions about your Project WET training. After each question, you will hear this sound: [beep]. After this sound, say your answer. When you are finished, stop talking and wait for the next question.

The survey question prompts were:

1. Question 1: What is your name?
2. Question 2: How have you used the Project WET materials since the training?
3. Question 3: What results or changes in student behavior have you noticed after using the Project WET materials?

Of 49 participants who received this version of the survey, 31 completed the survey successfully, 9 achieved partial success, 4 hung up in the initial instruction period, 3 could not complete the survey because of environmental factors, and 2 calls were an-

swered by the wrong person[5]. None of the participants fell into the 'failure' category; excluding the categories above, every participant was able to answer at least two of the three survey questions. The overall complete and partial success rates are 63% and 18% (82% total). If we exclude the calls that failed due to factors external to the interface, the complete and partial success rates are 78% and 23% (100% total). This success rate is much higher than that of the previous voice interface. Figure 5-4 shows a breakdown of the call outcomes. Average time to complete the survey was 3 minutes, 5 seconds.

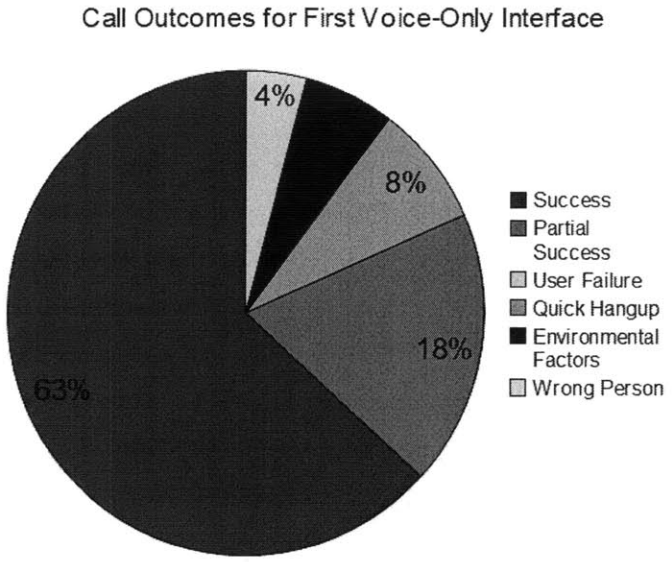

Figure 5-4: Pie chart showing call outcomes for the final voice-only Project WET interface.

The success of this interface suggests that leveraging conversational and turn-taking conventions of normal conversation are much more successful than detailed instructings in elicting desired user behavior. In the first version of the voice survey, questions were asked as statements (e.g. 'Please say your name and the name of the school where you work') and instructions were repeated during the question prompts. In the final version, we asked questions as questions and relied on turn-taking to

[5]An additional 23 (excluded) participants did not pick up the phone when called, or had incorrect phone numbers.

signal when the user was supposed to speak. This turned out to be dramatically more successful. The use of a native Ugandan speaker with a Ugandan accent and speaking style also appeared to improve understanding to a large degree.

Interestingly, the responses to this version of the survey were more concise, and more slowly and clearly enunciated than in previous versions. The literature reports that people tend to emulate the speaking style of their conversational partner in a voice dialogue[23]. Therefore, since the question prompts were more concise, the responses were also more concise. Since the prompts were recorded more slowly and in a more understandable accent, the responses were also spoken more slowly and clearly.

On a similar note, we found that in both versions of the interface the question 'timeout' (i.e. the amount of silence before the recording terminates) had a great influence on response length. When open-ended questions are asked, such as 'How did you use the materials?', participants often start with a concise answer, take a short pause, and continue to elaborate with longer and longer pauses until they are interrupted. Interestingly, participants did not usually seem frustrated when they were interrupted during their response (unless they failed to provide even the basic information requested) - they seemed to assume that the speaker was interrupting them once their answer was sufficiently detailed.

Call quality (both intermittent connection and background noise) continued to be a challenge, but unlike previous versions, participants were usually able to answer the questions even if they did not hear or understand the instructions completely, because the desired behavior was implicit in the conversational nature of the survey. If the speaker said 'What is your name?', the participant usually responded with their name and then waited for a response, even if they hadn't heard the instructions.

Unfortunately, one recurring problem was that in calls with very loud background noise, the response recording would not terminate automatically (until the hard cutoff

time was reached). We considered detecting background noise and instructing users *in those calls* that they could terminate the recording by pressing a touchtone key; however, we decided against this because participants tended to stay on the line even when the recording didn't timeout, which was a better outcome than what we experienced when we introduced touchtone input.

	Touchtone	**Voice 1**	**Voice Final**
Success	1	13	31
Partial Success	3	8	9
User Failure	9	19	0
Early Hangup	4	22	4
Call Quality	1	6	3
Wrong Person	0	2	2
Total	**20**	**70**	**49**
Text Failed		19	17
Call Failed	1	17	23

Table 5.1: Call outcomes for Project WET survey by interface.

Women had greater success using this interface than men. In the first interface, men were at least partially successive 45% of the time; Women were at least partially successful 85% of the time. The gender discrepancy in success rate in a Fisher's exact test ($p = 0.06$ one-tailed; $p = 0.09$ two-tailed) did not meet the standard significance criterion ($p < 0.05$). Results are shown in Figure 5-6 and Table 5.2. A similar observation of the second voice interface would not be meaningful, since there were no examples of user failure. The gender discrepencies observed support the findings of Indrani et al. that women had a higher probability of success than men [21]. Qualitatively, women generally listened more quietly to the instructions and answered more slowly and clearly, whereas men tended to talk during instructions (e.g. 'Hello? Hello?') and speak at the wrong time.

Every teacher surveyed responded that they *had* made use of the Project WET materials, and extremely positive results were reported from practically all of the respondents. Most participants reported that the materials had been 'rolled out'

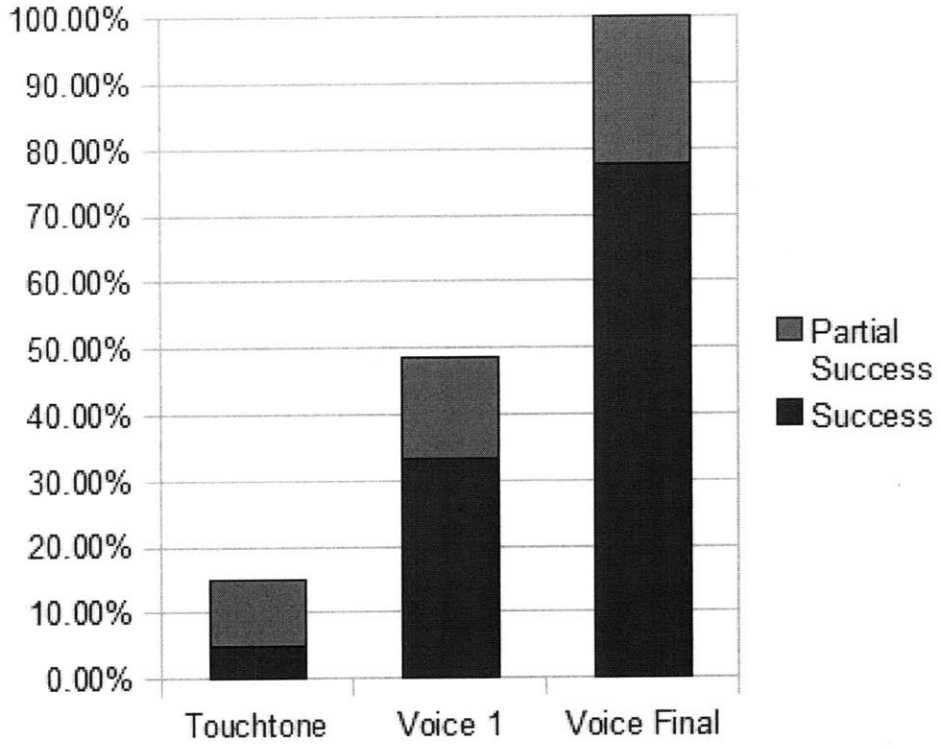

Figure 5-5: Call success rate for Project WET survey by interface version.

	Success / Partial Success	**User Failure**
Male	14	18
Female	5	1

Table 5.2: Call success rate for Project WET survey by gender.

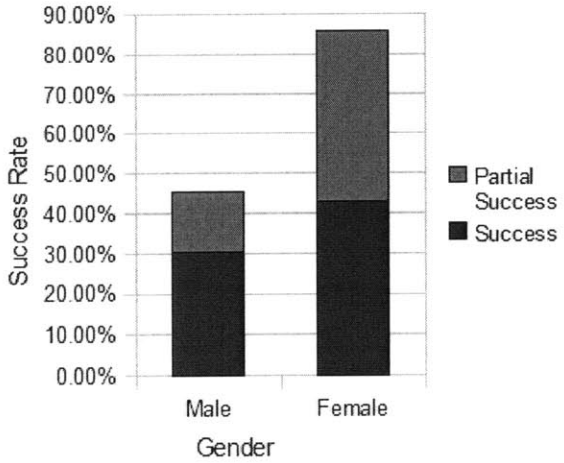

Figure 5-6: Call success rate for Project WET survey by gender.

to students and other teachers, and that students had begun to wash their hands properly, clean water containers, etc. We make no claims on the external validity of the survey methodology. Since Project WET was providing training, materials, and funding to the participating teachers, they almost certainly felt that it was in their interest to report positive results, particularly since the survey was not anonymous. In the most extreme cases, teachers may have interpreted the survey call as a test of *their* competence with the materials; for example, one teacher responded "I've realized many changes, since the Project WET has 7 topics, 7 activities: healthy habits, [etc.]", proceeding to list each of the Project WET topics. Much greater care would have to be taken in formulating the instructions and questions[6] if one hoped to draw meaningful quantitative conclusions from such a survey.

On the other hand, initial cross-validation of the survey data supports its accuracy. Approximately 50 of the surveyed schools were visited by Project WET employees; written feedback from teachers was elicited and direct observations of Project WET implementation and results was performed. While it would be difficult to rigorously cross-validate the responses - there is no real ground-truth data, and elicited responses may be different in different contexts; see Chapter 6 - cross-validation supports the general conclusions reached by the phone survey. All but one of the schools had been using the Project WET materials, and major changes were observed in student behavior such as handwashing, etc.

The results of this study do not *ipso facto* suggest that ASR-based interfaces would outperform touchtone interfaces; ASR was not used at any point in this work, and we make no claim about users' abilities to operate successfully under the constraints of limited vocabulary and recognition accuracy. This work *does* however highlight the importance of interfaces that leverage the conversational and turn-taking behaviors

[6]For example: making the survey anonymous; instructing participants that the quality of the materials was being evaluated, not their performance; rigorously cross-validating a subset of responses with live classroom observation.

already in use during natural conversation.

The iterative design process, including interviews with volunteers and observation of transcribed phone calls, dramatically improved the task success rate for the ODK Voice interface for this survey. The results of this process suggest a number of general principles for voice UI design for similar user demographics. See Chapter 7 for a summary of these user interface design principles.

Chapter 6

Measuring Accuracy with a Scenario-Based Study

A scenario-based study was conducted to measure the accuracy of data collection using ODK Voice. For this study, a form was built to implement a patient appointment scheduling protocol in ODK Voice. Brief written instructions told participants to read a scenario, call ODK Voice and make an appointment, then fill out an attached questionnaire. The scenario provided fictional information such as patient medical history and doctor, which were needed to make the appointment. A randomized, artificial scenario was used because ground truth data was needed to determine response accuracy. See Appendix B for instructions, scenario, and questionnaire given to participants. The voice survey questions were:

1. Please enter the phone number of the phone you are calling from. If you don't know the number, press 0. *[numeric]*
2. Please enter the last name of the doctor you would like to schedule an appointment with. *[string]*
3. For what day would you like to schedule the appointment? *[date]*
4. What is the reason for your appointment? *[single select]*

5. Do you suffer from any of the following chronic conditions that your doctor should be aware of? *[multi-select]*
6. Would you like to record a message for your doctor to provide additional information? *[branching yes/no]*
7. Please record a message for your doctor after the beep. *[audio]*

The original purpose of this experimental setup was to compare automated voice interface accuracy to a mobile software interface and a live operator in India, to extend the work of [25]. However, we have not yet found participants for this study. Therefore, the experiment described above was conducted in the United States designed to mimic the Project WET conditions (i.e. no participant training), to measure performance by US participants compared to that observed in the Project WET study. We hope to carry out the original accuracy evaluation in the future (see Section 8.1).

13 of 14 participants answered the survey to completion. These participants were all located in the United States and considered themselves fluent in English. Ages ranged from 18 to 76 years, and all participants had used a voice interface at least "several times" in the past. Education ranged from high school to PhD.

The error rate across all question types was 6.1%, and the average response time per question was 38 seconds. A breakdown of error rate and response time is shown in Figure 6-1. Too much weight should not be placed on the response times, since the majority of this time was spent listening to the question prompts; therefore, the response time is more dependent on the particular question and the prompt speed than the question type[1]. There were 2 participants over the age of 70 and 2 participants for whom English was not their first language. For these groups, the error rate was 27% and 17% respectively; for all other participants, the error rate was 0%.

[1]In this study the prompts were read very slowly and clearly. Contrary to our initial impressions, we found later that increasing prompt speed to a normal conversational pace did not decrease comprehension (and of course improved response time and satisfaction).

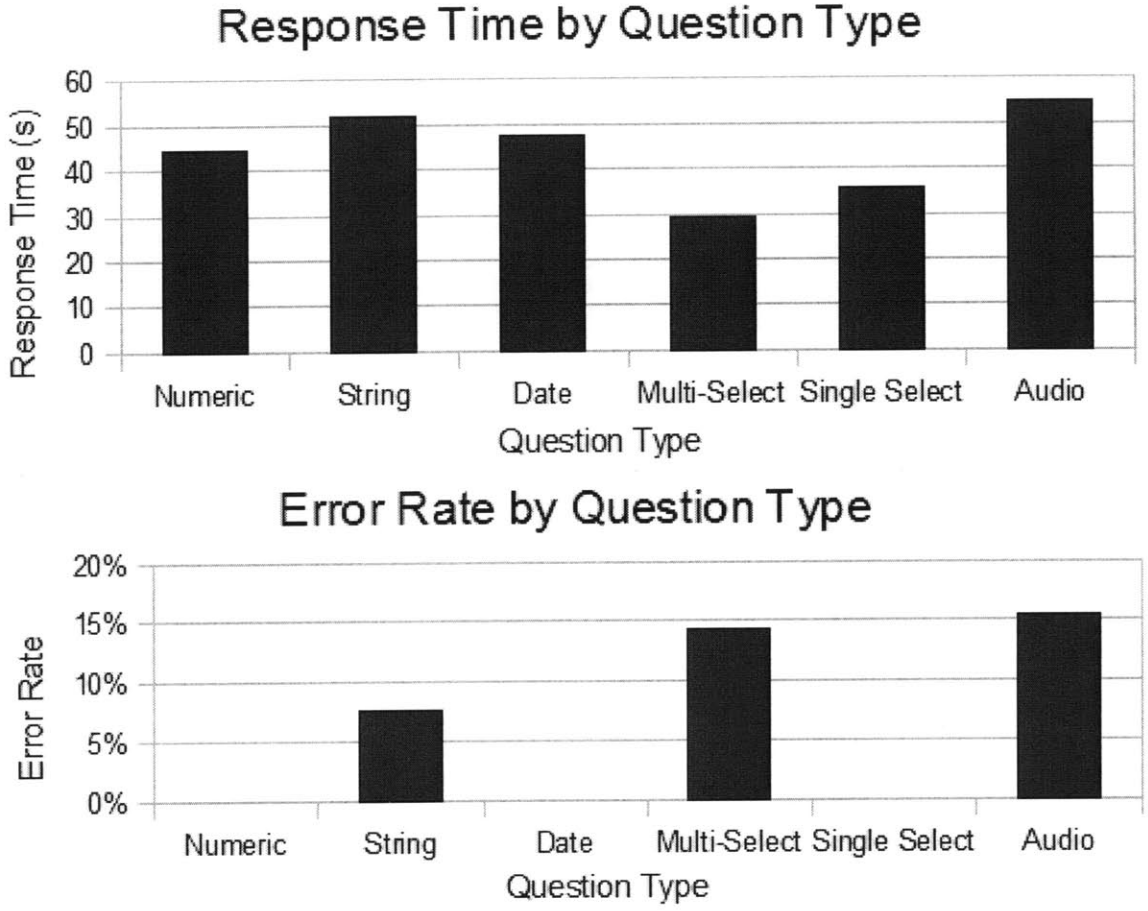

Figure 6-1: Measured averages of (a) response time, and (b) error rate for each question type in the scenario experiment.

Participants were asked to compare their satisfaction with ODK Voice to (a) a live operator, and (b) an online form, using Likert scales. As shown in Figure 6-2, most users preferred both a live operator or an online form to ODK Voice. This is not surprising, as voice interfaces are notoriously difficult to use, and ODK Voice is designed to improve accuracy over the satisfaction and efficiency of experienced users, especially in the configuration tested.

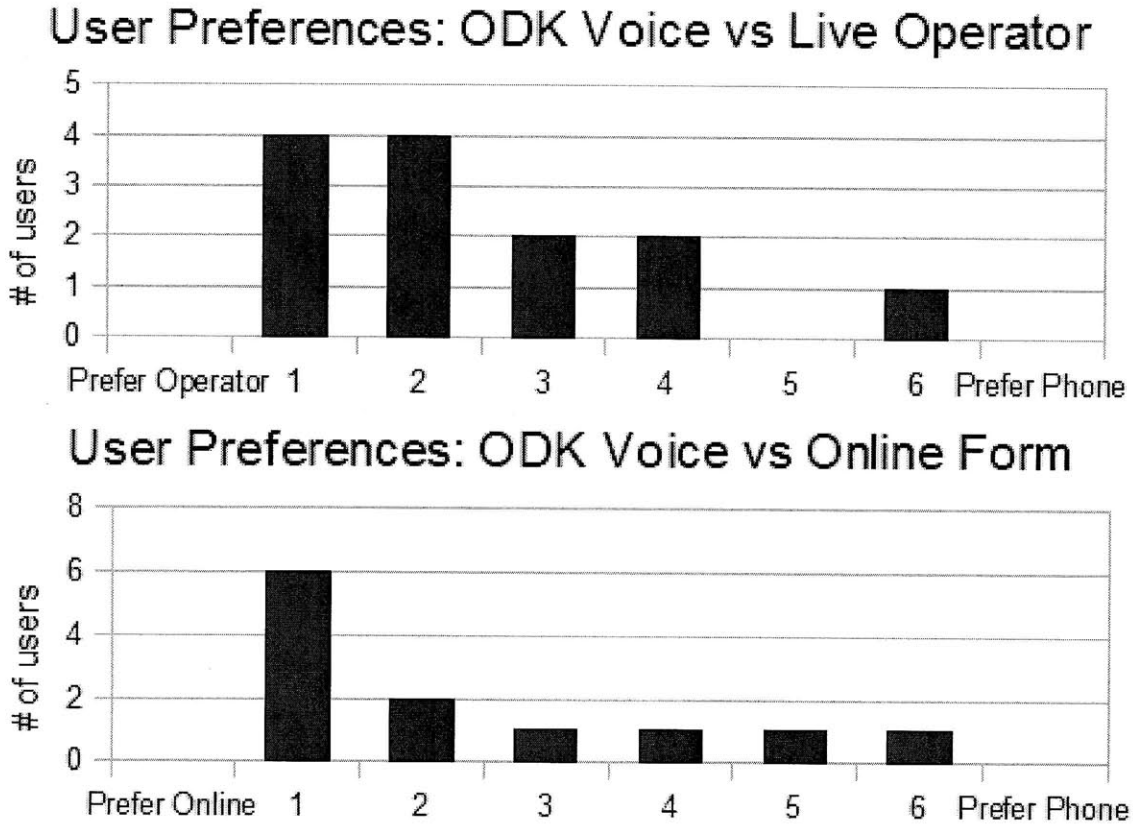

Figure 6-2: Reported user preferences on a Likert scale for the ODK Voice survey compared to (a) a live operator, and (b) an online form. Most users preferred both the live operator and the online form to the ODK Voice system.

The qualitative feedback received suggests that participants found the interface easy to use and trusted its accuracy, but were frustrated with the amount of time required to enter a small amount of data. Some participants appreciated the response confirmations, but several remarked that they were applied too broadly; not every question required explicit confirmation. One participant found the pace so slow

that she barged in, and then answered incorrectly because she had not heard all the instructions.

The results of this study suggest that the usability problems experienced by teachers in rural Uganda (see Chapter 5) are not replicated in literate, technically experienced users in the United States. This suggests that language and technical competence, literacy, and environmental factors (noise, call quality) are responsible for the usability problems faced in Uganda.

Chapter 7

Conclusions

This work describes two main contributions: ODK Voice, a platform for rendering data collection protocols (written in the XForms form specification language) over automated voice/touchtone dialogues; and a set of design guidelines for building voice interfaces for untrained users in the developing world based on evaluation and iteration in a real-world survey deployment in rural Uganda.

ODK Voice was successfully implemented and supports the majority of JavaRosa features, including a variety of question and data types, multi-language support, branching, and constraints. In addition to incoming call support, outgoing calls can be scheduled through a web interface. ODK Voice supports recorded prompts as well as text-to-speech, and provides a simple over-the-phone mechanism for recording prompts. ODK Voice is implemented as a Java Tomcat web server, and the voice dialogues are produced using the W3C VoiceXML specification, relying primarily on touchtone input. ODK Voice interoperates with the ecosystem of XForms-based data collection applications being developed by the OpenRosa consortium; in particular, it integrates with an XForms aggregation and analysis backend such as ODK Aggregate.

A 4-question survey including audio, multi-select, and numeric questions was developed in collaboration with the Project WET organization. This survey was de-

signed to collect feedback from teachers in rural Uganda about a recent training. The initial survey design had a very low success rate among the target population, but after several iterations of testing and redesign, 82% of participants completed the survey successfully and 100% completed at least 2 of the 3 questions (not including calls that failed for other reasons; see Chapter 5). Based on the results of the scenario experiment described in Chapter 6, even a significantly more complex survey could be completed with high accuracy by participants in the United States with previous IVR experience; this suggests that the usability problems encountered in the Project WET survey were a result of the characteristics of the target population. Based on this work, we provide below a summary list of design recommendations for voice interfaces targeted at similar populations.

Design Recommendations

- If users are being called by an IVR system, they must be warned in advance that they are going to receive an automated call. Text messages sent 24 hours in advance are an effective way to provide advance notice.
- Avoid use of the touchtone keys whenever possible; untrained users experience difficulty pressing keys during a voice dialogue. Particularly avoid switching between touchtone and voice input in a single interface, as users have difficulty distinguishing when they should be providing touchtone versus voice input.
- If users are not aware that the call is automated, they must be forced to stop talking at the beginning of the interaction and informed that the call is automated. Beginning the call with a sound effect such as a chime, followed by 'This is a recorded call. You are not talking to a real person' is a successful strategy for accomplishing this goal.
- Users are more easily able to understand prompts recorded by a speaker in their native accent. Similarly, prompts should be designed with members of the target population to ensure that the vocabulary is suitable to the target population.
- Instead of depending on detailed instructions, interfaces should leverage the implicit conventions and turn-taking behavior of interpersonal conversation. For example, the prompt 'Please say your name after the beep. [beep]' would not be as successful as 'What is your name? [beep]'. In addition to being more natural, this strategy allows users to succeed even if they cannot hear or understand the instructions.

- Responses tend to mimic question prompts in form, length, speed, and clarity. Question prompts should therefore imitate the type of response desired.

Chapter 8

Future Work

There are a number of opportunities for continuation of the work described. First, there are a number of ways that ODK Voice could be extended that might improve learnability or data quality. Second, there is much more work to be done to characterize the types of human-computer interaction that are successful when using automated voice interfaces for data collection on populations with low income, low language and literacy skills, and a lack of technical experience.

8.1 Further HCI Research on Voice Data Collection Systems

As mentioned in Section 2.3, Patnaik et al. found that live operator data collection over voice outperformed graphical and SMS interfaces by an order of magnitude [25]. However, it is unclear from this work whether the improvements in data quality result from the voice modality or from the presence of a live operator. In order to answer this question, the accuracy of automated voice interfaces in these environments must be evaluated. The experimental procedure for such an evaluation was performed with participants from the United States (see Chapter 6), but we must carry out such an

experiment on cohorts similar to [25][1], using one or more of mobile software, SMS, and live operator interfaces as controls, to determine how automated voice interfaces compare to these other data collection approaches.

Another interesting research question is how much 'bang' you can get for a limited amount of training. Each of the data collection projects described in Chapter 2 have provided participants with training, usually consisting of hours to days of classroom time in which the technology is demonstrated and practiced by the participants. This training makes sense in the context of some applications, such as health worker use. The Project WET deployment described in Chapter 5 provides a data point at the opposite extreme: no training whatsoever (besides a text message warning) and unpredictable call quality, noise and distractions since the participants are *receiving* the survey calls. Between these two extremes there is the possibility of limited training. For example, a CHW could visit a patient, teach them how to use a voice interface in 5-10 minutes, and have them call to report problems or monitor medication adherence. It is not currently known precisely how user performance varies with training time. Acceptable performance after 5-10 minutes of training would enable a number of interesting applications in patient self-reporting/self-help and mobile banking.

A related technical question is whether a voice interface can actually train a user as it is being used. For example, suppose a voice interface only recognizes 'yes' and 'no' speech input. The interface can start by saying "Say yes to continue" [2]. Likely incorrect responses include (a) the user saying nothing, (b) the user saying the wrong thing, (c) the user rambling. In each of these scenarios, the interface can provide help and give the user another opportunity to try to enter "yes". This process could mimic the interactive training that would occur in a live training session. It is not clear whether this approach could succeed, since users are often so confused by the

[1]Such an experiment could also be performed using a cohort of *untrained* patients instead of trained health workers.

[2]The interface should not ask an actual question at the beginning, because the user is likely to answer incorrectly.

format of automated dialogue that they don't listen to or understand what they are being told over the phone.

A final possibility for further research is an operator-assisted interface. In this approach, an automated voice interface is supplemented by a live operator on the line when a user first uses the interface, or when the user encounters problems using the interface. Operator assistance is standard in most IVR systems in the developed world, usually in response to a user request or repeated user failure with the interface. However, there are other ways that operator assistance could be used to actually *train* users on the interface. In one setup, the user could be initially connected to an operator, who would prepare the user and then transfer to the automated system. Or the operator could be present 'on the line' as the user uses the interface for the first time, providing help over the phone as necessary; this is again a voice analogue of what would occur in live training sessions, without being physically colocated with the users.

8.2 Extending ODK Voice Technology

ODK Voice is still in development, and there are opportunities to both improve the general usefulness of the software, and to experiment with novel features.

8.2.1 Incremental Extensions

There would be a great benefit to creating and hosting a central 'cloud' ODK Voice instance so that organizations can start using ODK Voice just by signing up for VoIP and creating a form, instead of having to set up their own server. Having a prepackaged virtual machine running ODK Voice out of the box would also reduce the technical barriers to setting up an ODK instance.

Scheduled SMS delivery should be implemented to accompany the scheduled out-

bound dialer. Based on the results of the Project WET study, it will be necessary for organizations to send SMS messages to participants before calling them, so this process should be integrated with the outbound call scheduler.

The adaptive string widget can be improved by moving beyond simple adaptive priors for words to consider common misspellings. Adaptive priors are useful for inferring the letters associated with keys, but is not ideal for identifying misspellings. To illustrate, suppose that the word *blue* is a very common answer to a question, but many users type it as *bloo* instead. First, it will take a large number of entries for the prior of *blue* to be large enough to outweigh words such as *blow* and *blot*. Second, this prior will also cause the entry *blur* to be recognized as *blue*. The underlying problem is that adaptive priors don't take into account that *certain misspellings are more common than others*. A better approach for misspellings is to assign weights to 'misspelling' pairs (O, A), where O is the observed key sequence and A is the actual word desired. If an observed key sequence O' was associated with any misspelling pairs (O, A), the likelihood of A would be increased.

One major problem we identified in the Project WET study was variable call quality. When recording audio answers, the speech was often not understandable either because the user was not talking clearly enough or because the connection was simply too poor. It would be useful for ODK Voice to automatically be able to recognize if one of these situations is occuring and either ask the user to slow down or call the user back later.

Finally, speech recognition (ASR) capabilities could be added to ODK Voice. The results described in Chapter 5 suggest that users are more comfortable with voice responses, although we are not claiming that the responses we received would be amenable to ASR. As mentioned previously, ASR is only available in certain languages, but an approach similar [31] or [24] could overcome this problem.

8.2.2 Adaptive Features

Adding adaptive features to ODK Voice has the potential to greatly improve usability. An "adaptive" feature or interface is one that changes based on statistical properties of previous interactions within a particular session, across multiple sessions by the same user, and/or across multiple users in the population. The string widget described in Section 4.4 is one example of an adaptive feature. Adaptive features are well-suited to this problem for two reasons: first, the user population is heterogeneous, with different languages and levels of technical expertise; second, the form designers are not UI experts, so ODK Voice could improve the interface based on collected data. Listed below are some adaptive features that could be implemented in ODK Voice:

- Adaptive algorithms could determine which questions require confirmation based on error rate, improving both speed and accuracy. This adaptation could be across questions (i.e. determining which questions are error-prone) or users (i.e. determining which users are error-prone).
- The ordering of choices in a dialogue can be presented in order of popularity based on the user and/or population history.
- An intelligent interface can respond to a user's linguistic background in several ways, such as automatically switching to the user's language or dialect, or modifying the system prompts to match the users 'speed of speech, dialect, and sociolect' [23]. It is also possible to respond to linguistic emotional cues, such as escalation/de-escalation strategies when anger or frustration are detected [23].
- Chen et al. have used adaptive algorithms to identify multidimensional outliers in collected data and re-ask relevant questions [6], and have seen improvements in data accuracy. For example, after collecting hundreds of surveys, the application may notice that users rarely enter 'male' and 'pregnant', and re-ask this question if this combination of answers is given.

Project Information

ODK Voice is being developed as part of the Open Data Kit project. Information on the Open Data Kit project can be found at `http://code.google.com/p/opendatakit`. Source code and binaries developed in connection with this work are available under the Apache license and can be downloaded from the same location.

Appendix A

Sample XForm

The document shown below is a sample XForm that can be used with ODK Voice. This XForm is similar to the one used for the scenario experiment described in Chapter 6.

```
<?xml version="1.0"?>
<h:html xmlns="http://www.w3.org/2002/xforms"
        xmlns:h="http://www.w3.org/1999/xhtml"
        xmlns:ev="http://www.w3.org/2001/xml-events"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:jr="http://openrosa.org/javarosa">
  <h:head>
    <h:title>Automated Patient Check-In</h:title>
    <model>
      <instance xmlns="accuracy-eval-2">
        <health>
          <phonenumber/>
          <userphone/>
          <doctor/>
          <appointment/>
          <conditions/>
          <reason/>
          <details_option/>
          <details/>
          <complete>false</complete>
        </health>
      </instance>
```

```
      <bind nodeset="/health/phonenumber" type="string"
                                          jr:preload="property"
                                          jr:preloadParams="phonenumber"/>
      <bind nodeset="/health/userphone"   type="string"/>
      <bind nodeset="/health/doctor"      type="string"/>
      <bind nodeset="/health/appointment" type="date"/>
      <bind nodeset="/health/conditions"  type="select"/>
      <bind nodeset="/health/reason"      type="select1"/>
      <bind nodeset="/health/details_opt" type="select1"/>
      <bind nodeset="/health/details"     type="binary"
            relevant="selected(/health/details_option, 'y')"/>
      <bind nodeset="/health/complete"    type="string"
                                          jr:preload="complete"/>

    </model>
  </h:head>

  <h:body>

    <input ref="/health/userphone">
       <label>Please enter the phone number of the phone you are calling from.
              If you don't know the number, press 0.</label>
       <hint>digits=true</hint>
    </input>

    <input ref="/health/doctor">
      <label>Please enter the last name of the doctor you would like to
             schedule an appointment with.</label>
    </input>

    <input ref="/health/appointment">
      <label>For what day would you like to schedule the appointment? </label>
    </input>

    <select1 ref="/health/reason">
<label>What is the reason for your appointment?</label>
<item>
  <label>Routine Checkup</label>
  <value>routine</value>
</item>
<item>
  <label>Illness</label>
  <value>illness</value>
</item>
<item>
```

```
  <label>Immunization</label>
  <value>immunization</value>
</item>
<item>
  <label>None of the above</label>
  <value>none</value>
</item>
    </select1>

    <select ref="/health/conditions">
<label>Do you suffer from any of the following chronic conditions
       that your doctor should be aware of?</label>
<item>
  <label>Heart problems</label>
  <value>heart</value>
</item>
<item>
  <label>Asthma</label>
  <value>asthma</value>
</item>
<item>
  <label>Diabetes</label>
  <value>diabetes</value>
</item>
<item>
  <label>Allergies to medication</label>
  <value>allergies</value>
</item>
    </select>

    <select1 ref="/health/details_option">
<label>Would you like to record a message for your doctor
       to provide additional information? </label>
<hint>skipConfirmation=true</hint>
        <item>
  <label>Yes</label>
  <value>y</value>
</item>
<item>
  <label>No</label>
  <value>n</value>
</item>
    </select1>

    <upload ref="/health/details" mediatype="audio/*">
```

```
        <label>Please record a message for your doctor after the beep. </label>
      </upload>

    </h:body>
</h:html>
```

Appendix B

Scenario Experiment Instructions and Questionnaire

The following two paragraphs are the written instructions and scenario given to participants in the scenario experiment described in Section 6. The paragraph in italics was randomized for each participant.

> The following paragraph contains a fictional scenario in which you will be using an automated phone service for scheduling appointments at your doctor's office. Please read the paragraph, and when you are ready, call (857) 244-xxxx to schedule your appointment. Your call will be recorded for system evaluation. Feel free to refer to this page while you are on the phone. When you are finished using the phone service, please fill out the questionnaire on the opposite side of this page.
>
> *You will be calling an automated service for making doctor's appointments over the phone. You would like to make an appointment with Dr. Goldberg on 04/07/2010 because you would like advice on healthy eating. You suffer from asthma and you are allergic to Penicillin. You aren't able to miss work in the afternoon, so you need to let the doctor know that you will need a morning appointment.*

Figure B-1 contains a copy of the questionnaire completed by participants after completing the scenario experiment.

Automated Patient Check-In User Study Questionnaire

Name: _____ Age: _____

Gender: M / F Phone Number: _____

Highest level of education: _____

Languages Spoken: _____

Fluency in English: None 1 2 3 4 5 Fluent

How often have you used an automated system over the phone?

Never Once Several Times Less than Once A Month Once a Month Once a Week Daily

How much would you prefer to use this automated phone system compared to a live phone operator?

Greatly prefer operator 1 2 3 4 5 Greatly prefer automated phone

How much would you prefer to use this system compared to checking in online?

Greatly prefer online 1 2 3 4 5 Greatly prefer automated phone

Please describe your experience using the automated check-in, including any times that you were confused or made a mistake. We'd like to know what worked well and what you found difficult or confusing.

_____

_____

_____

_____

_____

_____

_____

Figure B-1: The questionnaire completed by participants in the scenario experiment.

Bibliography

[1] FrontlineSMS. `http://www.frontlinesms.com/`, April 2010.

[2] Open Data Kit. `http://code.google.com/p/opendatakit/`, April 2010.

[3] Project WET. `http://www.projectwet.org/`, April 2010.

[4] Yaw Anokwa, Carl Hartung, Waylon Brunette, Adam Lerer, and Gaetano Borriello. Open source data collection in the developing world. *IEEE Computer*, pages 97–99, 10 2009.

[5] Yaw Anokwa, Carl Hartung, Adam Lerer, and Gaetano Borriello. Deploying a mobile data collection tool in rural uganda. Unpublished. `http://web.mit.edu/alerer/www/surveyor_uganda.pdf`.

[6] Kuang Chen, Harr Chen, Neil Conway, Joseph M. Hellerstein, and Tapan S. Parikhhes. USHER: Improving data quality with dynamic forms. In *Proceedings of the International Conference on Data Engineering*, 2010.

[7] Michael H. Cohen, James P. Giangola, and Jennifer Balogh. *Voice User Interface Design*. Addison-Wesley, Boston, Massachusetts, first edition, 2004.

[8] B. DeRenzi, N. Lesh, T. Parikh, C. Sims, W. Maokla, M. Chemba, Y. Hamisi, D. S. Hellenberg, M. Mitchell, and G. Borriello. E-IMCI: Improving pediatric health care in low-income countries. In *CHI*, 2008.

[9] D. Forster, R. H. Behrens, H. Campbell, and P. Byass. Evaluation of a computerized field data collection. In *Bulletin of the World Health Organization*, volume 69(1), pages 107–11, 1991.

[10] D. Forster, R. H. Behrens, H. Campbell, and P. Byass. Development, implementation and preliminary study of a PDA-based bacteriology collection system. In *AMIA Annual Symposium Proceedings*, pages 41–45, 2006.

[11] J. R. Glass, T. J. Hazen, and I. Lee Hetherington. Real-time telephone-based speech recognition in the JUPITER domain. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 61–64, 1999.

[12] John D. Gould, John Conti, and Todd Hovanyecz. Composing letters with a simulated listening typewriter. In *CHI '82: Proceedings of the 1982 conference on Human factors in computing systems*, pages 367–370, New York, NY, USA, 1982. ACM.

[13] Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press, Cambridge, UK, first edition, 2004.

[14] JavaRosa. Home page. `http://code.javarosa.org`, April 2010.

[15] Kabutana Trust of Zimbabwe. Freedomfone. `http://www.freedomfone.org/`, April 2010.

[16] Open Data Kit. List of featured deployments. `http://code.google.com/p/opendatakit/wiki/FeaturedDeployments`, April 2010.

[17] Scott R. Klemmer, Anoop K. Sinha, Jack Chen, James A. Landay, James A. L, Nadeem Aboobaker, and Annie Wang. SUEDE: A wizard of oz prototyping tool for speech user interfaces. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 1–10, 2000.

[18] Jorn Klungsoyr, Peter Wakholi, Bruce MacLeod, Alberto Escudero-Pascual, and Neal Lesh. OpenROSA, JavaROSA, GloballyMobile - collaborations around open standards for mobile applications. In *Proceedings of The 1st International Conference on M4D Mobile Communication Technology for Development*, pages 45–48, 2008.

[19] P. Kotkar, W. Thies, and S. Amarasinghe. An audio wiki for publishing user-generated content in the developing world. In *HCI for Community and International Development (CHI Workshop)*, 2008.

[20] C. Kuun. OpenPhone project piloted in Botswana. `http://www.csir.co.za/enews/2008_july/ic_05.html`, April 2010.

[21] Indrani Medhi, S. N. Nagasena Gautama, and Kentaro Toyama. A comparison of mobile money-transfer UIs for non-literate and semi-literate users. In *CHI*, 2009.

[22] RW Millard and JR Carver. Cross-sectional comparison of live and interactive voice recognition administration of the SF-12 health status survey. *IEEE Computer*, 2(5):153–159, 1999.

[23] Frank Oberle. *Who, Why and How Often? Key Elements for the Design of a Successful Speech Application Taking Account of the Target Groups.* Springer, Berlin Heidelberg, 2008.

[24] Neil Patel, Sheetal Agarwal, Nitendra Rajput, Amit Nanavati, Paresh Dave, and Tapan S. Parikh. A comparative study of speech and dialed input voice interfaces in rural india. In *Proc. ACM Conference on Computer Human Interaction*, 2009.

[25] Somani Patnaik, Emmal Brunskil, and William Thies. Evaluating the accuracy of data collection on mobile phones: A study of forms, SMS, and voice. In *Proc. International Conference on Information and Communications Technologies and Development*, pages 74–84, 2009.

[26] M. Plauche, U. Nallasamy, J. Pal, C. Wooters, and D. Ramachandran. Speech recognition for illiterate access to information and technology. In *Proc. International Conference on Information and Communications Technologies and Development*, 2006.

[27] Roni Rosenfeld, Xiaojin Zhu, Arthur Toth, Stefanie Shriver, Kevin Lenzo, and Alan W Black. Towards a universal speech interface. In *Proceedings of the International Conference on Spoken Language Processing*, 2000.

[28] J. Selanikio and R. Donna. DataDyne brief. `http://www.datadyne.org/files/DataDyne_brief.pdf`, April 2010.

[29] J. Sherwani, N. Ali, S. Mirza, A. Fatma, Y. Memon, M. Karim, R. Tongia, and R. Rosenfeld. HealthLine: Speech-based access to health information by low-literate users. In *Proc. International Conference on Information and Communications Technologies and Development*, 2007.

[30] J. Sherwani and Roni Rosenfeld. The case for speech technology for developing regions. In *HCI*, 2008.

[31] Jahanzeb Sherwani, Sooraj Palijo, Sarwat Mirza, Tanveer Ahmed, Nosheen Ali, and Roni Rosenfeld. Speech vs. touch-tone: Telephony interfaces for information access by low literate users. In *Proc. International Conference on Information and Communications Technologies and Development*, pages 447–457, 2009.

[32] Bernhard Suhm. *IVR Usability Engineering Using Guidelines And Analyses Of End-to-End Calls*. Springer, US, 2008.

[33] Roger Tourangeau, Mick P. Couper, and Darby M. Steiger. Humanizing self-administered surveys: experiments on social presence in web and IVR surveys. *Computers in Human Behavior*, 19(1):1 – 24, 2003.

[34] UNCTAD. Information economy report 2007-2008: Science and technology for development - the new paradigm of ICT. In *United Nations Conference on Trade and Development*, 2008.

[35] International Telecommunication Union. ICT statistics. `http://itu.int/ITU-D/ict/statistics`, April 2010.