Open Data Kit: Tools to Build Information Services for Developing Regions

Carl Hartung, Yaw Anokwa, Waylon Brunette, Adam Lerer, Clint Tseng, Gaetano Borriello

Abstract— **This paper presents Open Data Kit (ODK), an extensible, open-source suite of tools designed to build information services for developing regions. ODK currently provides four tools to this end: Collect, Aggregate, Voice, and Build. Collect is a mobile platform that renders application logic and supports the manipulation of data. Aggregate provides a "click-to-deploy" server that supports data storage and transfer in the "cloud" or on local servers. Voice renders application logic using phone prompts that users respond to with keypad presses. Finally, Build is a application designer that generates the logic used by the tools. Designed to be used together or independently, ODK core tools build on existing open standards and are supported by an open-source community that has contributed additional tools. We describe four deployments that demonstrate how the decisions made in the system architecture of ODK enable services that can both push and pull information in developing regions.**

I. Introduction

OVER the last fifty years, advances in information and communication technologies (ICTs) have transformed the way we create, retrieve, update, and delete information. Despite this revolution in information management, much of the world has not benefited from these technological advancements. To address many of these disparities, there has been a push from development agencies to apply evidence-based development wherein best available data is used to inform development challenges.

In a sense, this approach is not new. From agricultural extension to immunization campaigns, services that push and pull information in developing regions have been at the heart of global development. However, even with many years of practice, providing these services is still a difficult task. Current practice, which is primarily paper-based, limits the scale and complexity of the services that can be provided, and thus the impact of the intervention.

With the growth of mobile phone usage in these regions [1], there have come opportunities to digitize and automate many of these services in a cost effective manner. Of course, computing is no panacea, as noted by Toyama et al. [2] and Brewer et al. [3]. Challenges ranging from user limitations to infrastructure constraints have proven to be particularly pernicious. In the rare instances where the introduction of computing has been successful, implementation has often required a level of technical expertise not readily found in situ [4].

For computing to truly address the information gaps in developing regions, information services must be composed by non-programmers, deployed by resource-constrained organizations, used by minimally-trained users, and remain robust despite intermittent power and connectivity. To address these challenges, we developed Open Data Kit (ODK) [5], a modular, extensible, and open-source suite of tools designed to empower users to build information services for developing regions. ODK currently consists of four tools: Collect, Aggregate, Voice, and Build.

ODK Collect is a mobile platform that renders complex application logic and supports the manipulation of data types that include text, location, images, audio, video, and barcodes. ODK Aggregate provides a "click-to-deploy" server that supports data upload, storage and transfer in the "cloud" as well as on local servers. ODK Voice renders application logic using automated phone prompts that users respond to with keypad presses. Finally, ODK Build is a drag-and-drop application designer that generates the logic used by the tools.

Designed to be used together or independently, ODK tools build on existing open standards and empower individuals and organizations to compose services that collect and distribute information in the developing world. ODK is supported by an open-source community that has contributed training documents, localization support, as well as additional tools.

Examples of how ODK can be used include:

- Government workers completing socio-economic surveys about households in a district.
- Agricultural extension workers creating an application with video and audio clips explaining farming techniques.
- Teachers implementing games with interactive questions and answer tutorials and automatic score recording.
- Crisis workers capturing images and locations of damaged areas after an earthquake.
- Funders receiving geo-tagged reports of interventions they have supported.
- Clinicians building decision support applications that use patient data to help determine when to administer tests.
- Microfinance institutions tracking transactions from lenders and borrowers.
- Indigenous tribes cataloging their trees to enable participation in global carbon markets.
- Community health workers managing household visits to pregnant women.

This work is supported by Google.org and Google.com through their extensive and continuing financial and technical assistance.

C. Hartung, Y. Anokwa, W. Brunette, C. Tseng and G. Borriello are with the Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195. Email: chartung@cse.uw.edu, yanokwa@cse.uw.edu, wrb@cse.uw.edu, cxlt@cse.uw.edu, gaetano@cse.uw.edu

A. Lerer is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139. Email: alerer@mit.edu

In this paper, we describe how ODK differs from previous work, detail the set of tools we currently provide, and evaluate four ongoing deployments. We also discuss how the design decisions made in the system architecture of ODK were key to enabling a large and varied set of applications for developing regions.

II. Motivation

We examined existing data collection and dissemination systems before deciding to build ODK. Here we briefly describe these systems, their limitations, and how they influenced our design decisions. We provide a more extensive discussion of related work in Section V.

Historically, information services for developing regions have been PDA-based [6]–[8]. The first example of these is CyberTracker [9], a system first developed in the mid-1990s as a way to enable non-literate animal trackers to record observations on PDAs (sometimes with attached GPS units) using a purely graphical and non-linear interface. Trackers, when observing a specific animal behavior, tap a representative icon on the screen to mark that behavior. For applications such as socio-economic surveys, CyberTracker replaced animal behavior icons with icons representing families, houses, and marriage status.

CyberTracker is still in wide use today and has added functionality including a form designer, data synchronization over the web, and image capture. While these upgrades build toward a more generic system, they do not change the fundamental use case and interactions. That is, CyberTracker is designed for gathering large quantities of geo-referenced data for illiterate field observers and synchronizing those observations to a local computer.

For broader use cases than CyberTracker targets, Pendragon Forms [10], [11] has been a popular and fully-featured commercial solution that includes a form designer, data synchronization, multimedia support and forms with navigation logic. Although designed for developed regions, Pendragon Forms has also been used all over the world [12], [13].

Our work differs from Pendragon Forms along four dimensions that are critical for resource-constrained environments: cost of deployment, ease of extensibility, available devices, and data transport. For the functionality ODK provides for free, Pendragon Forms requires $80 per user. Additionally, because of our open-source license, organizations are free to modify and customize the applications as needed. Finally, ODK runs on a variety of phones, netbooks, tablets and supports multiple methods for transferring data to other services.

There are free and open-source competitors to Pendragon Forms that we also considered. Java Platform, Micro Edition (J2ME) phone-based data collection clients such as FrontlineForms [14], EpiSurveyor [15], CommCare [16], and JavaRosa [17] have become popular as the prices of Java-enabled phones have fallen. Unfortunately, these phones live in a fragmented ecosystem that negatively impacts software development and usability.

Applications must often be signed by the vendor, carrier, or manufacturer before interactions with storage, networking, or hardware accessories are usable. Without the appropriate digital certificates and signatures, users are prompted with confusing dialogs before every such action. The signing process can require months of waiting and thousands of dollars. Even after signing authority is obtained, capturing images, audio, video, and location remains difficult because each device implements the interface to its underlying hardware differently. J2ME programmers are forced to test every software release on each physical device they wish to support – a requirement that nullifies the benefit of a wide phone base.

Like Pendragon Forms, many of these J2ME-based systems do not support the free flow of information that is captured. For example, to use FrontlineForms on a phone, a user can only transfer data to/from a PC running FrontlineSMS. Furthermore, FrontlineForms uses a proprietary format that makes it difficult for organizations to transfer data using another mobile client or service.

In the academic literature, the best example of data capture and survey software is Froehlich et al.'s MyExperience [18]. MyExperience gathers objective data such as user context (as sensed by the device. e.g., current location) along with sensor readings. It uses context-triggers to capture in situ subjective user feedback. MyExperience's use case is to collect quantitative and qualitative data in the field to support studies of mobile technology usage and evaluation and, more recently, psychological studies of human attitudes and behavior. However, there is no focus on building information services that deliver information or managing the specific challenges of developing regions.

CAM [19] by Parikh et al. is the first example of an information service toolkit designed for developing regions. Primarily used for data collection, key elements of the system include a close linkage to paper forms, an image and audio driven user-interface, support for both asynchronous or synchronous connectivity, and a scripting language that describes form logic. CAM was designed to augment rather than replace paper forms. Users fill out the paper form first, and then use the phone's camera to trigger data entry and submission.

In ODK, we believe the problems CAM solves with the linkage on paper are no longer an issue. The fact that later CAM applications do not rely on paper-based navigation suggests that others have reached the same conclusion [20]. While CAM's procedural scripting makes it easier to design iterative constructs in the forms, declarative languages offer more in the way of validation and optimization – functionality that is useful in developing regions [21]. Finally, CAM deployments have required forms to be designed by hand and necessitated custom programmed back-end applications to receive the submitted data.

In examining these systems, we found that many had similar limiting factors. That is, they were built as monolithic, siloed solutions using proprietary data formats and interfaces. Many were built for specific hardware platforms and required large-scale redesign to update to newer technologies. Thus, we set out to build ODK with this set of requirements:

- **Modular Components**. By focusing on creating small, composable modules, we can create a system that is easier to extend and modify.

- **Open Source**. By utilizing open source software and interfaces based on open standards we are able to leverage a wider developer base allowing more participation from the community.
- **Cutting Edge Technology**. Rather than building for a specific hardware platform, we developed our applications on systems that are likely to persist and evolve over the long-term, provide a diversity of available form-factors, and adapt to new capabilities made available by the rapid pace of innovation in this space.

The contributions of ODK center on how the choice of platforms and architecture design enable and encourage the growing number and variety of deployed applications. Though some of these applications were envisioned by CAM [22], the design requirements at the core of the ODK project have allowed many of them to be realized in a short time. Although one can certainly cobble together many of these services with existing software and hardware, ODK that makes this composition easier and more deployable for non-programmers. We argue these distinctions matter and as evidence we note that parts of CyberTracker, MyExperience, CommCare and EpiSurveyor have been ported to the ODK platform.

III. System Design

A. Example Scenario

To demonstrate the types of problems we intend ODK to help solve, we present an example. Imagine a community health organization working in rural Africa. This organization has several goals: to gather statistics about the prevalence of HIV, TB, and malaria, to treat as many of the affected people as possible, to educate the population about how different diseases are transmitted, and to build a database so that patients and their health care providers have complete medical records. To accomplish these goals, the organization has hired a handful of community health workers to travel through villages, meet with residents, administer voluntary testing and counseling, and record the results.

In this scenario, the community health worker needs:

- a way to record and/or retrieve patient information;
- training materials to educate the patients;
- knowledge of when to administer specific tests, and testing materials; and
- a way to deliver the collected information to the central clinic.

The organization needs:

- to train community health workers on how to register new patients, or record follow up visits and test results;
- collect the forms and digitize them in a timely manner;
- secure and maintain a place to store all of their data so it is easily retrievable; and
- potentially scale the size of their program from a few health workers to hundreds, and from tens of patients to millions.

This is what we consider an information service and to address needs such as these, we developed a client for a mobile phone that has the ability to record information, display education materials, provide decision support based on input data, and wirelessly transmit information to a central clinic. Additionally, we developed a scalable server-side data storage system to easily store and retrieve the digital records. Both client and server are designed to work in the challenging environments found in developing regions.

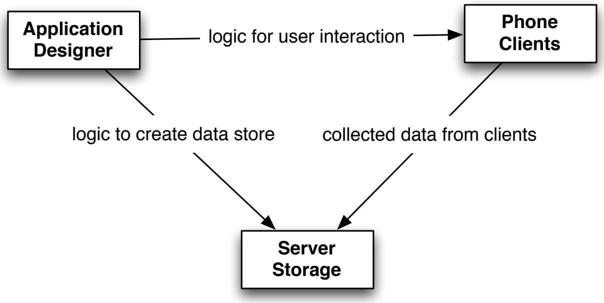

Application Designer
logic for user interaction
Phone Clients
logic to create data store
collected data from clients
Server Storage

Fig. 1. The current components of ODK. An application designer generates logic for user interaction and for data store creation. Phone clients can run the logic locally and send data to server storage.

B. Tool Design

Open Data Kit is designed as a modular set of components that can be used individually or in various configurations (including modules that are not part of ODK) to create information services in developing regions. The current components shown in Figure 1 are an application designer, a mobile device client, a basic interactive voice response (IVR) system, and a server for data storage.

1) Application Designer: In order to allow non-programmers to build applications with complex logic and interactions, we designed a web-based graphical designer that allows users to create applications using a drag-and-drop metaphor. Our application designer produces the logic used by all the other tools to render the application to the user as well as create databases from which data can be extracted for visualization and reporting.

2) Smart Phone Client: For our primary client, we designed a mobile phone application that allows users to download application logic, interact using physical or on-screen keyboards and touch, and send information to servers wirelessly. Mobile phones are ideally suited to the types of applications we targeted because of their small form factor, lower price relative to PCs and laptops, ability to run in disconnected environments with intermittent access to power, and almost universal familiarity amongst our typical users.

We made the controversial decision to build our application using current generation smart phones. We chose these phones for reasons of their programmability, robust feature sets, enhanced interaction modalities, and increased processing power. Current generation mobile phones have features such as built-in cameras, GPS, and touch screens. Unlike their predecessors, new operating systems and programming APIs give developers unfettered programmatic access to these components. Furthermore, memory and processors in the current generation of phones are approaching the speed of laptops, removing many of the constraints which impeded previous systems.

While others considered such phones to be expensive and unavailable in developing regions, we predicted that the trend of ever increasing mobile adoption and technology would, within the next few years, lower cost and increase availability. So far, prices of first generation phones have continued to drop sharply and carriers in these regions have started offering the phones for sale.

Since current generation smart phones would likely be too expensive for many living in developing regions, we targeted our tools at the employees of organizations (non-governmental organizations (NGOs), government ministries, etc.) working in these resource-constrained environments. Even so, given the current trends, we suspect that our system will soon be used for crowd-sourcing data, or even creating local businesses using ODK for information gathering and delivery.

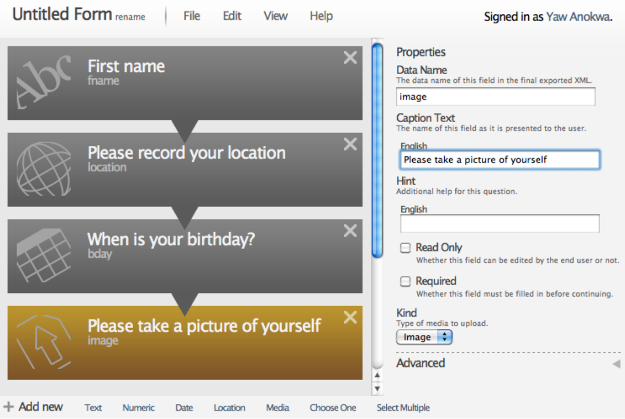
3) Server Storage: To simplify the process of data storage and management, we designed a server that could run either locally on a PC or on one of several cloud computing infrastructures. Many organizations lack the computing infrastructure to store and analyze large amounts of data, as well as the technical knowledge required to create and maintain a data storage system. Additionally, the costs associated with running, maintaining and scaling traditional database backends can be prohibitive.

Our server builds a data store for a specific application when that logic is uploaded by a user. That is, the same application logic has dual purposes: for collecting information when rendered on the client device, and for providing a guide to the server on how to build a corresponding database. The logic can then be sent to clients for user interaction, and clients can send resulting data back to the server. Users can view the results in a table format, or export the data into one of several other formats such as CSV (comma-delimited spreadsheets) for importing into statistical analysis tools, JSON (a standard serialization protocol) for external servers, or KML (mapping markup language) to be viewed on a variety of mapping software.

By leveraging the cloud version of our server, organizations can pilot a system without the burden of purchasing, configuring and maintaining local servers. From there, cloud services allow virtually unlimited scalability. Running servers in the cloud also alleviates the challenges of power outages, finding skilled IT managers, managing backups, hardware failures, and protecting from viruses. We support multiple cloud hosting solutions.

We realize, however, that many organizations may be hesitant to fully trust cloud computing services due to privacy or political concerns. Thus, we designed our server to also run locally on a PC for those wishing to keep their data in-country and on storage under their complete control. Users can choose either approach to store their data and migrate between them at any time as the interfaces to both types of servers are identical.

4) Basic IVR System: In order to enable users without smart phones to use our system, we designed an interactive voice response (IVR) system that can be used to call basic phones owned by the general population. To interact with the service, users connect to the system by responding to automated telephone calls, or by calling a central number. Once connected, users can then interact with application logic by entering numbers on their keypad or recording audio responses.



First name
fname

Please record your location
location

When is your birthday?
bday

Please take a picture of yourself
image

Properties

Data Name
The data name of this field in the final exported XML.
image

Caption Text
The name of this field as it is presented to the user.
English
Please take a picture of yourself

Hint
Additional help for this question.
English

□ Read Only
Whether this field can be edited by the end user or not.

□ Required
Whether this field must be filled in before continuing.

Kind
Type of media to upload.
Image

Fig. 2. In ODK Build, prompts appear on the left of the screen while properties appear on the right. Users rearrange prompts using a drag and drop interaction in the web browser.

Voice prompts are recorded in the local language or dialect to allow illiterate populations to be included. This basic IVR system was designed to use the same logic as our smart phone client, enabling a variety of ways to interact with the system. IVR to client's own phones can be especially useful for follow-up data gathering when sending a human worker may be cost-prohibitive or inefficient.

IV. System Implementation

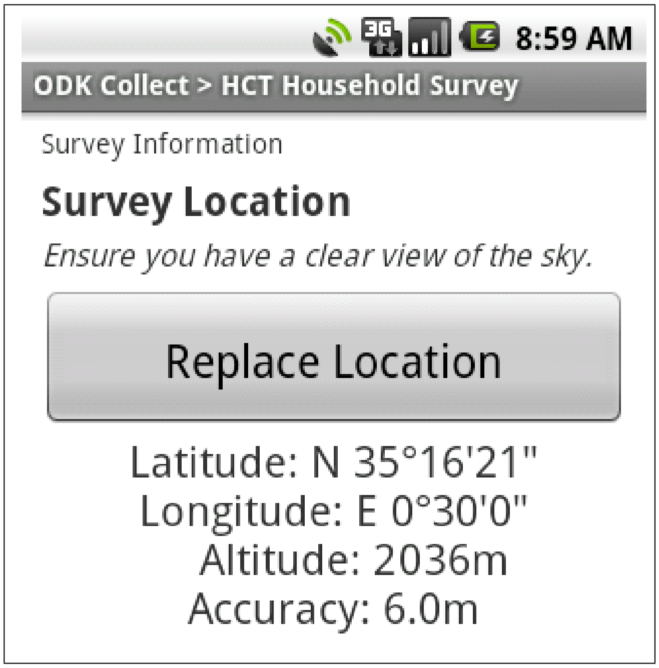
A. XForms: A Common Format

To ensure that each of the tools could be used independently but also with each other, we use the XForm standard [23]. XForms are an XML-based form description standard designed by the W3C for the next generation of web forms. We implemented the OpenRosa [24] subset of XForms that makes our tools compatible with many others' tools in the NGO community and enables these organizations to easily move to different technologies and systems without re-building their applications.

With OpenRosa XForms, designers can specify a wide variety of control and data types including: text, integer, decimal, select-one, select-multi, image, audio, video, barcode, and location. Designers can also create entry constraints, read-only prompts, required fields, multi-lingual translations, and branching logic.

B. Build: Application Designer

By using the XForms standard, ODK provides authors with flexible options for designing services to fit their needs. Unfortunately, this comes at the cost of ease of use. The structure and syntax of the XML-based format is more complex than is needed to build the majority of forms, which are usually short and linear. This often leads to confusion among users and presents a barrier to adoption.



ODK Collect > HCT Household Survey

Survey Information

Survey Location

Ensure you have a clear view of the sky.

Latitude: N 35°16'21"
Longitude: E 0°30'0"
Altitude: 2036m
Accuracy: 6.0m

Fig. 3. A location prompt in ODK Collect captures latitude, longitude, altitude and accuracy from the phone's GPS.

ODK Build enables users to easily create their application logic and generate XForms without a detailed working knowledge of the XForms standard. Build lets designers drag and drop each prompt the user will interact with onto a canvas as shown in Figure 2. Each prompt has a set of properties (prompt text, data type, etc.) that users can edit. Users can rearrange ordering of prompts or add custom logic to each prompt. Build is implemented as an HTML5 web-based application using Javascript and Ruby Rack.

In keeping with the aim of providing more accessible methods to author services, a number of compromises were made with regards to the more complex, but less commonly used, features of XForms. For instance, while we still provide a means to implement more advanced constraints, we highlight only basic range constraints for most data types; and, while in reality XForms defines whether or not to display each prompt based upon a set of rules specific to that prompt, we optimize for the more common case and represent conditional logic as a physical branch, with each branch containing multiple interrelated prompts.

To make Build widely available, we host a running instance on our own server so that anyone with a web browser on the internet can immediately begin authoring services for their own use. For offline support, users can download the source and run an instance on their local machine.

C. Collect: Smart Phone Client

Collect is ODK's mobile client that runs on any device capable of running the Android operating system. Collect takes the XForm logic and displays prompts to the user in a one-prompt-at-a-time format. Users navigate forward and backward by moving their finger across the screen (with a similar gesture to turning a page in a book). Collect also supports a jump-to-prompt feature allowing a user to skip forward or backward quickly within the application. Prompts can be specified as one of the following: free text, integer, decimal, select-one (radio buttons), select-multi (checkboxes), image, audio, video, barcode, or GPS location (as shown in Figure 3). Any prompt element can also be made read-only to convey, rather than record, information. Collect supports multiple languages, advanced constraint checking, and regular expression checking of entered data. Additionally, Collect supports branching and repeating groups of prompts to allow for more complex form interactions (e.g., collecting basic demographic data for every member of a household).

In order to support disconnected operation, Collect stores application logic and resulting data on the phone as XML files along with associated binary files (images, audio, video, etc.). The user can choose to synchronize with a server at any time using any available internet connection. Files are sent using a standard HTTP POST to any OpenRosa compatible server (such as the servers we provide but many others as well). These files may be transferred to and from the phone with a USB cable connected to a computer, or by removing the phone's SD card and readings its contents on a completely separate device.

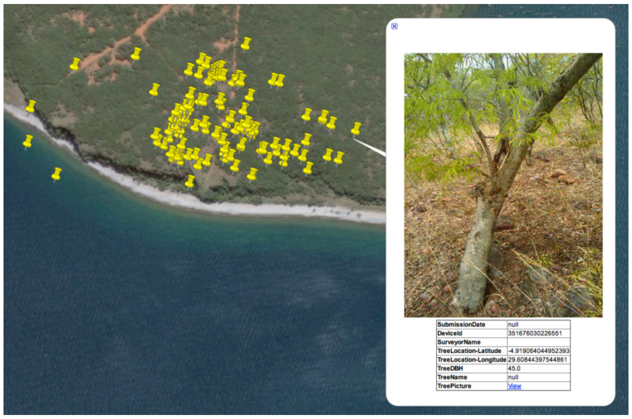
Collect is written in Java, and runs on smart phones that run Android, an open-source operating system for mobile phones. Development for Android is made simple because the Android API has been standardized so that applications do not concern themselves with the details of the interfaces of the underlying hardware. Also, since Android is not bound to a particular hardware implementation it runs on a variety of devices including smart phones, netbooks and tablets.

An additional benefit of the Android platform is its "intent" system, that allows other applications to launch Collect. For example, a developer can create a location-based application that launches a survey when a user reaches a specific location (in the style of MyExperience). Such an application does not require any modification to Collect but simply uses it as a subroutine. Existing phone applications can be exploited by Collect using the same mechanism. For example, Collect supports reading barcodes using a separate application available from Android Market. New capabilities can also be incorporated this way, allowing parallel development and keeping the size of the modules as small as possible.

D. Aggregate: Server Storage

Aggregate provides a server repository to manage collected data, provide standard interfaces to extract data (e.g., spreadsheets, queries, etc.), and integrate with existing systems via HTTP web requests. Aggregate supports returning data in many standard formats including CSV (for statistical analysis), KML (as shown in Figure 4), and JSON (for transport to other web services). Aggregate is designed to be a generic data storage service that will run on the user's choice of computing platform. Importantly, it can receive data from any phones and servers which are OpenRosa compliant.

To achieve this model, Aggregate uses the concept of a generic data storage wrapper to abstract away the details of



SubmissionDate	null
DeviceId	351676030226551
SurveyorName	
TreeLocation-Latitude	-4.919064044952393
TreeLocation-Longitude	29.60844397544861
TreeDBH	45.0
TreeName	null
TreePicture	View

Fig. 4. ODK Aggregate can export data as a map. Selecting an individual point on the map reveals the data collected at those coordinates.

a specific data storage service, thus allowing the application layer of Aggregate to call a common interface. This abstraction allows the majority of Aggregate's code to be storage platform independent. Aggregate is written in Java and designed to run in a standard Java web container – the only restriction to its portability. To verify the design, Aggregate was tested on two cloud services (Amazon Web Services and Google's AppEngine platform) and two traditional databases (MySQL and Postgres).

Aggregate's interface and abstractions were designed to simplify data management by creating an application that is usable by people with only basic computer skills. Users simply need to upload the XForm and Aggregate automatically creates the new relations in the data store based on the XForm prompt types. The creation of relations and data mappings at runtime is unlike traditional object relational mappings (e.g., Hibernate), that abstract database details at compile time instead of runtime. Users are able to construct queries using a web interface without knowing the syntax of the underlying database's query language. Aggregate's web interface also provides a report interface to retrieve data in addition to the ability to export the stored data in common data formats.

E. Voice: Basic IVR System

ODK Voice is a platform for rendering XForms through automated telephone calls. Users can interact by calling with a phone or receiving automated calls from the system and either entering answers on their keypad or recording audio responses. Voice enables application interaction to the millions of people in the developing world who already have their own mobile phones [1].

ODK Voice is implemented as a Java web server application that can run on Tomcat or other servlet engines (see Figure 5). Administrative tasks, such as uploading XForms, recording prompts, and scheduling outbound calls, can thus be performed directly over the web. To render automated (IVR) voice calls, ODK Voice produces VoiceXML, a W3C standard for "creating audio dialogs that feature synthesized speech, digitized audio, ... and DTMF key input" [25]. Using the open VoiceXML standard allows ODK Voice to interface with a number of voice engines.

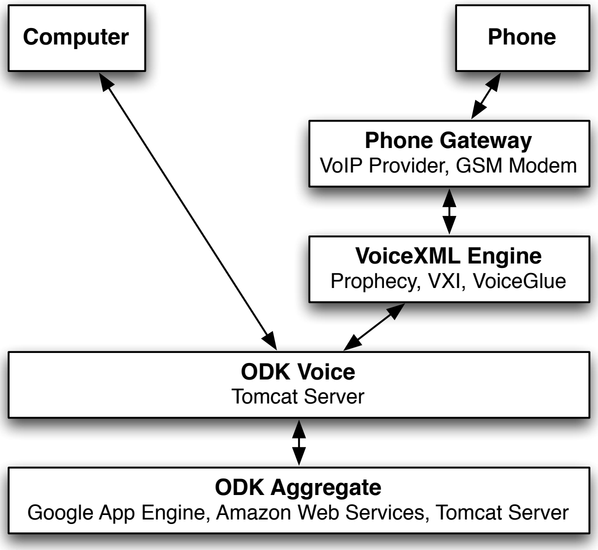


Fig. 5. The software stack that enables ODK Voice.

To connect to a telephone network, the VoiceXML engine interfaces to a gateway, which can either be a GSM modem connected to the server or a remotely hosted voice-over-IP (VoIP) gateway. Using a remote VoIP gateway provides the added advantage that the server can be hosted out of country. For simplicity of deployment, the entire ODK Voice infrastructure can be packaged and loaded onto a target server. Data collected from ODK Voice is submitted to an ODK Aggregate instance for aggregation and viewing.

ODK Voice optimizes for learnability over efficiency, since we expect most users to be novices; for example, there is a dialogue after each question where the user can confirm his answer or opt to repeat the question. Many default behaviors in ODK Voice can be overridden by special "hint" attributes in the XForm, but this is not generally necessary.

Some types of questions are better suited to the voice medium than others. In particular, string entry is very difficult over a telephone dialogue. Unlike writing a text message, filling out a telephone string provides no immediate feedback about what has been entered.

Our approach to string entry is to have users enter each word letter by letter (e.g. 733 for "red"), and then have the system "guess" the most likely word the user entered. Similar to predictive text systems like T9, the system maintains a dictionary of all possible words (either a general dictionary or a special corpus for that question); each word has a likelihood value assigned to it.

When a user enters a series of digits, the system calculates the number of errors between the entered numbers and each word in the dictionary; the likelihood for that word is calculated by multiplying the prior likelihood by the likelihood of the calculated number of errors. The user can then pick from a list of likely words. The dictionary priors are adaptively re-weighted so common answers are suggested first. However, we find that it is easier to avoid string entry and ask for answers through selection from a list or recording audio.

V. Related Work

Paper has been the perennial favorite for delivery and collection of information in developing regions. However, the very properties that make paper popular are also liabilities. For example, after a natural disaster, quickly and accurately documenting the extent of damage with GPS and images is essential to planning a response [26]. In rural hospitals, helping untrained nurses triage patients using complex medical protocols is critical to effective health care [27]. Even basic surveying without data validation at the point of entry can result in serious errors [28].

Radio and television fare no better. While good for broadcasting messages, they have yet to be used broadly for collection of many types of information [29]. Computing, as delivered by desktops and laptops has not been successful [3], [30] for developing regions. To build the complex services that can provide value in developing regions, we must look to systems built around mobile devices. Moreover, we must go beyond the obscure, like SIM Toolkit [31], or operator-controlled, like Unstructured Supplementary Service Data (USSD).

Voice is a good modality for the illiterate, and speech-based systems have shown promise for delivering information [32], for creating communities of practice [33], and providing health surveillance [34]. Voice, while sometimes successful for data collection [35], has not been shown to enable the variety of services we wish to provide.

The most common electronic information services in developing regions are SMS-based. Examples of these include FrontlineSMS [14] and RapidSMS [36]. The strength of these platforms is the wide availability of SMS service and the low cost of basic SMS phones. For systems like FrontlineSMS, the ease of use of the platform (just a computer and a mobile phone) has enabled wide adoption for organizations who primarily use it to communicate and gather unstructured data for short surveys, election monitoring, and even crisis reporting.

RapidSMS, on the other hand, is designed for larger scale deployments and is used primarily for structured data. With a structured SMS, users must compose commands following a particular syntax. For example, to report name, age and gender, you might send *'f:John l:Doe a:14 g:m'*. These message structures are hard to enter consistently and so organizations must contend with similar data cleaning problems as systems using unstructured messages. To accommodate longer sessions, some systems use a "question at a time" protocol but these can lead to lengthy message exchanges.

Despite the popularity of SMS-based tools, SMS is unreliable and expensive as a transport mechanism. Messages can arrive late (or never arrive) and the cost of sending 1Mb over SMS is over 3600 times more expensive than GPRS [37]. Additionally, since the collected data can only be easily sent in cleartext, there are security and privacy implications. For organizations that need to send sensitive data, to confirm data transmission, or to send more than a few hundred bytes of data, SMS is simply not viable.

VI. Discussion

In this section, we discuss ODK in the context of four implementations. Our aim is to capture the factors that go into choosing ODK and the consequences of that decision. We use a holistic qualitative approach, and focus on what, if anything, ODK enabled or enhanced for these implementers.

A. Methodology

We selected four organizations from our users' forum who were familiar with existing systems and had used ODK for some time. All four organizations had an implementation team with a software developer and a project manager.

We emailed each team with five open-ended questions designed to understand how they were using ODK, the alternatives they considered, their current usage, and the impact of the platform on their work. We asked each team to work together on the responses and send their combined answers back via email. The questions, answers and emails were in English.

We provide varying levels of assistance to all our users and that could introduce bias in the responses. In the case of these particular organizations, assistance included providing five sample phones to two teams and helping train users in the field for one team. We also fixed occasional minor bugs when the implementation developers requested help. After this initial support, all implementers have brought the projects to scale without our assistance.

To further address potential bias, the surveys were sent months after any assistance and implementers were asked to be *"accurate, objective, clear and verbose"* and to *"describe why [ODK] has been better or worse than the alternatives."*

B. Berkeley Human Rights Center

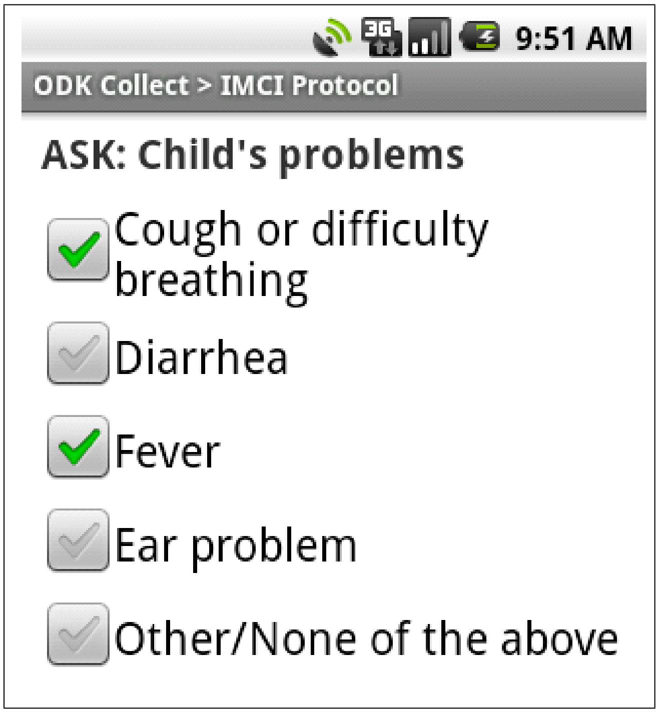
The Berkeley Human Rights Center (HRC) [38] uses innovative technologies and scientific methods to investigate war crimes and human rights abuses. They develop policy measures to protect vulnerable populations and train the next generation of human rights defenders. The HRC uses ODK to conduct field research (shown in Figure 6) in developing nations by performing large scale population surveys in areas vulnerable to post-conflict issues of accountability and transitional justice.

The environments in which the HRC uses ODK are quite demanding. *"Each survey can be 300 questions, and we do 2000 to 3000 surveys in 1-2 months. It is extremely intensive and requires speed and accuracy."*

The HRC had traditionally used paper in their survey work primarily because *"no programming [is] required, so it's easy to implement for non-technical people."* The choice to switch to ODK was driven primarily by cost, usability, latency, errors and scale. That is, *"there is a point where the money saved on paper production and shipping, and the money saved by removing the steps of data-entry by hand, balances everything out...the increase in data accuracy is an argument for spending the money."*

Because enumerators must be quickly trained when HRC starts a project in a new locale, usability was also a concern.

Fig. 6. An enumerator collects data in the Central African Republic for the University of Berkeley's Human Rights Center.



9:51 AM

ODK Collect > IMCI Protocol

ASK: Child's problems

☑ Cough or difficulty breathing

☐ Diarrhea

☑ Fever

☐ Ear problem

☐ Other/None of the above

Fig. 7. A sample prompt from an ODK Collect rendered medical flowsheet from D-Tree International. Such flowsheets can provide automated diagnoses.

"It is relatively easy to train enumerators using ODK as the interface is highly usable, even for people who have little or no computer experience." Decreasing training costs is an important consideration in overall system costs especially for organizations with high employee turnover rates.

Evaluating data as it was collected was also important. *"We have found that the ability to synchronize and analyze data daily, as it comes in significantly improves the quality of the data. We can look for errors in methodology from data...and make corrections in the methodology so that further data is clean."*. The HRC team plots the GPS locations of an individual enumerator to ensure sampling points are spaced correctly. Start and end times of surveys are also tracked to manage fraud.

HRC considered EpiSurveyor and VisualCE before choosing ODK. EpiSurveyor was not free for the number of surveys needed, did not work on a wide variety of devices, and (at the time) had an unreliable form builder. VisualCE had database synchronization, but the development environment was considered awkward for larger surveys.

HRC also noted that, the ODK development community *"is very active and friendly. Finding fixes for mutual problems and sharing development for common benefit."* Indeed, the HRC has played an important role in the community – localizing ODK Collect into French, providing training guides and the Kobo PostProcessor [39], an ODK-compatible tool to process data in the field.

C. D-Tree International

D-Tree International [40] is an organization that aims to reduce the high rates of illness and death from preventable and treatable diseases worldwide. Their approach centers on the development and use of treatment protocols for the most commonly diagnosed illnesses based on best field practices. The protocols are programmed into inexpensive PDAs and mobile phones for use by health workers in both clinical and community settings. D-Tree is using ODK to trial an electronic version of the WHO's Integrated Management of Childhood Illness [41] protocol in Tanzania (shown in Figure 7). They also plan on using ODK as part of a maternal health system.

For D-Tree, maintaining compatibility with their existing systems was important. *"ODK was the obvious platform to use since it is able to execute precisely the same XForms that we'd been using on cheaper phones."*

When asked to compare ODK to their existing J2ME and PDA systems, ODK had *"quick and easy user interaction"* as a positive trait. In one case, a clinical officer was more responsive to the health protocol once he had used it on ODK. This is important because if the treatment protocol is unpleasant to use, it will be ignored and likely result in worse patient outcomes. D-Tree noted there was *"a bit of technophobia by users when it comes to the higher end phones that ODK runs on."*

Despite these constraints, the switch to ODK was driven by an ability to *"integrate and test easily features such as photos, GPS with a few lines of code...J2ME had its limitations."* Some of these limitations are best explained by how D-Tree is using ODK for the cases that other platforms and devices do not support.

D-Tree has created a peer-to-peer application [42] that synchronizes patient information across phones and servers. They have exposed this data so that any application (including ODK) can programmatically create, read, update or delete

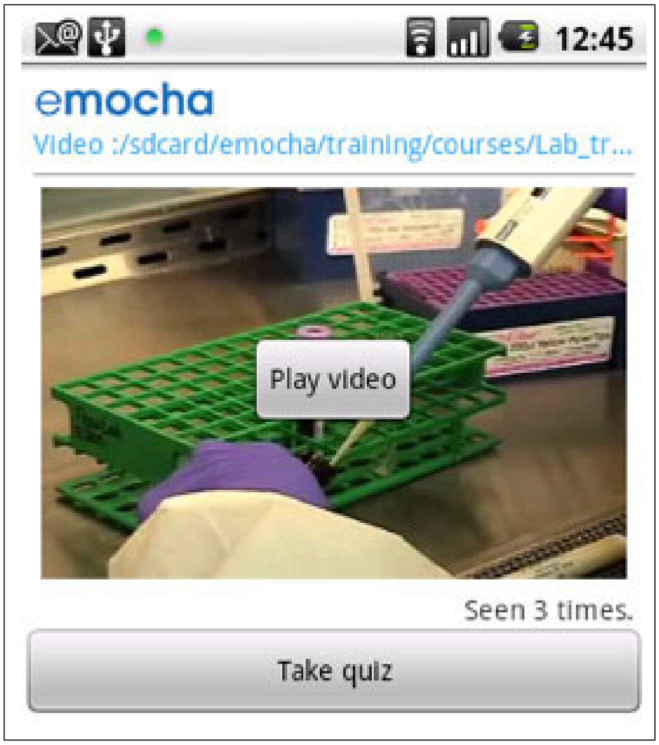


Fig. 8. The eMOCHA application provides video-based training for health workers. After each video, eMOCHA launches ODK Collect to test users.

patient data. ODK Collect can submit data to this on-phone medical record system and have the changes propagate to other phones via Bluetooth or WiFi. This functionality enables a portable patient record system that works across a variety of connectivity scenarios found in developing regions.

As D-Tree notes, *"features sometimes trump cost, one has to make the choice between enhanced UI vs. minimal UI, advanced features vs. basic functionality, in our case we needed to get more out of the phone and Android/ODK allowed us to do that."*

D. Johns Hopkins Center for Clinical Global Health Education

eMOCHA [43] is a free open-source application, developed by the Johns Hopkins Center for Clinical Global Health Education (JHCCGHE) [44]. eMOCHA is designed to support JHCCGHE's mission by improving health provider communication and education, as well as patient care in developing regions. eMOCHA does this through a mobile and server component.

For the mobile component, JHCCGHE chose to use ODK and customized the user interface. JHCCGHE's explains that ODK *"is used for patient data gathering, and to give an exam after watching a video course on the phone, to make sure the user understood the content."* In the latter case, eMOCHA launches ODK Collect to gather feedback after a video session as shown in Figure 8.

eMOCHA's server collects, analyzes and displays the collected clinical information and statistics. The mobile component gathers clinical data, provides training through video and distance learning, and enables messaging and consultation between local and remote clinicians.

When asked about alternatives they considered, JHCCGHE stated that their *"goal is not data gathering itself, but the creation of a tool for health workers. We were interested in the Android platform and at the time we did not find other open source alternatives we could integrate. We did consider creating a form system ourselves. We chose to go for ODK to save development time and follow the XForms standard, even if we would lose freedom and flexibility."*

Asked to explain the loss of freedom, JHCCGHE reported that *"there is always this dilemma, should I build it myself, or use this existing software/framework. What will take more time? Will the existing software do everything I want?"*. JHCCGHE also noted that ODK did not support encryption of potentially sensitive patient data. This is functionality JHCCGHE has recently added to their version of ODK Collect.

JHCCGHE was an early adopter of ODK and had to make this decision early on. *"At the beginning sending data to a server was not implemented, and later it only synchronized with Google App Engine, so we built our own [server] based on PHP."* ODK's use of standard communication interfaces allowed them to easily connect their server to client devices running ODK Collect. This server is itself available to the larger community as an alternative module to Aggregate.

E. Academic Model for the Prevention and Treatment of HIV

The Academic Model for the Prevention and Treatment of HIV (AMPATH) [45] is the one of the largest HIV treatment programs in sub-Saharan Africa and is Kenya's most comprehensive initiative to combat the virus.

ODK is being used for the USAID-AMPATH home-based HIV counseling and testing (HCT) program. This program aims to reach two million individuals in the program's catchment area, with the following specific goals: (a) counsel and test all eligible individuals for HIV; (b) identify pregnant women not in ante-natal care; (c) identify orphaned and vulnerable children; (d) determine immunization status for children; (e) identify people at high risk for TB infection and collect sputum samples; and (f) refer individuals to appropriate follow-ups. These goals are managed primarily through OpenMRS [46], an open-source medical record system.

AMPATH had extensive previous experience with data collection. *"Before using ODK, we had created data collection functionality on Palm TX devices using Pendragon Forms. GPS information was collected using eTrex devices, and these devices were connected via cable to the Palm device."*

In replacing their existing system, an important consideration was the ability to transfer data collected during HCT into their medical records system. ODK supported adding that functionality whereas Pendragon Forms did not. The move to ODK also added barcode scanning of AMPATH patient ID cards to minimize error (shown in Figure 9) and solved problems with the *"clumsy and unreliable"* external GPS system. Training was thus *"much easier"* on ODK and the data captured suggests fewer data entry errors. AMPATH attributed this to the logic support built into the ODK implementation.

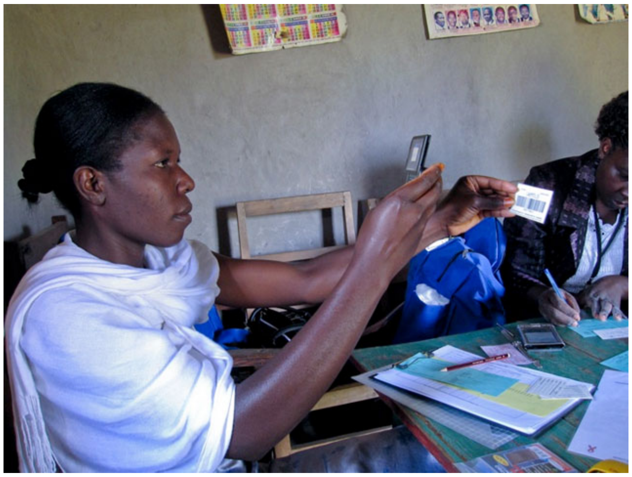


Fig. 9. A HIV counselor from AMPATH in Kenya scans a patient's barcode in ODK Collect before sending data to the OpenMRS medical record system.

Overall, AMPATH found ODK more cost-effective than Pendragon Forms for both software and hardware requirements. They have added their own user interface elements to speed up data entry and are moving to use ODK to help manage workflow using alerts and reminders. They note that *"this is dependent on other infrastructure outside ODK...the possibilities are however limitless."*

F. Summary

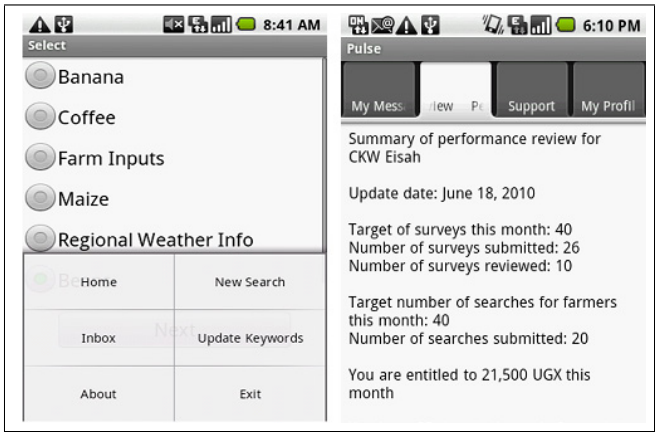
Based on the feedback from four implementers, ODK has demonstrated it can enhance information services in a variety of low-resource environments. Our users report ODK is easier to use, more capable, more cost-effective, and more accurate than the alternatives they considered.

For HRC, ODK's cost of deployment and short timelines drove much of their decision. For D-Tree, the impetus was backwards compatibility paired with ease of use and development. AMPATH needed to integrate with a medical record system and intuitive user interface was key, whereas JHCCGHE focused on integration with a larger set of server and mobile tools.

In all four cases, many of the design goals of ODK enabled each organization to use or modify ODK Collect and submit data to their preferred server solution.

Beyond the implementers evaluated in this paper, there are examples of others building on our design choices by using Aggregate as a gateway to other web servers [47], using their application designers to build services for ODK [48], adding web-based ODK-compatible clients [49], and creating on-phone analysis tools (shown in Figure 10).

Although ODK can be the solution in some scenarios, there are tradeoffs that must be considered. Both AMPATH and researchers at Columbia University using ODK noted problems with short battery life, occasionally slow GPS response, and lack of non-Western languages within the Android OS user interface itself [51].



Summary of performance review for CKW Eisah

Update date: June 18, 2010

Target of surveys this month: 40
Number of surveys submitted: 26
Number of surveys reviewed: 10

Target number of searches for farmers this month: 40
Number of searches submitted: 20

You are entitled to 21,500 UGX this month

Fig. 10. CKW Search and CKW Pulse from Grameen AppLab in Uganda integrates with ODK and Salesforce.com to provide real-time monitoring and evaluation [50].

There are also financial constraints that could be problematic. For example, the relatively higher cost of smart phones ($350 vs. $100) for somewhat less capable phones has been cited as a reason organizations are wary to begin using ODK. However, many ODK implementers have anecdotally described cost savings from less training and transportation as amply covering the difference in capital costs. We are currently running ongoing studies with these organizations to further understand the underlying tradeoffs.

VII. Conclusion

Open Data Kit provides organizations with a new way to build information services for developing regions. The modular, extensible and open-source design allows users to pick and choose the tools best suited for their specific deployments. We have detailed how ODK differs from previous work, described the current set of tools, and presented evaluations of four ongoing deployments. Finally, we have discussed how design decisions made in the system architecture of ODK enable a large and varied set of applications for developing regions.

Open Data Kit tools are freely available for download at http://opendatakit.org.

Acknowledgment

The authors thank the members of the Open Data Kit community, especially Academic Model for the Prevention and Treatment of HIV (AMPATH), D-Tree International (D-Tree), Berkeley Human Rights Center (HRC), Johns Hopkins Center for Clinical Global Health Education (JHCCGHE), Dimagi and the University of Washington's Change group for their ongoing support and use of our tools.

References

[1] International Telecommunication Union, "ICT statistics," July 2010. [Online]. Available: http://itu.int/ITU-D/ict/statistics/

[2] K. Toyama, "Ten myths of ICT4D," November 2009, http://research.microsoft.com/en-us/um/people/toyama/talks/.

[3] E. Brewer, M. Demmer, M. Ho, R. Honicky, J. Pal, M. Plauch, and S. Surana, "The challenges of technology research for developing regions," in *IEEE Pervasive Computing*, vol. 5. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 15–23.

[4] Inveneo, "ICT project and sustainability primer," July 2010. [Online]. Available: http://inveneo.org/download/Inveneo_ICT-Sustainability_Primer0809.pdf

[5] "Open Data Kit," July 2010. [Online]. Available: http://opendatakit.org/

[6] T. Groves, “SatelLife: Getting relevant information to the developing world,” in *BMJ*, 1996.

[7] D. Forster, R. H. Behrens, H. Campbell, , and P. Byass, “Evaluation of a computerized field data collection system for health surveys,” in *Bulletin of the World Health Organization*, 1991.

[8] “Epihandy,” July 2010. [Online]. Available: http://epihandy.org/

[9] “CyberTracker,” July 2010. [Online]. Available: http://cybertracker.co.za/

[10] “Pendragon Forms,” July 2010. [Online]. Available: http://pendragonsoftware.com/

[11] “Pendragon Forms case studies.” [Online]. Available: http://pendragonsoftware.com/casestudy/

[12] J. Blaya and H. S. Fraser, “Development, implementation and preliminary study of a PDA-based bacteriology collection system,” in *American Medical Informatics Association Annual Symposium*, 2006.

[13] K. Shirima, O. Mukasa, J. A. Schellenberg, F. Manzi, D. John, A. Mushi, M. Mrisho, M. Tanner, H. Mshinda, and D. Schellenberg, “The use of personal digital assistants for data entry at the point of collection in a large household survey in southern Tanzania,” in *Emerging Themes in Epidemiology*, vol. 4, June 2007, pp. 5+. [Online]. Available: http://dx.doi.org/10.1186/1742-7622-4-5

[14] “FrontlineSMS,” July 2010. [Online]. Available: http://frontlinesms.com/

[15] “EpiSurveyor,” July 2010. [Online]. Available: http://datadyne.org/

[16] “CommCare,” July 2010. [Online]. Available: http://dimagi.com/commcare/

[17] “JavaRosa,” July 2010. [Online]. Available: http://bitbucket.org/javarosa/

[18] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay, “MyExperience: A system for in situ tracing and capturing of user feedback on mobile phones,” in *MobiSys ’07: Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*. New York, NY, USA: ACM, 2007, pp. 57–70.

[19] T. S. Parikh, P. Javid, S. K., K. Ghosh, and K. Toyama, “Mobile phones and paper documents: Evaluating a new approach for capturing microfinance data in rural India,” in *CHI ’06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA: ACM, 2006, pp. 551–560.

[20] Y. Schwartzman and T. S. Parikh, “Using CAM-equipped mobile phones for procurement and quality control at a rural coffee cooperative,” in *MobEA V: Mobile Web in the Developing World*, May 2007.

[21] K. Chen, H. Chen, N. Conway, J. M. Hellerstein, and T. S. Parikh, “Usher: Improving data quality with dynamic forms,” in *International Conference on Data Engineering (ICDE)*, 2010.

[22] T. S. Parikh, “Designing an architecture for delivering mobile information services to the rural developing world,” Ph.D. dissertation, University of Washington, 2007.

[23] “XForms 1.1,” July 2010. [Online]. Available: http://w3.org/TR/xforms/

[24] “OpenRosa Consortium,” July 2010. [Online]. Available: http://openrosa.org/

[25] “Voice extensible markup language,” July 2010. [Online]. Available: http://w3.org/TR/voicexml20/

[26] P. Meier and J. Leaning, “Applying technology to crisis mapping and early warning in humanitarian settings,” September 2009. [Online]. Available: http://hhi.harvard.edu/images/resources/reports/whitepapers_f_11.pdf

[27] 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 ’08: Proceeding of the 26th Annual SIGCHI Conference on Human Factors in Computing systems*. New York, NY, USA: ACM, 2008, pp. 753–762.

[28] R. M. Groves, F. J. F. Jr., M. P. Couper, J. M. Lepkowski, E. Singer, and R. Tourangeau, *Survey Methodology*. Wiley-Interscience, 2004.

[29] S. R. Sterling, J. O’Brien, and J. K. Bennett, “Advancement through interactive radio,” in *Information Systems Frontiers*, vol. 11, no. 2. Hingham, MA, USA: Kluwer Academic Publishers, 2009, pp. 145–154.

[30] R. Veeraraghavan, N. Yasodhar, and K. Toyama, “Warana unwired: Replacing PCs with mobile phones in a rural sugarcane cooperative,” in *International Conference on Information and Communication Technologies and Development*. IEEE/ACM, 2007.

[31] “SIM toolkit,” July 2010. [Online]. Available: http://www.bladox.cz/devel-docs/gen_stk.html

[32] J. Sherwani, S. Palijo, S. Mirza, T. Ahmed, N. Ali, and R. Rosenfeld, “Speech vs. touch-tone: Telephony interfaces for information access by low literate users,” in *International Conference on Information and Communication Technologies and Development*. IEEE/ACM, April 2009.

[33] N. Patel, D. Chittamuru, A. Jain, P. Dave, and T. S. Parikh, “Avaaj Otalo - A field study of an interactive voice forum for small farmers in rural India,” in *CHI 2010: Proceedings of the 27th International Conference on Human Factors in Computing ystems*. ACM, 2010.

[34] W. H. Curioso, B. T. Karras, P. E. Campos, C. Buenda, K. K. Holmes, and A. M. Kimball, “Design and implementation of Cell-PREVEN: A real-time surveillance system for adverse events using cell phones in Peru.” in *American Medical Informatics Association Annual Symposium*, 2005.

[35] S. Patnaik, E. Brunskill, and W. Thies, “Evaluating the accuracy of data collection on mobile phones: A study of forms, SMS, and voice,” in *International Conference on Information and Communication Technologies and Development*. IEEE/ACM, April 2009.

[36] “RapidSMS,” July 2010. [Online]. Available: http://rapidsms.org/

[37] “Vodacom Tanzania,” July 2010. [Online]. Available: http://vodacom.co.tz

[38] “Human Rights Center,” July 2010. [Online]. Available: http://hrc.berkeley.edu/

[39] “KoBo,” July 2010. [Online]. Available: http://sites.google.com/site/kobohrc/

[40] “D-Tree International,” July 2010. [Online]. Available: http://www.www.d-tree.org/

[41] “Integrated management of childhood illness,” July 2010. [Online]. Available: http://who.int/child_adolescent_health/topics/prevention_care/child/imci/

[42] “Android OpenMRS,” July 2010. [Online]. Available: https://bitbucket.org/routen/androidopenmrs/overview/

[43] “eMOCHA,” July 2010. [Online]. Available: http://emocha.org/

[44] “Johns Hopkins Center for Clinical Global Health Education,” July 2010. [Online]. Available: http://www.ccghe.jhmi.edu/ccg/index.asp

[45] “IU-Kenya Partnership/AMPATH,” July 2010. [Online]. Available: http://www.iukenya.org/hiv.aids.html

[46] “OpenMRS,” July 2010. [Online]. Available: http://openmrs.org/

[47] “Rhiza Insight Feature Overview: Open Data Kit Integration,” July 2010. [Online]. Available: http://www.youtube.com/watch?v=mVxRoKZSyOg/

[48] “RapidSMS XForms,” July 2010. [Online]. Available: http://nyaruka.github.com/rapidsms-xforms-builder/

[49] “PurcForms,” July 2010. [Online]. Available: http://code.google.com/p/purcforms/

[50] “Grameen Community Knowledge Worker,” July 2010. [Online]. Available: http://www.grameenfoundation.applab.org/ckw/section/index/

[51] F. JeffreyCoker, M. Basinger, and V. Modi, “Open Data Kit: Implications for the Use of Smartphone Software Technology for Questionnaire Studies in International Development,” March 2010. [Online]. Available: http://modi.mech.columbia.edu/2010/04/open-data-kit/