ODK Tables: Data Organization and Information Services on a Smartphone

YoonSung Hong, Hilary K. Worden, Gaetano Borriello
Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195-2350 [USA]
{hys235, hkworden, gaetano}@cs.washington.edu

ABSTRACT

Many information services require the transfer of only small amounts of information between a client and server. Furthermore, their deployment often requires an ecosystem of cloud services rarely present in developing world contexts. ODK Tables (a component of the Open Data Kit) provides a way of organizing data into database tables hosted directly by a smartphone. Clients can make new entries into the tables (under an extensible access control model) and make queries of existing information. ODK Tables supports SMS-based interactions and allows import/export of tables to other storage whether in the cloud or on another local computing device. The objective of ODK Tables is to lower barriers experienced by entrepreneurs or other information providers in the developing world to field their own information services. This paper describes ODK Tables' capabilities, user interface, performance characteristics, and some example use cases.

Categories and Subject Descriptors

H.4 [**Information Systems**]: Information Systems Applications – *groupware, spreadsheets.*

General Terms

Management, Design, Human Factors.

Keywords

Mobile databases, mobile phones, SMS, spreadsheets, data tables.

1. INTRODUCTION

Database tables and spreadsheets are a useful way of organizing information. These models have been very popular on personal computing platforms and been adapted to a wide range of applications through general-purpose tools such as Excel and Access. In the developing world, the desktop personal computer has not become the dominant platform; instead, the most commonly available computing platform is the mobile phone. Unfortunately, tools for organizing data are not as mature on phone platforms as on PCs. Furthermore, while data to and from PCs is transmitted over the Internet. In the developing world, the great majority of users use SMS for basic data communication.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
NSDR'11, June 28, 2011, Bethesda, Maryland, USA.
Copyright 2011 ACM 978-1-4503-0739-0/11/06...$10.00.

ODK Tables seeks to bridge this difference by providing a way to store data into tables directly on a smartphone and provide a user interface to those tables that displays them efficiently on the phone's smaller screen while allowing the user to easily search and modify data. In addition, ODK Tables uses SMS among the ways to enter data and make queries of what is already in the tables. In this way, users of basic phones can still interact with services that are actually hosted on smartphones.

ODK Tables has been implemented on Android smartphones and is linked to the mobile data collection tools of Open Data Kit (ODK) [1]. ODK Tables has two principal facets. The first is its user interface for SQLite databases. Users can import large amounts of data to build a new table from a standard spreadsheet file (.csv or .xls) and can also export data to these same file formats. The ODK Tables' user interface allows users to fix columns on the screen while scrolling others (so as to easily line up different data columns) and allows for compression of related data into "collections" (explained further below). We believe that these are two features novel to ODK Tables and that we have not seen in the other phone-based spreadsheet applications we surveyed. The other facet of ODK Tables is the use of SMS messages to query the tables as well as inserting new rows into the tables. In addition, ODK Tables facilitates the composition of SMS messages from the server (smartphone) to clients (using basic phones).

All the usage models of ODK Tables we investigated share the need to organize data into a database schema (or table) and then provide access to that data to a large number of users possibly using basic SMS phones. This is a common paradigm for many information services we have seen deployed in developing world contexts. In other words, making it possible for the services to be accessed from the cheapest and most common phones with the server running on a specialized device, in this case, a smartphone rather than a web server. In addition, we connect ODK Tables to ODK Collect (the XForm-based data collection tool of the Open Data Kit) so that new data can be entered in a structured form-based interaction and we connect ODK Tables to ODK Aggregate (the database storage component of the Open Data Kit) so that data can be submitted to cloud servers for backup and/or forwarding to other services including synchronization with other phones. We are refining these interoperability interfaces so that other specialized phone apps can be easily developed to modify the contents of the SQLite tables and be managed through ODK Tables.

2. ODK TABLES

Database tables are the underlying abstraction for ODK Tables. There is a user interface for browsing and manipulating the tables that is optimized for the small screens of mobile phones.

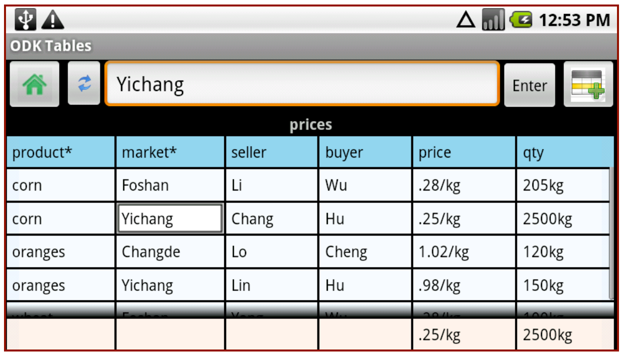
Properties of the tables and table columns provide aggregate functions. Finally, communications to the phone can cause new data to be inserted into the tables and/or can search the tables. The user interface provides a way for the smartphone user to answer an incoming message and/or generate a new one.

2.1 Table Presentation and Interface

ODK Tables render data stored in SQLite databases on the Android phone. A new table can be created from scratch by adding rows and columns manually. However, it is often easier to create a spreadsheet on a desktop computer to initially populate a database. ODK Tables has the ability to read .csv and .xls files and populate a table automatically. In addition, it can export existing tables into the same formats for transfer back to a PC. We are also developing direct import/export from cloud servers that will be used to backup data and synchronize tables across multiple phones.

A typical rendering of an ODK Table is shown in Figure 1. Given the relatively small phone screens, we focused on exploiting Android's interface design standards such as long-presses to deliver specialized functions for on-screen elements and contextual menus to provide access to general functions.

A table is displayed with a top bar for basic functions and two fixed rows across the top and bottom of the table. The top bar provides basic browser-like functions: a home button to return the UI to the master view of the table (more on other views below); a refresh button to re-render the table, if necessary; an edit box in which to enter new cell contents or search term; an enter button to cause action on the item in the edit box; and an add new row button that brings up a dialog that asks for a value for each column on a new row. The header and footer rows of the table (shown in blue and gray, respectively) only scroll horizontally and are, by default, always present on the screen (although the user can choose to hide the footer row) while the table contents, the rows between the header and footer, can be scrolled (by swiping) both vertically and horizontally. Obviously, a table can contain many more rows than can be viewed at one time.



product*	market*	seller	buyer	price	qty
corn	Foshan	Li	Wu	.28/kg	205kg
corn	Yichang	Chang	Hu	.25/kg	2500kg
oranges	Changde	Lo	Cheng	1.02/kg	120kg
oranges	Yichang	Lin	Hu	.98/kg	150kg
				.25/kg	2500kg

Figure 1. A typical ODK Table showing 4 rows between header and footer rows. The bar across the top includes global table controls such as "home" and "add row".

The top row of the table is used to represent column names. The bottom row provides summary information for all the other rows. This footer's cells can be independently set to summarize statistical information such as average, count, max, min and mode of each column. Properties for each cell in this row can be set through a menu obtained from a long-press.

To make good use of limited screen area, any column can be fixed to the left edge of the screen so that horizontal scrolling does not affect its position. This allows the user to compare values in any two columns easily.

Because ODK Tables is built on top of an SQLite database, internally generated SQL SELECT statements are used to populate the table. Users can arrange the order of columns, as well as set indexing and order-by columns. Indexing is equivalent to GROUP BY statements in SQL and multiple columns can be set to be index columns. Indexing forces tables to display only distinct entries within the indexed columns. Multiple rows with identical values in those columns are represented by just one row. For instance, if the "product" column is set to be an index column in the table in Figure 1, the table will display only one entry for each of the distinct products. The hidden rows can be viewed by selecting the row with the "product" value of interest and pivoting the table on that row, thereby changing the display to only show that collection of rows, namely, those rows with the same value in that column. The home button is used to return to the main view of the table. Users can also order the table contents with respect to a specific column. For instance, if the "unit" column is set to be an order-by column, the table will order entries from highest to lowest unit quantity.

Users can apply both indexing and order-by simultaneously to columns to generate tables for various purposes. In addition, multiple columns can be selected as index or order-by columns. For example, by selecting both "product" and "market" as index columns; the table shows only distinct rows for every distinct combination of those two column values. However, a single column cannot be set to be both an index and order-by column due to SQLite restrictions. Figure 1 shows an example of indexing and order-by used together. The table is indexed by the "product" and "market" columns (marked by "*") and ordered by the "qty" column. It follows that Li and Chang are sellers of corn at two different markets (Foshan and Yichang), and Chang sells the largest quantity unit of corn at 0.25 units per kilogram in the Yichang city market. The footer mode is set to min and max for the price and quantity columns, respectively.

ODK Tables offers an easy way to search over the database and generates dynamic table results. For instance, indexing the "product" column in the prices table example generates a table with only one row for corn. Users can search all corn sales by long-click on the cell where product equals to corn then select "View All Like This" in the option menu. Similarly, users can search for products in a specific market. Currently, the search does not support inequality conditions such as price >= 0.28/kg and general term search over all columns. The search must be column-specific and equality-based. We are investigating how to extend the current search interface without sacrificing simplicity and the ability to quickly pivot the table and scroll it to the row(s) of interest.

Menu options differ in the header and footer rows from those in the main content rows of the table. A long-press on a header cell brings up options to "Select This Column", "Make Index Column", "Make Order-By Column", "Column Properties", and "Set Column Width". A long-press on a footer cell lets users choose one of the statistical functions to apply to the column such as average, count (of non-blank entries), min, max and mode. Options for content row cells include "View All Like This" (used in concert with index column specifications) and "Delete This Row".

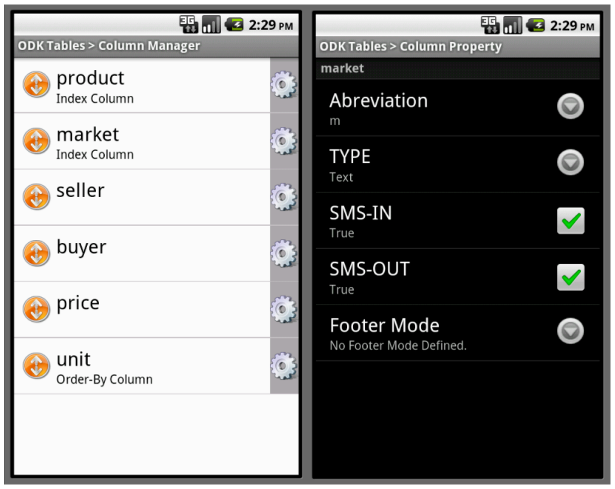
ODK Tables contextual menus (available through the menu button on the phone) allows user to select table-scale functions such as: table manager (a list-based interface where entire tables can be added, edited, or removed and their global paremeters such as fonts and access controls can be set and/or adjusted), column manager (allowing for the re-arrangement of columns along with insertion, deletion, and renaming), graphing functions (for mapping table data onto various types of graphs and explained in more detail below), and import/export to .csv and .xls files.

We conducted a basic usability study to test the ODK Tables user interface. We presented users with a scenario and a step-by-step tutorial to give users approximate instructions for how to set up a table for that specific scenario. After users completed the tutorial, they were given a set of new scenarios to test if users could generalize the knowledge they had gained from the tutorial to solve new problems. Of the seven participants, not one missed any problem in the new problem set. We observed that users could organize tables with indexing and searching without difficulty. In the survey, users gave an average rating of 4.14 out of 5 for overall application usability. It should be noted that our test subjects were all undergraduate students at UW with some computing background but with many not having experience with smartphones. However, we feel that a basic level of computing literacy is likely to be common in the younger generation even in developing countries.

2.2 Properties and Communication

ODK Tables is more closely tied to databases than spreadsheets. Properties are a fundamental property of columns rather than cells. All cells in a column are treated the same way and mathematical/statistical formulas can only be applied to entire columns, not individual cells as in a spreadsheet. In addition, ODK Tables extends the traditional column properties of databases with extensions to support communication.

Currently, ODK Tables supports the ability of incoming SMS messages to add new rows to a table or make queries of table entries. Column properties that deal with SMS communication include: abbreviation (a short string to be used in referring to a column); SMS-IN (a check box indicating this is a column that can be filled by the data from an incoming SMS); and, SMS-OUT (a check box indicating this is a column whose data is to be inserted into an outgoing SMS). Although, we are currently limited to SMS, it is easy to imagine extending the approach to incoming/outgoing e-mail messages as well.


2:29 PM
ODK Tables > Column Manager
product
Index Column
market
Index Column
seller
buyer
price
unit
Order-By Column

2:29 PM
ODK Tables > Column Property
market
Abreviation
m
TYPE
Text
SMS-IN
True
SMS-OUT
True
Footer Mode
No Footer Mode Defined.

Figure 2. Column ordering and property screens.

Users can manage columns and their properties through the column manager. When a user clicks on a specific column on the list in the column manager, it brings up the column property interface for that column. Currently, there are five fields including abbreviation, type, SMS-IN, SMS-OUT, and footer mode.

Abbreviations for column names are used to make better use of the 160-character limit for SMS messages. Currently, the following syntax is used to add a new row via an incoming SMS message (sent by a remote phone to the phone running ODK Tables):

```
@Table_Name {+Column_Name Value}*
```

A typical insertion message is:

```
@Prices +pro corn +m Foshan +s Li
    +b Wu +pri .28/kg +u 205kg
```

A typical query message is:

```
@Prices ?s =pro corn =m Foshan
```

which will return the names of all corn sellers in Foshan market to the querying phone. Without the "?" operator, it will return all columns with SMS-OUT property checked. A response to such a query, sent from ODK Tables to the remote phone, takes the form:

```
pro=corn s=Li pri=.28/kg u=205kg
```

If there is more than one entry to be returned, each entry is separated by a semicolon. ODK Tables tries to fit the entire response into a single message, but will extend the reply to multiple messages, if needed. Supported types are text, numeric values, date, phone number, and date/time range. In the future, we will add image, voice, and more as we use communication methods that can include these data types such as e-mail and MMS. In addition, we want to allow for more flexible syntax that is closer to how users compose unstructured SMS messages. However, we leave this for future work.

Users of ODK Tables can also send row data in useful ways. For each table, an outgoing message format can be defined. The formatted messages work much like "printf" in C; column names are placeholders in the message templates. When the user selects a row and a format, the template is filled in with data from that row and sent to the specified phone number. A typical message template could be:

```
    Buy %pro% from %s% at %pri% for %u%
```

which generates the SMS message "Buy corn from Li at .28/kg for 205kg" to a remote phone. It might be used to inform an agent in the market to take the specified action.

2.3 Access Control

Any information service application requires a way to limit access appropriately. ODK Tables makes use of its own tables to set up lists for read/write access. Tables can reference other tables that serve as access groups for write access (the ability to add new rows) and/or read access (to make queries) permissions. Tables without a reference to an access group table can only be accessed locally on the phone and are thus considered potential access group tables themselves.

Users can create as many access group tables as they need. Generating an access group follows the same steps as creating a regular table with the most likely scenario being an import from a previously prepared spreadsheet. However, to be established as an access group table, the table must have two required columns: name and phone number. Phone numbers from SMS headers are uniquely identified and indicate the phone number from which the SMS originated (as guaranteed by the cell service provider), thus phone numbers qualify as reliable unique identifiers for the access group. Name is a human readable identifier for the corresponding phone number. Additionally, users can include a password column to enhance security, for example, to provide a basic level of protection against a borrowed or stolen phone being used to insert bad data into a table. Each user in the access group can be assigned their own password or a single group password can be copied into every row. Access group tables can be applied as either read or write access control lists for regular tables but they themselves are not accessible via SMS or other remote messaging.

When query or insertion messages are received, ODK Tables first checks if the phone number is within the corresponding access group. And, if a password column exists in the access group table, that it matches with the password included in the message. Only if the above conditions are met, is querying or insertion permitted. Using the above access group approach, ODK Tables can limit who can access which tables with permissions on a phone number or password basis. For e-mail, this process may be enriched with the possibility of including signed certificates in the message.

2.4 Graphing

ODK Tables includes a number of graphical display options. For both the main table and collection tables (recall that collections are sub-tables that include only rows with the same value for the index columns), users can set a default graph type and columns to be used when "Graph" is selected from the contextual menu. For example, a user might set the main graph to be a pie chart broken into sections for each value in Column A, while the graph for the collection tables are line graphs with Column B used for the x-axis and Column C for the y-axis. The main graph will use all of the data in the table, but collection graphs will only use the data from the collection being viewed.

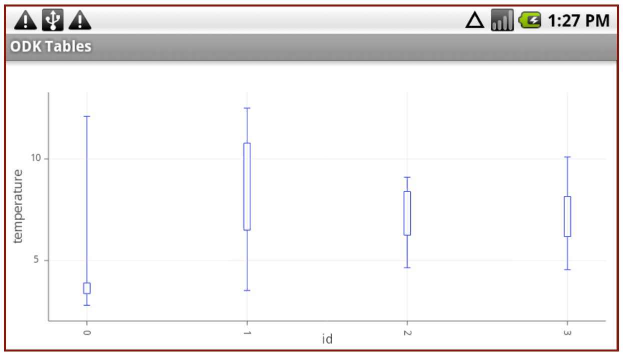


Figure 3. A whisker plot of temperature data showing min/max and 25th and 75th percentiles for each fridge.

One type of graph provided is a box-stem plot, which allows users to group rows by the value in a specified column and see an overview of their data. For example, when used with FoneAstra (discussed later in the section on use cases) to monitor refrigerator temperatures, a box-stem plot is used to get a quick overview of the operating temperature range of multiple refrigerators. It is easy to determine which may have been outside the acceptable temperature range and for the percentage of the time they were out of range. Shown below is a box-stem plot using data from a FoneAstra deployment, with refrigerator ID used for the x-axis and temperature for the y-axis as shown in Figure 3.

ODK Tables also provides line graphs and pie charts. A line graph for the collection views with the FoneAstra data, for instance, allows the table owner to view the temperature history of a single refrigerator suspected of malfunctioning (by graphics only the collection of rows that contain temperatures from that one refrigerator) as shown in Figure 4.

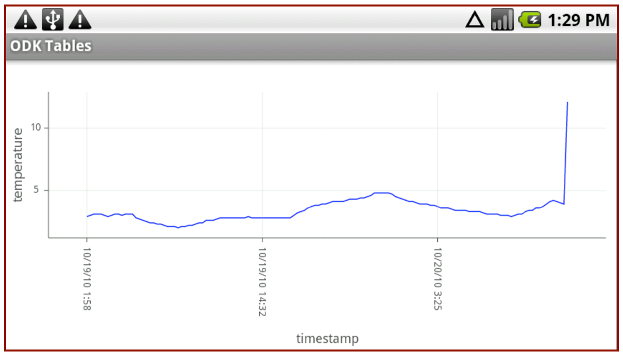


Figure 4. A line graph of the temperature variation for one particular refrigerator.

Apart from graphs for numeric data, ODK Tables also provides calendar views and maps. The map view uses the Google Maps API, which, although not available offline, provides complete maps of many areas when an Internet connection is available. The calendar view shows a day's schedule, and is useful for someone who uses ODK Tables to manage appointments.

3. Performance

We measured performance of loading a table from SQLite for display and generating line graphs. Our test databases each had one table, with the first column holding an integer (between 1 and 10, inclusive), the second a string representing a date and time, and the third a number between 0 and 50 rounded to the first digit after the decimal. The tables were indexed on the first column, and rows were evenly distributed among the 10 possible values in that column (with the exception of the test database that had only one row).

In testing table loading, we timed the display of a collection table (all rows where the value in the first column was 1; a tenth of the rows in the database). Similarly, the line graphs were generated from a collection view, so the number of data points was a tenth of the rows in the database. All of our performance tests were run on a Droid phone with Android version 2.0.1. Times shown are the averages of results from five trials.

As can be seen, with relatively large amounts of data in the database, growth is generally linear in the number of rows being displayed.

Most of the time seems to be spent displaying the table, rather than accessing the database. An additional test with 10000 rows distributed into 100 collections took an average of 2481.4ms to display, making it much closer to the fourth test above (which also had 100 rows per collection, but only 1000 rows in the database) than the fifth test (which also had 10000 rows in the database, but only 1000 per collection). This suggests that users could keep the table display time manageable by using indexes to reduce the number of rows displayed at once, even if their database contains a fairly large amount of data.

Table 1. Performance of ODK Tables for loading and graphing of different size tables.

Rows in Database	Rows Displayed	Collection Table Load	Line Graph Load
1	1	454.4ms	242.2ms
10	1	410.6ms	207.4ms
100	10	730.4ms	225.2ms
1000	100	2,426.8ms	437.8ms
10000	1000	21,563.8ms	858.8ms
20000	2000	48,258.2ms	1,671.4ms

4. Use Cases

Automated Data Collection

FoneAstra is a tool for deploying sensors by attaching them to low-cost mobile phones so data can be collected via SMS [2]. Last November, ODK Tables was used in connection with a FoneAstra deployment in which vaccine storage units were monitored to ensure temperature guidelines were followed. Data acquired by the FoneAstra temperature sensors was sent automatically to an Android phone running ODK Tables, which added the refrigerator's ID number, current temperature, and the current time to a new row in a table. This permitted supervisors to check the status of all storage units from their own phones, as well as store and view historical information.

Participatory Sensing

ODK Tables also allows users to collect data from large numbers of people through their own basic mobile phones [3]. Since row addition simply requires an SMS, an organization wanting to collect data from citizens would simply publish a message format and phone number (e.g., on a placard, flyer, or advertisement). ODK Tables could thus be used for conducting low-cost surveys as well as integrating data on particular resources (e.g., in a disaster response scenario or to monitor water availability).

Microfinance

Microfinance organizations must track a large number of small transactions that are handled by multiple agents [4]. ODK Tables would permit information about loans and payments to be collected on a supervisor's mobile phone, and would only require agents to carry a simple, low-cost phone. Our access control model would ensure that only those authorized to do so could add rows or access information from the database. This could include read access for customers to check their own accounts and transactions.

Basic Search

ODK Tables allows users to make databases of simple information available via SMS. For example, a table populated with weather forecasts could be queried based on location or date (or both), or a table of bus schedules could be queried by route and time. ODK Tables also allows queries with conditions such as "time < thu" (to find rows with a time value earlier than the upcoming Thursday) or "time > now" (to find only rows with time values in the future, which table owners may often want to set as a default).

Market Prices

One particular instance where both basic search and basic data collection could be useful together is the exchange of price information [5]. For example, farmers could send information about their crops to a phone with ODK Tables, and buyers could query the database to find the goods they are seeking. Data rows might include type of crop, quantity, price, and location, permitting buyers to limit results to their area or search based on asking price. Figure 1 is an example of such a table.

Scheduling

ODK Tables includes several features designed for use in managing schedules and setting appointments. Firstly, there is a "date range" data type, which is composed of a pair of date-times representing a time interval. Secondly, if a table has at least one date range column, it can marked as the "availability" column. If an availability column is used, queries return gaps in the schedule as results, rather than the rows themselves. For example, if a work day is defined as 8am to 6pm and rows already exist with date range values from 9am to 11am and from 3pm to 4pm, a response to a query would include the time from 8am to 9am, from 11am to 3pm, and 4pm to 6pm. Similarly, a table owner can set the availability column to control row addition to prevent the addition of a row that overlaps with an existing row. These appointment-setting options could be useful for a variety of small businesses, as well as other organizations like health clinics, to allow clients to query/set/confirm their appointments.

5. Related Work

There have been a few efforts that seek to realize similar capabilities of ODK Tables. Most efforts rely on a server in the infrastructure to process messages, thereby raising the bar to deployment. FrontlineSMS [6], RapidSMS [7], and UjU [8] are all of this type. FrontlineSMS has shown the viability of many use cases for SMS data. RapidAndroid [9] also uses the smartphone as a server but does not provide a general-purpose user interface to database tables. Finally, there are many applications for smartphones that implement traditional spreadsheets (i.e., Excel like functionality where formulas can be attached to individual cells). We specifically chose to use "tables" in the name of ODK Tables to highlight the distinction from spreadsheets. ODK Tables' formulas are all column-based rather than cell-based. Finally, there are actions that are user initiated by selecting rows and columns rather than by placing the actions within a cell.

6. Conclusion and Future Work

ODK Tables introduces a graphical interface to databases stored on smartphones. In addition to providing ways to browsing, re-ordering, indexing, and modifying the tables, it provides communication over SMS. SMS messages can insert new data into the table and query the table contents under a flexible access control mechanism. SMS message can be easily generated from the table using templates.

To assess ODK Tables' ease-of-use, we asked 7 users to follow a tutorial to build their first ODK Tables application. Initial results show that users are easily able to generalize the ODK Tables model to a variety of applications after the initial tutorial.

We are now investigating improving ODK Tables SMS syntax, adding synchronization with cloud storage as well as other smartphones, and more fully integrating with the rest of the ODK tools. We are engaged in discussions for possible deployments in developing world contexts.

7. Acknowledgments

Julia Chu, Thienan Le, and Henry Qin participated in the development of ODK Tables in CSE490D/HCDE419 at UW. We also thank our usability study participants.

8. REFERENCES

[1] C. Hartung, Y. Anokwa, W. Brunette, A. Lerer, C. Tseng, G. Borriello. Open Data Kit: Building Information Services for Developing Regions. 4th IEEE/ACM International Conference on ICTD, London, England, Dec 2010.

[2] R. Chaudhri, G. Borriello, R. Andesron, S. McGuire, E. O'Rourke. FoneAstra: Enabling Remote Monitoring of Vaccine Cold-Chains Using Commodity Mobile Phones, ACM 1st Annual Symposium on Computing for Development (DEV), London, England, Dec 2010.

[3] D. Estrin. Participatory Sensing: Applications and Architecture, 8th International Conference on Mobile Systems, Applications and Services (MobiSys'10), San Francisco, CA Jun 2010.

[4] T. Parikh, et al. Mobile Phones and Paper Documents: Evaluating a New Approach for Capturing Microfinance Data in Rural India, ACM Conference on Computer-Human Interaction (CHI), Montreal, Canada, Apr 2006.

[5] R. Veeraaghavan, N. Yasodhar, K. Toyama. Warana Unwired: Replacing PCs with Mobile Phones in a Rural Sugarcane Cooperative, 2nd International Conference on ICTD, Bangalore, 2007.

[6] Frontline SMS, http://www.frontlinesms.com/.

[7] RapidSMS, http://www.rapidsms.org/.

[8] L. Wei-Chih et al. UjU: SMS-based applications made easy, ACM 1st Annual Symposium on Computing for Development (DEV), London, England, Dec 2010.

[9] RapidAndroid, http://rapidandroid.org/.