DUCES: A Framework for Characterizing and Simplifying Mobile Deployments in Low-Resource Settings

Samuel R Sudar
Unversity of Washington
Seattle, USA
sudars@cs.uw.edu

Richard Anderson
Unversity of Washington
Seattle, USA
anderson@cs.uw.edu

ABSTRACT

Mobile devices are increasingly being used in data-focused workflows in low-resource settings. These deployments are frequently orchestrated by organizations with limited technical capacity, making fundamental architectural decisions difficult. We present DUCES, a framework for characterizing mobile deployments along five axes of design. DUCES allows organizations to better understand deployment requirements and simplify decisions regarding deployment architectures. It focuses on the workflow's **D**ata flow, **U**ser interface, **C**onnectivity model, **E**dit mode, and **S**erver requirements. We discuss five case studies of data-focused mobile deployments and evaluate them using the DUCES framework. We conclude by discussing how the DUCES framework can be used as a lens by organizations and researchers to understand and simplify mobile deployments.

Keywords

ICTD; mobile devices; application, app design

Categories and Subject Descriptors

H.4.0 [**Information Systems Applications**]: General; D.2.1 [**Software**]: Software EngineeringRequirements/Specifications[methodologies]

1. INTRODUCTION

Mobile devices are deployed in many low-resource settings for data-focused applications. Creating these applications is nontrivial, consuming considerable time and resources [8]. Successful custom-built deployments can require years of iterative refinements to work out stable technical architectures [11]. Smaller scale deployments often do not involve developers, making reasoning about technology difficult. We present the DUCES framework, which can be used to deepen understanding of a deployment and its requirements as well as to highlight which requirements are most challenging technically. If these can be altered in a way that simplifies the architecture, the deployment will become more sustainable without external expertise.

ACM DEV 2015, December 1–2, 2015, London, United Kingdom.
 ISBN 978-1-4503-3490-7/15/12 . . . $15.00.
DOI: http://dx.doi.org/10.1145/2830629.2830653.

We have observed that a number of common paradigms exist in data-focused mobile deployments conducted by groups in low-resource settings. Based on our experience, we characterize these deployments along five axes of design: whether the **D**ata flow is unidirectional or bidirectional; whether the **U**ser interface (UI) is form-based or non-form-based; if **C**onnectivity is required to function; if **E**dits are non-transactional or transactional; and if the supporting **S**erver is merely a data repository or if it encapsulates logic. Using technology always presents challenges. It is easy to argue for simplification, but developing intuitions around how to do so can take years of experience and can be specific to a single technology. The DUCES framework provides a way to approach simplification that is generalizable to a wide range of scenarios and tools.

DUCES is aimed at small and medium-sized organizations seeking to leverage mobile technology in low-resource settings. These organizations generally do not have the resources to devise a custom technical solution. They are not creating new technical frameworks and they do not have a developer on staff. They are seeking to build on top of existing solutions to leverage mobile technology. Such organizations frequently face difficulties when trying to reason about diverse requirements and their implications [4]. In these organizations, deployment architects are generally not developers themselves. This frequently makes the implications of requirements opaque. Many deployment architects lack even basic intuitions about what is easy and what is hard. For example, supporting two languages is a fundamentally different problem than working in the absence of an internet connection. Sending SMS reminders automatically based on HIV status is more challenging than capturing GPS data. Organizations frequently treat all requirements as equal, even though disconnected operation or server-side automation might complicate the deployment by orders of magnitude.

In this paper we explore five case studies of mobile deployments that leverage technology in different ways. We analyze these deployments to gain a comprehensive understanding of the various technical requirements that exist in mobile workflows in low-resource settings. The contributions of this paper are to formalize a framework for understanding and simplifying these mobile-based workflows. This framework, which we refer to as DUCES, elucidates characteristics and intuitions that are latent in many data-focused mobile apps, including those in high-resource environments, but that take on increased importance in low-resource settings. Using DUCES, organizations can identify early in their process what components are likely to require outside technical support and what should be able to be accomplished in-house.

The paper is structured as follows. In Section 2 we outline five case studies of mobile deployments in low-resource settings. We summarize the requirements and goals of each case study. In Sec-

tion 3 we describe the five axes of design that define the DUCES framework. We revisit each case study, exploring how the requirements of the deployment impacted the deployment architecture. In Section 4 we discuss how the traits of our framework highlight fundamental challenges that exist in mobile deployment architectures, how certain features are in tension with one another, and the ramifications of these considerations on mobile deployments. We revisit each case study to describe how DUCES was used to simplify or could potentially simplify each architecture. We close by discussing the ramifications this work has on organizations deploying mobile apps in low-resource settings.

2. CASE STUDIES

The goals of a mobile deployment define the technical requirements. This is not always a straight forward process. In this section we present five data-focused mobile deployments to serve as case studies. They were chosen to provide a broad sample of requirements. Characterizing them through the lens of the DUCES framework lends insight into what sorts of technical solutions are appropriate given the constraints of the deployment. Three of these case studies have been conducted by the authors, allowing insight into how DUCES was used during development of the deployment.[1] All have been deployed and used in the field. Two case studies are based on published literature.

2.1 Longitudinal HIV Study

The first deployment is support for a study of HIV discordant couples in Kenya [13]. The study itself was designed by global health researchers in order to longitudinally monitor couples where one partner is HIV-positive and the other is not. Participants are screened, at which point data is collected, and at time points in the future additional sets of data are collected. For both screening and follow ups participants are administered a survey on a mobile device. Different data is collected at different time points about male and female participants. A particular form is administered to a participant based on the time they have been enrolled in the study and their gender. This model of an entry form with follow up forms is common in research studies [7].

Each day study coordinators perform basic analysis and create a list of participants that require follow up. This data includes the subject's unique identifier and the form that is required to be completed. Enumerators are hired to take this information into the field, locate the subject, and complete the specified form. They are equipped with mobile devices that render the forms and provide a simple app-like user interface. Upon opening the app, enumerators arrive at a home screen and are trained that they can screen a new participant, perform a follow up interview, or submit collected data to the server. Sample screens are shown in Figure 1.

2.2 Tuberculosis Test Results

In 2009 a digital form-based workflow was deployed in Lima, Peru to digitize tuberculosis (TB) test results [2]. Sputum smears are collected at local health centers and the test results are written in a ledger. Enumerators visit these health centers with personal digital assistants (PDAs) that are equipped with digital forms. The forms have been designed to collect the information that has been written in the ledger, and enumerators transcribe the contents of the ledgers to the digital forms on the PDA. Upon returning to the central office, the data is uploaded from the PDAs to an Oracle databased managed by Partners in Health, a non-governmental organization (NGO) operating in Lima. Data upload takes place over the internet using the open database connectivity (ODBC) standard. The study designers also extended the database to include automated processing of the data as well as web pages that allow for summaries of the data and provide data quality checks. With this workflow, processing times for samples were greatly improved.

[1]Two of these three deployments are presented here for the first time.

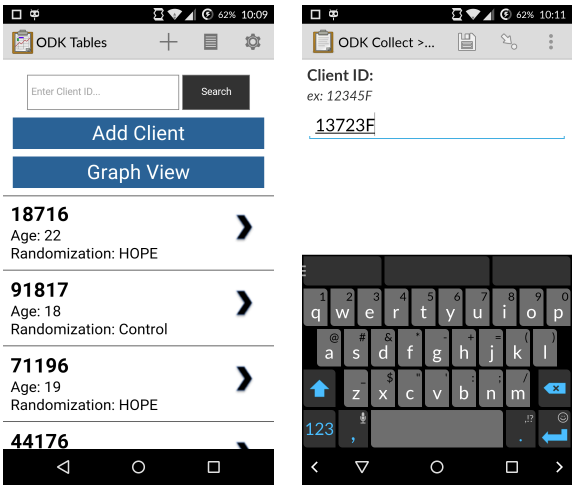

Figure 1: Examples of the mobile app for the HIV study. The participant list supports prepopulation of identifiers (left). If not available, the identifier can be entered manually (right).

2.3 Supply Chains Using Mobile Phones

Mobile devices have been deployed to improve the performance of rural supply chains in resource-constrained environments [12]. For this deployment an organization (Logistimo) noticed that stock outs were occurring at health centers in large part due to poor information management and communication. They created a Java application for feature phones that allows pharmacists to enter stock-related data, including the sale of items and stock counts. This data is transmitted to a server via a cellular data connection or SMS. The server component processes the data and removes duplicates that have arisen due to network errors. The server is also responsible for sending alerts to supervisors via SMS or voice calls. The server also provides a "bulletin board" web application that shows streaming information about the state of the supply chain. Synthesizing data in this way created actionable items that resulted in a drastic increase of availability of vaccines at local health centers.

2.4 Chimpanzee Monitoring

The Jane Goodall Institute (JGI) has employed a complex chimpanzee monitoring system for a number of years. Under the system, a ranger follows a group of chimpanzees through the forest over the course of a day with a complex paper worksheet. This activity is referred to as a "follow", and is broken into 15 minute time intervals. Data is collected about each interval. Various data is recorded, including when chimps arrive and depart, the estrus state of the female chimps, the foods consumed by the chimps, and the presence of other species. All this information is captured on a dense paper worksheet consisting of a matrix with 15 minute intervals on the y-axis and chimp identifiers on the x-axis. Arrivals and departures indicated by drawing a line in the corresponding 15 minute interval. A copy of the paper form is shown in Figure 2 in order to convey the density of information on the worksheet. These sheets

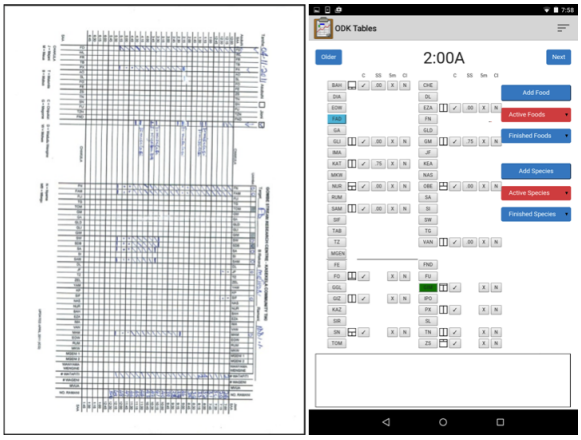

Figure 2: The Chimpanzee Monitoring Worksheet (left) and Application (right) The left image shows the paper worksheet employed by JGI rangers. It permits continuous review of data as it is updated. The right image shows this technique replicated in our application. This permits stylized reviews mimicking the paper-based workflow.

are periodically sent to researchers that transcribe the data into a database.

One of the strengths of this model is that rangers are able to see a summary of the day's data at a glance, making it easy to visually audit data and revisit time points as the day progresses. It also facilitates non-standard data entry. For example, times are not recorded by writing an hour and a minute. Instead the ranger draws a line in the first third of the box representing 9:00 to 9:15 to show that the chimp arrived or departed between 9:00 and 9:05. We designed an application to run on a 10 inch tablet that mimics this workflow. The tablet was smaller than the normal paper form employed by the rangers and displayed data of a single 15 minute interval at a time instead of the whole day's data. Crucially, however, data is displayed as it can be edited, affording rangers similar auditing power to the original paper form. The application also uses icons to achieve the same pictographic data entry to represent time that is provided by paper. The familiar tabular structure is preserved. A comparison of the paper form and the tablet application is shown in Figure 2.

2.5 Aid Distribution

The International Federation of Red Cross and Red Crescent Societies in the Americas (IFRC) often handles aid distribution after natural disasters. They recently piloted a program where debit cards were distributed instead of physical goods. We devised a mobile system to support this workflow. The pilot took place over two days in Kingston, Jamaica, and involved 93 participants at two locations.

Registration and distribution are separated into two distinct phases. In the registration phase, beneficiaries are entered into the system using digital forms on four mobile phones. Basic data like name and address are collected. Each beneficiary is also be assigned a paper card with a bar code that will later entitle them to receive their debit card. After screening and before distribution, each screened patient is assigned a debit card number. Distribution occurs an hour later. During the distribution phase, beneficiaries present their beneficiary card and bar code, which is used by the mobile app to retrieve their information. After confirming that it is correct, they are presented with the cash card and marked in the database as having received the card.

3. THE DUCES FRAMEWORK

The DUCES framework provides a way to characterize mobile deployments. The framework can be applied to mobile-based workflow, including applications produced by organizations with significant resources. However, it is most useful in low-resource contexts where many simplifying assumptions appropriate in high-resource settings are inadequate. DUCES consists of five axes of design:

1. Data Flow (Unidirectional vs Bidirectional)
2. User Interface (Form-Based vs Non-Form-Based)
3. Connectivity Model (Connected vs Disconnected)
4. Edit Mode (Non-Transactional vs Transactional)
5. Server Model (Bucket-Based vs Processed)

Understanding where a deployment falls along these axes provides meaningful insight into the technical requirements of a deployment. Deployment architects without a strong technical background often lack sound intuitions surrounding mobile deployments. For example, we have seen non-technical collaborators assume that altering the text accompanying a question in a form will be as difficult as adding bidirectional data flow to an existing data entry tool. DUCES is intended to better scaffold reasoning about deployment requirements in order to prevent this sort of misunderstanding. If a deployment requires a bidirectional data flow, a data entry tool that does not support the bidirectional movement of data can be discounted out of hand. Better still, the bidirectional data flow requirement might be obviated by a slight change in deployment protocol. DUCES provides a set of primitives that can guide an understanding of technical requirements.

In the following sections, each axis of design is described in detail. The case studies from Section 2 are used to illustrate how DUCES can be used to describe a variety of mobile deployments.

3.1 Data Flow

Mobile devices are frequently used to collect and manage data, aggregating it on a server. DUCES asks if this flow of data is unidirectional or bidirectional. In other words, does collected data move only from the mobile device to the server, or does it also move from the server to the mobile device? A unidirectional data flow is simpler to implement than a bidirectional data flow, but it is also less versatile. Deployments can use each design to great effect.

3.1.1 Longitudinal HIV Study

The HIV study is a **unidirectional** data flow. Subject data is collected using forms and later submitted to a central server. Data is never sent from the server back to the device. Enumerators are essentially replicating a paper-based data collection workflow, which is a unidirectional data flow.

3.1.2 Tuberculosis Test Results

This too uses a **unidirectional** data flow. Data is transcribed from the ledgers at health centers into the forms on the PDAs carried by enumerators. At their central office they then submit data to the server.

3.1.3 Supply Chains Using Mobile Phones

In this case the data flow is **bidirectional**. Pharmacists use a Java-based application for feature phones that sends sale and stock count information to the central server. The server processes this data and broadcasts resultant information to all users of the system via SMS. Further, the list of materials at each health center is pulled from a central server. The authors of the tool took care to degrade gracefully in the case of poor connectivity, but the flow of data in the deployment is still bidirectional.

3.1.4 Chimpanzee Monitoring

Data collected by rangers is stored locally. It can be reviewed and revised until it is pushed to the central server. No data is ever sent from the server to the device. Further, this deployment is a replacement for paper-based data collection. All of these characteristics indicate that it employs a **unidirectional** data flow.

3.1.5 Aid Distribution

Beneficiary information is entered on mobile devices in a digital form. Screening takes place on four mobile devices. During distribution, beneficiaries must be able to be processed on any device, not just the device that screened them. This indicates that data must be shared between devices, indicating that the data flow is **bidirectional**.

3.2 User Interface

Mobile deployments can be described in terms of two distinct modes defining their users' interactions: form-based and non-form-based. The distinction between these modes is motivated by two factors. First, form-based data entry is extremely common in low-resource deployments [7]. Second, a large number of tools exist that facilitate form-based data collection on mobile devices. Many are designed specifically to be leveraged by lightly technical users. This host of data-entry applications includes Google forms, CyberTracker [1], Red Cap [9], Open Data Kit (ODK) Collect [10], ODK Survey [3], Pendragon Forms, Magpi, CommCare, and many others.

One hallmark of form-based tools is that deployment architects do not normally need sophisticated control over the presentation layer of form-based data entry: presenting a piece of text indicating what data should be entered is usually sufficient. In a form-based UI, a user is making changes to the database by stepping through a series of questions, potentially with branching based on responses. Non-form-based UIs do not have as clearly defined workflow. Almost all mobile applications produced by highly technical organizations and aimed at high-resource settings (e.g. email apps, to-do lists, calendars, and chat clients) do not use a form-based workflow. This stands opposed to the applications used by organizations in low-resource settings, where workers are often employed to collect data using mobile phones.

Any workflow can be viewed through these two modes, but they have increased relevance in low-resource settings. Which of these modes is appropriate and necessary for a given deployment can define immediately the types of interactions users will have with mobile devices. Understanding and describing a deployment can be greatly simplified if deployment architects identify early if they can model their deployment's user interface using a form-based workflow. In general, the more a deployment can be forced to follow a form-based workflow, the easier it will be to manage by local organizations with tools that have a relatively low barrier to entry.

3.2.1 Longitudinal HIV Study

The HIV study employs a predominantly **form-based** user interface. Forms were designed for screening participants and for a number of follow up visits. A lightweight non-form-based skin was designed to present a list of existing participants, but the vast majority of enumerators' time is spent completing digital forms.

3.2.2 Tuberculosis Test Results

Here again the user interface is predominantly **form-based**. Pendragon Forms was used to create forms mimicking the data collected on paper ledgers and transcribed on PDAs. A non-form-based component was used to perform data quality checks on the server, but enumerators spent most of their time completing digital forms.

3.2.3 Supply Chains Using Mobile Phones

Pharmacists in the supply chain deployment used a custom-built Java app for feature phones. The bulletin board aggregating results from pharmacists was written as a web page. Thus the user interface was **non-form-based**.

3.2.4 Chimpanzee Monitoring

Although the JGI was replicating paper-based data collection, the tablet-based app did not follow a standard digital form workflow. Unlike in conventional forms, the JGI had strict requirements for the presentation layer: it must be tabular, show previously entered data to allow visual auditing, and data entry must employ stylized icons rather than text. This workflow was **non-form-based**.

3.2.5 Aid Distribution

Both screening and distribution phases took place using digital forms, making this a **form-based** user interface.

3.3 Connectivity Mode

Mobile deployments can follow one of two connectivity models: connected and disconnected. A disconnected model is one where full functionality is capable without connecting to a central server. In this model, workers would be able to go into the field for periods of time and use a mobile tool without degraded quality. For example, data enumerators might leave the city for several weeks at a time collecting data about the state of a country's refrigeration infrastructure at its health centers. When they return they submit their information to their supervisor. The tasks they were expected to perform were not dependent on a reliable connection to the internet or to their superior via a telephone.

Connected operation, on the other hand, requires a connection to achieve the full functionality of an application. Mobile deployments in low-resource environments can adopt either model: organizations might have an enumerator in the field without connection or a researcher working at a large hospital with a strong wifi connection.

3.3.1 Longitudinal HIV Study

Many of the participants in the HIV study are expected to be contacted in the field without internet connectivity. The mobile app was designed to work entirely offline, with forms created using ODK Collect. Collect permits data entry offline, storing data locally until it is uploaded when internet connectivity becomes available. Enumerators thus were able to use the app's full functionality without an internet connection, making this a **disconnected** connectivity model.

3.3.2 Tuberculosis Test Results

Data was entered on PDAs using Pendragon Forms. This did not require connectivity until data was uploaded at a central office.

This is a **disconnected** model, as full functionality did not require a connection.

3.3.3 Supply Chains Using Mobile Phones

The authors of the supply chain intervention took great care to ensure that their mobile devices would accommodate a **disconnected** connectivity model. After an initial download, data is persisted locally. Service degrades gracefully, defaulting to SMS data transfer if an internet connection is not available, and allowing full offline entry if neither SMS or data is available.

3.3.4 Chimpanzee Monitoring

The mobile tool for the JGI was designed from the outset to embrace a **disconnected** connectivity model. Data can be collected entirely offline and only requires an internet connection to send data to a server.

3.3.5 Aid Distribution

The aid distribution deployment requires a **connected** connectivity model. Distribution cannot follow screening without first aggregating all the data centrally, pairing beneficiaries with a debit card, and downloading this new information to all the devices. Further, without a connection workers cannot prevent double distribution—a beneficiary might visit two distribution stations.

3.4 Edit Model

Data edits within a deployment can also be characterized as non-transactional or transactional. Transactional data is data where edits are dependent on one another. The order of edits matter and are conceived of as a unit. Non-transactional data is data where the order does not matter and edits are independent of each other. For example, records of medical visits are non-transactional. Each record refers to a separate visit. Reports may be submitted out of order and remain coherent. Transactional data, meanwhile, places stricter requirements on ordering. An example is financial data, where a sequence of withdrawals and deposits must be ordered to ensure that the balance is sufficient to address subsequent requests.

3.4.1 Longitudinal HIV Study

The HIV study treats data entry as a sequence of reports. The six month follow up is not dependent on the six week follow up. This is a **non-transactional** edit model.

3.4.2 Tuberculosis Test Results

Again each collected data point is isolated and independent of the others, making this a **non-transactional** edit model.

3.4.3 Supply Chains Using Mobile Phones

In this case data collected from pharmacists consists of stock counts and stock disbursements. This data is only useful if ordered. If all stock disbursements were sent two days late, the functionality of the bulletin board system would be severely impacted. Consequently this deployment requires a **transactional** edit model.

3.4.4 Chimpanzee Monitoring

Data is collected about each time point and is independent of the others, making this a **non-transactional** edit model.

3.4.5 Aid Distribution

A cagey beneficiary might try and cheat the system by visiting two distribution systems in order to receive double aid disbursement. This implies that ordering matters and edits are not independent of each other, making this a **transactional** edit model.

3.5 Server Requirements

Broadly speaking, the server requirements for a mobile deployment can be bucket-based or processed. Bucket-based servers are the simplest, acting as receptacles or sources for data. Processed servers are everything else. This distinction is purposefully broad, as the moment a server stops being bucket-based it becomes significantly more complicated. Bucket-based servers are those where all form data is submitted to a single location or pulled from a single location. Processed servers might serialize data for consumption or analyze data and perform notifications. Bucket-based server workflows can be replicated with a wide variety of tools, while processed servers require more customization and configuration.

3.5.1 Longitudinal HIV Study

This is a classic **bucket-based** server configuration. Data from each form is sent to a table on a server. No processing is required. The forms are written using ODK Collect and the bucket-based workflow is facilitated by ODK Aggregate, which serves as a bucket for form data.

3.5.2 Tuberculosis Test Results

This deployment uses Pendragon Forms to send data to an Oracle database. A module was added to the back-end that supported validation of submitted data, highlighting errors in red. This represents a **processed** server requirement.

3.5.3 Supply Chains Using Mobile Phones

The server in this deployment performs a number of tasks. It supports bidirectional data flow of JSON data to the Java feature phone application, it synthesizes data and presents it on a helpful bulletin board, and it sends broadcasts to registered users of critical events. This represents a high degree of customization and configuration and demonstrates what can be accomplished with a **processed** server.

3.5.4 Chimpanzee Monitoring

Data is collected locally and sent to the server. Nothing is required of the server beyond being a receptacle for data, making it **bucket-based**.

3.5.5 Aid Distribution

Screening and distribution data is entered using a digital form and submitted using a **bucket-based** server configuration.

4. DISCUSSION

The power of the DUCES framework is twofold. First, it provides a schema by which to understand the requirements of a mobile deployment. Second, it provides a means by which to simplify a deployment.

4.1 Understanding

DUCES permits deep insight into the requirements of mobile deployments in low-resource settings. With appropriately scoped requirements, solutions can be created by leveraging existing technology. With sufficient technical knowledge and resources, custom-built solutions can be created to meet any set of requirements. Electronic medical record systems have been deployed successfully on custom technology using commercial-quality servers and custom work stations in Haiti for thousands of patients [11]. Touchscreen PCs running custom software have been used in Malawi to improve point of care treatment and provide immediate reporting to doctors in the field [5]. These are testaments to the potentially transforma-

tive power of technology, but unfortunately such technical feats are not available to a number of organizations with fewer resources.

A crucial observation is that the axes of design underpinning DUCES do not exist in isolation. A bidirectional data flow might affect server requirements, for instance, perhaps necessitating a processed configuration. The supply chain case study used bidirectional data flow to send lists of items and alerts to mobile phones. Consequently they required a processed server to synthesize data and generate alerts as well as to present a list of items to mobile phones in a specialized format (JSON). Unfortunately, however, there are no hard and fast rules when reasoning about the impacts of architectural decisions. The aid distribution case study, for example, similarly uses a bidirectional data flow but manages to use a bucket-based server. This seeming contradiction is one example of why it can be difficult for organizations to hone their intuitions surrounding technology requirements.

Why the discrepancy? In short, the supply chain case study required logic on the server that would create events and broadcast them to registered devices. It also needed to expose data using a custom format—JSON—that could be consumed by devices. The aid distribution study, meanwhile, was implemented using ODK Survey and Aggregate, which supports an out-of-the-box bucket-based server for producing and consuming data. This is difficult to recognize prima facie without extensive knowledge of the capabilities of the tools being used to implement the study. For deployment architects without a strong technical background, this will be an especially difficult conclusion to draw.

Instead, DUCES claims that the most easily satisfied configuration is **unidirectional**, **form-based**, **connected**, **non-transactional**, and **bucket-based**. If a set of requirements can be modeled using this configuration, it will be more likely to be managed successfully without outside technical resources. Deviations from this model will create additional dimensions and edge cases that will complicate the deployment and potentially require technical assistance.

For example, consider trying to support transactional data using a disconnected workflow. In the aid distribution scenario, the edit model is transactional. This is necessary as it is important that aid is not distributed twice to the same beneficiary. If errors cannot be tolerated, this requires a connected connectivity model. This is a familiar problem in distributed systems, as it is essentially an extension of the CAP theorem [6]. The CAP theorem states that a system cannot be partition tolerant, available, and provide a consistent view of the data simultaneously. In the context of the aid distribution case study, no worker would be able to become disconnected (essentially partitioning the network) while the other users are able to maintain a consistent view of the data (not double distributing) but not have to wait for all users to reconnect. This highlights a fundamental tension between a transactional edit model and disconnected workflows. The supply chain case study was able to use both transactional data and a disconnected workflow by adding processing logic to their server and, in the worst case, simply tolerating late data that was no longer actionable. This was acceptable in their deployment and was a necessary concession to support disconnected operation.

DUCES also provides insight into why mobile deployments in low-resource settings are fundamentally challenging. The simplest configuration can be at odds with the realities of the environment. It is common for many deployments to require disconnected operation, for instance, because they occur in regions without reliable data connectivity. As we have seen, this complicates the handling of transactional data but is simply unavoidable in some settings. To take another example, the chimpanzee monitoring study required a non-form-based workflow to be effective. The JGI had previously tried to encode the workflow using traditional form-based building tools without success. In the end they required a custom solution that could support their non-form-based workflow.

4.2 Simplification

DUCES can also be used to guide the simplification of mobile deployments. It posits that deviations from the simplest configuration of **unidirectional**, **form-based**, **connected**, **non-transactional**, and **bucket-based** will invite technical complications that may be insurmountable without the aid of a developer.

For example, bidirectional data flow is more difficult to support than unidirectional data flow. If a bidirectional data flow requirement can be loosened to a unidirectional data flow, this will have positive ramifications for the sustainability of the deployment. In many cases it is easier to alter the requirements of a deployment than to devise a sophisticated technological solution that will add complexity and hurt sustainability. We now discuss the five case studies in the context of simplification using the DUCES framework.

4.2.1 Longitudinal HIV Study

This study has been running successfully for over 24 months in a hospital in Kenya. The researchers first requested a bidirectional data flow and a processed server model. Five to ten phones were to be shared between enumerators and used to screen participants and perform follow up interviews. The researchers had previously seen enumerators make errors typing subject identifiers, making it difficult to perform post-hoc analysis. They reasoned that bidirectional data flow would allow all participant records to exist on all phones. Whenever a participant was contacted for follow up, enumerators would not have to re-type the identifier, reducing the likelihood of errors.

However, supporting bidirectional data flow would complicate requirements. First, the server would have to support presenting captured data in a machine-consumable way, similar to how the supply chain example provided JSON. Second, it might require authentication and access control to prevent the collected HIV data from being visible to non-study devices. Third, enumerators would have required a stable internet connection at headquarters before leaving for the field. If a connection issue interrupted the bidirectional data flow, the flow might be interrupted.

Instead, the study designers were able to refine their study procedures to work within the confines of a unidirectional data flow. Enumerators were made responsible for the same cohort of patients and assigned specific devices rather than a shared device. This greatly increased the likelihood that a participant would already be present on the device without requiring bidirectional data flow. However, participants might still be seen for follow up interviews on devices that were not used to screen them. This might occur if the screening device was lost or stolen or if they were visited by a new enumerator for logistical reasons. To accommodate this eventuality, the follow up workflow was modified to allow entering an existing patient identifier. This was not completely in-line with the original requests of the researchers, as they requested that the identifier not be entered manually more than once, but with the advent of individually assigned devices the likelihood of a manually entered identifier was less common and was deemed acceptable.

A processed server was desired to calculate when participants were due for follow up visits. This information would be generated each morning and provided to study coordinators. Automating this task would not be complicated for a computer scientist, but the smooth operation of the study would depend on this task functioning without interruption. This might prove difficult with-

out a technical staff capable of supporting the server. Instead, the researchers directed study staff to manually review the data on the bucket-based server and generate a list of follow up participants each day by hand. This might seem less than elegant to a computer scientist, but it is much more sustainable with local talent.

In this way a bidirectional, processed workflow was transformed and simplified to use a unidirectional, bucket-based workflow. These simplifications are a large part of the reason that the study has remained in successful operation with limited involvement from outside technical staff for over two years.

4.2.2 Tuberculosis Test Results

The authors were not involved in this deployment, so DUCES will be used to describe how the deployment might have been further simplified from its current incarnation rather than how it was simplified in practice. The configuration was almost an ideal DUCES configuration, being unidirectional, form-based, disconnected, and non-transactional. The server, however, was processed, performing validation logic and displaying it as a web page. This required adding a module to an Oracle back end managed by an NGO [2].

Although the authors do not state it, this likely required a developer with the technical ability to create a web page and encode validation logic. This processed server requirement may have been able to yield to an unprocessed server if server-side validation was not automated. Instead, data could have been exported to a format like comma-separated values (CSV) that can be consumed by a number of programs. At that point it could be manually validated by a staff member, or validation logic could be encoded in a Microsoft Excel worksheet rather than a web page. This would slow the cycle of validation but would not require web programming skills. Alternatively, validation logic is supported by a number of form-based data entry tools. The researchers could instead have performed validation upon data entry rather than during server auditing. Both of these solutions would yield a near optimal configuration under the DUCES framework.

4.2.3 Supply Chains Using Mobile Phones

This deployment serves as a testament to the rich functionality that can be achieved using custom solutions. A custom Java application for feature phones communicated with a custom processed server capable of performing analysis and broadcasting alerts to users. Here again the authors were not involved with the deployment, so application of the DUCES framework will be aim to demonstrate how an organization with less technical resources might try to replicate the success of this workflow.

First, the non-form-based workflow for feature phones could be replaced by a form-based data entry tool for smart phones. As discussed in Section 3.2, a number of form-based data entry tools have been designed to support use by non-programmers.

The processed server model is crucial to this deployment. The authors of the study argue that consumers of the information submitted by the mobile devices are too busy to synthesize the reports without automation. This domain knowledge suggests that simply converting to a bucket-based server model is inappropriate. The authors also note explicitly that their edit model is transactional and that their mobile application functions offline. It is informative to look at how they circumvent the requirement put forth in Section 4.1 that transactional workflows require connectivity.

The answer is twofold. First, they apply server-side processing to deduplicate and process errors that are created as a result of network errors. Second, although their data is transactional, they are able to tolerate errors. In terms of the CAP theorem, they are able to tolerate a loss in consistency as long as the system remains available during periods of no data connectivity. The ramifications for the deployment are that events might not be shown on their web-based bulletin board in real-time. A stock out might be reported late, but this is likely uncommon and is thus deemed acceptable.

4.2.4 Chimpanzee Monitoring

The chimpanzee monitoring case study attains a near-optimal DUCES configuration. It fails by being non-form-based and disconnected. In reality the only simplification that might be afforded by a connected model would be that a wider variety of tools could be used to implement the framework. This would thus accommodate a wider range of user interface components and back ends, including potentially a web-based application. Practical implications of this change are low due to the fact that the data is non-transactional and thus ordering is not significant.

Arriving at this configuration was straight-forward, as the JGI was seeking to replace an existing paper workflow. It is important to note that the non-form-based workflow necessitated the involvement of the assistance of developers, which is an added technical burden. The JGI has extensive experience creating form-based workflows, but the creation of a non-form-based workflow requires an additional skill set. In this case the DUCES framework did not simplify the deployment, but it did clearly delineate where external resources would be required.

4.2.5 Aid Distribution

The aid distribution case study was bidirectional, form-based, connected, transactional, and bucket-based. In the HIV case study, the bidirectional data requirement was able to be eliminated by having participants ideally interact with only a single device, obviating the need to share data between multiple devices. This was not possible in the aid distribution case study, as the nature of the distribution environment required that beneficiaries not be confined to an individual device. With this requirement, tools immediately had to be chosen that could support bidirectional data flow. This eliminated some possibilities like Pendragon Forms, Magpi, and ODK Collect, which support only unidirectional data flows.

A connected connectivity model was required to accommodate the transactional nature of the data. In this case, if connectivity was lost, workers might see an inconsistent view of the data. In other words, a beneficiary may have received aid from one disconnected distribution station and then received aid from a second station that was not aware aid had already been distributed. To prevent this, the system required a connected model. This would in turn require that whatever tools were used to implement the deployment support an online, connected workflow. However, this requirement was able to be circumvented by the real-world details of the aid distribution.

In this case the aid itself was a debit card that had been uniquely assigned to each beneficiary. Once distributed, it could not be distributed again, preventing double distribution. This relaxes the requirement slightly, although at the cost of masking errors during distribution. Coordinators would know a card was missing, but they would not know if it had already been given to the correct recipient or had simply been lost or misplaced. Further, this approach would not accommodate distribution of goods that were not uniquely paired to beneficiaries.

5. CONCLUSION

Mobile devices are increasingly integrated into the workflows of organizations working in low-resource settings. We have presented the DUCES framework as a means to elucidate the requirements of data-focused mobile deployments. We have described five case studies with varying aims and requirements and discussed how they

can be evaluated under the DUCES framework, permitting both a meaningful understanding of requirements and guiding simplifying assumptions. Using this framework, deployment architects can start to identify what requirements will add significant complexity. This in turn facilitates the selection of technologies. If a deployment requires bidirectional data flow and that requirement cannot be loosened, technologies that do not support bidirectional data flow can be discounted immediately.

It is important to note that the DUCES framework is not simply an attempt at formalizing a series of known tradeoffs in system design. For example, encrypting data can result in decreased usability while increasing data security. DUCES instead provides a mental scaffold by which a single set of requirements can be described and satisfied in a number of different ways. A processed server might require a custom-built solution with a custom database and a suite of scripts running every night. It instead might be transformed into a bucket-based server by having an administrator copy relevant rows between data sinks and data sources. This would seem an ugly solution to a computer scientist that prefers to automate all tasks. However, it would elegantly allow a lightly skilled deployment architect to compose simple tools in a powerful way that meets the needs of their deployment while remaining comfortably within their skill set.

Some things will always remain difficult in mobile deployments conducted by organizations and groups with limited resources. Transferring data between back-ends is a prime example. Inputting data collected into another system will always be difficult. No choice of tools will completely allay this difficulty unless a programmer has already taken the time to create a method by which data can be exported in a form consumable by the tool in question. Organizations are better off recognizing that this will require technical expertise than they are limiting their technological solutions to one that will be compatible with their current target repository out of the box.

The DUCES framework provides a useful set of considerations for architects of mobile deployments and data-based workflows. The framework can be applied to all systems, but it is most effective when considered by deployment architects operating in low-resource settings. In these environments DUCES can be used to re-imagine requirements in ways that will make mobile workflows easier to deploy and maintain. The DUCES framework is a valuable tool that organizations in low-resource settings can use to characterize and simplify their data-focused mobile deployments.

6. ACKNOWLEDGMENTS

The authors would like to thank USAID, NSF Grant IIS-1111433, the Jane Goodall Institute, the IFRC of the Americas, and the open source community that made this work possible.

7. REFERENCES

[1] Edwin H Blake. 2002. Extended abstract a field computer for animal trackers. *CHI 02 extended abstracts on Human factors in computing systems CHI 02* (2002), 532. DOI: http://dx.doi.org/10.1145/506461.506466

[2] Joaquín a. Blaya, Ted Cohen, Pablo Rodríguez, Jihoon Kim, and Hamish S F Fraser. 2009. Personal digital assistants to collect tuberculosis bacteriology data in Peru reduce delays, errors, and workload, and are acceptable to users: cluster randomized controlled trial. *International Journal of Infectious Diseases* 13, 3 (2009), 410–418. DOI: http://dx.doi.org/10.1016/j.ijid.2008.09.015

[3] Waylon Brunette, Mitchell Sundt, Nicola Dell, Rohit Chaudhri, Nathan Breit, and Gaetano Borriello. 2013. Open Data Kit 2.0: Expanding and Refining Information Services for Developing Regions. *HotMobile '13* (2013), 6. DOI: http://dx.doi.org/10.1145/2444776.2444790

[4] Kuang Chen, Akshay Kannan, Yoriyasu Yano, Joseph M. Hellerstein, and Tapan S. Parikh. 2012. Shreddr: pipelined paper digitization for low-resource organizations. *ACM DEV '12* (2012), 1. DOI: http://dx.doi.org/10.1145/2160601.2160605

[5] Gerald P. Douglas, Oliver J. Gadabu, Sabine Joukes, Soyapi Mumba, Michael V. McKay, Anne Ben-Smith, Andreas Jahn, Erik J. Schouten, Zach Landis Lewis, Joep J. van Oosterhout, Theresa J. Allain, Rony Zachariah, Selma D. Berger, Anthony D. Harries, and Frank Chimbwandira. 2010. Using Touchscreen electronic medical record systems to support and monitor national scale-up of antiretroviral therapy in Malawi. *PLoS Medicine* 7, 8 (2010). DOI: http://dx.doi.org/10.1371/journal.pmed.1000319

[6] Armando Fox and Eric Brewer. 1999. Harvest, yield, and scalable tolerant systems. *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems* (1999). DOI: http://dx.doi.org/10.1109/HOTOS.1999.798396

[7] Hamish SF Fraser, Christian Allen, Christopher Bailey, Gerry Douglas, Sonya Shin, and Joaquin Blaya. 2007. Information Systems for Patient Follow-Up and Chronic Management of HIV and Tuberculosis: A Life-Saving Technology in Resource-Poor Areas. *Journal of medical Internet research* 9, 4 (2007).

[8] Abhishek Gupta, Jatin Thapar, Amarjeet Singh, Pushpendra Singh, Vivek Srinivasan, and Vibhore Vardhan. 2013. Simplifying and improving mobile based data collection. *ICTD '13 - volume 2* (2013), 45–48. DOI: http://dx.doi.org/10.1145/2517899.2517929

[9] Paul Harris, Robert Taylor, Robert Thielke, Jonathon Payne, Nathaniel Gonzalez, and Jose Conde. 2009. Research electronic data capture (REDCap)-A metadata-driven methodology and workflow process for providing translational research informatics support. *Journal of Biomedical Informatics* 42, 2 (2009), 377–381. DOI: http://dx.doi.org/10.1016/j.jbi.2008.08.010

[10] Carl Hartung, Yaw Anokwa, Waylon Brunette, Adam Lerer, Clint Tseng, and Gaetano Borriello. 2010. Open Data Kit: Tools to Build Information Services for Developing Regions. *Proceedings of the International Conference on Information and Communication Technologies and Development* (2010). DOI: http://dx.doi.org/10.1145/2369220.2369236

[11] William B. Lober, Stephen Wagner, and Christina Quiles. 2010. Development and implementation of a loosely coupled, multi-site, networked and replicated electronic medical record in Haiti. *ACM SIGOPS Operating Systems Review* 43, 4 (2010), 79. DOI: http://dx.doi.org/10.1145/1713254.1713272

[12] Arun Ramanujapuram and Anup Akkihal. 2014. Improving Performance of Rural Supply Chains Using Mobile Phones: Reducing Information Asymmetry to Improve Stock Availability in Low-resource Environments. *ACM DEV* (2014), 11–19.

[13] Samuel Sudar, Saloni Parikh, Mitchell Sundt, and Gaetano Borriello. 2013. ODK Tables : Case Studies in Deployment. (2013), 12–14. DOI: http://dx.doi.org/10.1145/2537052.2537077