RAVEN: Using Smartphones For Collaborative Disaster Data Collection

Nicholas Palmer, Roelof Kemp, Thilo Kielmann, Henri Bal
VU University Amsterdam
The Netherlands
{palmer,rkemp,kielmann,bal}@cs.vu.nl

ABSTRACT

In this paper we describe our work in progress on RAVEN, a framework**,** which makes it possible to build applications for collaborative editing of structured data on Android. RAVEN offers developers compile time tools**,** which use only the schema to generate all database handling components, edit and list user interfaces, as well as those needed for data synchronization, significantly reducing development effort.

In addition, RAVEN also offers the ability to do the same work, entirely at runtime, using only a smartphone. With RAVEN it is possible to construct data oriented applications on phone at any time, including during a disaster. Users can share their applications simply by sharing the database and corresponding schema. Thus, RAVEN enables completely decentralized application creation, sharing, and data distribution, avoiding issues of connectivity to centralized resources. In this paper we show that with RAVEN it is possible to construct a new application at runtime and compare the re**sults with an equivalent custom-**built application.

Keywords

Disaster Management, Smartphone, Android, Collaborative Software, Database

INTRODUCTION

Disaster management is a complex and interesting domain. The needs of disaster situations are very different from those found in normal life and the domain presents unique constraints which are both interesting and challenging to work with. In this paper we discuss how the RAVEN framework is able to overcome the constraints imposed by disaster management in order to enable new digital technologies to better help out when disaster strikes. In particular we demonstrate the possibility of using modern smartphones in order to assist those attempting to deal with a disaster to better perform data collection oriented tasks that the situation thrusts upon them.

Smartphones are a relatively new class of devices, which have been created by the confluence of wireless networking technologies and continued miniaturization of computers. They are small hand held telephones with colorful screens, multiple sensor technologies, and multiple networks capable of exchanging data, in addition to being mobile general purpose computers. It has also been shown that cell phones were the first communication technology to recover after the Hurricane Katrina disaster (Farnham, Pedersen and Kirkpatrick, 2006). Cell phones have already been observed to be of high utility in disaster situations where they are used to interact, negotiate and establish connections between professionals in order to meet the challenges posed by a disaster (Landgren and Nulden, 2007). Thus smartphones hold an even greater potential for this challenging domain.

The rise of these new devices coincides with a shift in the approach to disaster management. Increasingly disaster management scientists have realized that citizens are critical to effective disaster response (Schafer, Carroll and Haynes, 2005; Gomez and Turoff, 2007), particularly as the size of the disaster increases. In such situations it has been demonstrated that centralized command and control structures cannot provide the flexibility to handle the organizational challenges posed by disaster situations (Harrald, 2006; Jefferson and Harrald, 2007; Gachet and Haettenschwiler, 2003). These challenges cannot be addressed by military style hierarchical decision-making. This is because this style of control structure does not offer the agility to handle the ad-hoc organizations, which emerge spontaneously in response to a disaster. This is particularly true in the case of larger scale events. Furthermore, it has been shown that smaller self-managed organizations are more productive than more traditional management structures (Wheatley, 1997). What is therefore required is new technology with the ability to network people and act as a platform for emergent behaviors during a disaster response.

Of importance for the work presented in this paper is that dynamic data collection gathered by people carrying cell phones which is shared and thus can be used both at the time of the disaster, as well as for later analysis can provide a shared value for the common good (Kanjo E., 2009). Unfortunately, this requires not only the development of a specific disaster management application before the disaster strikes, but also that the user anticipates being involved in such an event and installs the application. Thus, we argue that an emphasis needs to be placed on application development environments in which applications can be implemented and deployed extremely quickly.

In this paper we present our work in progress on RAVEN, a framework for the creation of collaborative data collection applications for Android powered smartphones. This framework enables a form of "end-user programming" (Liberman, 2006) for data oriented applications, which uses decentralized synchronization over standard or ad-hoc networks to provide the high data availability required during a disaster situation. The activities the user undertakes to develop an application with the framework can be thought of as Model-Based development, where the user is defining a model for the data the application will manage. This framework is a generic component, which can be used to rapidly develop any application, which has need for a database, and once installed for one or more applications can also be used on phone to develop novel data oriented applications.

Our hope is that many applications will come to use the framework and thus it will be more likely to be pre-installed on users phones and familiar to users at the time disaster strikes. We thus hope that the framework will be familiar with RAVEN's capabilities because of the daily utility across many applications. Furthermore, applications built using our framework can be shared with other users by synchronizing the schema for the data with other users, allowing the distribution of applications using the frameworks using ad-hoc networks during a disaster. Thus, we hope applications built using our framework will be more easily deployed to new devices than would be possible with a purpose built application which must be installed from a centralized server, mitigating the during disaster deployment problem.

We begin by describing how the RAVEN framework has been built and what it provides. We then evaluate this framework using a sample "People Finder" application, implementing a version both with and without RAVEN in order to qualitatively compare development effort. We then outline our future work on this platform to quantitatively define the frameworks applicability in disaster situations.

RAVEN: A FRAMEWORK FOR DATA APPLICATIONS

The RAVEN framework leverages a number of existing technologies in order to provide a versioned database frame- work. For database-oriented operations RAVEN takes advantage of the SQLite embedded database system provided with the Android runtime system. This database is stored inside a Git repository in order to add versioning and sharing features to the system. Git is a distributed version control system targeted at source code developers. It enables users to commit, synchronize and merge repositories in a completely decentralized framework. RAVEN uses Git in order to provide commit, branch and sharing features using Git push and pull operations. Git functionality is provided via the jGit implementation of Git which is part of the Eclipse project. On top of these two layers is a Versioned Content Provider abstraction. Content Providers are an abstraction of Android used for storage and retrieval of data across applications. Android uses Content Providers to expose platform data such as contacts and media information, while applications can use existing content providers or write new ones. Content Providers expose an SQL-like interface composed of URIs, which specify a table, which can be queried, inserted or deleted. RAVEN extends the Content Provider interface by embedding version information inside the URI, which is used to specify a resource to query against and managing checkouts of the required version of the database on disk. Finally, RAVEN provides a number of user interface components for managing databases and database sharing, as well as listing and editing interface generators, which give users direct access to the data stored inside the database.

The framework is intended to be used by both developers and users in order to create new data oriented applications. Developers can use the framework at compile time in order to create a Content Provider for their data. This can be done using either an Object Relational Mapping interface, which uses Java class annotations to specify the data model, or using an Avro Schema. Avro is a schema and data-encoding framework, which is part of the Apache Hadoop project. RAVEN uses Avro schemas internally in order to represent database schemas as well as to handle schema evolution. RAVEN is also able to generate an edit user interface based on an Avro schema for the database allowing runtime generation of both a database and listing and edit user interfaces. Readers interested in a more detailed discussion of the framework internals are invited to read our previous work with more details (Palmer 2011).

Additionally, RAVEN offers a schema editing application as a portion of the framework in order to allow users to define a new database on the phone at runtime without having to know how to write an Avro schema, or alternatively, allowing users to change the schema for an existing application at runtime when new data fields are required. This application uses a "schema for schemas" which treats a schema as just another type of data to be edited. This "schema for schemas" is then passed to the user interface generation components of the RAVEN framework and the resulting database entry is read and used to generate a new Avro schema which in turn is used to instantiate a new database at runtime. With this application, RAVEN can be used to construct data oriented applications both by developers working with a laptop and by inexperienced users working with just the RAVEN application components on their phone. The later is the focus of the rest of this paper.

Finally, and of great importance for disaster management, because the user interface is generated from the schema for the database, by sharing a database and schema with another user using ad-hoc networking techniques, the framework is able to share new data collection applications and all collected data, using only a smartphones, avoiding the problem of the failure of networking to centralized resource.

EXAMPLE APPLICATION: PEOPLE FINDER

In order to evaluate the power of the RAVEN framework from a qualitative point of view, we have implemented a simple disaster management related application for tracking lost and found people. The 'People Finder" application consists of a user interface for editing records in a database which can be used to track lost and found people. We have implemented this application both using our framework, and without our framework in order to be able to compare and contrast the development effort required as well as the usability of the resulting applications. This application is similar to a component of the Sahana Eden(Amin and Goldstein, 2008) platform. However, Sahana uses a centralized approach and thus the loss of networking common in a disaster area may make the Sahana interface unusable for data collection in the field, where as RAVEN takes a phone centric approach in order to enable higher data availability.

Standard Android Implementation

In order to have a basis for comparison for the version of this application developed using RAVEN we have implemented a version of our application using standard Android development tools and following the standard practices suggested by Google in the documentation and examples for the Android platform.

This implementation required a number of days of effort by an experienced developer and nearly 3500 lines of code. This represents a significant development time, preventing the rapid deployment of new applications when a disaster happens.

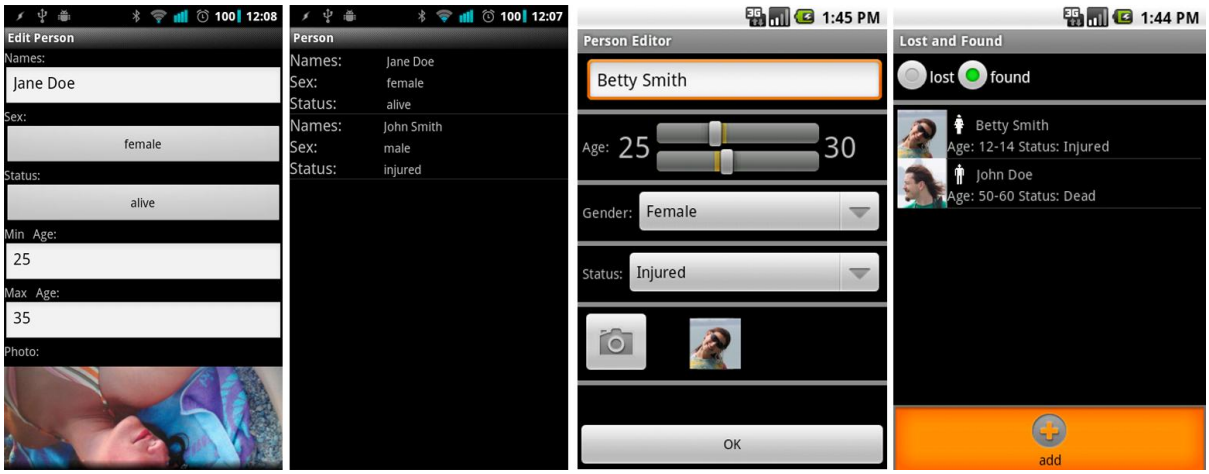

Figure 1. People Finder UI: Left: Framework Implementation, Right: Standard Implementation

The user interface for both versions is fairly simple and can be seen in Figure 1.

Of importance is the fact that the standard implementation relies on access to a centralized server, and does not offer any management of write/write conflicts. It is assumed that the value at the server is authoritative, and since any conflict must be handled without user intervention within the standard Android approach, the server adopts a "last write wins" strategy for dealing with conflict. This may result in incorrect data in the case of a write/write conflict since the version in the server is selected based solely on the time of synchronization and not

based on any causal information. Of course this solution also has the disadvantage of being a fully centralized solution, which is not appropriate for a disaster management application.

Note that the sample code provided by Google does not really address these issues at all nor does it lead the developer to deal with issues of synchronization properly.

RAVEN Implementation

The implementation of the People Finder application that uses RAVEN was undertaken entirely on the phone. Using the "Schema Creator" application bundled with the framework we constructed a schema, which is logically equivalent to the schema used in our standard implementation. The resulting schema was then installed in the system as a new data type. The user interface for this version of the application was then generated based solely on the schema for the application by RAVEN.

Note that the schema created using our system is not seen by the user, but rather is only used internally within the system to construct the database. From this schema the framework is able to generate an edit and list user interface as described in the previous chapter. This user interface is shown in Figure 1.

Comparison

In order to validate the value of the framework we compare the results of these two applications from a qualitative perspective from the perspectives of development effort, usability and synchronization feature sets.

In terms of development effort, the RAVEN version requires no code. However, until we conduct usability studies as part of our future work we are not sure that training in schema construction can be avoided. In the case of the standard application almost 3500 lines of code and several days of development effort by an experienced developer were required. This kind of time is simply not available when disaster strikes. The framework version required just a few minutes to define the schema for the data to edit using the ``Schema Creator'' application and could be done without any specialized knowledge. Of importance for our desired application domain, the former can only be done using a laptop, which would require access to a power source for the several days of development time, while the latter can be accomplished using only the smartphone in a few minutes, allowing data oriented application development on a phone in the field. Nonetheless, RAVEN is able to offer rapid application development on the smartphone directly when disaster strikes and new data gathering efforts are required. In terms of usability, the custom application is better, since it is able to take advantage of more advanced widgets.

For example, it can be seen that the gender is able to be expressed in terms of images in the list view with the custom application while no such ability currently exists within our framework since we it does not currently have support for image based enumerations. While we could add support for such it is unclear how much this matters without further usability studies.

In terms of the overall feature set we argue that the framework version is superior because of the ability to support commit, signing and branch operations, as well as peer-to-peer discovery and synchronization. These features simply do not exist in the custom application and would require even more effort in order to add them, however they are vital to enabling the highest utility of gathered data when a disaster strikes. Of particular importance is the ability to synchronize over ad-hoc networks such as can be setup directly between multiple phones. The centralized server solution in the naive implementation is simply not appropriate for the domain. It also requires a great deal of code to be written as well as the running of a centralized server in order to provide synchronization at all. This makes such an approach of much less utility in a disaster situation where networking cannot be relied upon and where the accuracy of the data is important.

We thus believe that the framework is promising but realize that further validation with users is required in order to fully demonstrate the utility of our approach in a disaster situation.

CONCLUSIONS

In this paper we have examined the utility of RAVEN for data collection applications on smartphones using the construction of a simple disaster management application. We have implemented a "People Finder" application both with and without our framework.

The work presented in this paper has made it possible to define a schema for a structured data store at runtime and generate a user interface to edit instances of that schema on mobile devices. This system provides not only developers with tools to quickly create data oriented applications, but also enables users to easily create collaborative data oriented applications directly on the phone in a short amount of time. The system is able to

generate an edit user interface at runtime, allowing the system to support dynamic schemas for structured data stores, with editing and storage on mobile devices. Finally, the framework is able to take advantage of the inbuilt networking technologies of the smartphone devices without the need for fixed infrastructure, which is often damaged in a disaster scenario. Our hope is that ability to easily construct all kinds of data oriented applications using our framework will make it more likely to be pre-deployed on uses phones when disaster strikes.

The work presented in this paper is still in the early stages of development. Our future work will involve user studies with this framework to determine the usability of the framework and address any shortcoming we find. In addition we anticipate further engineering work to expand the expressivity of the framework to allow users to easily construct more advanced interfaces.

We saw in our evaluation that the hand-coded version of our application was able to offer a slightly richer version of the edit user interface components at the expense of considerable developer time. As part of our future work we would like to expand the types of widgets available to the system at runtime so that users can build richer data oriented applications directly on the phone. We also intend to add support for validation related properties to the Avro schema. This will allow the user interface components to validate and control input from the user at record creation time. Keeping the user interface simple is important in order for this to work well. Features we intend to support are validations like range limits on numeric values, string pattern matching, enforcement of non-null values, minimum and maximum number of entries in arrays, as well as required keys in a map. We intend to further drive the need for validation as we build additional applications using RAVEN. Of course the added complexity of such widgets and validations has to be balanced with the usability of the application creator directly. Again, we intend to evaluate the usability in order to strike the right balance so that the framework is expressive but simple. Finally, we also intend to explore making it easy to query the stored data using both structured and unstructured queries while focusing on ease of use.

REFERENCES

1. Amin, S., Goldstein, M. Data against natural disasters: establishing effective systems for relief, recovery, and reconstruction. World Bank Publications (2008).
2. Farnham, S., Pedersen, E., Kirkpatrick, R. Observation of Katrina/Rita Groove Deployment: Addressing So- cial and Communication Challenges of Ephemeral Groups In Process. In B. V. de Walle, M. Turoff, editors, Proceedings of the 3rd International ISCRAM Conference, pages 39–49 (2006).
3. Gachet, A., Haettenschwiler, P. A decentralized approach to distributed decision support systems. Journal of Decision Systems, 12(2):141–158 (2003).
4. Gomez, E., Turoff, M. Community Crisis Response Teams: Leveraging Local Resources through ICT E-Readiness. Proceedings of the 40th Annual Hawaii International Conference on System Sciences (2007).
5. Harrald, J. Agility and Discipline: Critical Success Factors for Disaster Response. The ANNALS of the American Academy of Political and Social Science, 604(1):256 (2006).
6. Jefferson, T., Harrald, J. Collaborative technology: providing agility in response to extreme events. International Journal of Electronic Governance, 1(1):79–93 (2007).
7. Kanjo E., R. D. L. P., Bacon J. Mobsens: Making smart phones smarter. EEE Pervasive Computing, 8:50–57 (2009).
8. Landgren, J., Nulden, U. A study of emergency response work: patterns of mobile phone interaction. Proceedings of the SIGCHI conference on Human factors in computing systems, pages 1323–1332 (2007).
9. Henry Lieberman, Fabio Paternò, Markus Klann and Volker Wulf. (2006). End-User Development: An Emerging Paradigm. Human-Computer Interaction Series, 2006, Volume 9, 1-8, DOI: 10.1007/1-4020-5386-X_1
10. Nicholas Palmer, R. K. T. K., Emilian Miron, Bal, H. Towards collaborative editing of structured data on mobile devices. In MDM 2011 - 12th International Conference on Mobile Data Management (2011).
11. Schafer, W., Carroll, J., Haynes, S. Emergency Management as Collaborative Community Work. In ECSCW 2005: Ninth European Conference on Computer-Supported Cooperative Work. (2005).
12. Wheatley, M. Goodbye, Command and Control. Leader to Leader, 5:21–28 (1997).