Building Mobile Sensor Networks Using Smartphones and Web Services: Ramifications and Development Challenges

Hamilton Turner, Jules White, Chris Thompson,
Krzysztof Zienkiewicz, Scott Campbell, and Douglas C. Schmidt
Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN 37203, USA
{jules, schmidt }@dre.vanderbilt.edu, {hamilton.a.turner, chris.m.thompson,
krzysztof.k.zienkiewicz, scott.h.campbell}@vanderbilt.edu

Section 1. Introduction

Creating a large-scale sensor network has traditionally required physically deploying and managing many customized sensor nodes. Harvesting sensor data efficiently has required developing complex networking techniques [4]. Moreover, once sensor data was collected, substantial effort was needed to process and visualize the data, or to take responsive actions. Physical upkeep of the sensor nodes required teams to visit and maintain sensors deployed in the field.

Modern smartphones are sophisticated computing platforms with complex sensor capabilities, such as detecting user location, recording high-quality audio, recording ambient light, sensing geomagnetic strength, and sensing orientation. Due to widespread use of smartphones, it is now possible to create large-scale sensor networks by deploying applications on end-user devices to collect and report sensor readings back to servers, using built in cellular network technology. End-users also have an inherent interest in maintaining their phones, alleviating researchers, operators, and users from much of this burden.

Apple has sold over 13 million iPhones [1] and roughly 1 million HTC Google Android-based phones have been sold. With this many smartphones on the market, the size of sensor networks could easily be in the hundreds of thousands. Much previous work on sensor networks, such as environmental monitoring and first-responder systems, can be adapted to mobile smartphones, and likely achieve more dispersion and adoption per unit of effort than conventional methods of deploying mobile sensor networks.

After sensor data has been collected, it must be processed, visualized, and shared with users. Web service application programming interfaces (APIs) are another emerging trend that can help in this task. For example, Google offers public services for geocoding addresses, sharing pictures and video, displaying maps, and overlaying data across satellite imagery. Moreover, some services, such as Google's App Engine and Amazon's EC2 compute cloud, offer free or low cost computational grids for analyzing data. Utilization and composition of web service APIs has allowed rapid application development.

Using web services, and the advanced computational power of smartphones, applications can contain real-time information filtered using the metadata of individual users (such as location, relation to wrecked user, and application settings). Data from multiple users can be combined in an arbitrarily complex manner, and used in conjunction with available web services to create powerful applications involving real-time, location-aware content. The combined data can also be shared through content distribution networks, such as YouTube.

This chapter presents the challenges and benefits of developing large-scale sensor-based applications using smartphones and web services. First, we discuss how smartphones, along with web services, can be used to aggregate complex sensor data provided by users. Second, we address openly available web services for displaying the data, performing complex calculations, and sharing it with other smartphone users and first responders. Next, we present the challenges of using publicly available services and untrusted end-user smartphones for mission critical applications, particularly the challenges resulting from services not guaranteeing reliability. Finally, we look at the software engineering ramifications of building sensor-based applications on smartphone platforms and public web services.

Outline of the Rest of the Proposed Chapter

Section 2: Motivating Example. To motivate benefits and challenges of building a large-scale sensor-based application from mobile phones and publicly available services, we will present *Wreck Watch*, a mobile phone application, built on the Google Android platform, for detecting car wrecks in real time. Wreck Watch leverages a phone's accelerometer and GPS data to detect potential collisions. These collision events are transmitted to a central server for accident analysis and response.

The Wreck Watch server provides map-based plotting of accidents through Google Maps. Moreover, it allows accident victims and bystanders to share pictures and video of the accident location with first responders. In the event of high-speed collisions, first responders can be contacted automatically and the application can share data, such as route traveled before the crash, maximum acceleration during the wreck, and the status of the mobile device (can it still be used to contact the individual, or is it unusable). Users can also specify contacts to notify after a wreck, either from the server via text message or email.

Section 3: Ramifications in Developing Mission Critical Applications by Reusing Publicly Available Web Services and Mobile Computing Platforms. This section addresses the following ramifications that arise when combining smartphone applications with web services, with a focus on powering mission critical applications with enterprise class web services:

1. *Complications involved with mixing mission-critical and enterprise class applications or services.* Most web services run on top of existing enterprise class infrastructures, allowing very high reliability and fault tolerance. For this reason, it can be assumed that services will be available, although this is not guaranteed. We will discuss techniques to minimize the risks involved with combining mission-critical and enterprise class systems, demonstrating how web service support from large sites, such as Google, are acceptable for most situations.
2. *Rapid, complex, interconnected service development can be achieved with low resource overhead.* Creating a third party application to tie together multiple web services can be completed quickly and easily. For example, displaying the hometown's of Facebook friends on a Google map simply requires connecting to both API's, retrieving the data from Facebook, and passing it to a Google map. The complexity of web services can be utilized to power large parts of the application's data processing and visualization, saving time and money. We will discuss potential applications and ramifications of combining these services, as well as detailing some existing combinations available on smartphones.
3. *Utilization of existing services conserves resources, but restricts flexibility.* A consequence of using web service APIs is a loss in customization of what the API provides. Many of the available APIs are not open-source, and an API service client (a developer using the API) may not have control over how a particular service is implemented or what services are offered. We will discuss patterns used to supplement services, and methods to combine various services to create a more cohesive final product.
4. *Common distribution methods promote application distribution.* The Apple Store, the Android Market, and the Palm App Catalog are examples of a uniform application distribution method. Although some devices come with the ability to install applications from non-standard locations, it is discouraged. These prevalent distribution methods promote application attention on a greater scale than simply releasing the application.
5. *Technological advances in smartphones allow increased flexibility.* The networking and computational power of smartphones has increased greatly in the last decade, and many smartphones are now coming with a plethora of advanced sensors built in. Common sensors include an accelerometer, an ambient light sensor, a proximity sensor, and of course a microphone. Some devices contain even more sensors, including a geomagnetic sensor and an orientation sensor. These increased computing, sensing, and communication abilities open up many unexplored areas of development, especially research pertaining to large scale sensor networks.
6. *Smartphone platforms and architectures allow new features to be combined with existing applications quickly*. Many smartphone platforms provide methods to use existing applications or features of the device as part of the currently running application. This design allows rapid development, similar to web services, but with local, rather than remote, services. We will cover Google Android's Intent system in detail, detailing how it promotes interconnected application development and code reuse.

Section 4: Social, Security, Privacy and other Issues with Using Crowd-sourced Sensor Data and Help. Pulling sensor data from unverified users (the 'crowd') introduces security concerns. For users of the smartphones, privacy could cause hesitation to adopt a new technology, as well as other social concerns. Using sensors costs battery life, among other things, and a user may not want their phone utilized in a first responder scenario, as they may prefer to have full control in that environment. Although some research has been performed on these concerns, there still remains a large number of unexplored areas [2,3,4,5,6]. We will discuss potential user responses to their smartphone being networked, responses to perceived and real privacy and security invasions, and verification of crowd-sourced data.

Section 5: Related Work. This section will describe related work on the topics of commercial-off-the-shelf (COTS) products (such as web services), smartphone use in mission-critical applications, and social considerations with typical mobile device software (privacy, security, hardware utilization, etc). This section will investigate topics ranging from error recovery techniques [7] to service composition approaches [8].

Section 6: Concluding Remarks. This section will present concluding remarks, summarize key lessons learned (both pro and con), and provide links to open-source project source code and demo examples.

Relevance for this Book

The integration of smartphone devices and web services is already a practical application development method. The large number of smartphones enables sensor networks at a scale that was previously hard to create and maintain. Combining these topics and identifying ramifications for application developers is a key area for the immediate future. The issues involved with crowd-sourced application control are also under-researched, and will have a large impact on how effective applications are for mission-critical uses.

References

1. Betanews. Press Release. Apple: 13 M iPhones sold in total, 2.6 M Macs sold last quarter. October 2008. <http://www.betanews.com/article/Apple-13-M-iPhones-sold-in-total-26-M-Macs-sold-last-quarter/1224624147>
2. Banerjee, N.; Rahmati, A.; Corner, M.; Rollins, S. & Zhong, L.
   Users and batteries: Interactions and adaptive energy management in mobile systems
   Lecture Notes in Computer Science, Springer, 2007, *4717*, 217
3. Krieger, M.; Stark, E. M. & Klemmer, S. R.
   Coordinating tasks on the commons: designing for personal goals, expertise and serendipity
   CHI '09: Proceedings of the 27th international conference on Human factors in computing systems, ACM, 2009, 1485-1494
4. Zhang, H. & Hou, J., Maintaining sensing coverage and connectivity in large sensor networks
   Ad Hoc & Sensor Wireless Networks, 2005, *1*, 89-124
5. Lyytinen, K. & Yoo, Y., Issues and challenges in ubiquitous computing: Introduction, *Communications of the ACM*,45:12, pgs 62—65, 2002
6. Consolvo, S. et al., Location disclosure to social relations: why, when, & what people want to share, Proceedings of the SIGCHI conference on Human factors in computing systems, April, 2005, Portland, Oregon
7. Tartanoglu, F., Issarny, V., Romanovsky, A., and Levy, N., Coordinated forward error recovery for composite web services, In Proceedings of the Symposium on Reliable Distributed Systems, October, 2007, Beijing, China
8. Ponnekanti, S.R. & Fox, A., Sword: A developer toolkit for web service composition, In Proceedings of the Eleventh International World Wide Web Conference, Honolulu, HI, May, 2002

Author Biographies

Hamilton Turner is a Research Assistant at Vanderbilt University. He has worked with the Institute for Software Integrated System (ISIS) for two years, and currently focuses on Mobile Computing. He is working on his B.E. in Computer Engineering from Vanderbilt University, and has previously focused on Unit Testing Non-Functional Properties of Large Scale Computing Systems.

Dr. Jules White is a Research Assistant Professor at Vanderbilt University. He received his BA in Computer Science from Brown University, his MS in Computer Science from Vanderbilt University, and his Ph.D. in Computer Science from Vanderbilt University. Dr. White's research focuses on applying a combination of model-driven engineering and constraint-based optimization techniques to the deployment and configuration of complex software systems. Dr. White is the project leader for the Generic Eclipse Modeling System (GEMS), an Eclipse Foundation project.

Chris Thompson is an undergraduate Research Assistant at Vanderbilt University. He is currently pursuing a Bachelor of Engineering degree in Computer Engineering. He is currently focusing on mobile device performance analysis and in the past has worked with constraint-based deployment configuration and automating hardware and software evolution analysis.

Krzysztof Zienkiewicz is an Undergraduate Research Assistant at Vanderbilt University. He works for the Institute for Software Integrated Systems (ISIS), focusing on application development on mobile devices. He is currently working on his B.S. in Computer Science and Mathematics from Vanderbilt's School of Engineering and School of Arts and Science respectively.

Scott Campbell is a Graduate Research Assistant at Vanderbilt University. He has worked with the Institute for Software Integrated System (ISIS) for two years, and currently focuses on Mobile Computing. He received his BS in Computer Science from Vanderbilt University, and has previously focused on remote collaboration in the Generic Eclipse Modeling System (GEMS).

Dr. Douglas C. Schmidt is a Professor of Computer Science and Associate Chair of the Computer Science and Engineering program at Vanderbilt University. He has published 9 books and over 400 papers that cover a range of topics, including patterns, optimization techniques, and empirical analyses of software frameworks and domain-specific modeling environments that facilitate the development of distributed real-time and embedded (DRE) middleware and applications. Dr. Schmidt has over fifteen years of experience leading the development of ACE, TAO, CIAO, and CoSMIC, which are open-source middleware frameworks and model-driven tools that implement patterns and product-line architectures for high-performance DRE systems.