Engineering Challenges of Deploying Crowd-based Data Collection Tasks to End-User Controlled Smartphones

Hamilton Turner
Virginia Tech
Blacksburg, VA, USA
hamiltont@vt.edu

Jules White
Virginia Tech
Blacksburg, VA, USA
julesw@vt.edu

Jeff Gray
Univ. of Alabama
Tuscaloosa, AL, USA
gray@cs.ua.edu

ABSTRACT

The emerging hardware resources available in smartphones has increased the potential for effective crowdsourced data collection. Multiple researchers have already created smartphone applications to enable crowdsourced data collection. However, given the recent emergence of smartphone-based data collection systems, software engineering techniques specific to measuring or improving performance for these systems have yet to be developed. In this paper, we propose applying recent advances in deployment optimization to the problem of effectively deploying data collection tasks to smartphone systems, and describe how task deployment can be useful for optimizing data collection metrics such as data coverage and data accuracy. We introduce challenges specific to deploying data collection tasks to end-user controlled smartphones and present initial solutions to the various identified challenges.

Keywords

smartphone, data collection, crowdsourcing, deployment optimization

1. INTRODUCTION

Large-scale information gathering requires an expensive, top-down coordinated effort. For example, a United States census is only completed once every ten years, requires massive coordination in order to be successful, and costs are in excess of two billion dollars [3]. Mobile technologies have shown promise in reducing both the manual effort and the cost associated with large-scale information gathering. For example, numerous smartphone applications have been developed to enable citizens in the Gulf Coast region to report field data, such as images showing environmental impact, text describing animal behavior, and location-tagged deceased animal sightings, thereby providing scientists with a wealth of data at a significantly lower cost than traditional information gathering approaches[5, 12, 19]. Mobile devices

such as smartphones have enabled Brazil to complete a census of "unprecedented [...] accuracy, reach, speed and cost effectiveness," where 48% (92.7 million people) of the population was counted within 30 days of data collection[6].

There are several motivations for using smartphones (e.g., Apple iPhone, Google Android, or Windows Phone 7) to power crowdsourced mobile data collection. For example, the number of consumers carrying smartphone devices is rapidly increasing. Smartphone sales increased by 96% worldwide from 2009 to 2010 [1]. Moreover, smartphones possess a variety of sensors, including: accelerometer, camera, GPS, ambient light, temperature, geomagnetic, proximity, microphone, and others. Moreover, smartphones have significant local processing and storage capabilities, a number of communication mechanisms (e.g., WiFi, 3G, EDGE, or Bluetooth) and end-users perform smartphone maintainence and upkeep (e.g., including frequent recharges). Smartphones can be programmed in high-level languages such as C#, Java, and Objective-C, allowing rapid application development and enabling reuse of libraries which provide complex functionality, such as image analysis. Due to these attributes, researchers have developed smartphone applications for a number of crowdsourced activities, such as tracking user activities for health purposes [14], tracking and analyzing CO_2 emissions [4], detecting traffic accidents and providing situational awareness services to first responders [16, 8], measuring traffic [13], and monitoring cardiac patients [10].

Although smartphones crowdsourcing applications have significant potential , and a number of applications have been developed to enable crowdsourced data collection, few researchers have investigated software engineering techniques for mapping data collection tasks to a group of smartphone computing resources. Current data collection systems are typically *laissez-faire*, with users entering data without guidance regarding the usefulness of that data. We propose to optimize the crowdsourced data collection process, by deploying data collection tasks to end-user smartphones in a manner that would positively impact system data collection goals. Numerous smartphone properties can be used to optimize the mapping of data collection tasks to end-user smartphones in order to increase the quality of data collection results. For example, smartphone properties such as the geographic distribution of the smartphones throughout the area of interest and rapid changes in smartphone resource availability can be included in a data collection task deployment algorithm, so that deployed tasks are more probable to be completed accurately and rapidly. Additionally, data

collection task properties can be considered when optimizing the mapping, such as tasks that yield better results on devices that have a high resolution camera, tasks that require a fast CPU to handle intensive data processing, or tasks that require a fast network connection in order to transmit large amounts of data. Manually accounting for these complex considerations when determining how to map data collection tasks to computing resources is hard.

Software engineers have faced similar challenges in the field of deployment optimization, which aims to map software components to hardware nodes in a manner that optimizes system metrics of interest. By modeling data collection tasks as software components, and end-user computing resources as hardware nodes, software engineering techniques ,created for use in deployment optimization challenges, can be re-purposed to deploy data collection tasks to end-user smartphones. For complex systems with a large number of software components and hardware nodes, traditional deployment optimization algorithms can take upwards of minutes, hours, or even days, to find an optimal deployment mapping. Although the speed of optimization algorithm execution is a secondary priority for systems with static properties or long development cycles, such as optimizing software component deployment for an airplane, in a smartphone powered data collection system this is unacceptable. Properties such as smartphone distribution can change rapidly, and frequent, rapid re-deployments may be needed. By using a combination of metaheuristic and heuristic algorithms, researchers in the field of deployment optimization have been able to reduce the time required to under a minute, making frequent re-deployments a possibility[20].
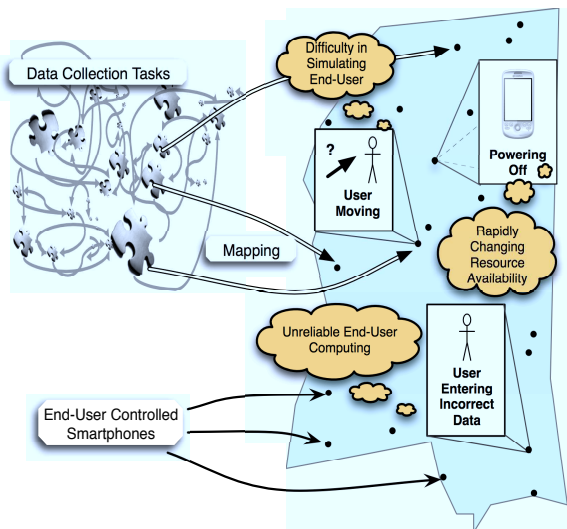

Figure 1: Challenges in Mapping Data Collection Tasks to Smartphone Hardware

However, there are multiple new challenges that arise when attempting to deploy data collection tasks to end-user controlled smartphone resources. Some of these challenges, such as the unreliability of end-users in completing assigned tasks, are shown in Figure 1. Section 3 explains these challenges in further detail.

Paper organization. The remainder of this paper is organized as follows: Section 3 outlines challenges of optimizing data collection by properly mapping data collection tasks to end-user smartphones; Section 4 approaches these challenges and discusses future work and initial solutions for these challenges; and Section 5 presents concluding remarks.

2. MOTIVATING EXAMPLE: CONTINUOUS MEASUREMENT OF CELLULAR NETWORK COVERAGE

In order to motivate the challenges and benefits associated with deploying data collection tasks to smartphones, we present a motivating example for monitoring real cellular network coverage using end-user smartphones. Current cellular network coverage maps are generated by a combination of manual field measurements and advanced mathematical analysis, including propagation modelling, interference modelling, traffic modelling, and mobility modelling [7]. These generated network coverage maps are generalizations and cannot be used as a guarantee of the exact signal quality that a specific cellular chip will experience at a specific location. Moreover, the field measurements required to verify and improve a generated network coverage map make this approach invalid in situations where rapid regeneration of the coverage map is required, such as disaster situations or lightning strikes on network base stations.

A method of rapidly and accurately generating cellular network coverage maps would have a number of advantages. Key advantages would include the ability for current propagation models to be juxtaposed with real-world data to be either verified or improved, the ability for cellular service providers to analyze and react to areas of poor coverage, and the ability for consumers to have confidence in the network coverage map for their region and with the specific mobile phone hardware they purchase. One method of generating an accurate map of cellular network coverage involves creating a smartphone application that allows users to input data readings, and aggregating the results of numerous such data readings into a generated cellular network coverage map. Each data reading could contain the perceived cellular network signal strength, the location the reading was taken, and other meta information such as the chipset used and the hardware model and software version numbers. The architecture of such a system is shown in Figure 2.

First, a smartphone application must be programmed and deployed to numerous smartphones, which are preferably well geographically distributed. As smartphones are moved geographically, they use the built-in cellular chipsets to sample the cellular coverage in their region. These samples of coverage are combined with other meta information, such as the location and time of each reading, the network chipset on the smartphone, and the particular operating system version being run. When a smartphone's internet connection is enabled, these data readings can be transmitted to a centralized cloud-based data aggregation server. As readings are entered, data aggregation algorithms are used to combine the data into a smaller format that can then be represented as a coverage map. System policies can be used to control various aspects of the process, such as the rate of data collection, the speed at which data becomes outdated, or the method used to aggregate the incoming data.
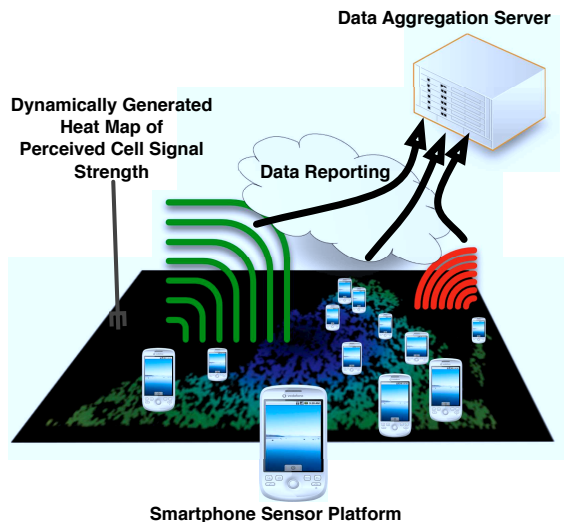
Figure 2: Architecture of Smartphone Data Collection System for Generating Cellular Network Coverage Maps

3. CHALLENGES OF MAPPING DATA COLLECTION TASKS TO SMARTPHONES

This section discusses the challenges of effectively deploying software tasks to end-user controlled smartphone hardware resources. Solving these challenges is key to developing engineering approaches for effectively deploying data collection tasks to end-user smartphones.

3.1 Challenge 1: Rapidly Changing Resource Availability

Traditionally, deployment optimization is done during the design phase of the software lifecycle. While this makes sense in a system such as an aircraft, where neither the hardware or software components will change substantially during the production stage, in a smartphone based data collection system hardware properties change frequently. For example, smartphones can join the data collection system frequently, and can leave the system just as quickly, as shown in Figure 1. Moreover, smartphone properties such as location can be drastically changed in a short period of time. Therefore, performing a mapping of software tasks to hardware resources would likely only present an optimal solution for a short period of time, before the natural volatility of the resource properties in the data collection system reduced the quality of the mapping of data collection tasks to specific smartphones. However, recalculating an efficient mapping too frequently would waste CPU resources, as one deployment might still be executing when another started. Conversely, calculating the deployment too infrequently could lead to significant departures from the optimal deployment because smartphone resource properties would have changed substantially. For example, many of the smartphones may have changed their geographic area, and would be unable to complete their assigned data collection task.

Additionally, the networking and power restrictions on smartphones can introduce challenges to the traditional centralized deployment scheme. For example, the power re-

quired to frequently update a central server of a mobile device's geographical location, which would require both network traffic and GPS operation (which are both power-intensive operations), may be overly greedy and waste smartphone battery power. Moreover, smartphones in a crowd-sourced mobile data collection system are likely to only be accessible to a deployment server for a short period of time, as smartphone users are likely to disable the internet when not directly using it, or to travel in and out of internet range. Therefore, calculating an optimal task to hardware mapping in response to a smartphone joining the system may not be a viable option, as that smartphone resource could leave the network before the deployment calculations are complete and downloaded to the device (smartphone devices often have slow internet speeds). Moreover, in various information gathering scenarios (e.g., disaster impact analysis), end-users may travel outside locations where internet is available, and valuable opportunities to collect novel information could be missed due to an outdated deployment model that cannot be updated due to a lack of connectivity to the deployment server.

One example of this challenge involves a user that is currently involved with a data collection system going for a hike. As the user, and his or her smartphones by association, leave urban environment the data collection system detects the device is entering an area with few measurements, and performs a deployment of data collection tasks that encourage the device to frequently sample the cellular signal strength as the user travels. However, in a potentially dangerous situation such as the user becoming lost while hiking, it is critical that the system does not consume precious battery power. Moreover, if the device seems to be travelling towards a location where data readings are sparse, such as a hiking trail, and loses network connectivity (which can frequently happen in remote regions, especially in mountainous areas) in transit, then a change of direction may go unnoticed. In this case, the smartphone hardware may end up generating a number of useless data readings that consume smartphone resources such as battery, network throughput, or CPU power.

3.2 Challenge 2: Unreliable End-user Computing Resources

In traditional smartphone based systems that utilize deployment optimization, software components that have been mapped to hardware nodes can be assumed to eventually execute, assuming that there are no errors in the deployment optimization constraint enforcement. However, when deploying data collection tasks to end-user controlled smartphone hardware, there is no guarantee that a data collection task will be completed. Moreover, even if a data collection task is completed, there is no guarantee about the timeliness of data reporting, e.g., the central server may not receive the data reading until the smartphone is in range of network services and the user has enabled internet access. Additionally, there is little guarantee for the quality or veracity of the data collection task results. While traditional deployment optimization can assume that software operates favorably, smartphone applications can be modified with fairly limited technological ability, and an end-user can be generating invalid data either maliciously or unknowingly. Faulty smartphone sensors may also result in invalid data readings. These issues make it difficult for a smartphone data collec-

tion system to predict how much data will be collected by the current distribution of data collection tasks on end-user smartphones, because some of the tasks may never be completed and some of the tasks may be completed incorrectly. This unreliability of the end-user controlled computing resources complicates software to hardware mapping, and systems must attempt to compensate for unreliable or inaccurate input from end-users.

For example, in generating a cellular network coverage map using a smartphone powered data collection system, a smartphone may be able to record a number of cellular signal strength readings, but be unable to upload those readings to the central server until an Internet connection is established. During this interval, other phones may be tasked with locating measurements in the same region in order to fill a perceived gap in data coverage that does not truly exist. Moreover, there is significant motivation for an individual or a group of individuals to attempt to game the data collection results to make a generated network graph appear better or worse than actuality. While removing the effect of a single inconsistent smartphone, such as a smartphone with a sensor error, is relatively simple, a focused group attempting to manipulate the data results is harder to detect and to manage.

3.3 Challenge 3: Optimizing System-Level Goals using Metrics

In traditional deployment optimization, the optimization algorithms can include metrics that are optimized in some manner, such as attempting to reduce overall system power consumption. However, in a mobile data collection system it is difficult to know how much information is required to ensure an accurate picture of the real environment. For example, how many measurements should be taken at location x, before the data collection system can reasonably assume that it knows the true value at location x? Multiple factors complicate this problem, such as properties of the sensors (e.g., sensor error margins), the volatility of the environment at x (e.g., is the value at x more static or more dynamic?), and system configuration policies (e.g., will the data reading be reported immediately or only when convenient?). Many environments of interest, such as military intelligence data collection, environmental disaster impact monitoring, and demographic information monitoring, are constantly changing value. Moreover, an optimized deployment algorithm should not only optimize for the current system, but should ensure that future data collection chances are not wasted by prematurely using up smartphone hardware resources. Avoiding this wastefulness is complex due to the number of emergent properties in a system where each smartphone is carried and controlled by an end-user. The complexities of measuring correctness of the data and predicting the future effects of data collection task deployments makes it challenging to optimize system-level goals, such as obtaining 80% data coverage as rapidly as possible, by effectively deploying data collection tasks to smartphones.

One situation where this challenge may arise is in generating a cellular network coverage map using a smartphone powered data collection system. In such a system, it is difficult to know which regions of the environment require more data readings before they can be accurately predicted by the system. A rural environment with few base stations may have a relatively static local coverage signal and only re-

quire a data reading once a month, while a region contained within multiple cellular base stations and surrounded with multiple forms of interfering bodies such as buildings may require fresh readings every few minutes. Additionally, a weather front can rapidly invalidate a large number of data readings. It is challenging for the data collection system to understand which regions are actually incorrect and need more data collection tasks assigned to them and which regions are under a temporary influence and will return to normal shortly without any extra data collection readings. Moreover, it is difficult for a data collection task deployment algorithm to know which tasks should be assigned to which nodes in order to produce desired results, such as more data readings from a specific region.

3.4 Challenge 4: Difficulty in Simulating End-user Computing Resources

As with many mobile systems, field testing of smartphone-powered systems can be a challenge and require significant amounts of time [9]. Many mobile systems avoid the complications of field testing by using modelling and simulation. However, smartphone-powered system properties are heavily dependent on smartphone owners. Although some human actions, such as mobility patterns, have been shown to be highly predictable, there is no similar research on other properties, such as enabling or disabling smartphone sensors[15]. An important constraint in many smartphone based systems is avoiding over-utilization of the participants (e.g., asking too much of volunteers). For example, users have been shown to be conscious of their battery usage, and to be willing to take steps to extend the number of tasks they can perform before draining their battery[17]. There is little available data on the impact of mobile data collection software systems on everyday lives, making it difficult for a deployment algorithm to quantify the effect of various data collection tasks on smartphone users. Preliminary information on the impact of mobile software was published in [2]. However, the number of unknown emergent properties makes modelling and simulation challenging for smartphone based systems.

For example, in generating a cellular network coverage map using a smartphone powered data collection system, it is difficult to identify metrics of interest, such as the minimum number of smartphones required to ensure some high-level system goal e.g., 80% coverage of a region. For the reasons listed in the previous paragraph, it is challenging to simulate this system. Without the ability to determine such high-level metrics at design time, many smartphone powered data collection systems may never be developed past the design stage.

4. MAPPING CROWDSOURCED DATA COLLECTION SOFTWARE TO SMARTPHONE HARDWARE

This section approaches the challenges introduced in Section 3 and discusses future and ongoing work in these areas that provide an initial strategy for addressing the emerging concerns.

4.1 Utilizing Incremental Deployment to Deal with Rapidly Changing Network Graphs

Section 3.1 mentions three issues that suggest an incre-

mental deployment model would be a valuable asset. Delaying final deployment decisions would allow the most recent properties of the smartphone to be considered. For example, suppose a user connects to the main system and downloads a partial deployment model. The user then disconnects and goes for a hike. Along the way, the system notices the user is passing a particularly sparse region of coverage, and performs a re-deployment in order to capitalize on the user's location.

The desirable traits of an incremental solution would be a core set of deployment goals that are infrequently changed, and the ability to rapidly apply changes to the solution. One potential approach is a geographic decomposition of the deployment (e.g., have a deployment for the entire geographical area that takes significant time to update and smaller models to limited geographical regions that can be updated rapidly). This mix would allow the overall system goals to be preserved while also allowing personalized task deployment for each peer.

4.2 Integrating User Reputation into Deployment Algorithms to Increase Reliability of End-user Computing Resources

Section 3.2 introduces the issue of unreliability of end-user controlled smartphone resources. One large issue presented is detection of a user entering incorrect information, either through ignorance or deliberate attempts to misinform the data collection system. One common method of validating user input in a social collaboration system is to have multiple users feed the system data from the same problem, task, or environment. In this manner, anomalous users can be detected and removed either automatically or by other users. The reCAPTCHA system effectively utilizes this approach – invalid entries are easily detectable relative to the millions of users entering the correct values. However, this kind of statistical validation is only successful when the number of malicious entities in the system is minimal relative to the total number of entities. For reCAPTCHA, even attempts by large groups to enter incorrect information will fail as long as the majority of users are entering correct values[18].

Unfortunately, in a mobile collaboration system the number of users that can be compared to each other is often limited by the geographic location of each user. For example, in an information gathering system, information from two different users in vastly different locations cannot be compared, as shown in Figure 3a. This reduces the ability to compare users, and makes it easier for malicious users to control key geographical locations.

One strategy for dealing with this issue is to implement behavioral strategies into the deployment algorithm. For example, "trustworthy" users could be allocated more critical tasks. Also, users can be allocated similar tasks in an attempt to refine the trust model of each user. This strategy is shown in Figure 3b. Although this still relies on the majority of users in the system being trustworthy, and would work best with systems having a slow rate of change and high rate of user mobility, this strategy would make it more difficult to maliciously control the system.

The deployment system can accept as input the given knowledge about each user, and either utilize that knowledge (e.g., send this trusted user to region x) where the system needs information, or attempt to increase that knowledge (e.g., task this user so that they can later be compared to
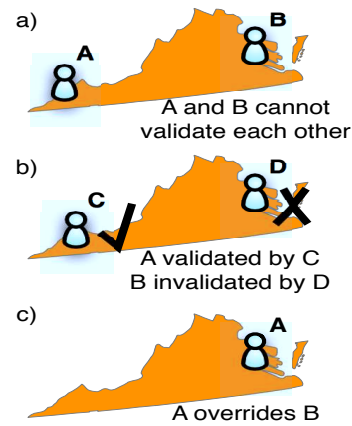

Figure 3: Validating Users By Reputation

an already trusted user). As more peer collaboration systems emerge, field data would allow the creation of a group of standard models that can be initially used as an individual's habits are learned by the system.

4.3 Using Statistical Confidence to Measure Information Coverage and Rate of Change

As mentioned in Section 3.3, two critical feedback metrics for a mobile collaboration deployment optimization algorithm are the information coverage and the rate of change estimation. By comparing incoming values from similar locations, it would be possible to determine if the incoming values in a particular location had converged. If a location can be determined to have converged, then by observing the time between convergences at the same location, it would be possible to determine the rate of change estimation for that location. Although this would obviously not work for all locations, the lack of a convergence would itself be an indicator that a particular area has a high rate of change. In particular, this approach would likely be effective at assisting the deployment algorithm in avoiding over-measuring areas that do not change rapidly, and allow it to focus resources on areas that are unclear. However, this assumes a system that is fairly steady with respect to the rate of change in various locations, and would not work for systems where the entire graph was highly variable, such as monitoring a short radius around a violent weather event.

4.4 Using Small-scale Field Trials To Test Deployment Optimization Effectiveness

Section 3.4 discusses the difficulty software engineers face in testing mobile deployment optimization effectiveness, largely due to the challenges in modelling the human components of the system. Trace data from social collaboration systems is readily available, and can be utilized while data for mobile social systems are entering production. For example, data from Wikipedia has been used by scientists to determine if participatory journalism is a reliable method[11]. In an attempt to capture the human element, small scale trials with the assistance of testbeds may be a viable option. For example, the Cognitive Radio Network Testbed at Virginia Tech allows new application technologies to be easily field tested. By utilizing this approach, the deployment

effectiveness can be tested in a real environment.

5. CONCLUSION

Crowdsourced mobile data collection is an emerging approach for large-scale data collection. Multiple researchers have focused on creating crowdsourced systems, but little research has been performed on the software engineering techniques for ensuring that crowdsourced systems achieve specific goals, such as collecting accurate environmental data. Software data collection tasks are often mapped to inadequate or suboptimal end-user smartphone hardware resources. The quality of the mapping is affected by multiple properties, which makes the mapping challenging. Given the high potential and large complexity of crowdsourced systems, the software engineering problems posed in crowdsourced data collection represent a growing problem as mobile computation increases.

6. REFERENCES

[1] Gartner says worldwide mobile phone sales grew 35 percent in third quarter 2010; smartphone sales increased 96 percent. `http://www.gartner.com/it/page.jsp?id=1466313`, November 2010.

[2] M. Bell, M. Chalmers, L. Barkhuus, M. Hall, S. Sherwood, P. Tennent, B. Brown, D. Rowland, S. Benford, M. Capra, et al. Interweaving mobile games with everyday life. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 417–426. ACM, 2006.

[3] B. Edmonston and C. Schultze. *Modernizing the US census*. National Academies, 1995.

[4] J. Froehlich, T. Dillahunt, P. Klasnja, J. Mankoff, S. Consolvo, B. Harrison, and J. Landay. UbiGreen: Investigating a mobile tool for tracking and supporting green transportation habits. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1043–1052. ACM, 2009.

[5] A. Gahran. Reporting on the gulf oil spill from your cell phone. `http://articles.cnn.com/2010-06-11/tech/oil.spil.app_1_cell-phones-apps-geotagged?_s=PM:TECH`, 2010.

[6] M. Gunther. Brazil's all-digital networked census. `http://newsroom.cisco.com/dlls/2010/ts_100410.html`, Oct. 2010.

[7] S. Hamalainen, H. Holma, and K. Sipila. Advanced WCDMA radio network simulator. In *Personal, Indoor and Mobile Radio Communications*, volume 2, pages 951–955.

[8] W. Jones. Forecasting traffic flow. *IEEE Spectrum*, 38(1):90–91, 2001.

[9] T. Kallio and A. Kaikkonen. Usability testing of mobile applications: A comparison between laboratory and field testing. *Journal of Usability Studies*, 1:4–16, 2005.

[10] P. Leijdekkers and V. Gay. Personal heart monitoring and rehabilitation system using smart phones. In *Proceedings of the International Conference on Mobile Business*, page 29, 2006.

[11] A. Lih. Wikipedia as participatory journalism: Reliable sources? metrics for evaluating collaborative media as a news resource. *Nature*, 2004, 2003.

[12] W. Park. Gulf oil spill apps let you track and report on bp deepwater horizon disaster. `http://www.intomobile.com/2010/06/01/gulf-oil-spill-apps-let-you-track-and-\` `\report-on-bp-deepwater-horizon-disaster/`, June 2010.

[13] G. Rose. Mobile phones as traffic probes: practices, prospects and issues. *Transport Reviews*, 26(3):275–291, 2006.

[14] T. Saponas, J. Lester, J. Froehlich, J. Fogarty, and J. Landay. iLearn on the iPhone: Real-Time Human Activity Classification on Commodity Mobile Phones. *University of Washington CSE Tech Report UW-CSE-08-04-02*, 2008.

[15] C. Song, Z. Qu, N. Blumm, and A. Barabasi. Limits of predictability in human mobility. *Science*, 327(5968):1018, 2010.

[16] C. Thompson, J. White, B. Dougherty, and D. C. Schmidt. Optimizing Mobile Application Performance with Model-Driven Engineering. In *Proceedings of the 7th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, 2009.

[17] K. Truong, J. Kientz, T. Sohn, A. Rosenzweig, A. Fonville, and T. Smith. The design and evaluation of a task-centered battery interface. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 341–350. ACM, 2010.

[18] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465, 2008.

[19] J. White. clearmobile: Cloud environmental analysis and relief. `http://code.google.com/p/clearmobile/`, July 2010.

[20] J. White, B. Dougherty, C. Thompson, and D. Schmidt. ScatterD: Spatial Deployment Optimization with Hybrid Heuristic/Evolutionary Algorithms. 2010.