On Heterogeneity in Mobile Sensing Applications Aiming at Representative Data Collection

Henrik Blunck
Aarhus University
Aabogade 34
Aarhus, 8200 DK
blunck@cs.au.dk

Niels Olof Bouvin
Aarhus University
Aabogade 34
Aarhus, 8200 DK
bouvin@cs.au.dk

Tobias Franke
German Research Center for Artificial Intelligence
Trippstadter Strasse 122
Kaiserslautern, 67663 DE
Tobias.Franke@dfki.de

Kaj Grønbæk
Aarhus University
Aabogade 34
Aarhus, 8200 DK
kgronbak@cs.au.dk

Mikkel Baun Kjærgaard
Aarhus University
Aabogade 34
Aarhus, 8200 DK
mikkelbk@cs.au.dk

Paul Lukowicz
German Research Center for Artificial Intelligence
Trippstadter Strasse 122
Kaiserslautern, 67663 DE
Paul.Lukowicz@dfki.de

Markus Wüstenberg
Aarhus University
Aabogade 34
Aarhus, 8200 DK
markus@cs.au.dk

UbiComp'13 Adjunct, September 8–12, 2013, Zurich, Switzerland.

http://dx.doi.org/10.1145/2494091.2499576

Abstract
Gathering representative data using mobile sensing to answer research questions is becoming increasingly popular, driven by growing ubiquity and sensing capabilities of mobile devices. However, there are pitfalls along this path, which introduce heterogeneity in the gathered data, and which are rooted in the diversity of the involved device platforms, hardware, software versions and participants. Thus, we, as a research community, need to establish good practices and methodologies for addressing this issue in order to help ensure that, e.g., scientific results and policy changes based on collective, mobile sensed data are valid. In this paper, we aim to inform researchers and developers about mobile sensing data heterogeneity and ways to combat it. We do so via distilling a vocabulary of underlying causes, and via describing their effects on mobile sensing—building on experiences from three projects within citizen science, crowd awareness and trajectory tracking.

Author Keywords
Mobile Sensing, Heterogeneity, Data Collection, Representativeness

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous.

Introduction

The advent of rich sensors in common smartphones has created new possibilities for using mobile sensing [17] or mobiscopes [4] to gather data about the world, e.g., reflecting users' actual rather than reported behavior. This has found use in areas as diverse as traffic monitoring [12], leisure activities [9] and air pollution control [22], as well as in a rich and growing set of social networking applications [18]. Concomitantly, the rise of "Big Data" has kindled the hope that we may have the tools to analyse large and rich data sets. However, the quality of any analysis rests on the quality of the data being analysed, and here a number of compounding factors may adversely affect validity.

A data collection campaign using mobile sensing is based on questions selected by involved stakeholders. This could include questions such as: *When and where is traffic congested?*, or *What is the level of crowdedness in individual areas during a major event?*. Most questions will refer to a target group thereby pinpointing the people that could provide the most informative data about the questions, e.g., car drivers or event participants. To facilitate the collection via a mobile sensing app, campaign stakeholders have to make a number of decisions with the target group in mind with regards to promotion, user benefits, interface design, technical choices and development process. The choices made will affect the quality of data in the end. For instance, promoting a mobile sensing app as the official app of a public event may increase uptake, designing an app in a style that appeals to the target group may positively impact usage, enabling the app on the most popular device platforms may boost downloads or a participatory design approach may align the app better with users' preferences and wishes. Cramer et al. [8] have previously motivated some of these issues from a perspective of doing ubicomp research where as we in this paper in a more structured manner focus on the quality of the resulting data.

An important concern of data collection is data quality. Collected empirical data is often incomplete, sometimes inaccurate, and generally prone to biases, e.g., due to the collection methods or strategies. For empirical data involving humans, further biases may arise from situational parameters, and thus by choices of collection procedure, which prescribe in which situations data is collected. E.g., when using interviews or questionnaires, the collected data may be biased by the type, amount, and focus of the interview questions as well as by any biases and lapses of the participants' memory. For human surveying, biases arise from the scheduling of the (time-limited) observations, as well as by the focus of the observer. Mobile sensing as a collection method reduces the influence of the latter by allowing continuous collection of a broad spectrum of information. Still, also mobile sensing is prone to biases: E.g., existing work has highlighted the problems of heterogeneity in sensor quality across smartphone types [16] and variations in availability of data [4]; additionally, the biases mentioned above for traditional surveying have to be taken into account, in case of participatory sensing, i.e. when also explicit user input data is to be gathered; proposals to address the latter issue include usage of the experience sampling method within mobile apps[7].

In this paper, we discuss the above issues as sources of data heterogeneity, and introduce a taxonomy of heterogeneity types grouped into the main categories of device, user and project heterogeneities identified throughout three case studies of community-based mobile sensing apps. Examples of device heterogeneity include

missed opportunities for participatory sensing due to sluggish UIs or failing apps due to OS updates, examples of user heterogeneity include difficulties in getting a representative demographic distribution of users and examples of project-internal heterogeneity include evolving app functionality, and extended scope in terms of supported device and deployment types. We furthermore argue for that the identified types of heterogeneities increase the complexity of the analysis and potentially reduce validity and overall quality of data and analysis results. Therefore, a general goal in collective sensing might be to take design and development decisions that consciously assess and effectively reduce data heterogeneity to acceptable levels.

The rising ubiquity of smartphones does not solve the mentioned problems or does it make it easier to make development decisions—more data may be collected, but the heterogeneity and quality of it will remain a concern. Furthermore, the rise and fall of manufactures and operating systems also adds to the problem. Thus, although we in this paper describe current issues in a contemporary context and draw in this paper on concrete experiences made within recent years, we believe that the identified underlying problems, as formulated herein, will persist and that they therefore have to be addressed—by and within the mobile sensing research community, since this community–other than the benefiting sciences and projects, utilizing collectively sensed data—is at the forefront of developing sensing platforms. If we, as a research community, can establish good practices and methodologies for mobile sensing, we can help to ensure that, e.g., scientific results and policy changes based on collective, mobile sensed data are valid. With this paper we would like to provide the basis for and promote discussions of heterogeneity challenges in the research community.

The structure of this paper is as follows: The considered case studies are presented in Section 2. Afterwards our categorisation of types and causes of heterogeneity are discussed in Section 3 with outset in examples from the cases. Finally, conclusions are given and future work is discussed in Section 5.

Case Studies

To gather knowledge about data heterogeneity, we chose to analyse a number of cases where the authors have been directly involved in the development and deployment.

We consider three different cases: *Socionical App Framework (SocioAF)*: an event guide with crowd awareness deployed at several major events (see Figure 1) [23]; *F.Oskar*: a project within citizen science (see, e.g. [6, 19]), utilizing an app deployed to capture data for identifying transportation patterns (see, e.g., [5, 11, 24]) within an expanding industrial neighborhood with the goal of quantifying users' environmental impact through models of transportation behavior (see Figure 2); and *EnTracked*: an energy efficient tracking app deployed to capture position and trajectory traces [15, 13]. To highlight differences among the cases Table 1 lists properties of the cases including application area, device platforms, development process and tools, and users and phone models in deployments.

Table 1: Details for cases

	Description	Platforms	Development Process	Development Tool	Users	Phone Models
SocioAF	Crowd Awareness	Android / iPhone	Sequential	Native	1000+	Many
F.Oskar	Citizen Science	Android	Individual	Native	50-100	Many
EnTracked	Position and Trajectory Traces	Symbian / Android	Sequential	Native	10	Few

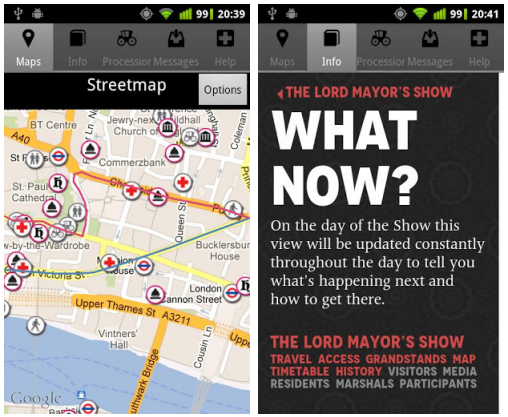

Figure 1: *SocioAF* app interface

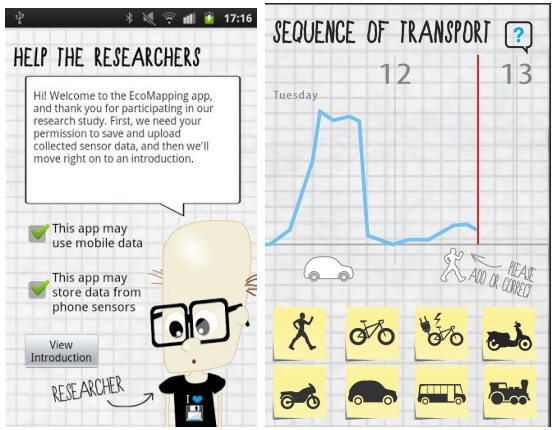

Figure 2: *F.Oskar* app interface

Heterogeneity in Mobile Sensing

Collecting data from several sources is fundamental in mobile sensing. Analysis and interpretation of collected data is challenged by the heterogeneity of the gathered data across sources and over time. For mobile sensing, such sources resemble usually individual users and their individual devices via which data is gathered. We argue for that the research community has to build up a more comprehensive vocabulary of types and causes of heterogeneity to be able to document and adjust for the resulting biases and heterogeneities. The process of being aware of, documenting and adjusting is worthwhile not only for the mobile sensing project at hand, but also the community at large to learn from experiences to improve practices for follow-up or unrelated future projects.

In the following, we will attempt to answer the question: *which kinds of heterogeneity impact the heterogeneity of data collected via mobile sensing apps?*. We will distinguish by types and causes of heterogeneity in mobile sensing drawing from the three cases introduced earlier to exemplify and discuss resulting effects. We will introduce three top-level heterogeneity types: i) *user heterogeneity* denoting the heterogeneities in demographics and practices of the users participating in the mobile data collection, ii) *device heterogeneity*, covering the heterogeneity in properties and abilities of the actual

devices used during data collection and iii) *project heterogeneities* covering the heterogenities originating over time from within mobile sensing projects themselves.

User Heterogenities

User heterogeneities refers to heterogeneity causes rooted in the demographics for different phone models and in concrete practices of the users participating in the mobile collection. We will discuss below the following heterogeneity causes individually: i) across user demographics ii) across app usage patterns and iii) across device usage patterns.

Device Type Demographics

Whenever data is collected with human participants in the loop, the identity of the respective users will have an influence on the produced data. This applies to mobile sensing, where data is gathered via the user's mobile device, but also when gathering data, e.g., via questionnaires: A correct and complete interpretation of the data gathered by a user would always have to be individualized, taken into account the individual users' characteristics, such as background, preferences, routines, attitudes, etc. While thus the ultimate goal of correct interpretation and complete understanding of such gathered data is not achievable, it can be approximated via reasoning about such characteristics, and account for resulting biases in the respective individual user's data.

Examples from the cases examined herein provide several angles on this issue. Within collective mobile sensing, users can be divided in groups according to the type of mobile device they use in their daily life. Survey data suggests, that user characteristics, such as technical affinity, age, preferences for leisure activities and places to visit differ significantly across, e.g. mobile phone platforms or models [1]. Thus, for collective mobile sensing projects, limiting the range of user devices implies a smaller and less representative user group—which then effects the collected data and may introduce user group-specific biases. Generally, it often applies that the range of supported devices should be as wide as possible, and at least reflect the targeted user groups in a representative manner: In projects, such as the *SocioAF* case, users of different demographics may be attracted to different exhibitions or events within, e.g., a fair or large festival. For the mentioned project, this applies to major user groups such as grandparents with their grandchildren versus users without children versus younger people without children; for a correct assessing of crowding hazards or bench-marking logistics it becomes crucial to support mobile devices as commonly used by all of targeted user groups.

App Usage

Getting an app onto a user's device does not guarantee that the user will use the app or that the user will use the app as intended by the developers. Especially participatory sensing scenarios are prone to this type of heterogeneity. For instance, when users provide observations, e.g. as advocated by the experience sampling method [7], some of them may report events just as intended, whereas others may have misunderstood what types of events should be reported, or have difficulties matching the perceived event into a categorization provided by the app interface. Furthermore, app usage, and frequency thereof, may depend significantly on the individual user's affinity and motivation towards using the app and fluctuate, e.g., with his level of attention. In *F.Oskar* we experienced, that a feature designed to enable people to pause sensing and intended to allow users to protect their privacy, was used instead to limit the sensing for the sake of saveing battery power.

Device Usage
There is a great variability in how devices are used by users; e.g do they carry their device with them all the time and when so in trouser pockets or in a bag [19]; do they recharge their phone frequently or do they let it run out of power from time to time [21, 10]. Users may also customize their devices in manners that impact sensing, this includes enabling and disabling location or communication features, such as, Bluetooth or WiFi, sleep settings, lock settings or access rights for apps. In *F.Oskar* we experienced users providing negative comments and uninstalling the app because the app as a side-effect impacted their Bluetooth settings and holes in data was also experienced due to phones running out of power. In *EnTracked* extra effort had to be made to enable robust motion recognition across different transportation mode options.

Device Heterogeneities
The device heterogeneity type covers heterogeneity of which the cause lie in the heterogeneous properties and abilities of the actual devices used during data collection. We will discuss below the following kinds of heterogeneity in turn: i) across software platforms, ii) across device hardware, and iii) across OS, firm- and hardware evolution.

Platform
Mobile user devices, such as smartphones, are nearly always tied to a specific platform and a respective operating system, such as iOS, Android or Windows Phone. The differences between these platforms can have significant effects on data collection and on the resulting data. Technical differences relevant in mobile sensing projects include sensor APIs, scheduling options, and data representation. Furthermore, platforms differ not only in mere technical aspects, but also on basis of fundamentally different long-term platform policies. These include: i) policies affecting app functionality and development (user-protective versus open-market policies) and ii) distribution channel policies.

Distribution policies: The difference and policies in i) and ii) can be observed, e.g. by comparing Apple's iOS, where the protection of the user (e.g. from battery draining or privacy-violating apps) is emphasized, with Google's Android, which instead emphasizes an open-market policy. Additionally, Apple's concept enforces lengthy review processes. While on stores such as Google Play submitted apps are available virtually instantly, on Apple's store it takes on average about a week, but longer review periods have to be considered, see [2] for related statistics.

Especially for mobile sensing projects these restrictions can be severe: Often, the apps to distribute are developed iteratively, or for separate data collection events, resulting in a large number of app and app update submissions; this was experienced e.g. in the case of *SocioAF*, where the concept was to provide app versions tailored to specific events. Furthermore, for participatory sensing projects where user actively provide feedback about the system, a highly iterative development process creates a need for several releases and updates. In all such cases, if several platforms are supported, the stall times for awaiting reviews are determined by the platform with the longest review process—the alternative being to accept version heterogeneity across platforms.

Policies Restricting Functionality: Commonplace among the restrictions inflicted by user-protecting app store policies such as Apple's are those concerning (potentially battery draining) background services for sensing and data communication—services thus, which from a mobile sensing perspective are crucial. For many mobile sensing

projects, including *SocioAF*, Apple reviewers could not be convinced to accept background GPS sensing and data uploading, respectively. Specifically, in the *SocioAF* case the reviewers persisted that no 'tangible benefits to the user' were provided by background uploads of user position data—despite the developers explanations that users benefits were achieved in the form of increased awareness and user safety in emergency situations during the targeted public events, and although this claim was backed up by providing proofs of official approval from the involved 'common good' stakeholders, i.e. charitable organizations or public authorities such as the City of London Police. To comply with the restrictive app store policy, the *SocioAF* app was altered to send position updates were sent from user devices, only if 'significant', i.e., in case of position changes of at least 100 meters. As a result for mobile sensing projects, homogeneity of collective sensing app functionality across platforms including iOS can often only be achieved by limiting functionality to what the most restrictive of the targeted platforms allows.

Sensing API: Further heterogeneities between platforms, which are handicapping mobile sensing, result from differences in APIs, foremost those for user interaction and for sensing. For sensing, differences across platforms, but also across OS versions, exist e.g. in regards to localization options: Most modern platforms have introduced by now API calls which allow to specify i) desired levels accuracy and frequency of localization and/or ii) desired trade-offs with power consumption. As there are different and differently effective means for achieving such levels and trade-offs, also the APIs and the underlying implementations differ [17].

The *EnTracked* system, specifically designed for this, initially for Symbian, uses adaptive duty cycling, i.e. the time of localization events is chosen adaptive to the specified levels or trade-offs. To port *EnTracked*'s behavior completely homogeneously to, e.g., Android or iOS proved not possible; Android offers a method for periodic duty cycling, and as alternative a method for requesting a one time GPS fix; the latter call, though, by default contacts an A-GPS server over the cellular network–thus causing additional energy costs for communication, which are usually unnecessary when running EnTracked, since the last one-time fix request, and thus the last update from the A-GPS server is usually only a few minutes, or even less, old. The API in current iOS (6.1.3. at time of writing), on the other hand, does not allow to specify a period for static duty cycling, but only offers to specify desired accuracy—and it is not revealed how, and how energy-efficiently, the accuracy is achieved. This limits the options for mobile sensing projects, and potentially the resulting data quality and amount, severely, as experienced also, e.g., in the *SocioAF* project. Similar issues exist across platforms with duty-cycling other sensors such as the accelerometer. Such duty-cycling, and using an accelerometer, e.g., as an energy-efficient wake-up sensor during periods of user inactivity, proved doable on Symbian within *EnTracked*. On Android, though, such accelerometer scheduling proved inefficient in practice or alternatively ineffective—both due to the sleep scheduling behavior inherent in all investigated Android versions, see also [20]. For enabling background accelerometer sensing on iOS, app store restrictions as detailed above apply. Furthermore, the iOS API only allows accelerometer sensing in the background only as a byproduct of requesting a localization—which in itself consumes

significantly more energy costs than the mere accelerometer sensing, see, e.g. [15].

Hardware
A user's mobile device and its capabilities and behavior in a mobile sensing context is not fully described by the software platform it is tied to; there are still crucial differences across *manufacturers*, *models*, and *hardware revisions*. From these, we can infer for mobile sensing relevant information such as hardware available for sensing and processing data, as well as screen sizes for adequate user interaction. While many chipsets are more or less standardized across different makers of mobile phone, CPUs, network chips, antennas, and sensors are continually being developed for better performance, improved features, and cheaper manufacture. This makes it difficult without prior rigorous calibration to directly compare sensing data from one phone to another, even if the phones are of the same model, as they may well be of different hardware or firmware revisions. This problem quickly becomes intractable when many manufacturers produce many different models within the same platform, as is the case with Android, Windows and the upcoming Tizen [3] phones; the problem is then further enhanced as some manufacturers provide with their phones own variants of a platform's OS and UI for the sake of product differentiation.

Performance: Mobile phones have in recent years seen an impressive rise in processing power, as dual or even quad core mobile phones are now commonplace. The ability to preprocess collected data on the mobile devices have thus similarly increased. However, this goes only for the latest generation of phones—older devices are still in use and should thus be included in the testing for a mobile sensing project, as they may not be able to sufficiently match the desired performance. In the two user-interactive cases covered *F.Oskar* and *SocioAF*,respective experiences of varying performance were made with a variety of UI elements, prominently with those involving map views.

Sensing capabilities: Also in regards to sensing capabilities, the evolution of user devices is rapid. More and more sensor types become common-place in, e.g., smartphones and thus offer new types of mobile sensing data. Secondly, advances in more energy-efficient sensing have been made—not only in hardware design, but also in utilization of the hardware within the devices' OS'es, see also the following subsection for more details. Thirdly, these new kinds of data change how higher-level context can be inferred; e.g. in cases such as the *F.Oskar* app, code for inferring transportation mode automatically and movement type for *EnTracked* had to be written individually and custom-tailored for phone types with different sensor type sets—or even for individual phone models, if measures such as inference accuracy and battery life are to be optimized. Fourthly, it is noteworthy, that due to the pressure to reduce cost, form factor and weight, newer devices may have actually worse sub-devices, e.g. GPS antennas, and in effect worse localization accuracy than their predecessor models which was experienced for *EnTracked*. Overall, to support hardware of differing sensing properties and capabilities results often in having to deal with heterogeneity in regards to accuracy, sensing data types, and in the algorithm design and implementation for inferring high-level context data types.

OS and Device Versioning
Within a specific platform, there is still heterogeneity, most pronounced on the Android platform. Some manufacturers and telcos support long term

upgrade-ability of their devices, whereas others do not (or only support it for a subset of their models). This means that a service may not be available on all phones, or that a service may behave differently across different operating system revisions. Even *if* both manufacturer and telco support upgrading the operating system on a particular phone, it is by no means given that the owner is inclined, aware, or capable of performing an upgrade. Updating of apps also differs between platforms. On e.g. iOS, users have to manually initiate an update, and likewise on older versions of Android (or rather, the Google Play app, which handles app updates through their app store). Newer versions of Android give the user an option to automatically keep apps updated, which relieves the user of the burden to do so. These update policies have to be kept in mind when deploying, as an arbitrary combination of app versions may be "in the wild" at any given time. Similar applies to OS version updates, which during data collection periods may result in data heterogeneity, prominently in the form of data outages, if the updates either contain bugs, or if the mobile sensing project's code, running on the updated OS, leads to unexpected behavior. This was experience for the *SocioAF* app where an iOS update changed the behavior of an iOS module making the app starting fail.

For mobile sensing projects, crucial differences across versions are, as for across platforms, within sensing and user interaction. For user interaction, metaphors such as drag'n'drop functionality, multi-touch gestures and respective functionality such as panning, zooming, etc. are only incrementally added to the OS versions of platforms. Specifically, for the case *F.Oskar* drag'n'drop was considered very helpful, but was not supported via API calls in Android versions earlier than 3.0—and thus not by a large share of the user devices in operation at time of the writing. For sensing APIs, and their actual implementation, the statements about heterogeneity across platforms apply also across versions: Significantly, the quest for longer battery life implied that incrementally—and additional to hardware improvements mentioned above—more API options and less power-consuming implementations of these are provided on most platforms—specifically in regards to sleep behavior, customized accuracy levels and trade-offs with power consumption, c.f., e.g. [13, 14]. Note also, that updates of a phone's firmware may change its behavior, especially in regards to sensor and communication scheduling, significantly, although this may not be revealed in any release notes [14].

Project-internal Heterogeneities

Under project heterogeneities we subsume causes for heterogeneity that are rooted in the mobile sensing project setup itself. We will discuss below the following heterogeneity causes individually: i) across the software life cycle and ii) within and across deployments.

Software Life Cycle

As generally within software projects, also mobile sensing software evolves over time—as developers learn, and new requirements are discovered or refined. It is overly optimistic to assume that one software version will suffice during even a single data collection campaign. Bugs and inadequacies will be discovered, and may be important to fix. Updates such as fixes lead to heterogeneity of the collected data over time. This heterogeneity is even higher, since a heterogeneous penetration of pushed upgrades and bug-fix releases during or between data collection phases is to be taken into consideration: Pushes may be preceded by app store review processes of a priori unknown, but significant (and across platforms varying)

length; furthermore, users may install software updates either not all, automatically, or manually at a user-chosen point in time. As data is collected over time, and since developers may refine collection methods, make use of hitherto unused sensors, user input elements, or change data representations to better suit the task, This requires attention, so that old data is not abandoned, but kept, possibly separately, along with later revisions, and that for merging data from different revisions adjustments may have to be undertaken. For illustration the range of frequency of changes to and updates of mobile sensing software updates, we provide records from the three investigated cases. For *SocioAF*, flawless operation was only achieved after two problematic public events and after significant bug fixes; these fix issues prominently with data communication and with slow and ill-displayed UI elements on a variety of devices from the diverse and highly fragmented Android platform. Both similar UI issues on specific phones, and data communication issues led to updates in the *F.Oskar* case, fixing battery draining issues: one fixing cases of unintended Bluetooth use, and one fixing cases of unintended repetitive data uploads that also may lead to high data plan costs for the user; the latter fix also introduces an efficient compression of data on the phone before uploading it.

Updating an app during its life cycle may also become advisable, when developers e.g. refine collection methods, make use of hitherto unused sensors, user input elements, or change data representations to better suit the task. For instance, in the *EnTracked* case, a number of updates were issued, not to fix bugs, but to extend and improve functionality, specifically to reflect and make use of novel properties of each next smartphone generation, such as new types of sensors, additional sensing API elements or smart battery interfaces [13, 14].

Additionally, in the *SocioAF* case, the range of supported platforms and devices was increased after experiences with the app on just one platform in initial major sensing events had been gathered. As has been illustrated here, the support of additional platforms may suggest or even imply updates of the software on the already supported platforms, e.g. in order to mitigate device heterogeneity.

Development Practices
Development for different platforms brings with it the risk of heterogeneity of software code, as well as of the developers' testing and debugging practices. Such hetereogeneity is to a large extent rooted in the differences of platform versions in regards to programming language, APIs used and potentially also in platform-specific implemented functionality. Nonetheless, also the evolution of the mobile sensing project will contribute to heterogeneity, since milestones, feature additions and especially testing campaigns may diverge for the platform-specific software variants—and will thus contribute to the challenge of fighting the resulting data heterogeneity across platform variants. The issues has been observed and hindered project development and sensing campaigning in the *SocioAF* case.

Conclusions

In this paper, we discussed data heterogeneity within mobile sensing, and introduced a taxonomy of heterogeneity types grouped into the main categories of device, user and project heterogeneities as listed in Table 2 identified throughout three case studies of community-based mobile sensing apps. If we, as a research community, can establish good practices and methodologies for handling heterogeneities within mobile sensing, we can help to ensure that, e.g., scientific results and policy changes based on collective, mobile sensed

data are valid. With this paper we aimed at providing a basis for and promote discussions regarding heterogeneity challenges for mobile sensing in the research community.

As a natural next step for future work in this direction we envison a discussion and accessment of the choices and options for developers within mobile sensing, and how such choices may impact and mitigate different types of data heterogeneity. Prominent among these choices are which platforms to support, and whether to as far as possible adhere and exploit what is provided on individual platforms, i.e. UI design, sensing APIs, and app store distribution channels, or whether to instead aim for a maximum of app homogeneity across platforms, in terms of, e.g., appearance, user interaction, and distribution and sensing procedures. To this choices are related also decisions about whether to aim for a heavy or instead thin client in terms of data processing; whether to develop as mobile websites instead of native apps; or whether to develop app solutions sequentially for individual platform to be supported, or in parallel, either natively or using cross-platform development tools.

Table 2: Identified categorization of heterogeneity types

User Heterogeneities
- Device Type Demographics
- App Usage
- Device Usage
Device Heterogeneities
- Platform (Incl. Distribution policies, Policies Restricting Functionality, Sensing API)
- Hardware (Incl. Performance and Sensing Capabilities)
- OS and Device Versioning
Project-internal Heterogeneities
- Software Life Cycle
- Development Processes

References

[1] http://blog.hunch.com/?p=51781, Mar. 2013.

[2] http://reviewtimes.shinydevelopment.com/, Mar. 2013.

[3] http://www.tizen.org/, Mar. 2013.

[4] T. F. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. J. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *IEEE Pervasive Computing*, 6(2):20–29, 2007.

[5] H. Blunck, N. O. Bouvin, J. Mose Entwistle, K. Grønbæk, M. B. Kjærgaard, M. Nielsen, M. Graves Petersen, M. K. Rasmussen, and M. Wüstenberg. Computational environmental ethnography: combining collective sensing and ethnographic inquiries to advance means for reducing environmental footprints. In *Proc. fourth international conference on Future energy systems (e-Energy 2013)*, pages 87–98. ACM, 2013.

[6] J. P. Cohn. Citizen science: Can volunteers do real research? *BioScience*, 58(3):192–197, Mar. 2008.

[7] S. Consolvo and M. Walker. Using the experience sampling method to evaluate ubicomp applications. *Pervasive Computing, IEEE*, 2(2):24–31, 2003.

[8] H. S. M. Cramer, M. Rost, N. Belloni, F. Bentley, and D. Chincholle. Research in the large. using app stores, markets, and other wide distribution channels in ubicomp research. In *UbiComp (Adjunct Papers)*, pages 511–514, 2010.

[9] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. The bikenet mobile sensing system for cyclist experience mapping. In *In SenSys*, pages 87–101. ACM, 2007.

[10] D. Ferreira, A. Dey, and V. Kostakos. Understanding human-smartphone concerns: a study of battery life. *Pervasive Computing*, pages 19–33, 2011.

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

[12] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher. Greengps: a participatory sensing fuel-efficient maps application. In *In MobiSys*, pages 151–164, 2010.

[13] M. B. Kjærgaard, S. B. Bhattacharya, H. Blunck, and P. Nurmi. Energy-efficient trajectory tracking for mobile devices. In *Proc. 9th International Conference on Mobile Systems, Applications, and Services (MobiSys2011)*, pages 307–320. ACM Press, 2011.

[14] M. B. Kjærgaard and H. Blunck. Unsupervised power profiling for mobile devices. In *Proc. 8th International ICST Conferene on Mobile and Ubiquitous Systems (Mobiquitous '11)*, 2011.

[15] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær. Entracked: energy-efficient robust position tracking for mobile devices. In *Proc. 7th international conference on Mobile systems, applications, and services*, page 221–234, 2009.

[16] N. D. Lane, H. Lu, S. B. Eisenman, and A. T. Campbell. Cooperative techniques supporting sensor-based people-centric inferencing. In *In Pervasive*, pages 75–92, 2008.

[17] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell. A survey of mobile phone sensing. *IEEE Communications Magazine*, 48(9):140–150, 2010.

[18] E. Miluzzo, N. D. Lane, K. Fodor, R. A. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *The ACM Conference on Embedded Networked Sensor Systems*, pages 337–350. ACM, 2008.

[19] S. Patel, J. Kientz, G. Hayes, S. Bhat, and G. Abowd. Farther than you may think: An empirical investigation of the proximity of users to their mobile phones. *UbiComp 2006: Ubiquitous Computing*, pages 123–140, 2006.

[20] A. Pathak, A. Jindal, Y. C. Hu, and S. P. Midkiff. What is keeping my phone awake?: characterizing and detecting no-sleep energy bugs in smartphone apps. In *Proc. 10th international conference on Mobile systems, applications, and services*, page 267–280, 2012.

[21] A. Rahmati, A. Qian, and L. Zhong. Understanding human-battery interaction on mobile phones. In *Proc. 9th international conference on Human computer interaction with mobile devices and services*, pages 265–272. ACM, 2007.

[22] W. Willett, P. Aoki, N. Kumar, S. Subramanian, and A. Woodruff. Common sense community: scaffolding mobile sensing and analysis for novice users. In *In Pervasive*, pages 301–318. Springer-Verlag, 2010.

[23] M. Wirz, T. Franke, D. Roggen, E. Mitleton-Kelly, P. Lukowicz, and G. Troster. Inferring crowd conditions from pedestrians' location traces for real-time crowd monitoring during city-scale mass gatherings. In *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*, pages 367–372. IEEE, 2012.

[24] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. Understanding transportation modes based on gps data for web applications. *ACM Transactions on the Web (TWEB)*, 4(1):1, 2010.