SENSE-SATION: An Extensible Platform for Integration of Phones into the Web

Alireza Sahami Shirazi, Christian Winkler, Albrecht Schmidt
Pervasive Computing and User Interface Engineering, University of Duisburg-Essen, Germany
{alireza.sahami, christian.winkler.pc, albrecht.schmidt}@uni-due.de

Abstract—The “Web of Things” vision promotes the integration of smart devices into web, using web technologies and protocols as underlying interaction mechanisms with smart devices. Technically many phones include various sensors and provide a large-scale sensing platform. To make this functionality available to a large number of (web) developers our research aims at easing the creation of applications that use mobile context data (e.g., movement data). We present a sensing platform for mobile devices, called SENSE-SATION, which gathers and stores available information on a mobile phone and makes this information directly accessible via RESTful web services – locally and remotely. By using a concept of virtual sensors, the platform supports the full development process, from the initial idea via adapting/extending the platform to incremental development of virtual sensors and applications via Web 2.0 controls. Hence, (web) developers can create arbitrary context-aware applications that make use of mobile hardware capabilities without touching mobile phone programming at all. In this paper we discuss the design issues and provide case studies to describe how this platform can empower the development of social, context-aware applications.

Keywords: mobile; web; context; web 2.0

I. INTRODUCTION

The World Wide Web has evolved into a ubiquitous information and communication medium. Over the last years a rapid increase in user generated content, online collaboration, peer-to-peer interaction, and social activities could be observed, often summed up as Web 2.0 phenomena. Social computing and social network systems (SNS) have become popular and an important part of everyday life – for interaction and communication. SNS such as Facebook, MySpace, and Twitter connect users to each other and provide interaction and communication channels for social activities (e.g., social applications/games) and information sharing. Up to now the majority of users provide information, such as status updates and profile information, on SNS manually using PCs or mobile phones. With the increasing pervasion of smart phones there is a clear trend towards continual mobile usage. Users set and share status updates explicitly and manually on their phones with others. However, sensors on these mobile devices allow implicit and automated communications, e.g., location and activity can be determined from the phone’s sensors and be provided to friends via SNS.

In our research we investigate how to ease the integration of mobile phones, beyond explicit and manual uses, into context-aware and social applications and networks. Currently mobile phones are one of the most ubiquitous personal computing technologies. They are enhanced with various sensors, such as microphone, light sensor, GPS, accelerometer, compass, and camera. These sensors can provide the basis for the phone to acquire different types of context information and activity recognition. Hence, phones essentially are a very large-scale sensing platform, e.g., presented as Sensonomy [26]. With the ubiquity of the mobile phone and its input, sensing, and output capabilities users can implicitly and explicitly contribute at any moment. However, currently most applications that use context are very basic (e.g., sharing location) and we believe one reason is that including sensing, and in particular distributed context reasoning, is still difficult. Other platforms, especially the WWW, have shown that providing a simple means for creating and developing systems leads to a large and diverse set of applications and services. Though mobile phones have become powerful enough to run and handle very complex applications, developing applications for mobile phones is still not as easy as the web programming. Current mobile platforms are often difficult to customize – if at all – and lack means for reusing sensing and context reasoning across different applications (and mobile OS). Thus, typically these parts are reinvented and reimplemented again and again for new applications.

We envision a middleware and online platform that provides reusable sensors, context, and activity information from a single phone as well as from a set of phones. Based on this platform developers should able to incrementally create (web-based) applications, for mobile and desktop systems alike, which utilize contextual information gathered in real-time from users’ mobile phones. Our goal is to create a platform based on open, standardized, and secure web protocols, which are familiar to a large number of web developers who have not developed for mobile devices.

In this paper we introduce SENSE-SATION, a platform we have implemented for mobile phones to expose the phone’s functionality locally or remotely over web protocols. The platform provides means to collect and manage information available on phones and makes it accessible through RESTful APIs locally as well as remotely over the Internet. Local applications on mobile phones and web applications can take the advantages of information collected from sensors, the input, and output capabilities of the phone. Using web technologies, and without any programming on the phone, sensor-based and context-aware applications can be created. For example, to create an application that makes use of a phone’s accelerometer or GPS sensor web developers can retrieve the information through web protocols. Equally they can create more complex sensors by combining existing sensors. Interaction with all resources on the phone can be customized and extended directly through a web interface, and therefore the platform facilitates a seamless integration of mobile phones and their capabilities into Web 2.0.

The main contribution of this paper is an assessment of requirements and the description of the architecture and imple-
This paper is structured as follows: after providing an overview of the related works, we describe the core design concepts behind the platform, especially the concept of Virtual Sensors. Next the system architecture and implementation are explained. Later, we provide two case studies to describe how this platform facilitates the integration of mobile phones into Web 2.0. After discussing privacy and security issues, we finish with a conclusion and prospect on future work.

II. RELATED WORK

Ubiquitous computing has been dealing to a great extent with integration of digital artifacts with the physical world. The vision of “Internet of Things” and “Web of Things” is that the physical world can be combined with computer networks and it promotes the integration of smart devices not just to the Internet – at network – but also to the Web – at application level, e.g., [28] and [29]. This enables web technologies to interact with the devices. Various platforms have been developed which mainly act as data storage between the clients and sensors. Pachube is a web service that enables storing, sharing, and discovering sensors data real-time over the Internet. SenseWeb is a peer produce sensor network that eases developing sensing applications [16]. In the same line is the SenseShare [27] project, which lets users share sensor data with others. Additionally it allows users to apply different filters to the data before sharing. Furthermore, Liu et al. [18] introduce virtual sensors for a Web 2.0 virtual watershed that provide information through Internet protocols. In [13] a platform is proposed which enables users to share their Web-enabled devices so that others can use them.

On the other hand, using mobile devices carried by users as sensors creates interesting opportunities. This approach is often referred to as people-centric sensing [5][6]. Examples of early works on people-centric-sensing with non-standard devices are pollution measurement [25] and charting of emotions on a map [22]. In both cases sensing on people is linked to GPS positions. With more and more sensors on the phone researchers explore various applications for people-centric-sensing with standard devices but also based on heterogeneous systems [1]. Gaonkar et al. [11] describe a micro-blogging service, which uses phones to record and share multimedia contents in real time. In [2] an application is explained which uses location information on phones to infer personal environmental impact. Mobile phones have been used to extract and learn about the daily patterns of human behavior and complex social systems [8],[9].

Also, acquiring information on mobile phones has been explored for providing context-aware applications, e.g., in [19]. The iCAMS [21] system estimates the situation of a person by using schedule and location information and allows users to choose the preferred method of communication (e.g., in person, home/work phone, email). Context-Phonebook is an application showing callee’s context information such as details on the user’s connection status in the potential caller’s phonebook [24]. IYOUIT, described in [4], is a mobile application that eases recording, storing, and using context information for end-users by applying their automatically inferred mobile context data. Miluzzo et al. [20] present CenceMe, a mobile application, which deduces the presence of individuals by using sensor-enabled mobile phones and shares information through social networks. We refer to [3] and [7] for surveys about context-aware systems.

Besides, researchers have proposed various platforms for mobile phones to facilitate creation of con-text-aware applications [10]. Betelgeuse is a data collection platform for mobile devices that uses a microkernel Java ME architecture with plug-ins in more native programming languages to collect sensor data [17]. The data can be used by local mobile (web) applications. The ContextPhone platform follows a similar idea, yet it also supports web protocols for publishing information to SNS [23]. Apart from mobile phones, Hong et al. in [15] report an infrastructure to context-aware computing.

All the mentioned platforms and frameworks use data from mobile phones and/or sensor networks. Some infer higher-level context data and/or share it with local or remote applications. But as no platform can be pre-tailored to the need of applications, (web) developers require mobile programming knowledge to extend these platforms. Besides, the platforms are not customizable while being deployed and lack privacy mechanisms, since most of them do not deal with personal mobile data. Based on the authors’ knowledge, no platform can be found which targets mobile phones as “Web of Things” and provides a simple way to integrate phone resources into Web 2.0. In contrast to mentioned projects, in our project we aim at integrating mobile phones into Web 2.0 by providing a platform and community for mobile phones and enabling the Web of Phones in a customizable and typical Web 2.0 fashion. The platform is responsible to gather and store data and provide a suitable way to access the data through the community. Thereby we bring mobile phones not only to the cloud, but also to the crowd.

III. DESIGN CONSIDERATIONS AND DECISIONS FOR A MOBILE CONTEXT-BROKERING PLATFORM

During envisioning and designing the SENSE-SATION platform, three core challenges have been identified:

1. How to support data acquisition from mobile phones in an easily extensible, hardware and mobile OS independent fashion? And how to provide context reasoning at the platform level?

2. How to make the data 24/7 available to applications in a user privacy respecting manner?

3. How to support a full development workflow, comprising mobile phones, solely from the Web?

The aforementioned challenges are addressed by three key concepts of SENSE-SATION: (1) “Virtual Sensors” (VS) providing data acquisition and context reasoning, (2) RESTful APIs enabling easy data sharing, (3) and an SNS that supports developers with a Web development workflow for incremental ap-
application building. In the following sections we present the details of how the key concepts facilitate the full integration of mobile phones into the Web and the development of context-aware applications. The sections will also introduce some components of the SENSE-SATION platform, which are summarized at the end of this part (see Figure 2).

A. Data acquisition: Virtual Sensors

1) How VS provide access to sensors and other hardware

Phones include different types of sensors, ranging from physical to informational sensors. The SENSE-SATION platform provides access to all information available on mobile phones and achieves this through the concept of virtual sensors. Virtual sensors are small software units within the system that provide information supplied from an available physical sensor on the phone (e.g., accelerometer), from the phone’s operating system (e.g., a sensor that provides the current profile setting), or from the user through GUI interaction. Additionally a virtual sensor may provide a way to use the phone’s output capabilities (e.g., using the vibration actuator, or sending a message). The novel aspect we bring to the concept of Virtual Sensors is the idea of remote programming of the software units from the Web.

Furthermore, Virtual Sensors consist of a set of data output fields (e.g., latitude, longitude, accuracy, etc.), an optional set of input parameters (e.g., minLatitude, maxLatitude) and a set of program codes for each targeted mobile OS. The program codes are in a respective well-supported programming language of the targeted mobile OS (e.g., Java for Android, Python for Symbian), thereby allowing full access to most phone features. But at the same time, the concept of Virtual Sensors provides hardware abstraction across heterogeneous devices and mobile OS. Similarly, maintainability is supported since changes in hardware or software components can easily be adjusted in the respective virtual sensor without touching other components being required.

2) How VS provide context reasoning and software reuse

In many cases, the raw data provided by sensors does not supply meaningful information to applications (e.g., a GSM cell ID is not helpful without comparing it to a database of cell ID geolocations). To free context-aware applications from processing raw sensor data and inferring context information on their own, the platform offers means to encapsulate this functionality. In contrast to mentioned platforms in related works the concept of Virtual Sensors does not dictate a fixed strategy for context reasoning like ontologies or predicate logic. Instead, it leverages the power of the SENSE-SATION developer community. Through the definition of new virtual sensors, which are the combination of already existing and defined virtual sensors to an unlimited hierarchy, SENSE-SATION allows application developers to move the required context reasoning logic to the platform level. For example, a “sleep” virtual sensor could take the input of a virtual accelerometer sensor, the light level of a virtual light sensor, and the current system time to estimate if the mobile user is currently sleeping. Similarly, a virtual “traffic jam” sensor can take the output of virtual location sensors from the SENSE-SATION users to estimate possible traffic jams. However, sensors could as well use sophisticated reasoning techniques such as machine learning, ontologies, etc.

By providing these hierarchically structured virtual sensors on platform level, developers can create and share virtual sensors across different applications. Hence the implementation of raw data processing and context reasoning is only required once across several applications, thereby providing a high opportunity for software reuse.

3) How VS facilitate data synchronisation

The Virtual Sensor Framework (Figure 1) runs on mobile devices as well as online back-end servers, which accumulate synced data from mobile phones. Since the similar data structures and software components exist on both sides, data from single user can be easily collected on a mobile phone and bidirectional synced with back-end servers store data from several phones.

B. Data sharing: RESTful APIs

1) How the RESTful API provides central data access

To simplify accessing virtual sensors’ information, a standardized web protocol is required. The RESTful architecture and its principles (statelessness, idempotence, etc.) play very well with distributed mobile data access (which is characterized by regular changes in connection quality) since they are designed for stateless connections. The resource-oriented design fits also very well the concept of Virtual Sensors, which serve as individual resources. Examples of the API structure are provided later in the case studies. Every mobile phone as well as the back-end server provides an API to access information. Through this APIs, the context-information can be retrieved by any HTTP-enabled applications locally or remotely.

2) Sensed data collection and offline access

There might be applications that use the history of sensed data to provide a specific service to the user or the phone may run out of battery. Therefore, the SENSE-SATION platform supports synchronization and access to sensed data – even if the device is offline. All virtual sensors collect their data in a database table that fits their field definitions (e.g., latitude, longitude, etc.) locally and on the back-end server (on the Internet). On the phone a robust database system secures sensed data against loss. The online database and API provide access to all synced mobile phone data, even when devices are offline. Still mobile phones also store the history of sensed data to supply it to local applications (which may benefit from higher level sensors’ context reasoning). This avoids the need for a persistent Internet connection that still may not be affordable, desired, or available.

C. Integration into Web 2.0

Besides collecting and sharing context information, SENSE-SATION addresses two additional issues to enable the Web of Phones: (1) making mobile phones and their resources as addressable as just another web resource and (2) providing a typical Web 2.0 development workflow that integrates mobile phone resources.

1) Make phones and their data directly available

The most important step towards integrating mobile phones into Web 2.0 and making the mobile phones’ peculiarities transparent to web developers is to equip them with their own Internet address. This way, the mobile phone client is address-
able and reachable from the Internet just like any other web resource and the data can be retrieved in real-time by any (eligible) application. This is achieved by a permanent connection between mobile phones and a gateway server, which forwards requests from the Internet to corresponding mobile phones.

2) Extending the platform from the Web

Traditional platforms supporting context-aware computing usually follow traditional development workflows. A team of software engineers develops the platform and releases it. From time to time, they adjust it to changed requirements and deploy updates to their software. For application – and especially web – developers, this is very limiting as customizing the platform is not easily achievable. Furthermore, own context reasoning logic that might be useful for other developers cannot be easily merged back to the platform.

To address this big shortcoming of present platforms, SENSE-SATION moves developing and extending the platform to the Web. The key concepts of Virtual Sensors and the corresponding RESTful API facilitate the web-based approach. Virtual Sensors are not built into the platform, but are created and managed on SENSE-SATION’s website. A web GUI allows developers to define a virtual sensor by name, description, the set of data fields (output), and the set of input parameters, among others. Further, the web GUI provides code editors for each available target platform (Android, Symbian, etc.), which developers use to either write or at least input their sensor codes. As mentioned before, these codes collect data from available phone resources or other virtual sensors and store it in the database. Based on requests through the RESTful API, they are also responsible for outputting their data depending on the input parameters. The required operations are supported through the “Virtual Sensors Framework” which provides default implementations (store, publish, etc.) and helper methods (schedule, buildSQL, etc.) for sensor processing.

3) Supporting incremental development

Moving the customization of the platform to the Web raises the question, how a typical iterative development (program > compile > test > deploy) of virtual sensors that run on mobile phones can be supported from the Web. The SENSE-SATION platform supports an agile development process in several ways. During implementation of a virtual sensor on the website, developers can use “Push to Phone” and “Push to Server” buttons to push the sensor code for testing purposes directly to the phone associated to the currently logged-in account and to a sandboxed test version of the back-end server. The code is directly executed on the target and debug output is logged. The logs can be retrieved via the RESTful API and update automatically. When creation of a virtual sensor is finished, the developer can submit its for publication (see Privacy & Security). Once the sensor is accepted, it is published and automatically integrated into SENSE-SATION’s mobile client and the back-end server. This way, except for on-device-debugging, a full development workflow is provided and mobile phones can be programmed by mobile and web developers alike (Figure 1).

For building applications that leverage SENSE-SATION, the RESTful API not only delivers XML and JSON, but also an XHTML representation that lets developers explore and test all virtual sensors and the API. This is complemented by an AJAX-Application, which automatically calls the RESTful API with reasonable input values. Also a dummy provider generates dummy input for sensors that might not yet be used by other SENSE-SATION users. Together, these components help getting started with the API easily. When the appropriate function is explored it can be included into a mobile application or AJAX web page and further developed incrementally.

IV. System Architecture

The SENSE-SATION platform consists of the following components (see Figure 2): (1) a mobile client running on each phone that handles virtual sensors for gathering, storing, and sharing data; (2) an online backend server, which provides cached data synced from mobile phones and also runs virtual sensors; (3) a gateway server that provides access to mobile phones from the Internet; (4) a website, used to manage the SENSE-SATION community and extend/customize the platform.

A. The mobile phone client

The mobile client runs primarily in the background on SENSE-SATION users’ mobile phones. It gathers, stores, and shares data by means of virtual sensors mentioned before. For providing the RESTful API, the phone client also runs a local web server. Further it provides a minimal graphical user interface (GUI), which is used to supply account information and gives a complete control over sensing and sharing data. Through this GUI users can manage each sensor individually. Besides, they can enable or disable all data collecting and sharing with local or remote applications. The mobile client and gateway connection can be launched automatically when the phone is started and run unobtrusively in the background.

B. Backend server

The backend server runs almost the same software as the mobile client. However, the back-end server is responsible for securing access to context data (to free the mobile client of this responsibility). To secure access to data the OAuth² authentica-

2 http://www.oauth.org (accessed September’10)
other platforms such as Facebook, 3rd party applications can be full control over their privacy settings. Additionally, similar to browse the data created by their phones. The users have also a the mobile phone client, have access to information, and can ties of a typical SNS. Users register on the site, can download the SENSE-SATION website provides all functionalities (clustering), and support for multi-threading.

V. SYSTEM IMPLEMENTATION
In this section we describe how the functional prototype and its components have been implemented.

A. Phone side
We implemented mobile clients for Android and Symbian OS, because they are the most widespread platforms that provide a mandatory requirement for SENSE-SATION’s concepts, namely background tasks/processes. For the Symbian version Python for S60 was used because in contrast to Symbian C++, Python allows dynamic code execution, a prerequisite for pushing code from the Web while the client is running. In addition, Python is easier and more expressive to program than Symbian C++. For Android standard Java is used to provide Android developers with their familiar programming language. Java’s class loading allows for dynamic code execution as well. The Java code is compiled and transformed to a DEX file on the server, then pushed and dynamically loaded on the phone client. As a database is essential to store the sensed data on the phone and Android has built-in SQLite support, for Symbian SQLite (SQLite for S60 with pysqlite2 python module) is used as well.

B. Backend server
For the web server running on mobile clients to provide the RESTful API, on Symbian a Python web server called Cherry-Py was used. This web server runs on the S60 platform with minimal modifications because it does not rely so much on a popular OS environment. For Android we ourselves built the web server, which was not difficult with the availability of good Java HTTP libraries. All client implementations use custom RESTful dispatchers and templates to deliver XML, JSON representations. The Python version leverages the Routes and the Genshi templating module for this task. Communication between virtual sensors is achieved through a publish/subscribe model that is built of the information on the website. Additionally, an AJAX based user interface powered by prototype.js and script.aculo.us JavaScript libraries is implemented. The GUI is mostly implemented with web technologies to make it platform independent so that it can be used across different mobile OS and also remotely from a desktop PC browser. This possibility is enabled by the local web server and the addressability from the Internet. The GUI uses the RESTful APIs to communicate with the main client.

Because of the similar tasks to perform, the backend server runs almost the same software as the mobile Python client. However, it is responsible for securing the data access. Thus, it uses the Python OpenSocial library to validate OAuth requests. Wrong OAuth requests as well as requests by applications, which do not have permission, are denied. Furthermore, if data from multiple users is requested, the data from users who have not authorized the application is silently left out. Hence, application developers do not need to know beforehand which users authorized their application. Additionally, each developer is assigned a dynamic debugkey, which is valid during the logged in session on the website. Developers can use this key with the RESTful API instead of OAuth authentication to retrieve only their own sensors data, bypassing authentication. This allows for example exploring the data with the RESTful API through the address bar of the web browser. Despite the authentication mechanisms, another difference lies in the back-end using MySQL instead of SQLite due to the better performance, scalability (clustering), and support for multi-threading.

C. Gateway server
To manage connections between the mobile clients and the back-end server, we use a customized version of Nokia’s Tomcat implementation from the Mobile Web Server project. Customizations include the validation of OAuth or debugkey requests before proxying requests to the mobile phone. However, with the Android version we are currently evaluating using XMPP-connections with HTTP-tunneling to leverage the features of XMPP, among those automatic presence and roster management and particularly decentralized connections, which can solve the bottleneck of the central Gateway. Finally, a DNS

3 http://www.cherrypy.org (accessed September’10)
4 http://genshi.edgewall.org (accessed September’10)
5 http://www.prototypejs.org, http://script.aculo.us (accessed September’10) – these were the only JavaScript libraries found working on Nokia S60 3rd. and 5th. Edition
server maps wildcard subdomains to the gateway server and an Apache server proxies requests on Port 80 to the web, backend, or gateway server. Together they unite the whole system under a single Internet domain name.

D. SENSE-SATION website

As previously described the SENSE-SATION website undertakes critical tasks within the overall system that go beyond presenting information, i.e., user management, 3rd. party application development, and OAuth management. Various open source Content Management Systems (CMS) already fulfill the requirements for some of the services. Drupal5 (a PHP-based CMS with a large community providing over 1000 extension modules) Drupal includes different modules for OAuth and Apache Shindig6 integration with user accounts. Apache Shindig is the reference implementation of OpenSocial framework, a framework specification that unifies common functionality of social network systems especially regarding 3rd party applications. Together with Drupal’s good support for virtual sensors and applications, Drupal’s content types also allow for being natively extended with virtual sensors and applications.

VI. INTEGRATION WITH THE WEB 2.0

To present the usefulness of the SENSE-SATION community and describe how it can be used in developing various applications, we provide two case studies that we have implemented using the SENSE-SATION platform.

Figure 3: A picture taken by a mobile phone is shown in the map

A. Case Study “Take a picture”

Scenario: Imagine an online news website that wants to retrieve almost live pictures from an event, but only by the help of the crowd attending the event. On their website, they ask users of the platform to allow their “Take a picture” application access to some sensors of the SENSE-SATION platform. Now during an event, an employee of the news site uses the web-application to define the area of the venue on a Google Map and specifies instructions how and why users should take a photo (e.g., 50€ for the best photo of the artist). Through the SENSE-SATION platform, the users who are currently at the venue get a popup on their mobile phones. Then the users can take a photo of the artist, which is then directly returned to the news-site’s web application together with its geo-location and the name of the user who took it (see Figure 3).

Used Sensors
(1) Location Sensor: every mobile phone client runs the Location Sensor which combines information from several location-based sensors (such as GPS, GSM, Geo GSM, Dead Reckoning) and stores this in the local database as well as shares it with the back-end server. Through its input parameters (latitude, longitude, radius) the output can be restricted to a specific area. (2) mobile PictureRequest Sensor: this sensor’s arguments include an instruction (text field) as PUT request and opens a camera viewfinder on the phone. Once the user takes the picture he liked, the picture is saved to the phone and a reference is stored in the database. The request returns a job resource (URL) that can be queried (GET) the result.

Implementation: Apart from the implementation needed for defining the area on Google Maps, the application logic only involves following actions:

1. Make a request to the platform’s backend database: http://api.sense-sation.de/all1/Location2/data/username3/10min4/index.json?center_latitude=x&center_longitude=y&radius=1(km) with (1) specifying a request for all users, (2) the name of the requested Sensor, (3) only returning the username field and (4) only returning results from the last 10 minutes. The returned JSON response contains a list of all users (user-names) that have been found within the given area within the last 10 minutes and by means of any available localization technique.

2. Loop through the returned list of usernames and for each make a PUT request to: http://username.sense-sation.de/sensors/PictureRequest1/data/all2/latest3/index.json?instructions=Win…, with (1) being the Sensor requested, (2) specifying to return all available fields of the sensor and (3) to only return the latest set of data.

If the user denies making the picture, 403 Request rejected is returned. Instead if the user is going to take the picture, a 202 Accepted response is returned with the URL to a created job resource. Now by using the GET request the picture can be retrieved. If the picture has been taken, the request returns username, timestamp and the path to the picture, which might be http://username.sense-sation.de/sensors/images/028734012938.jpg. Finally, standard HTTP requests can be used to download the image directly from the mobile phone.

B. Case Study “Automatic Profile Change”

Though mobile phones become more powerful and include more functionality, on their own they are still not intelligent enough to know when to ring, vibrate, or do nothing at all to

5 http://www.drupal.org (accessed September’10)
6 http://shindig.apache.org (accessed September’10)
indicate incoming phone calls or SMS. The “Automatic Profile Change” application is an example for a SENSE-SATION powered mobile application that solves this shortcoming by automatically adapting the current profile, which is the common concept of unioning notification preferences.

**Scenario:** Imagine at least 80% of the people use a silent profile on their mobile phone during watching a movie in the cinema or in a meeting. This profile information is shared through the SENSE-SATION mobile client. Then, when a user who runs the “Automatic Profile Change” application enters the cinema his mobile phone’s profile can be automatically switched to a silent profile. This makes the mobile phone semi intelligent by just applying common crowd knowledge. By letting the user define a threshold for **loud > silent and silent > loud** transitions, he can adjust how robust his phone will be against automatic changes in each direction (the latter probably being much higher than the former). In the aforementioned example, only a threshold of 80% or lower for **loud > silent** will trigger a profile change. The change is done according to a user-definable mapping of profiles (meeting, outdoor, etc.) to the global silent or loud states.

**Used sensors:** (1) online **LocationProfile** sensor: The **LocationProfile** sensor combines data from the online Location sensor and the online **dynPhoneinfo** sensor (which includes a history of active profile information of all users) to dynamically compute a percentage of how many people use a **silent or loud** profile within a given region. (2) **mobile Location sensor** (optionally): if the phone runs a mobile SENSE-SATION client, more precise geo location (compared to standard GPS) can be received from SENSE-SATION’s mobile Location sensor through the local API.

**Implementation:** The application can be implemented on any phone that has Internet connectivity and is aware of its location by any means. Our envisioned implementation uses the S60 Web Run Time (WRT) to create a widget that runs on, for example, the home screen of the Nokia N97. At user specified intervals, the application does the following:

1. The Location sensor is used to receive the mobile’s current geo location (http://localhost/sensors/Location/data/all/latest).
2. The application queries the online **LocationProfile** sensor for the **silent** and **loud** fields, providing its current geo location, by making a request to: http://api.sense-sation/sensors/all/LocationProfile?/data/silent, loud/latest/index.xml?latitude=x°&longitude=y°&radius=0.3", with (1) being the name of the sensor, (2) the data fields to return, (3) returning only the latest value, (4,5) the current geo location of the phone and (6) the search region specified to 300 meters around the current location.
3. The returned percentages for **silent** and **loud** are compared to the user specified thresholds and the profile is changed accordingly if it is necessary.

Naturally, the usefulness of the application highly depends on the accuracy and robustness of the location technique. Since SENSE-SATION’s **Location sensor(s)** can include any techniques (or web services) for outdoor and indoor positioning (e.g. WiFi fingerprinting or other technologies), it is well placed for robust location delivery.

**C. Discussion**

The mentioned case studies show how applications can retrieve relevant context data with few requests from the platform and how the different APIs play together. The 1st case study presents the advantage of combining the large-scale network of SENSE-SATION with the possibility of direct interaction with users. The following collaboration supports the utility of the RESTful APIs: first the application picks only those users that satisfy a certain requirement from the backend server database cache. Then it requests relevant users directly to take the picture according to given instructions.

The 2nd case study demonstrates the utility of the SENSE-SATION platform for a mobile phone application. The application can leverage the local RESTful API available on the mobile web server to retrieve the user’s current location. Further, the application uses SENSE-SATION remote API and particularly a virtual sensor that uses average information of all users of the network. This functionality lines up with the field of people-centric sensing and depicts how anonymously collected data can be crucial for context-aware (mobile) applications.

**VII. PRIVACY & SECURITY**

In SNS privacy is a diversely discussed topic ranging from technical requirements to societal impact. In the following we will not discuss advantages and threats that arise from sharing information. We believe that users should be in control of their information and that they should have means to control (retrospectively) what they share with whom. Platforms like SENSE-SATION dealing with a multitude of sensors and personal data has to provide users means for securing their privacy.

SENSE-SATION’s main privacy concept is similar to most of popular social networks, namely privacy is adjusted in regard to applications and users. When an application is created, it declares which of SENSE-SATION’s available sensors it is going to use. This setting cannot be changed later. Thereby, it limits itself to a subset of users’ personal data it is going to use to achieve its functionality and which will possibly be shared with other users. Using applications, which exploit SENSE-SATION platform, causes the users to grant access to the required sensors during the OAuth handshaking. OAuth guarantees that no personal data can be read from any unauthorized application and all types of granting access can be revoked at any time through the user’s profile. Besides, the users have complete control over collecting and sharing the data. User privacy is also highly influenced by the privacy default rules of a system. As it is reported in [12], only a small percentage of users adjust their privacy rules, hence making the – often not very restrictive – default rules the applied rules. Therefore SENSE-SATION follows an opt-in approach, where any access to private user data is denied by default and must be explicitly granted by the user.

Another aspect of privacy specific to SENSE-SATION is the question how the whole system can be secured against multi-
cial sensor code, which can be introduced through the website from any developer by creating a new virtual sensor. Our solution is two-fold: first, developers need the possibility to test their code. This is easily done remotely on their own phones, since neither the system nor other users can be influenced by malicious sensor code on a developer’s mobile. The second is how publishing a new sensor to the community is handled. While there is the possibility to use a sandbox environment for 3rd party applications to secure the system against malware, it always limits developers. Therefore and because creation of new sensors is expected to happen less frequent (at least during the course of time) than creation of new applications, a group of trusted developers inspect the sensor code manually before it is automatically transferred to the online backend server and mobile clients. Once published, sensor code cannot be altered again. This may seem like a restriction for developers, but can be easily overcome by just using a versioning system for created sensors. Apps that want to use a newer sensor version have to ask the user for granting permission again. Finally, all sensor code and the code of applications hosted on SENSE-SATION website is open source and publicly accessible which is known to firm trust regarding the privacy.

VIII. CONCLUSION & FUTURE WORK

In this paper we presented the SENSE-SATION system, which facilitates the development of (web) applications based on a community of mobile phones. The system consists of a runtime environment that is installed on the phone and a web based application platform. The central goal behind this research is to ease the development of distributed web (and mobile phone) applications, which make use of phone sensors. Our approach allows developing incrementally applications with web programming skills. The advantage of the SENSE-SATION platform is that all typical sensors (accelerometer, GPS, camera, etc.), data (e.g., phone book, profile settings), and functionalities (e.g., sending SMS) on the phone become available to the developer, without (or only limited) prerequisite knowledge of phone operating systems and development. Besides, the approach allows sharing and reusing of components between different applications. By defining new functionalities and virtual sensors within the web-based platform, these functions and virtual sensors can be used in following projects and also with the community of the developers.

The provided use cases show that typical context-aware applications that rely on phone sensors are very easy to implement through the SENSE-SATION platform and we expect web developers with no experience in phone programming to be able to use it instantly. The presented API calls show relevant context information can be retrieved with few requests from the platform and how the different APIs work together. However, we are aware of the potential threat to the privacy of the user by providing remote access to a sensor from a device carried by the user. Therefore we have created mechanisms to give end users control over what and with whom data should be shared.

Our next step is the evaluation of the system with web and mobile developers before we are going to release the system and its source code publicly.

REFERENCES