AN EXTENSIBLE MOBILE SENSING PLATFORM FOR MHEALTH AND TELEMEDICINE APPLICATIONS

Gabor Novak, Darren Carlson, Stan Jarzabek

National University of Singapore, Felicitous Computing Institute, Republic of Singapore

Abstract

Smartphone apps with self-monitoring and sensing capabilities can help in disease prevention; however, such context-aware applications are difficult to develop, due to the complexities of sensor data acquisition, context modeling, and data management. To ease the development of mHealth and Telemedicine apps, we developed the Mobile Sensing Framework (MSF), which dynamically installs device appropriate context sensing plug-ins that provide a wealth of information about users' mental and physical states. The MSF automatically collects information about incoming/outgoing/missed calls; apps usage; sound pressure levels; light sensor values; movement data (e.g., step count); location; heart rate; etc. The MSF also includes a searchable object-based persistence layer, which is capable of rapidly serializing and de-serializing detected context data. Collected data are stored securely in the phone's database, where they can be retrieved by applications for local analysis, remote monitoring and alert generation. We developed a fully operational prototype of the MSF platform that was validated using several Android-based devices. This paper presents an overview of our approach along with a description of the experiments that are being conducted using the MSF prototype.

Keywords

mHealth, Telemedicine, Mobile Sensing, Context-awareness, Ambient Dynamix, Android.

Introduction

Smartphones represent powerful platforms for disease prevention and health interventions. Disease risk conditions can be mapped to data collected via phone sensors and self-reports, providing users and doctors access to rich health data or broadcasting alerts to caregivers. In this article we introduce our rapid prototyping, mobile sensing platform for mHealth and Telemedicine applications that can dynamically adapt to the device's capabilities while automatically monitoring and reporting user behavior using the device's inbuilt sensors, connected external sensors, and virtual sensors.

Motivation

As mobile technologies advance, developers are increasingly interested in creating applications that are able to fluidly adapt to the needs and circumstances of their users. Contextual information extracted from the user's environment can be used to enable an app to adapt its runtime behavior and capabilities to better fit a user's changing situation and requirements [1]. Due to the complexity of context sensing and acting, middleware is often used to orchestrate context-aware adaptation in mobile applications [2]. Unfortunately, existing mobile context frameworks are often difficult to use, lack appropriate security features, support few context types, and are unable to integrate new (or updated) capabilities at runtime [3]. The lack of context framework support is particularly evident in the mHealth and telemedicine domains, which require advanced sensing capabilities.

The Mobile Sensing Framework

To address these challenges, we developed an extensible Mobile Sensing Framework (MSF), which dynamically installs device appropriate context sensing plug-ins into commodity mobile devices that provide a wealth of information about users' mental and physical states. The MSF automatically collects information about phone usage patterns such as incoming/outgoing/missed calls; apps usage; sound pressure; light sensor value; movement (e.g., step count); location; etc. The MSF also contains the necessary mechanisms to manage and persist data from the different sensor sources, provides additional functions, such as location based notification system, self-reporting capabilites and includes flexible query mechanisms that allows applications to query current and historical context data using simple interfaces.

The MSF leverages the Ambient Dynamix plug-and-play context framework for Android [4]. Dynamix enables mobile apps and websites to fluidly interact with the physical world through sensing and actuation plug-ins that can be installed *on-demand*. Dynamix runs as lightweight background service on the user's mobile device, leveraging the device itself as a sensing, processing and communications platform. Dynamix automatically discovers, downloads and installs the plug-ins needed for a given context sensing or acting task. When the user changes environments, new or updated plug-ins can be deployed to the device at runtime, without the need to restart the application or framework. An overview of the Dynamix framework is shown in Figure 1.

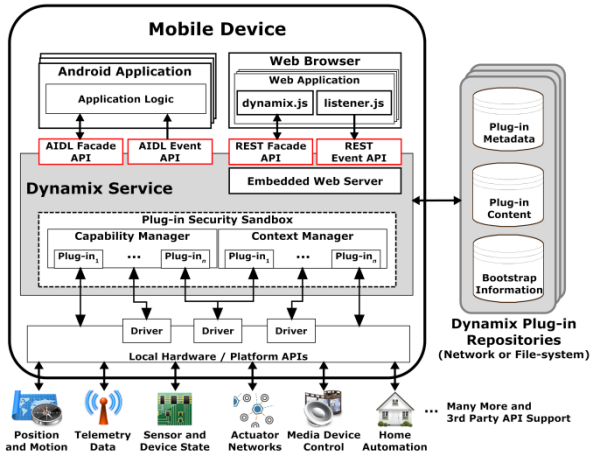

Figure 1: Overview of the Dynamix Architecture

The MSF builds upon Dynamix, which enables the MSF to benefit from the rich contextual information provided by various plug-ins. The relationship between the MSF and Dynamix is shown in Figure 2.

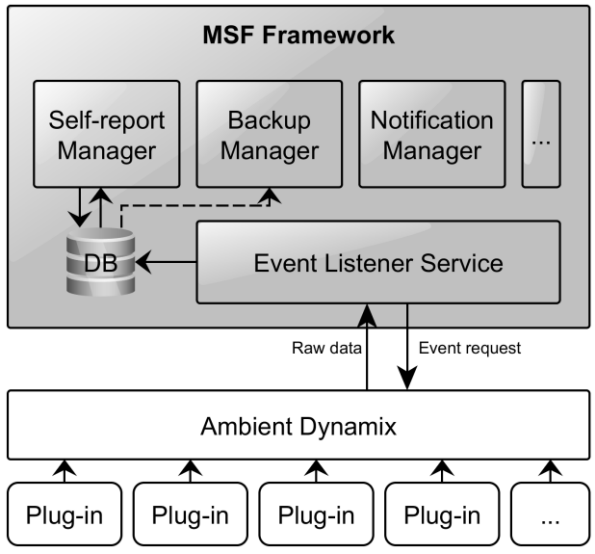

Figure 2: Overview of the MSF and Dynamix

Depending on the types of data sources and sensors available on the user's device, the MSF utilizes Dynamix to download and install appropriate context sensing plug-ins at runtime. Dynamix plug-ins can run exclusively on the device or in tandem with additional processing and storage facilities in the cloud. The MSF can utilize existing Dynamix plug-ins or custom Dynamix plug-ins, which can be hosted in private repositories or on the file-system. We created plug-in specific persistence classes that handle data serialization into the MSF's object-based database. The developer can access the data from the database any time via object-based queries. An example of the various context data available from the MSF is shown in Table 1.

Table 1: Example MSF Context Data

Context Data	Description
Application log	The name of running application and their run duration.
Battery level	The current battery charge level.
Browser history	Visited Websites (identified by a unique hash for privacy).
Heart rate	Heart rate data from a connected Zephyr HxM HRM device.
Calls	Incoming, outgoing, and missed calls (identified by a unique hash for privacy) with call duration.
Location	The device's geo-location.
Light Level	Lux values from the device's photodetector sensor.
SMS Data	SMS data (identified by a unique hash for privacy) that includes the number of words.
Sound pressure level	The calculated decibel value of the user's ambient environment.
Pedometer	Calculated step information from the gyroscope and accelerometer.
Wireless devices	Nearby Bluetooth devices and WiFi hotspots.
Weather information	Local weather information.
Real life event	Labeled data (e.g., the user drinks a cup of coffee).

The connection between the MSF and Dynamix is two-way, because there are two kind of events in the MSF: *request type* events and *auto triggered* events. Auto triggered events, such as a phone call or received SMS data, happen without user interaction. When these events occur, Dynamix automatically sends the raw data to the MSF. For request type events, such as sound pressure level or light level, the MSF framework periodically sends context scan requests to Dynamix, which responds with the requested data. For request type events, the developer can configure the time interval between two event requests.

The MSF is configured through an XML file that defines which events the will be collected. There are also

options to define the collection behavior for each event type, such as how often to collect data based on the device's battery level (e.g., collect less location data when the battery level is low to conserve power).

Additional MSF Functionality

Although the MSF is capable of fully automatic context sensing, it also provides built-in functionality for collecting and storing labeled data based on self-reports entered by the user. Since users may forget to enter self-reports, the MSF's built-in location based notification system can be used to alert the user during specific time intervals and locations. The user receives location based notifications when the target location is entered within a specified time interval.

The MSF provides several predefined self-report classes, and developers can easily add new types. The default self-reports contains classes based on PAD model [13], which provide information regarding user arousal, valence (pleasure-displeasure), and dominance (control). There is also default self-reports for collecting discrete emotions [5] and labeled values.

The collected context data and self-reports are stored in the phone database using an ORM framework. We also created a backup system within the MSF for offline data analyses. The backup system is able to export the stored data into standard CSV files, which are stored on the phone's SD card. The framework provides functionality to create an archive file from the saved data and send it to a server. The developer can configure this mechanism in the MSF's XML configuration file. In the near future we would like to include popular cloud storage mechanisms, such as Google Cloud Storage.

The MSF also provides mechanisms for creating and displaying chart-based visualizations of the collected data using the achartengine library[1]. As an example, the MSF can generate pie charts that show the dispersion of the application log and bar charts that visualize the average sound pressure level and the light level values.

Experiments and Applications

We are conducting experimental studies using the MSF to facilitate the rapid development of medical mobile applications with context sensing capabilities. Our current study aims at inferring the user's mood from data collected by MSF. Emotions have much to do with our health, and monitoring mood changes of patients and patient performance in life situations is important for the effective treatment of many medical conditions. We are working together with therapists from the Psychological Medicine department at the National University of Singapore (NUS) to provide mobile support for Cognitive Behavioral Therapy (CBT). Interventions to be completed by patients in-between therapy sessions (i.e., *Homework*) is a central concept in CBT. We are creating a CBT Assistant app, which conducts CBT Homework assignments using the user's mobile device. When the CBT Assistant is equipped with mood sensing capability, it will be able to trigger interventions that are customized to the patient's situation, and assess the efficacy of CBT interventions in the short- and long-term, providing invaluable input to both the patient and the doctor.

For the mood inferring study, we implemented a self-reporting app called Emotion Tracker and integrated it with MSF. Emotion Tracker provides self-reporting functionality that allows the user to log mood using AffectButton [12] and the PAD model [13], and input labeled data such as productivity at work, socializing, sleep quality, diet, exercise, and hobbies. Emotion Tracker annotates context data collected by MSF with self-reported data.

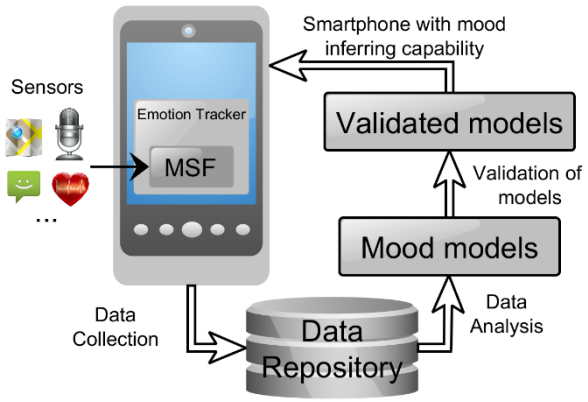

Figure 3. Mood inferring study

The mood inferring study includes data collection, data analysis (building mood prediction models), and validation of mood models (see Figure 3).

Data collection. Participants carry smartphones with the Emotion Tracker installed and the MSF collects sensor data. Participants self-report their mood three times daily and activities once, at the end of the day. Sensor data annotated with self-reported data is stored on the phone and then uploaded to our server.

Data analysis. We analyze the collected data to find statistically significant correlations between a person's mood and data collected via mobile phone sensors. We build *mood prediction models* based on these correlations. A mood prediction model can infer a user's mood from new data collected by sensors.

Validation of mood prediction models. Participant's carry phones with the Emotion Tracker and mood prediction models installed. Mood models infer user's mood from sensor data collected in real life. Inferred mood is compared with self-reported mood.

[1] http://www.achartengine.org/

In addition to mobile support for CBT, we plan to investigate the application of MSF data in risk prevention. Risk conditions for a given disease can be mapped to data collected via sensors and information self-reported by phone users. Having detected risk conditions, smartphones can alert users or their doctors about the situation. In this context, physiological information about the patient becomes highly relevant.

Related Work

There are several context sensing frameworks related to our work, notably the Fünf Open Sensing framework [6], the Emotionsense [7] framework and Purple Robot[2]. These projects all collect contextual data using phone sensors and provide several data storage options, including the file-system, remote servers and cloud-services such as Dropbox. These projects utilize the notion of sensing plug-ins, which are discrete units of code that are statically compiled into the hosting application, which allows the frameworks to be extended by additional sensing functionality.

Although these projects are similar in spirit to our approach, there are several notable differences. First, since the MSF utilizes Dynamix, it supports the discovery and installation of new or updated context sensing plug-ins *during runtime* (enabling dynamic adaptation of sensing capabilities to the device and user). The MSF also provides a flexible query interface that enables applications to search for historical context information during runtime *in addition* to support for offline data mining. Finally, the MSF provides integrated self-reporting capabilities and provides data visualization support.

Conclusion and Future Work

In this paper we presented an overview of the Mobile Sensing Framework (MSF), which enables the rapid creation of mHealth and Telemedicine applications that can dynamically adapt to a mobile device's capabilities. Our approach leverages the Dynamix Framework, which enables the MSF to request the dynamic installation of context sensing plug-ins that are best suited to the user's environment and application scenario. Collected data are stored securely in the phone's database, where they can be retrieved by applications for local analysis, remote monitoring, and alert generation. We developed a fully operational prototype of the MSF platform that was validated using several Android-based devices. We also implemented various MSF plug-ins that provide a wealth of information about users' mental and physical states.

In terms of future work, we are planning to extend our approach to support additional wearable sensor's [9], such as Electrocardiography (ECG) [10] and Galvanic skin response (GSR) [11]. We are also considering privacy-centric methods of extracting sound features [14] from incoming and outgoing calls [8] as a mechanism of determining emotional states. These data will be used to augment our ongoing mood inferring study, which will be presented in an upcoming paper.

[2] http://tech.cbits.northwestern.edu/purple-robot/

Acknowledgement

The work has been supported by Microsoft research grant R-252-000-521-592 and NUS grants R-252-000-473-133 and R-252-000-473-750.

References

[1] B.N. Schilit, N. Adams and R. Want, "Context-Aware Computing Applications." Proc. of the Workshop on Mobile Computing Systems and Applications, IEEE Computer Society, 1994, pp. 85-90.

[2] M. Modahl, et al., "Toward a Standard Ubiquitous Computing Framework." Proc. of the ACM Workshop on Middleware for Pervasive and Ad-hoc Computing, ACM Press, New York, NY, USA, 2004, pp. 135 - 139.

[3] M. Baldauf, S. Dustdar and F. Rosenberg, "A Survey on Context-aware Systems." International Journal of Ad Hoc and Ubiquitous Computing, vol. 2, no. 4, 2007, pp. 263-277.

[4] D. Carlson and A. Schrader, "Dynamix: An Open Plug-and-Play Context Framework for Android." Proc. of the 3rd International Conference on the Internet of Things (IoT2012), 2012.

[5] P. Ekman. "An argument for basic emotions," Cognition and Emotion, 6(3/4):169-200, 1992.

[6] N. Aharony, W. Pan, C. Ip, I Khayal, and A. Pentland. "The social fmri: measuring, understanding, and designing social mechanisms in the real world." Proc. of the 13th International Conference on Ubiquitous Computing (UbiComp '11), ACM, New York, NY, USA, 2011.

[7] N. Lathia, R. Kiran, M. Cecilia and G. Roussos. "Open source smartphone libraries for computational social science." Proc. of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication, pp. 911-920. ACM, 2013.

[8] D. Ververidis and K. Constantine. "Automatic speech classification to five emotional states based on gender information." Proc. of the 12th european signal Processing conference, 2004.

[9] A. Haag, et al. "Emotion recognition using Bio-Sensors : first steps towards an automatic system." Proc. of the Affective Dialogue Systems, Tutorial and Research Workshop, Kloster Irsee, Germany, 2004, 36–48.

[10] Xu, Y., Liu, G., Hao, M., Wen, W., & Huang, X. (2010). "Analysis of affective ECG signals toward emotion recognition." Journal of Electronics (China), 27(1), 8-14.

[11] C. Tronstad, et al. "Electrical measurement of sweat activity." Physiological Measurement 29, no. 6 (2008): S407.

[12] J. Broekens and W.P. Brinkman. "AffectButton: Towards a Standard for Dynamic Affective User Feedback." Proc. of the 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops, 2009, pp. 1-8

[13] A. Meharabian. "Comparison of the PAD and PANAS as Models for Describing Emotions and for Differentiating Anxiety from Depression." Journal of Psychopathology and Behavioral Assessment, Vol 19, No. 4, 1997, pp. 331-357

[14] S. Sarda, et al. "Real-Time Feedback System for Monitoring and Facilitating Discussions." International Workshop Series on Spoken Dialogue Systems Technology, 2012.