ContextProvider: Context Awareness for Medical Monitoring Applications

Michael Mitchell, Christopher Meyers, An-I Andy Wang, Gary Tyson

Abstract—**Smartphones are sensor-rich and Internet-enabled. With their on-board sensors, web services, social media, and external biosensors, smartphones can provide contextual information about the device, user, and environment, thereby enabling the creation of rich, biologically driven applications.**

We introduce ContextProvider, a framework that offers a unified, query-able interface to contextual data on the device. Unlike other context-based frameworks, ContextProvider offers interactive user feedback, self-adaptive sensor polling, and minimal reliance on third-party infrastructure. ContextProvider also allows for rapid development of new context and bio-aware applications.

Evaluation of ContextProvider shows the incorporation of an additional monitoring sensor into the framework with fewer than 100 lines of Java code. With adaptive sensor monitoring, power consumption per sensor can be reduced down to 1% overhead. Finally, through the use of context, accuracy of data interpretation can be improved by up to 80%.

I. Introduction

Early detection remains a powerful tool in the fight against illnesses. Unfortunately, despite regular health exams, the symptoms of many disorders appear sporadically. Treatable conditions may evade early detection, limiting treatment options. However, studies have shown that up to 20% of mortalities can be prevented with personal monitoring systems [1, 2].

Personal health-monitoring systems have historically relied on sensors connected to external storage, as the gathered data is offloaded to an external computer for analysis. However, the advent of sensor-rich and Internet-enabled smartphones is enabling new applications for health, wellness, and entertainment. With their built-in sensors, web services, social media, and external biosensors, smartphones can offer context-rich hints to guide application behaviors.

While many context-based monitoring systems exist, most such systems lack interactivity [6, 9]. The lack of online feedback can result in missed opportunities for users to help disambiguate the collected data. Other monitoring services either have limited extensibility for new sensors due to vendor-specific context formats [5, 8], or they rely on third-party support for storage and processing and impose deployment restrictions [5-7]. For most systems, power management is not a first-order concern, and thus is either not incorporated into the design or not evaluated [5-9].

This work presents ContextProvider, a framework for integrating contextual data streams collected from smartphones, which provides a unified, query-able interface to all contextual data on the device.

ContextProvider provides the online information needed for users and caregivers to react quickly to life-threatening events. It also provides long-term information needed by healthcare providers to make lifestyle and medication recommendations. ContextProvider can be tailored to the individual, providing customizable thresholds, contextual response, and the ability to integrate more sensors. Finally, with its adaptive sensor monitoring, ContextProvider keeps power overhead low. In combination, these benefits afford the user more independence and a higher quality of life.

II. Design

ContextProvider is a framework that collects, analyzes, and archives the daily context extracted from on-device sensors, web services, and social media. ContextProvider layers between underlying sensors and various applications (Figure 1). It consists of an online supervised context learning component, an adaptive frequency polling module, a long-term data repository, the capability to offload analyses for non-time-sensitive data trends, a graphical user interface for user feedback and configurations, and context-aware medical applications.

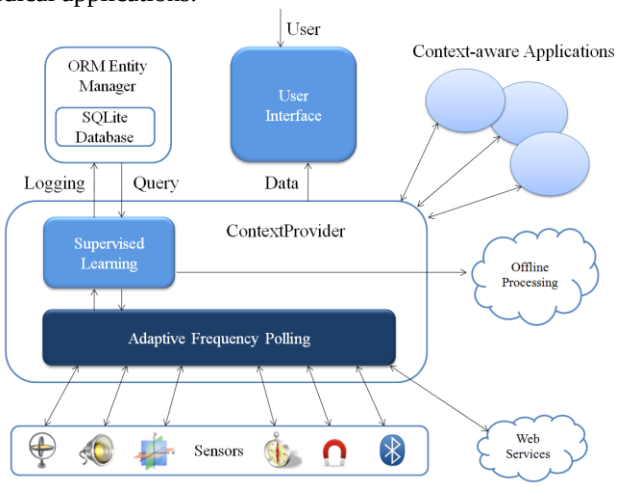

Figure 1 ContextProvider System Architecture.

A. Supervised Context Monitoring

Based on the current state and data patterns from biological and environmental sensors, ContextProvider can prompt users for feedback and confirmations to categorize

Manuscript received June 20, 2011.

M. Mitchell is with Florida State University, Tallahassee, Florida 32313 United States (mitchell@cs.fsu.edu).

C. Meyers is with Florida State University, Tallahassee, Florida 32313 United States (meyers@cs.fsu.edu).

A. Wang is with Florida State University, Tallahassee, Florida 32313 United States (awang@cs.fsu.edu).

G. Tyson is with Florida State University, Tallahassee, Florida 32313 United States (tyson@cs.fsu.edu).

events [16]. For example, if the heart-rate monitor encounters an increase in pulse rate with unfamiliar data patterns from other sensors, ContextProvider will prompt the user to verify whether the user needs medical attention. If medical attention is not required, the user is prompted to identify the current activity (e.g., exercising), so that various sensor readings and user feedback can be correlated to build a new context. Thus, the next time the heart-rate monitor detects an increase in pulse rate with similar data patterns from other sensors, it will infer the user-supplied context (e.g. exercising). Threshold parameters are adjusted automatically to reduce the rate of user confirmations.

This feedback is stored to be used in machine-learning algorithms to guide future interactions and reduce false alarms and unnecessary interactions [12]. This design is comparable to systems such as automated wandering detection for dementia patients [11]. For such cases, the data set can be flagged for analysis by medical professionals.

By incorporating feedback into the system, users are engaged in a dialog about their activities, an approach that makes it easier for them to better manage their health.

B. Self-Adaptive Sensor Polling

In sensor-rich smartphones, frequent readings from sensors may shorten the battery life significantly. Therefore, one goal for ContextProvider is to poll data infrequently, while yielding accurate contexts. This frequency varies by sensor and the context to be analyzed. For example, accurate gait analysis requires polling rates in excess of 50 Hz, while footstep detection requires only 20 Hz polling rate.

To reach this goal, ContextProvider lengthens data polling interval delays when it senses little or no change from specific sensor channels. That is, the context itself is used to optimize the context collection. If a change in context is detected, the system reverts to normal polling rates. Note that certain sensors, such as heart-rate monitors, need to have a maximum polling interval set for safety reasons.

The user can also specify a lower update rate (to the extent that it is safe) for each sensor in order to conserve power. Sensors such as GPS can be deactivated and, instead, rely on a less accurate position triangulated by cellular and WiFi networks.

With self-adaptive polling, the reduced frequency reduces the amount of data acquired as well as the associated storage. Furthermore, having fewer sensor polls reduces the number of system wakeups, which consume a significant amount of power.

C. Example of a Context-aware Medical Application: Tiered Emergency Response

When ContextProvider suspects a potential emergency (e.g., a person falling on the ground), the system checks for context to filter out obvious false alarms (e.g., an accidental drop of a phone followed by an immediate pickup of the phone and continuation of a phone call). If no such condition is met, notifications are then issued to the user, who may identify additional false positives (e.g., monitoring strap adjustment). If there is no response within a defined time, communication to the user's pre-specified social contacts is attempted. If contact is made, these individuals can evaluate the situation further and take appropriate action, such as alerting emergency personnel. If no contact is made with the user's social contacts, the system further escalates the alert and contacts user-specified emergency services directly. This escalation process reduces the number of, false alerts. All threshold values can be customized based on individual needs.

D. Extensibility

To ease the incorporation of new sensors from diverse vendors, ContextProvider uses existing APIs and standards wherever possible. The synthesis of each context can be based on multiple sensors in a one-to-many fashion. Sensor-reading requests from multiple sources of context synthesis are coordinated and consolidated for better efficiency. Contexts can also be derived from other contexts, structured in a general graph to make fuller use of sensor and contextual information.

E. Long-term Data Analysis

In addition to online information, patterns and trends can be extracted from incremental checkpoints over time, thus enabling early detection of an emerging condition. With the user's permission, health history and analysis can be uploaded from the device to health professionals periodically or on demand.

III. IMPLEMENTATION

Leveraging Google's open-source Android platform, ContextProvider extends the Android content provider [10] to acquire data, process contexts, and facilitate the sharing of contextual information. ContextProvider was developed on the Android 2.2 platform with additional support from Smart-Entity Object Relational Manager [15] to store context objects in SQLite. Current supervised learning mechanisms remain primitive and rely heavily on empirically derived rules to guide context fusion. ContextProvider consists of 1,200 lines for monitoring 6 hardware sensors (e.g., accelerometer) and 4 data sources (e.g., website services); 1,100 lines for deriving the 30 context subcategories; 700 lines for communication services; and the remaining for the user interface. Combined, the framework consists of 4,627 lines of Java code and 712 lines of XML.

ContextProvider is an always-on application and consists of multiple system services and Android broadcast receivers (event handlers) coordinated primarily through Android intents (an asynchronous mechanism that allows data sharing between processes) [10].

A. Context Monitors

ContextProvider centers around six context categories (Table 1). Each category is managed by a construct called a monitor, which is either a task running periodically to monitor context or an extension of broadcast receivers. The

monitors themselves can handle multiple sensors and fuse information.

The location monitor tracks GPS and network-based positions and is aware of user-entered places such as home and work, as well as reverse geo-coded address and establishment lookup. The movement monitor utilizes bearing and speed calculated from GPS data, as well as accelerometer motion and magnetic orientation to determine the movement of the user. The weather monitor uses the position determined from the location monitor to query web services for localized conditions and forecasts. The social monitor tracks phone and SMS usage, as well as following user interactions with social networking. The system monitor receives all transmissions broadcast from the Android system, including power, service states, and user/device interactions. Finally, the derived monitor fuses the data gathered from the other sensors, as well as other external sensors, in order to predict higher-level contexts, including device location relative to user, activity, and mood.

The system and social monitors are implemented as broadcast receivers. Rather than polling at periodic intervals, these monitors are event-driven, acting as listeners for intents and various handlers responding to specific broadcasts. System messages, power state changes, user/device interaction, and phone state changes are examples of intents received by these listeners.

The ContextProvider framework can be extended to integrate new polling-based devices and services through the static timer task method.

Location	**Movement**	**Weather**
GPS	State	Temperature
Address	Speed	Condition
Neighborhood	Bearing	Humidity
Zip Code	Step Count	Wind
Altitude	LastStep	HazardLevel
Social	**System**	**Derived**
State	State	Place
Last Comm.	BatteryLevel	Activity
Contact	LastPlugged	Shelter
Msg LastIn	UserLastPresent	OnPerson
Msg LastOut	WiFi SSID	Mood

Table 1: Partial Enumeration of Collected and Derived Contexts.

B. Sensors and Web Services

Sensor information is read using asynchronous callbacks, abstracted as Android sensor event listeners. An event-listener function can be registered to be executed when a particular sensor's state changes. Read sensor values are passed to the function. Parameters such as event-trigger rate, accuracy, and type can be configured for the sensor.

Data is obtained not only from sensors but also from web services and social networking. Information such as harsh weather conditions, combined with location and time of day, can be useful in determining if a dementia patient is wandering [11]. External data requires a different access method than sensor data but with the same goal, which is to provide an up-to-date view of the world. Rather than receiving local notification of data changes, outside data sources must be polled to detect changes.

C. Data Acquisition

The data-acquisition code translates raw data from on-board sensors, web services, social media, and external biosensors into a low-level state for building a higher-level context. Android sensor-event listeners are used to gather data from accelerometers, GPS, network, and light, which are then stored as local objects and logged to a database for future processing.

Simple contextual processing also takes place at the data acquisition phase. For example, footstep detection can be performed online with the current and previous accelerometer readings. Although this context formation can be done at a later time, the footstep detection, combined with the user's orientation can assist in determining indoor locations and contexts in a timely manner when GPS is not available.

D. Processing

Higher-level context is constructed from the state provided by data acquisition. For example, GPS and network provide location awareness; accelerometer, GPS, and network are used to derive movement. The location provider builds a higher-level context that includes: current address, city, zip code, proximity to requested businesses or addresses, and whether the user is indoors or outdoors. Movement context can derive the current action being performed, such as walking, running, or driving as well as the current speed and direction in which the user is traveling.

E. Sharing

ContextProvider is most useful when utilized by outside applications, thus providing an interface to context- aware applications. ContextProvider supports the use of Android intents to provide a query-able always-on interface.

IV. EVALUATION

For the evaluation, we used three HTC G1/Dream Android devices running Android 2.2 platform, CyanogenMod 6.1-DS, and Linux 2.6.35. Between experiments, each phone was reset to the factory preset setting with a freshly installed system image and cellular radio disabled. The battery had a 1300mA rating.

Overheads: PowerTutor [14] was used to measure the power overhead of ContextProvider running on a fully charged idle phone over an hour with three repetitions. The polling rate of the accelerometer was varied between 50 Hz and 0.1 Hz, the range allowed by ContextProvider's adaptive polling mechanism. The resulting power requirement varied from 964 to 45 mW per hour. Thus, during periods of idle intervals (e.g., phone placed on table), the overhead imposed by ContextProvider per sensor was within 1% (based on the phone's 3.7V voltage rating). (Of course, if a sensor is chosen to be disabled, the overhead will be zero.)

The storage requirement is also dependent on the desired context tracking intervals. For low-resolution tracking, the system requires 11 KB/hour (1.1KB/hour with compression). For high-resolution tracking, such as performing gait analysis, the system requires 1 MB/hour (700 KB/hour compressed).

Extensibility: Adding a new sensor into the ContextProvider framework requires fewer than 100 lines of code; a new context, as few as 20 lines of code.

Example context usage: Two sets of $3^2 6$ full factorial experiments [13] were conducted evaluating ContextProvider's ability to use an ambient-light sensor and accelerometer to deduce the number of footsteps taken (and implicitly traveled distance indoors). The first set was a walking test (100 steps); the second set was a staircase ascension (40 steps) experiment. First, the ambient-light sensor is used to deduce the phone location relative to the body (in pocket, in hand, or in backpack). Low, medium, and high sensitivity settings (60, 100, and 140 m/s^2) were then tested for the footstep detection algorithm [16] against these three phone locations.

Table 2 shows the percent error under each experimental setting. The 90% confidence intervals for the walking and stair tests were within 16% and 36%, respectively. The results show that by exploiting context (location of the phone relative to the body) and by using the appropriate sensitivity setting within the context (low for pocket, high for hand, and middle for backpack), footstep prediction accuracy can be improved by 3% to 80%. Combined with user orientation information, this improvement can subsequently improve prediction of the user's indoor location and activity.

	Walk test (100 steps)			**Stair test (40 steps)**		
	Low	Middle	High	Low	Middle	High
Back-pack	43.50	**0.33**	8.17	85.00	**5.42**	105.4
Hand	84.33	55.17	**3.50**	72.50	50.42	**24.17**
Pocket	**37.17**	40.00	115.3	**12.50**	40.42	136.7

Table 2 Percent Error in Predicting the Number of Footsteps.

V. Future Work & Broader Implications

ContextProvider can be useful as a medical research tool when deployed on a large scale. A large volume of data could be provided to researchers and used in correlation studies to help identify new early warning signs of disease. It also has the potential to be used to facilitate large-scale user feedback for clinical trials and new treatment options.

VI. Conclusion

We have presented the design, prototype, and evaluation of ContextProvider, a novel context-aware framework. ContextProvider allows interactive user feedback to guide system monitoring behavior; it can be tailored to the individual; it uses the popular open-source Android framework and local processing in order to avoid reliance on third-party infrastructure; and it reuses existing APIs wherever possible to ease deployment of new sensors and development of new contexts and context-aware applications. ContextProvider can provide online information for users and caregivers to react to life-threatening events. It also provides the long-term information needed for health-care providers to make lifestyle and medication recommendations. In combination, these can afford the user greater independence and a higher quality of life.

With the ease to add new sensors and contexts, the ContextProvider system can be applied well beyond the medical applications discussed, with possibilities abound.

References

[1] Shah NB, Der E, Ruggerio C, Heidenreich PA, Massie BM. Prevention of hospitalizations for heart failure with an interactive home monitoring program. *Proceedings of the 69th Scientific Sessions of the American Heart Association*, 1997.

[2] Clark RA, Inglis CC, McAlister FA, Cleland JF, Sweet S. Telemonitoring or structured telephone support programmes for patients with chronic heart failure: Systematic review and meta-analysis, *BMJ*, 334:942 2007.

[3] Mitchell M, Sposaro F, Wang A, Tyson G. BEAT: Bio-Environmental Android Tracking. *Proceedings of the IEEE Topical Meeting on Biomedical Radio and Wireless Technologies, Networks, and Sensing Systems (RWW)*, 2011

[4] Ledijdekkers P, Gay V. Personal heart monitoring system using smart phones to detect life threatening arrhythmias. *Proceedings of the 19th IEEE Symposium on Computer-Based Medical System.* 2006.

[5] Jung TM, Lee YS, Cho SB. Mobile sync-application for life logging and high-level context using Bayesian network. *Knowledge Management and Acquisition for Smart Systems and Services*, 6232:223-234, 2010.

[6] Sridevi S, Sayantani B, Amutha KP, Mohan CM, Pitchiah R. Context aware health monitoring system. *Medical Biometrics*, 6165:249-257, 2010.

[7] Bardram, JE, Hansen TR, Initials. The aware architecture: supporting context-mediated social awareness in mobile cooperation. *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, 2004.

[8] Krause A, Smailagic A, Siewiorek DP, Context-Aware Mobile Computing: Learning Context-Dependent Personal Preferences from a Wearable Sensor Array, *IEEE Transactions on Mobile Computing*, 5(2):113-127, 2006.

[9] Fabiana G. Marinho, Fabrício Lima, João B. Ferreira Filho, Lincoln Rocha, Marcio E. F. Maia, Saulo B. de Aguiar, Valéria L. L. Dantas, Windson Viana, Rossana M. C. Andrade and Eldânae Teixeira, et al. A Software Product Line for the Mobile and Context-Aware Applications Domain. *Software Product Lines: Going Beyond, Lecture Notes in Computer Science*, 6287:346-360, 2010.

[10] Google. Android developer's site. July 2010. developer.android.com.

[11] Sposaro F, Danielson J, Tyson G. iwander: An android application for dementia patients. *Proceedings of 2010 IEEE EMBS*, 2010.

[12] Sposaro F. and Tyson G. ifall: An android application for fall monitoring and response. *Proceedings of 2009 IEEE EMBS*, 2009.

[13] Jain R. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc. 1991.

[14] Zhang L, Tiwana B, Qian Z, Wang Z, Dick R, Mao ZM, Yang L. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones. *Proceedings of CODES+ISSS*, 2010.

[15] Smart-entity. A simple ORM for Android. June 2011. code.google.com/p/smart-entity.

[16] Pedometer. Android app that watches your every step. June 2011. code.google.com/p/pedometer.