MyExperience: A System for *In situ* Tracing and Capturing of User Feedback on Mobile Phones

Jon Froehlich[1], Mike Y. Chen[2*], Sunny Consolvo[3], Beverly Harrison[3], James A. Landay[1,3]

[1]Computer Science and Engineering
University of Washington
Seattle, WA 98195
{jfroehli, landay}@cs.washington.edu

[2]Ludic Labs
107 South B Street, Suite 200
San Mateo, CA 94401
mike@ludic-labs.com

[3]Intel Research Seattle
1100 NE 45th Street, 6th Floor
Seattle, WA 98105
{beverly.harrison, sunny.consolvo}@intel.com

ABSTRACT

This paper presents MyExperience, a system for capturing both objective and subjective *in situ* data on mobile computing activities. MyExperience combines the following two techniques: 1) passive logging of device usage, user context, and environmental sensor readings, and 2) active context-triggered user experience sampling to collect *in situ*, subjective user feedback. MyExperience currently runs on mobile phones and supports logging of more than 140 event types, including: 1) device usage such as communication, application usage, and media capture, 2) user context such as calendar appointments, and 3) environmental sensing such as Bluetooth and GPS. In addition, user experience sampling can be targeted to moments of interest by triggering off sensor readings. We present several case studies of field deployments on people's personal phones to demonstrate how MyExperience can be used effectively to understand how people use and experience mobile technology.

Categories and Subject Descriptors

H.1.2 [**Information Systems**] User/Machine Systems – Human Factors; H.5.2 [**User Interfaces**] Evaluation/Methodology, User-Centered Design; D.2.2 [**Design Tools and Techniques**]: Miscellaneous; J.4 [**Social and Behavioral Sciences**]: Psychology

General Terms

Design, Experimentation, Human Factors, Measurement, Performance

Keywords

In situ Evaluation, Experience Sampling Method (ESM), Context-Aware Systems, Mobile Computing, SmartPhones, User-Centered Design, Field Studies, Usage Logging, User Surveys, Toolkit

1. INTRODUCTION

Mobile computing has become an integral part of everyday life for many people, providing ubiquitous information access, entertainment, and helping people stay connected to work, friends, and families. The most popular mobile device, the mobile phone, has been adopted faster than any other technology in human history [9]. In 2006[1], the number of mobile phone subscribers surpassed 2.5 billion and has more than twice the number of PC users worldwide.

Researchers are struggling to catch up. Tools and techniques that have long been refined for studies of static computing environments do not translate well to the mobile environment [26]. In the mobile research community, automated tracing is widely used to provide insight into *what* and *when* [21], [34], [35]; however, it does not provide the *why*, such as user motivation, perception, and satisfaction. Also, infrastructure-based tracing does not have access to the variety of interesting sensing capabilities available on devices, such as wide-area location, device usage, mobility modes, and social situations. In addition, due to the lack of research tools to collect *in situ* feedback, user impact is often overlooked in the evaluation of a new system. Although many new emerging mobile research methods have been introduced recently, they still suffer from issues of scale, breadth, and artificial environments [17], [19], [28], [30].

Our goal is to collect quantitative and qualitative data *in the field* on people's personal devices in order to support studies of mobile technology usage and evaluation. For example, previous studies of SMS (short message service, *i.e.*, text messaging) have analyzed infrastructure system logs to understand text messaging behavior [35]. Though this method scales well and provides useful quantitative characterizations of SMS usage, it cannot be used to understand *why*, for example, the user chose SMS over another means of communication or *if* they perceived lag in the delivery system. Such information could be instrumental in designing and building future mobile communication applications and systems.

We have defined four key research challenges in supporting studies of mobile technology:

[*] This work was done while at Intel Research Seattle.

[1] http://www.wirelessintelligence.com

Coverage: *collecting rich features about the usage of interest.* Although there has been significant advancement in the sensing capabilities of mobile devices, technical and practical limitations to what can be sensed remain. For example, current global positioning system (GPS) technology does not have the resolution to distinguish between a bookstore and a café next door. Moreover, subjective data such as user perception, intention, and satisfaction cannot be sensed or observed directly, and requires user feedback.

Situated: *collecting real usage data as it occurs in its natural setting.* Mobile devices are used in a variety of contexts. To understand usage, data must be collected from people's personal mobile devices in their actual contexts. Also, because of recall bias, user feedback should be collected as close to the usage events as possible.

Scale: *collecting data with many users and devices over long periods of time.* This requires minimal obtrusiveness on the user experience. Approaches that require additional devices or have noticeable impact on the normal user experience lead to increased user burden. Also, because user sampling requires active user attention, irrelevant surveys should be avoided.

Robustness: *data durability on mobile devices that only have intermittent connectivity.* Since we cannot assume constant connectivity for mobile devices, it is important to support disconnected operations. Opportunistic synchronization to networked storage helps provide data durability and enables researchers to have early access to study data.

Our approach is to combine both automatic tracing and *in situ* user experience sampling to collect quantitative and qualitative data in the field. MyExperience is an open-source[2] data collection platform that allows researchers to automatically log sensor and phone usage data [28] and conditionally trigger self-report surveys based on sensed context [16].

MyExperience runs continuously with minimal impact on people's personal devices (*e.g.,* commodity cell phones and PDAs). It has an event-driven, Sensor-Trigger-Action architecture that efficiently processes a variety of sensed events including:

- Device usage such as communication (*e.g.,* phone calls, SMS), application usage (*e.g.,* games, music, and video), and media capture (*e.g.,* photos, video).
- User context such as calendar appointments, talking on phone, and contact book information.
- Environmental sensors such as Bluetooth, 802.11 wireless, Global System for Mobile Communications (GSM) scanning, and GPS.

To improve data durability and to cope with intermittent connectivity, MyExperience supports disconnected operations and can opportunistically synchronize sensor data and user feedback data whenever connectivity is available, such as via GPRS or 802.11 wireless, or when the device is connected to a networked PC.

To lower the barrier for researchers to use MyExperience, we have developed a lightweight XML interface for researchers to define survey questions and configure sensors, triggers, and actions. Embedded scripts are used to provide flexibility and expressiveness in specifying the conditions to trigger surveys. MyExperience supports sophisticated survey logic including multiple branching, parameterized questions, and persistent states.

We have conducted several field studies using MyExperience to study battery charging behavior, SMS usage, and place ratings related to travel patterns. Two of these studies had participants running MyExperience on their personal mobile phones configured to log more than 140 types of events, and participants reported no perceivable impact on the phones responsiveness or battery life.

Our contributions are as follows: 1) a non-proprietary system that collects *in situ* qualitative and quantitative usage data on people's personal mobile devices (*e.g., phones)* and 2) lowered the barrier for researchers to collect *in situ* usage data by providing a rich set of extensible sensors and actions with lightweight XML-based configuration.

1.1 Background

Current approaches to capturing mobile usage can be categorized into four classes: direct observation, lab-based evaluation, self-report, and automatic logging—each offering different, limited visibility into human behavior and user experience.

Automatic tracing typically records usage information passively without requiring user intervention (from the infrastructure [34] or directly from the device [28]). This technique scales well across users and collects large amounts of data; however, important information such as user intention and perception is lost. In contrast, *in situ* self-reports such as the Diary Method [26], [29] and the Experience Sampling Method (ESM) [5], [7], [16] offer insight into these otherwise imperceptible details, but at a cost of user involvement. Thus, the sampling rate is much lower than in automatic logging and the method does not scale as well (*e.g.,* participant compliance diminishes over time). Direct observation methods like shadowing [18], [30] can provide rich qualitative accounts of device usage and human behavior; however, the method can only be applied to a small number of participants at a time and not all contexts are conducive to being studied (*e.g.,* a formal business meeting) [12], [26]. In addition, it is subject to observer bias, and the small form factors of mobile devices make it difficult to observe both the participant and their device screens. Finally, laboratory methods offer an environment to rigidly control device and context parameters for experimentation; however, usage is artificial and removed from its natural setting. Figure 1 summarizes the relative situatedness and scalability of these techniques.

The rest of the paper is organized as follows: Section 2 provides a description of our design considerations; Section 3 describes the MyExperience architecture; Section 4 discusses our implementation; Section 5 presents our performance evaluation and three case studies; Section 6 discusses limitations, lessons learned from our deployments, and preliminary researcher feedback; Section 7 presents related work; and Section 8 summarizes our contributions and concludes with future work.

[2] http://myexperience.sourceforge.net

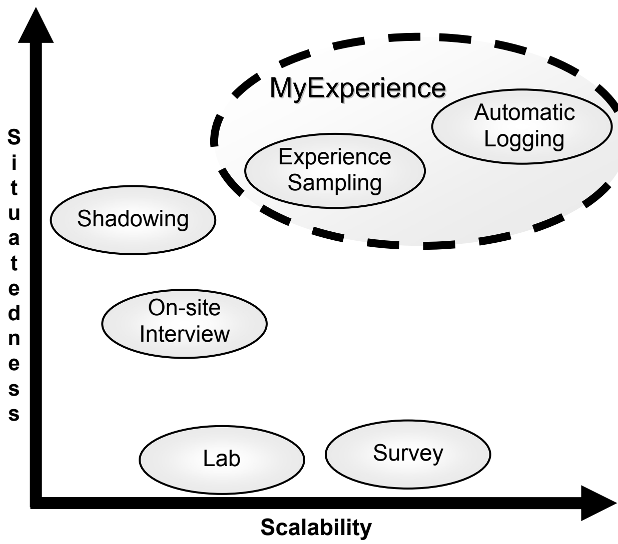

Figure 1: Summary of the scalability and situatedness of current data collection approaches and where MyExperience fits in.

2. DESIGN CONSIDERATIONS

As a mobile data collection platform, MyExperience has two distinct audiences: the *researchers* who will customize the tool for their studies and the *participants* who will run the tool in various capacities based on these customizations. As such, the design goals can be broken down into two sources of interrelated concern: those which affect the researchers and their ability to study the research element(s) of interest; and those which affect the participants and their mobile device.

2.1 Design Goals for Researchers

In enumerating properties of successful tools, Myers et al. [24] define two important characteristics: the *threshold* which represents how difficult it is to learn the system and the *ceiling* which describes how much can be accomplished by using the system. The ideal, of course, is the most challenging: a tool with a *low threshold* of learning and a *high ceiling* of functionality. Given the range of studies that we would like to support on the mobile phone (from studies of human behavior to technology use), we expect a broad user base, not all of whom will have backgrounds in computer science. Therefore, one overarching goal of our system is to provide a low barrier to configuring many elements in the system (*e.g.,* sensors and actions), while providing a high ceiling by making it easy to extend MyExperience with new classes of sensors and actions.

A second high level goal is to provide mechanisms to gather both qualitative and quantitative data. Objective data such as sensor streams and phone usage logs can be gathered without direct user intervention. This results in large quantities of traces, which can be used to discover usage patterns, correlate failure modes with inferred context, used as cue points during interviews, etc. However, automatically logged data is not always sufficiently descriptive. Self-report surveys can be used to collect data that is otherwise imperceptible (*e.g.,* user satisfaction, perceptions, or intentions) or account for limitations in sensor technologies (*e.g.,* by asking for self-reported location when GPS signal is lost).

Finally, because these studies will occur outside of the lab, the tool should allow researchers to retrieve collected data without requiring physical access to the device under study. This has several benefits: first, it creates a backup of data to reduce the window of data loss in the event that the mobile device is lost or damaged. Second, it provides immediate access to study data allowing for early detection of failure and preliminary data analysis. Such analysis could be used, for example, to customize interview sessions per participant according to their ongoing collected data. Lastly, researchers can monitor participation in near real time and intervene (*e.g.,* via an email or phone call) if a participant's responsiveness wanes over time. In addition, early discovery of noncompliance would allow researchers to replace those refractory participants before losing valuable study time.

2.2 Design Goals for Study Participants

To collect realistic usage data, it will be necessary to install the tool on a user's personal device (*e.g.,* his/her mobile phone). This has two interconnected implications for design. At the system level, the data collection tool should not noticeably impact the performance of the user's mobile phone (*e.g.,* by saturating the processor). A tool that affects the responsiveness of mobile phones will introduce user annoyance and possibly change the usage pattern.

Secondly, at the user level, the tool should be considerate of the situated use of a mobile phone. People carry their phones nearly everywhere they go and use them in a variety of contexts. Therefore, the tool should provide mechanisms to avoid interruptions at inopportune moments (*e.g.,* while the user is on the phone or in a meeting). Similarly, the tool should be able to abide by the current phone's profile (*e.g.,* silent vs. normal mode).

Our third design goal relates to the security and privacy of collected data. Given that the device may be collecting sensitive information (*e.g.,* location streams), the tool should offer mechanisms to protect the security and privacy of the data.

Finally, the tool's user interface must be designed for ease of use. Although mobile phones offer a familiar interface to most users, their screen size and input capability can make interactions difficult. Therefore, the user interface should require a minimal amount of interaction for successful use. Also, its accessibility should be configurable. For example, increasing color contrast, font size, and providing multimodal components when possible (*e.g.,* text-to-speech question interfaces).

3. ARCHITECTURE

MyExperience has an object-oriented, three-tiered, event-driven architecture of Sensors, Triggers and Actions. The *triggers* combine streams of *sensor* data with conditional logic to invoke *actions* (see Figure 2). A local database is used for persistent storage, which can be synchronized with a remote server.

Figure 2: The Sensor, Trigger, Action architecture.

3.1 Sensors

Sensors provide an abstraction to model device state, user interaction, and the environment. A sensor in MyExperience refers to both hardware sensors (*e.g.,* microphone, GSM radio, key presses) and software sensors (*e.g.,* sensing the current application, calendar appointment, incoming SMS).

MyExperience is designed around a state-based sensor abstraction. When a new state is entered, the previous state is automatically exited and a sensor event is generated. A sensor's state history (*e.g.,* time entered/exited) is automatically saved to the database, thus providing a log of sensor information.

Those sensors which do not require polling can extend directly from the top level abstract Sensor class. Each Sensor descendant must override three methods: start the sensor, stop the sensor, and return state type information. Other sensors may require polling; this functionality is provided by our framework. The PollingSensor extends from Sensor and defines an additional abstract method, Poll, which is called automatically on a thread from a thread pool based on a configurable polling interval.

A sensor's underlying state type can be a primitive type (*e.g.,* float, int, string) or a higher level object type (*e.g.,* calendar entry). The base State object supports tests for equality, and sensors that have comparable states can extend ComparableState to support comparison operations (*e.g.,* less than or greater than). MyExperience dynamically checks the state types and uses the comparison operations when appropriate.

MyExperience currently provides access to a multitude of sensor events including device usage, user context, and environmental sensing (*e.g.,*using the DeviceIdleSensor, SmsSensor, PhoneCallSensor, CalendarAppointmentSensor, RawGpsSensor, etc.). The prepackaged set of sensors within MyExperience may not be sufficient for every study (*e.g.,* a new sensor must be created to interface with new external hardware, such as a Bluetooth heart rate monitor). Sensors that interface with new hardware or abstract raw sensor bits into meaningful data are non-trivial to develop. However, once this low-level code has been written, it can be easily wrapped to fit into the MyExperience Sensor architecture and reused in future studies. In addition, as MyExperience is open-source, it is our hope that this new code would be uploaded and shared with others.

External sensors (*e.g.,* those that exist in an external .dll) can be loaded and configured dynamically without recompiling MyExperience. This plug-in architecture works for triggers and actions as well.

3.2 Triggers

Triggers provide a flexible, expressive mechanism for handling sensor data. In particular, they define the conditional logic that controls when to execute actions based on sensor states. For example, an upload action could be triggered every time a new digital photograph is taken.

Triggers maintain individual subscriptions to sensor events and evaluate a subscribed sensor's conditional logic only when it changes state. In the simplest case, triggers need only be associated with one sensor—say, a cell ID sensor—to achieve the desired behavior. For example, a trigger could be configured to fire an action upon every sensor state change (*e.g.,* every time the mobile phone connects to a new cell ID) or simply those state changes that are deemed relevant (*e.g.,* every time the mobile phone connects to a known cell ID like home or work).

To provide a low threshold and high ceiling, a trigger's conditional logic can be specified in two ways: 1) writing native C# code or 2) scripting. In the first case, a developer extends the base Trigger class, subscribes to relevant sensors, and overrides the OnSensorStateChanged method (which is invoked each time a subscribed sensor's state changes). In the second case, a script is dynamically loaded into a base Trigger object (via a set accessor), and the appropriate sensor subscriptions are made automatically based on the sensors that are referred to in the script.

Given that a sensor's state history can be automatically logged to a database, triggers can utilize more sophisticated conditional logic based on previous state behavior. MyExperience automatically maintains metadata about sensor state changes, such as the amount of time a sensor has existed in a given state, the time since the last state transition, or whether or not a state has ever been reached. These are useful for expressing logic like "has the user ever been to this place before?" or "when was the last time the user went here?" These can also be used to ensure that the user is in a stable state before triggering an action. For example, triggering an action after the user has been at a location for at least 10 minutes. In addition, this metadata can be used to detect sensor failure by observing no state changes after an expected amount of time, and triggering reminders such as recharging the sensor batteries or notifying research staff about potential malfunctions.

Trigger scripts can also be used to conditionally start / stop sensors. For example, to increase battery life, sensors that have significant power consumption (*e.g.,* 802.11 wireless and GPS) may be triggered to power down during certain periods.

3.3 Actions

Actions are code snippets that are triggered to execute based on sensor events. All actions derive from the Action class and implement the abstract method, Run.

3.3.1 The Action Scheduler

To better manage concurrent resource usage, we use a thread pool and a priority queue to execute actions based on start time. By default, an action's start time is set to run immediately. To allow flexible control over action execution, the start time, expiration time, and time-to-live are configurable through properties in the base Action class.

The priority queue is maintained by an action scheduler thread, which resides in a sleep state unless the following conditions are met: the queue is non-empty, the device is in an interruptible state (*e.g.,* the phone is not being used), and the impending action's start time has either been reached or surpassed. Once the scheduler removes an action from the priority queue, it is either executed on a thread or ignored depending upon its expiration interval.

3.3.2 Available Actions

Although we have not yet focused on providing an extensive action library, eleven actions have been created, a subset of which include:

- **CreateNewProcess:** launches an external application.
- **DatabaseSync:** synchronizes the local device data with a server backend.
- **Notification:** displays an alert with a customizable user interface, sound, vibration pattern, etc.
- **Player:** plays a .wav file, vibration, or LED flash pattern
- **ScreenShot:** takes a screenshot of the device's current screen.
- **SendSms:** sends a text message to one or more recipients.
- **Survey:** displays a fully customizable user-sampling survey (see Section 4.4).

4. IMPLEMENTATION

The MyExperience tool is implemented in approximately 30,000 lines of C# code using the .NET Compact Framework 2.0, which runs on Windows Mobile 2005[3] devices including SmartPhones, Pocket PCs, and Pocket PC phones. For scripting, MyExperience uses a JavaScript-like language called Simkin[4]. Simkin was originally an XML embeddable language for Java which we ported to C#. Data acquired from sensors, user interactions, and other elements are stored locally on the device in a SQL Server 2005 Mobile Edition database. Database replication is used to synchronize study data with a remote server.

MyExperience is designed to be used in two distinct ways: as a stand-alone application or as a library within another application. As a stand-alone application, MyExperience can be fully configured via XML and scripting, and does not require any C# programming or extensive mobile phone expertise. As a library, MyExperience is called as a .dll from within another application. Researchers who utilize MyExperience in this manner will likely do so to forgo the XML/scripting interface or to maintain tighter control over the user interface. Note that when MyExperience is invoked as a library, XML and scripting functionality can still be used.

4.1 Database

MyExperience uses SQL Server 2005 Mobile Edition (SQL Mobile) for local storage on the device and SQL Server 2005 (SQL Server) for the remote backend. Database replication is used to share and synchronize both the table schema and the data across the mobile device and the backend. As the underlying SQL technology shares a consistent API, the same set of data analysis tools can be applied to either database. Also, the transaction features of SQL Mobile help ensure data consistency in the face of device or power failure and intermittent network connectivity.

Opportunistic synchronization is used whenever the device has network connectivity (*e.g.,* via GPRS, EVDO, or 802.11 wireless). Also, connections can be tunneled over ActiveSync whenever the device is connected to a networked computer. Network synchronization is an optional feature (all data can be stored locally on the device).

To secure the transport and access to study data, we configured the backend to use HTTPS and per-user authentication. To help protect privacy while allowing usage patterns to be collected, we provide the option to use strong cryptographic hashing, SHA-1, to map personal information, such as phone numbers and contact names, to pseudonyms.

4.2 XML Interface

The XML interface provides an optional, lightweight method to control features of MyExperience without writing C# code. Our objective here is twofold. First, we believe that the XML interface lowers the barrier of use. Researchers who have limited experience programming mobile devices can still use the various features of MyExperience to conduct their studies. Second, we wanted to give experienced developers a straightforward method of specifying the user interface so that they could focus on building more complicated MyExperience components such as custom sensors and actions.

The XML file is broken down into four main sections: sensors, triggers, actions and the user interface. Each one is defined by specifying an XML header element with name and type information followed by an optional list of configurable properties. The type attribute in the XML header directly maps to a C# namespace and class. Custom classes (*e.g.,* sensors, actions or user interface widgets) external to MyExperience can be loaded dynamically without recompilation by specifying the classes' full namespace in the type attribute field. Similarly, the list of

---

[3] On February 12, 2007 Microsoft released Windows Mobile 2006. At the time of this writing, we have not yet obtained a Windows Mobile 2006 device to test MyExperience; however, we believe MyExperience should be compatible with the new operating system.

[4] http://www.simkin.co.uk/

configurable properties refers to the specified class's own C# accessor properties. Properties that are not explicitly set in the XML file are automatically assigned default values. C# reflection and object factories are used to dynamically instantiate the specified classes on demand.

The thin separation between XML and the underlying MyExperience codebase offers three primary benefits: first, it enables new, custom developed sensors and actions to be easily integrated and configured. Second, direct mapping of the namespace reduces ambiguity between declaration and function while allowing documentation to be automatically generated (*e.g.,* using NDoc). Finally, the specified elements can be easily type checked for correctness at load time.

4.2.1 Example Scenario

To illustrate how the MyExperience research tool may be configured via XML, we present the following simple scenario from one of our test deployments. In this example, we are interested in finding out how cell signal strength correlates with perceived voice call quality. We would like to automatically log cell ID and signal strength information and ask targeted questions about the users' perception of the quality of voice calls.

In our XML file, we refer to three sensors: a phone call sensor, which reveals state information about phone calls, a cell signal strength sensor, which provides a value indicating overall signal strength, and the cell ID sensor, which supplies data about connected cell towers. Other sensors may be relevant as well for an extended version of this study (*e.g.,* a headset sensor to contrast hands-free voice quality, an ambient audio sensor to determine background noise, etc.).

The sensors are defined as shown in Figure 3 and their state histories are automatically logged to the database; however, we still need to acquire qualitative user feedback about voice quality. To do this, a trigger and survey action must be defined. We continue the example in the next section.

```
<!--Activate phone call & cell related sensors-->
<sensor name="PhoneCall" type="PhoneCallSensor"/>
<sensor name="CellStrength"
        type="CellSignalStrengthSensor"/>
<sensor name="CellId" type="CellIdSensor"/>
```

Figure 3: Defining sensors in XML

4.3 Scripting

As a markup language, XML is quite good at creating structure for static information; however, it is less amenable for defining dynamic relationships. In our initial design, XML was used for both. The conditional logic for triggers, for example, was specified via element tags and attributes. For even the simplest logical expressions, however, this syntax became unreasonably complicated. As a result, we incorporated a lightweight scripting language into our design.

In MyExperience, scripting is meant to serve two purposes: 1) it provides a flexible interface for gluing components together from within the XML file and 2) it provides a method for injecting dynamic behaviors into MyExperience on the fly. Although a wide variety of scriptable features exist in our system, the most common use is in specifying trigger logic (which typically requires only a few lines of code).

The scripting design also allows for *in the field* modifications of program behavior. For example, new behaviors can be sent as scripts embedded in SMS or e-mail messages. This allows researchers to customize MyExperience behavior in the wild without having physical access to the participants' devices. Although we realize that this type of functionality could be used maliciously, we believe that it presents only a very limited security hazard. First, the attacker would have to know that the victim is a participant in a study and running MyExperience. Second, new scripts arriving from unknown addresses can be ignored (this is configurable). Third, the embedded scripts are constrained by the confines of MyExperience. Finally, although not currently implemented, the scripts themselves could be encrypted using public-key cryptography.

4.3.1 Example Scenario Continued

In the previous example scenario section, we defined three sensors in our XML file—one of them is of interest here: the PhoneCallSensor. Rather than survey a user randomly about their voice call quality, we will prompt the participant immediately following the completion of a call. This will allow us to correlate the automatically logged cell data with user response. To reduce annoyance, we sample the user only 20% of the time after completing a call (see Figure 4). Now that we have defined the appropriate sensors and triggers, the last step is to define relevant actions. We do this in the next section.

```
<!--Create the phone call completed trigger-->
<trigger name="PhoneCallCompletedTrigger">
  <script>
    phoneCall = GetSensor("PhoneCall");
    rand = GetRandom();
    if(phoneCall.State=="Completed" and rand < 0.2)
      RunAction("PhoneQualitySurvey");
  </script>
</trigger>
```

Figure 4: Defining a trigger in XML

4.4 The Survey Action

A survey action displays one or more self-report questions to the screen as defined in an XML file. There are two distinct types of questions: open form (*e.g.,* textbox, numeric textbox, etc.) and closed form (*e.g.,* checkbox list, combobox, radiobutton list, etc.). MyExperience also supports video, photo and audio capture. Because question flow can branch based on answers to the current question, the user must answer the current question before moving on to the next one.

To support dynamic user interface customization and question branching, an optional prescript and postscript can be associated with each question. Prescripts are executed the moment before a question is asked while postscripts are executed the moment after a question is answered. Prescripts can be used, for example, to parameterize question text with dynamic information (such as the user's current location). Postscripts allow for sophisticated question flow. For example, the next question can be scripted such that it depends on current sensor states, global properties, and/or previous response histories.

The look and feel of questions can be modified in XML to increase the accessibility of questions (*e.g.,* by setting color contrast, font size, etc.). Question text can be played audibly when presented to the screen (via prerecorded audio files). Closed form questions can be answered with two key presses. For open form questions, we've taken steps to mitigate the input burden by exposing the T9 predictive text entry, building auto-completion widgets, and including audio recording options.

4.4.1 Example Scenario Completed

To complete our example scenario, we must define a survey action and its associated entry question. Questions are specified independently of the survey action; this allows for question reuse across multiple surveys. Here, the phone quality survey contains only one question (see Figure 5), which asks the user to rate the voice quality of the call he just completed (see Figure 6 for screenshot). If the user does not respond within 30 seconds, as specified using the TimeOutInterval property, the survey automatically disappears.

```
<!--A survey action to ask about call quality-->
<action name="PhoneQualitySurvey" type="Survey">
  <prop name="TimeOutInterval">00:30</prop>
  <prop name="EntryQuestionId">CallQuality</prop>
</action>

<!--Define the call quality question-->
<question id="CallQuality" text="Please rate
  the voice quality of that phone call.">
  <prop name="ImageFile">cellnetwork.png</prop>
  <response widget="RadioButtonList">
    <option>Bad</option>
    <option>Poor</option>
    <option>Fair</option>
    <option>Good</option>
    <option>Excellent</option>
  </response>
</question>
```

Figure 5: Defining a survey action and questions in XML

A more sophisticated survey could ask follow-up questions based on user responses. For example, if the user responds "bad" or "poor," a follow-up question may ask if the call was bad or poor because of echo, delay, voice drop out, etc. The participant responses in this example could be correlated to the automatically logged cell sensor data to investigate, for example, the relationship between cell signal strength, specific geographic areas, and perceived voice quality.

Note that the branching logic works for both closed- and open-form questions and can be specified in C#, XML or the scripting interface. In addition, questions can be dynamically branched to during a survey based on current or past question responses and sensor data.

5. EVALUATION

We first characterize the performance of MyExperience running on commodity mobile devices. Although the current implementation has not been tuned for performance, we aim to quantify the impact of running MyExperience continuously on participants' primary phones. We then present several case studies describing our experiences using MyExperience in the field.

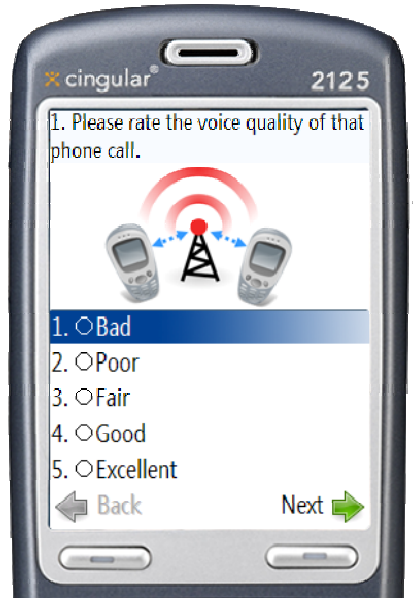

Figure 6: A screenshot of the phone quality survey

5.1 Performance Analysis

To understand the impact of MyExperience, we conducted experiments to measure its effect on CPU, memory, storage, and battery life on commodity mobile devices. The two types of devices we used were the HTC Tornado SmartPhone and HTC Universal Pocket PC Phone, both of which have been available since 2005. The Tornado has been branded as the T-Mobile SDA and Cingular 2125 in North America and has a 200MHz TI OMAP 850 processor (see Figure 7). The Universal has been branded as the i-Mate JasJar, and has a 520Mhz Intel Xscale PXA270 processor. Both devices have 64MB of RAM and run the Windows Mobile 5 operating system.

We also ran two case studies to get feedback on the *perceptual impact* that MyExperience incurred on the phone. We installed MyExperience on four participants' personal mobile phones for two weeks and surveyed the participants to compare their perception of the phones' responsiveness and battery life. Two of the participants worked in our research lab; the other two were students. Because one of the four participants was new to Windows Mobile 5 phones, we only surveyed the other three participants, who had used the HTC Tornado for more than six months and were familiar with its performance. While the number of participants is small, this feedback provides evidence about the perceived impact on the user experience.

5.1.1 CPU Utilization

Because MyExperience is event-driven, there is no measurable increase in CPU utilization when no events are taking place. To measure how fast triggers can be evaluated and how fast actions can be triggered, we measured action throughput by varying the rate of a timer-based sensor changing states and triggering a null action (*i.e.*, a no-op).

Figure 7: (left) HTC Tornado branded as the T-Mobile SDA; (right) HTC Universal branded as the i-Mate JasJar

We measured the CPU utilization of MyExperience using *pps*[5], which lists all processes and associated CPU usage. For each action frequency, we conducted five one-minute trials and averaged the results. The throughput results are shown in Figure 8. We measured < 3% utilization at a rate of 0.05Hz, which is equivalent to 4320 actions per day. To put this in context, we expect that a typical study would have 20 or fewer user surveys per day because of the user attention required. The CPU utilization will obviously be affected by the number of loaded sensors, their state change frequency, and the types of actions that are launched.

To understand the perceptual impact on user experience, we surveyed users after they completed the study on how they rated the relative responsiveness of their phones with and without MyExperience on a scale of 1-7. All 3 participants gave a rating of 4, with 1 being "much slower", 4 being "the same", and 7 being "much faster".

5.1.2 Data Storage

The storage requirement is highly dependent on the types of events, sensors, and surveys that are being logged. For example, a raw GPS sensor can produce about 1KB of data each second depending on its configuration. A location sensor, on the other hand, may use GPS as input and only output place events, such as arriving at home and work, a few times a day.

We measured the storage used to log 146 types of events available on the Tornado SmartPhone, without any external sensors or periodic Bluetooth scanning. Across four participants over the course of a week, each participant recorded an average of 4027 events per day, 52% of which came from the sensor that monitored GSM signal strengths. These timestamped events were 24KB on disk after gzip compression. Therefore, a 1GB miniSD card could hold 42 days of data.

[5] http://www.xs4all.nl/~itsme/projects/xda/tools.html

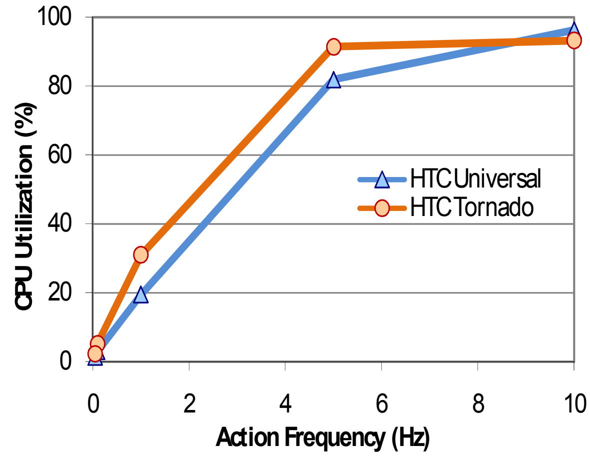

Figure 8: CPU Utilization vs. Action Frequency for two WM 2005 mobile devices, showing < 3% utilization at a rate of 4320 actions per day (0.05Hz).

Because of the high redundancy in sensor logs, we observed an 8.2x compression ratio using gzip at its normal compression level. More sophisticated log compression techniques, such as the 350x compression demonstrated in the Windows Flight Data Recorder [33], may further reduce the storage overhead.

5.1.3 Memory and Installation Footprint

MyExperience has a memory footprint of 5.6MB, which is less than 20% of the 28.7MB and 31.4MB available memory on the HTC Tornado and Universal, respectively. We were able to run MyExperience with more than 15 applications on both devices, including a web browser, a media player playing a video, a map browser, two games, and a video recording application.

In terms of installation size, the current build of MyExperience is 3.5MB and includes support for Bluetooth, GSM, 802.11 wireless, GPS as well as a Bluetooth-based accelerometer, light/infrared sensor, and barometer [22]. Because it is built on the .NET Compact Framework v2.0 and SQL Mobile 2005, additional 5.5MB and 2.3MB will be needed, respectively, if those are not already included in the ROM[6].

5.1.4 Battery Life

We measured the impact on battery life by recording the time it took for a fully charged HTC Tornado to be completely drained. To minimize the effects of other sources of power consumption, Bluetooth and WiFi were turned off for the battery experiments.

We first recorded the baseline battery life of the HTC Tornado to be 4 days and 17 hours. We then configured it to have MyExperience log all events as well as to trigger 20 surveys per day. Each survey prompt played an audio notification, turned on the backlight, and vibrated the phone for 5 seconds. We measured the battery life to be 4 days and 3 hours, which is a 12% decrease compared to the baseline. MyExperience's overall impact on battery life corresponds to the type and number of loaded sensors,

[6] Windows Mobile 6 devices ship with both .NET CF v2.0 and SQL Mobile 2005 built into the ROM.

their state change frequency, and the type and frequency of action executions. For example, a study which relies heavily on WiFi-based sensors and executes frequent database synchronization actions will have a more depreciative impact than 12%. Researchers should test battery life and device responsiveness for their particular study protocols before deploying to participants.

To understand the perceptual impact on user experience, we surveyed participants after they completed the study on how they rated the relative, perceived battery life on a scale of 1-7. All 3 participants gave a rating of 4, with 1 being "much shorter", 4 being "the same", and 7 being "much longer". Thus, participants did not perceive a major difference in battery life with MyExperience running on their phones.

5.2 Case Studies

We now present three field deployments to illustrate how combining *in situ* tracing with user feedback helps researchers gain deeper and more complete insight into mobile computing activities—understanding not just *what* the users did, but also *why*. These deployments range from 4-16 participants and 1-4 weeks, and cover: 1) battery life and charging behavior, 2) SMS usage and mobility, and 3) "Voting with your feet," a study on place visit pattern and personal preference. The results from the first two cases are preliminary and are presented for illustrative purposes rather than statistically sound findings.

5.2.1 Battery Charging

Battery life continues to be a challenge in mobile computing research. Significant progress has been made in power efficiency and battery capacity so that current mobile devices such as smart phones have a typical battery life of several days before recharging is necessary. However, activities such as web browsing over WiFi or watching videos can drain the battery in a matter of hours.

Because battery life is dependent on usage and charging behavior, it is important to capture statistics on application usage, battery charging, and remaining battery to help researchers understand and model battery life. In addition, because people adapt their charging behavior, such as when they notice a low-battery indicator, it is also important that we understand the motivation behind charging and the burden perceived by users. Understanding how battery design tradeoffs affect users can significantly improve future mobile device designs, such as creating smaller batteries that still meet users' usage requirements.

We conducted a study using MyExperience to monitor and log events relevant to battery life, such as power line status, battery life percentage, active applications, WiFi status, and phone calls. In addition, we surveyed the participants about the motivation behind charging, their perceived device usage, and charging method (*e.g.,* wall charger, car charger, and USB charger). By targeting the self-report survey to the exact moments of charging, we improved the accuracy of the user responses and eliminated irrelevant and redundant surveys.

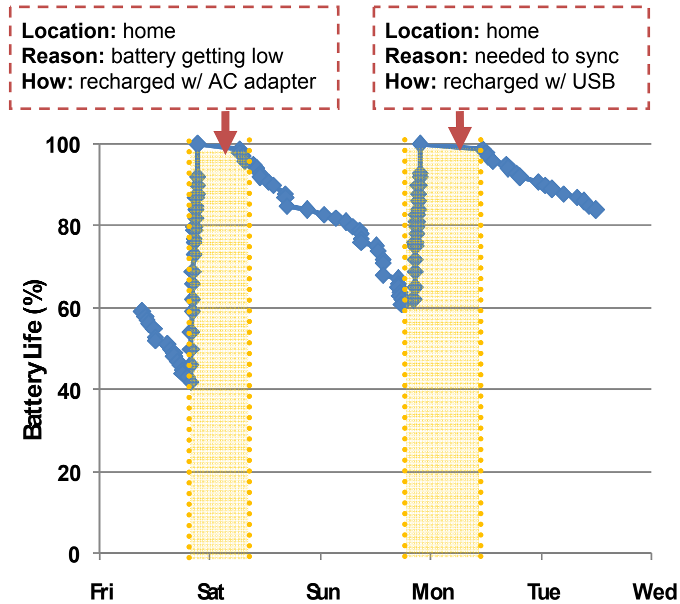

Figure 9: Battery life over a 4-day period for one of the participants, showing two recharging sessions. The survey results showed the location, reason for charging, and the type of charger used for each of these sessions.

We deployed the study on four participants' phones for two weeks. We revised the study in two ways based on early feedback: 1) displaying the time of the last charging session in the survey questions, and 2) surveying the participants when they have the device in hand—that is, surveying the participants when they *un*plug the device rather than when they plug it in.

Similar to the findings in the Llama project [32], our results suggest that people charge their phones when significant battery life remains. Participants recharged their phones every 1.3 days on average when an average of 68% battery life still remained.

Beyond understanding the remaining battery life over time, the *in situ* user feedback data significantly increases our understanding of the charging behavior. For example, the three primary reasons for recharging were: "Needed to synchronize phone" (48%), "Battery was getting low" (29%), and "Habit" (19%). Rather than simply assuming that users' behaviors are static and are not affected by increased power consumption due to background tasks (*e.g.,* pre-fetching), the survey results suggested that the user experience may be impacted because they would notice a low battery indicator more frequently. A longitudinal study with more participants would be necessary to provide more general arguments about battery charging behavior.

Figure 9 shows how automatically logged battery sensor readings can be combined with *in situ* survey results. It visualizes the automatically recorded battery level for one of the participants over a period of 4 days. The participant charged the phone twice in this period. Both charging sessions were at home, but for different reasons. For the first session, the participant charged the phone using an AC adapter because the battery was running low. For the second session, the participant needed to synchronize the phone with her computer, and the phone battery was recharged over USB in the process.

Figure 10 illustrates how the recharging sensor is declared in the MyExperience framework with a polling interval of 10 seconds. The figure also shows how we trigger a survey at the end of a charging session, and how we use the persistence API to store the time of the previous charging session.

The total number of lines of XML for the battery study was 89 lines, which included 6 lines of script to define the logic to trigger a survey, 14 lines to declare the sensors and actions, and 69 lines to define the 4 survey questions and response options. It took less than 30 minutes for a developer familiar with MyExperience to implement and test the study.

```
<sensor name="ChargingSensor" type="PowerLineSensor">
  <prop name="PollInterval">10000</prop>
</sensor>

<trigger name="StoppedChargingTrigger" type="Trigger">
  <script>
    chargeSensor = GetSensor("ChargingSensor");
    // check for online -> offline state transition
    if (chargeSensor.Transition("Online","Offline"){
       RunAction("SurveyNotify","BatterySurvey");
       // save timestamp of charging session
       SetProperty("StoppedCharging", GetTime());
    }
  </script>
</trigger>
```

Figure 10: Defining the sensor and trigger used in the battery charging case study

5.2.2 SMS Usage and Mobility

SMS is one of the most popular mobile communication mediums, with an estimated one trillion messages sent worldwide in 2005[7]. Given how difficult text entry is on mobile devices, SMS's explosive growth and popularity begs many interesting research questions. For example, when and why do people choose SMS over voice calls? Where do people use SMS? How often do people use SMS when they are in motion?

The mobile nature of SMS has made it challenging for researchers to study its usage. Previous studies have used techniques such as infrastructure-side monitoring which only captured send/receive frequency and time [34], interviews which only captured qualitative recollection of aggregate usage [2], [13], [31], or manual diaries in paper logs [12], providing interesting but narrow insight into SMS usage. By combining automatic sensor logging with qualitative user feedback, MyExperience enables researchers to gain a more complete understanding of the contexts in which SMS is used.

Using MyExperience, we were able to easily combine SMS and GSM-based sensor logging with SMS-triggered surveys. The self-report surveys are triggered as each SMS is sent and are used to capture user intention and motivation—information that cannot be captured through sensors. Three questions were asked in total about: 1) the participant's location (*e.g.,* home, school, bus, etc.), 2) message category (*e.g.,* reminder, status, coordination, etc.), and 3) reasons for using SMS over other communication means (*e.g.,* convenience, couldn't use voice, cost, etc.).

We deployed MyExperience on four participants' phones for two weeks, and collected a total of 71 SMS-related surveys. We correlated the mobility modes of the participants with SMS usage and found that 27% of SMS messages were sent while the participants were mobile.

The top three reasons participants used SMS over other types of communication were "Responding to a previously received SMS" (32%), "Convenience/Faster" (28%), and "Couldn't use voice" (14%). A longitudinal study with more participants would be necessary to provide statistical data on SMS usage behavior.

Figure 11 shows the script that triggers surveys after SMS are sent. The total number of lines of XML for the SMS study was 73 lines, which included 4 lines of script to define the SMS trigger, 21 lines to declare the SMS and GSM sensors and survey actions, and 48 lines to define the 3 survey questions and response options. The entire study was implemented and tested in less than 30 minutes by an expert user.

```
<trigger name="SmsSentTrigger" type="Trigger">
  <script>
    smsSensor = GetSensor("SmsSentSensor");
    if(smsSensor.PreviousState lt smsSensor.State)
      Execute("SmsSurveyNotification", "SmsSurvey");
  </script>
</trigger>
```

Figure 11: Trigger script for SMS case study

Figure 12 shows an example of how automatically logged GSM signal strengths can be combined with *in situ* survey results to help researchers better understand usage contexts.

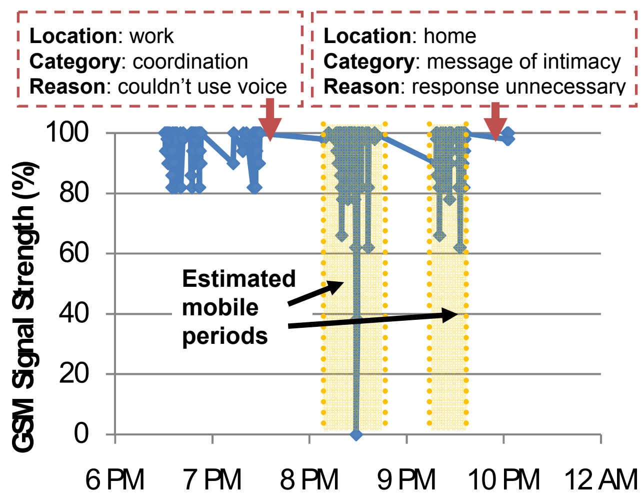

Figure 12: GSM signal strength over a 4-hour period for a single participant, showing the estimated mobile periods. The participant sent 2 SMS messages from two different locations for different reasons.

5.2.3 Voting With Your Feet

With location capability such as GPS, GSM [4], and WiFi [21] becoming more pervasive on mobile devices, we were interested in exploring the feasibility of building a mobile recommender system based on personal location history. We used MyExperience to investigate the relationship between explicit place ratings and implicit aspects of travel behavior such as visit frequency and travel time.

[7] http://www.gsmworld.com/services/messaging.shtml

Over the course of a four-week study with 16 participants, MyExperience logged up to seven GSM cells and associated signal strengths every second and surveyed participants up to 11 times a day about their perception of the current place and their social situation at the moment. Because there was no practical location technology precise enough to give a true label of a place, we effectively simulated such a location sensor by surveying the participants to get place labels.

To increase the number of places sampled, we used a GSM-based mobility detection algorithm to target the surveys to mobility transitions. We also used time-triggered surveys in case mobility was not correctly detected by our sensors. Overall, we collected 3,458 *in situ* surveys on 1,981 place visits. These survey results were automatically uploaded daily over GPRS. On average, 7MB of compressed GSM signal strengths were recorded per participant per day, for a total of 3.1GB for the entire study.

Results showed that for certain place types (*e.g.,* bars) there is a weak, positive correlation between place visit frequency and preference, as well as between travel time and preference. The results also showed that sensor-triggered, targeted sampling increased the relevancy and completion rate of surveys. Details of the study and our findings are presented in [10].

6. DISCUSSION

In this section, we present a discussion of issues faced when designing for "real use" as well as preliminary feedback from five researchers who are currently using MyExperience in their studies. Finally, we discuss some system level limitations of MyExperience.

6.1 Designing for Real Use

One challenge in designing a data collection platform for deployment on user's personal mobile devices is its potential to negatively impact the user experience. This is especially challenging on mobile phones because users perceive them as a much more stable platform than desktop and notebook computers (*e.g.*, most people are not used to rebooting their phones because of a system crash). Users also rely on mobile phones for critical communication functions, such as calling family members to coordinate pickup times or staying reachable in case of emergency. Interfering with these primary functions is at best socially disruptive and at worst legally liable—all of which may affect a participant's future use of the device.

In our 18 months of running development versions of MyExperience on our personal phones, we have observed several cases where the user experience was significantly impacted due to system-level issues. For example, using an early version of MyExperience, one of the co-author's phones crashed during an incoming call, leaving the ringer continuously on. The battery had to be taken out to reset the device. In another example, running an earlier, non-event-driven prototype version of MyExperience, the CPU overhead lead to delays in key presses and stuttering ring tones. The delay would intermittently make it difficult to write SMS messages or select a contact to call in the address book.

Even though the current version of MyExperience is robust with little performance overhead, there are several important user interaction issues that we learned over time. For example, early versions of MyExperience would trigger surveys and disrupt phone calls that were in progress. As another example, a participant in our SMS case study discovered that the survey notification would play an audible alert even when she set her phone profile to "silent." This was of particular issue because she used text messaging to inconspicuously communicate with her friends during class. We have since ensured that each of our actions default to abiding by the current phone profile as well as not interrupting phone calls. Furthermore, we provide a set of sensors to detect whether or not the user may be in an interruptible state (*e.g.,* device is idle, a voice call is not active, the current calendar appointment is free), enabling researchers to configure MyExperience appropriately for their studies.

6.2 Participation

One known drawback in using *in situ* surveys during field studies is that participant compliance tends to decrease over time. MyExperience addresses this problem, in part, by allowing the researcher to target surveys towards specific moments of interest thus lowering the burden of participation by reducing the number of irrelevant prompts. In addition, as previously highlighted, MyExperience incorporates sensed context into its sampling strategy to avoid interrupting the participant during inopportune periods (*e.g.,* while on the phone or in a meeting). Still, context-sensing is limited—not all inconvenient moments can be inferred. Researchers must carefully balance their sampling strategy with participant fatigue taking into account factors like study length, maximum prompts per day, maximum prompt frequency, survey length, and even the participants' jobs[8].

Not every survey will be responded to. MyExperience survey notifications can be dismissed manually or automatically after a preset amount of time. These interactions are logged to the database so that researchers may investigate completion rates and correlate periods of inactivity with sensor data (*e.g.,* participant X had a low survey completion rate but appeared to be driving often during survey prompts).

Both human and technical factors contribute to completion rates—participants may forget their phones, sensors may fail, etc. In some of these cases, MyExperience can be used to actively troubleshoot issues with the participant *in the field*. For example, in one study currently being piloted at our lab, participants must wear a wireless sensor platform on their waistbands in addition to carrying their mobile phones. If the mobile phone becomes disconnected from the sensor platform, MyExperience alerts the participant and triggers a set of troubleshooter dialogs. If the problem persists, an SMS message is sent to the research team.

To reduce irritability, we are exploring methods to incorporate machine learning techniques to dynamically change sampling strategies per participant based on sensor data and response history. Our goal is to decrease the number of required surveys over time as the system learns more information about the participant.

6.3 Preliminary Researcher Feedback

MyExperience is currently being used to support four research projects ranging from studying how an individual's mobile phone may be used to encourage physical activity to exploring heart rate

[8] Certain professions are not amenable to any interruption (*e.g.,* a trial judge, an emergency room doctor, etc.). Prior field studies conducted by our lab have screened participants based on their career and job descriptions.

variability and psychological stressors using wireless ECG measurements. Two of the research projects have entered the pilot phase (one of which has undergone extensive piloting for months); the other two are in development. All projects use the MyExperience XML and scripting interface extensively, particularly for managing triggers and *in situ* surveys. Two of the projects invoke MyExperience as a library from their own applications.

To gather preliminary feedback on the design and use of MyExperience, an informal survey was sent to members of each of the four research projects. We had five respondents, all of whom were directly involved in configuring MyExperience for use in their studies. This survey is not meant to provide a formal evaluation of our tool but rather to illustrate the benefits and challenges of incorporating MyExperience into real studies, as perceived by the researchers themselves.

Four of the five researchers were experienced programmers; however, none had significant mobile phone development background. In addition, each researcher indicated that s/he had limited experience with C# and XML. However, all felt comfortable with the concept of a markup language; most had manually written HTML in the past. When asked about MyExperience's XML schema, all five researchers seemed to agree that it was "intuitive" and "straightforward." One researcher felt that her programming background made it easier to understand the branching logic she created for her survey.

All five researchers indicated that the scripting interface was a powerful feature. However, we received mixed feedback on what threshold of technical proficiency is necessary to write and comprehend the scripts. The researcher with the least amount of programming experience found it easy to use, "I think it just took me a few minutes to pick it up enough to write my XML file." Others pointed to the importance of providing examples[9] to support comprehension and better script debugging tools.

Finally, when asked about the most beneficial parts of the tool, researchers mentioned the following:

- The ability to "trigger anything in response to such a wide range of events or combination of events." This was brought up by four of the five researchers.
- "An easy way for a semi-technical designer to set up user-experience studies for cell phone applications."
- MyExperience can be used "to collect a rich set of data." Three of the five researchers mentioned this.
- "The XML structure is excellent and is deeply expandable through C# extensions to MyExperience"

6.4 System Limitations

The current event model is asynchronous and unidirectional: sensors → triggers → actions. Triggers that execute multiple actions cannot conditionally execute a second action based on the result of a previous action. Also, although the action scheduler serializes actions within a given trigger context, actions from different triggers may be interleaved because of their configurable priority and start time.

In addition, as scripts are interpreted, they are less efficient than their native counterparts in C/C#. Although we use a subscription model to reduce unnecessary trigger evaluations, there is still a limit to how fast the state events can be evaluated. One approach to using high rate sensors, such as accelerometers, is to wrap another sensor around its output. This new sensor can process the high rate data in native code, and generate higher level state changes at a lower rate. With this approach, low-level sensor state changes can still be logged to the database without the cost of invoking triggers. A related issue concerns the difference between discrete and continuous states for sensors. Given that our current sensor model uses a state-based abstraction, the same approach can be used to wrap continuous sensors and discretize their output.

7. RELATED WORK

Automatic logging can record usage on the infrastructure side [34], [35] or directly on devices through instrumentation [8], [9], [27], [28]. The infrastructure approach scales to a large number of users; however, it is limited to only observing network service usage, such as phone calls, SMS, wireless access, and cannot capture sensor data and device usage, such as playing music and failed call attempts. Device-side approaches, such as ContextLogger [28], provide access to device usage and sensor readings, but still do not capture important information such as user intention and perception.

In situ self-report methods like ESM [7] and diaries [29] provide details about a user's context, intentions, and actions that system activity logs cannot capture [6]. These techniques, however, do not scale as well as logging because they require user attention. A range of computerized experience-sampling tools have been developed to elicit user response in the field [1], [3], [5], [16] but none provide an extensible framework for combining automatic logging and user sampling on a participant's personal device. The Context-Aware Experience Sampling (CAES) [16] tool incorporates sensor data to trigger self-report surveys at specific moments of interest. However, CAES was not designed to run on a participant's own personal device. Thus, it lacks the ability to avoid prompting during inopportune moments (*e.g.,* when the participant is on the phone or in a scheduled meeting) as well as the ability to interrupt ESM surveys during critical device usage tasks (*e.g.,* answering an incoming phone call). In addition, CAES does not offer the flexibility of specifying dynamic trigger conditions and generic actions. Momento [3] takes a different approach; it is built around a thin client/heavy server architecture with SMS/MMS as its communication medium. This approach expands the range of supported devices but at a cost of not being able to automatically capture context and usage data.

One common approach in combining logging and qualitative feedback is through the use of interviews. Logs can be used to cue a participant's memory during interviews, thereby reducing recall biases [27], [36]. However, interviews do not scale well across large numbers of participants. In addition, participants may still suffer from some form of recall bias or memory lapse even with cueing.

SenseCam [11] offers an entirely different approach in collecting qualitative and quantitative data; digital photographs are automatically captured and annotated with sensor data via a pendant worn around the neck. The photographs allow for qualitative

[9] No formal documentation was available to the research teams when they began using MyExperience. Documentation including examples has now been produced and exists online at http://myexperience.sourceforge.net

assessments of ground truth (*e.g.,* user appears to be indoors) and provide good cue points for interviews; however, they do not collect user feedback *in situ* and the continuous photography raises privacy concerns. Moreover, participants are required to wear an additional device. MyExperience is designed to run on devices that people already carry (*e.g.,* a mobile phone).

Most relevant to our work, in a workshop on combining logging and qualitative methods, Mankoff et al. [23] propose but do not implement a tool that combines logging with *in situ* capture. More recently, the SocioXensor project [15] shares our approach in logging sensor data with subjective data on mobile phones; however, its focus is on gaining insights into social phenomena. In addition, their system has not yet been deployed so it is unclear how they collect qualitative and quantitative data and whether it is designed for use on user's personal devices. We believe MyExperience could also be used for studies benefiting the social and behavioral sciences, though more work is needed to understand its threshold of use for semi-technical researchers.

8. CONCLUSION AND FUTURE WORK

We have presented MyExperience, a system that combines automated logging of sensor data and targeted, *in situ* user experience sampling to collect real usage data on mobile phones. MyExperience supports sensing of more than 140 types of events including communication (*e.g.,* phone calls, SMS), device usage (*e.g.,* key presses), user context (*e.g.,* calendar appointments), location (*e.g.,* GPS), and environmental sensors (*e.g.,* microphone, Bluetooth-enabled barometer).

In particular, we have made two contributions: 1) a non-proprietary system that collects real *in situ* qualitative and quantitative usage data on people's personal mobile phones, and 2) lowered the barrier for researchers to collect *in situ* usage data by providing a rich set of extensible sensors and actions with a lightweight XML-based configuration.

MyExperience runs on current commodity mobile phones and Pocket PCs with minimal impact to both measured and perceived responsiveness and battery life, and is well-suited for deploying on people's personal devices to collect realistic usage data. It supports disconnected operations and secure, opportunistic synchronization to improve data durability and provide early access to study data.

We have presented several field deployments to illustrate how combining objective traces and subjective feedback provides insight into the what, when, how, and why in certain mobile technology behaviors (*e.g.,* battery charging and SMS usage). Preliminary feedback from researchers that have used MyExperience suggest that its XML and scripting-based configuration has a low learning threshold and its extensibility provides a high ceiling of functionality.

We are currently exploring novel, scalable techniques to help researchers visualize multiple event streams of rich qualitative and quantitative datasets for analysis and a front-end tool to further reduce the learning threshold of the XML. We are also exploring adaptive sampling techniques to reduce data rate and user burden while maintaining data fidelity.

The MyExperience tool is open source software (under the BSD license). Please see http://myexperience.sourceforge.net/ for the source code, binary release and additional documentation.

9. ACKNOWLEDGMENTS

We thank Adrienne Andrew, Bill Deleeuw, Cassandra Hearn, Pedja Klasnja, Anthony LaMarca, Ryan Libby, Fred Potter, Ian Smith, Tammy Toscos, and David Wetherall for their input on design goal system requirements and running development versions of the software. We also thank Kate Everitt, Ben Greenstein, Ramki Gummadi, Michael Toomim, Scott Saponas, and the MobiSys anonymous reviewers for comments and feedback on the paper.

10. REFERENCES

[1] Barrett, L.F. and Barrett, D.J. An Introduction to Computerized Experience Sampling. *Psychology. Social Science Computer Review., V. 19, No. 2*, S01, pp. 175-185.

[2] Barkhuus, L. Why everyone loves to text message: social management with SMS. (Poster). *In Proceedings of GROUP '05* (Sanibel Island, FL, USA, Nov., 2005)..

[3] Carter, S., J. Mankoff, and J. Heer. Momento: Support for Situated Ubicomp Experimentation. *In Proceedings of the ACM Conference on Human Factors in Computing Systems* (CHI 2007), To appear. 2007.

[4] Chen, M., Sohn, T., Chmelev, D., et al. Practical Metropolitan-Scale Positioning for GSM Phones. *In Proc of Ubicomp 2006*, Orange County, California, 2006.

[5] Consolvo, S. and Walker, M. Using the Experience Sampling Method to Evaluate Ubicomp Applications. *IEEE Pervasive Computing Mobile and Ubiquitous Systems: The Human Experience, Vol. 2, No. 2* (Apr-Jun 2003), pp. 24-31.

[6] Consolvo, S., Harrison, B., Smith, I., Chen, M., Everitt, K., Froehlich, J., and Landay, J. Conducting *In situ* Evaluations for and with Ubiquitous Computing Technologies. *Intl Journal of HCI, 22(1)*, 107-122, 2007 (to appear)

[7] Csikszentmihalyhi, M. and Larson, R. Validity and Reliability of the Experience-Sampling Method. *Journal of Nervous and Mental Disease*, 1987, 175:526-536.

[8] Demumieux, R. and Losquin, P. Gather customer's real usage on mobile phones. *Proc. Intl. Conf. Human Computer Interaction with Mobile Devices & Services* (MobileHCI 2005), pp. 267–270.

[9] Eagle, N. and Pentland, A. Social Serendipity: Mobilizing Social Software. *IEEE Pervasive Computing, vol. 04, no. 2*, pp. 28-34, April-June, 2005.

[10] Froehlich, J., Chen, M., Smith, I., and Potter, F. Voting With Your Feet: An Investigative Study of the Relationship Between Place Visit Behavior and Preference. *Proceedings of Ubicomp 2006*, Orange County, California, 2006.

[11] Gemmell, J., Williams, L., Wood, K., Lueder, R., and Bell, G. Passive capture and ensuing issues for a personal lifetime store. *In Proc of the 1st ACM Workshop on Continuous Archival and Retrieval of Personal Experiences* (New York, New York, USA, October 15 - 15, 2004).

[12] Grinter, R. E.; Eldridge, M. Y do tngrs luv 2 txt msg? *In Proc ECSCW 2001*, September 16-20; Bonn, Germany. Kluwer; 2001; 219-238.

[13] Häkkilä, J. and Chatfield, C. 2005. 'It's like if you opened someone else's letter': user perceived privacy and social practices with SMS communication. *In Proc. of the 7th*

Conference on HCI w/Mobile Devices & Services (Salzburg, Austria, September, 2005). MobileHCI '05.

[14] Harle, R. K. and Hopper, A. 2005. Deploying and evaluating a location-aware system. *In Proc. of the 3rd international Conference on Mobile Systems, Applications, and Services* (Seattle, WA, June, 2005). Mobisys 2005.

[15] ter Hofte, Otte, R., Peddemors, A., Mulder, I., What's Your Lab Doing in My Pocket? Supporting Mobile Field Studies with SocioXensor. *(Demonstration) CSCW2006 Conference Supplement*, Nov. 4-8, 2006, Banff, Alberta, Canada.

[16] Intille, S. S., Rondoni, J., Kukla, C., Ancona, I., and Bao, L. A context-aware experience sampling tool. *In CHI '03 Extended Abstracts* (Ft. Lauderdale, Florida, USA, April 05 - 10, 2003). CHI '03. ACM Press, New York, NY, 972-973.

[17] Isomursu, M., Kuutti, K., and Väinämö, S. Experience clip: method for user participation and evaluation of mobile concepts. *In Proc. of the 8th Conference on Participatory Design: Artful integration: interweaving Media, Materials and Practices*, Volume 1.

[18] Kellar, M., Reilly, D., Hawkey, K., Rodgers, M., MacKay, B., Dearman, D., Ha, V., MacInnes, W. J., Nunes, M., Parker, K., Whalen, T., and Inkpen, K. M. It's a jungle out there: practical considerations for evaluation in the city. *In CHI '05 Extended Abstracts* (Portland, OR, USA, April 02 - 07, 2005).

[19] Kjeldskov, J. and Graham, C. A review of Mobile HCI Research Methods. *In Proceedings of Mobile HCI 2003*, Springer-Verlag, LNCS 2795, 2003, 317-335.

[20] Kotz, D. and Essien, K. Analysis of a campus-wide wireless network. *In Proc. of the 8th Annual international Conference on Mobile Computing and Networking* (Atlanta, Georgia, USA, September, 2002). MobiCom '02.

[21] LaMarca A., Chawathe Y., Consolvo S., et al. Place Lab: Device Positioning Using Radio Beacons in the Wild. *In Proc Pervasive'05*, Munich, Germany.

[22] Lester, J., Choudhury, T., and Borriello, G. A Practical Approach to Recognizing Physical Activities. *In Proc of Pervasive 2006*, May. 7, 2006

[23] Mankoff, J. and Carter, S. Crossing Qualitative and Quantitative Evaluation in the Domain of Ubiquitous Computing. *Presented at the CHI 2005 Workshop on Usage Analysis: Combining logging and qualitative methods*. Portland, OR, April 2005.

[24] Myers, B., Hudson, S. E., and Pausch, R. Past, present, and future of user interface software tools. *ACM Trans. Computer.-Hum. Interact. 7, 1* (Mar. 2000), 3-28.

[25] Nicholson, A. J., Chawathe, Y., Chen, M. Y., Noble, B. D., and Wetherall, D. Improved access point selection. *In Proc. of the 4th international Conference of MobiSys 2006*. (Uppsala, Sweden, June 19 - 22, 2006).

[26] Palen, L. and Salzman, M. Voice-mail diary studies for naturalistic data capture under mobile conditions. *In Proceedings of the CSCW 2002* (New Orleans, Louisiana, USA, November 16 - 20, 2002).

[27] Patel, S.N., Kientz, J.A., Hayes, G.R., Bhat, S., and Abowd, G.D. Farther Than You May Think: An Empirical Investigation of the Proximity of Users to their Mobile Phones. *In Proceedings of Ubicomp 2006*, Orange County, California, 2006.

[28] Raento, M.; Oulasvirta, A.; Petit, R.; Toivonen, H., ContextPhone: a prototyping platform for context-aware mobile applications. *Pervasive Computing, IEEE, vol.4, no.2*, pp. 51- 59, Jan.-March 2005

[29] Rieman, J. The diary study: a workplace-oriented research tool to guide laboratory efforts. *In Proceedings of CHI '93.* (Amsterdam, The Netherlands, April 24 - 29, 1993).

[30] Salovaara, A., Oulasvirta, A., and Jacucci, G. "The panopticon": a method for observing inter-group interactions. *In CHI'06 Workshop on Reality Testing*

[31] Schusteritsch, R., Rao, S., and Rodden, K. Mobile search with text messages: designing the user experience for Google SMS. *In CHI '05 Extended Abstracts* (Portland, OR, USA, April, 2005). CHI '05.

[32] Tilkidjieva, D., Banerjee, N., Kazandjieva, M., Rollins, S., and Corner, M. LLAMA: An Adaptive Strategy for Performing Background Tasks on Mobile Devices. *(Poster/Demo) at HotMobile '06*, Semiahmoo, WA, 2006.

[33] Verbowski, C. et al. Flight Data Recorder: Always-on Tracing and Scalable Analysis of Persistent State Interactions to Improve Systems and Security Management. *In Proc. of OSDI 06*. Seattle, WA, 2006.

[34] Yoon, J., Noble, B. D., Liu, M., and Kim, M. Building realistic mobility models from coarse-grained traces. *In Proceedings of the 4th international Conference on Mobile Systems, Applications and Services* (Uppsala, Sweden, June 19 - 22, 2006). MobiSys 2006.

[35] Zerfos, P., Meng, X., Wong, S. H., Samanta, V., and Lu, S. A study of the short message service of a nationwide cellular network. *Proc. of 6th ACM SIGCOMM on Internet Measurement* (Rio de Janeriro, Brazil, Oct, 2006). IMC '06.

[36] Zhou, C., Ludford, P., Frankowski, D., and Terveen, L. An Experiment in Discovering Personally Meaningful Places from Location Data. *In Extended Abstracts CHI 2005.*