June 11th, 2007
MobiEval Workshop : System Evaluation for Mobile Platforms
Metrics, Methods, Tools, and Platforms

myexperience

Increasing the Breadth: Applying Sensors, Inference and Self-Report in Field Studies with the MyExperience Tool

Jon Froehlich[1]

Mike Chen[2], Sunny Consolvo[2], Beverly Harrison[2], and James Landay[1,2]

dub design: use: build:
university of washington[1]

intel
Intel Research, Seattle[2]

introduction

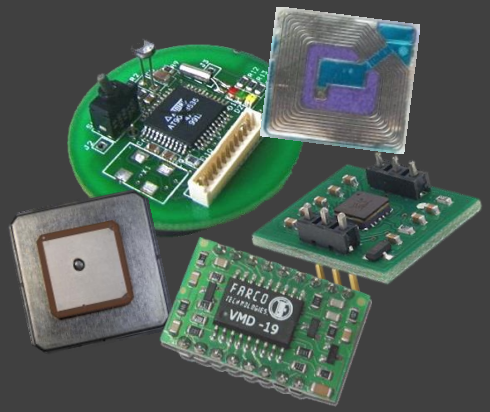
- Context-aware mobile computing has long held promise…
  - But building and evaluating context-aware mobile applications is hard
- Often encompasses a range of disciplines / skills
  - Sensor building and/or integration
  - User modeling
  - Statistical inference / machine learning
  - Designing / building application
  - Ecologically valid evaluation

motivating questions

- How can we easily *acquire labeled sensor datasets* in the field to inform our user models and train our machine learning algorithms?
- How can we *evaluate* the applications that use these user models / algorithms *in the field*?
- How can we *extend the evaluation* period from days to weeks to months?

the myexperience tool

sensors



+

user

=

context

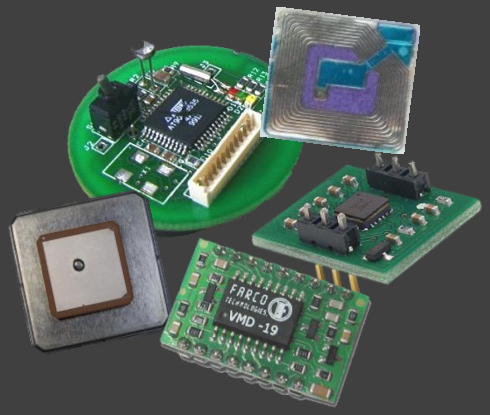
self- report

myexperience

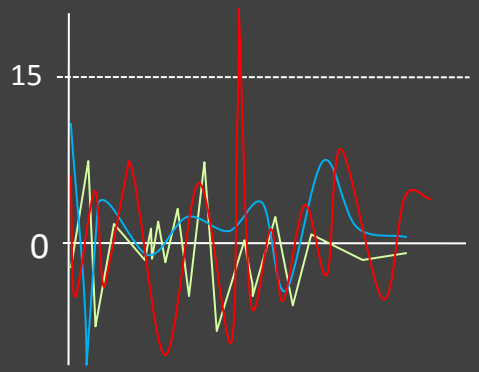
MyExperience combines automatic sensor data traces with contextualized self-report to assist in the *design* and *evaluation* of mobile technology

sensors, triggers, actions

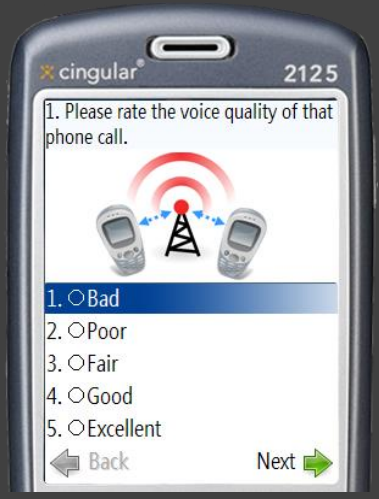
Sensors



Triggers



Actions



Example Sensors:

- DeviceIdleSensor
- SmsSensor
- PhoneCallSensor
- RawGpsSensor
- PlaceSensor

Example Triggers:

- DeviceIdle > 15 mins
- IncomingSms.From == “Mike”
- PhoneCall.Outgoing == true
- Calendar.IsBusy == false
- Gps.Longitude == “N141.23”

Example Actions:

- SurveyAction
- ScreenshotAction
- SoundPlayerAction
- VibrationAction
- SmsSendAction

Project #1

votewithyourfeet

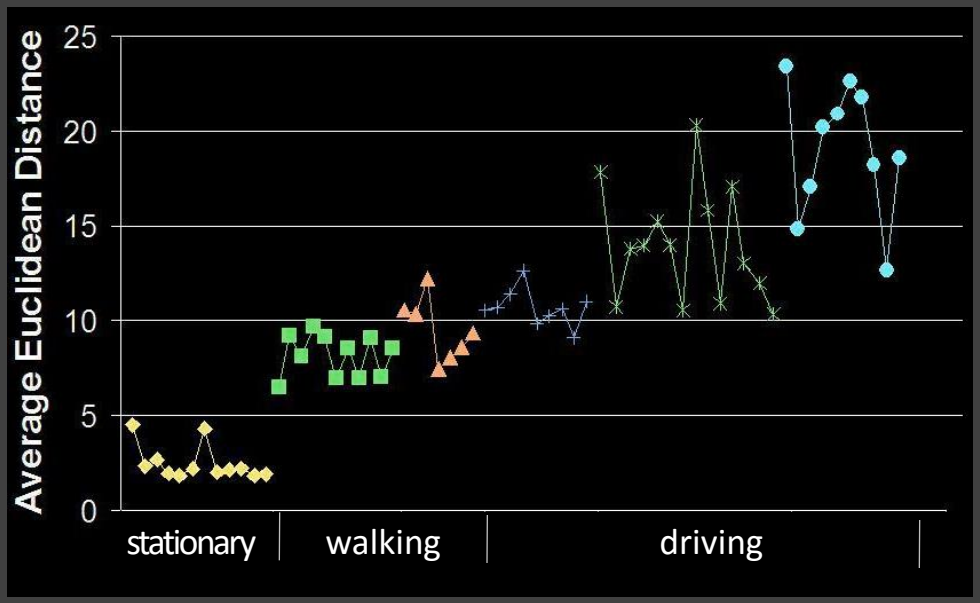
Jon Froehlich[2], Mike Chen[1], Ian Smith[1], Fred Potter[2]

Intel Research[1] and University of Washington[2]

project overview

- Formative study to determine relationship between movement and place preference
- Four week study
  - 16 participants
  - Up to 11 *in situ* self-report surveys per day
  - Carried Windows Mobile SMT5600 (provided) w/MyExperience in addition to their personal phone
  - Logged GSM sensor data @ 1Hz
- No external sensors required

two survey triggers



Mobility Sensor

- Similar to Sohn et. al UbiComp 2006
- GSM signal variation to detect movement
- No external sensors required
- Stationary for 10 minutes → trigger survey

Pseudo-Random Time Trigger

- No movement detected for 1 hr → trigger survey randomly within next hour
- Ensures consistent sampling

survey questions

lessons learned

- Near-real time access to study data is extremely beneficial
  - Web sync provides data redundancy
  - Allows early analysis of data
  - Can detect problems in the field as they occur
  - Data can be used as *cue points* during interview
- Additional mobile phone can be problematic
  - Forget device
  - Have to remember to charge
- Limit number of open-form self-report questions

lessons learned

- Need flexibility in configuring the sensors, triggers, and actions
  - Could already setup the user interface in XML
  - Expanded this to include sensors, triggers, and actions
- Current version uses XML + scripting combination to provide both declarative and procedural functionality

Project #2

ubifit

Using Technology to Encourage Physical Activity

Sunny Consolvo[1], Jon Froehlich[2], James Landay[1,2], Anthony LaMarca[1], Ryan Libby[1,2], Ian Smith[1], Tammy Toscos[3]

Intel Research[1], University of Washington[2], and Indiana University[3]

project overview

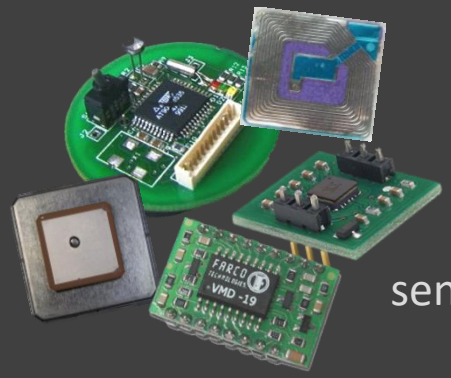
- Initial 3-week study planned followed by longitudinal 3-month study
  - Female participants from Seattle area
  - Participants use *lab-provided* WM5 devices with ubifit instead of their own personal phones
- UbiFit application
  - Built off of MyExperience
  - Collects both inferred activity and self-report activity data
  - Data is sync'd with Intel Research's web server once/hr throughout the study

mobile sensing platform

MSP Features

- Built on iMote2
- Linux OS
- 32MB RAM
- 2 GB Flash Storage
- Zigbee and Bluetooth
- 12-16 hours battery life

wearable msp



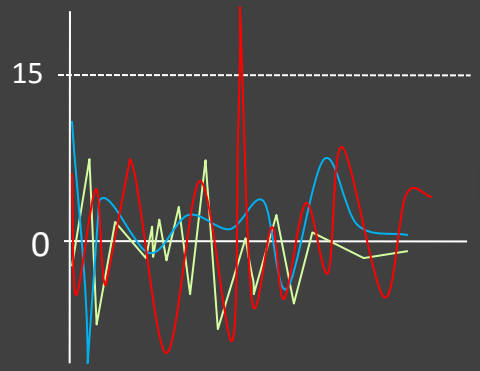
sensors

10 Built-in Sensors

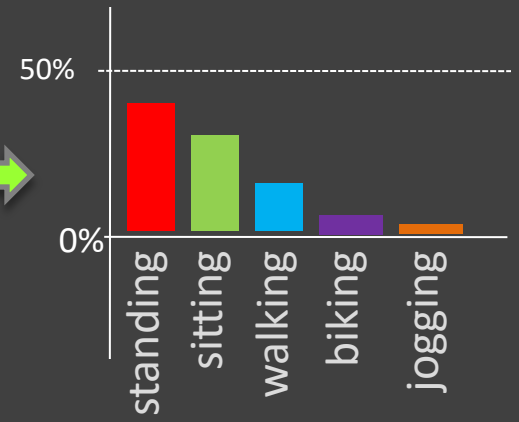
- 3D Accelerometer
- 2D Compass
- Barometer
- Humidity
- Visible light
- Infrared light
- Temperature
- UART, GPIO breakouts for additional sensors

msp + myexperience

Bluetooth



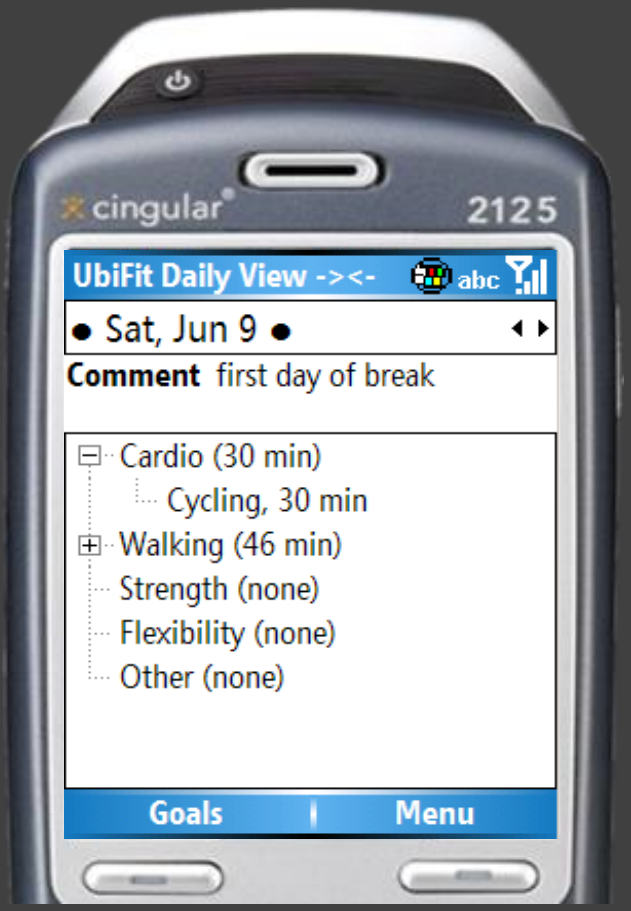
raw data



inference data

inferred ubifit activities

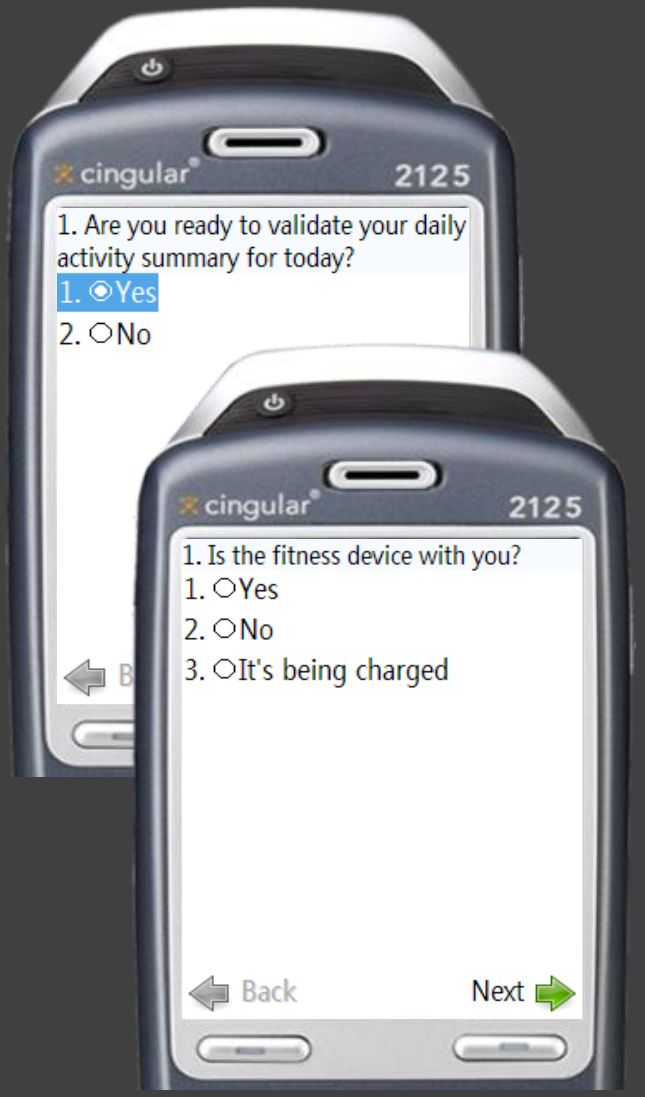
- Six activities are automatically detected
  - Bicycling
  - Elliptical trainer
  - Running
  - Sedentary
  - Stairmaster
  - Walking



manual activity entry

subset of ubifit triggers

- Journal reminder
  - If journal has not been used in ~2 days and it's past 8PM, launch journal reminder
- Uncertain activity occurred
  - If the system *knows* an activity occurred but couldn't determine the exact activity, a survey is launched
- MSP troubleshooter
  - If the MSP hasn't been seen in ~2 hrs and it's after 10AM, launch a troubleshooter

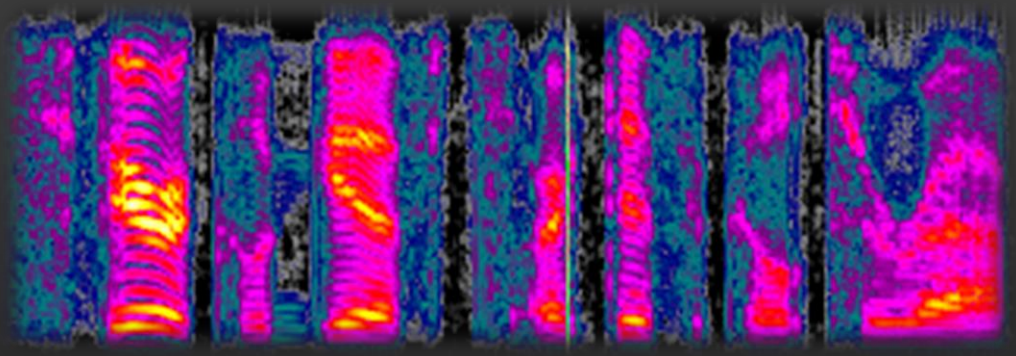


lessons learned

- MyExperience must *not* impede normal phone operation
  - Reduce interaction “lag”
  - Avoid interrupting phone calls
  - Follow phone profile (e.g., silent)
  - Respect battery life
- Sensor, trigger, action architecture
  - Can be used to actively troubleshoot prototype technology in the field

Project #3

methodology evaluation



Evaluating Context-Aware Experience Sampling Methodology

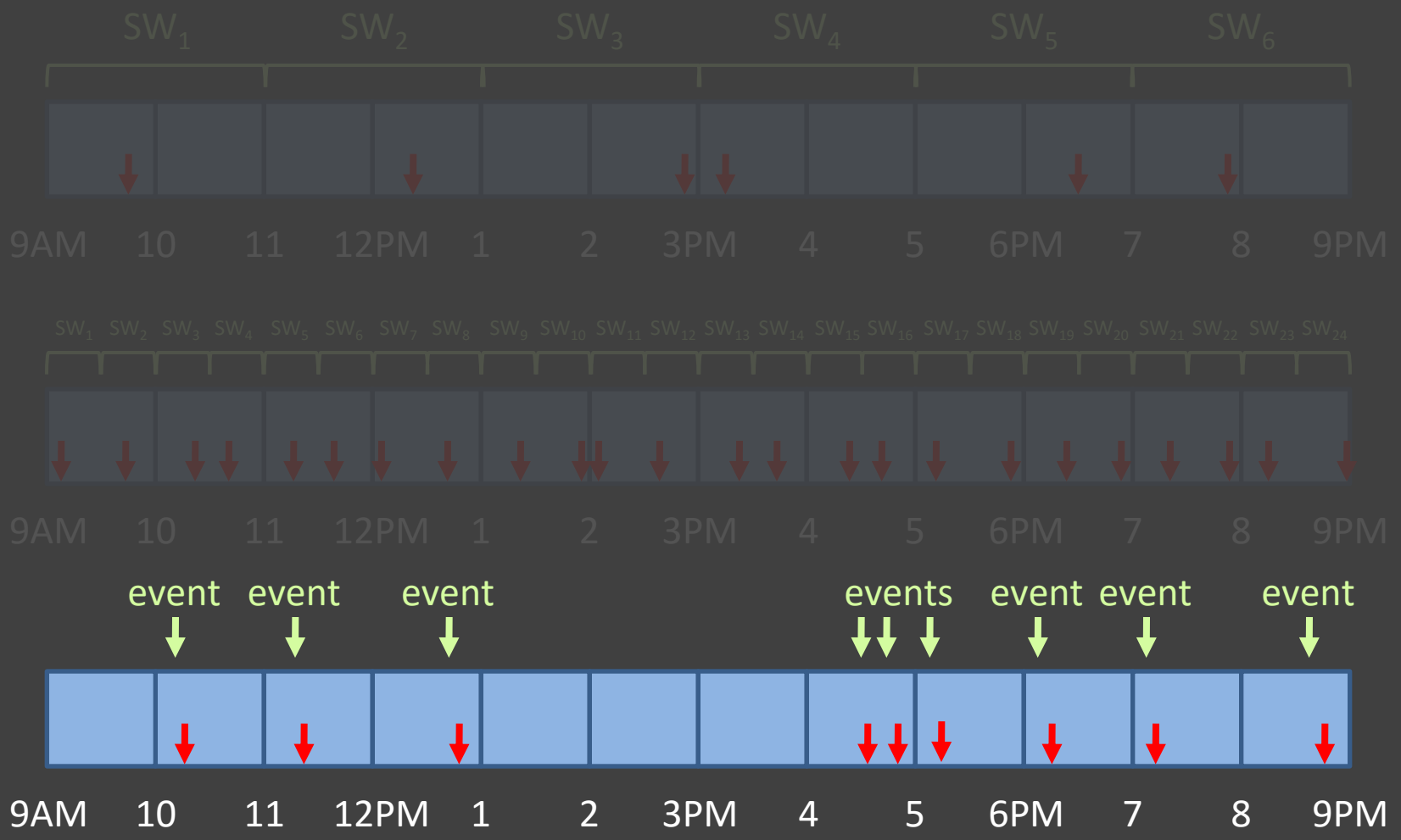
Beverly Harrison[1], Adrienne Andrew[2], and Scott Hudson[3]

Intel Research[1], University of Washington[2], and Carnegie Mellon University[3]

project overview

- Comparing traditional and context-aware experience sampling
  - Investigating the tradeoff between effort expounded and the data obtained
- Like UbiFit
  - Participants use their *own* phones with MyExperience
  - Relies on MSP for sensor data
- Unlike UbiFit
  - Uses MyExperience as a data collector

self-report sampling strategies

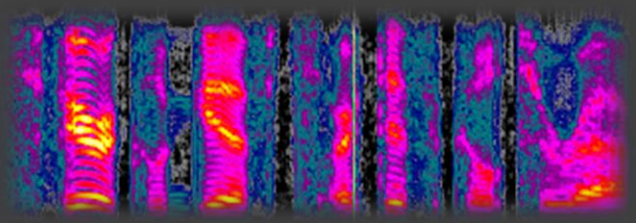


study: phase 1

- Pseudo-randomly sample participants throughout workday about self-reported interruptibility
  - Also, ask about effort and irritability
- Automatically capture sensor data
  - speech, calendar appointment information
- At the end of phase 1, look for correlations between sensor data and self-rated interruptibility
  - Also, analyzing response times

msp

+



speech feature extraction

+

device calendar

study: phase 2

- Continue study with same participants but change sampling strategy
- Pseudo-random sampling but use sensors to try and avoid times of “high interruptibility”
- Continue asking questions about interruptibility, effort and irritation
  - Expect these self-reported values to decrease and response rate to increase

lessons learned

- Windows Mobile SystemState API
  - Not always implemented
    - (e.g., Outlook calendar appointments)
- Generalized conditional deferral mechanism
  - Previously:
    Actions could queue when user on phone
  - Now:
    Actions can be queue based on *any* sensor state

beyond technology studies

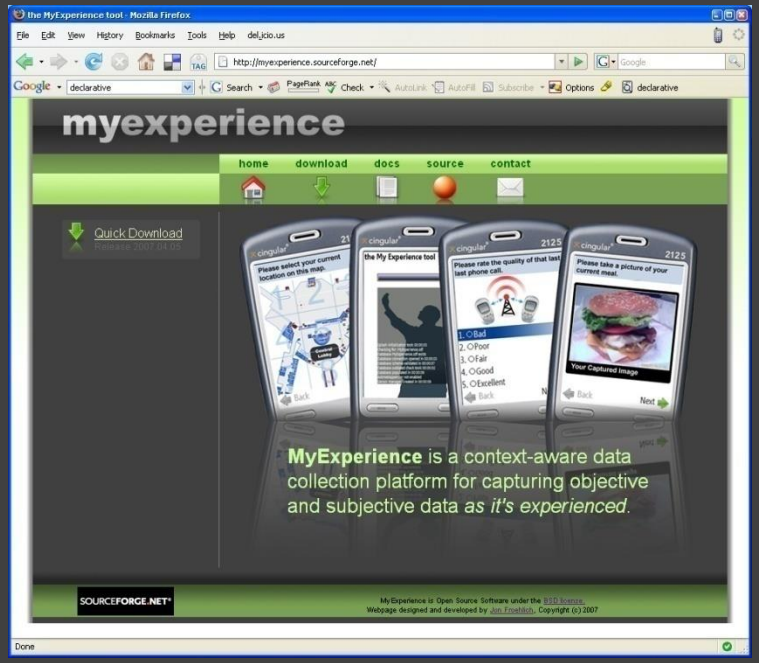
- Mobile therapy
  - Margie Morris, Bill Deleeuw, et al.
  - Digital Health Group, Intel
- Multiple sclerosis pain and fatigue study
  - Dagmar Amtmann, Mark Harniss, Kurt Johnson, et al.
  - Rehabilitative Medicine, University of Washington
- Smartphones for efficient healthcare delivery
  - Mahad Ibrahim, Ben Bellows, Melissa Ho, Sonesh Surana et al.
  - Various departments, University of California, Berkeley

conclusion

- MyExperience enables a range of research involving sensors and mobile computing
  - Automatic logging of sensor streams
  - *In Situ* sensor-triggered self-report surveys
- MyExperience can be used to
  - Acquire labeled data sets for machine learning
  - Evaluate field deployments of prototype apps
  - Gather data on device usage
  - Human behavior, healthcare, dev world studies

thankyou

source code available



My Talk Tomorrow

Tuesday, June 12th

Tools & Techniques (2nd talk)

"MyExperience: A System for In Situ Tracing and Capturing of User Feedback on Mobile Phones"

Acknowledgements

Intel Research, Seattle

http://www.sourceforge.net/projects/myexperience

Backup Slides

xml / scripting interface

- XML : Declarative
  - Define sensors, triggers, actions, and user interface
  - Set properties
  - Hook up events

```
<sensor name="Place" type="PlaceSensor">
  <p name="PollingInterval">00:00:01</p>
</sensor>
```

- Script : Procedural
  - Create fully dynamic behaviors between elements specified in XML
  - Interpreted in real time
  - New scripts can be loaded on the fly

```
<trigger name="Silent" type="Trigger">
  <script>
    placeSensor = GetSensor("Place");
    if(placeSensor.State = "Work"){
      SetProfile("Silent");
    }
  </script>
</sensor>
```

xml / scripting interface

Two Primary Benefits

- Lowers the barrier of use
  - Allows researchers unfamiliar with mobile phone programming to use MyExperience
- Straightforward means to specify self-report UI
  - Simply edit the XML file to change the interaction
  - Control flow logic from one question to the next
  - Specify response widgets

```
<trigger name="ConnectivityTrigger" type="Trigger">
  <script>

   curDate = GetCurrentDate().Date;
   curTimeOfDay = GetCurrentTime();
   curDateTime = curDate + curTimeOfDay;
   suppressUntil = GetProperty("suppressUntil");

   connectivitySensor = GetSensorSnapshot("ConnectivitySensor");

   if ( curTimeOfDay >= "10:00:00“ and
        connectivitySensor.TimeSince("true") gt "02:00:00" and
        (suppressUntil = null or curDateTime gt suppressUntil)){

          tomorrow = curDate.AddDay(1);
          SetProperty("suppressUntil", tomorrow);
          GetAction("FitnessDeviceQuery").Run();
   }

  </script>
</trigger>
```

```
<trigger name="RandomWithDeferralTrigger" type="Trigger">
  <script>

    randSensor = GetSensorSnapshot("RandomSensor");
    if( randSensor.State = true){

      timeOutInSeconds = 15 * 60;
      WaitUntil("return GetSensorSnapshot(\"TalkSensor\").State
                and GetSensorSnapshot(\"CalendarSensor\").State",
                timeOutInSeconds);

      GetAction("InterruptibleSurvey").Run();

    }

  </script>
</trigger>
```