Using Ontologies to Reduce User Intervention to Deploy Sensing Campaigns with the InCense Toolkit

[1,2]Marcela D. Rodríguez, [1]Roberto Martínez, [1]Moisés Pérez, [1]Luis A. Castro, [1]Jesus Favela
[1]Computer Science Department, CICESE Research Center, Ensenada, Mexico
[2]Computer Engineering School, Autonomous University of Baja California, Mexicali, Mexico
marcerod@uabc.edu.mx, {rmartine, perezg, lcastro, favela}@cicese.mx

ABSTRACT

This paper presents the InCense research toolkit to facilitate researchers with little or no technical background to implement a sensing application for mobile phones. To reach this end, InCense provides a GUI and an interactive ontology to enable users to define the configuration of the sensing application, i.e. what sensing components to add, and the flow of the sensing session. We illustrate the ease of use of the InCense platform through a scenario in which both opportunistic and participatory sensing paradigms are used.

Author Keywords Mobile sensing, High-level programming, Social computing

ACM Classification Keywords C.1.3[Other Architecture Styles]: Adaptable Architectures; H.5.2 [User Interfaces]: Graphical User Interfaces (GUI).

General Terms Design, Human Factors, Languages

INTRODUCTION

Mobile devices are being used to capture behavioral cues and social signals to infer interesting patterns and rhythms that would be difficult to detect otherwise or may need a large task force for collection and analysis of data. Several projects have enabled mobile phone users to participate either actively by submitting data or passively as the bearer of a collecting mobile device [1,2,3]. Either perspective constitutes an area of opportunity not only to technical researchers interested in deploying cheap sensors across various locales but also to social scientists interested in understanding the dynamics of specific milieus. However, to deploy a large sensing campaign, several technical considerations have to be faced. That is, deciding the granularity of the sensed information depends of the scope of the project, e.g. using an accelerometer to register movement of a mentally ill elder when facing a problematic behavior (e.g. anxiety, or aggressive behavior) vs registering his physical activity level. While the former may require implementing a simple software component that collects accelerometer raw data to identify a correlation; for the later, it may be desirable to implement a more complex software component that estimates participants' steps, or that provides minute-by-minute activity counts. Additionally, with the aim to collect first-class samples, components may need to be calibrated for the population to be monitored. That is, indicating particular participants characteristics such as height and weight which are used to count steps and measure stride length [4], or by selecting a calibration criteria, e.g. to link activity counts with physical activity energy expenditure [5]. To address the aforementioned drawbacks, we are developing a general-purpose tool for behavioral data collection from mobile phone users: InCense. In this paper, we present that InCense is flexible enough to enable researchers with little or no technical background to implement a sensing application. To reach this end, InCense provides specialized graphical user interfaces that enable users to interact with the different InCense architecture layers. These layers include mechanisms to generate source code of an application sensing campaign by requiring a minimum user intervention. An interactive ontology enables users to register new sensing components to the InCense architecture, and facilitates to customize a sensing campaign, which includes calibration, and configuration tasks (i.e. selecting the sensing components to be used).

UbiComp'12, September 5-8, 2012, Pittsburgh, USA.
ACM 978-1-4503-1224-0/12/09.

INCENSE DESIGN

We created InCense with these design principles:

1) User-centered design. InCense was designed for researchers (users) with low technical skills and interested in conducting large-scale social and behavioral studies.

2) Scalable. InCense is scalable in two aspects: i) to enable the extension of its sensing infrastructure and ii) to handle N mobile participants without affecting the performance of the system. Thus, InCense architecture enables integrating new user-generated components for capturing raw data from sensors such as, Bluetooth and GPS; and components that preprocess these data to outputting higher-level data such as, detecting when a mobile phone user is walking.

3) Flexible. To build a sensing project, researchers need to easily establish the appropriate configuration of the InCense components (i.e., by indicating what components to add, if they need to be activated under certain contextual conditions). Additionally, some of the components need to be calibrated for the population to monitor. To enable this, InCense includes mechanisms to customize the InCense components according to particular requirements for the study (e.g. by indicating threshold values, sample sizes, or participants characteristics). Finally, the InCense mobile application can automatically transfers the collected data to a contextual database to be analyzed.

4) Enabling opportunistic and participatory sensing. InCense can collect participants' information from two main sources: surveys and sensors.

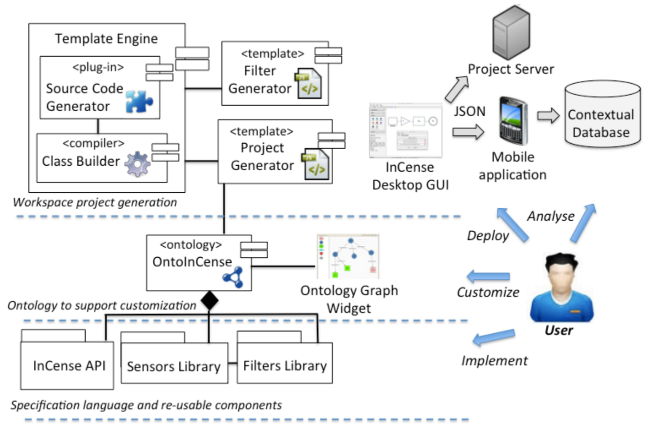


Figure 1. InCense Architecture

IMPLEMENTATION MODEL

The InCense implementation model defines the main software modules or components used to build a sensing campaign project. As presented in Figure 1, users interact with the components of the different layers of the InCense Architecture to develop a project:

Specification language and re-usable components

InCense provides an API, which provides the classes and abstractions which are necessary to implement the modules of a sensing campaign project:

- *Sensor and Filter.* Sensors are software components implemented to interface with sensors available on the mobile phone. And filters are software components implemented to receive raw data from sensors components with the aim of preprocessing or condensing these data, outputting higher-level data to facilitate analysis and decision-making. For example, implementing filters to obtain features commonly used in signal processing such as peaks, standard deviations, and/or means (such as the MeanFilter). The InCense API provides abstract classes with methods to implement, configure and control the execution of filters and sensors, such as setSampleSize(), start(), stop(). Currently, InCense includes a package with Sensor components implemented to get data from the accelerometer, audio, GPS, NFC, Bluetooth, and to monitor the battery level; and a package with Filters having different design granularity levels (i.e. a filter may consist of several lower level filters).
- *Trigger.* A set of classes is provided to define triggers, such as the DataTrigger class, which is instantiated to add decision-making as an array of conditions. Triggers receive as input raw or processed data from sensors or filters. When certain conditions are met, they can start other sessions or surveys.
- *Survey.* A survey is formed by multiple choice or open-ended questions that mobile phone users respond to.
- *Session.* A session is a group of different components connected to achieve a specific sensing goal. Sensors, filters, triggers, surveys, sinks, and even other sessions could be added to the workflow of a session. This session is programmed in the main class of the project as presented in Figure 3b.
- *Sink.* It is a data pool wherein the information of a sensing session is assembled into files before being sent to a context database. The DataSink class enables users to create a file with information formatted according to the Javasript Object Notation (JSON), and the AudioSink class is used to create .RAW audio files.

Ontology to support customization

An ontology is defined as a set of representational primitives to model a domain of knowledge or discourse [6]. The OntoInCense ontology, presented in Figure 2, was designed to enable a user to customize an InCense project by: i) specifying the configuration of the sensing session, i.e. the arrangement of the session modules according to the goal of the sensing project; ii) calibrating the sensing modules, filters and sensors for the population. To accomplish this purpose, the ontology's higher level is a representation of the workflow of a session, showing that a *Project Session* is composed of *Capturing Modules* (*Sensors, Filters* and *Surveys*), *Triggers*, and *Sinks*. It also represents the relationships allowed among these capturing modules. That is, a capturing module has a condition to activate other capturing module, and the collected data is outputting to a *Sink* which can be a *DataSink*. For instance, a RFID sensor may activate a Survey to ask the participant how she feels after finishing her physical activity. The lower level of the ontology represents the set of sensors and filters provided by InCense, such as the accelerometer sensor (*Acc*) and the *StepCounter* filter, which can be used

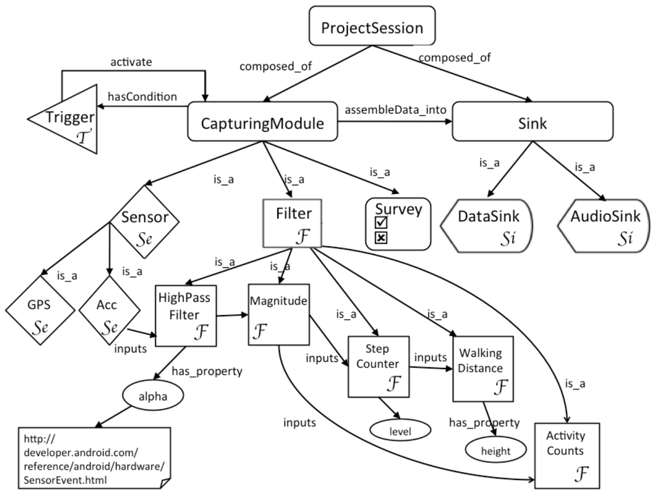


Figure 2. Design of the OntoInCense Ontology

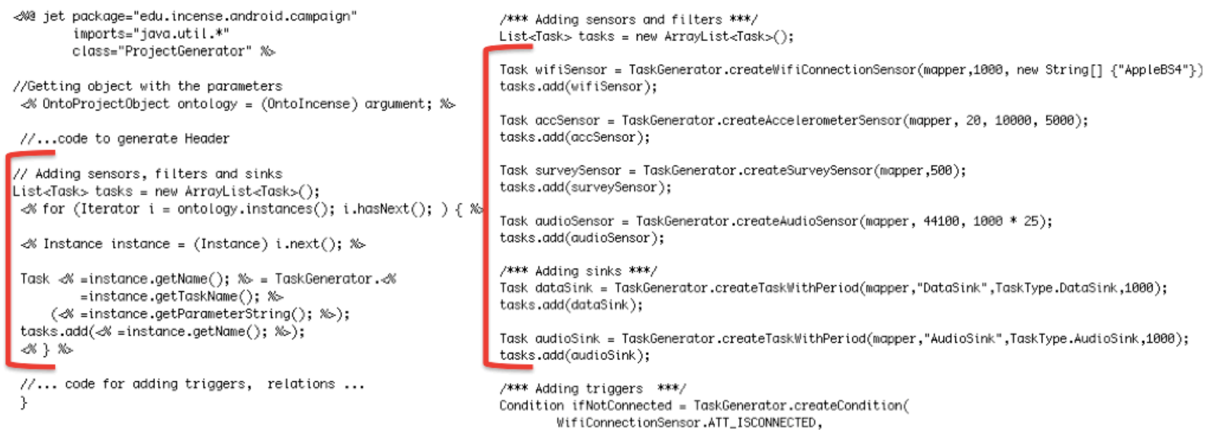
to monitor participants' physical activities. The relations among these Capturing Modules nodes represent dependencies, i.e. a filter needs data from other filter or sensor. Finally, the Capturing Modules have attributes, which are represented as leaves in the ontology tree. Attributes specify the information that can be changed by the user to calibrate a component for the requirements of the sensing project. Finally, descriptive information can be associated with these attributes to explain the criteria used for the calibration. Thus, OntoInCense not only enables the customization of an InCense project but also facilitates users to understand the implementation model of InCense. To provide a GUI for interacting with the ontology, we are using the Graph Widget for creating, and populating instances of classes in Protégé [7]. This facilitates not only the customization of InCense, but provides a means for visualizing the configuration of InCense as a network of nodes and relationships among nodes.

Workspace project generation

Aiming to speed up the implementation model of InCense, we developed a desktop-based Graphic User Interface (GUI) presented in Figure 4. It consists of an environment wherein the researcher can select boxes from the left panel and plug them through, as showed in the right panel. Each box can represent a sensor, a filter, a trigger, a survey, a session, or a sink. This graphical user interface uses the ontology to check if the user is forming correctly a workflow sensing session. Thus, according to the ontology of Figure 2, the user can add an *Accelerometer* sensor (Acc) that inputs data into the *HighPassFilter*, but the Acc sensor cannot inputs data into the *MeanFilter* since it will not be able to process the accelerometer raw data. When the user has selected and connected boxes, he can proceed to deploy the application. The GUI uses a passive code generation, which requires a low users intervention to implement an application. This intervention is provided through the customization of the ontology through the Graphical Widget. The code generator is based on the Java Emitter Template (JET) Engine of the Eclipse Modeling Framework (EMF) [8] project. It provides a script-based language, which is a subset of the JSP syntax that makes it easy to write templates that express the source code we want to generate. As presented in Figure 1, this engine provides sub-systems to process a set of script-based templates that we designed to generate the source code for extending and implementing a sensing application. Templates receive an input model, which is a hierarchical representation of the elements (nodes) to include in the source code. This input model should be provided in XML format, which allows great expressiveness. For InCense, this input model is provided through the ontology. Figure 3a presents the template designed to generate the main InCense class, which implements a sensing session. It emphasizes the template section that parses the XML expressing the nodes and relations of the OntoInCense ontology. As presented in figure 3b, the source code generated corresponds to a sensing session that includes the WiFi, accelerometer and audio sensors, and a survey. Additional code of this template inserts the filters and triggers to define the complete workflow of the session. Similarly, using templates facilitates the integration of new sensors and filters into InCense. For instance, to add a filter, users should write in the code for preprocessing data at the FilterTemplate. Afterwards, users should extend the ontology (OntoInCense) to indicate the information necessary to customize and calibrate the filter. Finally, to deploy a project, the generated source code is compiled to produce an application descriptor JSON file, which is a configuration file stored on the Project Sever and installed on the mobile phone. The application on the mobile phone is capable of receiving an updated configuration file from the Project Server and re-programming itself.

A MOBILE SENSING APPLICATION

Based on previous works [1,2,9], we created scenarios to inform the design of InCense, such as the following: *"A public health organization (PHO) is interested in comparing the walking habits of older adults in the winter and in the spring. They began using InCense for data*



```
<%@ jet package="edu.incense.android.campaign"
        imports="java.util.*"
        class="ProjectGenerator" %>

//Getting object with the parameters
 <% OntoProjectObject ontology = (OntoIncense) argument; %>

 //...code to generate Header

// Adding sensors, filters and sinks
List<Task> tasks = new ArrayList<Task>();
 <% for (Iterator i = ontology.instances(); i.hasNext(); ) { %>

 <% Instance instance = (Instance) i.next(); %>

 Task <% =instance.getName(); %> = TaskGenerator.<%
         =instance.getTaskName(); %>
     (<% =instance.getParameterString(); %>);
 tasks.add(<% =instance.getName(); %>);
 <% } %>

 //... code for adding triggers,  relations ...
 }
```

a)

```
/*** Adding sensors and filters ***/
List<Task> tasks = new ArrayList<Task>();

Task wifiSensor = TaskGenerator.createWifiConnectionSensor(mapper,1000, new String[] {"AppleBS4"})
tasks.add(wifiSensor);

Task accSensor = TaskGenerator.createAccelerometerSensor(mapper, 20, 10000, 5000);
tasks.add(accSensor);

Task surveySensor = TaskGenerator.createSurveySensor(mapper,500);
tasks.add(surveySensor);

Task audioSensor = TaskGenerator.createAudioSensor(mapper, 44100, 1000 * 25);
tasks.add(audioSensor);

/*** Adding sinks ***/
Task dataSink = TaskGenerator.createTaskWithPeriod(mapper,"DataSink",TaskType.DataSink,1000);
tasks.add(dataSink);

Task audioSink = TaskGenerator.createTaskWithPeriod(mapper,"AudioSink",TaskType.AudioSink,1000);
tasks.add(audioSink);

/*** Adding triggers  ***/
Condition ifNotConnected = TaskGenerator.createCondition(
        WifiConnectionSensor.ATT_ISCONNECTED,
```

b)

Figure 3. InCense Code: a) Template for generating the source code of an application, b) Generated source code.

gathering from 392 individuals during two weeks in the middle of January, and then again in May. The application captures the individual location, the activity level obtained from the accelerometers, and temperature obtained from the Internet by using time and location. A filter infers from the GPS and accelerometer, if the individual is walking or in a vehicle as he leaves his home. When InCense detects that the user is back at home, the mobile phones, will ask the individuals to complete a survey with question related to the activity being performed and their wellness. The data captured from the individuals is sent to the PHO to find interesting correlations with standard statistical packages."

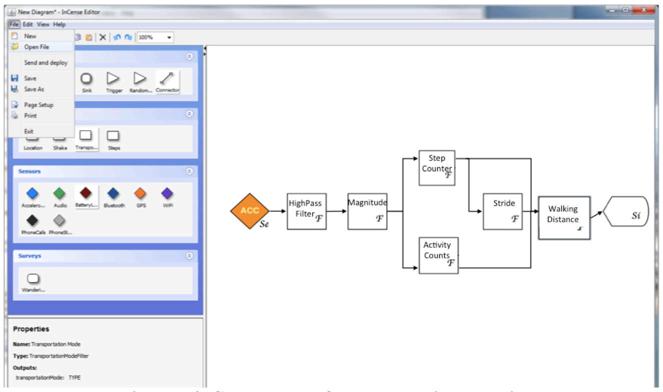
To implement the application sensing for this scenario, the PHO user opens the desktop-based GUI of InCense, which consults the ontology to obtain information of the available filters, sensors and their configurable attributes. Thus, when the user adds the StepCounterFilter, it will also incorporate the low-level filters that it uses according to the ontology design (such as the HighPassFilter and Magnitud Filter). The user also adds a trigger to launch a survey when it is detected that the user finished their walking. While the InCense modules are added, the configuration is being validated to check the modules connections in order for the project can be generated correctly during the compilation.

RELATED WORK

There have been several projects [12-14] that have aimed at exploiting the massive penetration of mobile devices as well as their increasing sensing capabilities. For instance, MyExperience [9] supports the easy re-configuration of the application via a file, automatically triggered sessions, event-triggered actions, and explicit user input. AnonySense [10] aims at protecting the anonymity of the subjects participating in an opportunistic campaign by providing a secured network. To reduce the amount of data that needs to be transmitted over the network, the BeTelGeuse [11] uses extendable plug-ins to condense raw data in real time before sending them to a server where all the data is collected. The CRN Toolbox [12] includes a desktop-based graphical environment to design ubiquitous computing applications by selecting boxes and connectors representing different sensors and filters. InCense is a novel toolkit that provides users interfaces based on ontological models to support the customization of sensing campaigns. OntoIncense acts as an interface between the InCense GUI and the layer containing the bundle of components for capturing data. This facilitates developers to extend InCense by adding specialized sensors and filters.

CONCLUSIONS AND FUTURE WORK

One of the main contributions of InCense is the graphical interface, which can lower the barrier of this type of technology for non-technical users, especially researchers that would be interested in conducting these types of studies. This interface can enable researchers with little or no technical expertise to implement projects in a reduced time as opposed to other tools that might need some programming. We plan to conduct an evaluation with researchers working on the social sciences to receive feedback that enables us to enhance our research kit.



Properties
Name: Transportation Mode
Type: TransportationModeFilter
Outputs:
transportationMode: TYPE

Figure 4. Snapshot of the graphical environment.

ACKNOWLEDGMENTS

This work was partially financed by CONACYT under contracts CB-20080-105674 and CB-2010-01-153863, and through the scholarships provided to the 2nd. and 3rd. authors.

REFERENCES

1. J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M.B. Srivastava, M.B. "Participatory Sensing," ACM SenSys Conf., Boulder, Colorado, USA, 2006.
2. N. Eagle and A. Pentland, "Reality mining: sensing complex social systems," Personal Ubiquitous Computing, vol. 10, pp. 255-268, 2006.
3. T. Choudhury A. Pentland, "Sensing and Modeling Human Networks using the Sociometer," IEEE Symposium on Wearable Computers, 2003.
4. N. Zhao. “Full-featured pedometer design realized with 3-Axis digital accelerometer”. Analog Dialogue, 44-06, 2010.
5. C.E. Matthews. “Calibration of accelerometer output for adults”. Med Sci Sports Exerc 37: S512–S522, 2005.
6. T.R. Gruber. “Toward principles for the design of ontologies used for knowledge sharing”. Int. J. Hum.-Comput. Stud. 43, 5-6 (December 1995), 907-928.
7. Protégé, http://protege.stanford.edu/doc/tutorial/
8. Eclipse, http://www.ibm.com/developerworks/
9. J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay, "MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones," ACM Conf. on MobiSys San Juan, Puerto Rico, 2007.
10. C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, N. Triandopoulos, "Anonysense: privacy-aware people-centric sensing," ACM Conf. on Mobile systems, applications and services, CO, USA,2008.
11. J. Kukkonen, E. Lagerspetz, P. Nurmi, M. Andersson, "BeTelGeuse: A Platform for Gathering and Processing Situational Data," IEEE Perv. Comp., vol. 8, 49-56, 2009.
12. D. Bannach, O. Amft, P. Lukowicz, "Rapid Prototyping of Activity Recognition Applications," IEEE Perv. Comp., vol. 7, 22-31, 2008.