


# Mobile sensing for behavioral research: A component-based approach for rapid deployment of sensing campaigns

International Journal of Distributed  
Sensor Networks  
2019, Vol. 15(9)  
© The Author(s) 2019  
DOI: 10.1177/1550147719874186  
journals.sagepub.com/home/dsn  


Ivan R Felix<sup>1</sup>, Luis A Castro<sup>1</sup> , Luis-Felipe Rodriguez<sup>1</sup> and Oresti Banos<sup>2</sup>

## Abstract

Collecting experimental data from multiple sensing devices has just recently become quite popular in behavioral and social sciences. Among existing devices, mobile phones stand out as they allow researchers to collect data from individuals in an unbiased, precise, unobtrusive, and timely manner. Current mobile sensing applications are typically developed from scratch, provide no reusable components, and frequently do not take advantage of the devices' processing capabilities. In light of such limitations, this work presents a novel tool that leverages mobile phones not only to collect data via their sensors but also to process them on the device as soon as they are gathered. The tool provides researchers with easy-to-use services that allow them to configure the required processing routines on the mobile phones. This work proposes a new approach for rapid deployment of sensing campaigns targeted at scientists with basic technical knowledge and requiring low effort. We performed an evaluation aimed at determining whether there is a significant improvement in terms of user effectiveness and efficiency in the definition of new components. The results suggest that the proposed tool speeds up the time and reduces the effort taken for setting up and deploying a sensing campaign.

## Keywords

Mobile phone sensing, sensing tool, data processing

Date received: 30 April 2019; accepted: 7 August 2019

Handling Editor: Jaime Lloret

## Introduction

Collecting data from humans has always been both a paramount and challenging task for researchers.<sup>1,2</sup> Although data are virtually needed to conduct any experiment or field investigation, obtaining such data is not always easy or even feasible. This is especially true for social and clinical studies, where relatively large groups of people or cohorts are to be observed in a natural setting, with multiple situations and variables to be measured. In fact, this takes a considerable amount of effort for both researchers and participants. In recent years, new techniques have been proposed to assist the collection of experimental data. Most recently, mobile phone-based sensing, also referred broadly to as mobile sensing, has emerged as a method used to assist the required work to obtain such data.<sup>3</sup> Mobile phones (or

smartphones) are particularly suitable for this kind of activities due to their inherent characteristics: portable, unobtrusive, ubiquitous, relatively easy to acquire, and affordable. These types of devices have sensors that allow researchers to obtain unbiased data about users and their environment.<sup>4,5</sup> Also, since mobile phones are small-sized computers, they have storage, processing, and communications capabilities that can be exploited by developing applications that take advantage of those

<sup>1</sup>Sonora Institute of Technology (ITSON), Ciudad Obregón, Mexico

<sup>2</sup>University of Granada, Granada, Spain

### Corresponding author:

Luis A Castro, Sonora Institute of Technology (ITSON), DES IyT, 5 de Febrero 818 Sur, Ciudad Obregón 85000, Mexico.

Email: [luis.castro@acm.org](mailto:luis.castro@acm.org)



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (<http://www.creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work

without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

capabilities. According to GSMA intelligence 2019, two thirds of the world's population are unique mobile subscribers, which accounts for almost 9 billion mobile connections worldwide, surpassing the current world's population. In this context, mobile phones seem an unparalleled tool for conducting human-centric experiments in the wild, continuously at region, nation if not world-scale.

Despite the aforementioned potential of mobile sensing to conduct human-centric studies, there are still various challenges to overcome. On one hand, researchers, particularly non-STEM (science, technology, engineering and mathematics), may find difficult to set up experiments compared to their traditional tools. On the other hand, mobile sensing applications have to be typically developed from scratch, thus delaying the realization of the study. Also, applications are most often tailored to the needs of a particular study and therefore their reusability for other (similar) studies is low, which could make it less practical and affordable for the long term. In view of these limitations, a new generation of general-purpose mobile sensing frameworks has been proposed.<sup>6–11</sup>

In this work, we propose a new approach for rapid deployment of sensing campaigns targeted at scientists with basic technical knowledge and requiring low effort. We performed an evaluation aimed at determining whether there is a significant improvement in terms of user effectiveness (i.e. new data processing routines work correctly) and user efficiency (i.e. they can be developed in a short time) in the definition of new components using the proposed approach. The results suggest that the proposed tool speeds up the time and reduces the effort taken for setting up and deploying a sensing campaign. This presented proposal is based on the component-based approach<sup>12,13</sup> as the aim is to support the creation of components that implement data processing routines used to define mobile sensing campaigns (from these pre-defined components). In this manner, the construction of mobile sensing campaigns relies on already developed components. This allows researchers to reuse data processing routines (i.e. components) in sensing campaigns, which typically vary in terms of when to trigger data collection (i.e. events) or what data to collect (i.e. raw or processed sensor data). Moreover, the component-based approach in mobile sensing applications allows researchers to download and incorporate pre-existing components (i.e. data processing routines) at runtime into the mobile phone when changes to the collection protocol (or data formats) are required and the detection campaign is already underway. The component-based approach is a widely used technique that has been utilized to approach crucial problems in domains such as face recognition<sup>14</sup> and cloud computing.<sup>15</sup>

The rest of the article is structured as follows. In section "Related work," we provide a brief literature review on specific- and general-purpose mobile sensing applications as well as a discussion about related work. In section "A proposal for rapid deployment of mobile sensing campaigns," we present a new component-based approach for rapid deployment of mobile sensing campaigns, and in section "Technical implementation of the proposed approach," we discuss its implementation details. We present results regarding the evaluation of the proposed component-based model in section "Evaluation." Finally, in section "Conclusion and future work," we provide some concluding remarks and future research directions.

## Related work

Collecting data through portable, sensor-enabled devices, such as mobile phones, has become an increasingly used technique to obtain experimental data. As a result, mobile sensing has gained enough relevance to be considered a research area by itself. Some work has been published about this subject that proposes concepts and classifications,<sup>5</sup> where sensing paradigms (e.g. participative and opportunistic) and sensing scales (e.g. individual, group, and community) were identified. In a similar fashion, Khan et al.<sup>16</sup> discussed the domains (e.g. health, social, human-computer interaction, and psychology) that leveraged mobile phone sensing and listed applications that were developed to collect data for specific purposes.

Two types of mobile phone sensing applications are identified from the literature: (1) applications designed for a specific purpose, and (2) applications that seek to provide an adaptable platform that enables researchers to match the requirements of their study or project with basic technical knowledge and low effort.

### *Specific-purpose mobile sensing applications*

Usually, researchers develop custom, domain-specific mobile sensing applications in order to obtain data from participants and their environment, monitor their activities, give feedback to them, and/or allow users to interact with the device in original ways. We describe some of the most representative mobile sensing applications designed for a specific domain and/or purpose.

One of the very first mobile sensing applications is Ubifit Garden.<sup>17</sup> The main idea behind this application is to promote the wellbeing of participants by motivating them to perform physical activity through visual compensation. Ubifit Garden could identify several activities (e.g. walking and sweeping) and keep a record of them. For each activity that Ubifit Garden recognizes, the user gets a flower that is displayed on the

background of the phone's screen. After completing a goal, that is, a series of activities, a butterfly is drawn on the screen along with the flowers, thus developing vivid allegories which are easy for users to relate to their current physical state. In a similar direction, BeWell is developed to monitor, model, and promote the wellbeing of people.<sup>18</sup> In this case, BeWell monitors, via the mobile phone's accelerometer, GPS, and microphone, three aspects of the user: the amount of sleep, the physical activity carried out, and the social interactions performed. The application quantifies these aspects of wellbeing and offers visual information to the users on how well they are doing. It uses a shoal to determine the level of social interaction, an orange fish to establish the level of physical activity, and the brightness level to show the amount of sleep, all of them being portrayed on the background of the mobile phone's screen.

Despite the high prevalence of mobile sensing applications for health and wellbeing purposes (e.g. assisting older adults<sup>19-23</sup>), there are several other areas that can benefit from the data collected via mobile phones. For example, CarSafe<sup>24</sup> uses both cameras of the mobile phone (for those cells that have both front and rear camera) to monitor the state of the driver and the road. This application uses computer vision and machine learning algorithms to detect whether the driver is tired or distracted and at the same time checks the road conditions. Similarly, Nericell<sup>25</sup> and PotHole<sup>26</sup> use mobile phone sensor data (accelerometer and GPS) to determine the conditions of the road. Noise pollution measurement through phones is developed in community scale applications such as EarPhone<sup>27</sup> and SoundOfTheCity.<sup>28</sup> These applications seek to map large areas and label zones according to their noise pollution level. Users of these applications provide their location and noise samples using the mobile phone GPS and microphone, respectively. Disaster management and crime prevention are other examples of domains that benefit from mobile sensing. Particularly, iSafe promises to evaluate the safety of users based on their spatial and temporal dimensions.<sup>29,30</sup>

### *General-purpose mobile sensing applications*

The development of a mobile sensing application from scratch is a task that normally requires considerable time and human resources as well as high technical knowledge. Researchers have previously proposed tools whose purpose is to reduce the effort required to prototype or just use a mobile sensing application. These tools provide preprogrammed and configurable functionalities that help solve common tasks that could be needed in a sensing campaign. Using a configuration interface provided by these platforms, users can establish an expected behavior from the application without having to modify the application's source code. Some

of the most relevant configurable mobile sensing applications are described next:

- MyExperience<sup>6</sup> is one of the very first configurable mobile sensing applications. MyExperience implements a comprehensive architecture that supports a fair number of sensors and allows the user to add more. The application can monitor data coming from the sensors and evaluate such data using user-defined conditions and launch actions using triggers. MyExperience also allows users to define their custom actions and has the ability to send the data to a remote repository. MyExperience provides an XML interface to allow researchers with low technological abilities to configure it.
- AndWellness<sup>7</sup> is another configurable mobile sensing application consisting of three main elements: (1) a server to configure studies and store collected data, (2) an Android application to perform the data collection, and (3) a dashboard used to display participants' statistics and data. The Android application mainly executes surveys so that participants can input their data. These surveys can be launched based on a schedule or on events detected using collected data through the phone sensors supported by AndWellness, that is, location traces (GPS) and activity inference (accelerometer).
- FunF<sup>8</sup> is an open-source framework aimed to help developers to create mobile sensing applications by providing libraries that allow us to easily access the sensors of the phone and send the obtained raw data to remote repositories. It also presents an open-source application named "funf journal" that is based on the funf framework, which allows the user to select the data sources (phone sensors) to obtain information and offers several means to export collected data.
- InCense<sup>9</sup> is a configurable mobile sensing application that implements a handful of practical features: a graphic user interface that is used to design the sensing campaign; remote servers that store sensing campaigns and available mobile phones to execute them; an Android client to run the campaign; and a context database server that is used to store the collected data. Similar to the aforementioned tools, InCense's client supports most of the commonplace sensors available on the phones, can evaluate data and execute actions using triggers, save the collected data in local sinks (i.e. phone's storage), and send it to remote repositories. To the best of our knowledge, this is the only tool that addresses on-device data processing using elements known as filters.

**Table 1.** Comparative analysis of general-purpose mobile sensing applications.

Feature	MyExperience <sup>6</sup>	AndWellness <sup>7</sup>	Funf <sup>8</sup>	InCense <sup>9</sup>	mHealthDroid <sup>10</sup>	Aware <sup>11</sup>	This work
F1 Dynamic reconfiguration	•	•		•		•	•
F2 User-defined filters or data processing units				•		•	•
F3 Data gathering	•	•	•	•	•	•	•
F4 Remote data transmission	•	•	•	•	•	•	•
F5 Event-based actions	•	•		•	•	•	•
F6 Explicit user data entry	•	•		•		•	•
F7 Provide a set of extensible libraries as a starting point			•		•	•	•
F8 Data processing routines load at runtime							•
F9 Web interface for contributing new data processing routines							•

- mHealthDroid<sup>10</sup> is an open-source framework that akin to funf is aimed at facilitating the rapid and easy development of mobile health and biomedical applications. The framework implements several functionalities to support resource and communication abstraction, biomedical data acquisition, health knowledge extraction, persistent data storage, adaptive visualization, system management, and value-added services such as intelligent alerts, recommendations, and guidelines.
- Aware<sup>11</sup> is an Android-based open-source mobile context instrumentation framework. It provides a client-server mobile framework that supports the collection of unobtrusive passive sensor data. It follows a modular approach: the AWARE client app abstracts the communication with the sensors and the acquisition of data; then, data are used to generate context through customizable code extensions named plugins.

From the researcher's perspective, having a flexible mobile sensing platform that can be used multiple times for different campaigns is ideal. That is, that the platform allows rapid, straightforward configuration for data collection in a new sensing campaign that may have different requirements or else change the configuration of an ongoing campaign, without too much hassle. Although, this may seem trivial, there are a lot of design features that a mobile sensing platform must consider in order to be as flexible as possible.

As shown in the previous works, some of the revised mobile applications are designed to be reconfigured so that users are able to define new data to be collected or new events without rebuilding the mobile sensing application (F1) as well as to support the triggering of actions based on user-defined events (F5), which is

important as mobile context is highly variable. Importantly, a key characteristic of these applications is to provide diverse mechanisms to support automatic data collection from mobile phone sensors (F3) and in the form of surveys to obtain user assessments (F6). Mobile sensing applications are also capable of sending these recollected data over the network at pre-defined events (F4); this is highly desirable since some users may be available to collect data manually. In order to extend the functionality of these applications, platforms are being designed to provide users with mechanisms to (1) allow the inclusion of new filters or data processing components to the platform (F2), enable on-device pre-processing or simple classification; and (2) access and modify existing libraries for developing sensing campaigns (F7). All of this is done using a user-friendly Web interface (F9) for researchers who may not have highly specialized technical knowledge and skills. Finally, mobile sensing applications include a means through which data processing routines can be loaded at runtime without redeploying the mobile sensing application (F8); this feature means that new data processing components can be added into the mobile phone without rebuilding the mobile phone application. In Table 1, we present a comparative analysis of the applications discussed earlier in terms of the mentioned key aspects of general-purpose mobile sensing applications.

### Limitations of existing mobile sensing tools

As previously mentioned, there are tools designed to work in particular domains, which can hardly be used for different purposes without changing their underlying design. On the contrary, there are other tools that can be reconfigured to be used with different sensing needs. On the latter category, most of the tools

discussed do not support on-device processing but they rather send all the collected raw data to a server for off-line analysis.

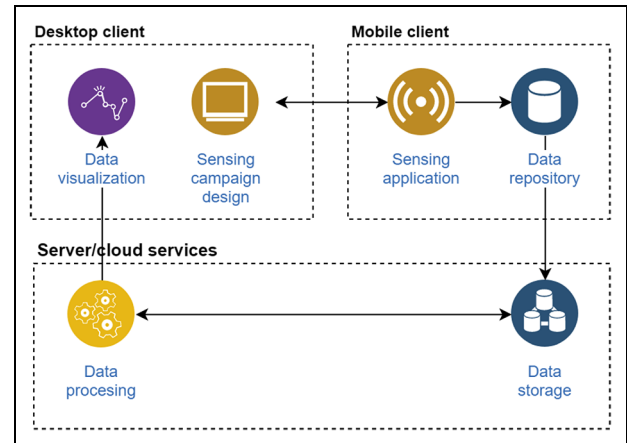
However, in many research studies it may be necessary to perform on-device processing. A crucial reason is privacy, as the ability to process data while in the phone helps to avoid tracking the identity of a user (e.g. personal info and voice). Ensuring privacy can in turn increase the levels of user participation in sensing campaigns, as people often refuse participation due to privacy concerns. Another great benefit derived from on-device processing is storage and network resource optimization. Sensors can generate large amounts of data in short time periods, quickly scaling up as the number of participants grows. By processing sensor data on the device, it is possible to obtain relevant features from it, like the activity performed by the user from accelerometer data or significant chunks from an audio signal, which are most frequently needed in scientific studies rather than raw sensor data.

Despite the clear advantages of on-device processing, some problems inherent to it may arise. One issue has to do with the difficulty in abstracting completely the programming of processing routines in such a way that the non-technical researcher or third party may not require deep knowledge about an entire software application. Another problem could be that as time goes by, many processing routines can be added, which would eventually translate into a large application that might consume a considerable amount of device resources (e.g. processing, memory, and storage), thus potentially deterring participants even more. At the same time, this could make it difficult to maintain in terms of software development and application deployment.

Creating data processing routines to obtain meaningful data from the device is not an easy task. If a place existed where third parties could share their work, researchers could benefit by saving time and effort when reusing that work, that is code. Most of the time, customized applications are not shared or made public by their authors and, therefore, other research groups rarely take advantage of them; thus, potentially slowing down progress of the research community.

## A proposal for rapid deployment of mobile sensing campaigns

The literature on configurable tools shows a recurrent, conventional approach that drives the design of such type of tool. This conventional approach has three main elements. The first is usually a Desktop client that enables the user to perform the *Sensing campaign design*. It also typically features a *Data visualization* module, either based on a Graphical User Interface (GUI) or commands/functions. The second is the



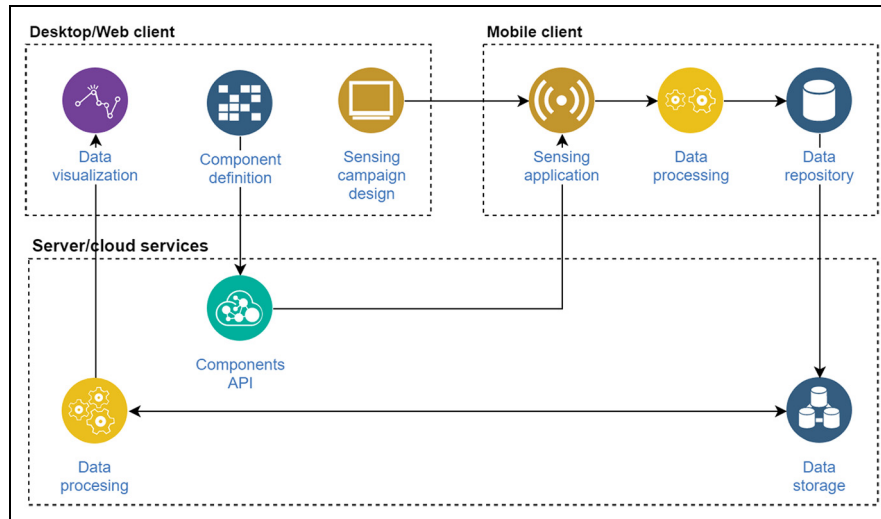
**Figure 1.** Conventional approach in mobile sensing.

Mobile client (i.e. mobile phone application) in charge of executing the sensing campaign through a *Sensing application*, which in turn stores collected data locally in a *Data repository*. The third is a series of services such as *Data storage* and *Data processing*. This approach is depicted in Figure 1.

One key disadvantage of this approach is that in order to create an on-device data processing routine, the user needs to deeply understand the mobile client and be able to rebuild it, which typically requires technical knowledge and skills. We propose a modification to this conventional approach that (1) adds an interface for *Component definition*, that is, create new data processing units such as step counter; (2) modifies the *Server/cloud services* so that it supports receiving, storing, searching, and generating components through the *Components API*; and (3) allows modifications in the Mobile client to support downloading and integrating new on-device *Data processing components* at runtime. The proposed component model is shown in Figure 2, which derives from previous works in the area.<sup>9,31</sup>

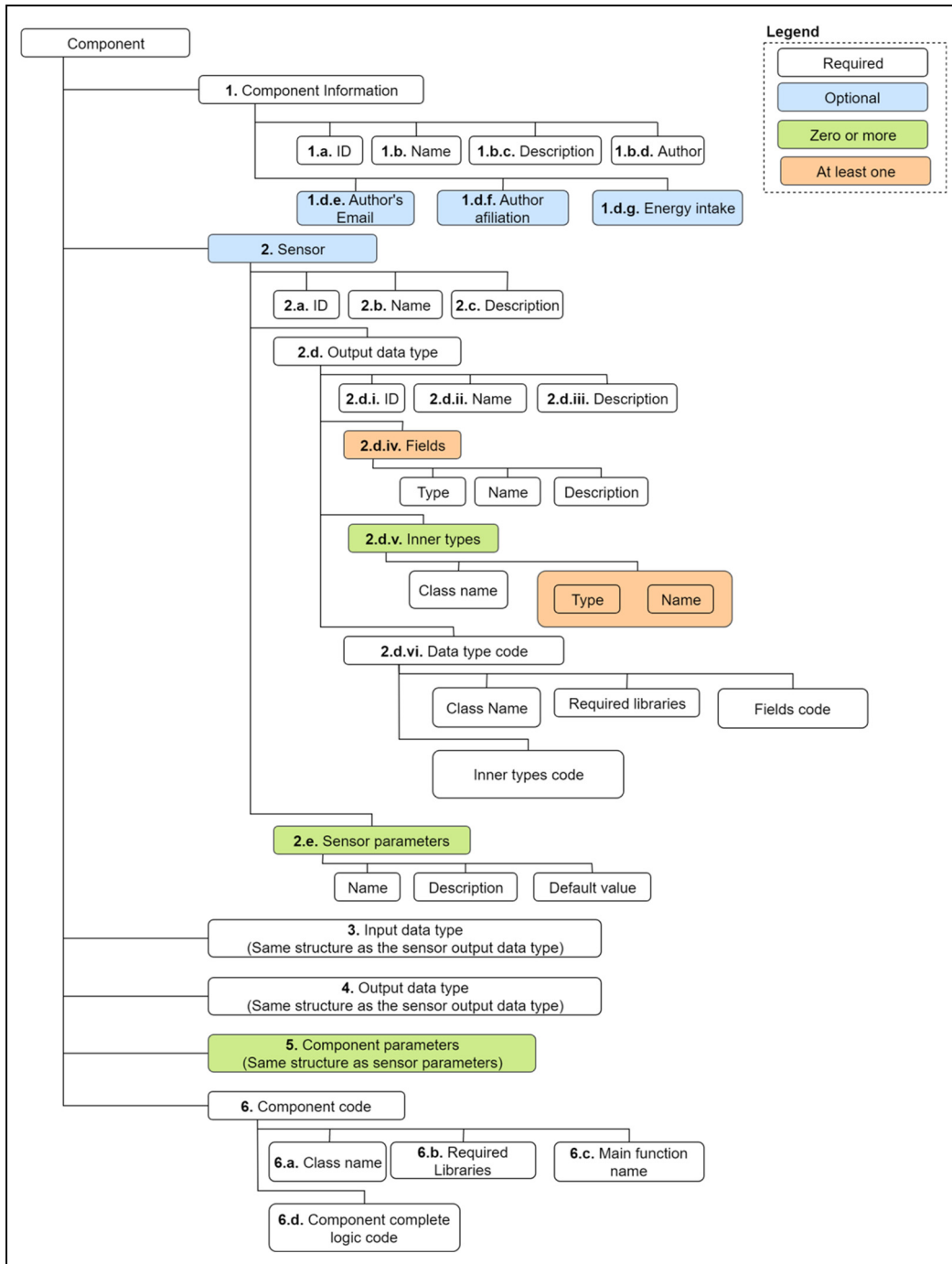
In particular, the presented proposal abstracts data processing routines and encapsulates them in data processing units called components. Components are written in JSON (JavaScript Object Notation). A component is a fundamental piece that is composed of a series of fields, which describe the component and store the logic to be executed, an expected input, and a defined output. Figure 3 shows the structure of the components, which is depicted as a high-level schema. The component structure is described as follows:

1. *Component Information*. This element was used to group a set of fields that are used to provide information about the component. All these fields will be used as inquiry criteria by a search engine so that other people can be aware of their existence and reuse them:



**Figure 2.** The proposed component-based model.

- (a) ID: unique identifier of the component.
  - (b) Name: tag that suggests what the component is for.
  - (c) Description: explanation of what and how the component does its tasks. Here, the creator should explain the format of the data that the component requires, the algorithms used for data processing, and the format of the output.
  - (d) Author: name of the creator.
  - (e) Author's email: optional field where the email of the author is stored; this field could be used for contact purposes.
  - (f) Author affiliation: affiliation details of the author.
  - (g) Energy intake: optional field that describes the approximate battery power consumption of the referred component. It represents the percentage of the battery the component will consume from a fully charged mobile phone until it is completely depleted. It should be noted that this field is rather informative and just an estimation as the value may change among different phones.
2. *Sensor*. This element is used to indicate that the data required by the component come from a logical sensor defined in the platform. The data will be in the format of the output of that sensor. This element can be optional since the data the component will use can be obtained from another component.
    - (a) ID: identifies the sensor as a unique resource since it can be used by several components.
    - (b) Name: name of the sensor.
    - (c) Description: explanation of what the sensor does, what type of data are collected, and their format.
    - (d) Output data type: complex field where a structured description of the sensor output format is stored. In the InCense platform, users can create custom data types and use them as the output of sensors and input/output of components and other elements of the sensing campaign.
      - (i) ID: data types are resources that can be reused among campaign elements, thus a unique identifier is assigned to them.
      - (ii) Name: name providing a general idea of the data type.
      - (iii) Description: when Data Types are customized such as Socialization Data Type, this includes an explanation of the contents of the data type.
      - (iv) Fields: represent each of the basic types that will comprise the data type (basic types could be int, char, float, string, etc.). For each field whose type is declared, both name and description should be stated.
      - (v) Inner types: in case none of the existing basic types complies with what the scientist needs, additional types can be defined within the existing data type and use them as a field. Inner types are declared with a name and a set of inner fields subsequently declared with a type and a name.
    - (e) Sensor parameters: some of the sensors of the mobile phone can be configured to behave in a certain way. These configuration options vary among sensors (e.g. sampling frequency per second and precision). For each sensor parameter that is defined,



**Figure 3.** A tree representation of the fields that describe the proposed component schema.

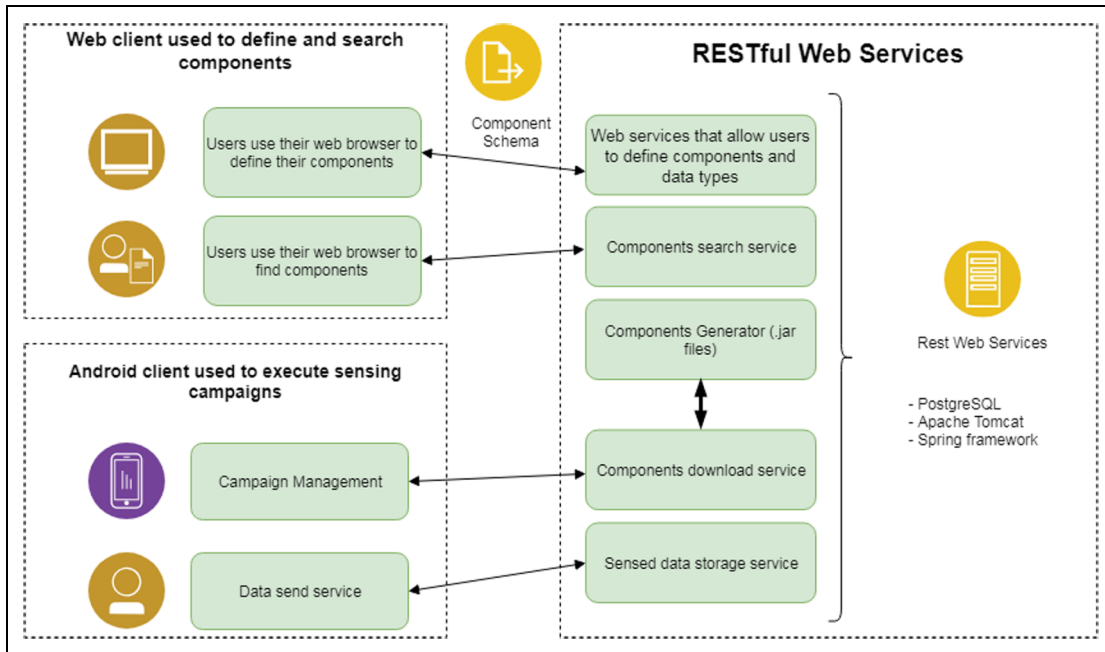
a name needs to be stated that suggests what the parameter is for, a detailed description and possible values within that description, and a default value that would be used in case this parameter is not provided.

3. *Input data type.* This element of the schema uses a defined resource as the one stated in the

output data type element of the sensor. It is used to define the data that can be received by the component. For two campaign elements to be compatible, the output data type of the source must be the same data type of the input of the destination.

4. *Output data type.* This element also uses a defined resource as the output data type element





**Figure 4.** Implementation of the component-based model.

of the sensor, which also defines the output data of the component.

5. *Component parameters.* Components can also be parametrized to customize their behavior just as sensors do (e.g. specify one method to calculate if a person is still or moving, choosing between several options for calculation like fuzzy logic or artificial neural networks). Each parameter must have specified its name, description, and the default value in case nothing is configured.
6. *Component code.* The algorithms that will process the data are defined. All the codes must be created as a java class with all the required functions declared inside the class.
  - (a) *Class name:* stores a suggested name of the class. Since there could be more than one component with the same name, an internally handled name will be used once the component class is integrated with the InCense client.
  - (b) *Required imports:* all the external libraries that are required by the component class must be listed in this field. Available libraries are the ones that exist in the Java and Android SDK.
  - (c) *Main function name:* this is the function that will be called so that the component performs its processing task. This method must receive a parameter of the input data

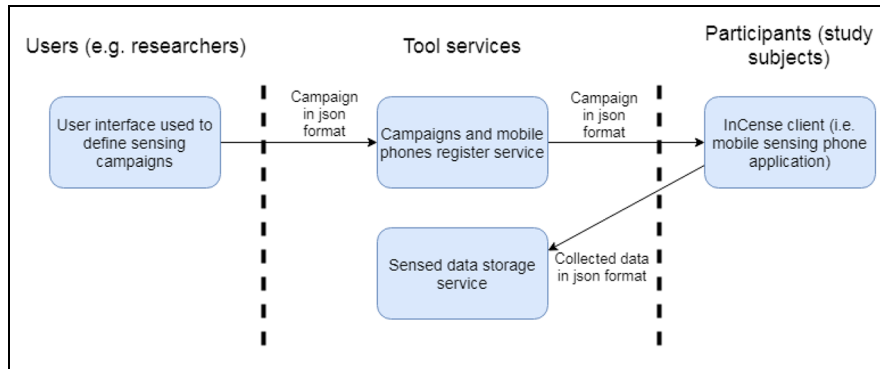
type and return a type of the output data type.

- (d) *Component complete logic code:* all auxiliary functions, properties, and variables required by this component are declared in this section.
- (e) *The creation of components aims to provide a place where researchers can contribute their components by taking advantage of the component schema as a standardized way to define them. Also, by seizing this schema it is possible to develop a service that generates packages that can seamlessly be integrated in the sensing client (i.e. mobile phone sensing application) without the need of updating it each time a new component is defined. All of this without requiring the user to have high technical skills or deep knowledge of the internal functioning of the proposed tool.*

### Technical implementation of the proposed approach

The proposed component-based model was implemented in the InCense mobile sensing tool using diverse technologies (InCense code repository: <https://github.com/incense-platform>). We used the PostgreSQL





**Figure 5.** Architectural model of the InCense tool before the implementation of the component-based model.

relational database for storing the information of components since it provides native support for JSON data types. It was also decided to use the REST architectural style<sup>32</sup> implemented with the Spring framework (<https://spring.io/>) to offer web services (using Apache Tomcat (<http://tomcat.apache.org/>) as the web server) to define components. Web forms were created and served with Apache Tomcat so that research have an interface that facilitates the definition of components. Figure 4 shows the implementation of the component-based approach. InCense was chosen because it manages on-device data processing by design, we had access to its source code, and we had means to interact with a person who worked directly with its design and development. A representation of the architecture of the InCense tool before the component-based model was implemented is shown in Figure 5.

### Web user interface for component development

A web user interface was developed to reduce the need for in-depth knowledge by users about the InCense tool when developing new components (see Figure 6). Still, when adding new components, some Java programming knowledge is needed. This user interface offers web forms to define a component and its elements, namely, sensors and data types (whether input or output). These web forms allow users to fill out the information about the components, the elements that specify the data types that a component utilizes, and the logic/algorithm employed by the component to execute its processing. The logic has to be programmed in Java, since it is the programming language used by the Android operating system (OS), which is the platform on which InCense builds. Developers may consider that libraries from the Android OS and Java Software Development Kit (SDK) will be available for them to use.

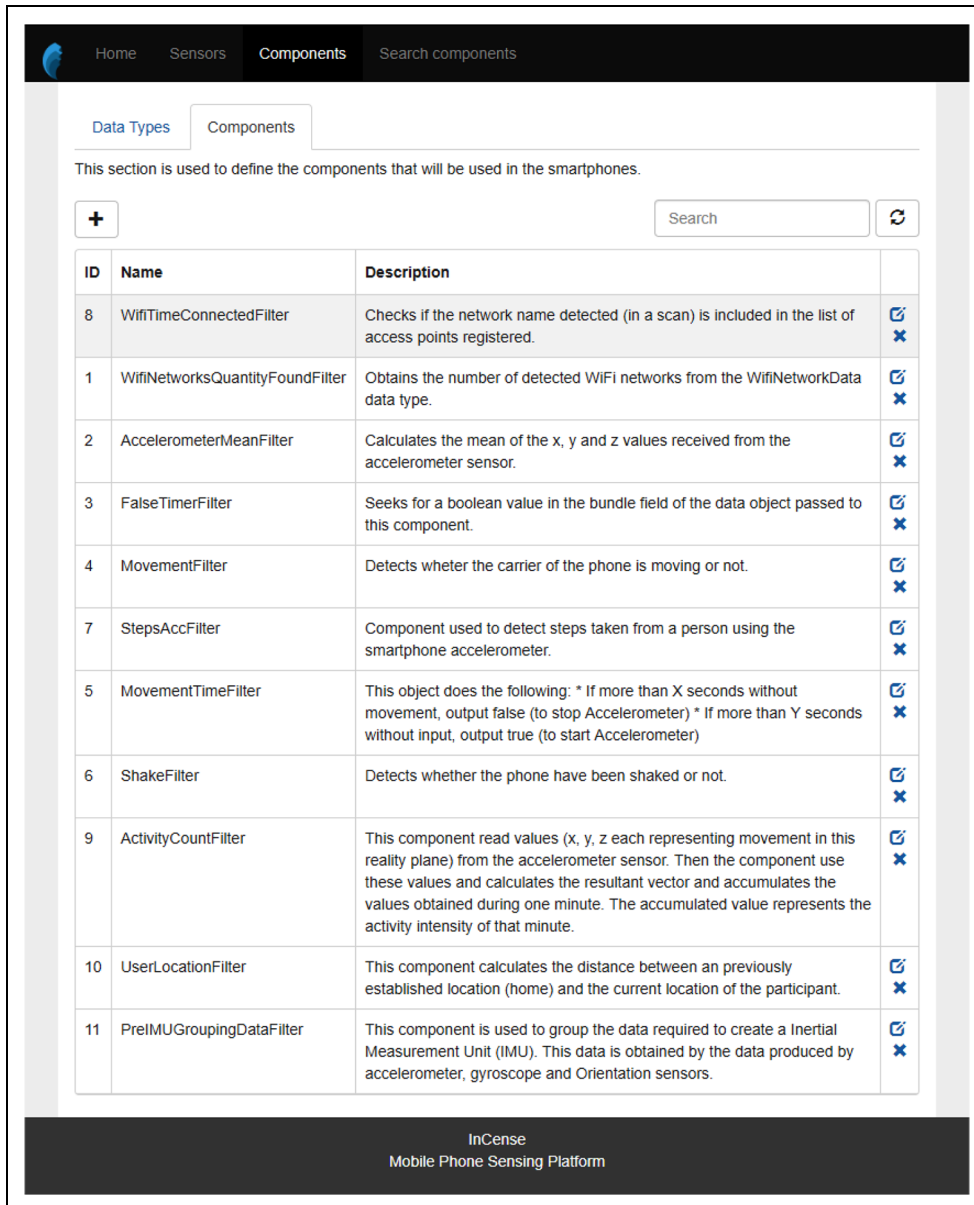
The web user interface transforms the data provided by the user with the web form to the component schema (see Figure 3) and sends it to the RESTful web services so that it can be stored in the repository. Figure 7 shows the web form to provide the processing logic of the component. Since components employ user-defined data as their output data, developers must ensure to return the expected data type. As for the input data, components can receive a sensor's output or another component's output. In any case, the function executing the processing must receive the expected input data type.

There is also a web user interface that allows searching components in the repository. This interface prompts for keywords as a search criterion that will be matched against the component description data in order to return relevant components for the user.

### Implemented RESTful web services

The web interface utilizes Web services for components to be viewed, defined, and/or searched. Web services are also used by the client (i.e. mobile phone sensing application) in order to request the package that contains the application in bits format, which is compatible with the Android OS and InCense. The operations that the RESTful web services offer are stated below.

1. *Sensor definition.* This is a way to provide the user with information about the sensors that are supported by the InCense client and the data type(s) that these offer. These services offer means to create, update, delete, and list sensors. Since this service is only informative and depends on what is supported by the InCense client, it should be used only by the administrator of InCense. The endpoint for this is in



**Figure 6.** Graphical user interface of the main screen.

`http://InCenseDomain/sensors` with the POST, PUT, DELETE, and GET methods implemented for creating, updating, obtaining, removing, and obtaining data regarding the logical sensors implemented. To obtain information related to a particular sensor or remove a particular sensor, the ID of the sensor must be included in `http://InCenseDomain/sensors/{id}` via the GET method.

2. *Data types definition.* Data types are necessary for users to establish the structure of the data that go in and out of the component. By knowing the input data type structure, the user knows the data that will be available to the processing.

By establishing the output data type, the user can let other users know what is available for them or any other component as a result, that is, after processing. The endpoint for this is in `http://InCenseDomain/datatypes` with the POST, PUT, DELETE, and GET methods implemented for creating, updating, obtaining, removing, and obtaining data regarding the logical data types implemented. To obtain information related to a particular data type or remove a particular data type, the ID of the data type must be included in `http://InCenseDomain/datatypes/{id}` via the GET method.

The screenshot shows a web form for defining a component. The form is titled "Components" and includes the following sections:

- Components:** A section for defining the component's basic information.
  - ID:** A text input field containing the value "8".
  - Name:** A text input field containing the value "WifiTimeConnectedFilter".
  - Description:** A text area containing the text "Checks if the network name detected (in a scan) is included in the list of access points registered."
  - Author:** A text input field.
  - Author's email:** A text input field containing the value "Ereter author email".
  - Author's institution:** A text input field.
  - Power consumption:** A text input field containing the value "1".
  - Sensor:** A dropdown menu with "WifiScanSensor" selected.
  - Input Data Type:** A dropdown menu with "WifiData" selected.
  - Output Data Type:** A dropdown menu with "WifiData" selected.
- Component Parameters:** A table with columns "Name", "Description", and "Default Value". The table is currently empty, with the text "No matching records found" displayed below it.
- Component Code:** A section for defining the component's code.
  - Class Name:** A text input field containing the value "WifiTimeConnectedFilter".
  - Required Imports:** A list of imports:
 

```
1 import android.util.Log;
2 import edu.incense.android.datask.Input;
```
  - Main Method Call:** A text input field containing the value "mainFunction".
  - Components logic:** A code editor containing the following Java code:
 

```
12  */
13  @Override
14  protected void compute() {
15      Data tempData;
16      for (Input i : Inputs) {
17          // Log.i(getClass().getName(), "asking for new data");
18          tempData = i.pullData();
19          if (tempData != null) {
20              computeSingleData(tempData);
21              // Log.i(getClass().getName(), "GOOD");
22          } else {
23              if (lastDataReceived != null) {
24                  lastDataReceiver.getTextView().putString(ATT_TIMECONNECTED,
25                      String.valueOf(getTimeConnected()));
26              }
27          }
28      }
29  }
```

At the bottom of the form, there are two buttons: "Save Component" and "Cancel".

**Figure 7.** Web form provided to users for component definition.

3. *Components definition.* These RESTful services allow users to define their components, that is, create, update, delete, or list components. The received web requests will be persisted in the database. In every allowed operation, the component schema is used to reflect what the component is all about and what it does. The endpoint for this is in <http://InCenseDomain/components> with the POST, PUT,

DELETE, and GET methods implemented for creating, updating, obtaining, removing, and obtaining data regarding the components implemented. To obtain information related to a particular component or remove a particular component, the ID of the component must be included in <http://InCenseDomain/components/{id}> via the GET method.

4. *Search service.* This service receives one or more keywords that are used as a search criterion that match one or more fields of the component description fields. The service returns a list of all the components that matched the search criteria. This is mainly useful to find one or more components that can be reused or used as a basis for new components.
5. *Components download service.* This service allows downloading a component in *.jar* format containing the component. This is primarily used by the client so that it can seamlessly obtain a component and hot plug it into the app at runtime. The endpoint is in <http://InCenseDomain/componentgenerator/{componentID}> with the GET method.

### Extended InCense mobile client

The client (i.e. the sensing mobile phone application) of InCense was extended so that it could effortlessly integrate the proposed model. As mentioned, the client is capable of interacting with the *component download service* and downloading the required components stated by the sensing campaign. Then, the client can integrate such components into the mobile phone application at runtime, thus only using the components that are needed for the campaign. This can be relevant if there are resource-deprived devices participating in the campaign. This seamless integration is executed automatically by the mobile phone application, thus the user or the participant does not need to intervene in any way. This functionality avoids redeployment of the application (e.g. rebuild and install on the phone) and use only what is needed on demand.

### Model implemented in InCense

As a result of the model implementation, now the InCense tool has a new interface for users to manage components and provide services that allow defining, searching, and downloading components. Also, the mobile phone client uses those services in order to provide on-device processing by using only what is needed when is needed. Figure 8 shows the outline of the InCense architecture with the component-based model implemented.

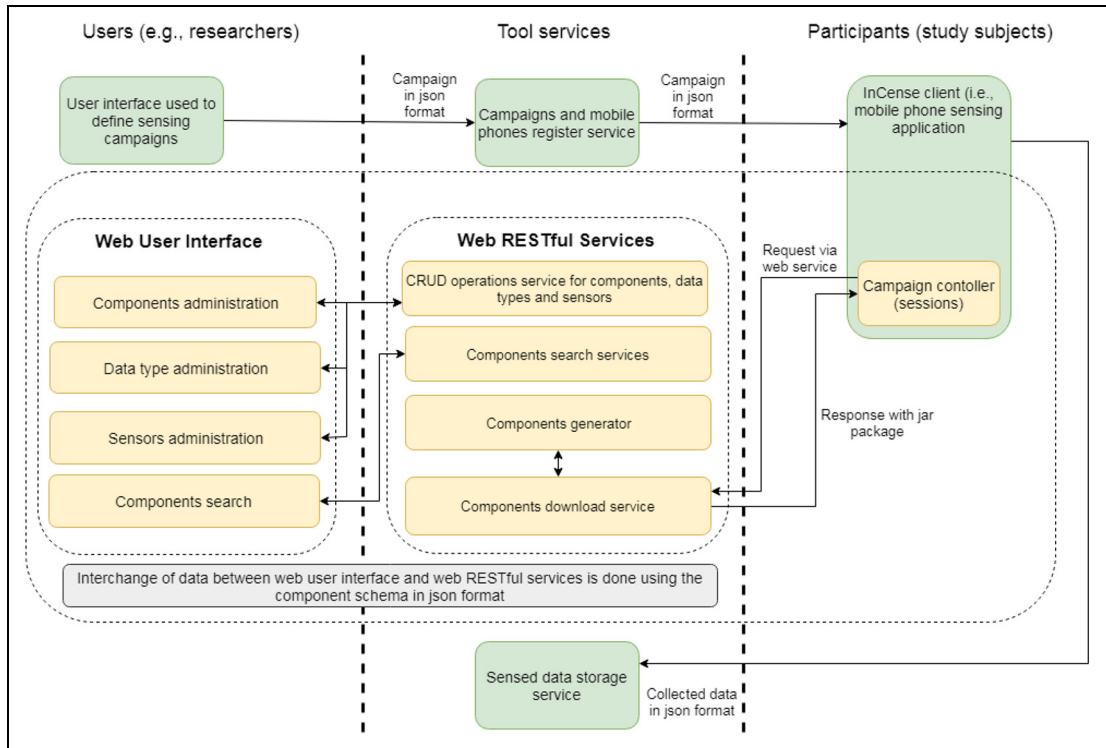


Figure 8. InCense tool with the component-based model implemented.

## Evaluation

We performed an evaluation mainly aimed at determining whether there is a significant improvement in terms of user effectiveness (i.e. new data processing components work correctly) and user efficiency (i.e. they can be developed in a short time) in the definition of components. We do so by contrasting the development of components using the proposed component-based model versus the conventional approach (i.e. making modifications directly in the source code).

## Participants

We invited 24 software engineering students in their fourth year who were mainly Java developers but were unfamiliar with the InCense source code. Sixteen of them accepted the invitation. The 16 participants were randomly allocated in two groups: a control group (CG) and an experimental group (EG). Four persons of the CG did not attend to all the sessions of the experiment. All participants ranged between 21 and 25 years old, with an average age of 22 years. No incentive was given to any of the participants.

## Procedure and description of task

Our participants were invited to participate in five sessions at a university laboratory equipped with

Integrated Development Environments (IDEs) and all tools needed for development. During the first session, depending on the group, a presentation was provided about InCense, in which we described the motivation for such a tool, the architecture, the organization of modules, web services implemented, structures of data types, and filter structures. Also, we provided a brief description of the study's purpose and the expected deliverables. Afterward, questions from participants were answered. We also provided each participant with a three-page cheat sheet containing that information. No formal technical documentation was provided.

We asked participants to develop a new component to compute the distance of the user's current location and a fixed point (e.g. home). This component was inspired on the concept of an individual's life space,<sup>33</sup> which basically classifies an individual based on the mobility of the user within an urban area. We provided the formula for computing the distance using math notation and JavaScript source code. We also showed participants an alternate method they could utilize using the Android's Location class and the distanceBetween method. In any case, the inputs and the type of outputs required were also provided. We also provided the names of the deliverables: component's name for the EG, and class name for the CG. We proposed this particular task (see Pseudocode 1) as it turns to be feasible, technically not challenging, and

**Pseudocode 1** User's location component

---

```

Set value of the homeLongitude variable to -109.44483164
Set the value of the homeLatitude variable to 27.06568788
Function to read the homeLongitude value
Function to set the value of homeLongitude
Function to read the homeLatitude value
Function to set the value of homeLatitude
Function calculateLocation (GPS location data as input) {
    Compute distance between two locations (homeLatitude,
    homeLongitude, and GPS location data)
    Return the distance results
}

```

---

GPS: global positioning system.

above all simple enough so that we can derive results from the approach used rather than the task itself.

### Hypotheses and variables

Our independent and dependent variables were as follows:

- Independent variable
  - The method used to develop the task: the use of the conventional approach versus the proposed component-based approach.
- Dependent variables
  - User efficiency: the time used to complete the task, that is, completion time.
  - User effectiveness: the developed component behaves as expected, that is, does it work as it should?
  - Ease of use: participants' perception regarding the ease of use of the approach.
  - Usefulness: participants' perception regarding usefulness of the approach.

Our main hypotheses were as follows:

- H1: More participants will be able to complete the task with the component-based approach than with the conventional approach.
- H2: Participants will be able to complete the task in less time with the component-based approach than with the conventional approach.
- H3: Ratings about ease of use will be higher in participants who used the component-based approach than with the conventional approach.
- H4: Ratings about usefulness will be higher in participants who used the component-based approach than with the conventional approach.

### Experimental design

The participants were randomly allocated in two groups (between-groups design) with the purpose of comparing the effectiveness and efficiency. The same task was assigned to both groups. All participants were assigned a PC for the duration of the experiment. We asked participants from the CG to complete this task utilizing the source code of InCense with the Eclipse IDE, a tool they were familiar with. That is, the members of the CG were asked to write the code from the new component and integrate it into the source code of InCense, and thus they had to get familiar with the source code of the platform before adding lines of code. On the contrary, the EG had to complete the task utilizing the web user interface of the implementation of the proposed approach in this work. The EG used no IDE. Both groups received an introductory presentation about InCense and its purpose.

We allocated participants 10 h to complete the task, divided into 5 days (2 h per session). After completing the task or the time was over, they were asked to answer a survey based on the Technology Acceptance Model (TAM).<sup>34</sup> This instrument consists of 12 items grouped into two constructs: one for evaluating the participants' perceived usefulness and another for evaluating the participants' perceived ease of use. The use of the TAM for evaluating user perception on potential future usage is commonplace in interactive systems. Although it may not be enough for certain application domains, it is convenient when evaluating systems that are difficult, expensive, or complex to evaluate in real working applications. In this case, comparing the actual use of both approaches in a real working sensing campaign can be difficult as it would need several research groups using and implementing new components.

The instrument featured assertions about either tool used, for example, "Using InCense could enable to create components rapidly." The answers to such items were 7-point Likert-type scale (1 = extremely likely, 7 = extremely unlikely), thus answers closer to 1 were better for the purpose of this study.

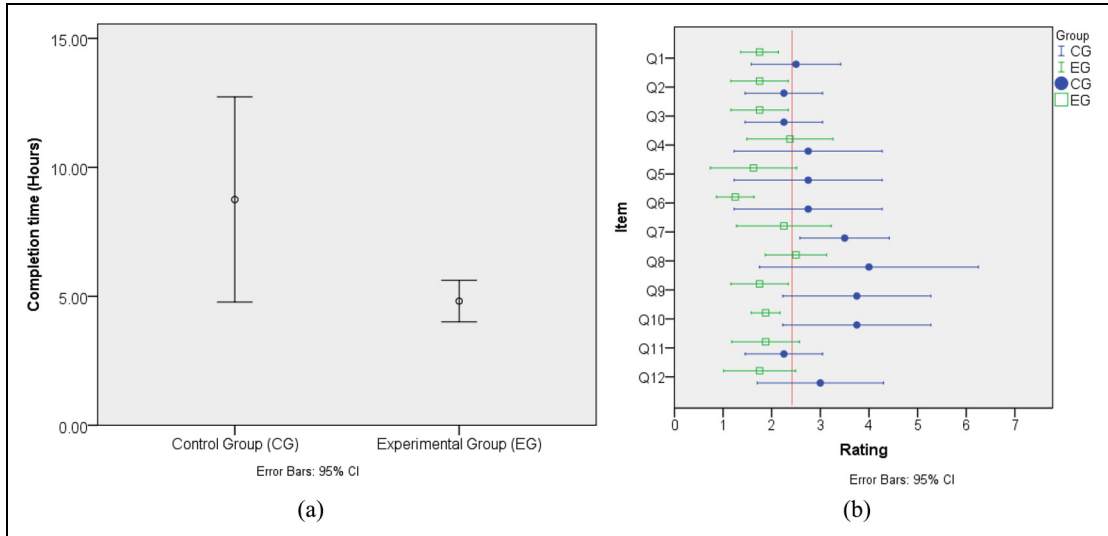
### Evaluation results

We excluded from the analysis all data related to the four participants from the control group who did not attend the five allocated sessions. CG in average utilized 8.75 h (standard deviation (SD) = 2.50), whereas the EG completed the task in 4.81 h (SD = 0.96). The difference between the mean values was 3.56 h, which was statistically significant ( $t = 3.040$ ,  $df = 3.452$ ,  $p = 0.047$ ; see Figure 9(a)).

**Table 2.** Participants' answers to the TAM instrument (1 = best, 7 = worst).

Participant's code	Time (hours)	Complete task?	Working component?	Usefulness items			Ease of use items										
				1	2	3	4	5	6	7	8	9	10	11	12		
<b>Control group—CG (N = 4)</b>																	
sm03	10	No	No	2	3	3	2	2	2	3	3	3	3	3	2	2	3
sm06	5	No	No	2	2	2	4	2	4	4	4	5	5	5	2	2	2
sm07	10	No	No	3	2	2	3	4	2	4	4	4	4	4	3	3	3
sm08	10	No	No	3	2	2	2	3	3	3	3	3	3	3	2	2	4
Mean	8.8	No = 4, Yes = 0	No = 4, Yes = 0	2.5	2.3	2.3	2.8	2.8	2.8	2.8	3.5	4.0	3.8	3.8	2.3	2.3	3.0
<b>Experimental group—EG (N = 8)</b>																	
cm01	4	Yes	Yes	2	2	2	3	2	2	2	4	2	2	2	1	1	3
cm02	5.5	Yes	Yes	1	2	2	1	1	1	1	1	2	2	2	2	2	1
cm03	5.5	Yes	No	2	1	3	3	4	1	1	1	2	1	1	2	2	1
cm04	5.5	Yes	Yes	2	1	2	2	1	2	2	2	2	1	2	1	1	1
cm05	5.5	Yes	No	2	3	2	3	1	1	1	1	3	1	2	3	1	1
cm06	3.5	Yes	Yes	2	2	1	4	1	1	1	3	4	2	2	3	2	2
cm07	5.5	Yes	No	2	2	1	2	2	1	3	3	2	2	2	1	2	2
cm09	3.5	Yes	Yes	1	1	1	1	1	1	1	3	3	3	2	2	3	3
Mean	4.8	No = 0, Yes = 8	No = 3, Yes = 5	1.8	1.8	1.8	2.4	1.6	1.3	1.3	2.3	2.5	1.8	1.9	1.9	1.8	1.8

TAM: Technology Acceptance Model.



**Figure 9.** Comparison between the CG and EG in terms of (a) completion time and (b) developers' ratings in terms of Usefulness and Ease of use (vertical line = overall mean rating).

None of the participants from the CG finished the assigned task. A chi-square analysis between the groups and the effectiveness showed that the members of the CG were statistically more effective ( $\chi^2(1) = 12.0$ ,  $p = 0.001$ ), meaning that there is a strong link between the group participants belonged to and their effectiveness.

On the contrary, all EG participants reported having completed the task and invested in average 4.8 h (SD = 0.96) to complete it. From the eight participants of the EG, five (62%) completed a fully functional component whereas three (38%) participants completed a component that did not work correctly. There was a common error from all three participants who did not yield a working component: they provided the parameters in the wrong order to the function that calculates the distance.

As for the TAM survey (see Table 2 and Figure 9(b)), the CG participants had an average perception rating of usefulness of 2.5, which could indicate that utilizing the source code can be useful, whereas for the EG the mean rating was 1.6. Regarding the ease of use perception, CG participants had also an average rating of 3.4, whereas for the EG it was 2.29. These results suggest that the use of the implementation of the component-based model might be easier and more efficient than its conventional counterpart.

### Limitations of the evaluation

Some of the limitations of this study design are as follows. First, as the study lasted for five successive days,

the activities carried out by participants between sessions were not controlled, that is, some students could have discussed the problems with others or look for information online. Second, we did not provide a documentation of the source code in either approach (conventional and component-based approach). Finally, this type of study designs can benefit from larger sample sizes.

### Conclusion and future work

In this work, we proposed a mobile sensing approach through the use of customized processing routines called components. By using the proposed approach, we aim to reduce the learning curve of users, particularly non-STEM, to implement new components, promote code reusability, and improve software maintenance and deployment.

The implementation of the component-based approach is composed of three main elements: (1) a web user interface to define components; (2) RESTful web services that support the web user interface, which allows generating and downloading components into the mobile phones; and (3) an extended mobile phone sensing application that leverages on the RESTful web services to utilize components on demand and on runtime in a transparent fashion to the user.

A usage and acceptance evaluation was performed, where participants were asked to perform a task regarding the creation of a component and asked them for their perception. This latter evaluation showed that the proposed approach could be more effective and



efficient than the conventional approach for deploying sensing campaigns. Future work in this regard includes not only analyzing the participants' perceptions toward the tool but also the quality of the source code generated.

There are several desirable features that may improve this work: (1) thus far, the component schema only considers one data type as input, hence having a component that supports several data types as input could be a feature that helps data processing; and (2) context-aware components could help saving resources (e.g. storage, battery, and processing), in other words, performing the processing of data only in designated places (e.g. home), or taking into account variable device resources such as battery. Finally, future work includes testing this platform in real working sensing campaigns to better understand how this platform can serve the needs of researchers in this area.


### Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partially funded by the National Council for Science and Technology (CONACYT) in Mexico through a scholarship provided to I.R.F. Also, this work was partially funded by the Instituto Tecnológico de Sonora (ITSON) through the PROFAPI program.

### ORCID iD

Luis A Castro  <https://orcid.org/0000-0002-1196-4919>

### References

- Ganti RK, Ye F and Lei H. Mobile crowdsensing: current state and future challenges. *IEEE Commun Mag* 2011; 49: 32–39.
- Birnbaum MH. Human research and data collection via the Internet. *Ann Rev Psychol* 2004; 55: 803–832.
- Konsolakis K, Hermens H, Villalonga C, et al. *Human behaviour analysis through smartphones* (Proceedings). Basel: Multidisciplinary Digital Publishing Institute, 2018, p.1243.
- Shoaib M, Bosch S, Incel O, et al. A survey of online activity recognition using mobile phones. *Sensors* 2015; 15: 2059–2085.
- Lane ND, Miluzzo E, Lu H, et al. A survey of mobile phone sensing. *IEEE Commun Mag* 2010; 48: 140–150.
- Froehlich J, Chen MY, Consolvo S, et al. MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones. In: *Proceedings of the 5th international conference on mobile systems, applications and services*, San Juan, PR, 11–14 June 2007. New York: ACM.
- Hicks J, Ramanathan N, Kim D, et al. AndWellness: an open mobile system for activity and experience sampling. In: *Proceedings of the wireless health 2010*, San Diego, CA, 5–7 October 2010. New York: ACM.
- funf. *funf Open Sensing Framework*, 2016, <http://www.funf.org/>
- Perez M, Castro LA and Favela J. InCense: a research kit to facilitate behavioral data gathering from populations of mobile phone users. In: *Proceedings of the 5th international symposium of ubiquitous computing and ambient intelligence*, Riviera Maya, Mexico, 5–9 December 2011. Ciudad Obregón, Mexico: InCense.
- Banos O, Villalonga C, Garcia R, et al. Design, implementation and validation of a novel open framework for agile development of mobile health applications. *Biomed Eng Online* 2015; 14: S6.
- Ferreira D, Kostakos V and Dey AK. AWARE: mobile context instrumentation framework. *Front ICT* 2015; 2: 6.
- Heineman GT and Councill WT. *Component-based software engineering: putting the pieces together*. 1st ed. Boston, MA: Addison-Wesley, 2001, p.864.
- Vale T, Crnkovic I, De Almeida ES, et al. Twenty-eight years of component-based software engineering. *J Syst Softw* 2016; 111: 128–148.
- Heisele B, Ho P, Wu J, et al. Face recognition: component-based versus global approaches. *Comput Vis Image Understand* 2003; 91: 6–21.
- Paraiso F, Merle P and Seinturier L. soCloud: a service-oriented component-based PaaS for managing portability, provisioning, elasticity, and high availability across multiple clouds. *Computing* 2016; 98: 539–565.
- Khan WZ, Xiang Y, Aalsalem MY, et al. Mobile phone sensing systems: a survey. *IEEE Commun Surv Tut* 2013; 15: 402–427.
- Consolvo S, McDonald DW, Toscos T, et al. Activity sensing in the wild: a field trial of ubifit garden. In: *Proceedings of the SIGCHI conference on human factors in computing systems*, Florence, 5–10 April 2008, pp.1797–1806. New York: ACM.
- Lane ND, Lin M, Mohammad M, et al. BeWell: sensing sleep, physical activities and social interactions to promote wellbeing. *Mobile Netw Appl* 2014; 19: 345–359.
- Sendra S, Granell E, Lloret J, et al. Smart collaborative mobile system for taking care of disabled and elderly people. *Mobile Netw Appl* 2014; 19: 287–302.
- Castro LA, Favela J, Quintana E, et al. Behavioral data gathering for assessing functional status and health in older adults using mobile phones. *Pers Ubiquit Comput* 2015; 19: 379–391.
- Wahle F, Kowatsch T, Fleisch E, et al. Mobile sensing and support for people with depression: a pilot trial in the wild. *JMIR Mhealth and Uhealth* 2016; 4: e111.
- Ben-Zeev D, Wang R, Abdullah S, et al. Mobile behavioral sensing for outpatients and inpatients with schizophrenia. *Psychiatr Serv* 2015; 67: 558–561.
- Hernández N, Arnrich B, Favela J, et al. mk-sense: an extensible platform to conduct multi-institutional mobile sensing campaigns. In: *Proceedings of the ubiquitous computing and ambient intelligence: 10th international*

- conference—part I (eds CR García, P Caballero-Gil, M Burmester, et al.), San Bartolomé de Tirajana, 29 November–2 December 2016, pp.207–216. Cham: Springer.
24. You CW, Lane ND, Chen F, et al. CarSafe app: alerting drowsy and distracted drivers using dual cameras on smartphones. In: *Proceedings of the 11th international conference on mobile systems, applications and services*, Taipei, Taiwan, 25–28 June 2013. New York: ACM.
  25. Mohan P, Padmanabhan VN and Ramjee R. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In: *Proceedings of the 6th ACM conference on embedded network sensor systems*, Raleigh, NC, 5–7 November 2008, pp.323–336. New York: ACM.
  26. Eriksson J, Girod L, Hull B, et al. The pothole patrol: using a mobile sensor network for road surface monitoring. In: *Proceedings of the 6th international conference on mobile systems, applications, and services*, Breckenridge, CO, 17–20 June 2008, pp.29–39. New York: ACM.
  27. Rana RK, Chou CT, Kanhere SS, et al. Ear-phone: an end-to-end participatory urban noise mapping system. In: *Proceedings of the 9th ACM/IEEE international conference on information processing in sensor networks*, Stockholm, 12–16 April 2010. New York: ACM.
  28. Ruge L, Altakrouri B and Schrader A. SoundOfTheCity: continuous noise monitoring for a healthy city. In: *Proceedings of the IEEE international conference on pervasive computing and communications workshops*, San Diego, CA, 22 March 2013, pp.670–675. New York: IEEE.
  29. Ballesteros J, Rahman M, Carbunar B, et al. Safe cities. A participatory sensing approach. In: *Proceedings of the 37th annual IEEE conference on local computer networks*, 2012, pp.626–634. New York: IEEE, <http://users.cis.fiu.edu/~carbunar/safety.pdf>
  30. Ballesteros J, Carbunar B, Rahman M, et al. Towards safe cities: a mobile and social networking approach. *IEEE Trans Parall Distrib Syst* 2014; 25: 2451–2462.
  31. Félix IR, Castro LA, Rodríguez LF, et al. Component-based model for on-device pre-processing in mobile phone sensing campaigns. In: *Proceedings of the ubiquitous computing and ambient intelligence: 10th international conference—part I* (eds CR García, P Caballero-Gil, M Burmester, et al.), San Bartolomé de Tirajana, 29 November–2 December 2016, pp.207–216. Cham: Springer.
  32. Fielding RT and Taylor RN. Principled design of the modern web architecture. *ACM Trans Internet Technol* 2002; 2: 115–150.
  33. Xue QL, Fried LP, Glass TA, et al. Life-space constriction, development of frailty, and the competing risk of mortality: the women’s health and aging study I. *Am J Epidemiol* 2008; 167: 240–248.
  34. Davis FD. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quart* 1989; 13: 319–340.