Lowering the Technical Threshold for Organizers to Create and Deliver Mobile Crowd Sensing Applications

Wang, J., Wang, Y. & He, Y.

Original citation:
Wang, J, Wang, Y & He, Y 2015, 'Lowering the Technical Threshold for Organizers to Create and Deliver Mobile Crowd Sensing Applications', International Journal of Distributed Sensor Networks, vol. 11, no. 11, 721647. https://dx.doi.org/10.1155/2015/721647

DOI 10.1155/2015/721647
ISSN 1550-1329
ESSN 1550-1477

Publisher: SAGE Publications

Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 721647, 15 pages
http://dx.doi.org/10.1155/2015/721647

Research Article

Lowering the Technical Threshold for Organizers to Create and Deliver Mobile Crowd Sensing Applications

Jiangtao Wang,[1,2] Yasha Wang,[1,3] and Yuanduo He[1,2]

[1]*Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, China*
[2]*School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China*
[3]*National Engineering Research Center of Software Engineering, Peking University, Beijing 100871, China*

Correspondence should be addressed to Yasha Wang; wangys@sei.pku.edu.cn

Received 6 February 2015; Accepted 28 April 2015

Academic Editor: Dakshnamoorthy Manivannan

In order to lower the technical threshold for creating and delivering Mobile Crowd Sensing Applications (MCSAs), several frameworks or toolkits have been developed, but they either fail to provide general support or still require relatively professional software development skills. Besides, the application delivery to the participants is not precise enough, because they only consider the constraints predefined by the organizers without taking the participant-side factors into account. In this paper, we propose a prototype toolkit for the organizers without software development skills to build MCSA in a rather quick and simple way. First, in terms of the application creation, it enables organizers to build MCSA by just doing some simple settings, which totally eliminates the requirement of programming skills. Second, in terms of the application delivery, it selects participants who are more likely to accept the created applications by mining their participation history. Finally, we demonstrate the expressiveness and usability for the application creation and evaluate the effectiveness of the willingness-based participant selection algorithm for the application delivery.

1. Introduction

Nowadays, mobile phones have evolved from merely being phones into full-fledged computing, sensing, and communication devices. These technology advances coupled with the sheer number of user-companioned mobile phones and their inherent mobility enables a new and fast-growing sensing paradigm, which is referred to as the *Mobile Crowd Sensing (MCS)* [1]. MCS encourage citizens to participate in certain campaigns to collect and share sensing data from surroundings by their mobile phones.

There are some examples of MCS in the following, which are used in the paper to better elaborate the research issues and corresponding solutions.

Urban Infrastructure Monitoring. It is very difficult for city administrators to monitor the masses of widely distributed urban infrastructures in the city, such as the manhole covers and street lights. Therefore, city administrators can launch a MCS campaign to recruit citizens for taking and uploading geo-tagged photos of the missing or broken infrastructures.

Spot Reporter. A journalist is under deadline pressure to write an article about a polluted river and its impact on nearby communities. He recruits volunteers with smartphones to send pictures of the river, video interviews with the local residents, and reports on the impact of the polluted water on their daily life.

Crime Investigation. Security officials are investigating a bomb attack in a building MCSRried out by terrorists and would like to find out people who were at the scene of the outbreak when the attack occurred. They request volunteers, who may have taken pictures during the event, to send snapshots of faces in the building.

Restaurant Recommendation. A tourist would like to find a nearby restaurant that meets certain requirements, so that he/she asks people who are familiar with this area to give

some advice. Hence he/she initiates a task by expressing his requirements for the restaurant, and people who are familiar with this area give suggestions.

MCS mainly has two stakeholders: *the organizers and the participants*. The organizers are those who would like to initiate and manage the MCS campaign, while the participants are those taking part in MCS campaigns to contribute the sensing data. In the urban infrastructure monitoring scenario, the city administrators launching the photo-taking campaign are organizers, while citizens taking and uploading relevant photos about missing or broken urban infrastructures are the participants.

The accomplishment of MCS campaigns needs the support of MCS applications (MCSAs) [2]. Since MCSAs have strong demand in personalization and short development cycle, they are more suitable to be developed by organizers rather than professional software developers. However, the development of MCSA requires complex programming skill, such as sensor access and sensing data process. Therefore, it is difficult for the organizers to develop MCSA due to their lack of programming skill.

In order to lower the technical threshold for creating MCSA, many frameworks and toolkits have been developed. Although reducing the requirements in software development skill to some degree, existing frameworks or toolkits still have some drawbacks. *In terms of application creation*, some are specific to their respective domains and fail to provide general support for MCS, while some others still have relatively high requirements for the software development skills. *In terms of application delivery*, they do not fully address how to deliver the created applications to the appropriate participants.

Therefore, to overcome above problems of existing works, the objective of this paper is to propose a prototype toolkit for helping organizers to do the following.

(i) Create MCSAs in a quick and simple way.

(ii) Deliver the created applications to the appropriate participants by taking participant-side factors into account.

However, fulfilling this objective is not straightforward, which entails following challenges.

First, the tradeoff between the expressiveness and the ease of use must be carefully considered. On the one hand, it should be able to support developing rich kinds of MCSA. Thus lots of functionalities must be integrated into the tool, which might make its usage more complex. On the other hand, since it is designed for the organizers, high complexity will pose high burden on them in terms of learning and using.

Second, with the increasing popularity of MCS, the participants may become overwhelmed when they receive a large number of recommended MCSA. Existing toolkits enable the organizers to define constraints for the participant selection (e.g., participant's location and device's sensing capabilities). However, these constraints are all from the perspective of organizes, and none of them consider the participant-side factors. In fact, the participants decide whether to accept and actually undertake a recommended application based on many participant-side factors. For example, whether the reward is attractive enough, whether the participant is interested for the application, and if accomplishing the MCS campaign may not cause too much privacy violation, and so forth. However, selecting participants who are willing to undertake a task is not easy. On the one hand, different factors could have different influence in deciding whether the participant is willing to undertake a task. It is very difficult for organizers to predict or learn these differences. On the other hand, even the same factor may have different impact on different participants, which is subjective and varies from one participant to another. For example, privacy preserving is more important than earning money for some participants, but it may be the opposite case for others.

With abovementioned objectives and challenges, this paper proposes a toolkit to reduce the technical threshold for organizers in the creation and delivery of MCSAs, and the main contributions of this paper can be summarized as follows.

(1) First, by compromising the expressiveness and the ease of use, we design and implement a prototype phone-side tool for the organizers to define their MCSAs in a quick and simple way.

(2) Second, we propose a willingness-based participant selection algorithm to deliver MCSAs more precisely by mining the participants' historical behaviors.

(3) Third, we demonstrate the expressiveness and usability for the application creation and evaluate the effectiveness of the willingness-based participant selection algorithm for the application delivery.

The rest of this paper is divided into 5 sections. Section 2 reviews related works. In Section 3, we will introduce the design and implementation of the toolkit; Section 4 describes the willingness-based participant selection algorithm. Section 5 is the evaluation and the demonstration of the toolkit. Finally, we conclude with directions for future work in Section 6.

2. Related Work

In this section, we review the related literature from two topics: tools support for organizers and the participation selection algorithm.

2.1. Supporting Tools for Organizers. There are several tools for the organizer to create MCS campaigns in specific domains. Epicollect [3] allows the creation of campaigns specific to epidemiology and ecology. Similarly, Project Noah [4] is a website for creating "missions" in which users can contribute images of wildlife. While both Project Noah and Epicollect eliminate the need for software developer skill, they are limited to specific domains and cannot provide general support for MCSA. Sensr [5] provides more generalized support by providing campaign organizers with a web interface for creating a campaign, which users then access via the mobile application. However, it is limited to camera data and text-entry input. All three of these tools fail to provide access

to many desirable sensors, including the accelerometer, and microphone.

There are some other MCS campaign creation tools which have been designed for general campaign creation, such as PRISM [15], Medusa [11], ohmage [16], Code in the Air (CITA) [17], Campaignr [18], and Open Data Kit [19]. Each of these provides high configurability of campaigns for the organizers, removes the user from the technical challenges of accessing phone sensors, and is robust to changes in the campaign. Though these tools reduce the software development technical threshold for organizers to some extent, they still require some programming capabilities, including knowledge about the format and ordering of a document, correct use of the programming language's syntax, or infrastructure knowledge.

MCSA creation tool that is the most similar to our toolkit is [20, 21]. In [20], MC Designer is specifically designed to lower the barrier for the organizer by providing an easy-to-use graphical user interface, which allows users to create a mobile application simply by supplying relevant MCS campaign parameters. However, MC Designer selects participants merely based on four optional constraints set by the organizer, including the gender, age range, ethnicity, and geographical region. In contrast, our toolkit further takes the participant-side factors into consideration to achieve a more precise delivery, thus reducing the overload for the participants to find their desired MCSA. The framework proposed in [21] also aims to help initiators of Mobile Crowd Sensing campaign define task. However, different from this paper [21], only targets at the application creation support without considering the participants selection based on their participation history.

2.2. Participant Selection Mechanisms. Though designed for professional developers rather than organizers without programming skills, some MCS mediation platforms also address the participant selection problem. Thus we still include them as related work as follows.

Anonysense [22] takes time, participant's location, and privacy setup into account when identifying participants. The work in [23] proposes an assignment policy to identify suitable participants based on their device's location, battery level, and the overall spatial coverage. CrowdRecruiter [24] and CrowdTasker [25] aim at minimizing incentive payments by selecting a small number of participants while still satisfying probabilistic coverage constraint. The work in [26, 27] develops a selection framework to enable organizers to identify well-suited participants for data collections based on geographic and temporal availability, transportation mode, and the coverage. Crowdlab [28] schedules the task based on location and battery resource budget. The work in [29] takes location as the only factor for the participant selection. Participant selection mechanism in above works is merely based on the constraints defined by the application developer, while this paper takes the participant-side factors into account. A QoI-aware energy-efficient participant selection framework has recently been proposed [30], which considers the QoI requirements, the energy consumption index, and the estimation of the collected amount of data. Literature [31] selects participants based on the data quality requirements, location, and budget constraints.

3. System Design

3.1. Requirements and Considerations. In order to reach the objective mentioned above, this prototype toolkit must provide assistance with specific considerations in following two aspects.

(1) MCS Task Creation Support. In the phone-side of organizers, there must be an easy-to-use application for them to create their MCS tasks. Once the organizer runs the application and clicks the button "create an application," an interface will appear to allow the organizer to define the MCS task by doing some simple settings. In this paper, we view a MCS task as a MCS application. After the task has been created, a virtual MCSA is considered to be generated. We call it "virtual" because at this time the task is defined and well-understood by our toolkit, but not in a running status.

(2) MCS Task Runtime Support. Once the organizer clicks the button "activate the application" in the phone-side app, the toolkit will start to provide runtime support for a MCS task, which mainly consists of following three aspects.

Frist, it must be able to select participants based on the settings defined by organizers in the task creation phase and push task to them. Hence, there must be an application in the participant's phone side, and it serves as an application store of MCSA. Once a task is pushed to a certain participant, his application will alter him that a new application is recommended. The participants can choose whether to accept it or not.

Second, if a participant accepts a task, the toolkit must provide assistant for him to collect sensing data and upload to the cloud. Energy consumption and the cost are important considerations. If the MCSAs are too energy-consuming or cost-consuming, their enthusiasm for collecting and sharing data will be reduced.

Therefore, the toolkit must provide mechanisms to make the data collection and sharing in a more energy and cost-efficient way.

Third, the management of MCS process is also very crucial. In fact, there are many factors to be considered to better manage the MCS. In this paper, we aim to handle following significant issues.

Intrusiveness. Participants do not want to be disturbed under certain circumstances. Therefore, if the potential participants are forwarded with too many MCSAs, the frequently bothering might reduce their interest for participation. Therefore, the tool must integrate certain mechanisms to reduce the intrusiveness on participants.

Trustworthiness. The toolkit must be able to identify malicious participants. For example, Tom developed his MCS application whose sensing task is to ask for recommendation of restaurants nearby with quiet environment and healthy food. In order to attract customers, some shopkeepers may

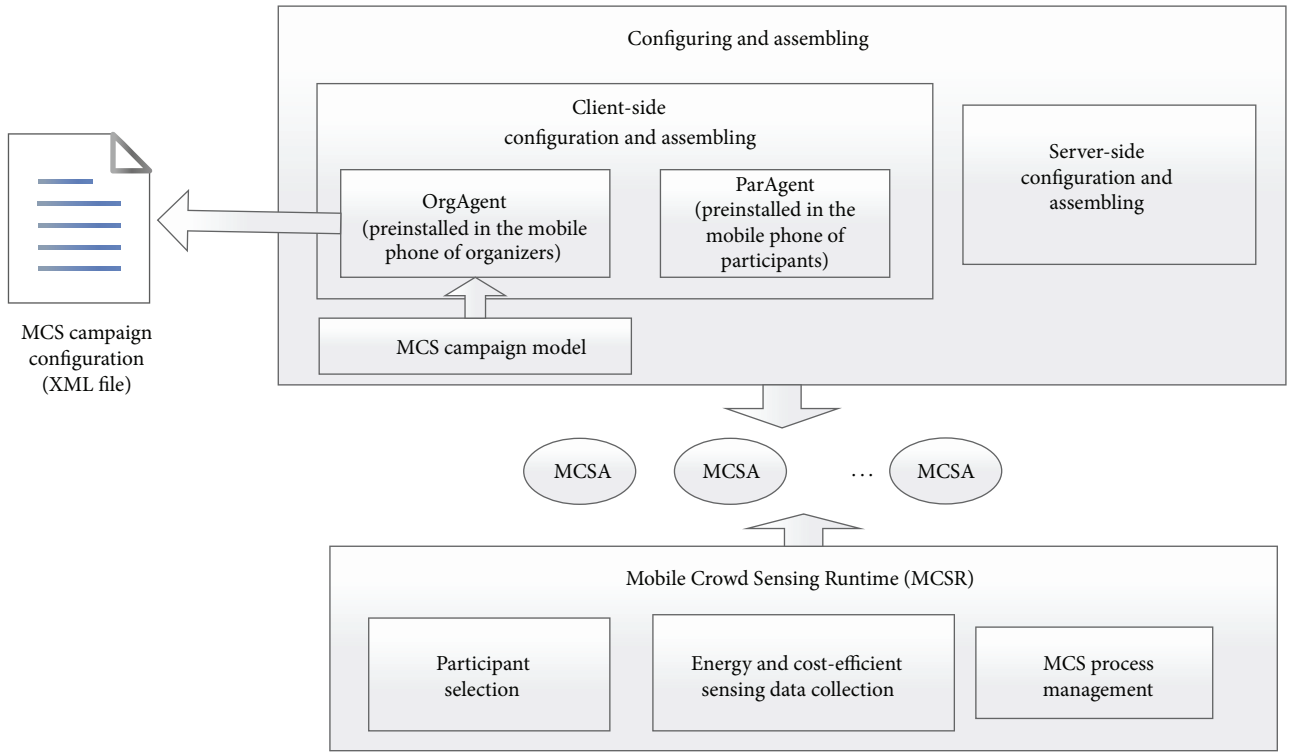


Figure 1: System design: an overview.

participate in the task to recommend their restaurants even if they are very noisy.

Privacy. Participants are also concerned about their privacy leak while collecting and sharing their sensing data. For example, people sometimes do not want to upload their location information. Therefore, when recommending the task, the toolkit must clearly tell the participants what information will be collected through the mobile phone and let them to decide whether accept it or not.

3.2. Toolkit Architecture: An Overview. In order to meet the abovementioned requirements, we design a prototype system (i.e., the toolkit), and Figure 1 provides the overview of its architecture.

The system primarily consists of two parts. One is the *configuring and assembling* for the MCSA and the other is a *runtime environment* (named *MCSR*) for the MCSA.

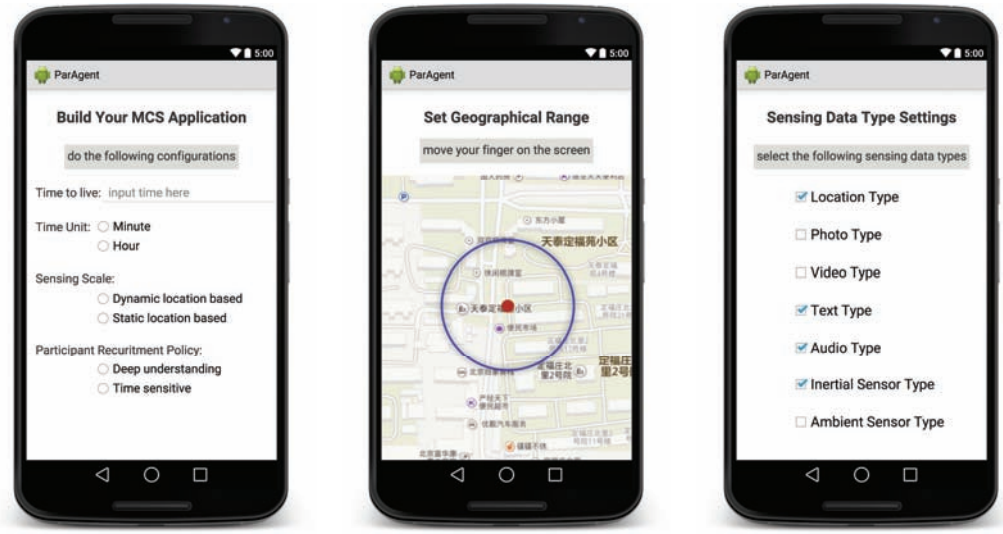
The configuring and assembling is for the organizers to define their expectations and constraints for the MCSA based on a predefined MCS campaign model. Then these expectations and constraints are modeled by our toolkit as a *MCS campaign configuration*. The generation of the MCSA includes the following two phases. First, the organizer uses the configuration interface of OrgAgent to do some settings (see Figure 3). Based on the built-in MCS campaign model, the settings are structured as a XML file named MCS campaign configuration. Second, by interpreting the MCS campaign configuration, the configuring and assembling module will configure the components to assemble MCSA. This process basically consists of three subprocesses at the same time. (1) The system configures the client-side layout of MCSA for participants. (2) OrgAgent generates the client-side of MCSA for organizer. (3) The server-side configuring and assembling module generate the server-side for MCSA (both for organizer and participant).

MCSR (Mobile Crowd Sensing Runtime) is to provide runtime environment, in which several predefined components are composed and configured at runtime to deal with different and important issues of MCSA. Frist, it selects participants and push task to them. Different from existing participant selection approach, our system considers not only the predefined constraints set by the organizer but the willingness of the participants to accept the task. Second, if a participant accepts a task, the ParAgent provide assistant for collecting sensing data and uploading to the cloud in an energy and cost-efficient manner. Third, it optimizes the MCS process management by taking intrusiveness, trustworthiness, and privacy into account.

Here we should note that Figure 1 describes the logical relationship between different modules, rather than the physical deployment. For example, though the main components of MCSR are deployed and running in the cloud, it also has components deployed in the phone-side of organizer and participant.

3.3. The Configuration and Assembling of MCSA

(1) MCS Campaign Model. It is a built-in model defining common parameters of MCSA. By assigning values for these



(a) (b) (c)

Figure 2: OrgAgent: configuring interface for the organizer.

parameters with an easy-to-use interface on the mobile phone, the organizers can define their expectations and constrains for the MCSA. Then it is used to generate the configuration file for each MCSA, based on which tailored mobile application serving the specific purposes can be created.

The MCS campaign model consists of following parameters.

Geographical Range. MCS campaigns usually require the participant's physical presence at a specific geographical range. In this paper, our toolkit enables the organizers to assign circular range by giving the center and radius in the map (see Figure 2(b)). In particular, it provides the organizers with two options to define the center. One is based on a fixed location and the other is based on a dynamic location, namely, binding to the organizer's current location sensed by his mobile phone. These two options are for different scenarios described in the introduction part. The fixed location is suitable in the scenarios in which the participants are selected in any given area defined by the organizer, such as the spot reporter and crime investigation. The dynamic location, on the other hand, is applied in the situations where the organizer launches a campaign nearby his current location, such as the restaurant recommendation example.

Shelf Life (SL). MCS campaigns have time limitation, which is referred to as the SL (shelf life). The organizers can assign a SL for MCS campaigns on the configuration interface (see Figure 2(a)). Over this time, the campaign is shut down since the sensing data has already become meaningless. For example, in the restaurant recommendation scenario, a tourist may set the SL to be a short period of time, say ten minutes. The setting of SL is optional, because in some scenarios such as the crime investigation, the organizers may not be able to decide how long the campaign should last until their goals have been accomplished. In such cases, MCSA generated by our toolkit will provide buttons to activate or deactivate the campaigns.

Participant Recruitment Mode. After specifying the geographical coverage, the tool has to recruit the most appropriate participants and deliver MCS task to them. There are two types of participant recruitment mode in terms of the geographical range. One is the *familiar* mode and the other is *presence* mode. In the *familiar* mode, the system selects participants who have a more comprehensive understanding about a certain community or district. For example, in the restaurant recommending case, a tourist wants to find an Italian restaurant within a certain geographical area. In this case, ideal participants are those who live in or always come to this region. The presence mode, on the contrary, requires the participants to provide the latest information about in a certain place. For example, a student wants to know how many people are in a Wal-Mart store now. In this case, ideal participants are those who are currently shopping in the Wal-Mart. Therefore, in order to make the toolkit distinguish between these two participant recruitment policies, the organizer can select either "familiar mode" or "presence mode" in the configuration interface (see Figure 2(a)). However, this setting is optional, since in some cases both these two types of participants are suitable candidates, such as the spot reporter example.

Sensing Capability. Since MCS requires the participant to collect sensing data through their mobile phone, it has constraint for the sensing capability of the participant's phone. The current version of our toolkit supports several types of sensing data, and they are photos, videos, texts, sounds, and accelerometer data. Relevant classes are derived from a base class *SenseDataType*, so that the sensing data type is easily to be extended if new requirements emerge in

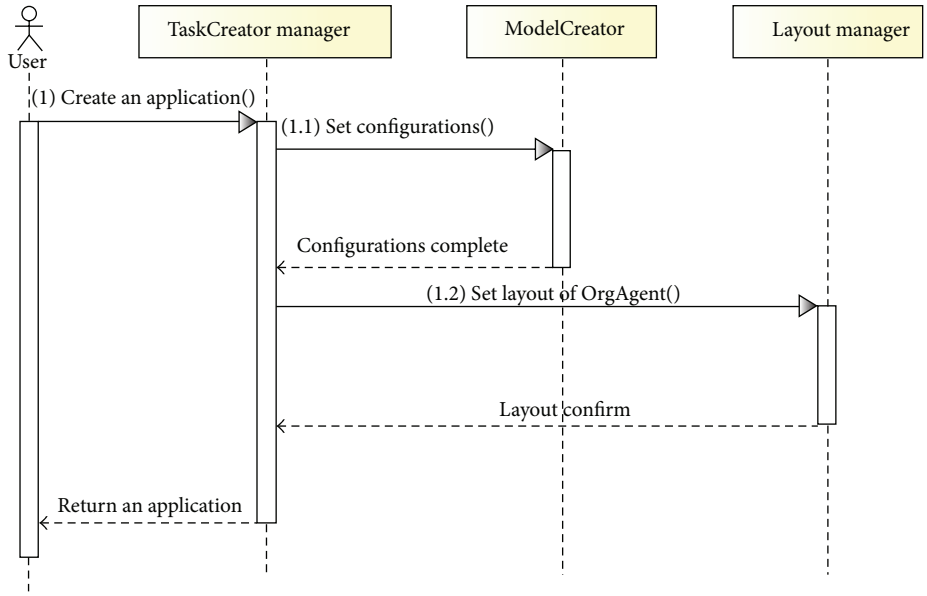


Figure 3: Sequence diagram for client-side configuring.

the future. The organizer has to select one or more of them in the configuration interface (see Figure 2(c)), based on which the toolkit generate different layout of user interfaces for the participants automatically.

Incentive. Since the participation of MCS may consume resources such as battery or cellular network flow, MCSA always provides some incentive (commonly money) to encourage participants. Therefore, the MCS campaign model also contains a parameter, the incentive, by which the organizer can define how much money he wants to give for each participant (in the current version, the unit is US dollars).

Assignments. This parameter is used to define how many participants the organizer want to recruit, and this value is used in the participation recruitment phase.

(2) Client-Side Configuring and Assembling. This module consists of two Android applications that people can download. The introduction page for the download is with two buttons for (create an application) and (campaign participation), corresponding to two subapplications preinstalled in the mobile phones of the organizers and participants, respectively. (a) The subapplication in the organizer's mobile phone is called the OrgAgent serving two purposes. One is to generate a configuration file based on the MCS campaign model by doing some simple settings and the other is to configure and assemble the MCSA in organizer's mobile phone for participant recruitment and data viewing. (b) The subapplication installed in the participants' mobile phone is called the ParAgent. It is responsible for configuring the MCSA in the participant's mobile phone and provides runtime support for the participants. Meanwhile, it serves as an application store of the MCSA. Both the OrgAgent and the ParAgent will predownload all interface components before the MCSA has been generated, such as map controller, buttons, scroll bars, and editing box. Figure 3 shows the sequence diagram, which demonstrates how the client-side of MCSA is created. First, when the organizer (user) creates an application, the object TaskCreator Manager sends a message to object ModelCreator, and then a configuring interface in Figure 2 is generated and displayed for the organizers to set parameters. After the settings are made, the Layout Manager shows a preview of the layout of MCSA. If the organizer confirms it, then the interface of application will be generated.

(3) Server-Side Configuring and Assembling. This module is responsible for selecting appropriate components from the component repository to assemble the server-side of MCSA. Some of these components will be composed and configured in the MCSR at runtime.

3.4. Mobile Crowd Sensing Runtime (MCSR). The second part is the runtime environment, named *MCSR (Mobile Crowd Sensing Runtime)*, in which reusable components are designed and implemented for the runtime support of MCSA. Due to the limit of space, we will only introduce functions of some main components without presenting the physical deployment structure of them. But please note that the runtime environment is deployed in both the phone side and the server-side.

3.4.1. Participant Selection. This component will select appropriate users and then deliver the generated MCSA to them. We adopt a two-phase selection process (see Figure 4) as follows.

(i) Phase 1: Location-Based Selection. The input of this phase is denoted as participant set PS[0] in Figure 4, which represents all the potential participants registered in our system. This phase selects participants who are geographically appropriate. Frist, the task interpreter will parse the MCS campaign

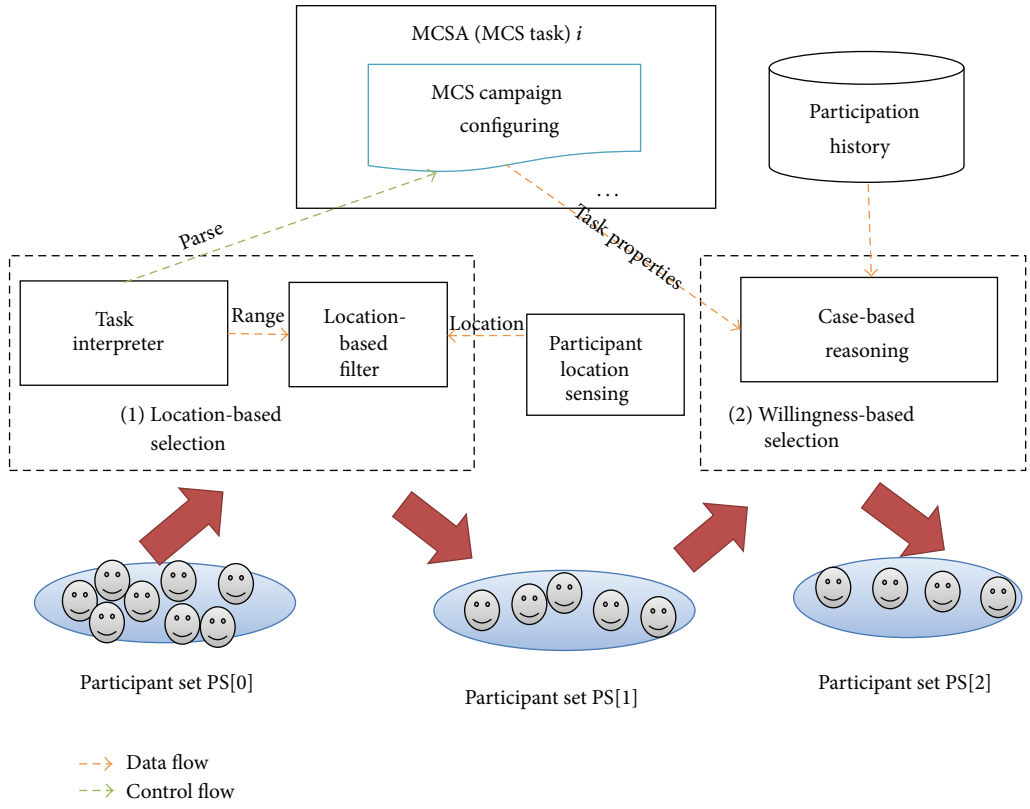


Figure 4: Two-phase participant selection process.

configuring file to get the predefined geographical range. Second, the selection performed by the location-based filter is on the basis of two factors set by the organizer. One is the geographical range and the other is the participation recruitment mode. For the "familiar mode," the participant selection is according to the regularity of the potential participants' historical location for finding persons familiar with this geographical coverage. For the "presence mode," the participant recruitment is according to potential participants' current location. The output of this phase is denoted as participant set PS[1] in Figure 4.

(ii) Phase 2: Willingness-Based Selection. After the location-based selection phase, if the number of selected candidates (the size of PS[1]) is still larger than the number of assignments defined by the organizer, the second phase (i.e., the willingness-based selection) will be started. This phase further selects those who are more likely to undertake a crowd sensing task by mining their participation history. The output of this phase is denoted as participant set PS[2] in Figure 4. To accomplish this goal, this paper proposes a case-based reasoning algorithm in Section 4.

3.4.2. Energy and Cost-Efficient Sensing Data Collection. Our tool supports several types of sensing data, and they are photos, videos, texts, accelerometer data, sound data, and location data. Therefore, the sensing data collection component should be responsible for acquiring them, which include several APIs correspondingly, that is *PhotoTaking*(), *VideoRecording*(), *TextObtaining*(), *AccCollection*(), *SoundRecord*(), and *GetLocData*(). To implement such interfaces to access sensing data is easy, but the energy consumption and the cost are important considerations. Therefore, our tool proposes some mechanisms to make the generated MCSA collect and share data in a more energy and cost-efficient way. First, the user's location information, such as Wi-Fi fingerprints, GPS, and Cell ID, is obtained from the localization modules in the phone. For energy-saving purpose, when the API *GetLocData*() is called, GPS data will not be recorded if Cell ID and Wi-Fi fingerprints can be captured and the localization accuracy can meet the requirement of the task. This is because continuous access to the GPS is very battery-consuming. Second, in some cases, the participants have collected sensing data but do not want to upload it immediately. For example, the participant's mobile phone cannot connect to free Wi-Fi and his monthly 3G traffic is used up, or his mobile phone's battery energy is not enough. Therefore, the MCSAs generated by our toolkit allow participants choose to cache the data in the phone and upload it as soon as he wants. Third, the tool also tries to reduce the energy consumption by sharing the GPS data with other MCSAs.

3.4.3. MCS Process Management. The runtime support also optimizes the MCS process by tasking the intrusiveness, trustworthiness, and privacy preserving into account, respectively.

Frist, in order to reduce the undesired intrusiveness, the toolkit proposes a mechanism to improve the usage experience of the participants. It sets the following default rules. First, do not disturb participants during some specific period of time. Second, do not disturb users who have recently received recommended MCSA as many times as their settings permit. Correspondingly, MCSAs generated by our tool have an interface where the parameters can be set by the participants, and the parameters are the time period that they do not want to be bothered and the maximum number of recommended applications. Second, inspired by the e-commerce system, a reputation-based approach is adopted to improve the trustworthiness of MCS participation. The organizer can give score for the contribution, based on which the participant's reputation is adjusted. The system can improve the participant selection by referring the reputation information overtime. Third, to preserve the privacy of participant, a notification will pop up in the ParAgent to clearly inform the participants of what information will be collected through the mobile phone. This step guarantees that the participants decide whether accept it or not after knowing all possible potential privacy leaks.

4. Willingness-Based Selection Algorithm

Willingness-based selection phase is to select participants who are more likely to accept a created MCSA by mining their participation history. To reach this goal, we design a case-based algorithm in this section.

There are many participant-side factors that may affect a participant's decision to undertake or decline a certain MCSA. However, the current implementation of our toolkit takes three factors (in Table 1) into consideration.

4.1. Problem Formulation. The problem is formulated as follows.

$\mathrm{PS}_1(T_h) = \{p_1, p_2, \ldots, p_N\}$ is the input, where wk_i ($i = 1, 2, \ldots, N$) is a participant satisfying the basid constraints of task T_h defined by the initiator. The goal is to find K ($K \leq N$) participants from $\mathrm{PS}_1(T_h)$ who have the highest likelihood of accepting the task T_h. The output is $\mathrm{PS}_2(T_h) = \{p_{q_1}, p_{q_2}, \ldots, p_{q_k}\}$, where K is desired number of assignments defined by the initiator.

4.2. The Algorithm. We first define the concept of historical case.

$\overrightarrow{\mathrm{PF}}_{i\theta} = (c_{1\theta}, c_{2\theta}, \ldots, c_{m\theta})$ is a vector defining the factors of task T_θ that would influence the decision (accept/decline the application) of participant i. A historical case is defined as $(i, \theta, \overrightarrow{\mathrm{PF}}_{i\theta}, \lambda)$, where i and θ are the identity of a participant and a task, respectively, $\overrightarrow{\mathrm{PF}}_{i\theta}$ is a m-dimensional participant-side factor vector, and λ is the two-valued outcome (accept/decline).

Our case-based reasoning algorithm consists of following two steps.

Step 1 (case selection). Since the influence of different participant-side factors varies from one participant to another, the algorithm first selects historical cases of participant i to compute the likelihood of a participant i to undertake task T_h. Using selected cases, following two reference matrices are established, which are the positive reference matrices $(\overrightarrow{\mathrm{RP}}_1, \overrightarrow{\mathrm{RP}}_2, \ldots, \overrightarrow{\mathrm{RP}}_{x(i)})^T$ and the negative reference matrix $(\overrightarrow{\mathrm{RN}}_1, \overrightarrow{\mathrm{RN}}_2, \ldots, \overrightarrow{\mathrm{RN}}_{y(i)})^T$, where $x(i)$ and $y(i)$ are the number of "accept" (positive) and "decline" (negative) cases, respectively. $\overrightarrow{\mathrm{RP}}_j = (a_{j1}, a_{j2}, \ldots, a_{jm})$ is a participant-side factor vector whose corresponding outcome λ is "accept". $\overrightarrow{\mathrm{RN}}_j = (b_{j1}, b_{j2}, \ldots, b_{jm})$ is a participant-side factor vector whose corresponding feedback λ is "decline"

$$\begin{aligned}
&\left(\overrightarrow{\mathrm{RP}}_1, \overrightarrow{\mathrm{RP}}_2, \ldots, \overrightarrow{\mathrm{RP}}_{x(i)}\right)^T \\
&= \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{x(i)1} & a_{x(i)2} & \cdots & a_{x(i)m} \end{pmatrix}, \\
&\left(\overrightarrow{\mathrm{RN}}_1, \overrightarrow{\mathrm{RN}}_2, \ldots, \overrightarrow{\mathrm{RN}}_{y(i)}\right)^T \\
&= \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ b_{y(i)1} & b_{y(i)2} & \cdots & b_{y(i)m} \end{pmatrix}.
\end{aligned} \tag{1}$$

Step 2 (likelihood measurement and participant selection). We use $U(i, h)$ to measure the likelihood of participant i to accept task T_h (see (2)), where $\mathrm{Dist}(\overrightarrow{\mathrm{PF}}_{ih}, \overrightarrow{\mathrm{RN}}_j)$ is the Euclidean distance between $\overrightarrow{\mathrm{PF}}_{ih}$ and $\overrightarrow{\mathrm{RN}}_j$ and $\mathrm{Dist}(\overrightarrow{\mathrm{PF}}_{ih}, \overrightarrow{\mathrm{RP}}_j)$ is the Euclidean distance between $\overrightarrow{\mathrm{PF}}_{ih}$ and $\overrightarrow{\mathrm{RP}}_j$. K participants with maximum $U(i, h)$ are what we want to select

$$U(i, h) = \sum_{j=1}^{y(i)} \mathrm{Dist}\left(\overrightarrow{\mathrm{PF}}_{ih}, \overrightarrow{\mathrm{RN}}_j\right) - \sum_{j=1}^{x(i)} \mathrm{Dist}\left(\overrightarrow{\mathrm{PF}}_{ih}, \overrightarrow{\mathrm{RP}}_j\right). \tag{2}$$

This equation is easy to understand. The more likely a participant i is to accept task T_h, the closer $\overrightarrow{\mathrm{PF}}_{ih}$ is to positive cases and the more distant it is from negative cases.

Note that different participant-side factors could have different impact on deciding the outcome, the weights of different factors have to be considered (see (3)) when calculating

Table 1: Measurable participant-side factor.

Participant-side factor	How it is measured	Possible values	
		Values	Meaning
Incentive (reward)	Measured by the reward (incentive) defined by the organizer	**1**	0~10 US cents
		2	10~50 US cents
		3	50~100 US cents
		4	1~3 USDs
		5	more than 3 USDs
Interest	Measured by how many tasks a participant has undertaken in the same domain	**1**	0 tasks
		2	1~3 tasks
		3	3~5 tasks
		4	5~10 tasks
		5	more than 10 tasks
Data privacy sensitivity	Measured by the sensitivity of the data the participant must provide when fulfilling the task	**1**	Do not provide private data
		2	Only provide coarse-grained location data (e.g. at Peking University)
		3	Provides very sensitive private data (e.g., sound data via microphone)

the distance, where $w_i \in (0, 1)$ is the weight of ith participant-side factor

$$\begin{aligned} \mathrm{Dist}\left(\overrightarrow{\mathrm{PF}_{ih}}, \overrightarrow{\mathrm{RN}_j}\right) &= \sqrt{\sum_{p=1}^{m} \left(c_{ph} - b_{jp}\right)^2 * w_p}, \\ \mathrm{Dist}\left(\overrightarrow{\mathrm{PF}_{ih}}, \overrightarrow{\mathrm{RP}_j}\right) &= \sqrt{\sum_{p=1}^{m} \left(c_{ph} - a_{jp}\right)^2 * w_p}. \end{aligned} \tag{3}$$

Now the key problem is how to precompute the weight w_p $(p = 1, 2, \ldots, m)$. We propose a weight calculation algorithm based on the sensitivity analysis principle [32]. This principle is to identify the key variables for a certain target, by calculating the effect of variable's change on the target's change. This principle mainly consists of following steps.

Determine the variables and target to be analyzed according to the problem.

Vary only one variable, observing and measuring the change of target.

Repeat the analysis for each variable repetitively for calculating the corresponding effect on the target.

Based on the above sensitivity analysis principle, we propose a personalized weight calculation algorithm according to our problem. Some key points of Algorithm 1 are explained as follows.

The variables are participant-side factors, and the target is the outcome (accept/decline). Matrix $A = (\vec{Q}_1, \vec{Q}_2, \ldots, \vec{Q}_{[x(i)+y(i)]})^T$ is all cases of participant i, which is the mergences of positive reference matrix $(\overrightarrow{\mathrm{RP}_1}, \overrightarrow{\mathrm{RP}_2}, \ldots, \overrightarrow{\mathrm{RP}_{x(i)}})^T$ and the negative reference matrix $(\overrightarrow{\mathrm{RN}_1}, \overrightarrow{\mathrm{RN}_2}, \ldots, \overrightarrow{\mathrm{RN}_{y(i)}})^T$.

Lines 3~19 calculate the weight of a certain participant-side factor x_k by changing x_k and remaining others unchanged.

In line 11~12, $L = \mathrm{Max}\{q_{1k}, q_{2k}, \ldots, q_{(x(i)+y(i))k}\} - \mathrm{Min}\{q_{1k}, q_{2k}, \ldots, q_{(x(i)+y(i))k}\}$ is the maximum difference among the possible values of the kth variable, and $L/|q_{jk} - q_{lk}|$ measures the effect of variable x_k's change on the target's change. It is reasonable, because when $|q_{jk} - q_{lk}|$ is smaller and the target has changed, it means that even the slightest change of x_k is enough to lead to the change of target (indicating that this variable's weight is heavier).

We adjust the weight to make sure that their sum is equal to 1 (in line 20).

4.3. Strategy for Cold-Start Problem. We may encounter the cold-start problem when a participant just registered in a crowd sensing mediation platform. In this situation, the number of one's historical cases is zero and our framework knows nothing about the new participant.

We deal with the cold-start problem by leveraging the cases of other participants, because users tend to make similar decisions under similar contexts. The approach is almost similar to the personalized weight calculation algorithm but different in the following two aspects.

(1) Case Selection. In the case selection step, we select historical cases of all the other participants.

(2) General Weight Calculation. The weight calculation algorithm is almost the same as the personalized weight calculation algorithm, but the input is changed to the entire datasets (all cases from all participants). The calculated weights are referred to as the general weight, which will be used in the likelihood measurement and participant selection.

Input: Matrix A is the merging of positive and negative reference matrix of the participant i, and vector B is the corresponding feedbacks ($\vec{Q}_j$ corresponds to λ_j).

$$A = (\vec{Q}_1, \vec{Q}_2, \ldots, \vec{Q}_{x(i)+y(i)})^T = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1m} \\ q_{21} & q_{22} & \cdots & q_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ q_{(x(i)+y(i))1} & q_{(x(i)+y(i))2} & \cdots & q_{(x(i)+y(i)m} \end{pmatrix}$$

$B = (\lambda_1, \lambda_2, \ldots, \lambda_{(x(i)+y(i))})^T \quad \lambda_i \in \{\text{accept, decline}\}$.

Output: w_i $(i = 1, 2, \ldots, m)$

```
(1)  ALGORITHM BEGIN
(2)    float total = 0, effect = 0;
(3)    FOR (k = 1 to k = m) {
(4)        /*Calculate the weight w_k.*/
(5)        FOR (from j = 1 to j = x(i) + y(i))
(6)            FOR (from l = j + 1 to l = x(i) + y(i))
(7)                IF (q_jk ≠ q_lk & other components of
(8)                    Q_j and Q_l are identical)
(9)                    total ++;
(10)                   IF (λ_j ≠ λ_l)
(11)           L = Max{q_1k, q_2k, ..., q_(x(i)+y(i))k}
(12)               − Min{q_1k, q_2k, ..., q_(x(i)+y(i))k}.
(13)                   effect = effect + L/|q_jk − q_lk|;
(14)               END FOR
(15)        END FOR
(16)        w_i = effect/total.
(17)    /*clear the parameters for calculating next weight.*/
(18)        total = 0, effect =0.
(19)   END FOR
(20)     FOR (j = 1 to j = m) w_j = w_j/ Σ_1^m w_i
(21)       //Adjust the weight to make their sum to be 1
(22)   ALGORITHM END
```

Algorithm 1: Personalized weight calculation algorithm.

5. Demonstration and Evaluation

This section demonstrates or evaluates the effectiveness of our toolkit from the following three aspects.

First, we will demonstrate the usability of the toolkit in Section 5.1. The usability is measured by whether organizers without programming experience can create MCSA with our tool accurately within short period of time. Therefore, our demonstration answers two questions, respectively. (1) Does the generated application meet the requirement? (2) How much time is required to create the application with our toolkit?

Second, we will demonstrate the expressiveness of the toolkit in Section 5.2. We design a tool for organizers to develop MCSA just by doing some simple settings, but we are also wondering how many applications can be developed by the tool which is an important question.

Third, since the willingness-based participation selection algorithm is an important contribution of this paper, we will evaluate its validity in Section 5.3.

5.1. Usability Demonstration. In order to evaluate the usability, we have conducted a user study consisting of following three steps.

Step 1 (developer recruitment). We have recruited 12 volunteers acting as the role of the organizer. The selection of volunteers is based on following two criterions.

Criterion 1. Volunteers are not required to have the experience to create software using traditional programming languages (e.g., Java, C++, and Python) and development tools.

Criterion 2. Volunteers should be familiar with the use of smart mobile phone. Besides, they should have enough competency of learning to use new software.

Step 2 (MCSA creation). Each volunteer is provided with a mobile phone and uses the tool to create two MCSAs in the following. The development process of each developer has been recorded for latter analysis.

Application 1: Manhole Cover Monitoring. Manhole covers are generally provided on a pedestrian or vehicle road. However, if they were stolen, the "trap" left may pose fatal damage to people living in the city. Of course, urban administrators have the responsibility to see if there is a cover lost and to fill the trap immediately for the public security. However, since the number of such administrators is limited, it is not enough by totally relying on them. Inspired by the MCS concept, administrators can develop MCSA to utilize the collective power of citizens on the roads. When citizens find a cover that was stolen, they can use this application to take photos for the missing manhole cover and upload them to the server with location tags. Then administrators can view the data and fill the missing manhole covers as soon as possible.

Application 2: Available Seats Seeking. When the final exam is approaching, it is often very difficult to find an available seat in the library or teaching building on campus. Sometimes you have to walk through a lot of rooms but in the end desperately find that there is no seat available. The use of MCS can be a good solution to this problem. You can develop MCSA by asking which classrooms have available seats, and the students currently in the classroom can answer it.

Step 3 (development process analysis). As mentioned above, there are two aspects for the usability evaluation. (1) Does the generated application meet the requirement? (2) How much time is required to create the application with our tool? Therefore, the analysis for the development process is divided into two parts.

Part 1: Accuracy of the Application. We will first demonstrate how the "accurate" application looks like and how it is developed.

With our tool, Manhole Cover Monitoring application is developed by just doing some settings as follows. The SL can be left empty, because we want the application to be running all the time. The geographical coverage should be set for the whole city. The recruitment policy is set to be "presence mode." The sensing data type is set to be photos. The user interfaces for organizers to view the uploaded data are shown in Figure 5.

Similarly, Available Seats Seeking application is developed by doing following settings. The SL is set to be the time the user want, say five minutes, because the organizer wants to find the seats within five minutes, otherwise he will go to find available seats by himself. The geographical coverage is be set for the whole campus. The recruitment policy is set to be "presence mode." The sensing data type is set to be "text." The user interfaces of this application (for both the participants and the organizers) are shown in Figure 6.

According to the applications developed by 12 volunteers, all of them have done the right settings and generate the applications with the same functionality as the accurate one. Furthermore, in latter interview, all volunteers think that the meaning of each setting is very easy to understand.

Part 2: Time Cost for Application Development. The time required to create the application with our tool was measured (see Figure 7). The average time to create an application was 4 minutes for Application 1 (Manhole Cover Monitoring) and 5.5 minutes for Application 2 (Available Seats Seeking). These results indicate that volunteers are easily able to create a mobile application, even in their first experience with the tool. According to Figure 7, we find that three volunteers (i.e., #3, #8, and #12) have relatively long development time. By further investigation, we find that these three volunteers use iPhone system in their daily life, while our tool has been implemented on the Android phone system. Thus it appears that their unfamiliarity with Android system might be reasons for their longer development time. With follow-up interviews they admit that the unfamiliarity with Android system indeed makes the development process slower.

5.2. Expressiveness Demonstration.

In addition to the usability of the tool, another evaluative indicator is the expressiveness. In other words, we are concerned about how many MCSAs can be developed with it and how many cannot.

We have listed 12 application examples in Table 2. After collecting those applications, we try to use our tool to build them one by one. The development result in Table 2 indicates that ten of them can be developed and only two cannot. Here when judging whether our tool can develop an application or not, we only judge whether the organizer can collect the data he wants and do not consider the data analysis process after gathering the data on the server. Therefore, the tool shows a good expressiveness for creating MCSA. The reason why our tool is not able to develop [13, 14] is that they require complex data processing in the phone-side before uploading to the server-side. However, our tool does not support for such complex and application-specific sensor data processing on the phone-side.

Table 2: Some MCSA examples.

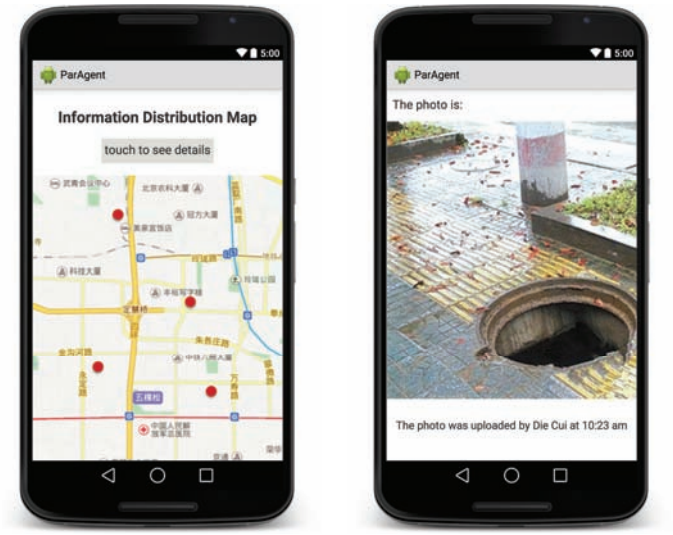
MCSA examples	Can be created by our tool or not
Petrol watch [6]	Yes
Haze watch [7]	Yes
Meteor counter [8]	Yes
Biketastic [9]	Yes
Citizen journalist [10]	Yes
Spot reporter [11]	Yes
Forensic analysis [11]	Yes
Auditioning [11]	Yes
Earphone [12]	Yes
Nericell [7]	Yes
Bus waiting time prediction [13]	No
QTime [14]	No

5.3. Willingness-Based Selection Algorithm Evaluation

5.3.1. Data Collection. We post a questionnaire on an online platform [33] to generate the dataset. First, we assume that there is a crowd sensing task with basic description. Then we design 15 questions, each of which ask for the outcome under different combination of participant-side factors. Second, we invite 26 volunteers (including the undergraduate, graduate students, and faculties in our institute) to respond to this questionnaire. Third, we collect all these questionnaires and generated the dataset. The generated dataset consists of $75 * 26 = 1950$ historical cases (each question and its answer can generate 5 cases of a participant, so that each participant has $15 * 5 = 75$ cases and the overall dataset consists of $75 * 26$ cases).

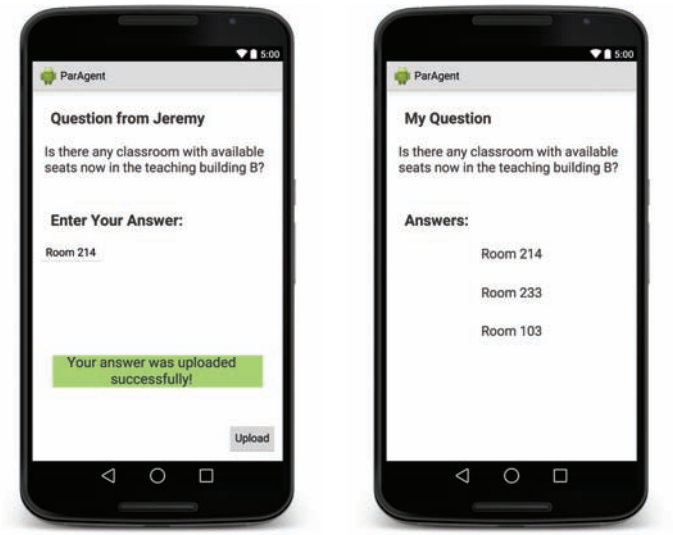
5.3.2. Experimental Methodology. After the dataset has been generated, we design a 6-round experiment to evaluate the case-based reasoning algorithm for willingness-based participant selection. The generated datasets are used both to establish the case database, which is the input of our algorithm, and as testing cases containing the ground truth. Our 6-round experiment is summarized in Table 3.

(1) The first round is for the baseline method. It is assumed that none of the cases have been preacquired, so that we can only randomly select k participants from 26 candidates. This round of experiment consists of the following steps: (a) randomly assign value to the components of participant-side factor vector $\overrightarrow{\mathrm{PF}}_{ih}$. (b) Perform the baseline method, that is, guessing k participants randomly. (c) Check how many of those guessed participants will undertake the task based on the ground truth in the datasets. (d) Perform (a)~(c) for 10 times and calculate the average number of right-selected participants.



(a) (b)

Figure 5: The user interface of Cover Monitoring application. (a) Interface for selecting a data point. (b) Interface for viewing the data in detail.



(a) (b)

Figure 6: The user interface of Available Seats Seeking application. (a) the interface for participants to answer questions. (b) interface for the organizer to view the answers.

Table 3: Summary of 6-round experiment.

Round	The number of cases as historical data	Selection approach
1st round	Zero	Baseline method (random selection)
2nd round	Each participant has 25 cases	Approach in Section 4.2
3rd round	Each participant has 50 cases	Approach in Section 4.2
4th round	Each participant has 75 cases	Approach in Section 4.2
5th round	Two participants have no historical cases Each of the other participants has 50 cases	Approach in Section 4.2 + approach in Section 4.3
6th round	Five participants have no historical cases Each of the other participants has 50 cases	Approach in Section 4.2 + approach in Section 4.3

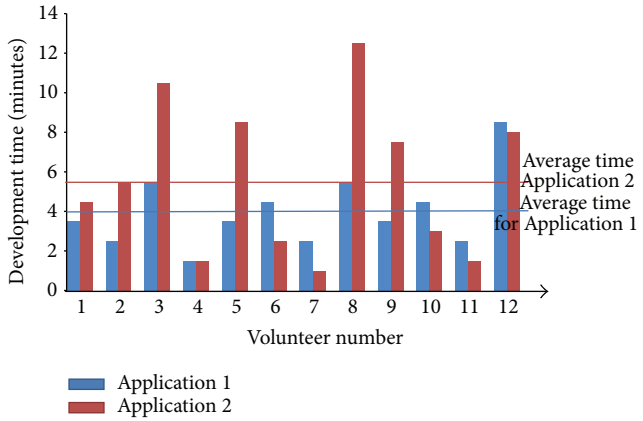


Figure 7: Development time for each volunteer.

(2) The 2nd~6th rounds of the experiments are to test our case-based reasoning algorithm when the number of cases in the case database changes. (1) The 2nd~4th rounds of experiments assume that each of 26 candidate participants has preacquired cases, and the difference among these three rounds is the number of cases. In these three rounds, we calculate the personalized weight based on the approach in Section 4.2. (2) The 5th~6th rounds assume that some participants do not have preacquired cases (3 and 5 participants, resp.). In these two rounds, we will use our mechanism in Section 4.3 for participants without historical cases. In both the 2nd~6th rounds, we first randomly generated 100 participant-side factor vectors as testing cases. Then, for each vector, we changed the second component (i.e., interest) randomly since each candidate participant's interest for the same task may be different and kept the other two components (i.e., reward and data privacy sensitivity) unchanged since they are the same among participants for a specific task.

5.3.3. Experimental Results. The willingness-based selection accuracy of each round is measured by Accuracy(i) = $T(i)/k$ ($i = 1, 2, \ldots, 6$), where $T(i)$ is the number of participants in the ith round who will undertake the task based on the ground truth and k is the number of selected participants.

Since the number of selected participants (i.e., k) has impact on the selection accuracy, we conduct our 6-round experiments for 3 times by changing k from 10, 15 to 20, respectively. Hence we have completed $3 * 6 = 18$ rounds of experiments in total, whose selection accuracy is demonstrated in Figure 8.

According to the observation of the results, we can draw following conclusions.

(1) Our case-based reasoning algorithm outperforms the baseline method no matter $k = 10, 15$ or 20, and no matter the number of cases as historical data is 25, 50, or 75. Therefore, it shows the effectiveness of our algorithm in different situations.

(2) The smaller, the bigger increase in selection accuracy of our algorithm compared to the baseline method.

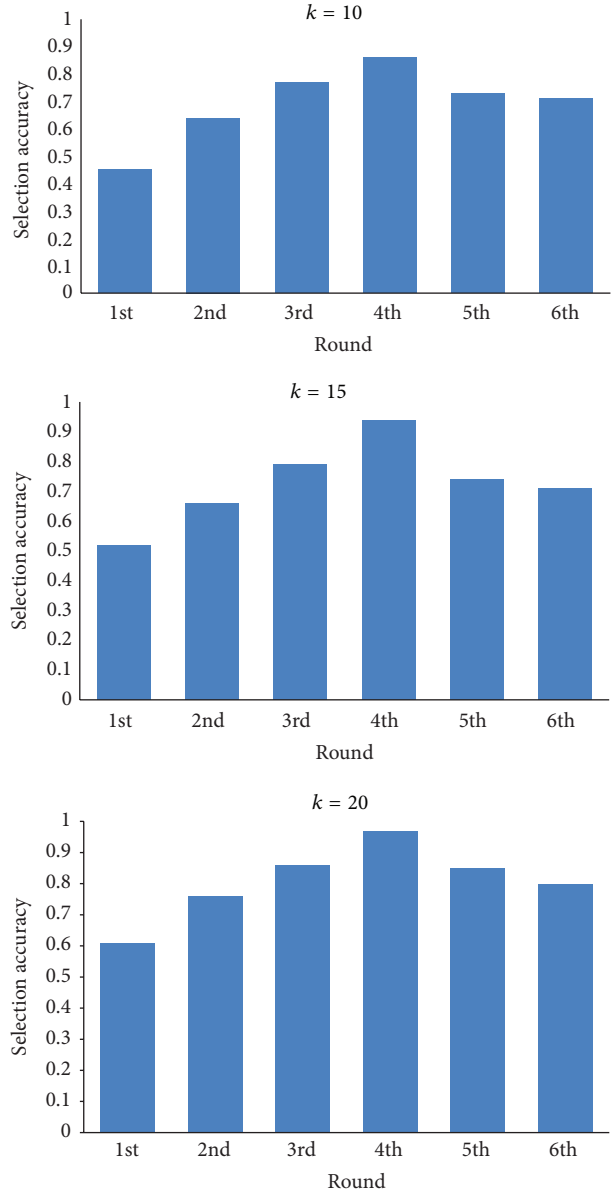


Figure 8: Selection accuracy of different rounds.

(3) Compared to the 2nd, 3rd, and 4th rounds, we can see that the selection accuracy of our algorithm is improved with the increasing of the number of historical cases.

(4) The experimental result of 5th and 6th round shows that our mechanism in Section 4.3 is effective to solve the cold-start problem to some extent. Although the accuracy of 5th and 6th round is lower than the 3rd round, it still significantly outperforms the baseline method.

6. Discussion and Conclusion

This paper develops a prototype toolkit for the organizers of the MCS campaigns to create applications. Comparing

with related works, it is highly configurable and suitable for many domains. Meanwhile, it enables nonprofessionals to build MCSA by doing some simple settings, which totally eliminates the requirement of programming skills. Moreover, we proposed a case-based reasoning algorithm to select participants based on the learning of their preference. The toolkit can enable the applications to be delivered to those who are more likely to accept them. The evaluation result shows that the tool has good usability and expressiveness.

However, it still has some limitations.

First, privacy preserving, which is important for MCS, has not been well considered in the current version. In this paper, we only preserve the privacy of participants without considering others who might be involved in the MCS. For example, organizers who want to help the poor may launch a campaign to call out people to take photos or videos about the lives of the poor. However, such actions are deemed illegal due to the invasion of the privacy. In the future work, we will introduce corresponding mechanisms into the tool to avoid such problems.

Second, it cannot support the creation of applications requiring complex data processing in the phone-side before uploading the information needed. Since the tool targets at organizers rather than professional software developers, we want to make the development as simple as possible by sacrificing some expressiveness. Besides, we believe that the development of MCSA should be accomplished both by professional software developers and organizers without programming experience. Professional software developers are responsible for building complex applications by other more powerful but complicated tools, while organizers create some simple ones with our toolkit.

Third, we use datasets generated by an online questionnaire, rather than real-world MCS participation history datasets, to evaluate the willingness-based participation selection algorithm. In the future, we plan to publish our prototyped toolkit onto the Android application store. If it is used by a large number of MCS participants, historical data can be accumulated with the passage of time to evaluate our participation selection mechanism.

Finally, the prototype system only provides the organizers with the capability of defining the MCS task, delivering the application, and viewing the uploading data. However, it does not include the data analysis function (e.g., statistical report and visualization), which may be very useful in some application scenarios.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is funded by the National High Technology Research and Development Program of China (863) under Grant no. 2013AA01A605.

References

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[2] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4W1H in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 42–48, 2014.

[3] D. M. Aanensen, D. M. Huntley, E. J. Feil, F. Al-Own, and B. G. Spratt, "EpiCollect: linking smartphones to web applications for epidemiology, ecology and community data collection," *PLoS ONE*, vol. 4, no. 9, Article ID e6968, 2009.

[4] Project Noah, http://www.projectnoah.org.

[5] S. Kim, J. Mankoff, and E. Paulos, "Sensr: evaluating a flexible framework for authoring mobile data-collection tools for citizen science," in *Proceedings of the 2nd ACM Conference on Computer Supported Cooperative Work (CSCW '13)*, pp. 1453–1462, ACM, February 2013.

[6] Y. Dong, S. S. Kanhere, C. T. Chou, and N. Bulusu, "Automatic collection of fuel prices from a network of mobile cameras," in *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '08)*, Santorini, Greece, June 2008.

[7] J. Carrapetta, *Haze Watch: Design of a Wireless Sensor Board for Measuring Air Pollution*, School of Electrical and Telecommunications Engineering, University of New South Wales, Sydney, Australia, 2010.

[8] S. Heggen, A. Adagale, and J. Payton, "Lowering the barrier for crowdsensing application development," in *Mobile Computing, Applications, and Services*, pp. 1–18, Springer International Publishing, 2014.

[9] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava, "Biketastic: sensing and mapping for better biking," in *Proceedings of the 28th Annual CHI Conference on Human Factors in Computing Systems (CHI '10)*, pp. 1817–1820, April 2010.

[10] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt, "Micro-Blog: sharing and querying content through mobile phones and social participation," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys '08)*, pp. 174–186, 2008.

[11] M. Ra, B. Liu, T. La Porta, and R. Govindan, "Medusa: a programming framework for crowd-sensing applications," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, pp. 337–350, 2012.

[12] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "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 (IPSN '10)*, pp. 105–116, Stockholm, Sweden, April 2010.

[13] P. Zhou, Y. Zheng, and M. Li, "How long to wait?: predicting bus arrival time with mobile phone based participatory sensing," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, pp. 379–392, ACM, June 2012.

[14] Y. Wang, J. Wang, and X. Zhang, "PRISM: platform for remote sensing using smartphones," in *Proceedings of the 8th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '10)*, pp. 770–777, May 2013.

[15] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: platform for remote sensing using smartphones," in *Proceedings of the 8th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '10)*, pp. 63–76, June 2010.

[16] N. Ramanathan, F. Alquaddoomi, H. Falaki et al., "ohmage: an open mobile system for activity and experience sampling," in *Proceedings of the 6th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth '12)*, pp. 203–204, IEEE, San Diego, Calif, USA, May 2012.

[17] L. Ravindranath, A. Thiagarajan, H. Balakrishnan, and S. Madden, "Code in the air: simplifying sensing and coordination tasks on smartphones," in *Proceedings of the 13th Workshop on Mobile Computing Systems and Applications (HotMobile '12)*, February 2012.

[18] A. Joki, J. Burke, and D. Estrin, "Campaignr: a framework for participatory data collection on mobile phones," Tech. Rep., UCLA Center for Embedded Net-Work Sensing, 2007.

[19] C. Hartung, Y. Anokwa, W. Brunette, A. Lerer, C. Tseng, and G. Borriello, "Open data kit: tools to build information services for developing regions," in *Proceedings of the 4th ACM/IEEE International Conference on Information and Communication Technologies and Development (ICTD '10)*, ACM, December 2010.

[20] S. Heggen, A. Adagale, and J. Payton, "Lowering the barrier for crowdsensing application development," in *Proceedings of the 5th International Conference on Mobile Computing, Applications, and Services (MobiCASE '13)*, Paris, France, November 2013.

[21] J. Wang, Y. Wang, and J. Zhao, "Helping campaign initiators create mobile crowd sensing apps: a supporting framework," in *Proceedings of the 39th Annual International Computers, Software & Applications Conference*, Taichung, Taiwan, July 2015.

[22] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "AnonySense: privacy-aware people-centric sensing," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pp. 211–224, ACM, June 2008.

[23] G. Cardone, L. Foschini, P. Bellavista et al., "Fostering participaction in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 112–119, 2013.

[24] D. Zhang, H. Xiong, L. Wang, and G. Chen, "CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*, pp. 703–714, ACM, 2014.

[25] H. Xiong, D. Zhang, G. Chen, L. Wang, and V. Gauthier, "CrowdTasker: maximizing coverage quality in piggyback crowdsensing under budget constraint," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom '15)*, St. Louis, Mo, USA, 2015.

[26] S. Reddy, D. Estrin, and M. Srivastava, "Selection framework for participatory sensing data collections," in *Pervasive Computing*, pp. 138–155, Springer, Berlin, Germany, 2010.

[27] S. Reddy, K. Shilton, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using context annotated mobility profiles to recruit data collectors in participatory sensing," in *Location and Context Awareness*, pp. 52–69, Springer, 2009.

[28] E. Cuervo, P. Gilbert, B. Wu, and L. P. Cox, "CrowdLab: an architecture for volunteer mobile testbeds," in *Proceedings of the 3rd International Conference on Communication Systems and Networks (COMSNETS '11)*, January 2011.

[29] Y. Xiao, P. Simoens, P. Pillai, K. Ha, and M. Satyanarayanan, "Lowering the barriers to large-scale mobile crowdsensing," in *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications (HotMobile '13)*, February 2013.

[30] Z. Song, B. Zhang, C. H. Liu, A. V. Vasilakos, J. Ma, and W. Wang, "QoI-aware energy-efficient participant selection," in *Proceedings of the 11th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON '14)*, pp. 248–256, Singapore, June-July 2014.

[31] S. He, D. H. Shin, J. Zhang, and J. Chen, "Toward optimal allocation of location dependent tasks in crowdsensing," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '14)*, pp. 745–753, IEEE, April 2014.

[32] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto, *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*, John Wiley & Sons, 2004.

[33] Questionnaire, http://www.sojump.com/jq/3999847.aspx.