KinoSense: Framework for Tasking Applications on Smartphones using Sensing and Co-ordination

Amruta Deshpande, Apurva Bhoite, Ashish Kalbhor, Sandeep Mane, Prema Desai

Abstract—A growing class of smartphones use tasking applications that run continuously, process data from sensors to determine the user's context (such as location), and fire certain actions when the right conditions occur. We propose a framework which enables smartphones to use device sensors to perform some actions by defining a rule. The idea is to develop primitives that would simplify the use of device sensors and services for both developers as well as users.We hereby overcome the issues in current approaches by creating broader and more meaningful definitions to be used on smartphones. KinoSense provides a task execution framework to automatically distribute and coordinate tasks, energy-efficient modules to infer user activities and compose them.These applications combine sensing from the device and the coordination between all sensors.

Keywords— Framework, Triggers, Actions, Sensors, Events

I. INTRODUCTION

Smartphones and other mobile devices are now equipped with a huge array of sensors based on three categories viz. Motion Sensors, Environmental Sensors, Position Sensors. These include Accelerometer, Gyroscope, Photometer, Magnetometer, GPS and many more. These capabilities provide smartphones the ability to discover more about users and their activities than any other commodity computing device ever invented.Application developers have now started taking advantages of powerful capabilities of device sensors. This has introduced a new class of tasking applications.Tasking applications process data from these sensors continuously to track the user's activity and context, in order to perform some tasks based upon the state of device.

These types of applications basically neither require any user input nor any other kind of explicit requirement. It works on the conditions that are provided to perform a task (e.g., Turn off the Ringer, when I am inside the movie theatre). Here, in the given example, the sensor that is being used is Location-based and the action gets performed only upon the confirmation of this trigger. Tasking applications are now getting popular with end users. E.g., Blackberry [1] has now introduced Location-based reminder for its smartphones.Our goal is to make it easy to develop and run new taskingapplications. KinoSense is a system which simplifies the rapid development of tasking applications. It enables non-expert end users to easily express simple tasks on their phone.

Manuscript received on April, 2013.

Amruta Deshpande, Computer Department, SKNCOE, University of Pune, Pune.

Apurva Bhoite, Computer Department, SKNCOE, University of Pune, Pune.

Ashish Kalbhor, Computer Department, SKNCOE, University of Pune, Pune.

Sandeep Mane, Computer Department, SKNCOE, University of Pune, Pune.

Using KinoSense, a variety of tasks becomes easier to express and run. These include single-device tasks (e.g., don't connect to Wi-Fi when I'm outside and moving), multi-user tasks (make my phone silent when I'm meeting with my boss), tasks with complex activities (map my path when I bike or run but not when I drive or ride the bus), and multi-device tasks (put my laptop to sleep if I've not been in the same room for more than 15 minutes).

II. PROBLEMS WITH CURRENT APPROACHES

1. POOR PROGRAMMING CONSTRUCTS.

Writing tasking applications involves both server-side and device code, and figuringout how to partition that code. A better approach for this type of application would be a *Macro-programming approach*, which only requires developers to write server-side code, with an execution framework thatautomatically partitions the code across one or moresmartphones and the server. This framework also supports end users, who have no interest or ability to write tasking code, is desirable.Such users should be able to combine existingcapabilities to specify their own tasks. In this way, auser need not be dependent on, or wait for, a developer to create the task. She/he can use the methods directly from the framework for a particular context according to the requirement.

2. POOR ABSTRACTION.

Today, writing tasking applicationsrequires grappling with low-level sensor data which is very hectic.Even something as easy to express as "user is wa king" is difficult because one needs to processdata from position, ccelerometer, and/or gyroscopesensors to make this determination. An ideal solutionwould allow developers, and even end users, to use (and reuse) an isWalking primitive directly from the KinoSense framework.

This paper addresses these shortcomings by proposing KinoSense, a system that lowers the barrier for programming and executing tasking applications. KinoSense helps two groups of people:

1. Developers, who can create tasks by writing only server-side code, even for tasks that involve multiple end users and their devices, a variety of sensors, and the server. This Framework allows developers to write task scripts, compiles them into server-side and mobile code, and manages the task execution run-time on mobile devices.
2. End users, who are able to specify their own tasks by "mixing and matching" available triggers and actions via a smartphone UI.

III. SYSTEM ARCHITECTURE

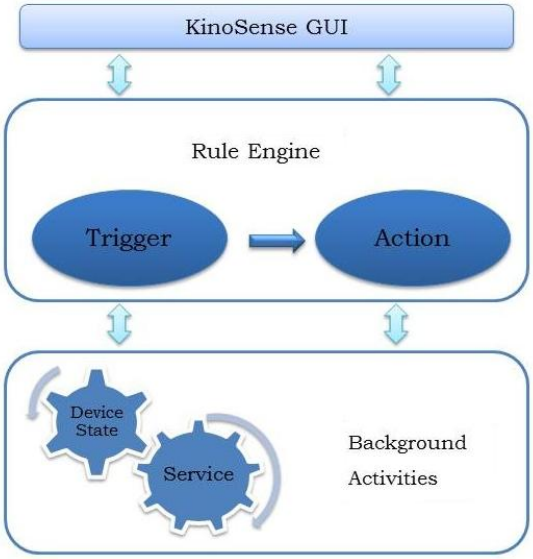

Fig. 1. KinoSense Architecture Diagram.

KinoSense architecture is based upon two basic building blocks: Triggers and Actions. These blocks run coherently on background activities that include Services, which actually are constantly working on the device state.

A. Triggers

Triggers are events that are triggered after some change in the device state. Triggers can be both, low level or high level. An Accelerometer event can be a low level trigger, mentioning user is turning his device around various axes. A User flipping the phone facedown can be a high level trigger. A High Level Trigger indicates an intention; in this case it can mean the user wants to put his/her phone on Silent. Triggers are to be used alone or in conjunction with other Triggers and Actions needs to be fired when triggers fulfills a set of rules.

B. Actions

Actions are tasks that the device performs automatically based upon the trigger that is invoked. Actions can be low level or high level. Phone going on Silent mode can be low level action. Phone going in a "Do Not Disturb and Auto Reply" mode is a high level action. This high level action is composed of phone going in a silent mode as well as disconnecting an incoming call and sending the caller a predefined SMS mentioning - "In a Meeting, Will call you back".

C. Rules

Rule is nothing but fulfillment of set of Triggers (and may be some context). If rules are fulfilled, then appropriate actions should be fired on the device.
The simplest rule can be

If (Hari gets an “Incoming Call”) then
{Put Phone on “Silent” Mode} ……..!
Here, the condition in ‘( )’ is the Trigger that maps to the Action that is put in ‘{ }’. In composite rules, there are multiple triggers and actions that would be considered.

IV. TRIGGER LAYER

KinoSense includes a set of high level triggers which make it easy for developers that reference such complex triggers, such as the fact that the user is “In a meeting”. It also provides an easy way to compose primitive triggers to build useful higher-level triggers. Both accuracy and energy efficiency are non-trivial challenges. High-level triggers may be a combination of one or more low level ones. Our system recognizes a number of low-level triggers such as low battery, headset connected/disconnected, flipping down the device.The triggers would be generally used as a set in our project so that a set of rules incorporate them according to the device state. Some of the basic triggers that we have used in our framework are:

A. Low Battery Trigger

When this trigger is made active, the sensor service will check the device battery status and fire the trigger only when the level is below a predefined threshold value, say 15%. Else, the battery status is scanned after sometime according to the code that has been implemented. Here, the threshold value to trigger “Low Battery” is modifiable according to developer’s demands.

B. Device Flipping Trigger

Accelerometer is used to detect the flipping down of device. The sensor service checks for the Z-axis of device. When the z-coordinate becomes less than 0, it means that the device has been flipped “Down” i.e. screen is facing down towards the earth. Similarly, changing of the z-coordinate to positive side tells us that device has been flipped “Up” now.

C. Unusual Activity on device Trigger

We define this trigger for the safety and security concerns of the user. Unusual activity can be defined when the device has not been used for a long time, which in turn can be detected by scanning the Logs of the device. Also, if the Sim card gets changed, then we can cross verify it with the saved details of default Sim and hence notify through emergency contacts.

D. Wi-Fi Trigger

When the device Wi-Fi is enabled, this trigger scans the network for predefined SSID value by the user, which when found would fire this trigger. Simplest way of using this trigger would be to connect to this set Wi-Fi point. Otherwise, we can also use it as a location-based trigger to identify Meeting Room, or College Department, or Home.

E. GPS Location Trigger

Global Positioning System gives the location coordinates to the device in the form of latitude and longitude, which can help to define specific locations for the users i.e. Office, School, Home, Friend’s Place, etc. Thus, this trigger would help the user for defining his location, and would also help the developers to further explore the prospects with complex triggers.
These are the basic comprehensive triggers which can be modified and used as a part of composition of triggers. Other simpler triggers are headset connected/disconnected, Incoming Call, Shaking the device, Power connected/disconnected, etc.
Trigger Composition is the advanced version of using this triggers that KinoSense framework provides.Our framework allows developers and users to compose lower-level triggers using logical predicates like ‘AND’,’OR’ to create high-level triggers. This is very powerful because it allows developers and end users to reuse trigger modules created by other developers to quickly write their own.

For example, consider the problem of detecting that a user is outdoors and walking, and switching to 3G from Wi-Fi when that happens. Since "user is outdoors" and "user is walking" are primitive activities, we can compose them using an AND predicate in CITA to build the complex event "outdoors and walking".

Another example: Hari wants his phone to go into silent mode automatically when an incoming call comes and he flips down his phone.

This is a Multi-trigger predicate, which can be expressed in KinoSense as follows:

• If Hari gets an Incoming Call, AND
• Hari flips down his Phone.

Then make Hari's phone silent.

This feature thus increases the flexibility of the code by adding logical predicates and in turn provides code reusability. Users can create combined triggers on their own as per their needs. Similarly, the developers can use this framework to generate compositions on their own.

V. ACTION LAYER

Actions are the events that occur because of a trigger or a set of triggers according to the rule specification. These can be both low-level as well as high-level depending upon the complexity and definition where it is put in use. We can say that "Putting phone on Silent Mode" or "Sending an SMS" are low level actions, but the combined action of "During an incoming call, putting phone on Silent and Sending SMS to the caller" would be defined as high level action. We have included various actions in the framework which can be utilized by developer or user as required.

A. Low-Level Actions

Simple actions which are basically supposed to be used with predicates are defined low level actions. KinoSense framework would provide following low level actions along with appropriate trigger:

1. Reduce display brightness.
2. Enable/Disable Wi-Fi.
3. Toggle Flight Mode.
4. Sending an SMS.
5. Reduce/Restore Volume levels.
6. Launch Music Player.

B. High-Level Actions

When the complexity of actions is increased such that the use becomes more specific and particular to the trigger, we introduce High level actions. These actions are used less frequently and are basically compositions of low level actions. Some of these include:

1. Sending email with auto-generated body.
2. Creating Alarm using calendar events.
3. Reading location and messaging coordinates.
4. Ignore an incoming call and send SMS automatically.

Such actions utilize more of device memory; hence they are used only with particular triggers so that the event frequency is reduced.

VI. TASKING FRAMEWORK

Tasking applications are based upon the tasking framework which handles the trigger receiving and action handling processes of the device. Basically, the triggers are divided into two categories viz. Sensor-based triggers and Broadcast-Receiver-based triggers. Triggers that are working upon device sensors directly are similar in their code structure, unlike the broadcast receiver based ones which stay idle and listen to the device state change. "Flip" trigger is an example of sensor-based one whereas "Low battery" trigger is a good example of broadcast-receiver-based trigger.Whenever a pair of trigger and action is selected in UI, the framework creates a rule, which starts the sensor service for the corresponding trigger. This job is done by Rule engine module in the framework. In this way, only the required sensors are used optimally and unnecessary battery drainage is avoided.

Actions and triggers contributed by developers are added to UI on phones for end users to use.The end user interface allows users to specify complex conditions (using the Activity composition framework) and build "Trigger-action" tasks. (Figure 2 shows a mockup of the end user interface).The UI generates scripts written using the developer programming model.

Fig. 2. End-user UI mockups for specifying Triggers and Rules.

Tasks are represented as JavaScript programs.KinoSense exposes phone devices as objects — a developer can simply call methods on the object to manipulate it. This object model is convenient because it enables a developer to work directly with higher level tasks. When a script creates Complex tasks,KinoSense partitions the task into sub-tasks, and executes and coordinates sub-tasks in the device.We explain our programming model using a single device tasking application.

A. Simple Task:

The following pseudo code shows a KinoSense script to express a simple application: "When Hari's device is out of Battery, then device automatically turns on the Silent mode, reduces the Brightness and turns off Wi-Fi."

```
Trigger = getDeviceStatus ();
If (Trigger is "BATTERY_LOW")
{
    Perform silentAction ();
    Perform wifiAction ("Off");
    Perform brightnessAction ("LOW");
}
```

Here, Trigger will get the device status and if it is "Low Battery" then KinoSense Framework will perform the given

Actions i.e. device will automatically turn on the Silent mode, reduce the Brightness of device display and turn-off the Wi-Fi.

B. Complex Task:

The following pseudo code shows a KinoSense script to express a Complex application: "When Hari gets an Incoming Call and he flips down the phone, the device turns into Silent mode automatically".

```
Trigger = getDeviceStatus ();
If (Trigger is “INCOMING CALL" AND “FLIP DOWN”)
    {
        Perform silentAction ();
    }
```

Here, if triggers received are both “Incoming Call” and “Flipped Down” at the same time, then device will automatically turn to Silent Mode.

VII. FUTURE SCOPE

Tasking applications are now trending in mobile application markets. Such applications let user specify context-aware tasks on the device. OnX [2] is an android application developed by Microsoft that primarily works on events on the device, but lacks the support of modular sets of triggers and actions which can be reused. KinoSense framework directly is able to provide a wide range of primary as well as complex tasks that can be further modified as per the developer’s needs. Hence, there is always some scope for adding and modifying the trigger and actions that are present ready-to-use in the framework. Some of the advancements that we wish to be added in future would be:

A. Cloud Hosting for Rules:

The rules that are created by the user according to his requirement can be hosted on the cloud, so that in case the user changes his device, he can download them and use them directly on new device. This provides flexibility to the framework on device-end and removes the limitation of local definitions of rules by the user.

B. Profiles:

A wide range of users are supposed to be using the app developed on KinoSense framework. Therefore the rules would be variable according to the user. A businessman would be more focused on triggers and actions like “In Meeting Room”, “Enable Wi-Fi“, “Silent Phone automatically”, etc. On the other side, a student would be more inclined towards using “Headset Connected”, “Shake Device”, “Music Action”, etc. We would be providing this feature to embed customization and personalization feature into framework. Profiles would be implemented by providing user specific templates.

C. Multi-device Tasks:

Going beyond single device context aware system, we can implement rules that would be working on more than one device. Consider a rule

```
If (“Hari’s device does X”)
{
    Perform Y on Riya’s device
}
```

Here, the trigger works on Hari’s device, and corresponding action is performed on Riya’s device.

VIII. CONCLUSION

We introduced KinoSense, a framework that eases the development and running of tasking applications on smartphones. The rules use sensor service efficiently at background and thus allows automatic “Detect and Perform” process in the device. Developers would be benefited by the framework to develop more complex utility apps by directly using primitives in their code. User would find it simple and handy to use device sensors for performing various events and actions. Hence, this coordination of sensing through the device benefits all types of customers. We are currently implementing the framework, and working parallel on developing apps based on it.

IX. ACKNOWLEDGMENT

We take this opportunity to express our deep gratitude towards all the people who have helped us for the completion of this paper successfully. The report is finished under the guidance of Prof.P.Desai. We would be very grateful to her for her help in the entire process. We also thank Smt. Kashibai Navale College of Engineering, Pune University for inspiring us to take up this paper.

REFERENCES

[1] Blackberry location-based reminder http://whenimclose.com/

[2] OnX. http://www.onx.ms/

[3] Lenin R., Arvind T., Hari B., Samuel M.;”Code In the Air”, MIT CSAIL, USA Jan 2012.

[4] Won-Jae Yi, Weidi J., Saniie J; “Mobile Sensor Data Collector using Android Smartphones”, IEEE August 2012

[5] Yong-Hua Cheng, Wen-Kuang Kuo, Szu-Lin Su; “An Android system design and implementation for telematics services”, IEEE ICIS 2010

[6] Sean J., Miguel A., Philip L, Nevine L.; “A General Architecture in Support of Interactive, Multimedia, Location-Based Mobile Applications”, IEEE Communications Magazine Nov 2006

Amruta Deshpande, Pursuing Bachelor of Computer Engineering from Smt. Kashibai Navale College of Engineering, Pune, Maharashtra. My current degree aggregate is 63%. I am working hard on our B.E. project titled KinoSense.

Apurva Bhoite, Pursuing Bachelor of Computer Engineering from Smt. Kashibai Navale College of Engineering, Pune, Maharashtra. My current degree aggregate is 65%. I am working hard on our B.E. project titled KinoSense.

Ashish Kalbhor, Pursuing Bachelor of Computer Engineering from Smt. Kashibai Navale College of Engineering, Pune, Maharashtra. My current degree aggregate is 77%. I am working hard on our B.E. project titled KinoSense.

Sandeep Mane, Pursuing Bachelor of Computer Engineering from Smt. Kashibai Navale College of Engineering, Pune, Maharashtra. My current degree aggregate is 66%. I am working hard on our B.E. project titled KinoSense.