EAI Endorsed Transactions
on Context-aware Systems and Applications

Research Article ICST.ORG

Design guidelines for rapid and simple context-aware mobile application development – an android case study

Hossein Shams[1],* and Kamran Zamanifar[1]

[1]Faculty of computer engineering, University of Isfahan, Isfahan, Iran

Abstract

Presenting a context-aware service and information is a key aspect of ubiquitous computing, but development of such applications is quite complicated. Context-aware applications should be able to obtain raw data from sensors, create high-level context information, detect the user's situation, and adapt the behavior of the application to the recognized situation. These complexities caused to reduce the impact of context-awareness in mobile computing while sensors of smartphones have made huge potential for developing context-aware mobile applications. In this paper, we explain some guidelines to overcome the existing obstacles by separating the context-aware application layers and make a loosely coupled connection between them. These guidelines will bring easy and rapid development, reusability of the code and flexibility for developers. Finally, we provide a case study example in the Android platform to demonstrate how the guidelines can be used in a real application.

Keywords: Context-aware computing, Context-awareness, Context-aware Application, Context-aware design guidelines, Context-aware development

Received on 19 June 2015, accepted on 25 July 2015, published on 05 November 2015

doi: 10.4108/_______________

1. Introduction

Mark Weiser [1] introduced the term of ubiquitous computing at the beginning of the 90s, and envisioned a world that processing devices becomes an integrated part of human life and computing recedes into the background of our lives. During the recent years, a lot of our peripheral devices equipped with the computational capabilities to act more intelligently. Smartphones are an obvious sample of such devices and will be performing the key role in the ubiquitous computing era. Advancement in processing units, memory capacity and sensors of smartphones in recent years, alongside that they are always accompanied their owners almost in everywhere and know a lot of information about the user, has made them an appropriate platform for hosting adaptable applications. These context-aware applications can perform some tasks in place of the user without interrupting him/her.

In the last two decades, context-aware computing has gained a lot of research attention and is regarded as an enabling technology for ubiquitous computing systems. An application is context-aware if it could be able to recognize the current situation, and adapts its behavior accordingly. The situation can be constructed from context clues that inferred from physical contexts like sensors that are embedded in the smartphone, or from logical contexts such as information that is available in the smartphone's operating system. Adding the context-aware capability to applications is not convenient task and it is inherently complex.

In order to simplify the development of context-aware systems, several programming frameworks and middleware infrastructures are introduced by researchers [2]. Nevertheless, context-aware management systems (CMS) are not widely used by developers, because they

*Corresponding author. Email: hos.shams@eng.ui.ac.ir are mainly focused on system level architecture and context management, while developers need simplicity and coding support in programming language level too. So developers need a design guideline or a programming style to use available APIs and bring context-awareness in their applications.

The aim of this paper is to introduce some guidelines for context-aware application developers to facilitate developing such applications and reduce the complexities. The proposed guidelines are going to separate situation definitions and context acquisition from the main business logic. This separation allows the developer to use a specific third-party CMS for acquiring context data or implement a simple CMS inside the application. Also, a separation of the situation definitions from the business logic brings a discipline to the process of development by allowing to utilize the situation definition domain experts [3] and increase code reusability.

The rest of this paper is structured as follows. In the next section, we will introduce some of related work in this area, and explain their drawbacks. In section 3 we will express some of the programmer problems in the application layer and introduce our guidelines to overcome these problems, then go on by describing our case study implementation based on the guidelines in section 4. Finally, conclusion and future work comes in section 5.

2. Related work

A considerable amount of research has been published on circumstance of providing context data through context-aware middleware. One of the first attempt in this area has been the Dey's Context Toolkit [4] and followed by several frameworks and middleware such as Java Context Awareness Framework (JCAF) [5], SOCAM [6], Hydra [7], etc. and several overviews have been published [2]. These approaches focused on implementation of the middleware and representation of context data and neglected to support developers to reduce complexity of using the contexts and adapt their applications to a specific situation.

Du and Wang [8] provided a programming model for facilitating the development of context-aware applications for mobile devices. Their programming model is a three-layered software architecture to support developers at programming level, which define contexts, behaviors and context-behavior binding rules in the XML-based specification. One of the weaknesses with this model is that it limits the programmer to use context-awareness only by the use of callback functions, and does not allow the programmer to utilize the situations in the conditional statements of the program.

Schuster et al. [9] introduced an approach to context-oriented programming for Android mobile devices. They explain how to use JCop language [10] in the development of Android applications. They introduced a declarative approach at the programming language level that uses context-oriented programming and pointcut-based activation of adaptation to reduce complexity of developing such applications. The main drawback of this approach is that both language definition and the compiler's bytecode generation of Android's architecture required to modification.

In this research, unlike the former studies that their main attention were on middleware layer or providing support for the programmer with some drawback in the development tools, we are going to concentrate on the application layer and the existence development tools and guide developers on how to use context middleware and defining the situations at the programming language level.

3. Design guidelines

In this section, we present the proposed design guidelines for developing mobile context-aware applications. The guidelines are based on our development experience and research on mobile application development and MVCC architectural pattern [11]. The guidelines can reduce development complexities and increase developing process speed by allowing to give this part to other developers and also brings code reusability by allowing to use the situation classes that previously created.

Determining and associating appropriate reactions to occurrence of situations, is the first step in developing context-aware applications. For example, suppose an application for handling incoming calls with the intent of rejecting unnecessary calls in some situations. Hence, determining the situations like driving, in a meeting, sleeping, resting, reading, and etc. are important to this application, because the application should decide to auto reject the incoming call or ringing the phone based on the caller person and the situation. To detect a situation, an application must determine which contextual properties are necessary for the situation and what value they should have. Contextual information is clues for the programmer to defines and recognizes the occurrence of a situation and this contextual clues can come from phone's sensors. For example, recognition of sleeping situation can be a combination of not moving for a period of time (using an accelerometer), a calm environment (using a microphone), a dim environment (using light sensor), and be at a reasonable time like between 11 PM until 6 AM. So feeding an application with enough context information can help developers to detect a situation more precisely.

Context-aware application needs a layer to collect contextual information from different sources and will be known as Context Data Adapter (CDP). This information can come from a separate part of the application that has the duty of preparing context data from different sensors like a CMS, or from a third-party CMS. Each CMS can have different communication protocol and diverse context modeling methods like key-value pair model, XML-based modeling, Ontology-based modeling (OWL),

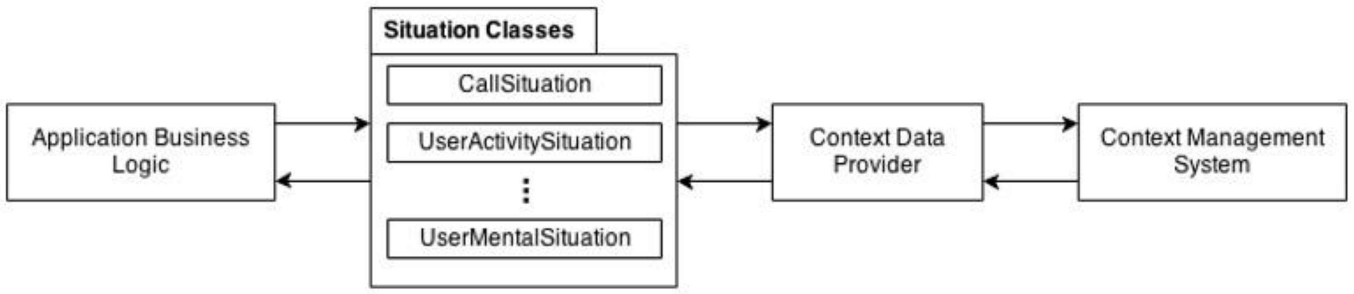

Fig. 1. Horizontal view of the guideline components and their connections.

Object-oriented models, etc [12]. The CDP should bridge the gap between the application and the CMS, and prepare context data to the application in uniform style. Changing the CMS can be done by just modifying the CDP code to adapt to new CMS.

In context-aware applications, recognition of a situation and associate it to a proper response, is a significant part of the source code. Using conditional statements for recognizing a situation is an unavoidable part of context-aware programming. These If-Then statements generally have long condition and contain several clauses and logical operator. The presence of these conditional statements in the several business logic files will cause hard-to-read programming style and brings several problems in the development process, code readability, code reusability, and software maintenance. Developers should separate the situation recognition codes from the business logic part to handle this chaos.

We suggest using situation classes to declare situations in a separate part of the project. Developers should create a class for each collection of related situations, and each situation should be defined by a Boolean method. These classes should subscribe to required contexts and update situations when the context get change. For instance, human's physical activities like driving, walking, sleeping, resting, and -reading can consider as a situation class, because they are related activities and will utilize a common set of contextual properties. The body of the situation methods contains several conditional statements that are a composition of context information for detecting if a situation is stable or not. By this separation of situation recognition from main application codes, we can reduce the complexity of the code maintenance and promote code reusability of the project.

Utilizing the situation methods should be as simple as possible. The developers would like to check for the establishment of a situation in their If-Then-Else conditions for handling the personalized behavior of the application based on the user situation. This could perform just by calling to the Boolean situation methods which defined in the situation classes. Also, developers need to perform an action right when a situation is occurring. So there still need a method that allows developers to run a callback function when a situation get change. By this method, developers are able to use the defined situations by just calling a method in a conditional statement or set a callback function on their occurrence.

The guideline has a high degree of loose coupling between the components. This loose coupling feature brings great code reusability and high flexibility in utilizing third-party CMSs that will cause to acceleration and convenience of the development process. Figure 1 shows the horizontal view of the components and their connections.

4. Implementation

For demonstration purpose, we implemented a context-aware application on Android. ContextPlayer is a context-aware music player that uses sensor data to play a customized playlist for a specific situation and adapt the application behavior to the user situation. For example, the application can play classical music for a period of time just before when the user gets asleep, or play Hip-Hop music when the user is doing exercises such as running. Even the ContextPlayer can adapt its behavior to the user situation, like stop playing music when the user has an incoming call or making an outgoing call. The

List.1. Subscribing to the changes of a context property.

```
attach("TIME.DayTime", "isSleeping", OnContextChangedListener() {
    @Override
    public void onContextChanged(String newValue) {
        if (newValue == "night")
            setNight(true);
        else
            setNight(false);
    }
});
```

List.2. Starting the CDP Service and listening for change in the sleeping situation of the user.

```
startService(new Intent(getBaseContext(), CdpService.class));
UserActivitySituation user = UserActivitySituation.getInstance();
user.sleeping().onChange(new OnSituationChangedListener() {
    @Override
    public void onSituationChanged(Object newValue) {
        boolean isSleeping = (Boolean) newValue;
        if (isSleeping)
            playListOf("classical", 30);
    }
});
```

hypothesis behind the development of the ContextPlayer is that the user's environment affects what kind of music should be played and what reaction should be performed.

The ContextPlayer has a simple internal CMS to collect required contextual information. The CMS contains a set of classes to access physical and logical context sensors, including TimeContext, CallContext, LightContext, MovementContext, etc. Each context class is an android service that contains a separate thread to monitor changes of the sensors in a specific period of time. If a context value got changed, the new value will broadcast to the CDP in a key-value pair model. The CDP will notify all of the classes that subscribed to receive the changes of that certain context and execute a callback function.

The ContextPlayer contains two situation class: UserActivitySituation and CallStateSituation. These classes inherited from a BaseSituation class that implemented the common methods of the situation classes and provided an observer and a singleton design pattern for them. Each situation class contains several properties for keeping context values, and Boolean methods that produce the situations from the context properties. Every situation class has a constructor that contains several calls to an attach method. The attach method will subscribe a callback function to the change of a context property, and get a list of situations that need to update after the modification of the context property. The code in List 1 shows that we are subscribed to DayTime property of TimeContext class, and the onContextChange method will execute when it get changed, and a notification will send to every callback function that listened to change of the "isSleeping" situation.

Using the situations in the main business logic is as simple as starting the CDP service, get an instance of a situation class, and listen to change of the situation by calling to the onChange method. The example in List 2 shows how we play some classical music for 30 minutes, when the user is going to sleep.

To conclude this section, we present a UML sequence diagram to show the task of each component and where each listener and situations get notified. Figure 2 illustrated this workflow.

5. Conclusion

This paper has explained some developers' problems in the development of context-aware applications. Then we have described several approaches to overcome these problems, and accelerate the development process. We proposed a guideline for separating the situation

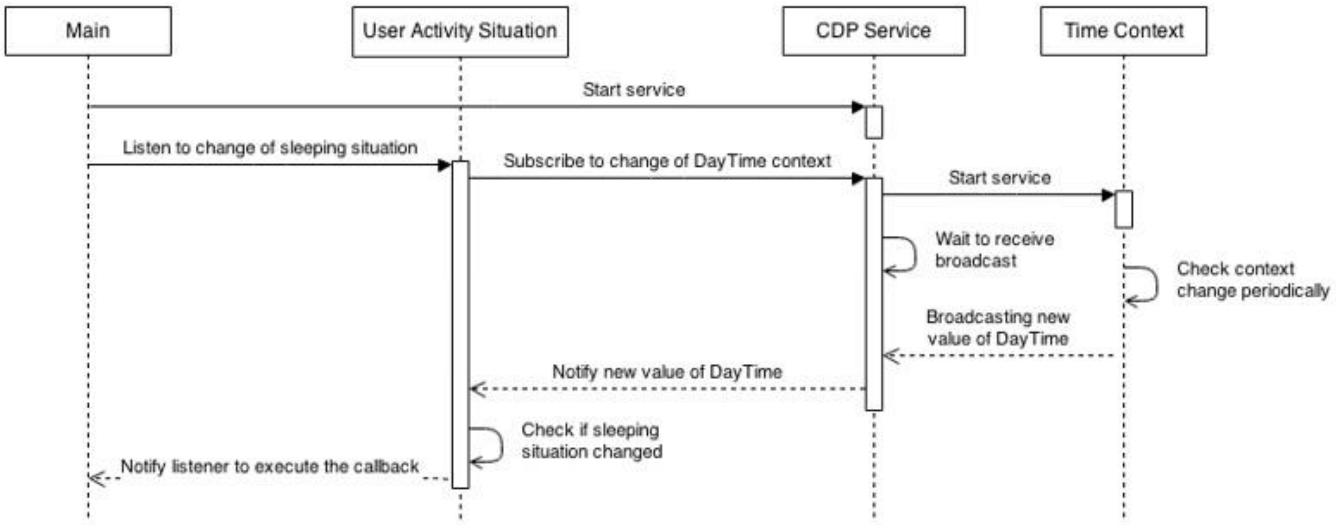
Main
User Activity Situation
CDP Service
Time Context
Start service
Listen to change of sleeping situation
Subscribe to change of DayTime context
Start service
Wait to receive broadcast
Check context change periodically
Broadcasting new value of DayTime
Notify new value of DayTime
Check if sleeping situation changed
Notify listener to execute the callback

Fig. 2. Workflow of executing a callback function, when the user situation changed to sleeping. To be simple, we just show subscription to the TimeContext service.

recognition codes from the main business logic codes. Minimizing of writing codes, acceleration and convenience of the development, reusability of situation classes, and high flexibility in utilizing third-party CMS are the results of this separation. For the future work, we can develop a context-aware framework based on these guidelines. This framework can implement common parts of the situation classes and services, and causing to reduce the size of the code that the programmers should write.

References

[1] M. Weiser, "The computer for the 21st century," Scientific american, vol. 265, pp. 94-104, 1991.

[2] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," International Journal of Ad Hoc and Ubiquitous Computing, vol. 2, pp. 263-277, 2007.

[3] D. Martín, D. L. de Ipiña, C. Lamsfus, and A. Alzua, "Situation-Driven development: a methodology for the development of context-aware systems," in Ubiquitous Computing and Ambient Intelligence, ed: Springer, 2012, pp. 241-248.

[4] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," Human-computer interaction, vol. 16, pp. 97-166, 2001.

[5] J. E. Bardram, "The java context awareness framework (JCAF)–a service infrastructure and programming framework for context-aware applications," in Pervasive Computing, ed: Springer, 2005, pp. 98-115.

[6] T. Gu, H. K. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services," Journal of Network and computer applications, vol. 28, pp. 1-18, 2005.

[7] T. Caus, S. Christmann, and S. Hagenhoff, "Hydra–an application framework for the development of context-aware mobile services," in Business Information Systems, 2008, pp. 471-481.

[8] W. Du and L. Wang, "Context-aware application programming for mobile devices," in Proceedings of the 2008 C 3 S 2 E conference, 2008, pp. 215-227.

[9] C. Schuster, M. Appeltauer, and R. Hirschfeld, "Context-oriented programming for mobile devices: JCop on Android," in Proceedings of the 3rd International Workshop on Context-Oriented Programming, 2011, p. 5.

[10] M. Appeltauer, R. Hirschfeld, H. Masuhara, M. Haupt, and K. Kawauchi, "Event-specific software composition in context-oriented programming," in Software Composition, 2010, pp. 50-65.

[11] H. Shams and K. Zamanifar, "MVCC: An Architectural Pattern for Developing Context-aware Frameworks," Procedia Computer Science, vol. 34, pp. 344-351, 2014.

[12] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, et al., "A survey of context modelling and reasoning techniques," Pervasive and Mobile Computing, vol. 6, pp. 161-180, 2010.