

Use of a Decision Tree to Represent Context Information

Sihan Cheng, Qian Xu, Mao Zheng
Department of Computer Science
University of Wisconsin-La Crosse
La Crosse WI, 54601
mzheng@uwlax.edu

Abstract

Our world gets more connected everyday. These connections are driven in part by the changing market of smartphones and tablets. Pervasive computing environments are fast becoming a reality. The term “pervasive”, introduced first by Weiser [1], refers to the seamless integration of devices into the user’s everyday life. One field in the wide range of pervasive computing is the so-called context-aware system. Context-aware systems are able to adapt their operations to the current context without an explicit user intervention and thus aim at increasing usability and effectiveness by taking environmental context into account.

We are interested in a context-based user interface in a mobile device: the mobile user interface will be automatically adapted based on the context information. The user interface can include many features such as font, sound level, data entry, etc. Every *feature* has some variables. For example, for the data entry, it has typing, voice and tapping. From the designer’s perspective, the adaptability of these features is planned either at the design time or during the runtime.

We use the decision tree to represent the adaption of mobile device user interface to various context information. The context includes the user’s domain information and dynamic environment changes. Each path in the decision tree, from the root to the leaf, presents an adaption rule. An e-commerce application is chosen to illustrate our approach. This mobile application was developed based on the decision tree in Android platform. The automatic adaption to the context information has enhanced human-computer interactions.

The e-commerce mobile app, the frontend, was linked to backend cloud storage, using the model of backend as a service (BaaS). BaaS providers form a bridge between the frontend of an application and various cloud-based backends via a unified API and SDK.

1 Introduction

Our world gets more connected everyday. These connections are driven in part by the changing market of smartphones and tablets. Pervasive computing environments are fast becoming a reality. The term “pervasive”, introduced first by Weiser [1], refers to the seamless integration of devices into the user’s everyday life. One field in the wide range of pervasive computing is the so-called context-aware system. Context-aware systems are able to adapt their operations to the current context without an explicit user intervention and thus aim at increasing usability and effectiveness by taking environmental context into account.

We are interested in a context-based user interface in a mobile device: the mobile user interface will be automatically adapted based on the context information. The user interface can include many features such as font, sound level, data entry, etc. Every *feature* has some variables. For example, for the data entry, it has typing, voice and tapping. From the designer’s perspective, the adaptability of these features is planned either at the design time or during the runtime.

We use the decision tree to represent the adaption of mobile device user interface to various context information. The context includes the user’s domain information and dynamic environment changes. Each path in the decision tree, from the root to the leaf, presents an adaption rule. An e-commerce application is chosen to illustrate our approach. This mobile application was developed based on the decision tree in Android platform. The automatic adaption to the context information has enhanced human-computer interactions.

2 A Context-aware Android App: E-Commerce System

There are two major platforms in the mobile device community: iOS and Android. This project chose Android development [2] mainly for the reason of its openness. In addition, all the tools in the Android development are free and no special hardware is required.

With traditional e-commerce applications, the user can browse the products, select a product and view the details. In the purchase process, the user will add the product to the shopping cart, enter or select payment options and enter a shipping address. From the application interface perspective, the inputs to the application are mainly through the user’s tapping, typing and clicking. The outputs of the application are in the forms of text, picture and video.

In our context-based mobile e-commerce application, the user interface will automatically adapt to the context information to improve the usability. We categorized the context information into two categories as shown in Table 1. We utilize the mobile device’s sensors to collect physical context information. The logical information is gathered through the user’s registration process. In addition, the mobile application’s input and output have additional forms: voice input and sound output.

Physical Context	Battery Level, Light, Noise Level, Wi-Fi, Network Speed
Logical Context	User Profile (age, gender, preferred input/output for the application, first time using the app or not) User's category (VIP or Non-VIP)

Table 1 Context Information Categorization

Note: VIP users are those who made more than 50 orders within three months, or users which purchased the membership to the system.

Some of example behaviors of our context-based mobile e-commerce application are listed below:

1. The user can search a product by simply talking to the device, or saying “check out” to enter the purchase stage.
2. If the user is using the app outdoors on a bright day, either the device will automatically adjust the screen brightness or sound out the product description.
3. If the device is currently running low on battery power or the device is not connected to a Wi-Fi signal, the app will display the product description in text format instead of picture or video forms.

3 Decision Tree

Our work depends on the internal sensors of a mobile device, the user profile and the adaption of the mobile user interface features for both entering and accessing data. The key point of the approach is to capture and represent the knowledge required for the mobile user interface to self-adapt at run time or to implement the adaptations at design time. The rule-based approach representation is what we are proposing. Figure 1 below shows our proposed approach.

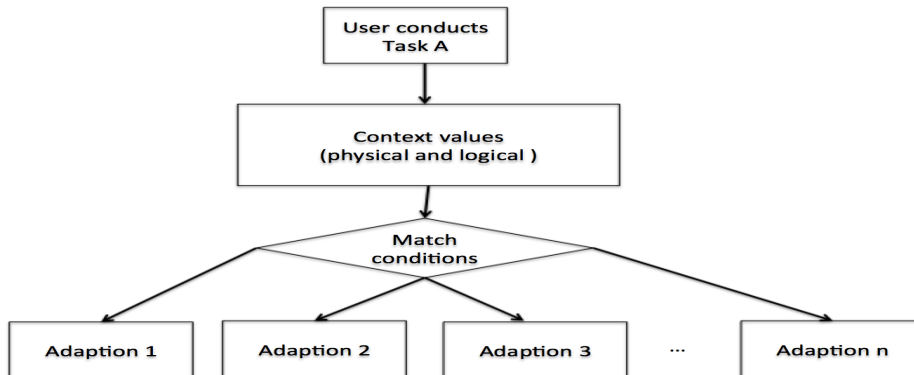


Figure 1 Rule-based Approach

Specifically in our approach, we are using the decision tree to describe and represent the adaption rules in the system. The decision tree is a graph that consists of nodes and edges. Each node represents a single or compound condition, and each edge represents the control flow. A path in the decision tree is the sequence of edges starting from the root node to a leaf node. Each path represents an adaption rule. In the decision tree, the priority of the conditions is shown as the position of the nodes in the tree. The higher of the node's position means the more important the condition. The system will check this condition first before moving to nodes found lower in the tree.

Figure 2 is the decision tree for our context-aware e-commerce application. The symbols used in the decision tree are explained in detail in Table 2. For example, if a user is a VIP user, he/she will have the option to change his/her user interface theme (f1). His/her screen will show a VIP account interface (f2) with many pictures of the products which are available for purchase (g1) for the user to browse. Sample screen shots are shown in Figures 3 and 4. In another scenario, if the device's battery level is high, but the device is not connected to a Wi-Fi signal, the product's video will not be presented (a2) regardless the speed of the network. If the network speed is low, the picture will not be presented (j2), otherwise the picture will be presented (j1).

Each path in the decision tree from root to the leaf represents how the app will automatically adapt to context information. However, the user is able to manually override the adaption: set the video, picture, sound or brightness according to his/her preference or for his/her special request/need at the particular time.

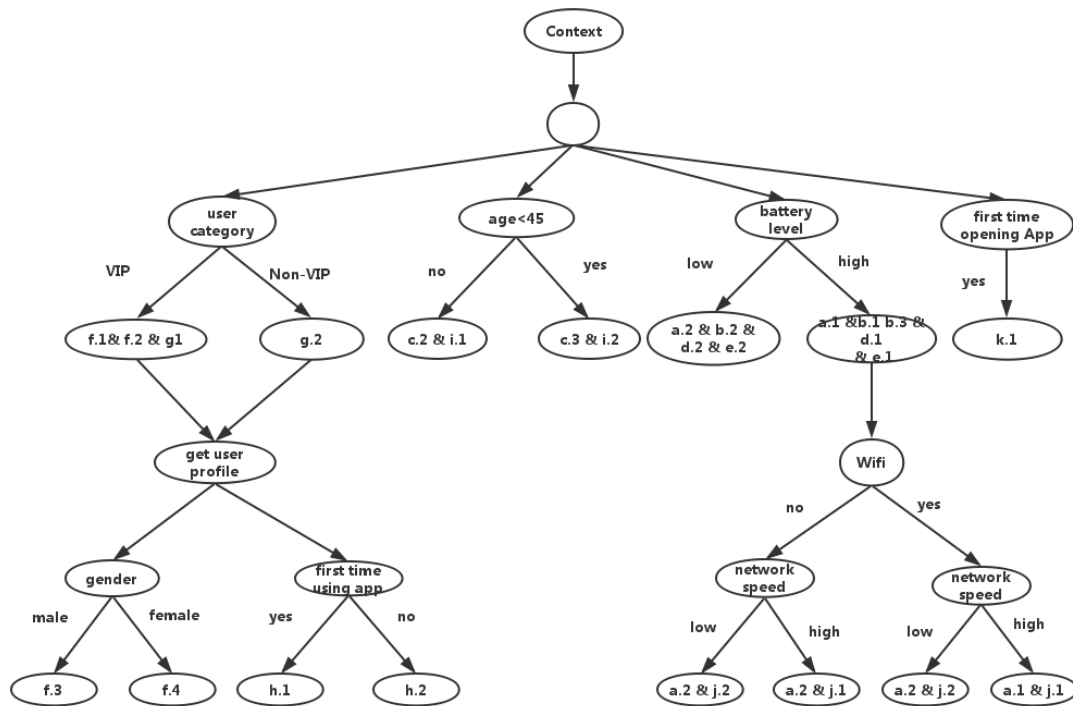


Figure 2 The Decision Tree for Context-aware E-commerce Application

GUI Features	Action
a.Video	1.Videos will be present
	2.Video will not be present
b.Media sound	1.Adjust the sound level to "sound on"
	2.Adjust the sound to "sound off"
	3.Adjust the sound to auto adjusted
c.Font	1.Adjust the font to "Medium"
	2.Adjust the font to "Big"
	3.Adjust the font to "Small"
d.Brightness	1.Adjust the brightness level to user preference
	2.Adjust the Brightness level to auto adjusted
e.Voice input	1.Enabled
	2.Not enabled
f. Background theme	1.Optional theme(Blue, Red)
	2.VIP account interface
	3.unchangeable color - Grey
	4.unchangeable color- Pink
g.Homepage	1.picture style
	2.plain text style(detailed classification)
h. Tutorial	1.display
	2.not display
i.Item Description	1.sound

	2.no sound
j.Picture	1.Present picture
	2.Not present picture
k.Welcome page	1.Present welcome page when opening the app
l.Voice Command	1.Enabled
	2.Not enabled

Table 2 The Legend of the Symbols used in the Decision Tree

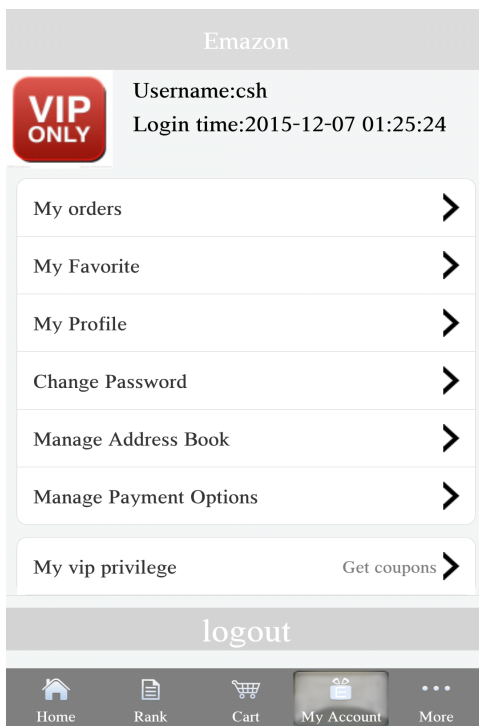


Figure 3 VIP User

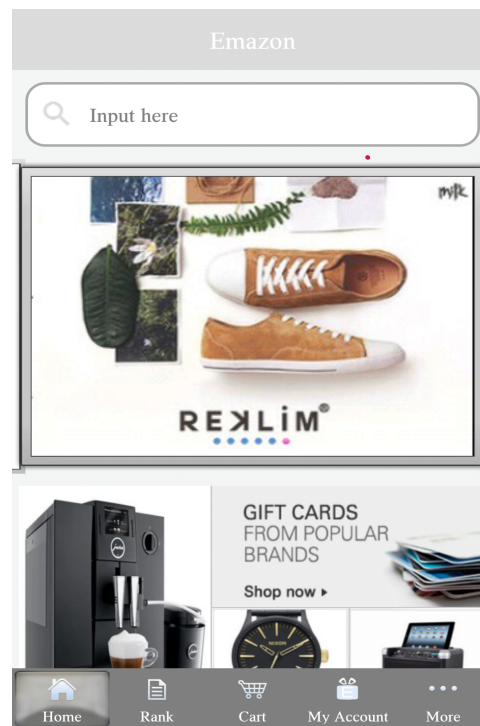


Figure 4 Picture Style for VIP User

4 Implementation

The implementation strictly followed the decision tree. Below is the sample code segment to check if the video will be present or not. The app will first check to determine if the user is able to watch a video in a given situation. The user's preference will override the context adaption. If not, the context adaption will start to check to determine if the mobile device is getting a Wi-fi signal, the network speed, etc.

```
public boolean getVideoSetting() {
    if (!video) // if video is closed
        return false;
}
```

```

if (getBatteryLevel() < 20) { // if battery level <20 %

    // video is open, not present video when low battery
    if (video && !lowb_video) {
        return false;
    }

    // video is open, present video when low battery
    if (video && lowb_video) {

        // get the wifi status
        String wifiStatus = getNetworkStatus();

        // if video is not present under 3G/4G mode , return false
        if (!data_video) {
            if (!wifiStatus.equals("WIFI"))
                return false;
        }

        // present video when no wifi environment, then consider the
        // network speed

        // else , present video when no wifi environment, then consider the
        // network speed, read the testing case first. If it is true, then
        // speed is 19
        int networkSpeed = 0;
        boolean testingNetwork = readBoolean(Constant.TESTING_DOWN);
        if (!testingNetwork) {
            String speed = readData("speed");
            // if no avg speed tested data found, set it to 21 kb/s
            if (speed.length() > 0 && !speed.equals(""))
                networkSpeed = Integer.parseInt(speed);
            else
                networkSpeed = 21;
        } else {
            networkSpeed = 19;
        }

        if (lowSpeed_video)
            return true;
        else {
            if (networkSpeed < 20) {
                return false;
            } else
                return true;
        }
    }
}
return false;
}

```

5 Testing

It is difficult to test a mobile device with different battery levels, network speeds, and environment noise levels in a real environment. We designed and implemented a testing

simulation to check each path of the decision tree. It simulates low battery, low network speed or high environment noise cases.

In order to detect the network speed of the mobile device, the mobile device has to conduct certain network activity. Otherwise a value of zero will be shown as the network speed. However, continuously making network activity will consume the user's data plan and will affect the device's performance. For these reasons, we tested network speed only when the app started. The specific test is to let the app download a small size picture and then we calculated the average network speed by dividing the picture size by the downloading time. If the network is not available, the testing activity for network speed will not be conducted. Figure 5 is a screen shot of the average network speed testing.

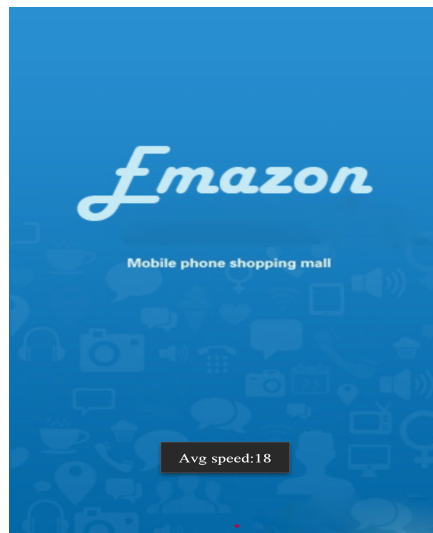


Figure 5 Testing Average Network Speed When the App starts

6 Conclusions

With ubiquitous computing, users access their applications in a wide variety of environments. To cope with various and dynamic execution environments, the adaptive mobile user interface is desired to enhance human-computer interactions. This paper discusses the design of the context sensitive mobile user interface that will enable automatic adaptations to the environment. The adaption built into a mobile user interface can enhance the accessibility in the e-commerce domain.

The future work of this research will fall into two directions: 1) discovering and verifying the completeness of the conditions and rules. 2) building a context model and reconfiguring the model for other applications.

References

- [1] Weiser, M. "The computer for the 21st century", Scientific American, 1991

pp. 94-104.

[2] Android Developer's Guide. <http://developer.android.com/guide/index.html>

[3] Dey A. "Providing Architectural Support for Building Context-Aware Applications", Ph.D. thesis, College of Computing, Georgia Institute of Technology, Dec. 2000.

[4] Derek Riley, Using Mobile Phone Programming to teach Java and Advanced Programming to Computer Scientist, ACM Special Interest Group on Computer Science Education SIGSCE 2012, pp:541-546. Feb. 29-March 3, 2012.

[5] B. N. Schilit, N. Adams, and R. Want. Context-aware Computing Applications. In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp. 85-90, Santa Cruz, CA, Dec. 1994. IEEE Computer Society Press.