Simplifying Mobile Application Development with Model-View-Controller

S. A. M. Rizvi1, Syed Zeeshan Hussain2 and Syed Imtiyaz Hassan3
1,2Department of Computer Science, Jamia Millia Islamia University, New Delhi.
3Department of Computer Science, Hamdard University, New Delhi.
samsam_rizvi@yahoo.com, szhussain@rediffmail.com and s.imtiyaz@yahoo.com

ABSTRACT
Mobile is getting more popular these days. To facilitate more and more features it is also getting more complex. Additional requirements are to be considered while developing mobile applications in relation to applications for desktop computers. These requirements primarily concern the support to different platforms on which such applications are performed, as well as the requirement for providing more modalities of input/output interaction. Unlike PCs, people consider mobile devices to be accessories, so, rich device design and presentation capabilities are needed. Such system will be easier to design if design pattern is integrated with object oriented method of software engineering. With the help of pattern a system can be designed for change. The present paper explores and demonstrates the use of Model-View-Controller (MVC) pattern known for promoting orthogonality, reusability and usability in interactive desktop applications by implementing a graphical mobile devices application on the J2ME platform. The authors believe that the proposed idea will help mobile developers to write better re-usable code and also enhance the overall usability.

KEYWORDS
Mobile Application, MVC, Design Pattern, Re-usability, Usability.

1. INTRODUCTION
Recently, the mobile gaming industry has experienced a tremendous growth. Mobile Industry demands extra ordinary features to compete in market, that makes mobile software complex. At the same time it is also required to reuse these features for various models of mobile.
Several problems can arise while developing such applications containing mixture of data access, business logic, and presentation code. Moreover, mobile devices tend to run their own versions of virtual machines and operating systems, making them prime candidates adapting core application to variety resources with different capabilities. These requirements primarily concern the support to different platforms on which such applications are performed, as well as the requirement for providing more modalities of input/output interaction [1]. Unlike PCs, people consider mobile devices to be accessories, so, rich device design and presentation capabilities are needed. These requirements have influence on the user interface and therefore it is needed to consider the usability as well as reusability. The core logic of such devices shall be reusable for multiple type of interfaces as devices are not standardized. Traditional structured or object oriented methods are not able to address these issues alone, as they offer limited means of reasoning about the validity of specifications. Such system will be easier to design if design pattern [2] is integrated with object oriented method of software engineering. A design pattern describes micro-architecture within a broad architectural picture which describes a particular recurring design problem, proposes a predefined scheme for its solution, and includes heuristic rules for how and when to use it [1]. In other words, a pattern is a form of solution intended to address problem and the purpose of the pattern is to reduce the complexity of user interfaces for a large and complex information system [3]. A design pattern (Formal method) makes one think very hard about the system one proposes to build [4]. With the help of pattern a system can be designed for change. In mobile development process use of design patterns can be explored so that applications for mobile devices can be deployed to a wide variety of handsets. The present paper explores how a mobile application can be developed with the help of a well accepted design pattern, Model-View-Controller (MVC) [5] known for promoting orthogonality, reusability and usability in interactive desktop applications. The present paper also demonstrates the use of MVC to develop graphical mobile devices application on the J2ME [6] platform.

2. ORIGIN OF DESIGN PATTERN
Design patterns are a modern software engineering problem-solving methodology that emerged from the object-oriented developer’s community. It has roots in many disciplines, including software engineering. The foundation of software design patterns is ascribed to the work of Christopher Alexander [7]. As a building architect, Alexander noted the presence of common problems and related solutions within a given context. A design pattern, as Alexander termed this problem/solution/context triad, enabled an architect to rapidly address issues in a uniform manner during building design [8]. But the concept of software design patterns was first introduced by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, also known as Gang of Four (GoF) in the year 1995 with the book [2]. The design pattern is the excellent design method and software based on patterns could easily deal with specific changes to achieve the software reuse [9]. Design patterns are providing generic solutions to commonly occurring
problems in software development and describe a design problem, the solution and the consequences.

3. MVC AND ITS BENIFITS
The MVC pattern consists of three components: model, view, and controller. The model represents the data and business logic of the application. The responsibility of a model is to handle the data, to make any necessary calculations, and to notifying all the views registered to that particular data of any changes. The view renders data and related commands to the user. Some applications offer multiple views of a given model, or even of the same data. The controller receives user events and dispatches them to the model. When the user modifies the data on the view and then it is the controller that determines what kind of action has been requested and calls the appropriate interfaces of the model. A major advantage of adopting the MVC pattern is that user interface is flexible enough for change and hence look and feel of the user interface can be easily modified. Change or upgrade of the client side technology such as Flash and mobile devices can be implemented with minimal rework. In other words MVC promotes orthogonality, reusability, multi-user interface rendering and consistency.

4. IMPLEMENTING MVC IN MOBILE
While mobile application can be developed using different technologies such as Windows mobile platform, Flash, Python, among others, J2ME platform has been considered. J2ME is a leading platform in for mobile devices [10]. The sample application below simply implements a system that can authenticate a user. Based on user name and password it simply displays alerts whether user is correct or not. For this purpose the system is factored into three components Model, View and Controller. There are one model, One Controller and one view.

Code snippet 1: Model.java

```java
//Model.java
package mobile;
public class Model {
    public Model()
    {
    }
    // Holds value of property userName.
    private String userName;
    // Holds value of property password.
    private String password;
    // Setter for property password.
    public void setPassword(String password)
    {
        if (password.equals("password"))
        {
            this.password = password;
        }
        else {
            this.password = null;
        }
    }
    //Validate user
    public boolean validate() {
        if (userName == null || password == null) {
            return false;
        }
        return true;
    }
}
```

Code snippet 2: View.java

```java
//View.java
package mobile;
import javax.microedition.lcdui.*;
public class View extends Form {
    //Text Fields for userName and password
    private TextField user = new TextField("USER NAME", ",", 20, TextField.ANY);
    private TextField pass = new TextField("PASSWORD", ",", 20, TextField.PASSWORD);
    //View constructor
    public View(Model model) {
        super("Login");
        this.model = model;
        this.append(user);
        this.append(pass);
    }
    //Retrives user name from text field
    public String getUser() {
        return user.getString().trim();
    }
    //Retrives password from text field
    public String getPass() {
        return pass.getString().trim();
    }
}
```
private Display display;
private boolean midletPaused = false;
private Command exitCommand = new Command("Exit", Command.EXIT, 1);
private Command okCommand = new Command("Login", Command.OK, 2);
private Alert alertSuccess;
private Alert alertFailure;
private Model model = new Model();
private View v = null;

public void commandAction(Command c, Displayable d) {
    if (c == exitCommand) {
        exitMIDlet();
    } else if (c == okCommand) {
        model.setUserName(v.getUser());
        model.setPassword(v.getPass());
        if (model.validate()) {
            switchDisplayable(getAlertSuccess(), getView());
        } else {
            switchDisplayable(getAlertFailure(), getView());
        }
    }
}

public void startMIDlet() {
    // write pre-action user code here
    switchDisplayable(null, getView());
}

public void switchDisplayable(Alert alert, Displayable nextDisplayable) {
    if (alert == null) {
        display.setCurrent(nextDisplayable);
    } else {
        display.setCurrent(alert, nextDisplayable);
    }
}

public Display getDisplay() {
    return Display.getDisplay(this);
}

public View getView() {
    if (v == null) {
        v = new View(model);
        v.addCommand(exitCommand);
        v.addCommand(okCommand);
        v.setCommandListener(this);
    }
    return v;
}

//Failure Alert
public Alert getAlertFailure() {
    if (alertFailure == null) {
        alertFailure = new Alert("alert", "Wrong username or password", null, null);
        alertFailure.setTimeout(Alert.FOREVER);
    }
    return alertFailure;
}

//Success Alert
public Alert getAlertSuccess() {
    if (alertSuccess == null) {
        alertSuccess = new Alert("Logged In", "Successfully logged in", null, null);
        alertSuccess.setTimeout(Alert.FOREVER);
    }
    return alertSuccess;
}

Code snippet 3: Controller.java

The above program is developed in NetBeans 6.9.1 IDE [11] and Device configuration Connected Limited Device Configuration (CLDC) version 1.1 [12] and Device profile Mobile Information Device Profile (MIDP) version 2.1 [13]. The output is generated by DefaultFxTouchphone 1 of Netbeans 6.9.1. The code in snippet 1, 2 and 3 depicts only the code required to understand the paper, other code that is generated by IDE has not been included.

Sample outputs are depicted in figure 1 to figure 3.

Due the factoring of application in three components: 1) Model class is independent of view and controller. Hence, Model can be changed without affecting other components. Similarly, one can add another view without affecting model. Hence, a developer of applications for mobile devices can target same model for different screen size, shape, and orientation in order to deploy to a wide variety of handsets.

5. CONCLUSION

Using MVC in collaborations with object-oriented abstraction has proven to be very useful in desktop and web applications. Design pattern approach of software development provides a guide to develop factored systems that facilitate design reusability. Recently, the mobile industry has practiced an incredible growth. As a result, development of mobile application may lead to recurring similar design problems. These, design problems can be addressed by using MVC design pattern. The present paper explored the applicability of MVC design pattern in mobile application by implementing a sample
application using J2ME. It has been observed that MVC patterns accommodate for changes without breaking up the overall architecture and allow modifications to be made without opening up too many classes.

Fig. 1: Input Screen

Fig. 2: Correct user message

Fig. 3: Unsuccessful login

6. FUTURE SCOPE
MVC has proven to be an important design pattern for facilitating the development, debugging, and maintenance of systems. While it was originally intended develop GUI systems, it can be successfully applied to mobile applications as well. The use of MVC for developing pervasive applications that can support multiple mobile devices as well as personal computer may be the further research direction.

REFERENCES
[9]. Chen Liyan, “Application Research of Using Design Pattern to Improve Layered Architecture”, in proc. of International Conference on Control, Automation and
Simplifying Mobile Application Development with Model-View-Controller


Model View Controller is one of the most widespread pre-sentational architectural patterns, being used to create desktop, mobile and web applications [20]–[22]. The purpose of MVC is to provide a simple separation of concerns for an application that embeds a user interaction component. One of the most important aspects of this pattern is that the application should work, and fulfil all its requirements even if we remove the View and Controller layers. At the centre of every iOS application, we find the View Controllers which act as bridges between the data of the application (Model) and the user interfaces (View). Advantages This flavour of MVC simplifies (from the classic MVC) the data flow between the layers, making the data flow clearer.