Journal of
Information Technology & Software Engineering

Shahbudin and Chua, J Inform Tech Softw Eng 2013, 3:3
DOI; 10.4172/2165-7866.1000122

Research Article Open Access

Design Patterns for Developing High Efficiency Mobile Application

Fadilah Ezlina Shahbudin and Fang-Fang Chua*

Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia

Abstract

With the advent of technology, mobile devices are becoming powerful and they are widely used as convenient computing devices. In order to develop high quality application in a short period while overcoming the limitations and challenges in mobile development, strong knowledge and understanding of the design purpose are the key factors in developing well- designed applications. Good design ensures that crashes or harmful actions can be prevented as well as increasing users' satisfaction. By implementing design patterns, developers can cater wide variety of users whereby they do not have to redevelop similar application for users who are using different types of devices. Application can be developed in a short period as developer can reuse design patterns to develop new application in similar domain. Our proposed work aims to implement design patterns to overcome limitations of current techniques and increase efficiency, usability and reusability in mobile application development.

Keywords: Design patterns; Mobile application; Efficiency; Usability; Development

Introduction

Mobile devices with sophisticated functionalities and applications have changed people's life. There are many organizations and individuals leaning towards mobile application development. Understanding of the design purpose plays important role in developing well-designed application. Design choices affect the quality of application and developers' design decision will have a significant impact on the applications. For example, the implementation of layout, graphics, and animation will have performance implications. Defining the core building blocks of application encourages reusability. Therefore, the design and implementation of a set of components can be optimized. Building the most appealing design is not the only goal in mobile development as the application must not only attract users but also to achieve balance in terms of functionality, aesthetics, usability and performance. Good design not only eliminating users' dissatisfaction, but it can prevent crashes or harmful actions. Hence, developers need to take into account different aspects when designing mobile application. The design used in mobile application influences how the application performs. Mobile applications need to be fast and reliable in order to be valuable in the dynamic environment. However, limitations of the medium impose significant challenges to design application that can meet all of those expectations. As architectural design plays a key role to overcome those constraints, there is a need for an improvement of the design patterns applied in mobile application development.

In this paper, we aim to identify and analyze architectural or design patterns for mobile application development, implement the design patterns in mobile application, evaluate and verify the effectiveness. In order to increase efficiency, usability and reusability, design patterns for mobile application development are proposed and design patterns are implemented in Android application. The paper is organized as follows: Section 2 outlines the problem statements, and Section 3 gives an overview of Mobile Computing and Mobile Application Development. Section 4 explains the Significance of Design Patterns in Mobile Application Development followed by an Analysis and Implementation of the proposed Design Patterns in Section 5. In Section 6, we provide the evaluation of the implementation results. Section 7 concludes the paper and we provide an outlook on future work in Section 8.

Problem Statements

Developing mobile application is a challenging task as there are many aspects and factors that need to be considered to achieve the specified quality attributes. Problems identified as below are related to mobile application development and these problems are the key motivation for our proposed work.

Fast evolution of mobile devices

Mobile devices are evolving in a fast pace. Various types of mobile devices are available in the market and they are differing in terms of sizes, display resolutions, operating systems, processor speed, memory size, and battery life. Hence, it is very difficult to design for different devices in a short time to market. With the use of design patterns, similar approach can be implemented to develop mobile application in the same domain thus reducing the time, cost and effort needed to develop well-designed application.

Mobile constraints

Mobile devices have limited computing capability and resources. Hence, complex application which consume large amount of resources could not run well on mobile devices. Developing mobile application differs from developing desktop or web-based application. For example, an application for Mac OS cannot be ported to iPhone. This is due to the difference of computing and processing capability in the mobile devices and desktop. However, similar application for the desktop can be developed for mobile application using the right approach. The use of design patterns has shown a significant impact in desktop application. Thus the same approach can be used to develop mobile application and improve the quality of mobile application.

Low efficiency application

In mobile application, efficiency can be categorized into time efficiency (i.e. consumption time and network time) and resource

***Corresponding author:** Fang-Fang Chua, Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Malaysia, Tel: +60-3-8312-5406; E-mail: fang2x81@gmail.com

Received October 09, 2013; **Accepted** October 30, 2013; **Published** November 16, 2013

Citation: Shahbudin FE, Chua FF (2013) Design Patterns for Developing High Efficiency Mobile Application. J Inform Tech Softw Eng 3: 122. doi:10.4172/2165-7866.1000122

efficiency (i.e. memory consumption, battery consumption and CPU consumption). Mobile application should respond and use resources appropriately when performing its function. Low efficiency application may cause problems when accessing the application such as slow load time, crash, or froze. The use of design patterns can improve the efficiency of mobile application by implementing heavy weight functionality to be executed on the server and allowing the client applications to invoke the functionality in the server.

Unstructured design

Unstructured design such as code duplication may result in the increase of software size, difficulty to maintain, and poor design. This might happen specifically when developing for interactive application which requires multiple types of user interface. Developing such application requires thorough consideration on the design factors as it might result in complexity problem. Poorly-designed application may consume more resources and slow down or block the device usage. Besides, poor designs make testing and maintenance activities becoming difficult.

Mobile Computing and Mobile Application Development

Mobile computing systems can be defined as "computing systems that may be easily moved physically and whose computing capabilities may be used while they are being moved" [1]. Examples include laptops, personal digital assistants (PDAs), and mobile phones. Mobile computing has changed the way computers are used. In fact, it is expected that many devices will become smaller and even invisible in future. Technologies improvements in certain areas such as in Central Processing Unit (CPU), Memory, Screen, Touch-screen interface, battery, and wireless have driven the rapid advances in mobile computing. Advances in hardware technology aligning with the current trends in web-based computing has led to a reduction in costs, thus increasing the availability of mobile computing paradigms. While the concept of mobile computing is well established, the research area and industry for Mobile Application Development has gained a lot of attention. However, there are many challenges and limitations that need to be considered as developing mobile application differs from desktop or web-based application development. Many mobile applications available today provide different services and functionalities while previously, mobile apps were developed mainly to support productivity (i.e. email, calendar and contact databases). With the increasing demand and high user expectations, application such as mobile games, context-aware and location-based services, banking, and e-commerce have emerged. In fact, today mobile devices are considered as computers first and phones second as described by Hayes [2].

There are different approaches of mobile application development. Hence, developers should know whether they want to deploy a native application, web application or hybrid application as different platform have different development requirements. Table 1 summarizes native application development for iOS, Android, BlackBerry and Windows Phone.

As we can see from Table 1, iOS, Android, BlackBerry and Windows Phone have different requirements for their mobile application development. Designing and developing mobile application is a challenging task. While taking into account the constraints, the application must achieve high level performance and usability. In order to develop well-designed application and ensure that requirements are met, developers need to consider different aspects in designing mobile applications [3] suggested six design considerations for mobile application which are:

1) Decide type of application (Native, Web, and Hybrid)
2) Determine type of device to be supported (Screen size, resolution (DPI), CPU performance characteristics, memory and storage space)
3) Consider limited-bandwidth scenarios (Hardware and software protocols based on speed, power consumptions and not just on ease of programming)
4) Design UI appropriately, take into account platform constraints (Simple UI design and architecture, and keep other specific design decisions in mind)
5) Design a layered architecture appropriate for mobile devices (Apply layered architecture to maximize separation of concerns, an improve reusability and maintainability)
6) Consider device resource constraints (i.e. battery life, memory size, and processor speed)

While taking the above mentioned consideration as a guideline in designing and developing mobile applications, developers must be able to tackle the challenges in developing mobile application as mentioned [4,5] . These challenges include:

- Wireless communication issues (availability and disconnection, bandwidth variability i.e. low or high, heterogeneous networks, and security risks)
- Mobility issues (address migration, location-dependent information, migrating locality)
- Portability issues
- Various standards, protocols and network technologies
- Limited capabilities of terminal devices (factors pertaining to low power, risks to data integrity, small sized user interfaces, and low storage capacities)
- Special privacy and customizability needs
- Strict time-to-market requirements

	iOS	Android	BlackBerry	Windows Phone
Languages	Obj-C, C, C++	Java (Some C, C++)	Java	C#, VB.NET, etc
Tools	Xcode	Android SDK	BB Java Eclipse Plug-In	Visual Studio, Windows Phone Dev Tools
Executable Files	.app	.apk	.cod	.xap
Application Stores	Apple iTunes	Android Market	BlackBerry App World	Windows Phone Market

Table 1: Summary of Native Application Development.

Different types of mobile application have different specifications. Our focus is to provide solution by identifying and analyzing design patterns and implement them in a mobile application to further improving the quality of mobile application in terms of efficiency, usability, and reusability. Since they are wide variety of mobile devices available in the market, we need to counteract the limitation and challenges in designing and developing mobile application by providing a solution to cater wide range of audience while keeping pace with the evolving mobile technologies.

Significance of Design Patterns in Mobile Application Development

Design Patterns have wide variety of usage and they have been used and tested in practice. They are proven to be effective in software development to simplify the overall application design. Design patterns make the software more reusable which can lower the production cost and reduce development time. Design patterns are very useful for developers and designers as they encapsulate experience, provide a common vocabulary, and enhance the documentation of software designs [6]. There are wide varieties of mobile devices available in the market with various sizes, display resolutions, operating systems, processor speed, memory size, and battery life. Hence, developing mobile application is a challenging task as developers need to take into accounts the boundaries and challenges.

Patterns for UI design are also emerging as UI is one of the most important aspects in designing mobile application. There is also design patterns used to predict user behavior as context aware such as Recommender application or location based services. Developers and designers are looking for ways to adapt the design patterns in mobile application development. Few researches have shown that some patterns can be used for mobile platforms in a very similar way as in classical architecture. In fact, some design patterns have been used in platform such as Android and iOS. For example, in Android, the Media Player Service class implements the Factory Method to create different types of concrete media players, Activity class in Android development uses the Template Method pattern, Intent uses the Command pattern and Cursor, Adapter and the Observer pattern is used for View classes. Android View and Widgets are implementations of Composite pattern.

Apart from that, Model View Controller (MVC) pattern was also being extended in Android development. The MVC is combined with Decorator pattern in Controller and the Strategy pattern in between the Controller and the View. The Observer pattern is applied so that the associated Views are notified when the Model changes without coupling the Model and View. Furthermore, MVC is also combined with Factory Method to create multiple Views and multiple Controllers. The extended version of MVC is modified to support dynamic properties, avoid complexities and improve flexibility. Since most devices come in multiple screen sizes and display resolutions, these flexibility needs to be adopted in mobile application development in order to enhance the look and feel, and improve the usability of the application.

Design patterns have also been used as the fundamental design in Cocoa development. Cocoa is an application environment for Mac OS X operating system and iOS, the operating system for Multi-Touch devices such as iPhone, iPad, and iPod touch. Command pattern is used in Cocoa for undo management and distributed objects. The purpose of the pattern is to make operations undoable. The pattern also describes the target-action mechanism of Cocoa in which user-interface control objects encapsulate the target and action of the messages they send when users activate them [7]. In fact, most of the design patterns used in Cocoa development (i.e. Abstract Factory, Adapter, Chain of Responsibility, Decorator, Facade, Iterator, Observer, and Proxy) are cataloged by Gamma et al [6].

Cocoa development also implements MVC patterns and it is the most pervasive design pattern used in the design of several technologies, including bindings, undoes management, scripting, and the document architecture. The MVC version in Cocoa is a combination of several patterns which includes Composite patterns in the View objects, Strategy pattern between Controller objects and View objects and Observer pattern in the Controller object. Table 2 shows different design patterns implementation used in various mobile platforms [8] applied balanced MVC Architecture for Developing Service-based Mobile Application by devising and adopting three architectural principles; being thin client, being layered with MVC, and being balanced between client side and server side. MVC were extended whereby client and server system embody its own separate layers. The authors present patterns of mobile application architectures by adopting MVC and client server architectures which consider efficiency as a quality attribute for designing mobile application. Time efficiency and resource efficiency are the two key factors for well-designed mobile application architecture. Biel et al [9] introduced five patterns for development of mobile applications running on mobile devices without accessing remote logic or data storage. The authors focused on improving the usability of mobile application for Android platform. Despite the fact that some design patterns have shown a significant usage in mobile application development, there are also design patterns that are not applicable for mobile application as they cannot fit in the design scenario of mobile application. For example, Singleton pattern is not applicable for light weight mobile application.

Analysis and Implementation of Proposed Design Patterns

Analysis

In order to identify the design patterns, we follow Object-oriented Analysis approach to study the existing design patterns to investigate whether they can be reused or adapted for mobile application development. Next, we analyzed them by following the guideline which has been used by most of the developers and designers. The design patterns format as shown in Table 3 is a template which describes the characteristics of the design patterns. The proposed template follows the template described by [6] with few modifications.

Platform	Design Patterns
iOS	Abstract Factory, Adapter, Factory Method, Template Method, Chain of Responsibility, Command, Observer, Composite, Decorator, Facade, Iterator, Mediator, Memento, Proxy, NSProxy, Receptionist, Singleton, Template Method, MVC
Android	Patterns such as Factory Method, Template Method, Command, Observer, MVC
BlackBerry	Patterns such as Factory Method, Template Method, Command, Observer, MVC
Windows Phone 7	MVC, Model View View-Model (MVVM), View-View Model Pairing, Model-View-Presenter (MVP)

Table 2: Design Patterns used in various platforms.

Part	Description
Pattern name	Design pattern name
Intent	The objectives or purposes of the pattern and the problem it solves
Motivation	A concrete scenario that illustrates a design problem and how the pattern solves the problem.
Applicability	Describe the situation in which patterns are applicable.
Structure	Provide a graphical representation of the classes in the pattern using object notation such as UML
Participants	Indicate the classes and objects that participate in the pattern
Collaborations	Indicate the collaboration of participants
Implementation	State guidance on the implementation of the pattern

Table 3: Design Patterns format.

The identified design patterns should be applicable for mobile application based on three quality attributes which are:

1) Efficiency –The capability of the application to exhibit the required performance with regards to the amount of resources needed.
2) Usability –The usability properties that exhibit the ease of the use of the application.
3) Reusability – The extent to which the design patterns could be reused for new application within similar domain.

Due to limited resources and processing capability of mobile devices, design pattern can be used to improve the efficiency of mobile application by ensuring that the amount of resources and processes are handled efficiently. As usability is also an important characteristic for mobile application, we also need to implement design patterns so that aspects such as screen size, resolution (DPI), and elements are handled properly. Design patterns are useful for application which supports different looks and feels (i.e. different appearances and behaviors for UI elements like scroll bars, windows, and buttons).

The proposed idea separates the presentation and the application code (i.e. Event handling, initialization and data model, etc) by implementing design patterns such as Model View Controller (MVC) which allow us to develop application with loose coupling and separation of concern. Instantiating look-and-feel for specific classes of elements throughout the application makes it hard to change the look and feel later. This results in poor design and makes it difficult to be tested and maintained. Hence, the idea of design pattern is to make sure that the modules or objects are loosely coupled. Each module only makes use of little or no knowledge of other modules, so that changes can be made easily without affecting other modules. The concept of separation of concern is to divide the modules into distinct features with as little overlapping in functionality as possible. With the limitation and challenges in designing and developing mobile application, design patterns can be a reusable solution to developers to develop new application in a short period as low coupling between modules allows easy reuse of a module.

Our aim is to specify the design patterns which are relevant to mobile application development and choose the most optimal design patterns while taking into account the characteristics of target mobile applications and the quality attributes that we want to achieve. In this case, design patterns are identified and analyzed according to our application requirement specification. We are proposing an Extended MVC which combines Observer, Command, Composite, Mediator, and Strategy design patterns in redesigning our Student Planner Android Application.

Analysis of extended MVC

Pattern name: Extended MVC]

Intent: MVC pattern separate the business logic (model) and the presentation (user interface) logic (view). It consists of 3 components which are Model, View, and Controller. In extended MVC, Observer, Command, Composite, Mediator, and Strategy are incorporated.]

Motivation: MVC is a pattern used to isolate the business logic from the user interface. Model represents the information (the data) of the application and the business rules used to manipulate the data. View corresponds to elements of the user interface such as text, checkbox items, and so on while Controller manages details involving the communication between the model and view. The controller handles user actions such as key press, tap, etc and pipes them into the model or view as required. There could be more than one controller and changes in future (i.e. to add new controller) can be easily made without changing the view class because view is decoupled from the model. Observer pattern allows decoupling of the display and the application logic.

Applicability: Controller is used when there is a need to choose from different controller or when a decision is to be made to select an object from different objects implementing the same interface. When we have several subjects and observers, the relations becomes more complex due to many-to-many relationship which makes it difficult to manage. The relation between subjects and observers can contain some logic. Mediator pattern is introduced so that an observer only notify when all subjects change their states. We introduce Change Manager object which is responsible to maintain the many-to-many relations between the subjects and their observers, encapsulate the logic of notifying the observers, and receive the notifications from subjects and delegate them to the observers. The Change Manager is a type of observer. It receives notification of changes of the subject while at the same time it is considered as the subject because it notifies the observers.

Structure: Activity Life Cycle (Figure 1).

Participants: MVC Pattern consists of 3 components. However, for extended MVC pattern as shown in Figure 1, the components incorporate the other design patterns (i.e. Observer, Command, Composite, Mediator, and Strategy patterns).

Collaborations: View and concrete controller class interact to select the controller to be used to process the request. View registers itself with Student Planner Model and request state information. When Student Planner Model notifies view of the state changes, View update itself to reflect the new state.

Implementation: The Model object incorporates observer pattern to allow its listeners (i.e. view object) to get updates (i.e. changes in state) from it through Controller object. Model implements the Observer pattern to keep the interested objects updated when the state changes occur. Observer pattern keeps the model completely independent of

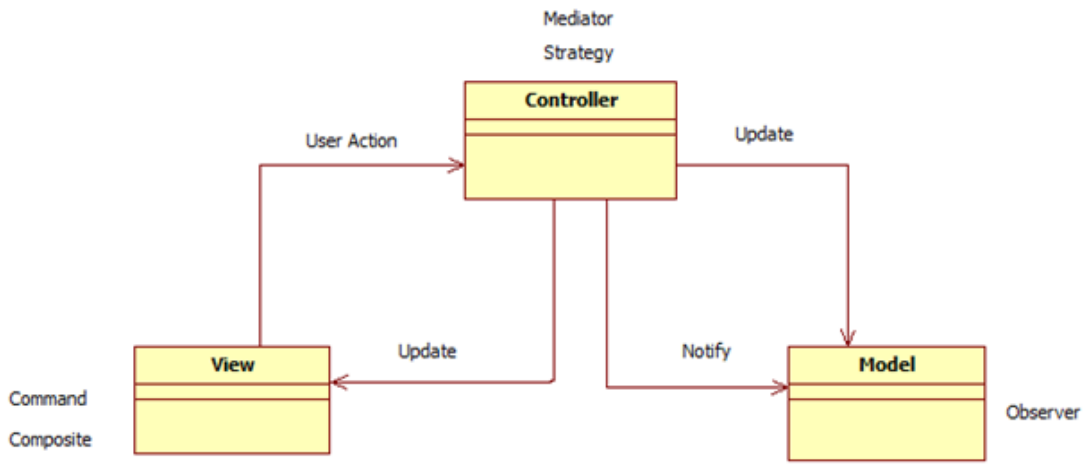

Figure 1: Extended MVC Structure.

the views and the controllers, and allows the use of different views with the same model, or multiple views at once. View is responsible for presenting the data and the state obtains from the model, to the user. View represents the output of the application. The View object incorporates Command and Composite pattern. The Composite pattern can be applied to the view giving it a hierarchical structure. The visual output of the application can be decomposed which form a hierarchy. The view objects are a composite of nested views that work together.

Controller is responsible to interact with the model to perform user's requests. Views invoke the appropriate controller, which acts on the model. It takes user input provided by the view, and converts to operations. The controller receives and translates the input, then requests on the model or view. Controller is responsible for calling methods on the model that change the state of the model. Controller object incorporates Mediator pattern and Strategy pattern. Mediator pattern mediates the flow of data between model and view objects in both directions. Changes in model state are communicated to view objects through the controller objects. Strategy pattern is implemented by Controller for one or more view objects to keep the view decoupled from the model. The view does not know how the operation is performed. The view object maintains visual representation, and delegates decisions about the application-specific of the interface behavior to the Controller. ApplicationController interface control the actions according to user's action such as entering data, saving data, canceling actions, etc. The concrete controller class implements Application Controller interface.

Implementation

The first step in development is to redesign Student Planner application into a new application which implements Extended MVC design Patterns. Our aim is to verify that the proposed Extended MVC increases efficiency, usability and reusability of mobile application. In addition, we also want to verify that proposed techniques could speed up the development process of mobile application (i.e. prevent issues that can cause major problems in development, reduce or remove duplication and improve code readability) and increase productivity of mobile application.

Overview of student planner application: Student Planner Application allows students to keep their university or college life organized. Students can track activities such as making appointments, completing assignments, attending quizzes/exams, and others. Students will be notified of any events (i.e. Tasks and schedule events). In addition, the application make use of Proximity Alert which allow users to be notified whenever they have tasks to be done near a predefined location. Student Planner also includes Speech to Text functionality whereby user can add new task easily without typing multiline subject. Conclusively, Student Planner Application consists of three components which are Activity, Service, and Broadcast Receiver.

Activity: An activity presents visual user interface from which a number of actions could be performed. It is independent of the others. Student Planner dashboard is marked as the first interface to be presented to the user when the application is launched. Intents are used to move from current activity to another. Figure 2 shows the activity life cycle of Student Planner Application.

A service: A service runs in the background for an indefinite period of time. Communication with the service is through an interface that the service exposes. Student Planner application implement Intent Service. The Intent Service is a base class for Service that handles requests on demand and terminates itself automatically. The Intent Service class uses the on Handle Intent method which is asynchronously called by the Android system. Student Planner makes use of Alarm Manager which provides access to the system alarm services. It allows scheduling of the application to be run at some point in the future. When alarm goes off, the Intent that had been registered is broadcasted by the system, and automatically starts the application if it is not already running. In Student Planner application, the scheduled alarm will start the service Schedule Reminder Service, Task Reminder Service, and Location Task Reminder Service. Student Planner also makes use of Location Service to find out the device's current location and request for periodic update of the device location information. The application registers an intent receiver for proximity alerts so that when the device is entering and existing from a given longitude, latitude and radius, the user will be notified. Location Manager, Location Provider and Location Lister classes in the Location API package are used to retrieve the location information of the user. Figure 3 shows the flow of Service.

A broadcast receiver: A broadcast receiver receives and reacts to broadcast announcements. Broadcast receivers start an activity in response to the information they receive, or as services. Student Planner makes use of broadcast receiver to fire notification to users by using notification manager. The flow diagram Figure 4 shows the flow of Broadcast Receiver used in Student Planner application to fire notification to user.

Evaluation

Evaluation criteria

By considering mobile devices characteristics such as sizes, display resolutions, operating systems, processor speed, memory size, and battery life, we define three verification criteria which are efficiency, usability and reusability. Efficiency is the capability of the application to exhibit required performance with regards to the amount of resources needed. Since the application make use of Location based service which requires GPS functionally to alert the user when they are near to certain location, it is important that the service is handled correctly and does not consume the battery life. Usability is the properties that exhibit the ease of the use of the application. As we aim to cater as many Android devices with different specifications as possible, we have to ensure that the application support those devices and behave as intended. User should be able to use the same features and perform the same operations regardless of devices they use. Reusability is the extent to which the design patterns could be reused for new application within similar domain. Since it is hard to test whether the application is reusable for other applications; our aim is to prove that the proposed design pattern is indeed reusable in mobile application development. Application which makes use of similar elements such as text view, edit text, scrolling, button, list view, radio button, etc should be able to utilize our proposed design patterns.

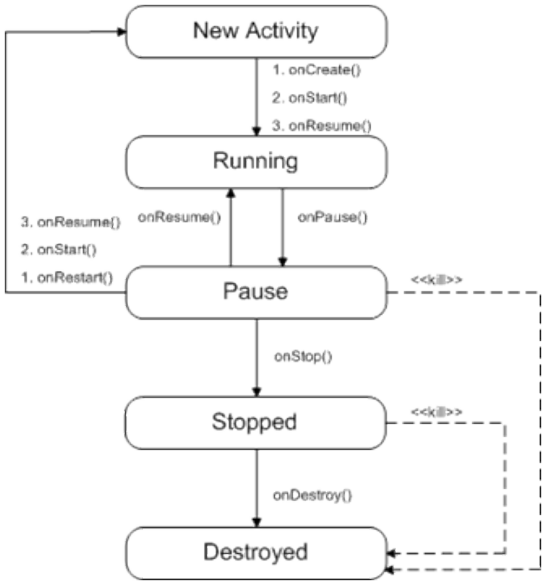

Figure 2: Activity Life Cycle.

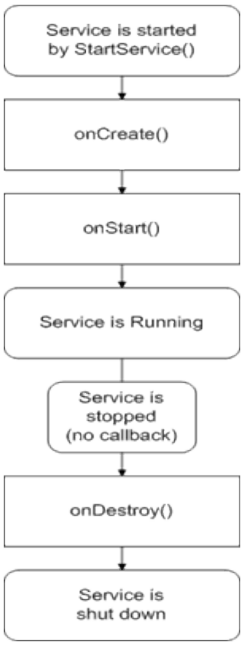

Figure 3: Service Flow Diagram.

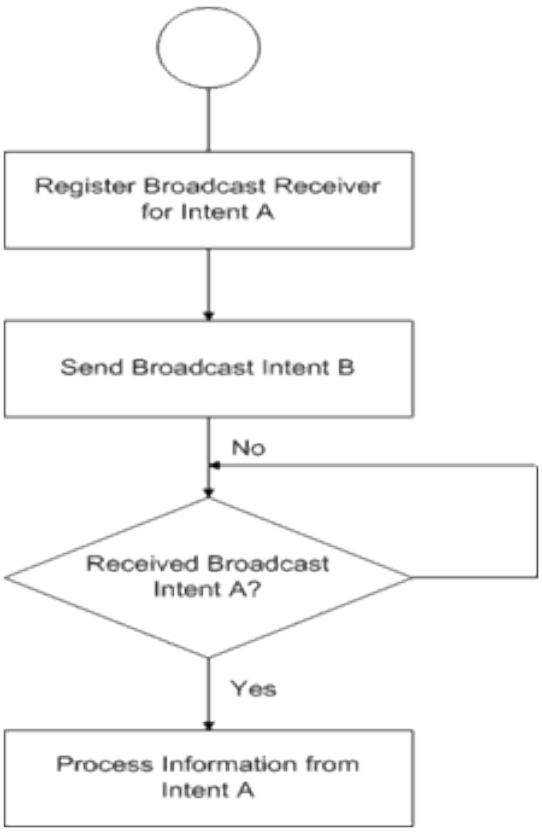

Figure 4: Broadcast Receiver Flow Diagram.

Evaluation methods

Evaluation and verification is the process of evaluating the mobile application in order to determine whether the implementation of the design patterns satisfies the requirements that we have specified in the earlier stage. In order to achieve the above criteria, we conducted Performance Analysis and Automated Testing. We conducted Performance Analysis Test which covers aspects such as CPU time

and object allocation. Several Performance Analysis tools were used to check performance problems such as garbage collection and memory leak when executing the application. Basically three types of analysis were performed which are heap usage examination, memory allocation tracking, and heap dump analysis. The heap usage examination was performed in Eclipse using Dalvik Debug Monitor Server (DDMS) for both version of application (i.e. normal application and MVC application). DDMS displays how much heap memory a process is using. This information is useful in tracking heap usage at a certain point of time during the execution of application. DDMS shows some basic statistics of the application heap memory usage which is updated after every Garbage Collector (GC). We also tracked the memory allocation of objects to track objects that are being allocated to memory and to identify which classes and threads are allocating the objects using DDMS. The purpose is to track the location of objects which are being allocated when certain actions are performed using the application in real time. This information is valuable to assess the memory usage that can affect the application performance. We created a heap dump for each application to track the problems (i.e. memory leak problem). The test was done using Memory Analyzer (MAT). MAT is a heap dump analyzer which immediately shows the biggest objects, categorizes objects by class loaders and adds application knowledge in a typical heap dump. In order to create a heap dump, we created HPROF file. The conversion is done using plug-in version of MAT (version 1.2) in Eclipse. Both applications were tested using Automated Testing Solution called Test droid Cloud. It is a cloud-based service, which allows developer to execute Android tests on various real devices from different manufacturers, with different Hardware platforms, OS version and screen resolutions hosted by the company. Testdroid provides screenshots, logs, exceptions, CPU, and memory consumption profiles of test execution. Every test run starts with rebooting the device, so that the tests are executed on clean devices, with no interfering processes running. Application and test package are then installed on the devices and tests are executed. Screen shots are taken during test execution to validate layout issues, or translation issues. We used Testdroid App Crawler which is an intelligent tool for checking applications device compatibility. We uploaded Student Planner application to Testroid server. After that, Testroid install the application on all recommended phones, crawl through all screens of the application and take a screenshot of each view. The application was analyzed and feedback on how the application works on various Android devices was gathered.

Test results

The results of testing are described in this section. The bar chart Figure 5 shows the number of passed and failed tests for each of the devices used in the testing.

The tests were performed on ten different devices with different specifications such as screen resolutions, Android version, internal storage, CPU, manufacturer, and RAM. All the tests passed for both version of Student Planner application which indicates that both versions are compatible with different Android devices used in testing.

Table 4 lists the result of test cases generated for each device with test case execution time.

As stated in Table 4, Student Planner using Extended MVC gives better execution time compared to normal Student Planner application. Table 5 shows quick summary of testing. From the table, we can see that both version of Student Planner Application are compatible with all devices used during testing. However, the average completion time of Student Planner using Extended MVC is faster than the normal Student Planner application.

Table 6 shows Compared to average results. For Student Planner Application, the slowest completion time is 91s while for Student Planner Extended MVC is 80.3s. In addition, the result shows 10% better full compatibility on Student Planner using Extended MVC compared to 0% on normal Student Planner application. Results also show that, the fastest installation time is 8s which is on Student Planner using Extended MVC.

Table 7 shows the timeline divided into main tasks of the test process for every single device used in testing for Student Planner

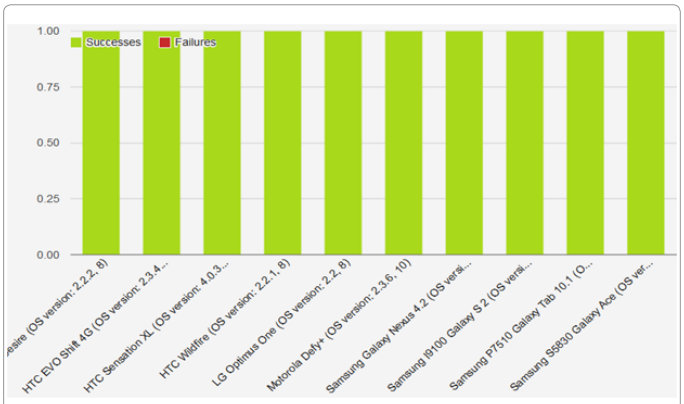

Figure 5: Test results by device.

Device	Student Planner	Student Planner Extended MVC
HTC Desire OS version: 2.2.2 8	313.67s	305.2s
HTC EVO Shift 4G OS version: 2.3.4 10	314.07s	308.16s
HTC Sensation XL OS version: 4.0.3 15	313.61s	305.12s
HTC Wildfire OS version: 2.2.1 8	339.26s	318.1s
LG Optimus One OS version: 2.2 8	340.47s	321.5s
Motorola Defy+ OS version: 2.3.6 10	314.31s	306.15s
Samsung Galaxy Nexus 4.2 OS version: 4.2.1 17	313.52s	305.6s
Samsung I9100 Galaxy S 2 OS version: 2.3.3 10	315.31s	310.12s
Samsung P7510 Galaxy Tab 10.1 OS version: 4.0.4 15	313.40s	305.55s
Samsung S5830 Galaxy Ace OS version: 2.3.3 10	356.63s	335.15s

Table 4: Test Case for Each Device with Test Case Execution Time.

	Student Planner	Student Planner Extended MVC
Device tested	10	10
Avg. completion time	323.425s	312.065
Installation compatibility	100%	100%
Full compatibility	100%	100%
Screenshots taken	602	602

Table 5: Quick Summary.

	Student Planner	Student Planner Extended MVC
Slower completion time	91s	80.3s
Worse installation compatibility	1%	0%
Better full compatibility	0%	10%
Faster installation time	10.7s	8s

Table 6: Compared to average results.

Device and Android Version	Cleaning device for testing		Rebooting device		Installing application		Launching application		Running Tests		Uninstalling application	
	SP	SPE	SP	SPE	SP	SPE	SP	SPE	SP	SPE	SP	SPE
HTC Desire OS version: 2.2.2 8	3s	2s	1m 33s	1m 3s	2s	2s	N/A	N/A	5m 15s	5m 10s	4s	3s
HTC EVO Shift 4G OS version: 2.3.4 10	3s	2s	1m 20s	1m 20s	3s	3s	N/A	N/A	5m 16s	5m 12s	2s	2s
HTC Sensation XL OS version: 4.0.3 15	6s	2s	1m 33s	1m 31s	3s	3s	N/A	N/A	5m 14s	5m 14s	2s	2s
HTC Wildfire OS version: 2.2.1 8	8s	5s	2m 26s	2m 26s	11s	10s	3s	1s	5m 42s	5m 42s	10s	8s
LG Optimus One OS version: 2.2 8	3s	3s	1m 23s	1m 23s	4s	4s	N/A	N/A	5m 15s	5m 13s	2s	1s
Motorola Defy+ OS version: 2.3.6 10	5s	4s	1m 28s	1m 23s	3s	3s	N/A	N/A	5m 42s	5m 15s	1s	1s
Samsung Galaxy Nexus 4.2 OS version: 4.2.1 17	4s	3s	68s	43s	2s	2s	N/A	N/A	5m 15s	5m 10s	1s	N/A
Samsung I9100 Galaxy S 2 OS version: 2.3.3 10	1s	1s	54s	49s	6s	5s	N/A	N/A	5m 16s	5m 13s	N/A	N/A
Samsung P7510 Galaxy Tab 10.1 OS version: 4.0.4 15	1s	1s	N/A	N/A	2s	1s	N/A	N/A	5m 16s	5m 12s	1s	N/A
Samsung S5830 Galaxy Ace OS version: 2.3.3 10	2s	2s	1m 20s	1m 18s	5s	4s	N/A	N/A	5m 57s	5m 42s	1s	1s

Table 7: Student Planner and Student Planner Extended MVC Performance Statistics.

Application and Student Planner using Extended MVC Application. As we can see from Table 7, from the beginning of the test process which is cleaning device until uninstalling the application, Student Planner using Extended MVC shows better results compared to normal Student Planner application. This shows that, the design patterns improved the efficiency of the application. The structure or design of application impacts the overall performance of the application because the design used in mobile application influenced how the application performs.

Conclusion

Design patterns have shown greater impact in classical software development. Since design patterns provide proven solutions, we believe that it can be implemented in mobile application to overcome the limitations and challenges in mobile application development. There are various considerations that need to be taken into account when developing mobile applications. This includes the type of application, supported devices, user interface design and device characteristics (i.e. battery life, memory size, and processor speed). With the advent of mobile computing, developers have to consider developing applications which can cater different specifications. However, developing mobile application can be a tedious process as each application have to go through the development cycles in order to ensure the application conform to standard quality attributes.

We have identified and analyzed design patterns for mobile application development, implement the design patterns in mobile application, evaluate and verify the effectiveness. We have proposed extended MVC design patterns and implement the design pattern in Student Planner application in Android. We catered different version of Android (i.e. 2.2 to 4.1) by developing the application in backward compatibility. In addition, we took into account different characteristics of mobile devices such as screen sizes, and orientations. Since most devices come in multiple screen sizes and display resolutions, Extended MVC can be used to support dynamic properties and improve flexibility of application. Thus, enhance the look and feel and improve the usability of the application.

While implementing Extended MVC design patterns, we verify the application in terms of efficiency, usability, and reusability. Results show that the efficiency of mobile application is greatly improved. Since mobile devices has limited resources and processing capability, the implementation of design pattern has improved the efficiency of mobile application compared to normal application. The Extended MVC reduces code duplication and improves the application design in terms of code readability. In addition, testing shows that the application is compatible with various devices and aspects such as such screen size, resolution (DPI), and elements are handled properly. Extended MVC is used to separate the presentation and the application code (i.e. Event handling, initialization and data model, etc). This allowed us to develop application with loose coupling and separation of concern. Each module only makes use of little or no knowledge of other modules. Therefore, adding and removing features can be made easily without affecting other modules. Low coupling between modules also allows easy reuse of a module. The resulting Extended MVC design pattern can be used as a basis to develop similar application which requires the use of internal storage and similar elements such as list view, edit text, text view, buttons, scrolling, action bars, etc.

Future Work

Further studies may include identifying other design patterns which can be integrated with the proposed design patterns or finding other design patterns which can yield better result in designing mobile application. In addition, future research works include investigating how to implement the design pattern for application which requires excessive use of network connections or implementing the design pattern on client and server side.

References

1. B'Far R (2005) Mobile Computing Principles: Designing and Developing Mobile Applications with UML And XML. Cambridge University Press, UK.
2. Saylor M (2012) The Mobile Wave: How Mobile Intelligence Will Change Everything. Vanguard press, USA.
3. David H (2012) Microsoft Application Architecture Guide.
4. Forman GH, Zahorjan J (1994) The Challenges of Mobile Computing. Computer 27: 38–47.
5. Hayes IS (2002) Just Enough Wireless Computing. Prentice Hall Professional, USA.

6. Gamma E, Helm R, Johnson R, Vlissides J (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, USA.

7. Apple Inc (2012) Cocoa Fundamentals Guide: Cocoa Design Patterns.

8. La HJ, Kim SD (2010) Balanced MVC Architecture for Developing Service-Based Mobile Applications. 2010 IEEE 7th International Conference on e-Business Engineering (ICEBE) 292-299.

9. Biel B, Gruhn V (2010) Usability-improving mobile application development patterns. Proceedings of the 15th European Conference on Pattern Languages of Programs. EuroPLoP 11: 1-5.