

Assessment 1 Final

by Oliver Houlden

FILE	9556_OLIVER_HOULDEN_ASSESSMENT_1_FINAL_164031_1315508859.D OCX		
TIME SUBMITTED	21-DEC-2018 09:07 PM (UTC+0000)	WORD COUNT	3320
SUBMISSION ID	97923888	CHARACTER COUNT	18888



DESIGN PATTERNS AND WHAT THEY ARE

Application Development



NOVEMBER 14, 2018

COLEG SIR GAR
By Olly

Contents

1.0: Introduction	2
2.0: Part A.....	3
2.1: What is a design pattern?	3
2.1.1: UX Design Pattern	3
2.1.2: Architectural Design Pattern.....	4
2.1.2.1: Architectural Design Patterns	4
2.2: Façade	5
2.2.1: Façade Concept.....	5
2.3: Observer.....	6
2.3.1: Observer Concept	6
2.4: Comparisons	7
2.5: MVVM	7
3.0.1: Dice Application	8
3.0.2: Launchpad Emulator	8
3.0.3: CSG Adventure Game	8
4.0: Testing Applications	9
4.0.1: Dice Application	9
4.0.2: Launchpad Emulator	10
4.0.3: CSG Adventure Game	11
4.0.4: Conclusion.....	12
5.0: References	13

1.0: Introduction

In this report I will be investigating two different software patterns. (put what they are here) are both design patterns. I will show an example on how these patterns are used. Finally comparing these two patterns against one another. To investigate further I will be talking more in the lines of different types of patters, UX, architectural and software design patterns.

Once these have been spoken about I will discuss what MVVM is and why it has become more important.

2.0: Part A

In this section I will be discussing Software, UX and Architectural Design Patterns. I will then decide on two types of these patterns and compare them with one another and explain which one would be better for which purpose. Finally, to explain what MVVM is and which overall pattern it would fit under, which it compares against and why.

2.1: What is a design pattern?

What is a design pattern? To know this, you must look at programming more than just a language and more as a solution to the problem at hand. Christopher Alexander says, *"Each pattern describes a problem which occurs over and over again."* [1]. Here Christopher is talking about building pattern, however building patterns or design patterns essentially mean the same thing. They both offer a solution to a problem. *"Design patterns were introduced in Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides's seminal work Design Patterns: Elements of Reusable Object- Oriented Software (Addison-Wesley)"* [2] A design pattern by itself is known for simply solving a recurring issue in a program, such as the façade pattern and the Factory Pattern. The façade pattern is a pattern that uses subsystems, in C# it would be known as namespaces and classes, to separate different sections of the program.

2.1.1: UX Design Pattern

UX means User Experience. In short, the UX pattern is to make the user interface usable by the user and programmer. In this case study it says, *"Formative testing is used to see how well a UX design addresses a product's anticipated use cases."* [3] This is saying that a good UX design comes by testing thoroughly. Without this thorough testing there would be no way to know that the users experience while using the app is a good one. There are four core aspects of a good UX design. These are; Usefulness, Usability, Appealing and Engaging. As said by Anthony Davies [4]. Usefulness is important as if the application / website is useless then the overall experience would be lacking. Sites such as Google+ have longed to live up to how popular Facebook is, but there was no need to use both. The usability, attractiveness and enjoyment of the application / website is also important as if something that looks cluttered, is a pain to use and unenjoyable to interact with, the user will likely pass on using it. To find out if the UX is up to standard vigorous testing must be attained.

2.1.2: Architectural Design Pattern

In this section I will be discussing Software architectural patterns. There are a few of my findings explaining what they include as well as what types of patterns these encapsulate.

2.1.2.1: Architectural Design Patterns

An architectural pattern is like the actual architecture of a building where you would have the downstairs as one whole section, the upstairs as another but all under one entire roof. As cited by the Software Engineering Association of Taiwan *“An architectural pattern is rather a concept that captures essential elements of software architecture.”* [5] In simple terms an architectural pattern would deal with a large-scale software system, such as one in an organisation. An example of a design pattern that would use the architectural design would be; MVC, MVVM and MVP.

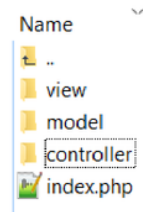


Figure 1

In figure 1 you can see the layout of MVC. As I said before about everything being under the one roof, here shows the index is that roof. Then there are three sections to this pattern to separate the rest of the program. *“Architectural patterns are larger-grained than design patterns, address- the structure of major subsystems of a system.”* [6] This is saying that an architectural design would be a design pattern that is made for larger-scale programs / systems.

2.2: Façade

The Façade Pattern is known as a structural pattern and similarly to the Adapter Pattern, it can be used to hide the inner workings of an application. *"The role of the Façade pattern is to provide different high-level views of subsystems whose details are hidden from users."* [2] What Bishop is saying here is that the Façade patterns main objective is to have a complex task happen in the background when a user would just press a button. Bishop goes on to talk about an illustration where the button on "www.Amazon.co.uk" to "buy now with one click". The way this works is to use the users Amazon account information, such as name, address and payment method, to quickly order the item without the user doing anything else other than clicking that button.

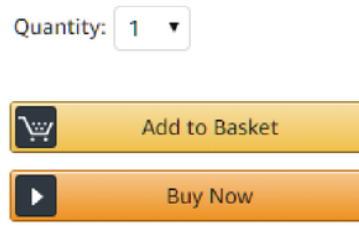


Figure 2

The aim of the Façade pattern is to simplify a complex task in a program. As said by Judith *"The aim of this pattern is to provide a simplified interface to a set of complex systems"*. [2] As shown by the amazon screenshot in Figure 2 this is how a complex system is made simple to the user. The façade pattern can be implemented by using the C# programming language. As said by Bishop *"The Façade pattern is simple to implement. It uses the C# concept of namespaces."* This is saying that the façade pattern simply uses the C# class system. A namespace is a feature in C# programming where it separates the variable names and classes from the rest of the program. For example, if you wanted to use a method called "TestMessage" that displayed the text "This is a test" as a message within the first namespace, you could also create another method in a different namespace called "TestMessage" and have it display something entirely different. The façade pattern uses this same concept. In simple terms the Façade pattern is to display a simplified interface to use the application / website / Operating System.

2.2.1: Façade Concept

To show the concept of the Façade pattern design I will show a simple diagram of interconnected subclasses behind a simplified interface. To explain the diagram below. The client will see the interface and will interact with only this. The Façade will then look through the Subsystem classes to show the client the result.

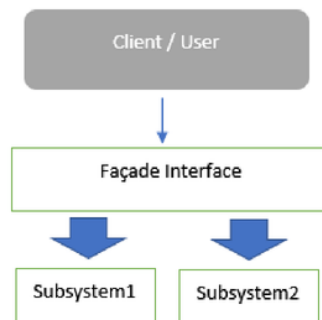


Figure 3

2.3: Observer

The observer pattern is a behavioural pattern. A behavioural pattern is a way of splitting up the program by separating it into classes. This has its downsides though as it makes the program more complex and difficult to manage. *"Behavioural patterns are concerned with algorithms and communication between them."* [2] Which is saying that not only is the program split up, it is reliant on algorithms. The behavioural patterns manage the communication between one class to the next. It manages it to a way until it is best optimised.

To explain how the observer pattern works Judith gave an example of a blog. The blogger would notify their subscribers on an update to the blog instead of the subscribers checking continuously if there was an update or not. This is a visualisation on how the Observer Pattern works. *"The Observer pattern defines a relationship between objects so that when one changes its state, all the others are notified accordingly."* [2] She is saying here that this pattern will notify one part of the program to the other depending on if the related classes need the data to be changed or updated. An example of how the observer pattern works is by using MVC as an example, as the View represents the Observer and the Model represents the observable object. As said by Kanjilan *"The MVC architectural pattern is used to build applications that are loosely coupled, easier to test and maintain."* [7]

"The Observer pattern is widely used in blog sites and various software systems" [2] These other software systems include Facebook, Microsoft Updates and Flickr to name a few. The observer pattern can also be used within the MVC Framework / architectural pattern under the "Model" section of MVC. One final example of the use of the observer pattern is the use of using live changing data, such as in spreadsheets where the graph may change depending on if the original data has been changed / updated. *"Different graphs reflecting data in a spreadsheet change when the data changes."* [2]. Kanjilal backs this statement up by saying that the observer pattern "is used when you would want to notify change to a number of classes" [7]

2.3.1: Observer Concept

I will now show a diagram explaining what the observer pattern does and I will explain the diagram.

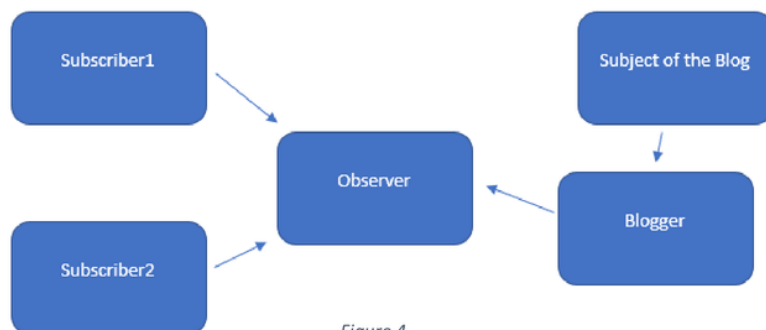


Figure 4

Here it begins at the subject of the blog, the blogger will then write about the subject. There are then subscribers to the blog itself. The observer will then send a notification to the subscribers if the blogger makes an update to the blog.

2.4: Comparisons

To compare the Façade and Observer Patterns I must first mention what types of patterns these are. The Façade pattern is a type of structural pattern, whereas the Observer Pattern is a behavioural pattern. The structural pattern is in play to keep the user from doing something complex, such as the example with amazons “one click buy now” feature, where the user would simply have to just click a button or press a button on the keyboard and they will see the result. However, there is complex code behind this simplistic view that the user sees. The observer pattern does not do any of the simple design. Where it would put its focus on allowing all the same subsystems, which Façade also uses, to communicate with one another and the observer will update the other subsystems depending on what data changes. Such with the example of a blog where the observer would update the blog and then notify that there are changes to its subscribers.

There are some similarities between these patterns, other than the languages. They both take good use of the subsystem class to separate code from what the user sees. Both patterns disallow the end user to see how the application works and to change what they want to. By this I mean that the observer pattern would not allow the subscriber to change what is written on the blog and the buyer on amazon using the Façade pattern would not be able to change how the system charges their account. This is, again, because of the use of subsystem classes. This simply hides what does not need to be seen by the general user and only shows what is needed.

2.5: MVVM

MVVM is an important and popular pattern when it comes to developing mobile applications, especially for larger scale applications. *“MVVM presentation is the same model (PM) presented by Fowler, is that both models have that feature an abstraction of a View, which contains state and behaviour of View.”* [8] Here it talks about Fowler, a BSC achieved software developer working at “ThoughtWorks”, where he presented an article talking about the presentation model, PM. It is saying here that both models use a view. The view is generally what the user would see and interact with. In this journal it goes on to say how similar PM is to MVVM. It is often described as the same model, but with a different name, as said here *“MVVM is identical to Fowler's Presentation Model, in that both patterns feature an abstraction of a View”* [9].

The purpose of MVVM was to simplify the creation of a UI or application. *“Gossman introduced MVVM as a standardized way to leverage core features of WPF to simplify the creation of user interfaces.”* [9] He goes on to say that MVVM is a more specialised model over PM purely on this goal. So, how does it make the creation simpler to develop? Well how you should imagine MVVM to work is by first having the View, ViewModel and finally the Model.

MVVM works by synchronising the View and ViewModel, where the Viewmodel updates the View. This is a benefit as there is less code to write. As said by Josh *“By binding properties of a view to a ViewModel, you get loose coupling between the two and entirely remove the need for writing code in a ViewModel that directly updates a view.”* [9]. He goes on to say that using this loose coupling method it supports data binding, which supports input validation to check for any errors. This error checking between the View and ViewModel would be essential to programmers as less time would be needed to code data validation.

Like with all patterns though, there are a few caveats that come with this pattern. One major issue with MVVM is when a developer decides to use this model in a small-scale application. The main benefit of this model is to use it within a larger scale application / system.

3.0: Part B

I will be creating three individual applications using Android Studio.

3.0.1: Dice Application

The first application will be a dice rolls game. This will allow the user to make two dice roll on screen. The number would then be totalled up and displayed into a scoring system. The aim would be to get a higher score than the computer with a best out of three approach.

3.0.2: Launchpad Emulator

This application will allow the user to play music samples by pressing buttons on screen.

3.0.3: CSG Adventure Game

This final application will be a multiple-choice adventure RPG game set in and around the college campus. This game will have multiple endings via the user choosing predetermined options throughout the story.

4.0: Testing Applications

I will now be testing all these applications to determine if they have reached the desired outcome. To test these applications, I will be using a method called “Black-box” This tests the functionality of the applications by someone without any prior knowledge of the application. The test results will come from a user instead of myself.

The second method of testing the application I will be using another method called “White Box”. This allows me, the developer / programmer, to test the code I will be using a debugger to the code and run through each line. I will then document any errors that occur.

4.0.1: Dice Application

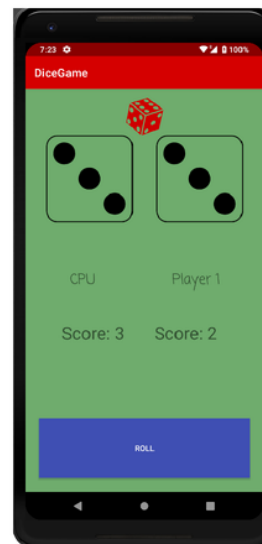
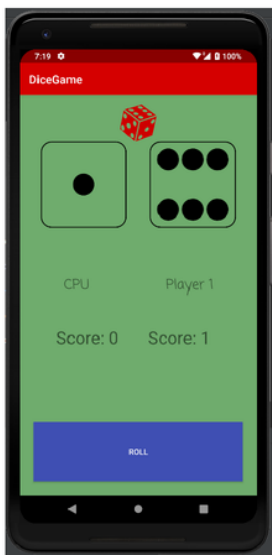
To test this application, I got Dan Samuel to black box test the application. This was done because he does not know the code I used, and I wanted someone to test the app who didn’t know the code. From testing the dice application Dan gave me some pointers so I could improve the app.

One improvement I could make is to add a limit to how many throws / turns you take. This will quickly determine who wins and who loses. As of now there is just an unlimited number of throws someone can do.

Another improvement was to include a time delay between the player and when the CPU takes its turn. The CPU as of now throws the same time the player does.

Over the improvements Dan said that the application works as it should. There is a counter to show the score each player has, and you can determine who wins by looking at this. He especially like the small animation of the dice roll once you use the button.

Here shows a screenshot of the application running on a Pixel 2 XL. It is showing that the player has 1 point because they got a 6 and the CPU had a 1. On the right it is showing that if they both get the same dice. They will both receive a point.



This concludes testing for this application.

4.0.2: Launchpad Emulator

Once again, I asked Dan to test this application. This app was to allow a user to play music by pressing buttons on screen. Dan gave me improvements to do on this too.

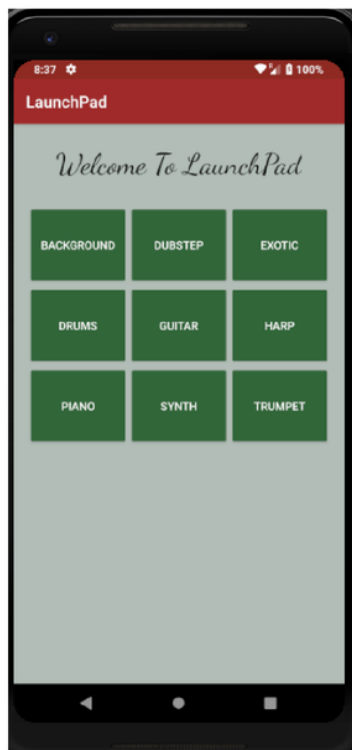
One improvement would be to add a few buttons that can allow the user to add their own tracks to the app. This would make the launchpad emulator app more like an actual launchpad.

Another would be to save and record the sounds that you make on the launchpad. As of now it just plays predetermined sounds.

After these improvements the application would have evolved into a different application.

There were a few bugs in this where sometimes when you start the application the sounds of the bottom three buttons would simply not play. This is something to look more into later. One last thing was the volume of the guitar, the volume was put down after this testing.

Overall the application works as intended. Below shows a screenshot of the layout of the application.



This concludes testing for this application.

4.0.3: CSG Adventure Game

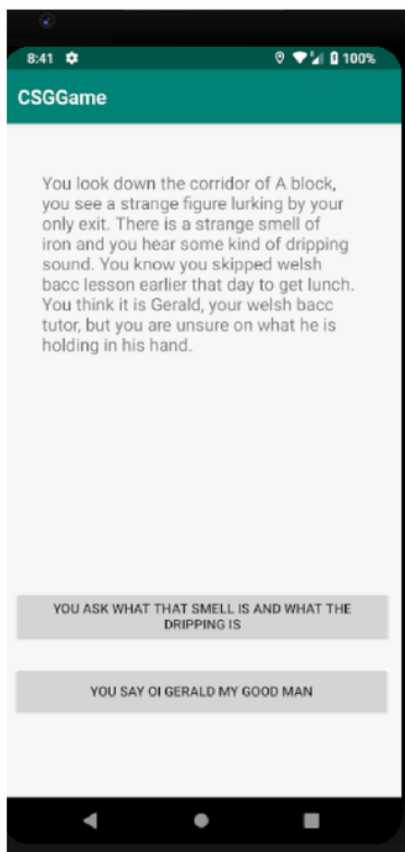
Finally, the text game. This game was made entirely using buttons and string values. Through Dans recommendations and improvements I am certain this app could be amended in the future, such as including a longer story, sound effects and possibly changing pictures in the background.

One of Dans improvements is exactly this, to add sound effects and pictures to the application. This will better immerse the player into the story.

One final improvement was to add a menu system to the app for the user to click “play” instead of going straight into the game.

After this the game works as intended with a few minor caveats with the story sometimes skipping right to the end without cause, it would not happen often either.

Below shows a screenshot of the story.



This concludes the testing for this application.

4.0.4: Conclusion

Throughout the application development stage, I was testing the apps to see if they worked throughout their development. This was how I was able to get all the applications working when it came to asking Dan to help test the applications. All apps work as I intended them too and am overall pleased with how they turned out.

5.0: References

- [1] R. H. R. J. J. V. Erich Gamma, Design Patterns, London: Pearson Education, 1994.
- [2] J. Bishop, C# 3.0 Design Patterns, O'Reilly, 2007.
- [3] *UX Design and Agile: A Natural Fit?*, 2011.
- [4] A. Davies, "What is a UX Designer," Freelancer UX Webmaster, 4 November 2015. [Online]. Available: <http://www.userexperiencedesigner.co.uk/new-what-is-ux-designer-ia.htm>. [Accessed 05 December 2018].
- [5] Software Engineering Association of Taiwan, "the Third Taiwan Conference on Software Engineering," *An Experience of Applying Pattern-based Software Framework to Improve the Quality of Software Development*, vol. 2, no. 6, pp. 185 - 194, 2008.
- [6] H. Gomaa, Software Modeling and Design, California: Cambridge University Press - M.U.A, 2010.
- [7] J. Kanjilal, "Implementing the Observer design pattern in C#," vol. 1, no. 1, p. 1, 30 Jan 2017.
- [8] M. I. M. Erik SØRENSEN, "Model-View-ViewModel (MVVM) Design Pattern using Windows," *Megabyte Journal*, pp. 1-5, 2010.
- [9] Josh Smith, Microsoft, "Patterns - WPF Apps With The Model-View-ViewModel Design Pattern," *MSDN Magazine*, 2009.
- [10] H. Gomaa, Software Modeling and Design, Virginia: Cambridge University Press, 2011.

Links used throughout creation of the app.

<https://colors.co/484a47-70ae6e-5c6d70-335145-a37774>

<https://freesound.org/>

<http://www.orange-freesounds.com/>

<https://www.dotnettricks.com/learn/designpatterns/gang-of-four-gof-design-patterns-in-net>

http://megabyte.utm.ro/articole/2010/info/sem1/InfoStraini_Pdf/1.pdf