VLC Android medialibrary

A deconstruction of VLC’s Medialibrary

Manukau Institute of Technology

Software Engineering Group 3 Assignment

Contents

[Introduction 2](#_Toc523055904)

[Use Case Diagram 4](#_Toc523055905)

[Class Diagram 5](#_Toc523055906)

[Architectural Components and Connectors 7](#_Toc523055907)

[Architectural Style/Pattern 7](#_Toc523055908)

[Why MVP? 8](#_Toc523055909)

[Architectural Configuration 9](#_Toc523055910)

[Software Quality Attributes 12](#_Toc523055911)

[Testability 12](#_Toc523055912)

[Portability 13](#_Toc523055913)

[Recoverability 13](#_Toc523055914)

[Software Metrics Analysis 14](#_Toc523055915)

[LOC: 14](#_Toc523055916)

[NOC: 15](#_Toc523055917)

[Number of Children (NOC) 15](#_Toc523055918)

[Overview 15](#_Toc523055919)

[Computation 15](#_Toc523055920)

[Nominal range 15](#_Toc523055921)

[Analysis 16](#_Toc523055922)

[3. WMC (Cyclomatic Complicity) 16](#_Toc523055923)

[WMC - Weighted Method Count A (the base metrics) 16](#_Toc523055924)

[Thresholds 17](#_Toc523055925)

[Coupling Between Objects (CBO) 18](#_Toc523055926)

[References 19](#_Toc523055927)

[Team Roles 20](#_Toc523055928)

# Introduction

VLC or VideoLAN Client is an open source and free cross platform media player that can play a wide variety of multimedia files such as video and audio files. VLC was an academic project for streaming video by a non-profit organization named VideoLAN in the mid-1990s in Paris, France. VLC runs on all popular operating systems and different mobile devices. (VLC, n.d.)

In this project we will be focusing on VLC for android which is a port of VLC Media Player for android devices. The VLC app can play almost all types of video formats such as mp4, avi, mkv, mov and audio formats such as ogg, flac, wv and aac files and also Network Streams such as adaptive streaming. VLC also supports disk share through Network Drives and Network Shares similar to the original desktop version on PC. VLC also support Closed Captions, Teletext and Subtitles in their software. (VLC, n.d.)

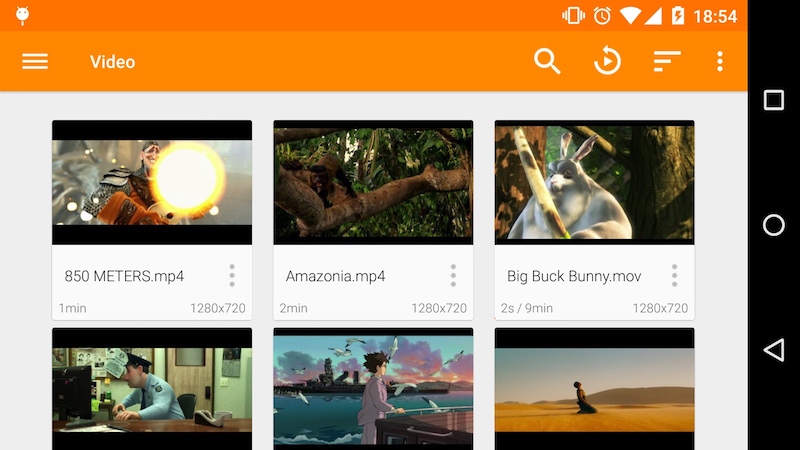


Figure 1. VLC Video Library (VLC, n.d.)

The VLC app has a media Library for both audio and video files, which allows browsing to the phone’s folders directly. It has a complete audio media library that supports headset controls and cover art for each of the songs in the library.

VLC has a wide variety of features which is why we are only focusing on the media library part of the app for this project.

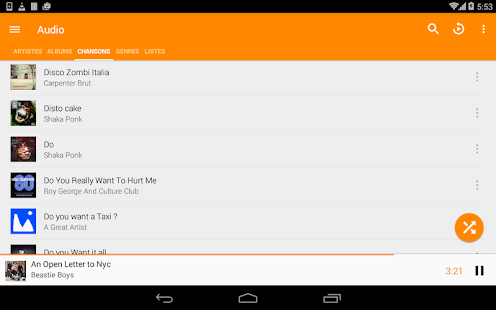
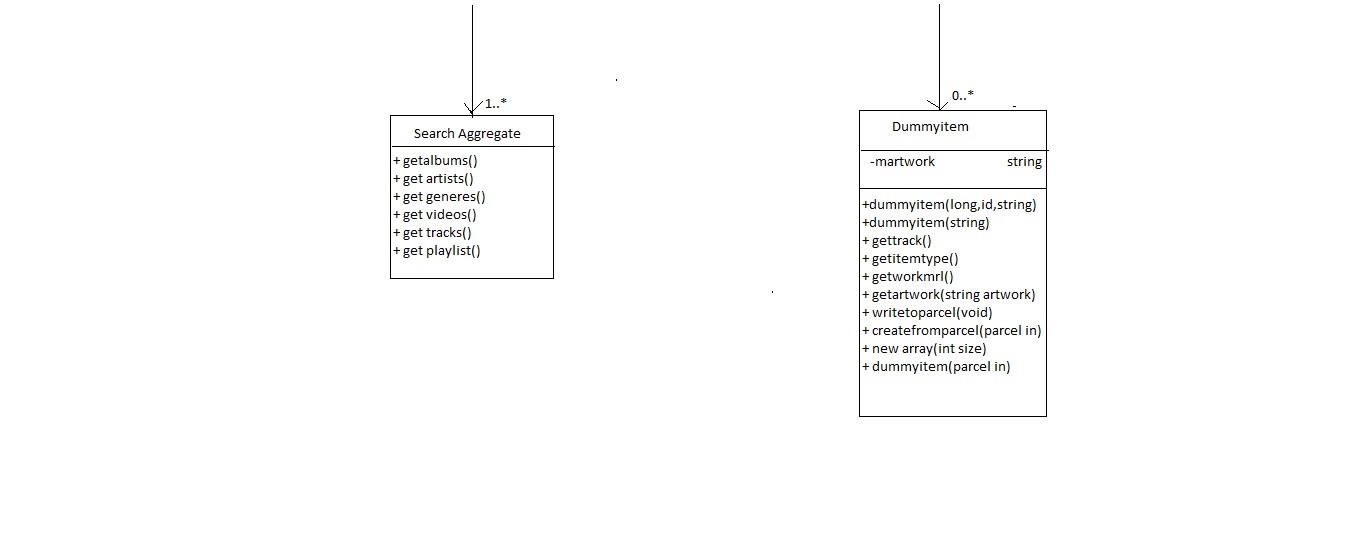
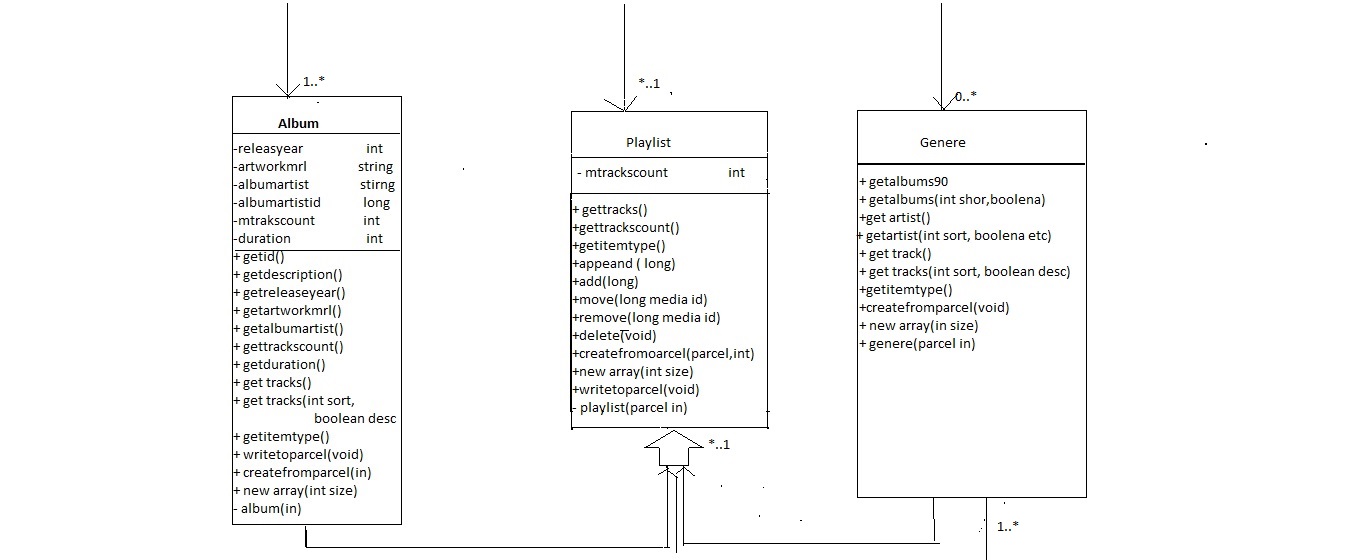
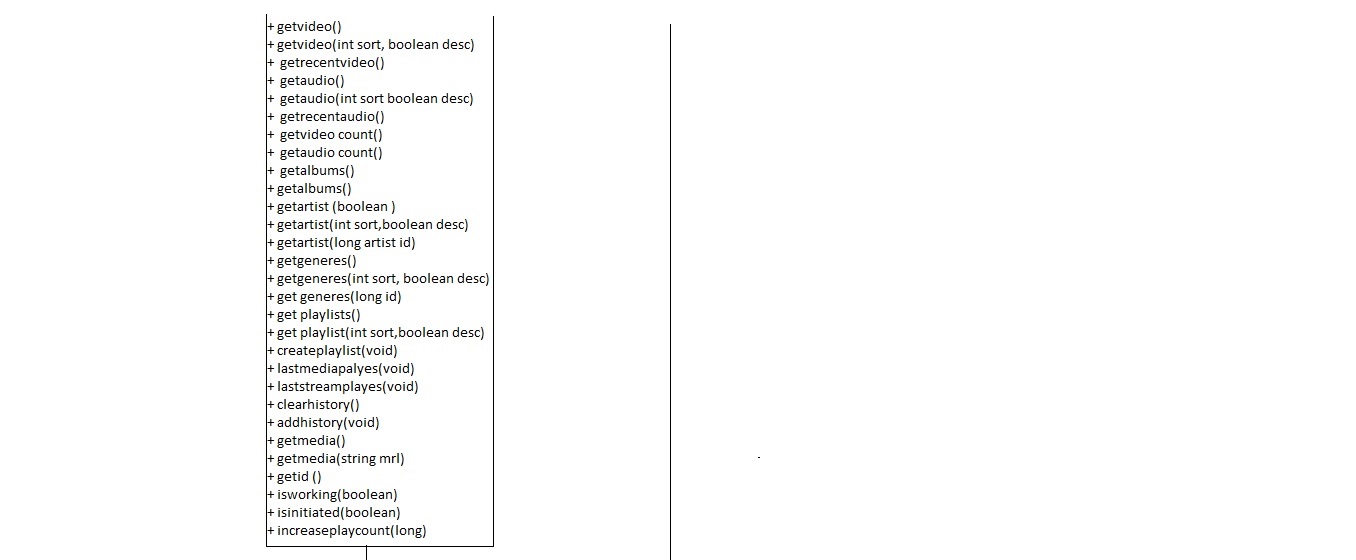
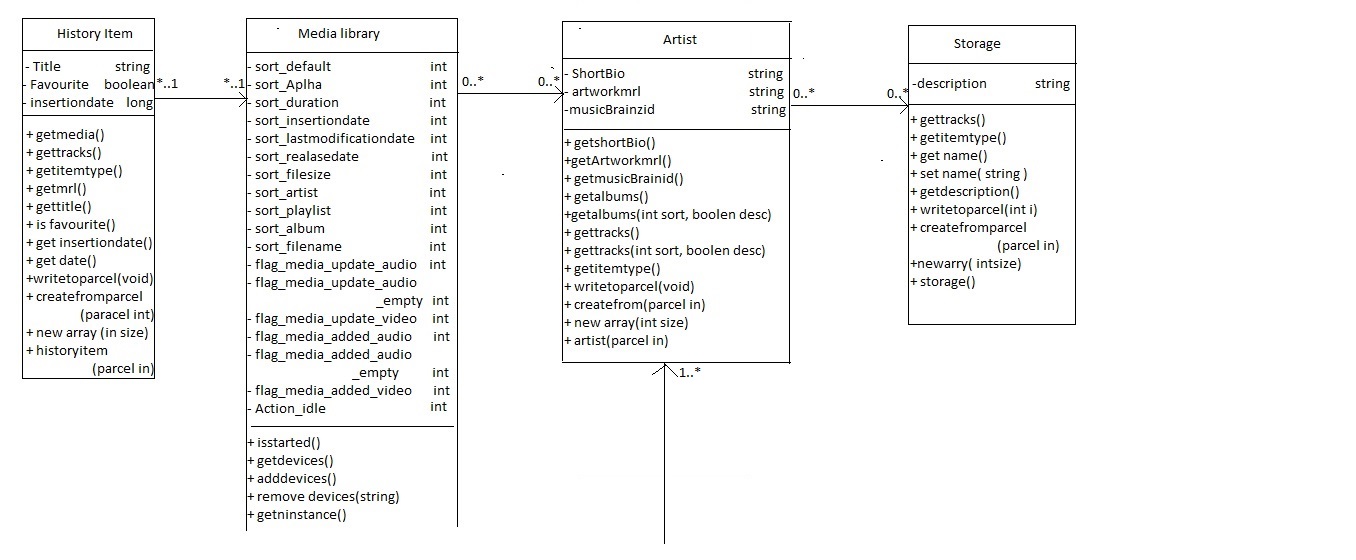


Figure 2. VLC Audio Library (htt)

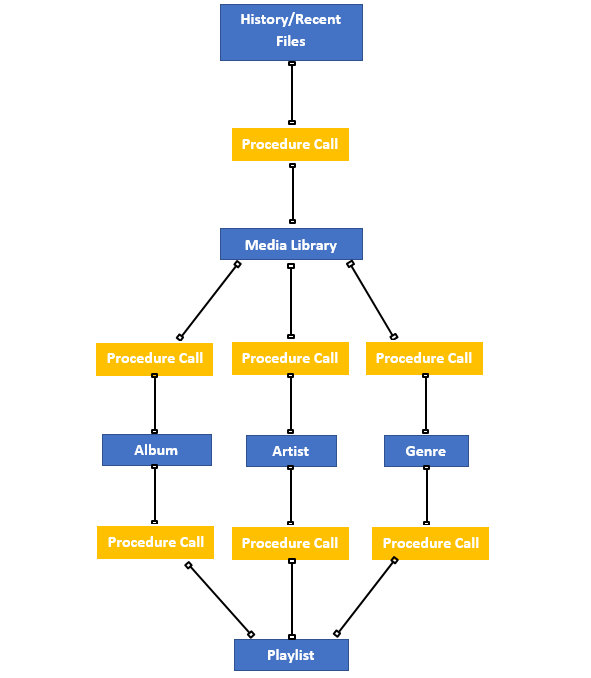
## C:\Users\Louis\Desktop\VLC use case diagram.pngUse Case Diagram

The media library has 4 main function or use in the VLC system. We also have 2 actors in the diagram, they are the User and the System. We have included the system as an actor in the diagram because the media library is just a part of a larger system that the VLC software controls. The 4 main functions that the media Library does are to open, create, update and delete a library. The open library function has 2 extends which are to randomize the songs and videos in the library and to play the video or audio files in the library and whilst playing the files it still has a few more extend functions which are to stop, pause, fast forward/rewind and to replay the song or video. The Create Library has 2 includes which are to add a name of the created library and to add songs and videos in the created library. The update function has 3 extends which are to rename the library, add new song or videos to the library and to delete old songs or videos in the library. Lastly the delete library function has an include that deletes all the files in the library.

# Class Diagram



# Architectural Components and Connectors



Architectural Style/Pattern

Since our group has chosen an Android app, this actually makes it easier to find the architectural pattern since most Android apps mainly use a couple of patterns and those patterns are similar to MVC but changed a little bit to fit Android. The problem with MVC is that it is tightly tied with the Android API’s, so this makes it hard to unit test. Another problem of MVC is that the controller is tightly coupled with the view and this basically makes it an extension of the view. If we change the view we must change the controller too. Because of this and the research I have done in to the VLC code and online examples I have concluded that the architectural pattern that the Android version of VLC uses is MVP (Model-View-Presenter). (Maxwell, 2017)

Why MVP?

With MVC there are too many issues that just don’t work with Android. MVP almost gets rid of most of these problems without there being a massive difference from MVC. MVP has clear separation of responsibilities between components. This allows for an easier understanding and maintenance of the code. As stated before there is not a massive difference, so MVP still has all the good stuff that MVC offers such as modularity, which allows you to change a component without changing other components and because the separation is well defined with components, testing become a lot easier because you can test a component in isolation. (Advantage of MVP in Android, 2016)

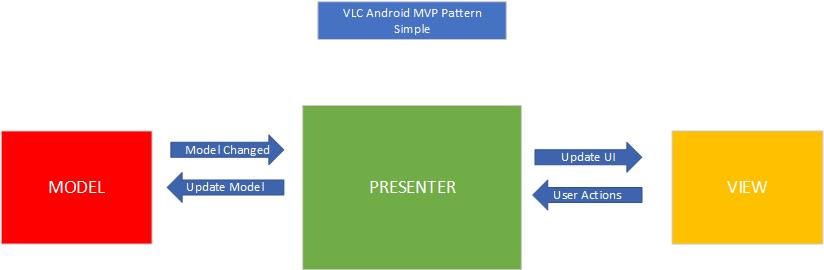
The M in MVP is still the same as MVC. It is still the model. It is the data logic layer, so nothing has changed. (Maxwell, 2017)

The V in MVP still stands for view. The view still has same description of the view in MVC but instead the Activity which is part of Android, is now considered part of the view. This gets rid of coupling to any specific view and allows tests with a mock implementation of the view. (Maxwell, 2017)

The P in MVP stands for presenter. The presenter is basically the controller except it’s not all tied to the view, it is just an interface. This helps with the modularity and testability that MVC has a problem with. The presenter just displays “what” it needs to not “how” to display it. (Maxwell, 2017)

From all this information I have gathered and by studying the code and other examples of MVP, I saw a big similarity where the example codes and the VLC code used an “interface” section with code looking very similar to each other. This was a major reason for me to decide this was an MVP architecture pattern, but I can’t really say why because there are problems with MVP and it is not perfect either, but I guess it just comes down to the preference of the team making this.

Architectural Configuration



The chosen model was MVP (Model View Presenter), which is not specific to android but does work well and is the chosen more common method for app development on android.

The Presenter is the one that connects all the other components together. The view interacts with the presenter by calling its functions. The presenter is the main chunk of this software as the view is provided by Androids API UI.

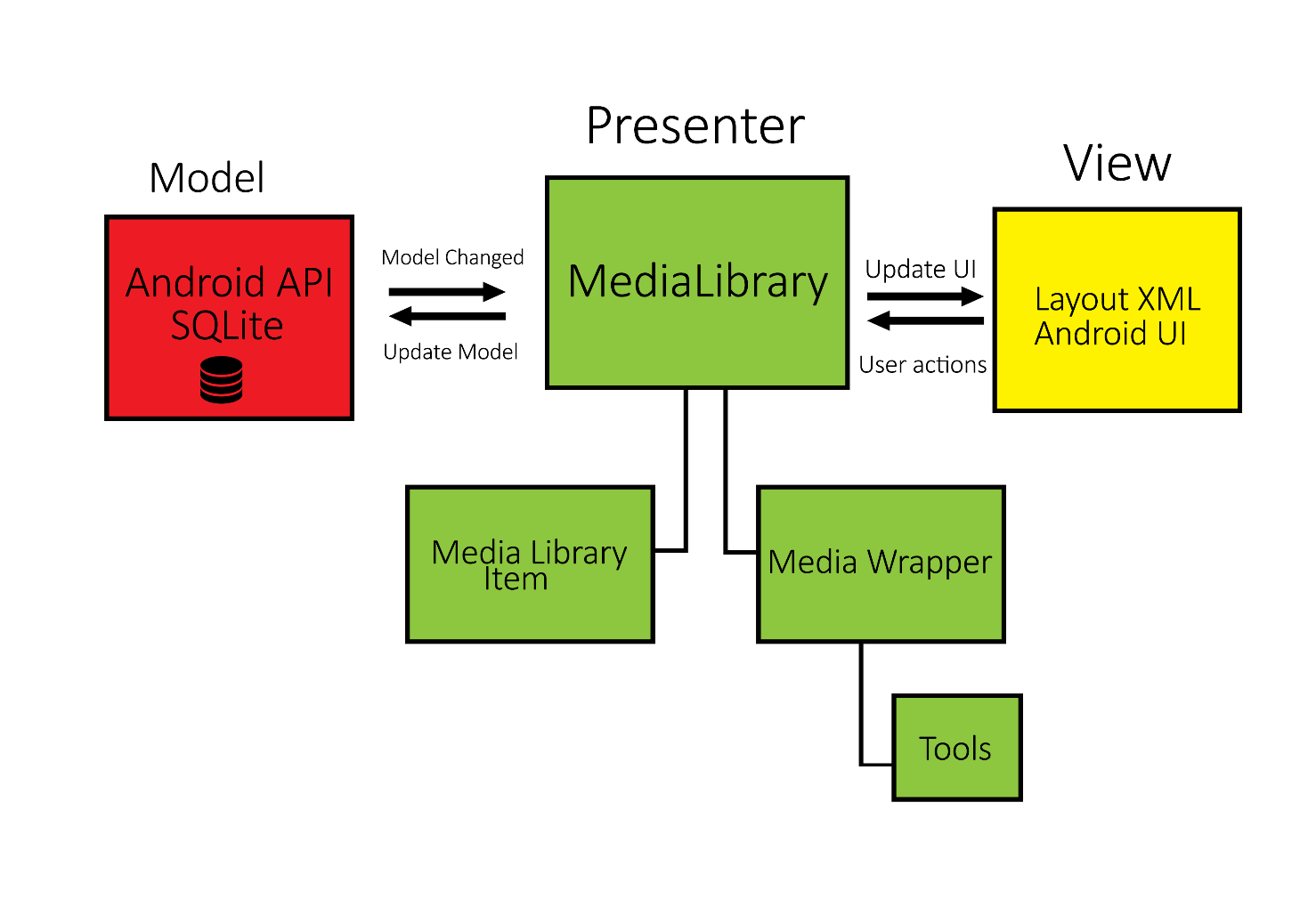
The model consists of two parts. There is a local small database and Androids SQLite service.

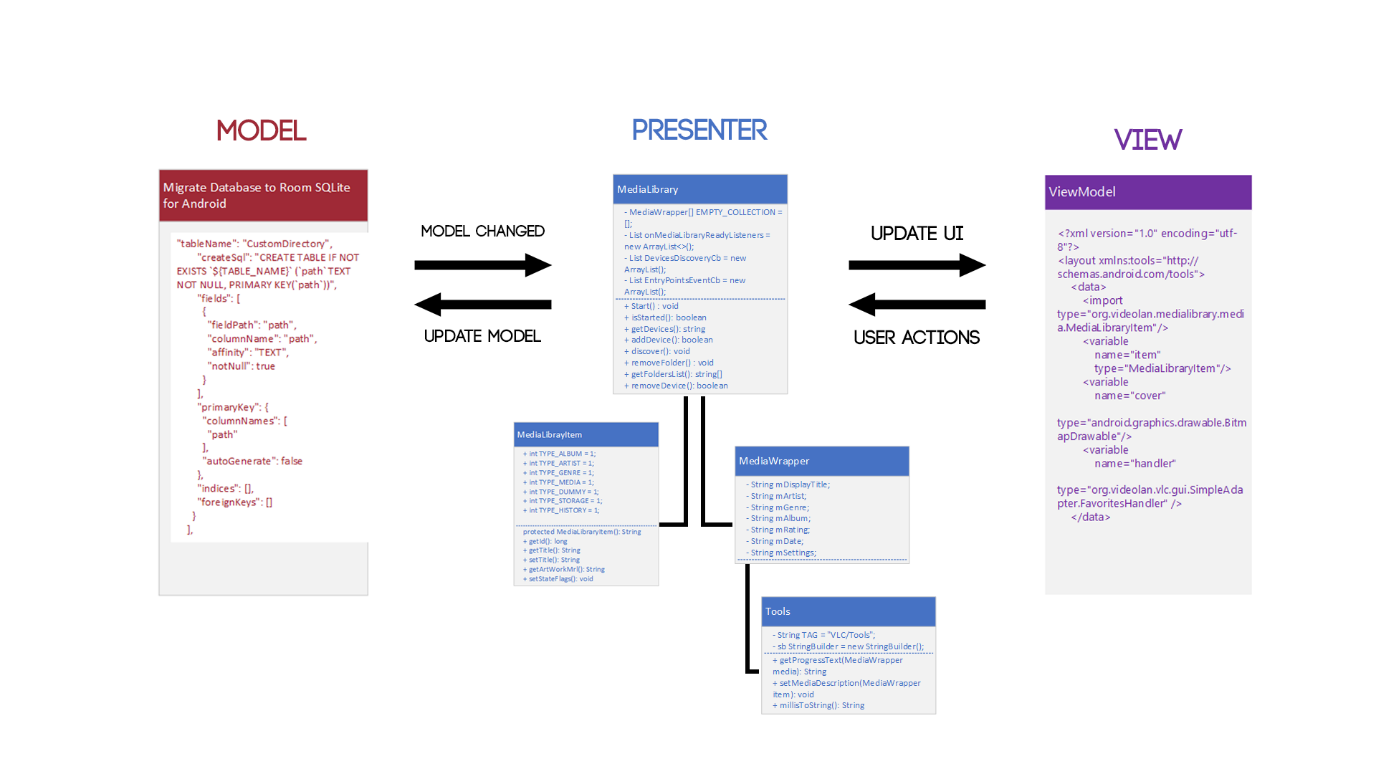
MVP in android is mainly used for making UIs such as VLC android. It also makes unit testing easier.

The Presenter is in between Model and the View. And, it triggers the business logic, and lets to know ‘the View’ when to update.

It recovers data received from the Model and shows it in the View.

It interacts with the Model, then fetches and transforms the data from the Model to update the view.



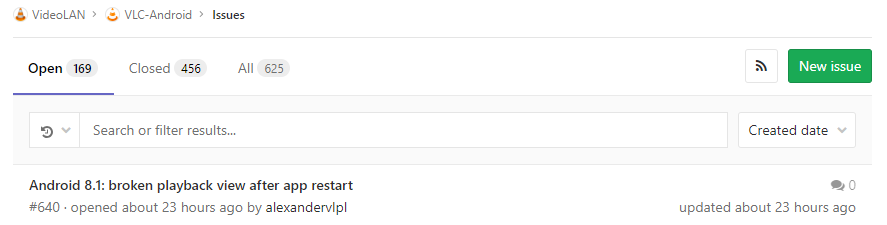


Software Quality Attributes

## Testability

VLC android is easily testable as its accessible from GITHUB, the software has an inbuilt bug tracker that stores and recorders errors. This particular software has had over 100 million downloads on the google play store with 1 million reviews, all on different android devices old and new.

[https://code.videolan.org/videolan/vlc-android/issues](https://code.videolan.org/videolan/vlc-android/issues%5C)



<https://forum.videolan.org/> https://lh3.googleusercontent.com/E1p5d58Z9dWmgcsCWKUiIVN2EZPq-6yNrbk4CXlvRmDcTC8IXoPbWUgQyx9s4Ss3oOvhw3iQti3Hqze65Gfljz4eiXyyspn-aROnAwNuvYxGuu6ep6avM0gEOs93s5UHp4X_rle1

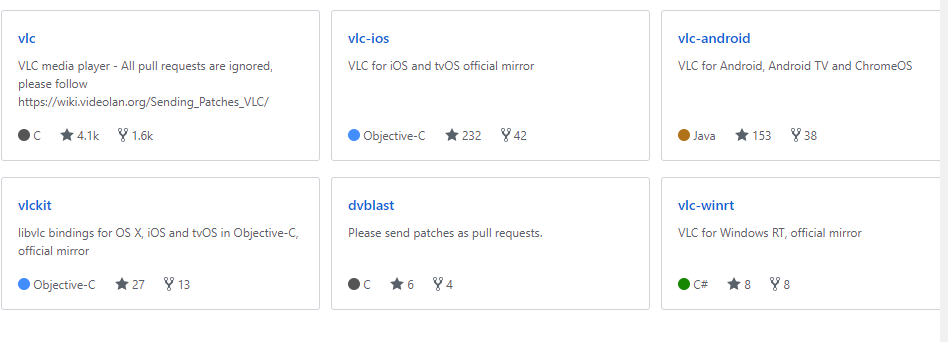
These are the two forums to report bugs and glitches with the software. Problems are addressed quickly with a lot of discussions on possibility.

The code is extremely easy to compile and run as everything is provided for, so anyone can test the software. I used Android Studio and imported the file.

The main chunk of the software is very basic to understand, using the MVP method. It was a bit difficult to navigate all the files as they are named “src” and “jni”. But once you know what you're looking for, it was very easy to find and identify the model, view and presenter models and subsequent classes.

VLC android on desktop is C language, in order for it to be multipurpose on different platforms. The code had to be rewritten into Java, C# and Objective C.

## Portability



Being open source; everyone with different skill sets in different languages can make contributions to the software. For example: VLC android has 91 contributes with vast experience. The language used was Java and there is program in place to convert Java to C++. So, the software is interchangeable and portable between platforms and operating Systems.

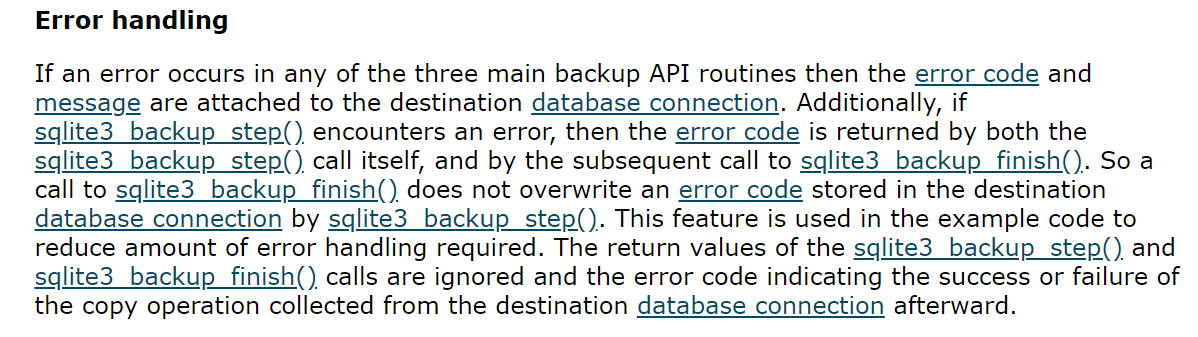
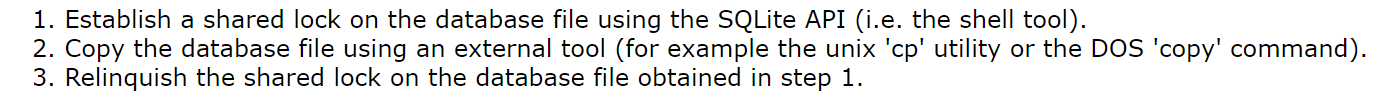
The basis of the software using MVP is relatively simple but with changing GUI’s, API’s and Database platforms it would be easy to create the same pattern but harder to implement it.

The software could not be easily executed on a different OS originally. Until it was rescaled and rebuilt on the other platforms.

## Recoverability

VLC android uses SQLite Database and API. This specific database comes with recovery options that backup the system if there is major failure. It also records and stores errors

<https://www.sqlite.org/backup.html>



Having VLC android on GITHUB means that the source files are permanently stored online. When changes are suggested they go through a vetting process by other experienced programmers. This means that major failure is very unlikely but if it does happen, the database can roll back the mistakes to be fixed.

The software also has API scaling in place in which if older software running older API can run on new devices.

Software Metrics Analysis

## LOC:

<https://en.wikipedia.org/wiki/Source_lines_of_code>

Source lines of code (SLOC), also known as lines of code (LOC), is a [software metric](https://en.wikipedia.org/wiki/Software_metric) used to measure the size of a [computer program](https://en.wikipedia.org/wiki/Computer_program) by counting the number of lines in the text of the program's [source code](https://en.wikipedia.org/wiki/Source_code). SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate [programming productivity](https://en.wikipedia.org/wiki/Programming_productivity) or [maintainability](https://en.wikipedia.org/wiki/Maintainability) once the software is produced.

LOC measures are somewhat controversial, particularly in the way that they are sometimes misused. That is, programs with larger LOC values take more time to develop. Thus, LOC can be very effective in estimating effort. However, functionality is less well correlated with LOC: skilled developers may be able to develop the same functionality with far less code, so one program with fewer LOC may exhibit more functionality than another similar program. LOC is a poor productivity measure of individuals, since a developer can develop only a few lines and yet be far more productive in terms of functionality than a developer who ends up creating more lines (and generally spending more effort). Good developers may merge multiple code modules into a single module, improving the system yet appearing to have negative productivity because they remove code. Also, especially skilled developers tend to be assigned the most difficult tasks, and thus may sometimes appear less "productive" than other developers on a task by this measure. Furthermore, inexperienced developers often resort to code duplication, which is highly discouraged as it is more bug-prone and costly to maintain, but it results in higher LOC.

In this assignment, we are doing a small project, there just about 1700 lines of code, that makes our project easy to check bug, and lower cost.

## NOC:

<http://support.objecteering.com/objecteering6.1/help/us/metrics/metrics_in_detail/number_of_children.htm>

## Number of Children (NOC)

## Overview

Inheritance, otherwise called generalization, is one of the fundamental concepts of object models, and must be used advisedly. Non-abusive use is a sign of quality and a good understanding of the concept. A class from which several classes inherit is a sensitive class, to which the user must pay great attention. It should, therefore, be limited, notably for reasons of simplicity.

## Computation

For a class, this is the number of child classes.

For a package, this is the number of child packages.

## Nominal range

Between 1 and 4.

## Analysis

The upper and lower limits of 1 and 3 correspond to a desirable average. This will not stop certain classes being the kind of utility classes which provide services to significantly more classes than 3.

In our project, most classes NOC is 0, only has one class has 8 NOC, because that class has 8 complexities, that make our project in this part has good quality and a good understanding of the concept.

## 3. WMC (Cyclomatic Complicity)

<https://pdepend.org/documentation/software-metrics/weighted-method-count.html>

<http://www.arisa.se/compendium/node97.html>

## WMC - Weighted Method Count A (the base metrics)

The Weighted Method Count or Weighted Method per Class metric was originally defined in A Metrics Suite for Object Oriented Design by Chidamber & Kemerer.

The WMC metric is defined as the sum of complexities of all methods declared in a class. This metric is a good indicator how much effort will be necessary to maintain and develop a class. There are three slightly different definitions of the WMC, where each definition uses another metric as a measure of the methods' complexity. Possible complexity values are:

* McCabe's Cyclomatic Complexity
* Lines of Code
* 1 (Number of Methods or Unweighted WMC)

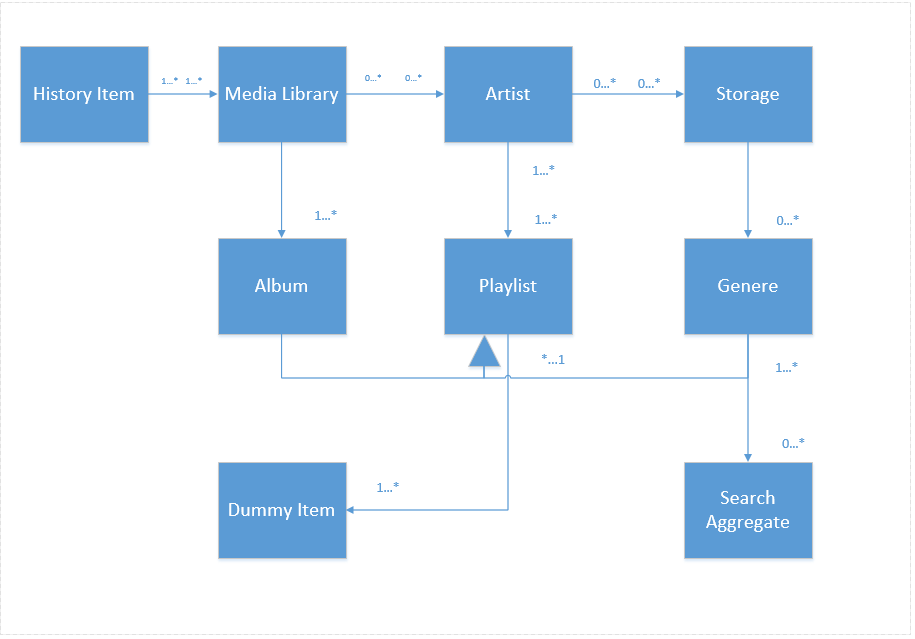
PHP\_Depend uses the sum of Cyclomatic Complexity Numbers of all methods and constructors declared in a class to calculate the WMC metric. A lower WMC usually indicates to a class with better abstraction and polymorphism. While a class with a high complexity value is a good indicator that it this class is very application specific and does more than one job, and therefore harder to test, reuse and maintain.

## Thresholds

An appropriate threshold for the WMC lower limit is 1, because a class should at least consist of one method. An upper limit for the WMC of a class is harder to define but it seems that an upper limit of 50 is a good reference point for most projects that start to use the Weighted Method Count metric.

In our project most class has good WMC is in the good range, only has one class got high WMC, because that class has high parameters (NOP).

## Coupling Between Objects (CBO)

****

History Item CBO =1

Media Library CBO = 3

Artist CBO = 3

Storage CBO = 2

Album CBO = 2

Playlist CBO =3

Genera CBO = 2

Dummy Item CBO = 1

Search Aggregate CBO = 1

# References

(n.d.). Retrieved from https://lh3.googleusercontent.com/PES21T6lSyGV3qG-xUGXQjPnrZssxIkquURtY1s8occr6jnoK9RbhKqduDCS-NAHLUg=w720-h310

*Advantage of MVP in Android*. (2016). Retrieved from Stack Overflow: https://stackoverflow.com/questions/40766185/advantage-of-mvp-in-android

Maxwell, E. (2017, January 26). *MVC vs. MVP vs. MVVM on Android*. Retrieved from Realm: https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/

VLC. (n.d.). *VideoLAN*. Retrieved from https://images.videolan.org/vlc/screenshots/android/library-video-landscape-medium.jpg

VLC. (n.d.). *VLC for Andoid*. Retrieved from VideoLAN: https://www.videolan.org/vlc/download-android.html

VLC. (n.d.). *VLC for Android*. Retrieved from Google Play Store: https://play.google.com/store/apps/details?id=org.videolan.vlc&hl=en

Software engineering 9th Edition by Ian Sommerville

<https://www.spaceotechnologies.com/mvp-android-architectural-pattern/>

<https://academy.realm.io/posts/eric-maxwell-mvc-mvp-and-mvvm-on-android/>

<https://mindorks.com/course/android-mvp-introduction>

<https://antonioleiva.com/mvp-android/>

<https://www.techyourchance.com/mvp-mvc-android-1/>

<https://www.youtube.com/watch?v=SGr5GrEndQI>

<https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=513803>

<https://en.wikipedia.org/wiki/List_of_system_quality_attributes>

<https://www.dropbox.com/sh/ijbyig8ls6rctp4/AAC8Tw2acGXkcbmQXBrIqrDxa?dl=0>

# Team Roles

Michael – Leader

The leader and did tasks 4 and 5 in the assignment sheet which involved identifying the architectural pattern/style and why it was applied and was able to finish it successfully.

Louis – Member

He did tasks 1 and 3 in assignment sheet which involves use case diagram and architectural connectors and components. Successfully did the tasks assigned to him.

Brayden – Member

He did tasks 6 and 7 in the assignment sheet which involves architectural configuration and software quality attributes. Overall finished it successfully.

Demin – Member

He did task 8 in the assignment sheet which involves software metrics. Not sure if done correctly but it looks like he made effort.

Amar – Member

He did task 2 in the assignment sheet which involved class diagram. Overall finished it but according to other members, it is done poorly, and it took way too long to finish. Other parts suffered because of this.