An Android Studio Plugin for Calculating and Measuring Code Complexity Metrics in Android Applications

by

Bo Wang

A thesis in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE

Major: Computer Science

Program of Study Committee:
Josh Dehlinger, Major Professor

Department of Computer and Information Sciences

Towson University

Towson, Maryland

2015

TOWSON UNIVERSITY
OFFICE OF GRADUATE STUDIES

THESIS APPROVAL PAGE

This is to certify that the thesis prepared by Bo Wang [INSERT Student's Name]

entitled_ [INSERT Title of Thesis] An Android Studio Plugin for Calculating and Measuring Code Complexity Metrics in Android Applications

has been approved by the thesis committee as satisfactorily completing the thesis requirements for the degree _ [INSERT Type of Degree] Master of Science
(for example, Master of Science)

[illegible]	Josh Dehlinger	12/15/15
Chairperson, Thesis Committee Signature	Type Name	Date
[illegible]	Siddharth Kaza	12/15/15
Committee Member Signature	Type Name	Date
[illegible]	Ziying Tang	12/15/2015
Committee Member Signature	Type Name	Date
Committee Member Signature	Type Name	Date
Committee Member Signature	Type Name	Date
Janet V. DeLany	Janet V. DeLany	1-11-16
Dean of Graduate Studies	Type Name	Date

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

One of the most widely used mobile operating systems (OS) in the world today is Google’s Android OS. With the increase in popularity of developing mobile applications, the Android development environment has progressed from using Eclipse to Android Studio. Android Studio is an integrated development environment (IDE) for developing Android applications but is currently lacking plugin tools to enable quality assurance of Android applications. This thesis includes the studies of complexity metrics and how they can help developers to create user-friendly and high-quality Android applications. The results from this thesis include an Android Studio plugin to support code complexity metrics that analyze Android application source codes. The plugin helps Android Studio developers in the future to create their applications and help researchers to study more about code metrics.

CHAPTER 1. INTRODUCTION

When people want to access the Internet today, most people will pull out their mobile devices to perform tasks. Mobile operating systems play the same role as computer operating systems. They provide a user interface platform that offers a wide range of services to users. For most people, mobile operating systems are more convenient and easier to use, allowing them to access information around the globe. They can check e-mails, browse the web, and communicate with friends. According to Danyl Bosomworth, the co-founder and Marketing Director of smartinsights.com, by the end of 2015, there will be approximately 1.9 billion mobile users around the world, which will surpass desktop users [2]. To put that in perspective, about a quarter of the world's population uses mobile devices. Not all of them have advanced smart phones and Internet access, but the number of mobile users has been increasing rapidly in the past decade.

1.1 Android Applications

The Android operating system is an open source platform that is currently the most popular mobile operating system in the world today. According to Jason Hahn, a journalist from digitaltrends.com, Android delivered 1.1 billion mobile devices in 2014, or 81.5 percent of the global market share [4]. In comparison, the second place was Apple's iOS, which delivered 192.7 million mobile devices, or 14.8 percent of the global market share. Samsung is the top manufacturer for Android phones, which has more phones delivered than the next five vendors combined.

One of the reasons why people like using mobile devices is because of the wide range of applications they offer. Google Play is the mobile application store where users can download applications on their Android phones. It has approximately 1.5 million apps available for download, which makes it the leading app store as of May 2015, according to statista.com, a website that analyzes statistics and sources [10]. With a massive market like mobile devices and applications, researchers have been studying and investigating methods to make application development more interesting and appealing to software programmers.

1.2 Android Studio IDE

One of the most popular integrated development environments (IDE) to build Android applications for the past few years is the Android Development Tools (ADT) for Eclipse. This is an extension for Eclipse that allows users to create, maintain, and debug Android apps. Then Google decided to develop its own IDE for Android development, and then Android Studio was released. This IDE is based on JetBrain's IntelliJ IDEA and became the first official IDE for Android development across Windows, Mac OS and Linux. It has most of the features from IntelliJ, and it offers new features like a new build system, code templates, and support for Google Cloud. Google also plans to support and focus fully on Android Studio to make it better and faster, and it will stop supporting other IDE's at the end of 2015, according to Android Product Manager Jamal Eason [11]. Since it is fairly new, it lacks supportive tools and plugins in its marketplace. Many useful tools have not been integrated or implemented in Android Studio. Android developers are in need of supportive plugins to analyze and maintain their applications.

CHAPTER 2. BACKGROUND

The work described in this thesis builds upon on the code complexity metrics for Android applications. This chapter discusses the background information and related work in the areas of what are metrics, what types of metrics are there, and how code complexity metrics can help developers to create high quality Android applications. In addition, the plugin development environment used to create the plugin for Android Studio is discussed.

2.1 Code Complexity Metrics

Building and maintaining mobile applications can be very challenging for developers. Without the proper tools for assistance, reading and understanding the source code can be difficult. If developers want to improve the quality of their applications, they must statically analyze the complexity of their source code so that they can refactor and reorganize it if necessary. Developers who maintain and test applications must also analyze the complexity of the source code in order to improve the readability and maintainability of the applications. Code complexity metrics are essentially statistics that are calculated using static analysis [12]. There are many types of metrics that have been researched and established to help developers to design and implement high quality applications. For Android applications, in particular, user interface and user experience are the most important factors to achieve high quality designs. This is where complexity metrics analyzing tools and plugins come in to help. These tools make it easy to analyze the complexity of the source code, and display metric results for developers to analyze.

For example, tools can have features like counting number of methods in a class, number of classes in a package, lines of code, number of subclasses, levels of the inheritance tree, and cyclomatic complexity, etc. These features can help developers to understand the source code from a high level point of view before doing any maintenance or refactoring work on the applications.

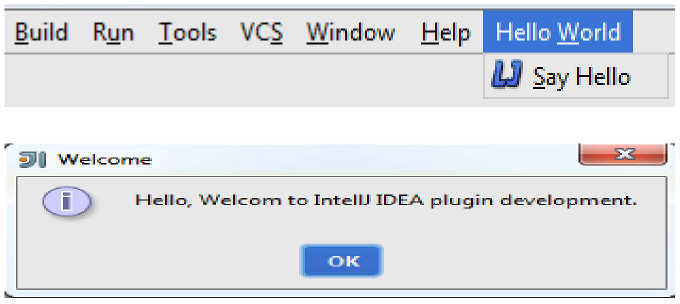
Most code complexity metrics are very common and can be used in all types of applications. However, there are very few of metrics that have been researched and proven to work for Android applications. One of the metrics for Android applications is UI complexity. According to the research paper "An Exploratory Study on the Relation between User Interface Complexity and the Perceived Quality of Android Applications" by authors Taba, Keivanloo, Zou, and Ng, the user interface has a significant impact on user-perceived quality of Android applications [12]. The authors conducted a study of analyzing open source Android applications with their ratings and popularity factors. They found out that the number of inputs and outputs in an activity can impact the usability and user-friendly of Android applications. Each activity is essentially a mobile screen that allows users to interact with the applications. For inputs, users can press buttons, enter texts, choose radio buttons, toggle switches, pick dates, click images, check boxes, and more. For outputs, the applications display result views to the users. These results include text views, list views, grid views, images, progress bars, and group views. The authors believe that when the applications have activities with too many inputs and outputs, it is more likely that the ratings are lower, and the interfaces have poor usability and become less user-friendly.

2.2 Plugin Development Environment

The plugin development environment for creating plugins for Android Studio can be set up in IntelliJ IDEA, using either the Community Edition or Ultimate as the IDE. Both editions have the complete set of plugin development tools. The complete guide on how to set up IntelliJ IDEA and the plugin development environment can be found in the IntelliJ Platform SDK documentation on JetBrains website [5].

The IDE used to develop the thesis plugin is the IntelliJ IDEA Community Edition 14.1.5. When creating a new project, there is an option to create the project as a plugin project using the IntelliJ Platform Plugin SDK. There are also other options to set up the JDK and SDK versions.. To verify the plugin project is configured correctly, according to Lakjeewa, you may write a simple Say Hello plugin for testing to make sure everything is working as expected [7].

Once the project is set up correctly, there should be a green right arrow icon on the top of the IDE, which can be clicked to run the plugin, and a new IntelliJ IDEA will open to display the plugin results. For the Say Hello plugin, a new tab called Hello World is created on the top of the IDE, with a dropdown option called Say Hello. When user clicks on it, a pop up window opens with the message displays in Figure 5.



Hello, Welcom to IntelliJ IDEA plugin development.

Figure 1 - Say Hello plugin example

CHAPTER 3. RESEARCH APPROACH / METHODS

This section expands on the purpose of the thesis and defines the problem domain. The goals and objectives of researching are discussed, as well as the approach and expected results. Then the design and features of the plugin are shown. Lastly, a demonstration of how to deploy and use the plugin will be included in this section.

3.1 Problem Definition

In today's society, mobile devices will soon take over personal computers as the most popular way for people to communicate. The number of mobile applications that are out there for people to download and use has increased rapidly in the past few years. These applications provide quality services and user-friendly interfaces to their users. The more downloads and uses of an application, the more money it makes for the developers. In order to develop high-quality and user-friendly mobile applications, developers must have the best resources possible.

The three biggest mobile platforms/operating systems are Google's Android, Apple's iOS, and Microsoft's Windows. They all have their advantages and disadvantages in terms of speed, security, available apps, interfaces, etc. It is up to the users to decide which one is the best for them. As developers, we care more about what language is used to build the applications, what type of IDE does it use, how supportive is the development community, and what other types of tools are available.

Google's Android has been consistently the most popular mobile operating system in the world. The increase in popularity of Android has gotten many more developers interested in learning Android development. The development process and tools for implementing Android applications have been changing in the last several years. The most popular IDE has changed from Eclipse to Android Studio. The complexity of the applications has changed as well to make them more competitive in the market. Even for the language itself, Java has changed and gone through a few releases and versions.

As mentioned, the new primary IDE for Android development is the new Android Studio, which was released in 2014. It is still fairly new to the market, but it has many good features that developers will get to use, and it is increasing in popularity all over the world. It is available to use on Windows, Mac, and Linux, so there are a lot of developers out there who will get their hands on this IDE.

The problem with Android Studio is that it is lacking support tools and plugins in its marketplace. There are currently only 591 plugins available in its plugin repository. In comparison, IntelliJ IDEA has 1558 plugins and Eclipse has 1823 plugins available to the public [3]. There are many open opportunities and ideas available for developers to implement plugin tools for Android Studio. The second potential problem and research topic is to explore and find out what types of plugin tools are available and not available, and then design a new plugin based on the research. The last problem is to research on how to implement a plugin for mobile applications, and then eventually implement it.

3.2 Objectives

The main objective of the thesis is to research, design and develop a complexity analysis plugin tool for the new Android Studio IDE. In order to meet the main objective, there are three sub-sections that will be completed.

The first objective of the thesis will be involved in researching and learning about Android mobile application development. This includes becoming familiar with Android applications, how they are developed, and the source code behind the applications. It also includes researching about the history of mobile application development. How it has evolved over the years, what type of framework does it use, and what is the popular way to develop mobile applications today.

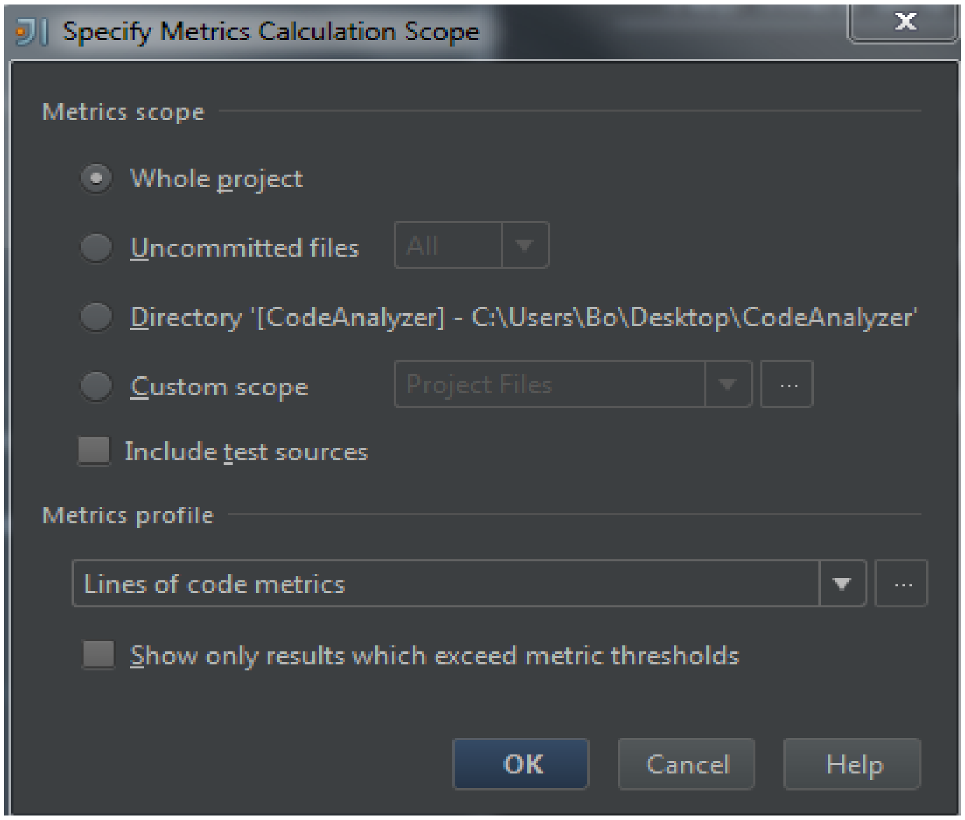
The second objective is to research about complexity metrics and static analysis plugins. These plugins are used in many popular integrated development environments to help developers and testers to analyze the complexity of the applications that they are working on. The goal in this objective is to find out which complexity analysis plugins are already available. Then information will be collected on what features will be included in the plugin that will be developed. The objective is to design a plugin that can help future developers and testers to maintain their applications.

The third objective is to develop the plugin, and have it ready to deploy in the Android Studio marketplace. After collecting information for the requirements, a design and layout of the plugin will be completed. Once the design report is completed, the implementation stage follows. Java will be used as the primary language to develop the plugin. The implementation stage will be divided into several iterations.

3.4 Existing Plugins

Even though Android Studio is fairly new and is currently lacking static analysis tools for Android applications, there are a few similar existing plugins out there for other IDE's. Two plugins for IntelliJ IDEA were used as examples to design and implement the plugin for Android Studio.

Metrics Reloaded is an IntelliJ IDEA that automates code metrics for all types of projects. It offers more than 250 metrics that help developers in static analysis of source code. When users run the plugin, a new window will show up that provides a list of metrics that users can perform on the current project. Figure 1 shows the plugin window. Some metrics include counting classes, complexity metrics, JUnit testing, lines of code, and Javadoc coverage metrics. The plugin also allows users to choose the scope of metrics to run on the source code, including whole project, package, files, etc. The results of the code metrics are displayed in a tool window at the bottom of IntelliJ IDEA. Figure 2 shows the tool window with the metric results. Overall, Metrics Reloaded offers variety of code metrics along with a user-friendly interface. The interface allows users to perform code metrics through different copes of the project, which is useful when users want to only run the plugin on a portion of the source code, instead of the entire project.


Specify Metrics Calculation Scope

Metrics scope

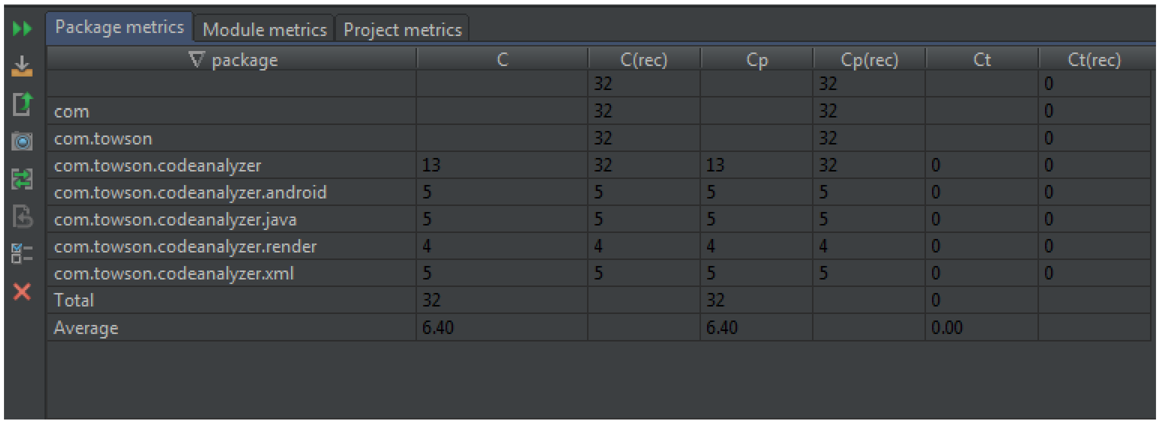
- [✓ Whole project]
- Uncommitted files: All
- Directory '[CodeAnalyzer] - C:\Users\Bo\Desktop\CodeAnalyzer'
- Custom scope: Project Files
- ☐ Include test sources

Metrics profile

Lines of code metrics

☐ Show only results which exceed metric thresholds

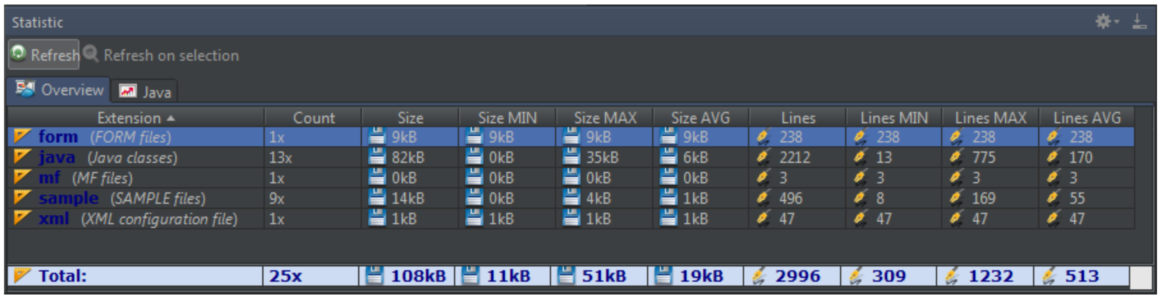
Figure 2 - Metrics Reloaded plugin, start-up window



package	C	C(rec)	Cp	Cp(rec)	Ct	Ct(rec)
		32		32		0
com		32		32		0
com.towson		32		32		0
com.towson.codeanalyzer	13	32	13	32	0	0
com.towson.codeanalyzer.android	5	5	5	5	0	0
com.towson.codeanalyzer.java	5	5	5	5	0	0
com.towson.codeanalyzer.render	4	4	4	4	0	0
com.towson.codeanalyzer.xml	5	5	5	5	0	0
Total	32		32		0	
Average	6.40		6.40		0.00	

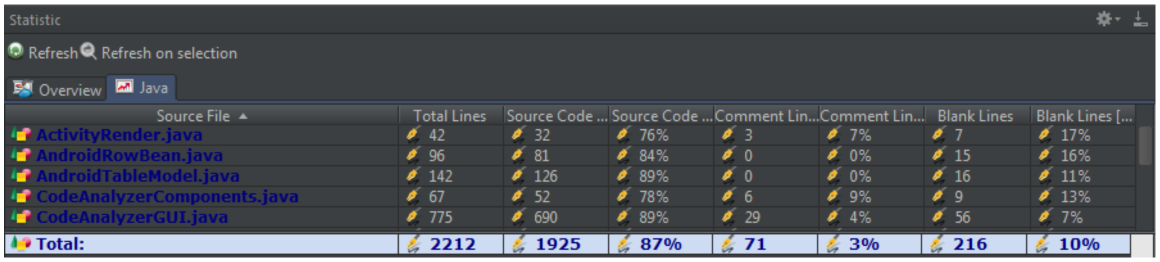
Figure 3 - Metrics Reloaded plugin, metric results tool window

Statistic is also a plugin for IntelliJ IDEA that displays project statistics for developers to understand and analyze the source code. Comparing to Metrics Reloaded, Statistic offers less code metrics. It does not have a pop up window when users run the plugin, and users cannot which code metrics to run. Instead, when users run the plugin, the results tool window shows up with an Overview and Java tabs on top. The users can choose to see the metric results for the entire project by clicking on the Overview tab, or they can choose to see only Java files metric results by clicking on the Java tab. Figure 3 shows the Overview tab metric results, and Figure 4 shows the Java tab metric results. In addition, users can also double click each type of file to open the file, and use the Refresh button to re-run the plugin. This provides a dynamic user experience because users can interact with the metric results. Overall, the design and layout of the tabs are easy to use and to understand. The plugin displays useful statistics of the entire project and java files.



Extension ▲	Count	Size	Size MIN	Size MAX	Size AVG	Lines	Lines MIN	Lines MAX	Lines AVG
form *(FORM files)*	1x	9kB	9kB	9kB	9kB	238	238	238	238
java *(Java classes)*	13x	82kB	0kB	35kB	6kB	2212	13	775	170
mf *(MF files)*	1x	0kB	0kB	0kB	0kB	3	3	3	3
sample *(SAMPLE files)*	9x	14kB	0kB	4kB	1kB	496	8	169	55
xml *(XML configuration file)*	1x	1kB	1kB	1kB	1kB	47	47	47	47
Total:	**25x**	**108kB**	**11kB**	**51kB**	**19kB**	**2996**	**309**	**1232**	**513**

Figure 4 - Statistic plugin, Overview tab with metric results



Source File ▲	Total Lines	Source Code ...	Source Code ...	Comment Lin...	Comment Lin...	Blank Lines	Blank Lines [...
ActivityRender.java	42	32	76%	3	7%	7	17%
AndroidRowBean.java	96	81	84%	0	0%	15	16%
AndroidTableModel.java	142	126	89%	0	0%	16	11%
CodeAnalyzerComponents.java	67	52	78%	6	9%	9	13%
CodeAnalyzerGUI.java	775	690	89%	29	4%	56	7%
Total:	**2212**	**1925**	**87%**	**71**	**3%**	**216**	**10%**

Figure 5 - Statistic plugin, Java tab with metric results

3.5 Plugin Design

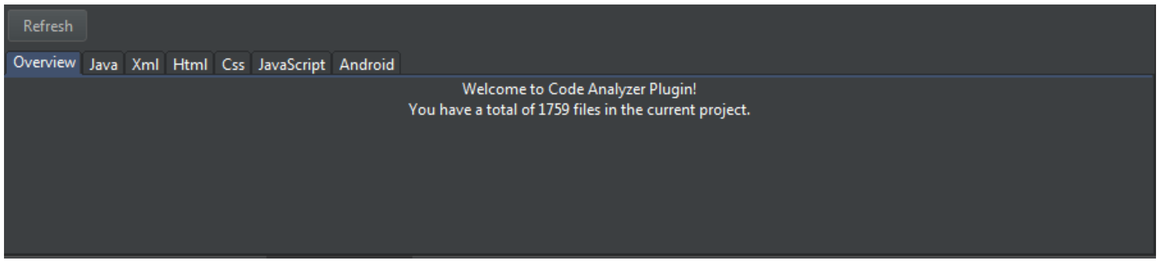
The design of the plugin is mainly based on Metrics Reloaded and Statistic plugins that were discussed in the previous section. To run the plugin, users must click a select or button in one of the tabs on the top of Android Studio. Logically, all the tools are available in the View > Tool Windows menu, therefore, the plugin icon for running it will be placed in there. Next, the metric results window will be the same like all other plugins, which will be a built-in window at the bottom of the IDE. The results of metrics will be displayed in tabs, similar to Statistic plugin. There should be an Overview tab that displays a few messages to the users, and there should be other tabs that calculate the metrics in different types of files. Lastly, in each tab there should be columns and rows that organize the metric results so that users can easily identify the purpose of each metric of their source code.

After clicking the Code Analyzer icon in View > Tool Windows menu to start the plugin, the users will be taken to the first tab, which is the Overview tab. It displays a welcome message, and it prints out the total number of virtual files and folders in the current opened project. See Figure 6 for the Overview tab window. The next tab is the Java tab, which prints out the names, total line count, source code line count, and method count of each Java file. The source code line count excludes the comments lines in the files. There is also a total row at the bottom of the tab, which sums up the totals of each column. See Figure 7 for the Java tab window.

The next few tabs in the plugin are Xml, Html, Css and JavaScript, which all share the same functionality. Each these tabs, the names of each type of files are printed, along with the total line count and source code line count. A total row is also displayed at the bottom that sums up each category. See Figure 8 for the Xml tab window.

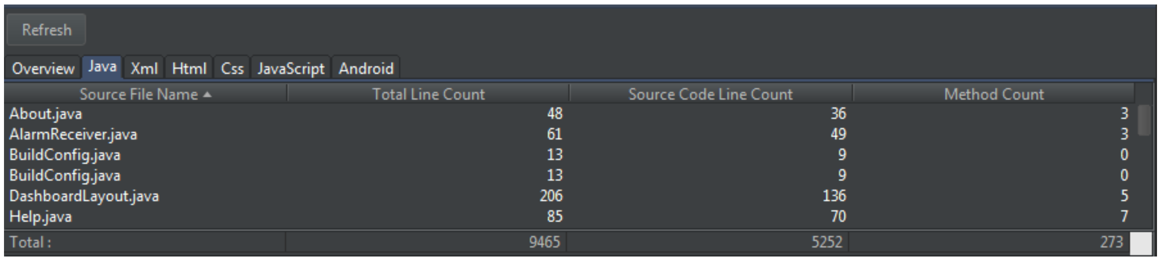
The last tab is the Android tab, and it is specific only for displaying metrics for Android applications. There are four columns in this tab: Activity Name column displays the names of Java files that define activities using the setContentView() method call. The Xml Layout File column displays the names of Xml files that are associated with the activities in Activity Name column. The Inputs Count column displays the number of inputs in each Xml layout file. The inputs being considered are: Button, EditText, RadioGroup, RadioButton, ToggleButton, DatePicker, TimePicker, ImageButton, CheckBox, and Spinner. The Outputs Count column displays the number of outputs in each Xml layout file. The outputs being considered are: TextView, ListView, GridView, View, ImageView, ProgressBar, and GroupView. By each Activity Name, a warning icon is displayed when the total count of inputs and outputs is greater than 10. When a user hover their mouse over the warning icon, a message is displayed that there are too many inputs and outputs in the associated xml layout file. See Figure 9 for the Android tab window.

In addition, there is also a Refresh button on the top of the tabs. It allows users to re-run the plugin during development of their applications. For example, when a user deletes a Java file, adds some inputs to the xml file, or adds an activity to a Java file, by clicking the Refresh button, the plugin will re-calculate the metrics and display newly updated metrics to the user. The files in all tabs are also double-clickable so that users can open the files directly using the plugin. Furthermore, all the columns in the tabs are sortable, so that the users can sort the columns in ascending or descending order.



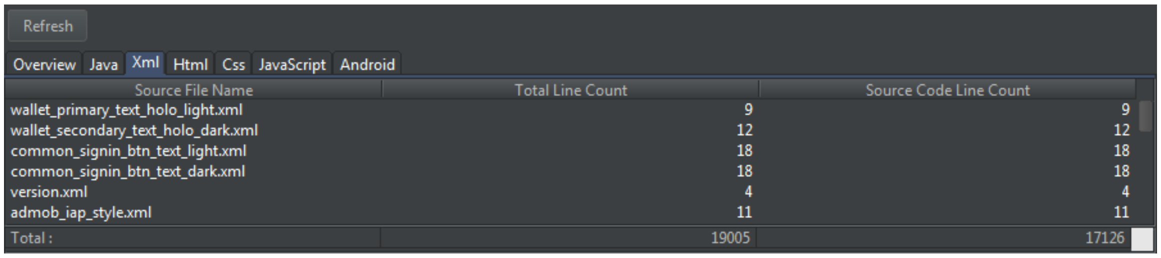
Welcome to Code Analyzer Plugin!
You have a total of 1759 files in the current project.

Figure 6 - Overview tab window



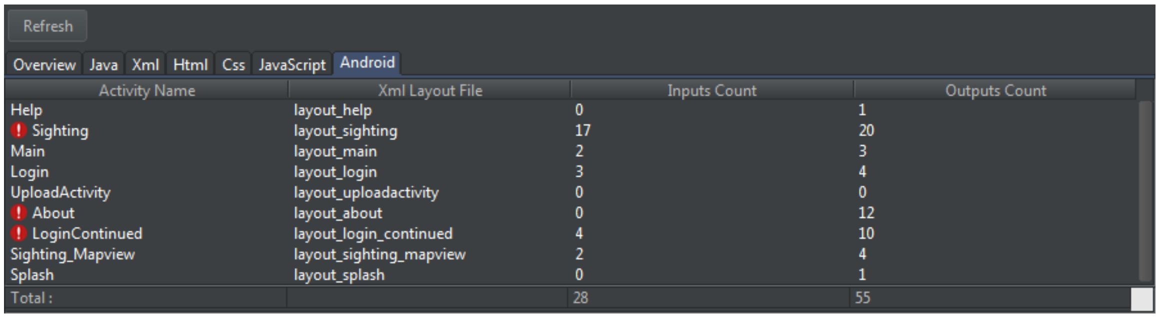
Source File Name ▲	Total Line Count	Source Code Line Count	Method Count
About.java	48	36	3
AlarmReceiver.java	61	49	3
BuildConfig.java	13	9	0
BuildConfig.java	13	9	0
DashboardLayout.java	206	136	5
Help.java	85	70	7
Total :	9465	5252	273

Figure 7 - Java tab window



Source File Name	Total Line Count	Source Code Line Count
wallet_primary_text_holo_light.xml	9	9
wallet_secondary_text_holo_dark.xml	12	12
common_signin_btn_text_light.xml	18	18
common_signin_btn_text_dark.xml	18	18
version.xml	4	4
admob_iap_style.xml	11	11
Total :	19005	17126

Figure 8 - Xml tab window



Activity Name	Xml Layout File	Inputs Count	Outputs Count
Help	layout_help	0	1
❗ Sighting	layout_sighting	17	20
Main	layout_main	2	3
Login	layout_login	3	4
UploadActivity	layout_uploadactivity	0	0
❗ About	layout_about	0	12
❗ LoginContinued	layout_login_continued	4	10
Sighting_Mapview	layout_sighting_mapview	2	4
Splash	layout_splash	0	1
Total :		28	55

Figure 9 - Android tab window

3.6 Plugin Installation

In order to deploy and test the plugin in Android Studio, the plugin project needs to be built and an executable jar file must be created. According to the IntelliJ Platform SDK documentation, there is an option under the Build tab in IntelliJ IDEA called Prepare Plugin Module 'Project Name' For Deployment. By clicking it, an executable jar file will be created to the build path that is given to the IDE. This jar file is used for deployment in Android Studio to install the plugin locally, or the jar file can be uploaded to Android Studio plugin repository if it is ready for the public.

The jar file of the plugin can be downloaded at my Github repository:

https://github.com/bowang616/CodeAnalyzer

Once the jar file has been downloaded to your local disk, follow these steps:

1) Open Android Studio
2) Open an Android project
3) Click File > Settings, the "Settings" window opens
4) Click Plugins, then click Install plugin from disk…
5) "The Choose Plugin File" window opens
6) Choose the CodeAnalyzer.jar file that you downloaded
7) Click Ok, then click Apply. Click Restart
8) Click View > Tool Windows > Code Analyzer
9) The plugin is started with the results on the bottom window

3.7 Future Work

In this thesis, researching has been done on the complexity metrics of Android applications. There have been many metrics and methods on how to improve the quality of regular projects, but not much has been discovered for Android projects. The groundwork has been conducted in this thesis to research about the available metrics and methods to improve the quality of design and usability of Android applications. This is only the basic foundation and theory of how metrics can help developers to analyze and create better applications. There is still a much larger area of researching and studying on other complexity metrics and static analysis techniques that can help to improve the quality of Android applications. In the future, more metrics specifically for Android applications will be discovered and adapted to this thesis.

In the development of the plugin, the basic layout of the results window has been established. More features and metrics can be added to the plugin and the plugin is very simple to modify. The packages used in plugin development have been imported to the source code so that it is easy to implement more features using those. In the future, the plugin will add more features and metrics that will be discovered during the research phase of the thesis.

CHAPTER 4. CONCLUSION

Android Studio has been released only about 2 years ago, yet it is already overtaking Eclipse as the most popular IDE to develop Android applications. However, the Android Studio marketplace for plugins and tools is still incomparable to Eclipse or even IntelliJ IDEA. There is a huge demand of plugins by Android Studio users to implement and maintain their applications. Code complexity metrics are calculated to help developers to analyze source code to improve the quality of the applications. In Android Studio marketplace, there is few to none available to assist developers with code metrics. This thesis conducted case studies and research on Android specific metrics, and implemented a static analysis and code complexity plugin for Android Studio based on the research and design of Android specific, mobile application complexity metrics. Users can now download the plugin and install it in their Android Studio to read and understand the source code of their applications.

BIBLIOGRAPHY

[1] "Android Studio Website." *Android Studio and SDK Tools*. Web. 12 June 2015. <https://developer.android.com/sdk/index.html>.

[2] Bosomworth, Danyl. "Mobile Marketing Statistics 2015." *Smart Insights*. Smart Insights, 1 May 2015. Web. 25 June 2015. <http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/>.

[3] "Eclipse Plugins, Bundles and Products - Eclipse Marketplace." Eclipse Plugins, Bundles and Products - Eclipse Marketplace. Web. 13 Dec. 2015. <https://marketplace.eclipse.org/>.

[4] Hahn, Jason. "Android Claims 81.5% of the Global Smartphone OS Market in 2014, IOS Dips to 14.8%." *Digital Trends*. 20 Feb. 2015. Web. 18 June 2015. <http://www.digitaltrends.com/mobile/worldwide-domination-android-and-ios-claim-96-of-the-smartphone-os-market-in-2014/>.

[5] Jemerov, Dmitry. "IntelliJ IDEA Plugin Development." *IntelliJ IDEA*. Ed. Nikolay Chashnikov. 3 Dec. 2014. Web. 12 June 2015.

[6] Jost, Gregor, Jernej Huber, and Marjan Hericko. "Using Object Oriented Software Metrics for Mobile Application Development." *CEUR Workshop Proceedings*. University of Maribor, 17 Sept. 2013. Web. 17 June 2015. <http://ceur-ws.org/Vol-1053/sqamia2013paper3.pdf>.

[7] Lakjeewa, Madhuranga. "IntelliJ Plugin Development Tutorial - Hello World Plugin."IntelliJ Plugin Development Tutorial - Hello World Plugin. Cyberspace, 20 Dec. 2014. Web. 1 Sept. 2015. <http://lakjeewa.blogspot.com/2014/12/intellij-plugin-development-tutorial.html>.

[8] Memon, Atif. "Test Cost-Effectiveness and Defect Density: A Case Study on the Android Platform." *Advances in Computers [89]*. Vol. 89. Amsterdam: Elsevier, 2013. 163-192. Print.

[9] "MetricsReloaded." *JetBrains Plugin Repository*. IntelliJ IDEA. Web. 14 June 2015. <https://plugins.jetbrains.com/plugin/93?pr=idea>.

[10] "Number of Apps Available in Leading App Stores 2015 | Statistic." *Statista*. Web. 24 June 2015. <http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/>.

[11] Swanner, Nate. "Google Plans to Support Its Android Studio IDE Instead of Eclipse." *TNW Network All Stories RSS*. 26 June 2015. Web. 28 June 2015. <http://thenextweb.com/dd/2015/06/26/google-plans-to-support-its-android-studio-ide-instead-of-eclipse/>.

[12] Taba, Seyyed Ehsan Salamati, Iman Keivanloo, Ying Zou, Joanna Ng, and Tinny Ng. "An Exploratory Study on the Relation between User Interface Complexity and the Perceived Quality of Android Applications." Queen's University. Web. 12 June 2015. <http://post.queensu.ca/~zouy/files/icwe2014-ehsan.pdf>.

[13] Wasserman, Anthony. "Software Engineering Issues for Mobile Application Development." Http://repository.cmu.edu/. Carnegie Mellon University, 7 Nov. 2010. Web. 15 June 2015. <http://repository.cmu.edu/cgi/viewcontent.cgi?article=1040&context=silicon_valley>.

Bo Wang

[redacted]

Profile

I am a recent graduate of Towson University with a Master's degree in Computer Science. I am working full time as a Software Engineer at General Dynamics Information Technology at Towson, MD. My Master's thesis is to research and develop a plugin for Android Studio.

Education

Towson University 01/2013 - 12/2015
Master's degree in Computer Science

St. John's University 08/2008 - 05/2012
Bachelor's degree in Finance

Employment

General Dynamics Information Technology 05/2015 - Present
Software Engineer - Internship, Full Time

Skills

- Solid foundation in full life cycle of web development
- Extensive experience programming in Java, C#, C++, Ruby
- Knowledgeable in web development technology and database management systems
- Worked in large enterprise-level projects using Agile development methodology
- Excellent communication skills and effective in working independently and in teams