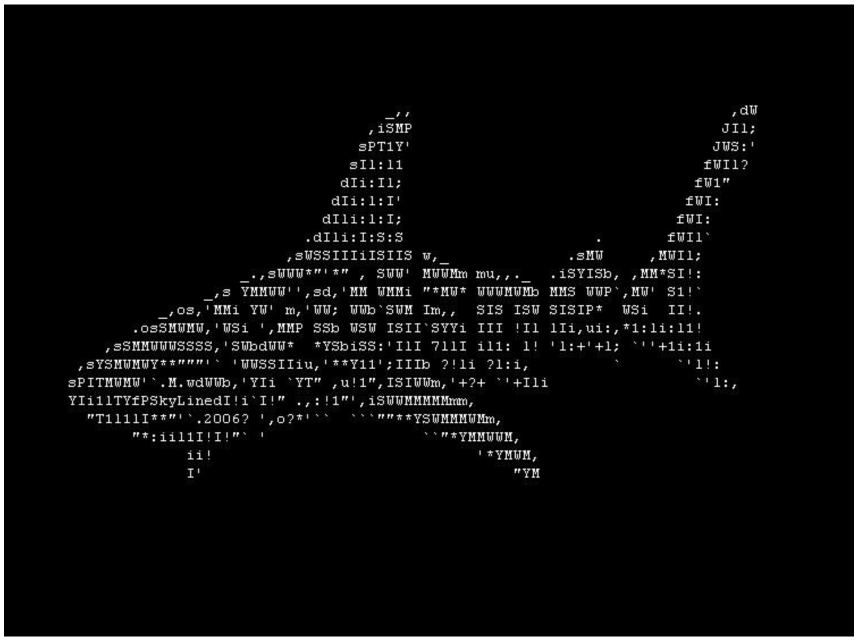
Product Requirements Document ver 4.0

ClassyShark

Google

Boris Farber
bfarber@google.com
10/10/2016



(http://skylined.deviantart.com/art/SkyLined-Shark-Ascii-Art-44110547)

Revisions:

- Version 1.0 (10/2015): Doc started
- Version 2.0 (11/2015): Added packages view
- Version 3.0 (3/2016): Added NDK
- Version 4.0 (10/2016): Updated with Java inspections

Introduction

Glossary[1]

1. APK - Android app self contained executable file (equivalent to exe file on Win platform). The users download and install this file from Google Play store.
2. Dex - Android executable binary format. Low level set of instructions that Android runtime knows how to execute.
3. Multidex - part of the Dex format, states that no more than 65K methods could be in one dex file (part of APK). If the app has more than 65K methods, the Android build will build a couple of dex files each having till 65K methods.
4. Java - standard Android development language. Java code converted to dex instructions via Android build chain
5. Class - Java executable file format. Low level set of instructions that Java runtime (JVM) knows how to execute. Not compatible with Dex format. However the Android toolchain know how to convert them to dex files
6. Jar - zipped folder of class files
7. NDK - Native Development kit for Android, to write pieces of app in C/C++
8. Dependencies - self contained jar files, written by 3-rd party and included in our Android app. The key reason is not to re-invent the wheel.
9. IDE - Development Environment, for Android it is Android Studio
10. Proguard - code obfuscator, applied as part of release tool chain to make code harder to reverse engineer.

Overview

ClassyShark is a tool for Android developers. When using ClassyShark, an Android developer can browse their application executable file, the APK and diagnose performance bottlenecks.

This spec does not discuss the algorithms used by the binary translation engine, which will be discussed elsewhere. It simply discusses what the user sees when they interact with ClassyShark.

Scenarios

Jane, an Android app developer

Once Jane finishes integrating her new feature, she will inspect the new release with ClassyShark. Jane will check if adding the new feature will increase her app method count and class count, will the new feature introduce other dependencies.

[1] The Glossary lists out the key terms to understand this PRD. Readers interested in more depth, please refer to Appendix A - Android Build

In addition, if the new APK will crash in release mode and not in debug, Jane will inspect the crashed classes, and fix the proguard configuration that likely caused the crash

Jill, an Android performance consultant

Once Jill gets an APK to inspect for performance issues, Jill will open an APK in ClassyShark and check for slow dependencies. The slow dependencies usually around JSON and XML parsing. For example Jill could recommend use ig-json parser instead of GSON and/or remove the slow security library and shift the work to the server side.

Joe, an Android support engineer

Joe is a support engineer in a cool company ABC that creates an SDK to help tracking analytics events. Sometimes the clients mis-configures the SDK and the data is not arriving. In this case the support ticket is open.

Joe obviously can't see his client's sources, but Joe can browse the misbehaving APK with ClassyShark and look for missing data in Android manifest (usually intents), thus help out his client.

Scope

The scope of ClassyShark is to show the Java classes structure of an APK, and to provide browsing inside APK.

ClassyShark will employ binary reading and parsing techniques, however decomposition is out of scope.

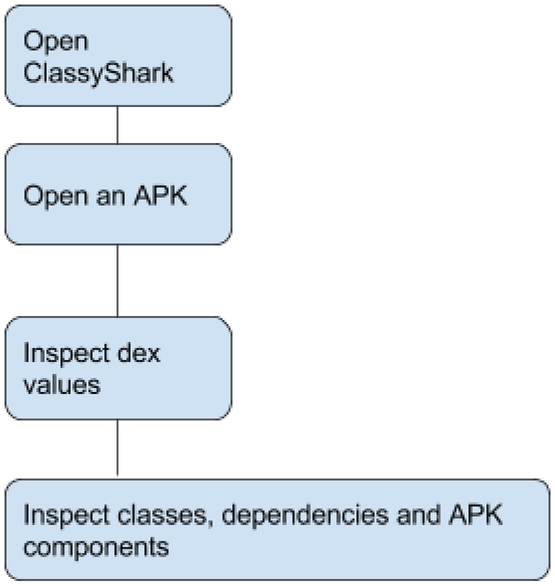
Market Assessment

ClassyShark is a tool for advanced Android developers, developing and maintaining non-trivial apps. Thus ClassyShark aims for top 1,000 - 1,500 Android apps. An average Android team has 10-15 developers, so we aim for 15,000 users.

Actually there are standalone Android executable browsers. The closest products are from .net world, namely .peek and .net reflector that offer the same functionality of browsing executables for .net platform. Note neither .peek nor .net reflector is open source.

User Flow

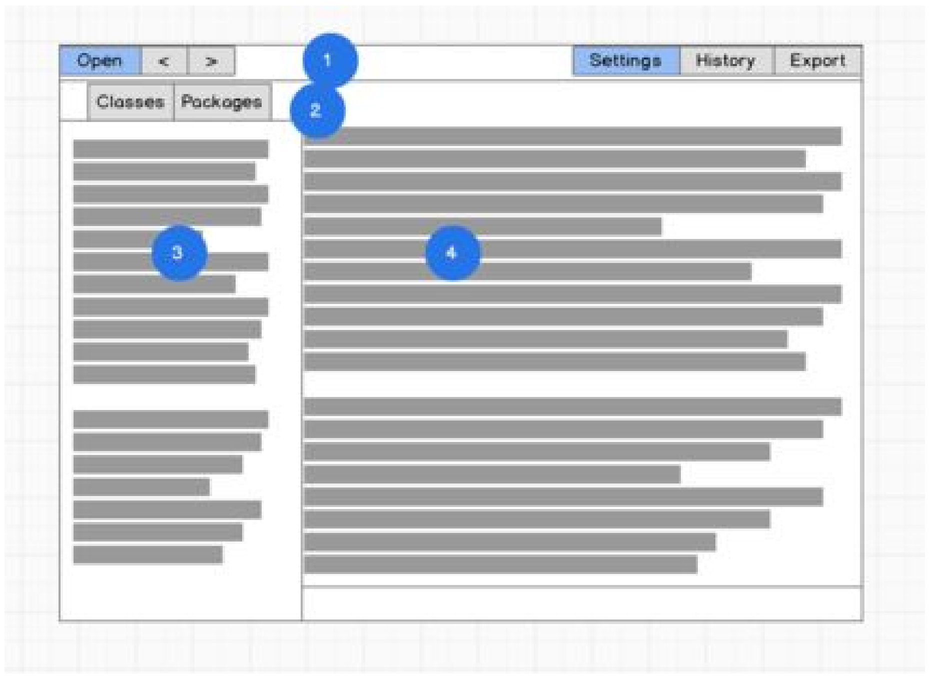
An Android developer will be able to use the UI client, within 5 minutes without any training. The structure of the UI client replicates an IDE, so the developer's knowledge will be transferred easily to the ClassyShark UI.



Thus the flow is easy. The developer open ClassyShark (either via double click or via command line).

As second step, the developer open their APK. Once the APK is open the developer inspects their APK for performance anti-patterns.

GUI Spec



1 - Toolbar

The toolbar will have the following components

1. Open APK button
2. < button - show the list of all classes in *4 - individual component view*
3. > - open the first class in *4 - individual component view*
4. Input text field - to look for specific class in APK, supports both camel and incremental search
5. Settings drop menu - TBD
6. History - a drop down list with recently viewed APK files
7. Export - TBD export the relevant data

2 - Classes & Packages tab

1. The Classes tab will display the classes breakdown in *4 - individual component view*
2. The Packages tab will display the dependencies breakdown in *4 - individual component view*, as a pie chart.

3 - APK components tree

The tree will display the following APK and Jar components. When the user clicks on a tree node the relevant is displayed in *4 - individual component view*

1. APK/Zip/Jar root with name
2. Classes.dex
3. Classes members of classes.dex
4. Classes members of zip/jar
5. Binary XMLs
6. Native libraries

4 - Individual Component view

Display the following data per tree node from *3 - APK Components Tab*

1. APK/Zip/Jar root with name
    a. Java dependencies list
    b. Native dependencies list
2. Classes.dex
    a. Dex header, including the method counts
    b. Classes with native methods
3. Classes members of classes.dex/zip/jar
    a. Imports list
    b. Class definition
    c. Fields list
    d. Constructors list
    e. Methods list
4. Binary XMLs

    a. XML readable version
5. Native libraries
    a. Native dependencies list
    b. Dynamic symbols list

API Spec

Provide an API and a command line usage for all info, accessible from UI. We will describe it somewhere else.

Technical Characteristics

- Platform - ClassyShark will be written in Java, thus will be cross platform
- Integration - ClassyShark will be released as a standalone Java executable (jar) file. So no integration will be necessary.
- Environment - the only environmental requirement ClassyShark will need, is having Java installed on the user's machine
- Support - as on open source project, we can't commit to any SLA support levels. We try to provide our best, depends on the situation.
- Workflow, Timelines And Milestones - ClassyShark will be hosted on Github, and we expect the usual Github distributed development flow.

Performance Metrics & Testing

As with any complex product, the performance is tricky and hard to define. We will have a testing set of 10 50 MBs APK.

Few usability characteristics

- ClassyShark shouldn't freeze when loading large APKs
- The search should be instant

Appendix A - Android Build

- Proguard - http://proguard.sourceforge.net/
- Android build - http://tools.android.com/tech-docs/new-build-system