Mining the Evolution of Software Component Usage

Yana Mileva, Andreas Zeller (adviser)

PhD Thesis, 2012

Ch.2 - Evolution of API Components

- **Problem:** When designing a piece of software, one frequently must choose between multiple external libraries and library versions that provide similar services.
  - Which library is the best one to use?
  - Is the latest version the best one to use?
  - Has it been widely adopted already?

- **Solution:** We mined hundreds of open-source projects and their external dependencies in order to observe the popularity of their APIs and to give recommendations of the kind:
  - “Projects are moving away from this API component. Consider a change.”

API Popularity

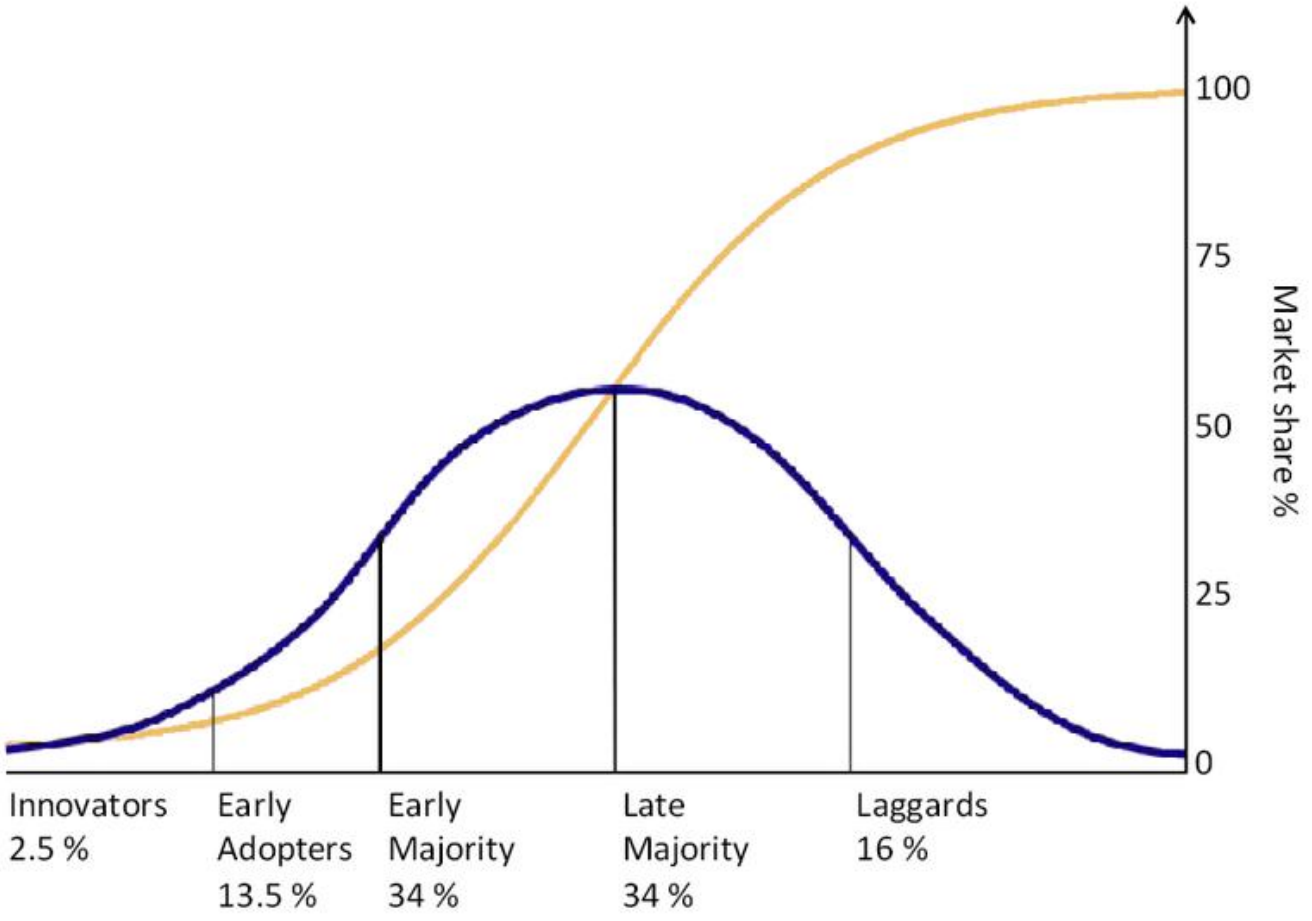
- We consider the usage popularity of an API to be indicative of its success. We consider the lack of popularity or the decrease in popularity to be indicative of lack of success.

- Popularity of an API is measured by the number of its users.

- Users can be classified in two groups:
  - Those who start using the newest library version immediately after it is out (early adopters)
  - Those who prefer to wait and see if it is safe to switch to (late followers)

- Every user has reasons for such behavior, such as the need for new functionality (early adopters) or security (late followers).

Diffusion of innovations



API Popularity

- Our analysis:
  - is being fed by the data collected from the early adopters
  - is being used to give recommendations to the late followers.

- Early adopters are rarely influenced by recommendations, as they know about the risks and benefits of being the first ones

- However, our approach is able to assist late followers in making an decision about which library and library version to use.

Study #1: Evolution of APIs

- We mined information from the history of 200 projects:
  - 50 from SourceForge
  - 150 from Apache
  - From Jan. 2008—Jan. 2009
- We analyzed the number of projects per month that used a specified import statement.
- By analyzing the usage of each import statement, we implicitly analyze the overall usage of the API itself and explicitly analyze the API elements.
- We detected the usage of 23401 unique import statements.

Trend Types

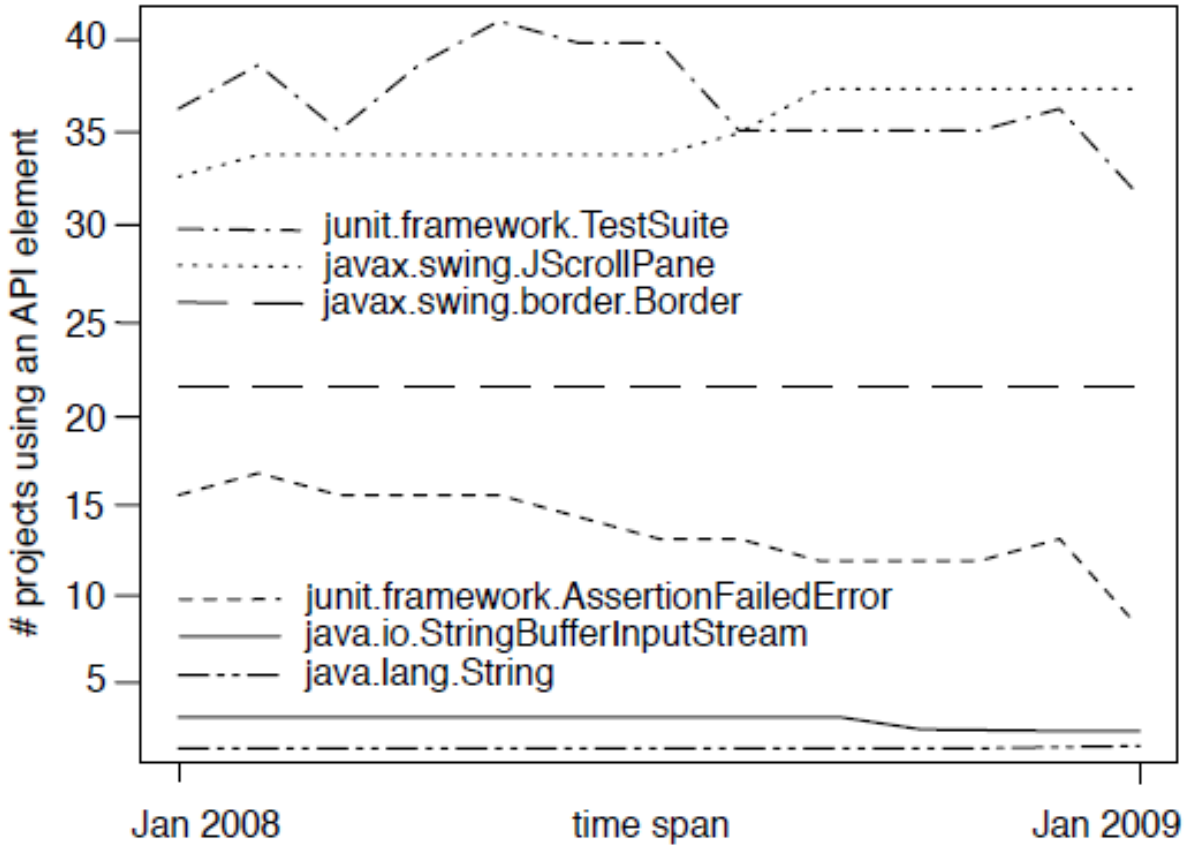


Figure 2.2: Examples of the four usage trend types.

Trend Types

- Four main types of trends:
  - Increasing trend (javax.swing.JScrollPane)
  - Decreasing trend (junit.framework.AssertionFailedError)
  - Stable (javax.swing.border.Border)
  - Undecided (e.g. junit.framework.TestSuite)

Study #2: Evolution of API Versions

- We mined information from the history of 250 Apache projects, in order to answer the following question:
  - Which library versions are the most popular ones?

- Challenge: there is no straightforward way to identify the API version used in the majority of the software projects.

- As our focus is analysis of Java projects we could focus on those projects, that are managed by Maven (project management tool)

- In Maven, library dependencies are stored explicitly in meta information files, which makes it possible to extract and analyze version usage

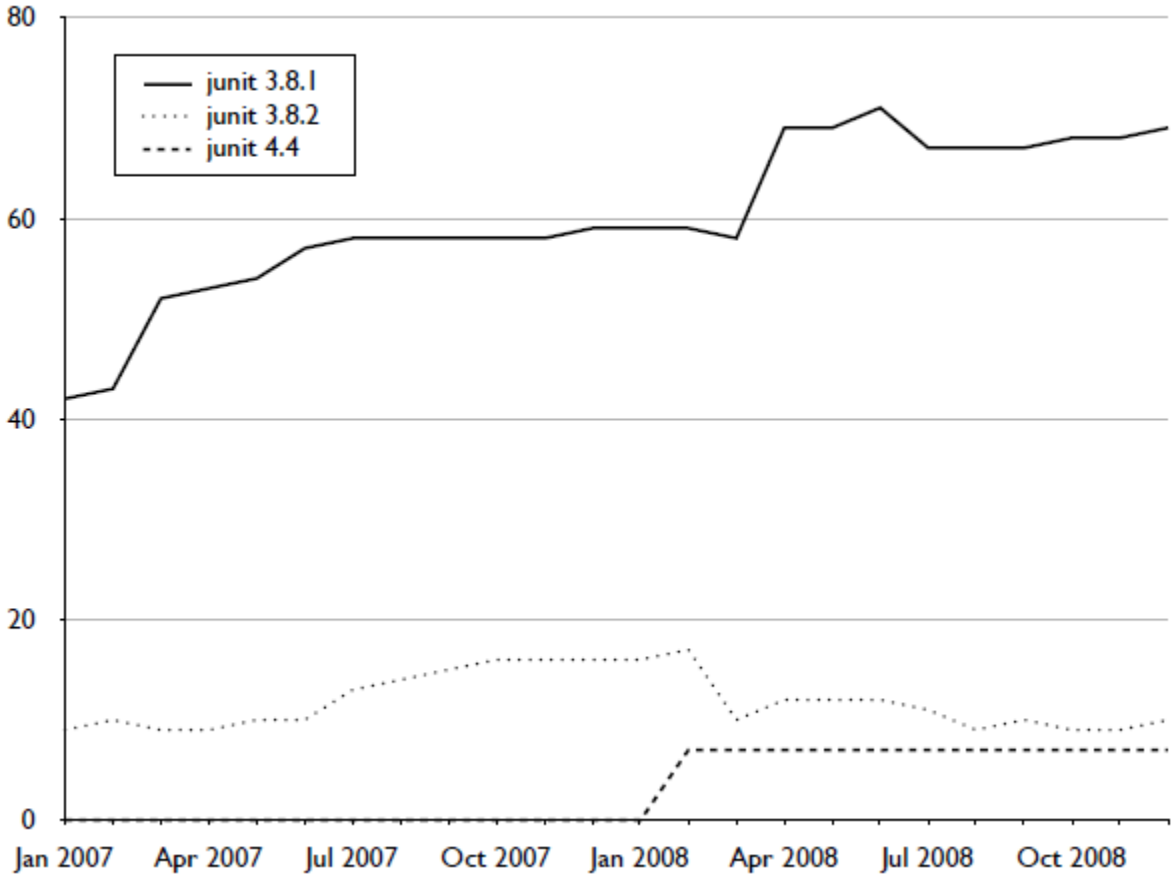
Maven

- Required libraries are described as dependencies in an XML file called pom.xml, a descriptive declaration of a POM

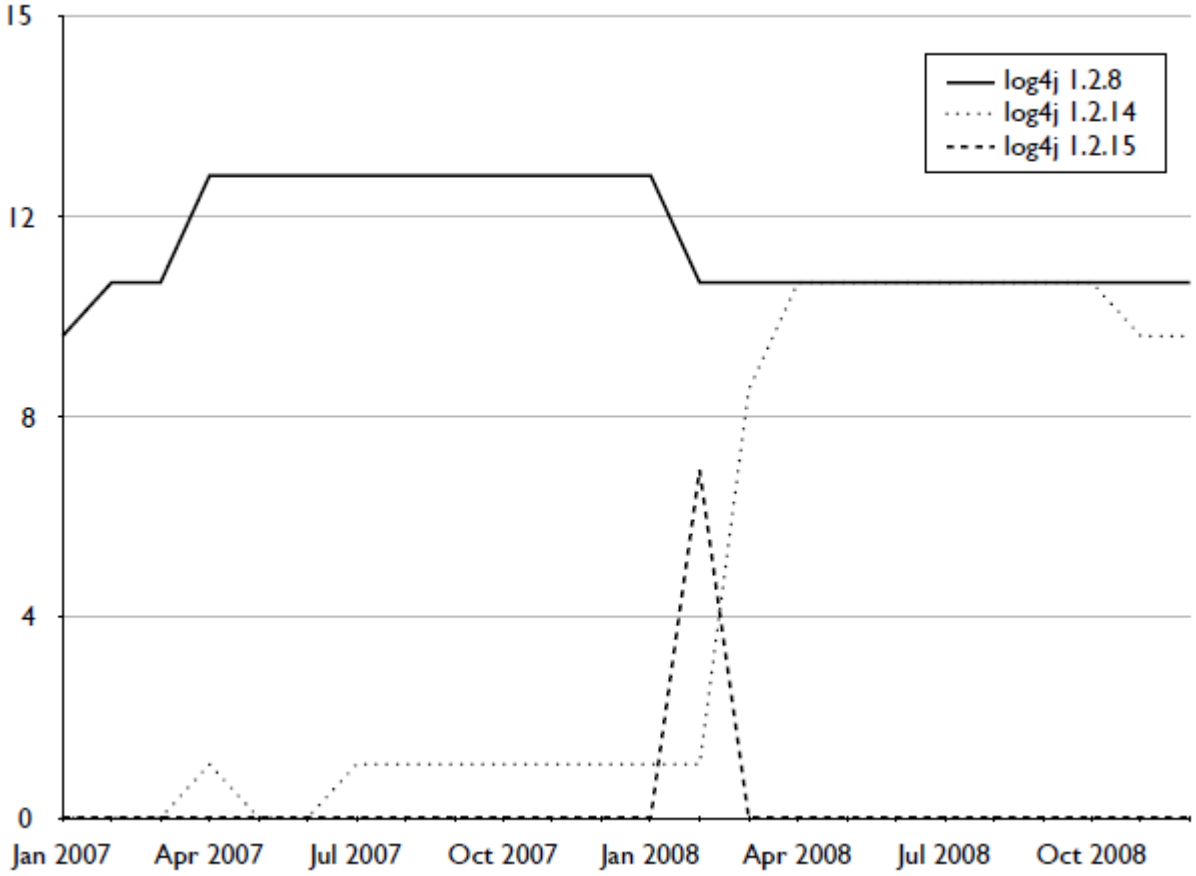
- Library usage information is stored in <dependency> elements

```
<project>
    <dependencies>
        <dependency>
            <groupId>servlet-api</groupId>
            <artifactId>javax.servlet</artifactId>
            <version>2.3</version>
        </dependency>
    </dependencies>
</project>
```

Junit – Usage Trends



Log4J – Usage Trends



Most Popular Versions

Table 2.2: Usage of library versions for January 2009

Library name and version	Times used
junit 3.8.1	60
junit 3.8.2	9
junit 4.4	7
log4j 1.2.8	10
log4j 1.2.14	9
log4j 1.2.15	0
servlet-api 2.3	4
servlet-api 2.5	1
derby 10.1	6
derby 10.2	1

Switching back to earlier versions

Table 2.3: Switching back to older library versions for the period January 2007–January 2009

Library	**# usages**	**# switched back**	%
junit 3.8.1	1501	0	0%
junit 3.8.2	293	1	<1%
junit 4.4	84	0	0%
log4j 1.2.8	269	3	2%
log4j 1.2.14	114	0	0%
log4j 1.2.15	**7**	**4**	**57%**
servlet-api 2.3	182	0	0%
servlet-api 2.5	10	1	10%
derby 10.1	147	0	0%
derby 10.2	31	0	0%

Aktari Tool

- In this chapter we presented approaches for analyzing the popularity trends of APIs and API versions.

- The presented techniques are combined in our general API analysis AKTARI platform.
  - http://www.st.cs.uni-saarland.de/softevo/aktari.php.

Understanding Reuse in the Android Market

Israel J. Mojica Ruiz, Meiyappan Nagappan, Bram Adams, Ahmed E. Hassan

ICPC 2012

Introduction

- This paper analyzes the adoption of code reuse for mobile app development.

- In particular, we analyze the mobile apps for reuse along the following two research questions:

- RQ1: What fraction of the classes reuse code by inheritance?
  - 27% of the classes inherit from a domain specific class.
  - 23% of the classes inherit from Android-specific classes.
  - Nine of the top ten base classes are from the Android API

Introduction

- RQ2: How often do mobile apps reuse classes from other apps?
  - 61% of the total number of classes in each category occurs in two or more mobile apps
  - Most of the classes reused are from third party developers like Apache and Google
  - 217 mobile apps have the exact same set of classes as another mobile app in the same category

Study Design – Subject Systems

- We analyzed apps from the Android Market
- We limited the apps in our case study to only the free ones.
- Since Android Market does not provide an interface for mass download of mobile apps, we had to manually download them.
- Hence, we restricted our study to just five categories.
  - Cards and Casino
  - Personalization
  - Photography
  - Social
  - Weather

Subject Systems

- Table I shows the total number of APKs that we were able to backup to our server using the Astro File Manager.
- The process of downloading and backing up the apps required approximately 100 hours

TABLE I
SET OF ANDROID APPLICATIONS UNDER ANALYSIS

Category	Number of applications under study
Cards and Casino	501
Personalization	1,115
Photography	1,034
Social	1,119
Weather	554

Methodology

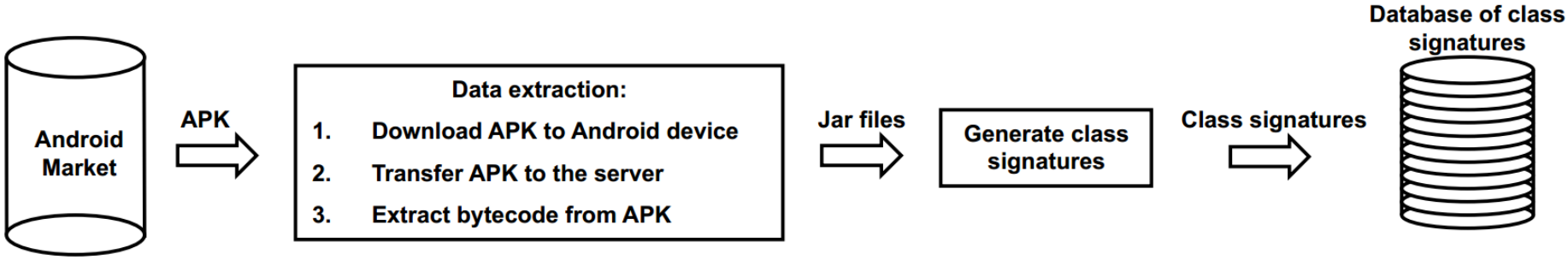


Fig. 1. Steps for the generation of class signatures from mobile apps in the Android Market.

```
package p.s;

public class ClassName extends j.l.E implements x.y.Z{

     public C(){
         //Class' constructor
     }
     syncronized static int a(java.lang.String s) throws
         pack.subPackage.K{
           /*[compiled byte code]*/
     }
}
```

sClass = public class p.s.ClassName extends E implements Z
sM1 = public C()
sM2 = default synchronized static int a(String) throws K
Bertillonage class signature = <sClass,<sM1,sM2>>

Fig. 2. Decompiled Java class, Class and Method lines, and Class Signature

Case Study – RQ #1

- **What fraction of the classes reuse code by inheritance?**
- We want to analyze the extent of inheritance present in mobile apps of the Android
- We want to determine:
  - What proportion of classes in each category inherit a base class?
  - What are the top base classes that are inherited?

RQ #1 - Results

TABLE III
INHERITANCE BY CATEGORY (PERCENTAGES CALCULATED WITH RESPECT TO THE TOTAL NUMBER OF CLASSES THAT INHERIT A BASE CLASS IN THE MOBILE APPS OF A SPECIFIC CATEGORY)

Category	Percentage of base classes from the Android API	Percentage of base classes from other domain specific classes
Cards and Casino	21%	29%
Personalization	29%	26%
Photography	29%	19%
Social	19%	31%
Weather	25%	25%

RQ #1 - Results

TABLE IV
TOP 10 BASE CLASSES (PERCENTAGES CALCULATED WITH RESPECT TO THE TOTAL NUMBER OF CLASSES THAT INHERIT A BASE CLASS IN THE MOBILE APPS ACROSS ALL FIVE CATEGORY)

Class name	Percentage of classes that inherit from the base class
android.app.Activity	9.1%
java.lang.Enum	3.4%
android.content.BroadcastReceiver	2.6%
android.widget.RelativeLayout	2.3%
java.lang.Exception	1.6%
android.widget.BaseAdapter	1.2%
java.lang.Thread	1.1%
android.os.AsyncTask	0.9%
android.os.LinearLayout	0.8%
org.apache.james.mime4j.field.address.parser.SimpleNode	0.8%

RQ #1 - Results

TABLE V
TOP 10 BASE CLASSES THAT ARE NOT PART OF THE ANDROID API (PERCENTAGES CALCULATED WITH RESPECT TO THE TOTAL NUMBER OF CLASSES THAT INHERIT A BASE CLASS IN THE MOBILE APPS ACROSS ALL FIVE CATEGORY)

Class name	Percentage of classes that inherit from the base class
org.apache.james.mime4j.field.address.parser.SimpleNode	0.77%
com.adwhirl.adapters.AdWhirlAdapter	0.65%
org.apache.http.entity.mime.content.AbstractContentBody	0.44%
org.apache.james.mime4j.field.AbstractField	0.43%
com.wsi.android.framework.wxdata.exceptions.WxFrameworkException	0.39%
org.apache.james.mime4j.field.ParseException	0.38%
org.jivesoftware.smack.packet.IQ	0.36%
gnu.expr.ModuleBody	0.35%
com.qq.taf.jce.JceStruct	0.34%
com.wsi.android.framework.wxdata.exceptions.WxFrameworkException	0.33%

RQ #1 - Discussion

- Prior research looked at the extent of reuse via inheritance in four open source Java projects (JHotDraw, Log4J, ArgoUML, and Azureus).
- The results from this research shows that the percentage of classes that inherit from base classes are between 29 and 36% (in ArgoUML and JHotDraw respectively).
- In comparison, the percentage of reuse via inheritance in each category of mobile apps under study is about 50%

Case Study – RQ #2

- **How often does a class from a mobile app gets reused in another app?**

- We want to determine:
  - What is the proportion of classes in each category that are reused at least once, i.e., occur in two or more apps?
  - What is the proportion of classes in each app that are reused in at least one of the remaining apps in the same category?

RQ #2 - PCSR

- Proportion of Class Signatures Reused (PCSR):

$$PCSR = 1 - \frac{Total\ Number\ of\ Unique\ Class\ Signatures}{Total\ Number\ of\ Class\ Signatures}$$

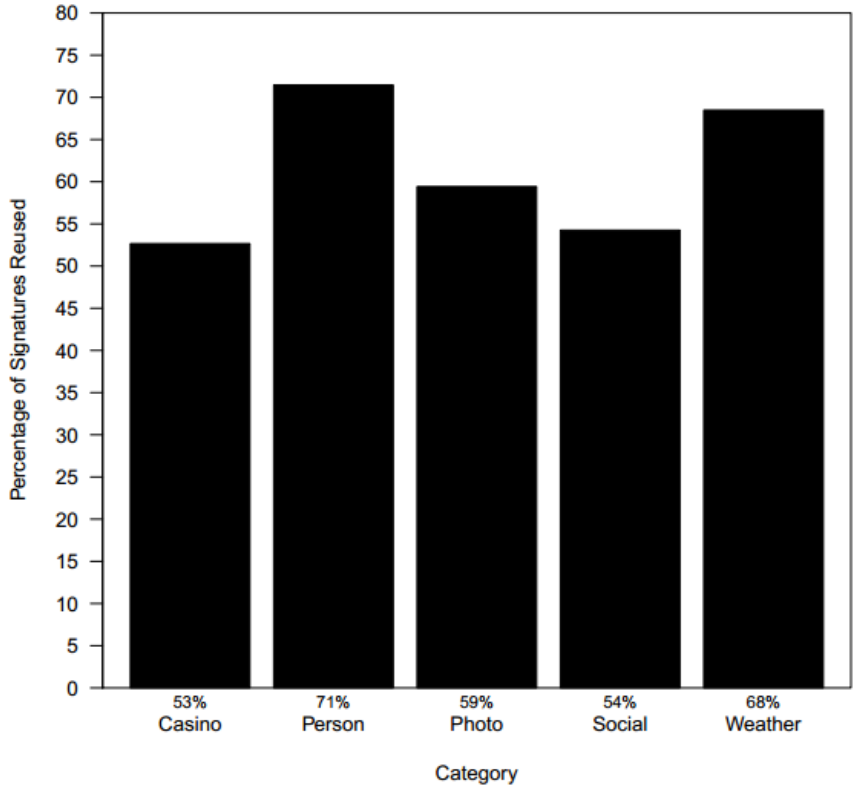


Fig. 3. Proportion of Class Signatures reused per category

Our results indicate that on average 61% of class signatures of a category are reused signatures.

This high percentage of reuse indicates that very few classes are unique to a mobile app.

RQ #2 – Global Reuse

- What percentage of class signatures in each mobile app occurs in other mobile apps of the same category?

- Global(A) is the proportion of class signatures found in a mobile app A that are also found in at least one other app in the same category as A.

- We define Global Reuse in a mobile app as:

$$global(A) = \frac{|s(A) \cap s(\bar{A})|}{|s(A)|}$$

Where: A = Current mobile app under consideration.
$\bar{A}$ = The apps other than A, in the same category as A.
s(A) = Set of class signatures in A.
s($\bar{A}$) = Set of class signatures in $\bar{A}$.

RQ #2 – Global Reuse

- When Global(A) = 1:
  - Every class signature in A occurs in at least one other mobile app in the same category (not necessarily all in the same mobile app).

- When Global(A) = 0:
  - No class signature in the mobile app A occurs in any other mobile app in the same category

Cumulative Global Reuse

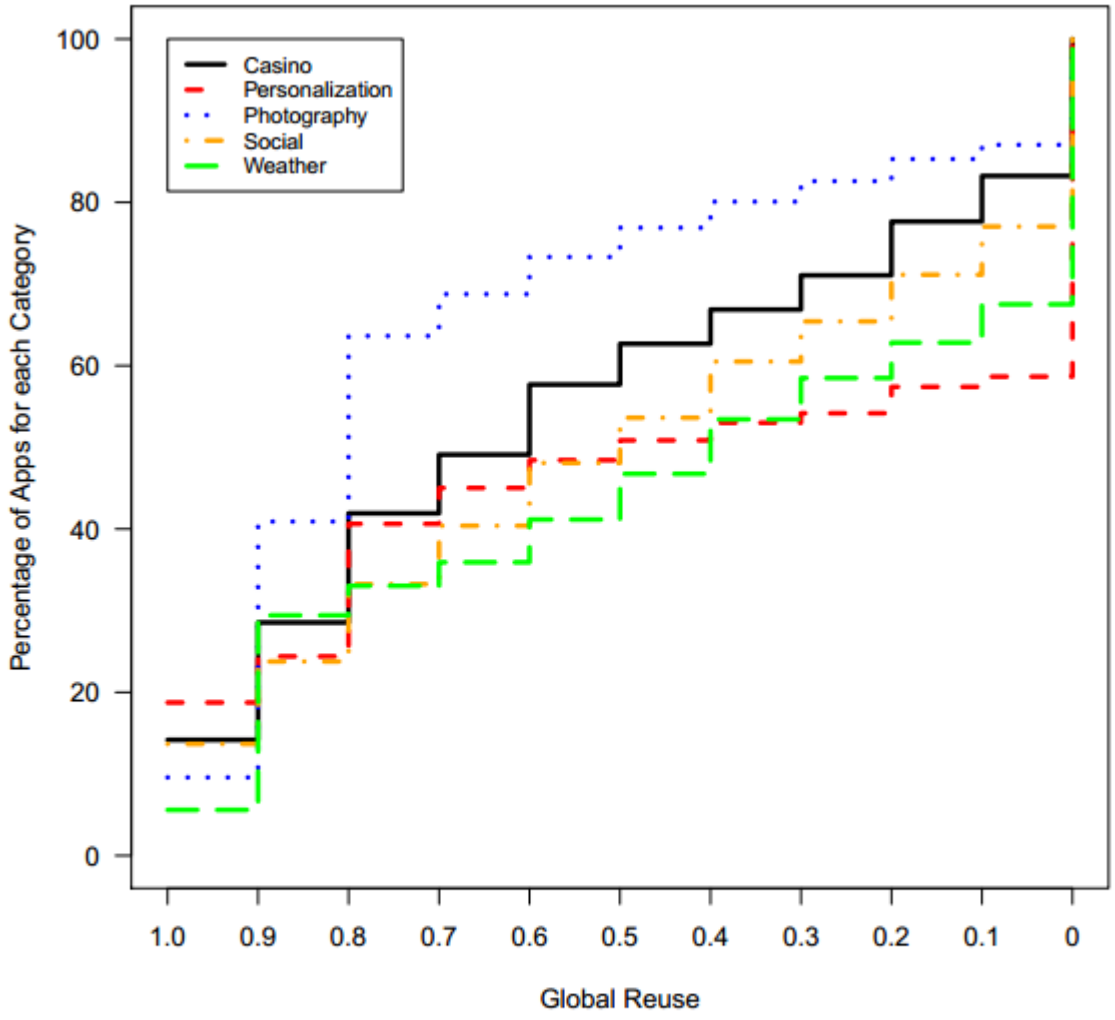


Fig. 4 shows that Personalization has the highest percentage of apps with Global Reuse = 1, that is every class signature in these mobile apps occurs at least in one other mobile app in the Personalization category

Fig. 4. Cumulative *Global Reuse* per app among the different apps in each category.

Apps with identical classes

- In the weather category, there were 11 mobile apps with the same set of class signatures that were all developed by the same developer.
- When we looked at the actual mobile app in the Android Market, we found that each app basically was the same weather app, but for different metropolitan cities in Europe.
- We think that the change could be the weather service that is called to get the data for the corresponding cities.

Top 10 Projects/Organizations

TABLE VI
TOP 10 PROJECTS/ORGANIZATIONS

Project / organization name	Number of unique classes in the project/organization	Total number of reused classes from the project/organization
Apache	4,370	86,489
Google	4,821	39,973
Wsi	311	21,205
Anddev	1,162	20,716
twitter4j	1,363	10,781
Gnu	700	10,220
Android	1,117	7,580
Admob	426	7,023
Jivesoftware	716	6,355
Millenialmedia	388	6,140

We can see that ten developers (in Table VI) account for 216,482 of the class signatures (almost 50% of the total class signatures across the five categories under study).
However, from column two we can identify that most of the unique classes come from the libraries in Google Code.

Top 10 Projects/Organizations

- The high number of Google packages reused are part of Google AdMob Ads API (com.google.ads).
- The purpose of this package is to add a banner on top of the apps.

TABLE VII
TOP FIVE CLASSES ACROSS THE 5 CATEGORIES OF MOBILE APPS

Class name	Number of apps used in
com.google.ads.AdSize	1,525
com.google.ads.AdRequest	1,280
com.google.ads.InstallReceiver	1,280
com.google.ads.InterstitialAd	8,29
com.google.ads.util.AdUtil	7,27

Conclusion

- We conclude that software reuse is prevalent (compared to regular open source software) in mobile apps of the Android Market.
- Developers reuse software through inheritance, libraries and frameworks. Most of these are from publicly available open source artifacts.
- Developers of these highly reused artifacts can use our results to make them more reliable and efficient.
- The results also help the developers of mobile apps, to understand the risk they run when the library/framework they build on, changes.