A Preliminary Study of Android Refactorings

R·I·T
B. Thomas Golisano College of Computing and Information Sciences

Anthony Peruma
Rochester Institute of Technology
axp6201@rit.edu

Introduction

With over 3 million apps on the Google Play Store, developers need to ensure that their apps remain competitive

Software maintenance is critical and is also a significant pain-point for developers

Source code rework (i.e., refactoring) impacts developer productivity and can introduce defects into the app

Goal: understand the refactoring habits of app developers and how they differ from traditional system developers. We study:

- Common refactoring operations
- Reasons for refactoring source code

Methodology

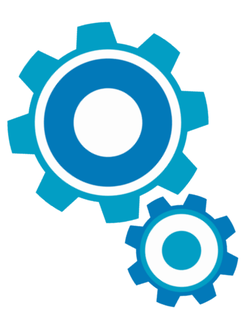
1,028 open source Android apps

F-Droid

GitHub

Clone projects & extract commit data

Source code refactoring detection via RefactoringMiner



Topic modeling using latent Dirichlet allocation (LDA)

Dataset: https://sites.google.com/g.rit.edu/refactoring

RQ 1: App Refactorings

Detected refactoring operations: 156,073
Average refactoring's per app: 47.79

Renames are one of the most frequent refactorings applied by developers

- Rename refactorings – 41.93%
- Non-rename refactorings – 58.07%

Elements of UI based datatypes tend to undergo more refactorings

- TextView, View, Intent, Context

Distribution of refactorings mostly differ between Android apps and traditional systems
Demonstrates the need to study Android apps separately from traditional systems

RQ 2: Motivation

LDA based analysis of refactoring commit messages to indicate why app developers refactor their code

Improve code comprehension

Terms *: *renam, code, method*

Example: "*Rename member variables as per README's coding style*"

Resolve defects

Terms *: *fix, bug, issu, improv, chang*

Example: "*Bug fixes to pseudo cursor requery()*"

Change in functionality

Terms *: *updat, add, remov, refactor*

Example: "*Update the progress dialog text when an open port…*"

* Stemmed terms