Identifying Features of Android Apps from Execution Traces

Qi Xin, Farnaz Behrang, Mattia Fazzini, and Alessandro Orso
Understanding a Program & its Features

- Refactoring
- Debloating
- Debugging
- Functionality Modification
- Testing
- Documentation
Understanding a Program & its Features

Identifying Features of a program by Analyzing its Executions
Program Understanding is HARD

Figure from Understanding Execution Traces Using Massive Sequence and Circular Bundle Views by Cornelissen et al.
Program Understanding is HARD

~50% maintenance effort spent on program understanding alone*

*Program understanding: Challenge for the 1990s by Corbi

Figure from Understanding Execution Traces Using Massive Sequence and Circular Bundle Views by Cornelissen et al.
Our Approach

• Identifies features by analyzing execution trace
• Targets Mobile (Android) apps
• In our context, a feature is a sequence of user events that exercise some functionality of the app
Our Approach

• Identifies features by analyzing execution trace
• Targets Mobile (Android) apps
• In our context, a feature is *a sequence of user events* that exercise some functionality of the app
Login Feature of WordPress

1. Log in to WordPress.com using an email address to manage all your WordPress sites.
   - Email address
   - Alternatively:
     - Log in by entering your site address.

2. Enter your email address.
   - Email address: xinq07@gmail.com

3. Proceed to the next step.
   - NEXT
Login Feature of WordPress

4. If you lose your password, WordPress will email you a magic link that'll log you in instantly, no password needed. Hunt and peck no more!

5. Enter your WordPress.com account name.

6. Enter your password.
Login Feature of WordPress
High-level Approach
High-level Approach
High-level Approach

Feature0  Feature1  Feature2  Feature3

S0  S1  S2  S3  S4  S5  S6  S7  S8  S9  S10  S11

Trace
FeatureFinder

Step 1: Instrumentation
User → App → Instrumented App

Step 2: Execution
Instrumented App → Trace File

Step 3: Splitting
Trace File → Segments

Step 4: Clustering
Segments → Clusters

Step 5: Labeling
Clusters → Labeled Clusters (Features)
FeatureFinder

Instrumentation

App → Instrumented App

Step 1

Capture execution information
- Stacks of Method Calls
- Activities & Fragments
- User Events
  - Touch event & widgets
  - Keyboard event & labels
FeatureFinder

Step 2

User executes the app to exercise its features
FeatureFinder

Splitting

Trace File → \[\text{Scissors}\] → Segments

Split the trace into segments by user events

Step 3
FeatureFinder

Step 4

Group “related” segments
- Compare execution info
- Use a classifier to decide “relatedness”
Clustering algorithm
Clustering algorithm
Clustering algorithm
Clustering algorithm

A numeric vector encoded as the comparison b/w S0 and S1
Clustering algorithm

Vector_{S0,S1} → A trained classifier
Clustering algorithm

Vector_S0_S1 → Merge
Clustering algorithm

S0  S1  S2  S3  S4
Clustering algorithm

Vector_S0_S1
Clustering algorithm

S0  S1  S2  S3  S4

Vector_S0_S1  →  Merge
Clustering algorithm

S0  S1  S2  S3  S4
Clustering algorithm

Vector_S0_S1_S2  ➔  Don’t Merge
Clustering algorithm
Clustering algorithm

S0  S1  S2  S3  S4

Vector_S3_S4

Merge
Clustering algorithm

S0  S1  S2  S3  S4
FeatureFinder

Labeling

Clusters → Labeled Clusters (Features)

Step 5

Label each cluster
- Use activity & fragment names
- Rank names by TF-IDF values
- Select the top-10
Case Study

- Conducted a study using 5 apps
- Exercised different app features and generated traces
- Used **4 apps** for classifier training
- Evaluated FeatureFinder on the other app **K-9 Mail**

Apps used for classifier training:
- WordPress
- DailyMoney
- PasswordMaker
- Music Player
- K-9 Mail

Two Trace for each App
Classifier Training

• Used FeatureFinder to split trace into segments
• Manually Identified clusters
• Generated 490 segments pairs labeled as “Merge” & “Don’t merge”
• Trained 10 classifiers
• Used the best: \textit{k-NN (k=10)}
Features for Trace 0

FeatureFinder

ft01
ft02
ft03

ft04
ft05
ft06

fh01
fh02
fh04

fh03
fh05
fh06

Human
Features for

FolderSettings
FolderList

MessageViewFragment
MessageList

MessageCompose
MessageList

MessageList
MessageListFragment

Email Checking

FolderList

Managing Folders

fh01
fh02
fh04

Email Checking

fh04

Email Composing
Evaluation Results

• Manually identified 11 feature clusters (ground truth)
• FeatureFinder generated 9 clusters
• Identified 6 of the 11 (55%) features
• Labels generated are in close meaning to the human labels
Conclusion & Future work

• FeatureFinder identifies *features* from app’s *execution traces*
• Case study results, albeit preliminary, are *promising*
• As future work
  • Perform a user study
  • Extend FeatureFinder to identify features hierarchically
  • Define a visualization for showing the features
Understanding a program & its features

High-level Approach

Evaluation Results

• Manually identified 11 feature clusters (ground truth)
• FeatureFinder generated 9 clusters
• Identified 6 of the 11 (55%) features
• Labels generated are in close meaning to the human labels