Towards Mutation Analysis of Android Apps

Lin Deng, Nariman Mirzaei, Paul Ammann, and Jeff Offutt

George Mason University

www.cs.gmu.edu/~offutt/

TOCSYC
TESTING OF CRITICAL SYSTEM CHARACTERISTICS

Background: Mobile Apps

Mobile App

A software program that runs on a mobile device

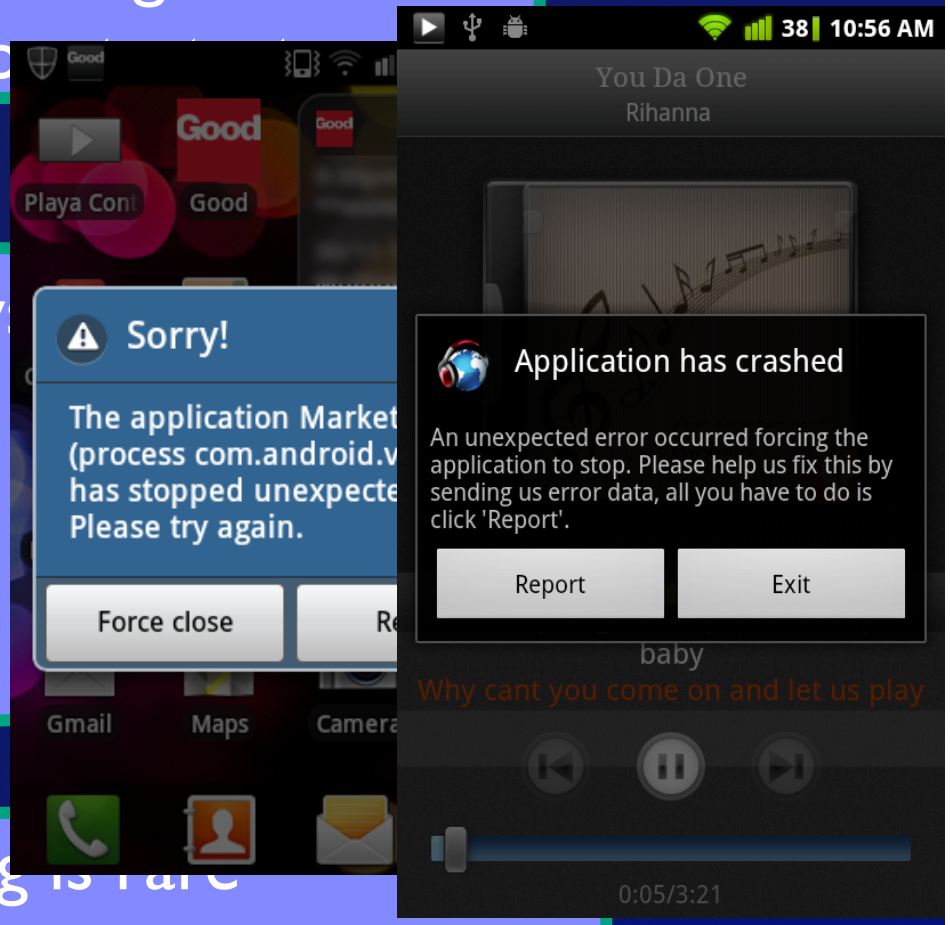
Android OS has 83% of the mobile market

Over a million apps on Google Play
Thousands added every day

Quality Problems

Little or no testing
Little knowledge for ho

Many apps have significant flaw
- Lack of robustness
- Runtime crashes
- Incorrect behavior
- Security vulnerabilities



Quality testing is rare
Ad-hoc and random testing common
New techniques have not reached practice

Motivation

Provide more sophisticated testing than current practice

Provide an evaluation criterion for other test selection strategies

Filter redundant pre-existing tests

Unique Programmatic Aspects

- Android apps are event driven
- Android programming components
  - *Activity* : A screen presented to users
  - *Service* : Performs long running background tasks (music)
  - *Content Provider* : Manages structured data (contacts)
  - *Broadcast Receiver* : Responds to system wide announcement messages (screen is off, battery is low)
  - *Intents :* Events that activities, services, and broadcast receivers used to communicate
- Unique execution engine
  - Novel JVM: Dalvik (4.4 and earlier), ART (5.0)
  - XML files define screen layouts and configuration
  - Testing is done on emulators

Research Objective

Improve our ability to deliver quality Android apps through stronger testing

Strategy

Apply existing technique (mutation testing) to a new type of software (mobile apps)

Plan

Design mutation operators based on the unique aspects of Android programming

Generate high quality tests by killing mutants

Preliminary Design Work

- 19 traditional (method level) muJava operators
- Eight novel Android mutation operators
  1. Intent Payload Replacement (IPR)
  2. Intent Target Replacement (ITR)
  3. OnClick Event Replacement (ECR)
  4. OnTouch Event Replacement (ETR)
  5. Lifecycle Method Deletion (MDL)
  6. XML Button Widget Deletion (BWD)
  7. XML EditText Widget Deletion (TWD)
  8. XML Activity Permission Deletion (APD)

Intent Mutation Operators

- *Intent Payload Replacement* (IPR)
  - Mutates the parameter to a default value

Original Type	Default Value
int, short, long, float, double, char	0
String	“”
Array	null
boolean	true / false

```
Intent intent = new Intent (this, DisplayMessageActivity.class);
intent.putExtra (EXTRA MESSAGE, “ ”);
startActivity (intent);
```

- *Intent Target Replacement* (ITP)
  - Replaces the target of each *Intent* with other classes

```
Intent intent = new Intent (ActivityA.this, ActivityC.class );
```

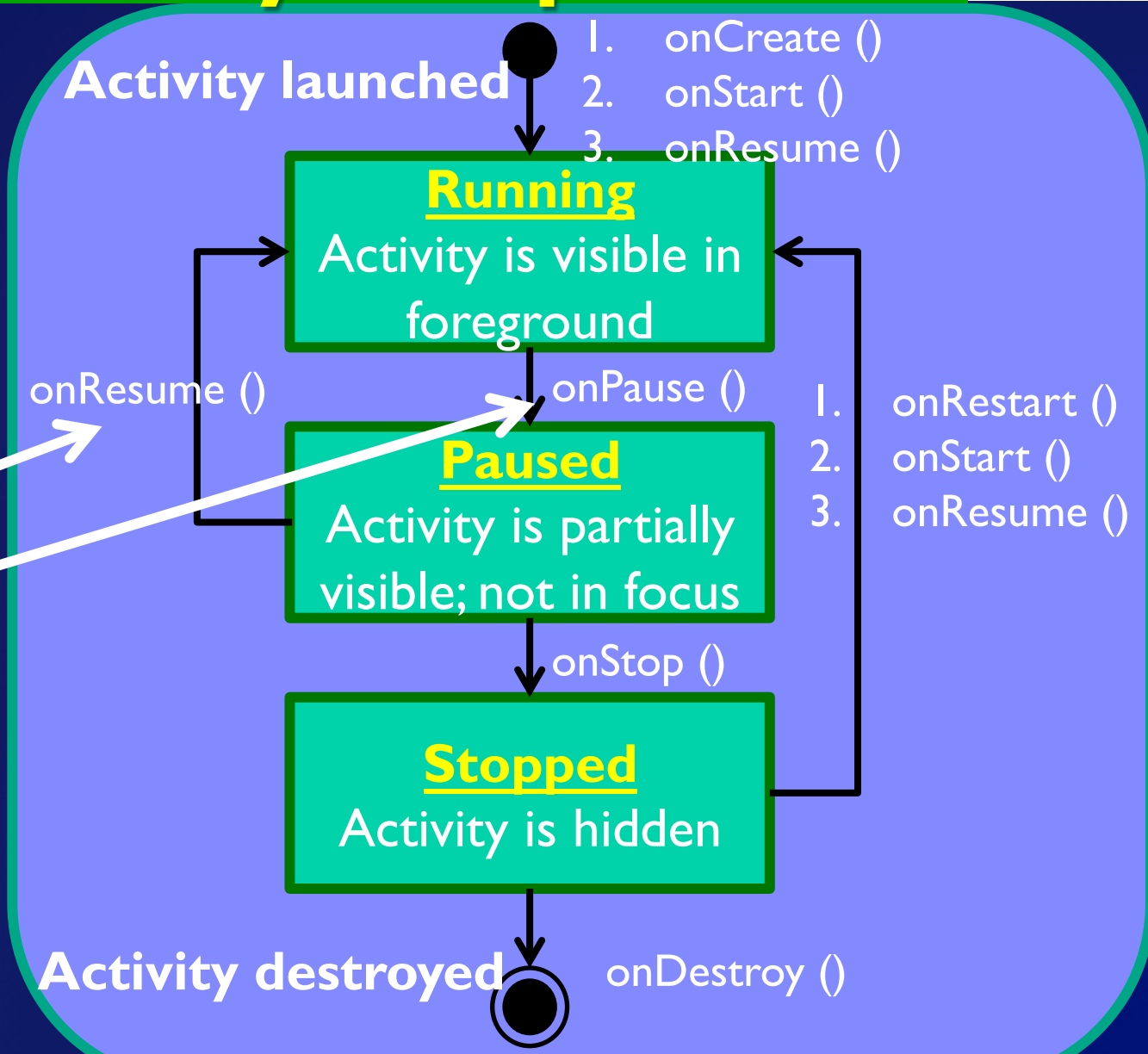
Event Handler Operators

- *OnClick Event Replacement* (ECR)
  - Replaces event handlers with other compatible handler

```
mPrepUp.setOnClickListener (new OnClickListener() {
    public void onClick (View v) {
        decrementPrepTime (); }
});
mPrepDown.setOnClickListener (new OnClickListener() {
   public void onClick (View v) {
      decrementPrepTime (); }
});
```

- *OnTouch Event Replacement* (ETR)
  - Replaces OnTouch events, similar to ECR

Activity Lifecycle Operator



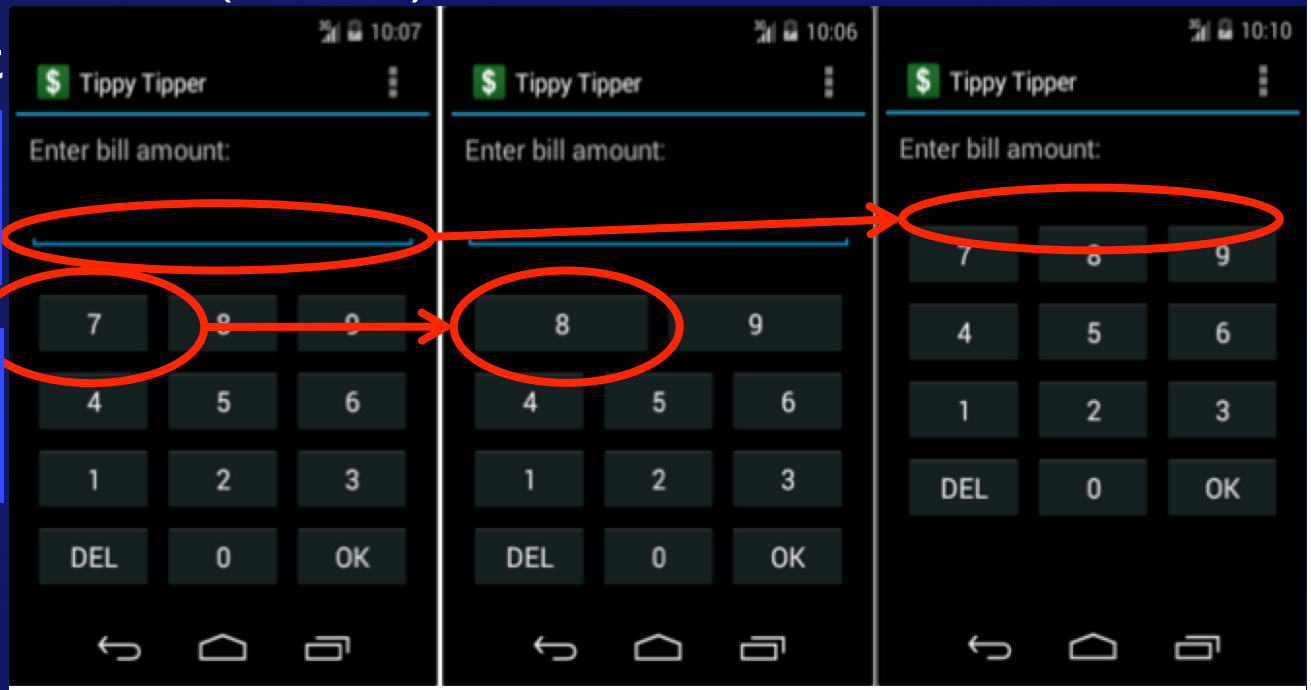
Lifecycle Method Deletion (MDL) deletes programmers versions

XML Mutation Operators

- *Button Widget Deletion* (BWD)
  - Deletes buttons one at a time
- *EditText Widget Deletion* (TWD)
  - Deletes EditText

TWD mutant

BWD mutant



Original | BWD Mutant | TWD Mutant

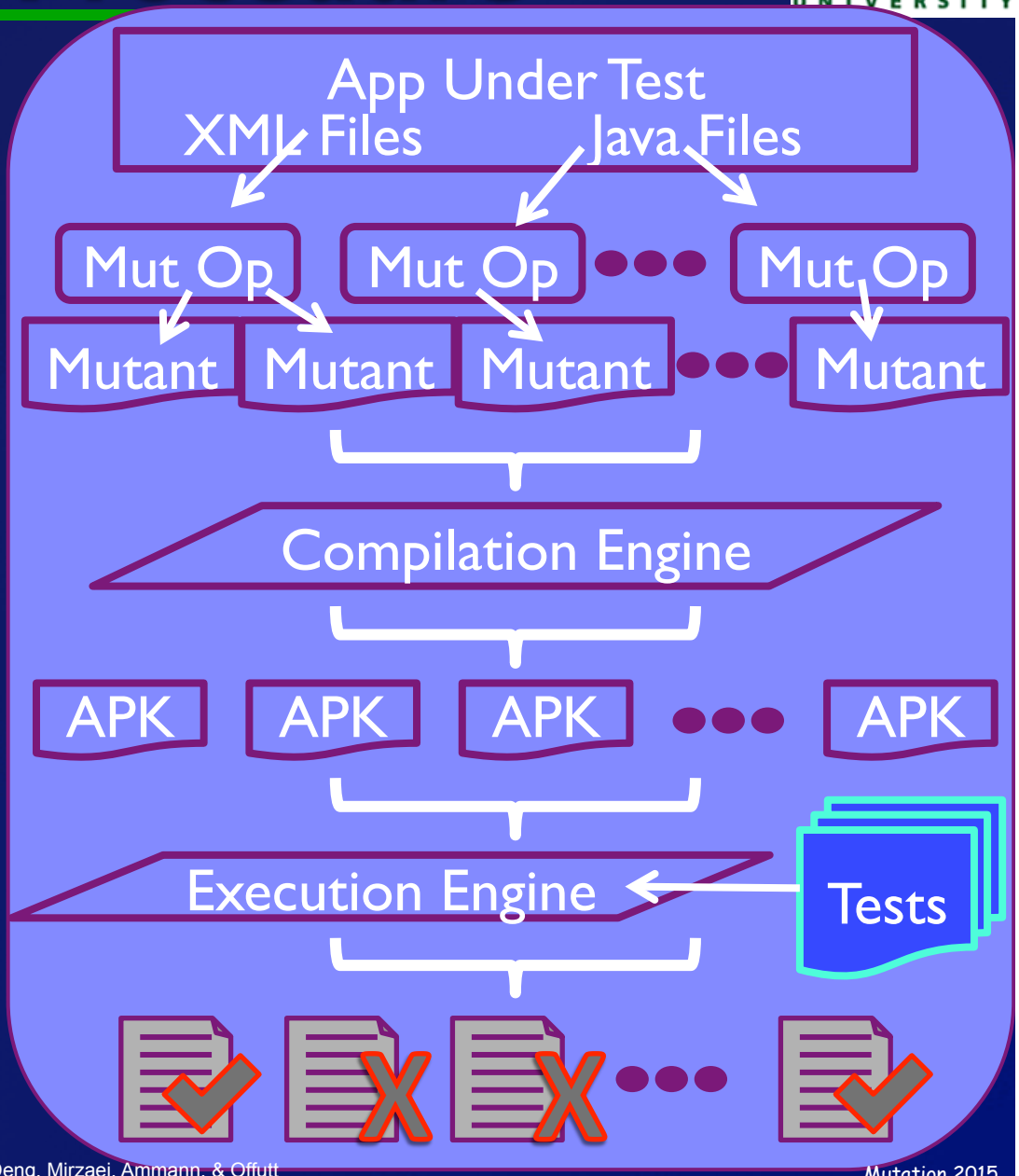
- *Activity Permission Deletion* (APD)
  - Deletes permissions from *AndroidManifest.xml*

Mutation Procedure

Mutate code and configuration files

Android intermediate code

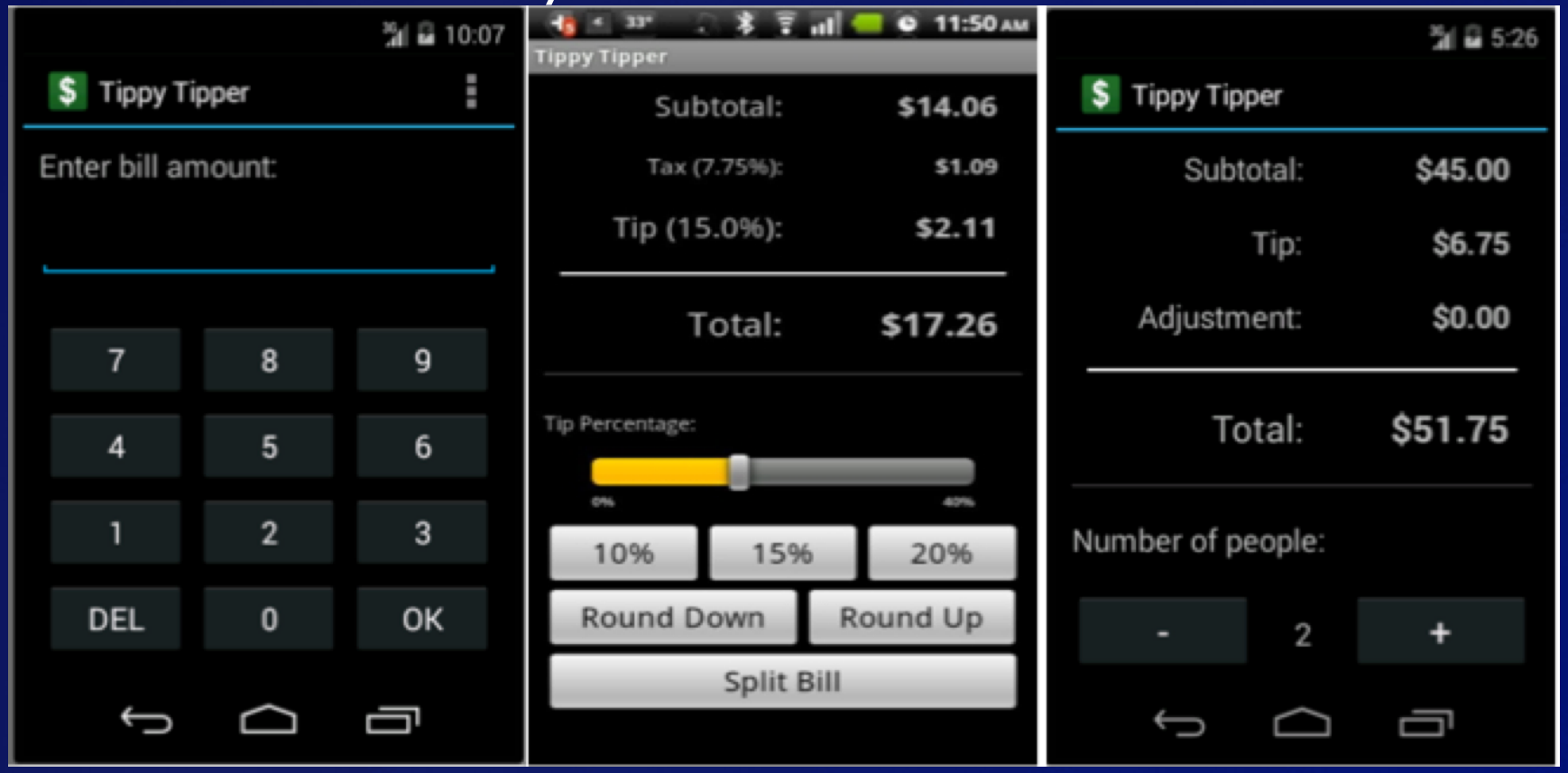
Emulator



Preliminary Study

TippyTipper : Computes amounts to tip waiters, splits bills

- Five activities, one service, 12 classes, 196 methods, 3575 blocks
- Tested the main activity TippyTipper.java : 103 LOC
- Mutated the XML layout main.xml : 93 text lines



Preliminary Study

- Tests generated by EvoDroid (*Mahmood et al., 2014*)
  - Uses an evolutionary algorithm
  - Generated 744 tests
  - Added test oracles by hand
  - 10 tests at the last generation selected (85% statement coverage)
  - Added one test by hand to achieve 100% statement coverage

- Mutation analysis tool built by extending muJava
  - Generate and compile APK mutants
  - Install APK files to an emulator
  - Execute tests and compute results

Results

- 85 Android mutants, 105 traditional Java mutants
- 85% statement coverage tests

Android Operators			
Operator	Mutants	Equivalent	Killed
ITR	5	0	5
ECR	66	0	45
MDL	1	0	1
BWD	12	0	6
TWD	1	0	0
Total	**85**	**0**	**57**

67.06%

No mutants for IPR, ETR, APD

Traditional muJava Operators			
Operator	Mutants	Equivalent	Killed
AOIS	8	4	0
AOIU	20	0	17
AORB	8	0	0
CDL	2	0	0
LOI	18	0	17
ODL	4	0	0
SDL	43	0	21
VDL	2	0	0
Total	**105**	**4**	**55**

54.46%

Results

100% statement coverage tests

Android Operators			
Operator	Mutants	Equivalent	Killed
ITR	5	0	5
ECR	66	0	66
MDL	1	0	1
BWD	12	0	12
TWD	1	0	0
Total	**85**	**0**	**84**

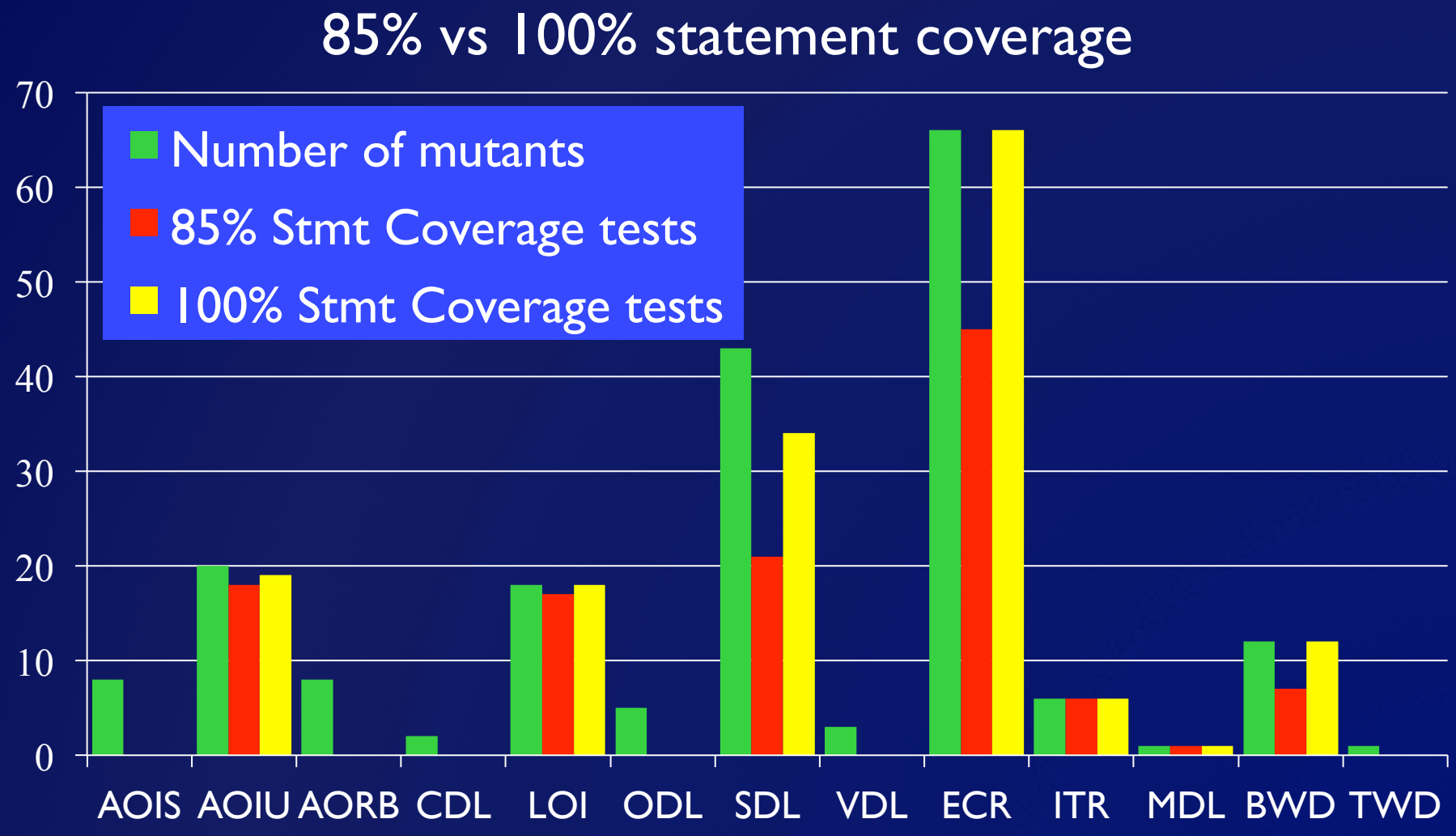
98.82%

Traditional muJava Operators			
Operator	Mutants	Equivalent	Killed
AOIS	8	4	0
AOIU	20	0	18
AORB	8	0	0
CDL	2	0	0
LOI	18	0	18
ODL	4	0	0
SDL	43	0	35
VDL	2	0	0
Total	**105**	**4**	**71**

70.30%

Combined mutation score: 83.33%

Results



Future Work

- Construct a comprehensive fault model based on existing apps with bug reports, leading to stronger mutation operators (in progress)
- Define mutation operators based on other Android aspects, e.g. context-aware
- More precise mutation system
  - Better algorithms
  - Fewer stillborn & crashing mutants
  - Stronger mutation operators
- More experimentation with more apps
  - Fault studies
- Speed up execution

Summary

Defined eight novel mutation operators specific to Android apps

Evaluated these mutation operators on an example Android app

Identified future research areas for mutation analysis of Android apps

Contacts & Questions

Lin Deng
ledng2@gmu.edu
cs.gmu.edu/~ldeng2/

Narimen Mirzaei
nmirzaei@gmu.edu

Paul Ammann
pammann@gmu.edu
cs.gmu.edu/~pammann/

Jeff Offutt
offutt@gmu.edu
cs.gmu.edu/~offutt/