SOFT620020.02
Advanced Software Engineering

Bihuan Chen, Pre-Tenure Assoc. Prof.

bhchen@fudan.edu.cn

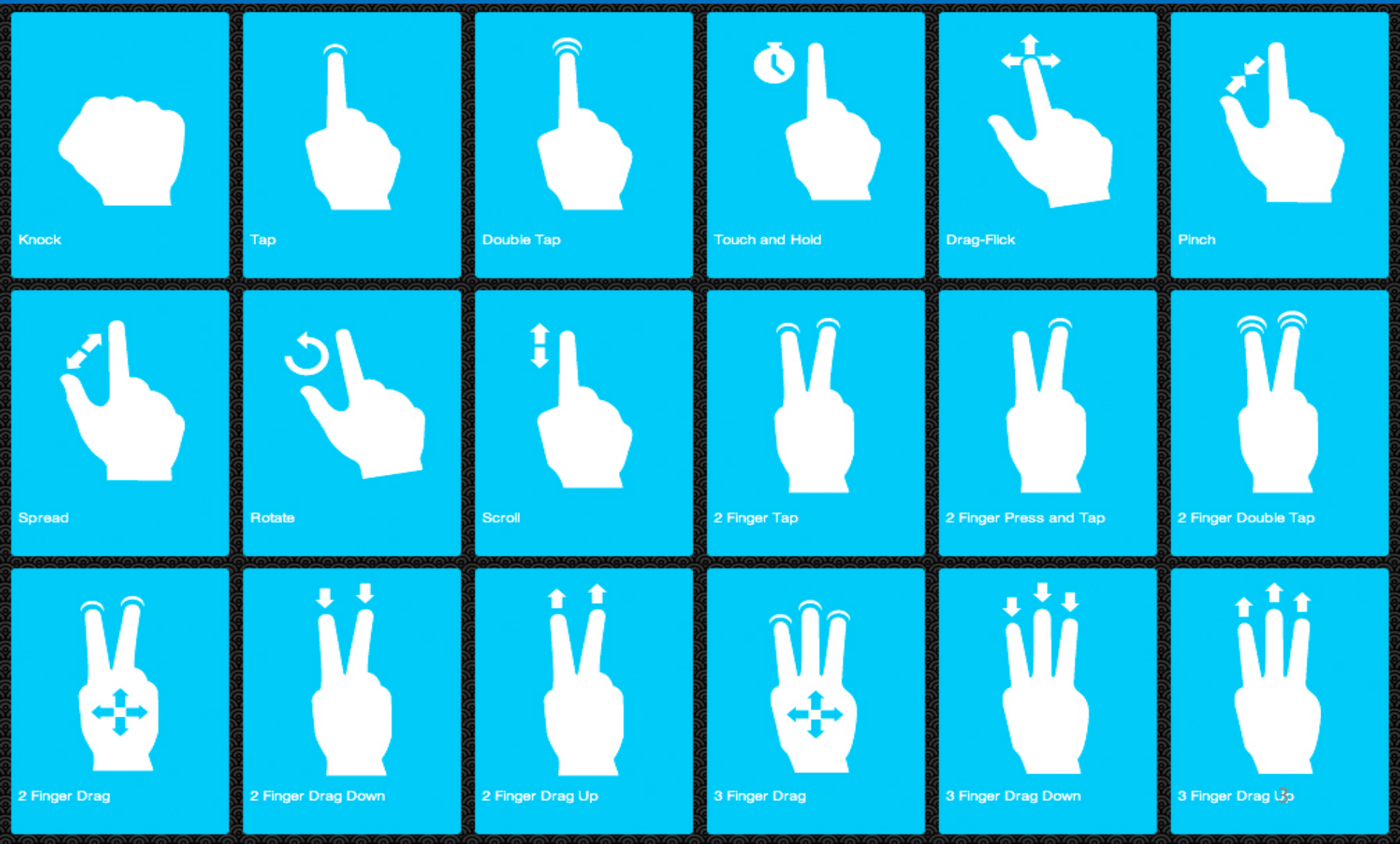
https://chenbihuan.github.io

Course Outline

Date	Topic
Sep. 10	Introduction
Sep. 17	Testing Overview
Sep. 24	Holiday
Oct. 01	Holiday
Oct. 08	Guided Random Testing
Oct. 15	Search-Based Testing
Oct. 22	Performance Analysis
Oct. 29	Security Testing

Date	Topic
Nov. 05	Compiler Testing
Nov. 12	**Mobile Testing**
Nov. 19	Bug Localization
Nov. 26	Presentation 1
Dec. 03	Delta Debugging
Dec. 10	Automatic Repair
Dec. 17	Symbolic Execution
Dec. 24	Presentation 2

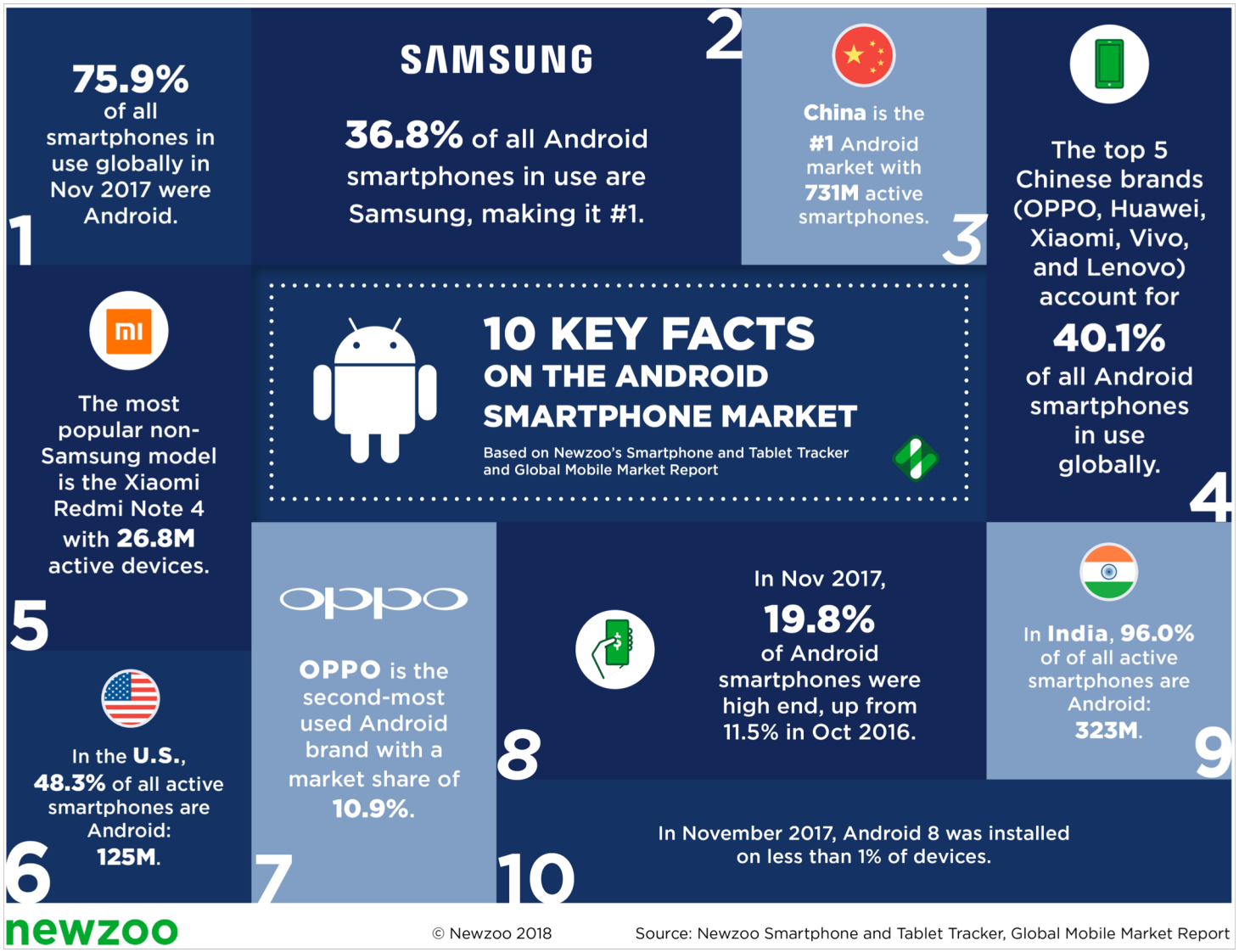
Discussion – Challenges in Mobile Testing?



The Growth of Smartphones and Tablets

- 1 million new Android devices activated every day
- 750 million total (March 2013)
- **2.3 billion total (Nov. 2017)**

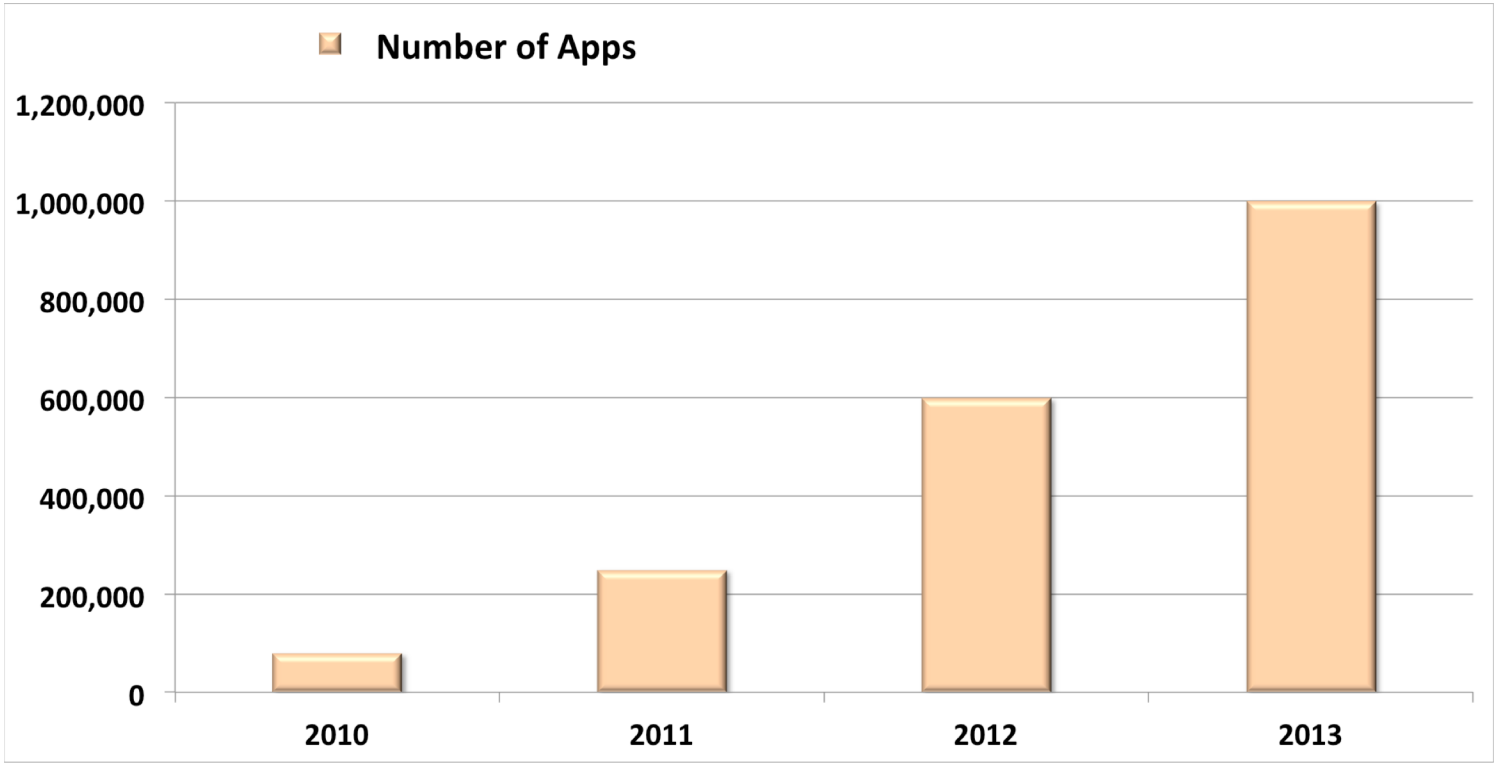
The Growth of Smartphones and Tablets



Source: Newzoo Smartphone and Tablet Tracker, Global Mobile Market Report

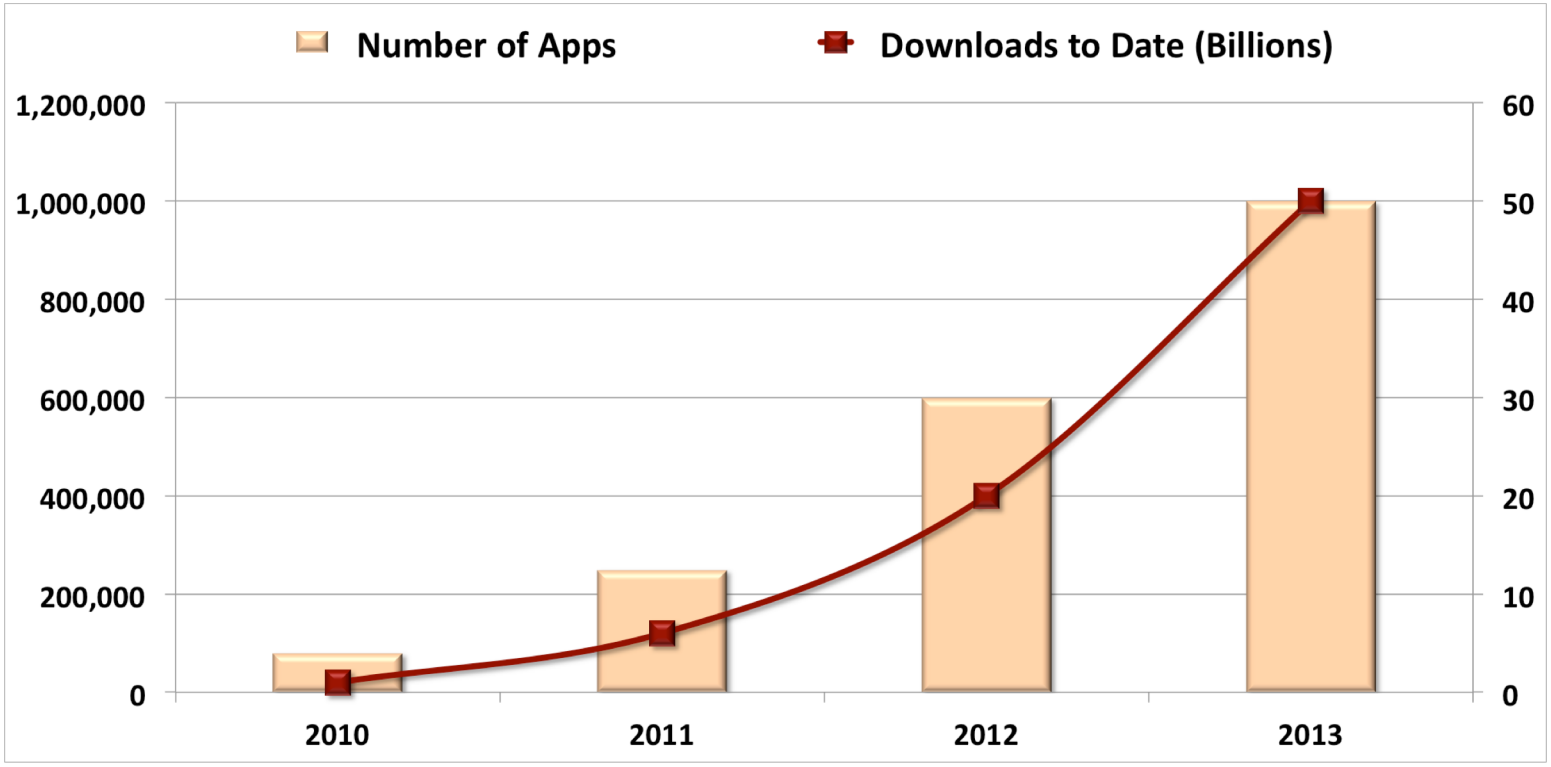
The Growth of Mobile Apps

- 30K new apps on Google Play per month
- 1 million total (July 2013)
- **3 million total (June 2017)**

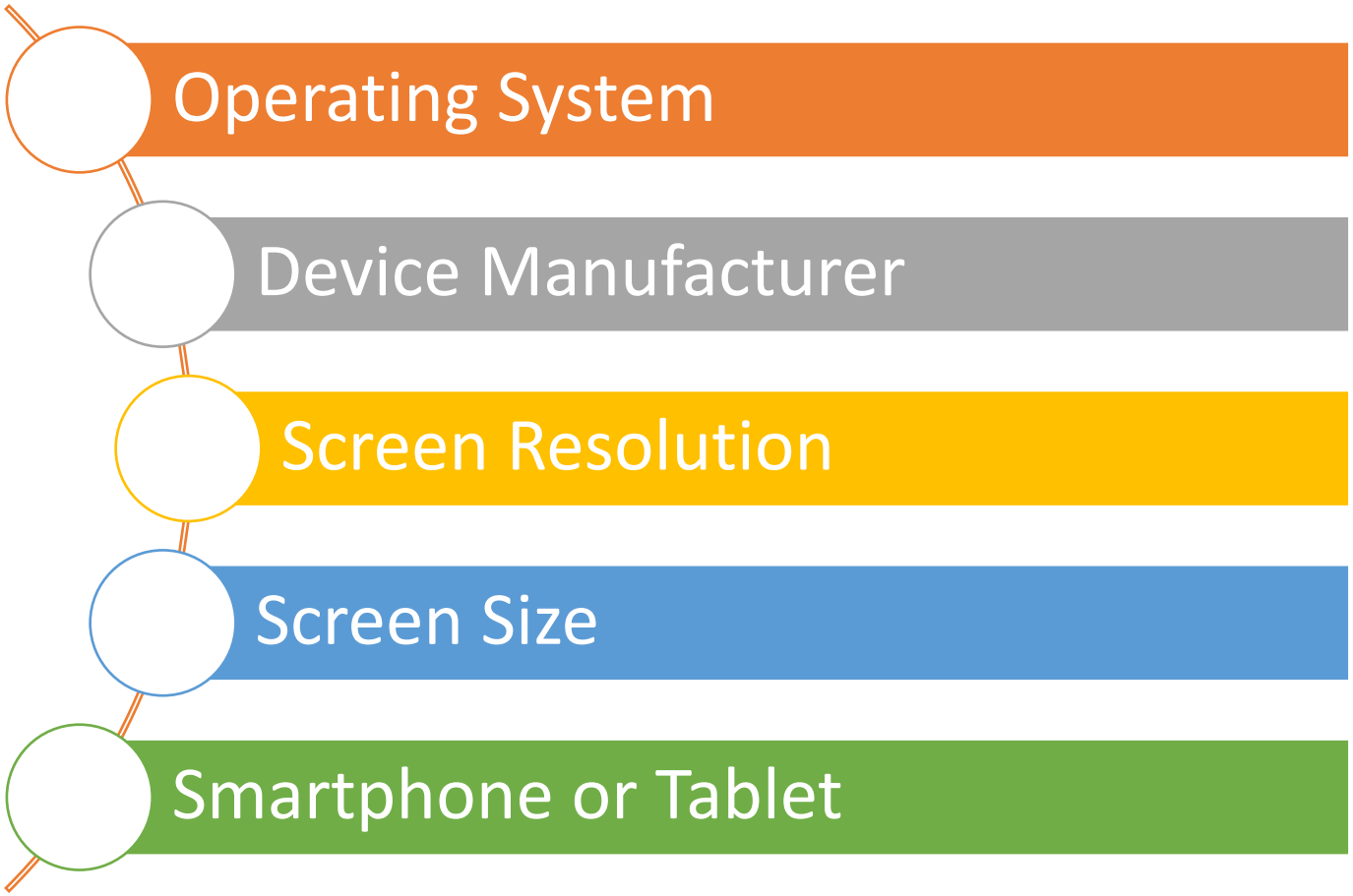


The Growth of Mobile Apps

- 1.5 billion downloads from Google Play per month
- 50 billion total (July 2013)
- **180 billion total (June 2017)**



Complexities in Mobile Landscape

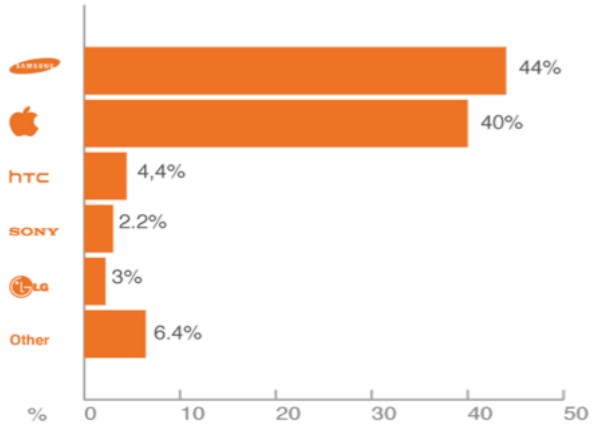


Operating System & Device Manufacturer

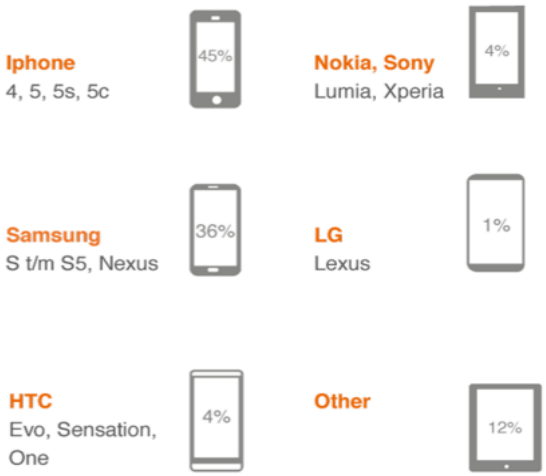
OS usage July 2014

		Android	iOS	Windows	BlackBerry	Nokia
	Worldwide	52.80%	24.42%	2.4%	2.4%	7.9%
	Europe	57.8%	34.2%	3.3%	2.1%	1.1%
	The Netherlands	54.22%	44.30%	1.07%	0.3%	0%

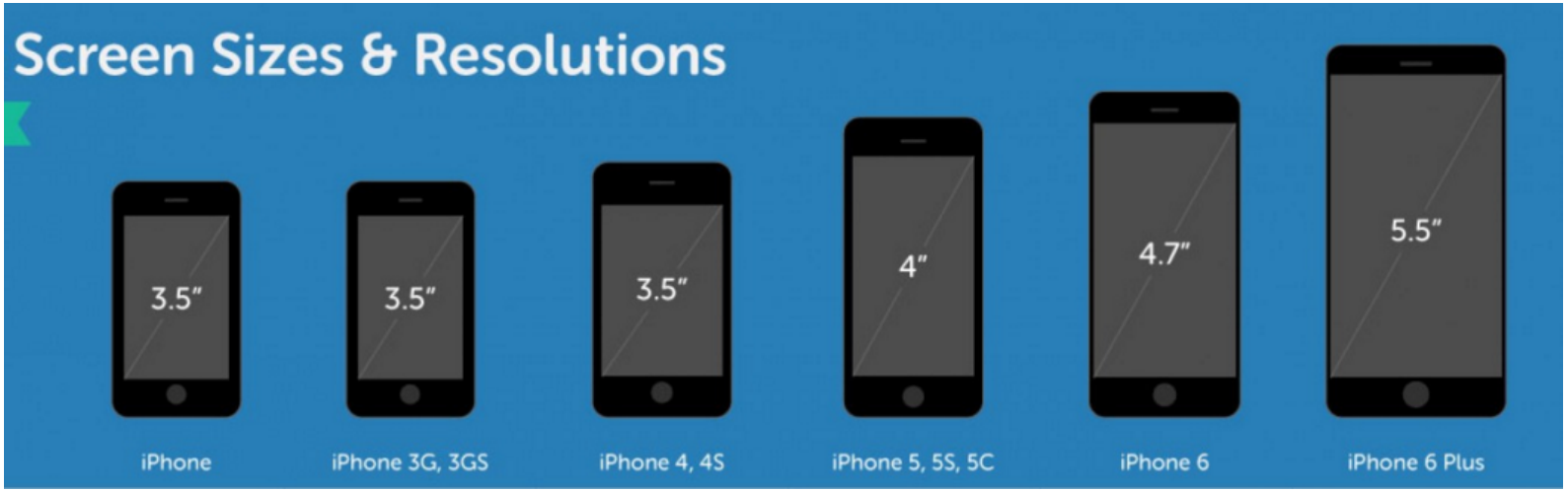
Mobile Vendors



Mobile Devices



Screen Size & Resolution



Globalization: The Big Hurdle

Globalization ready

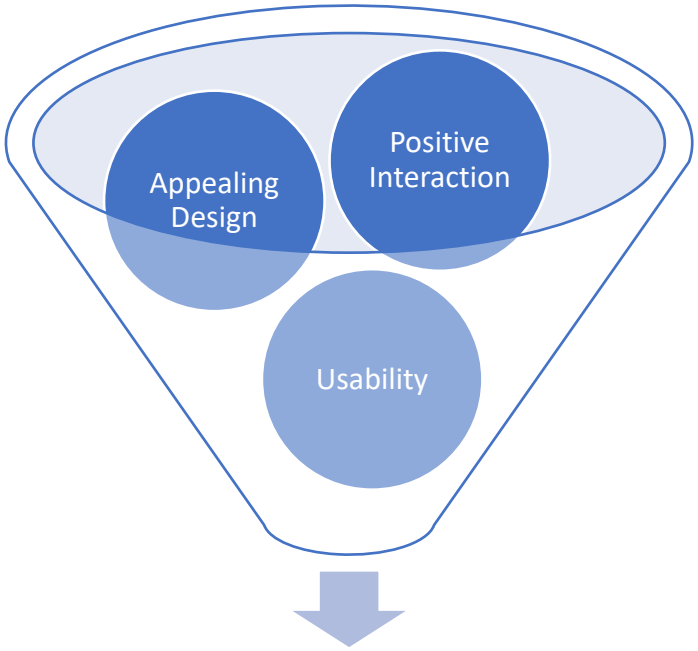
Mobile apps stores are global, Mobile apps should be too.

Supporting the app in around 18 locales

Mobile Testing

GUI Testing

Functional Testing



App Success

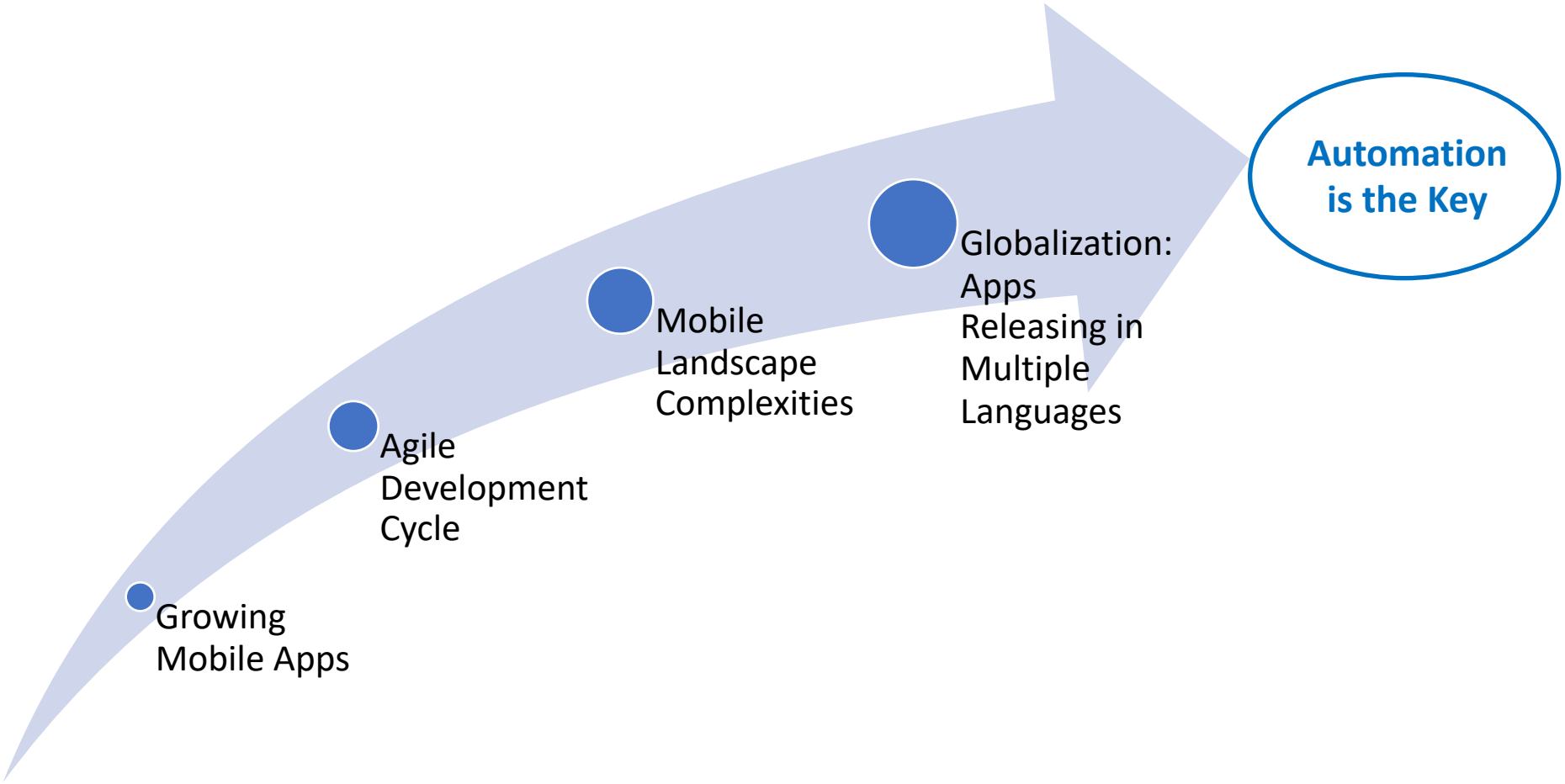
Why do automated UI testing for mobile apps?

Design and marketing will get you users

But if an app doesn't function properly

Users might give one star ratings and won't recommend it

Need of Automation in Mobile Testing



Mobile Testing Tools

- Random Testing
  - Google Monkey, Dynodroid [FSE'13]
- Symbolic Execution
  - ACTeve [FSE'12], JPF-Android [SSEN'12]
- Evolutionary (Genetic) Algorithm
  - Evodroid [FSE'14], Sapienz [ISSTA'16]
- Model-Based Testing
  - GUIRipper [ASE'12], A3E [OOPSLA'13], SwiftHand [OOPSLA'13], PUMA [MobiSys'14], MobiGuitar [IEEE Software'15], Stoat [FSE'17]
- Other Approaches
  - MonkeyLab [MSR'15], CrashScope [ICST'16], TrimDroid [ICSE'16]

Mobile Testing Tools

- Random Testing
  - **Google Monkey**, **Dynodroid [FSE'13]**
- Symbolic Execution
  - ACTeve [FSE'12], JPF-Android [SSEN'12]
- Evolutionary (Genetic) Algorithm
  - Evodroid [FSE'14], Sapienz [ISSTA'16]
- Model-Based Testing
  - GUIRipper [ASE'12], A3E [OOPSLA'13], SwiftHand [OOPSLA'13], PUMA [MobiSys'14], MobiGuitar [IEEE Software'15], Stoat [FSE'17]
- Other Approaches
  - MonkeyLab [MSR'15], CrashScope [ICST'16], **TrimDroid [ICSE'16]**

Monkey: Random Testing

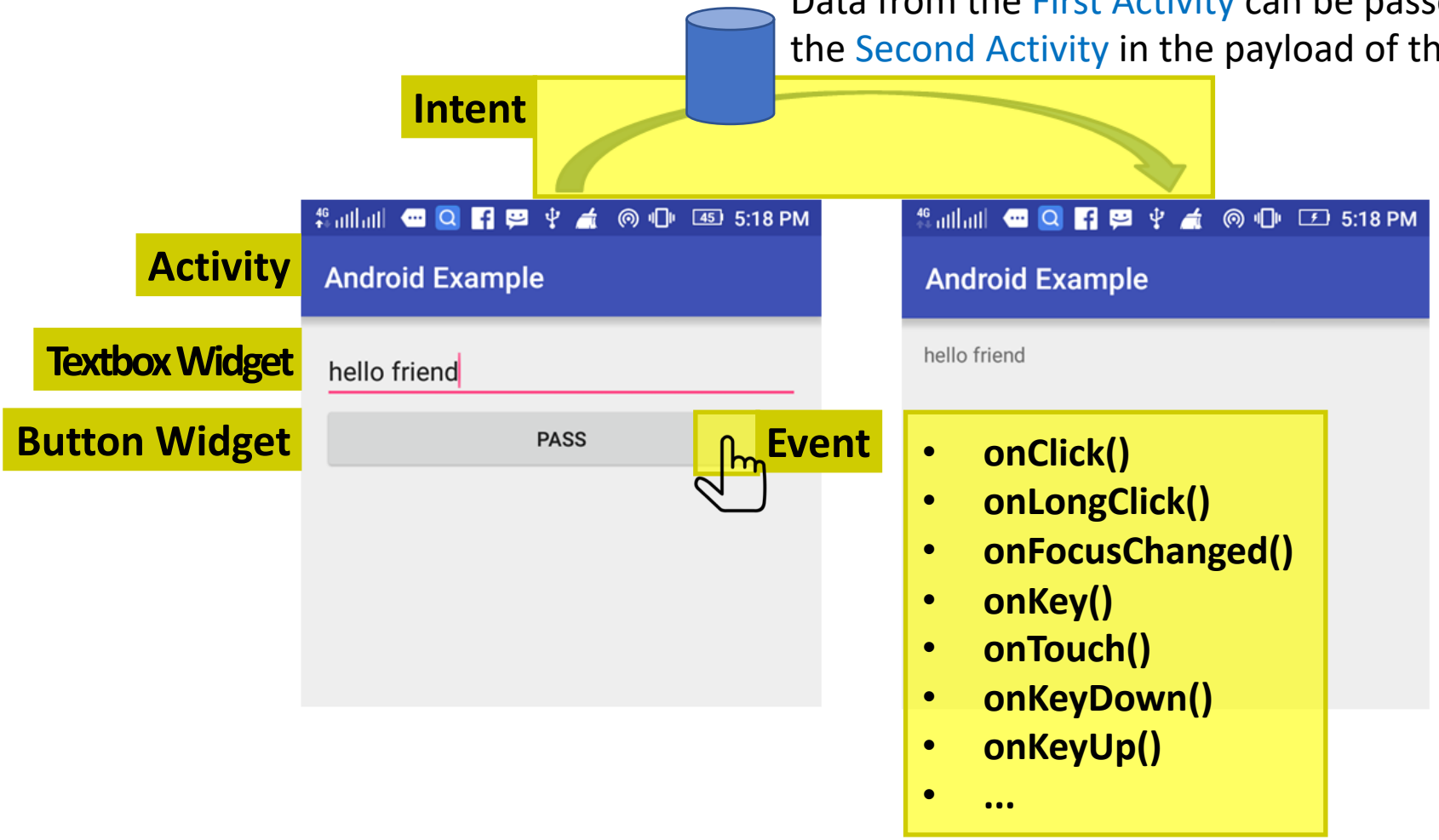
Google

https://developer.android.com/studio/test/monkey

Android GUI Terminology

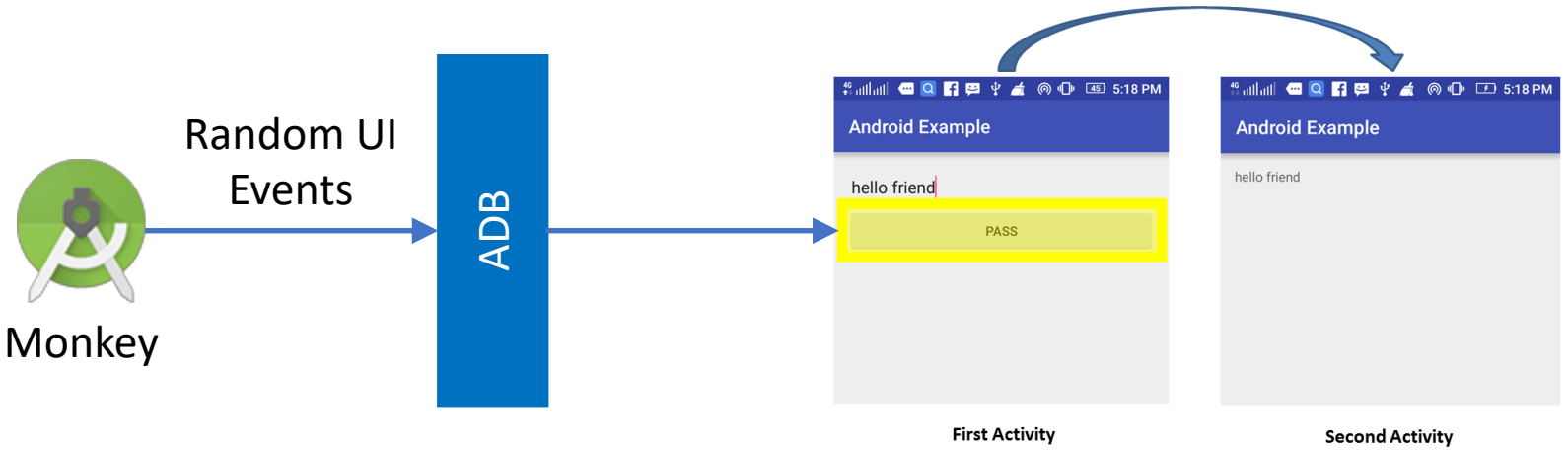
Image Source

Data from the First Activity can be passed into the Second Activity in the payload of the intent



Monkey Overview

- Come with the Android developer toolkit
- Send **random UI events** to the app
- Provide options to change the random behavior
  - Set the **number of events**
  - Restrict the testing to a **specific package**
  - Restrict **event types and frequencies**
  - Adjust **percentage** of various events



Monkey Limitations

- Code coverage can be low
- System events are not considered
  - Fail to cover app functionality controlled by system events
- Random selection strategy is weak
  - not consider the current context or state of the app

Monkey Demo

- https://www.youtube.com/watch?v=dfPNF-6RXVc

Dynodroid: An Input Generation System for Android Apps

Aravind Machiry, Rohan Tahiliani, Mayur Naik

FSE 2013, Distinguished Artifact Award, Citation: 398

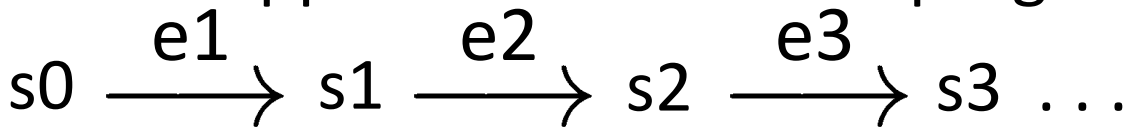
https://dynodroid.github.io

Key Criteria for Input Generation System

- **Robust**: handle real-world apps
- **Blackbox**: no need for sources and the ability to decompile binaries
- **Versatile**: exercise important app functionality
- **Automated**: reduce manual effort
- **Efficient**: avoid generating redundant inputs

Dynodroid

- View an app is an event-driven program



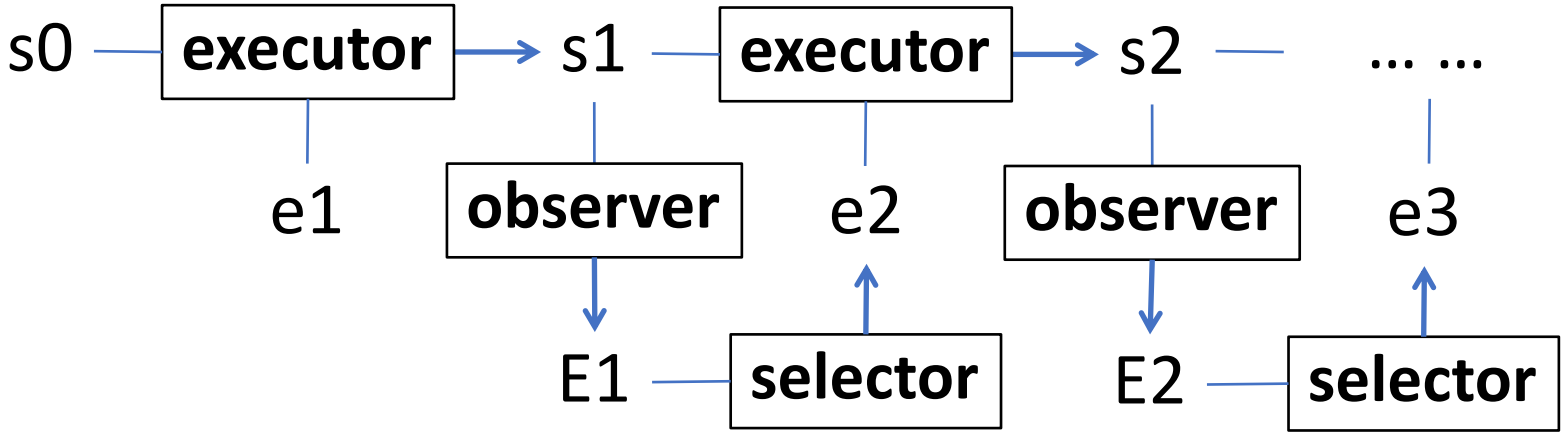
- Broadly two kinds of events
  - **UI event:** LongTap(245, 310), Drag(0, 0, 245, 310), …
  - **System event:** BatteryLow, SmsReceived(“hello”), ...

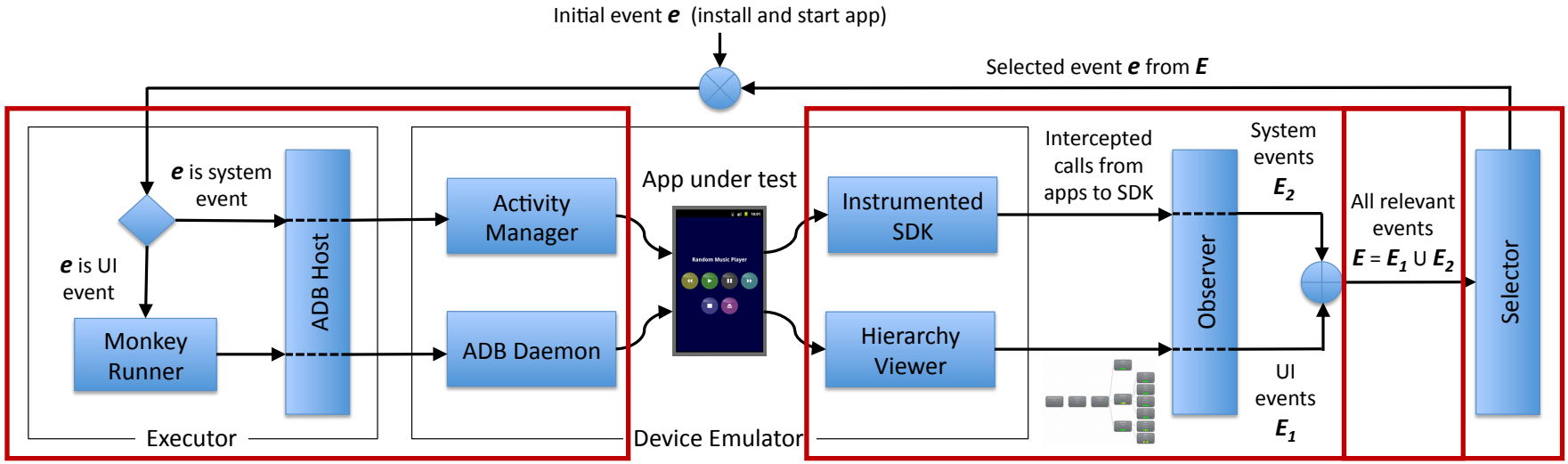
- Assumption: fixed concrete data in each event and environment (sdcard, network, etc.)
  - May cause loss of coverage

Relevant Events

- Key challenge: a large number of possible events
  - E.g., 108 system events in Android Gingerbread
- Insight #1: in any state, a few events are **relevant**
  - A vast majority of events are no-ops
- Insight #2: can identify relevant events by lightly instrumenting SDK once and for all
  - Does not require the instrumentation of the app

Observe-Select-Execute Cycle





Observer for UI Events

- Extract two kinds of data from **leaf-level View objects in the View Hierarchy** about the corresponding UI element
  - the set of callback methods registered and the set of methods overridden by the app, for listening to inputs to this UI element
  - the location and size of the UI element on the touchscreen (the position of its top left corner, its width, height, and scaling factor)

Event Type	Parameters	Relevant Event: Tap (short)	Tap (long)	Drag	Text	Description
Tap	Tap($l + w/2$, $t + h/2$)					Tap at center of view
LongTap	LongTap($l + w/2$, $t + h/2$)					trigger LongTap at center of view
Drag	random one of: Drag(l, t, $l+w$, $t+h$), Drag($l+w$, $t+h$, l, t), Drag(l, $t+h$, $l+w$, t), Drag($l+w$, t, l, $t+h$), Drag(l, $t+h/2$, $l+w$, $t+h/2$), Drag($l+w$, $t+h/2$, l, $t+h/2$), Drag($l+w/2$, t, $l+w/2$, $t+h$), Drag($l+w/2$, $t+h$, $l+w/2$, t)					randomly trigger one of gestures: TL to BR, BR to TL, BL to TR, TR to BL, ML to MR, MR to ML, MT to MB, MB to MT
Text	arbitrary fixed string					trigger arbitrary text input
	If app registers callback:					
	onClickListener	✓				
	onLongClickListener		✓			
	onTouchListener	✓	✓	✓		
	onKeyListener				✓	
	onCreateContextMenuListener		✓			
	If app overrides method:					
	onTouchEvent	✓	✓	✓		
	performLongClick		✓			
	performClick	✓				
	onKeyDown				✓	
	onKeyUp				✓	

Observer for System Events

- Broadcast receiver events, e.g., sms_received
- System service events, e.g., alarm and location

- An app registers a callback to be called by the SDK when the event occurs, and may unregister it to stop receiving the event

- Instrument the SDK to observe when an app registers (or unregisters) for each type of system event

Event Selection Strategies

- **Frequency**
  - Select the event that has been selected least frequently
  - Drawback: its deterministic nature leads the app to the same state in repeated runs
- **UniformRandom**
  - Select an event uniformly at random
  - Drawback: does not consider domain knowledge; no distinction between UI and system events, different contexts in which an event occurs, and frequent and infrequent events
- **BiasedRandom**
  - Combines benefits of above without drawbacks

BiasedRandom Event Selection Algorithm

- Global map **G(e, S)** tracks number of times **e** is selected in context **S**
  - Context **S** = set of events relevant when **e** is selected

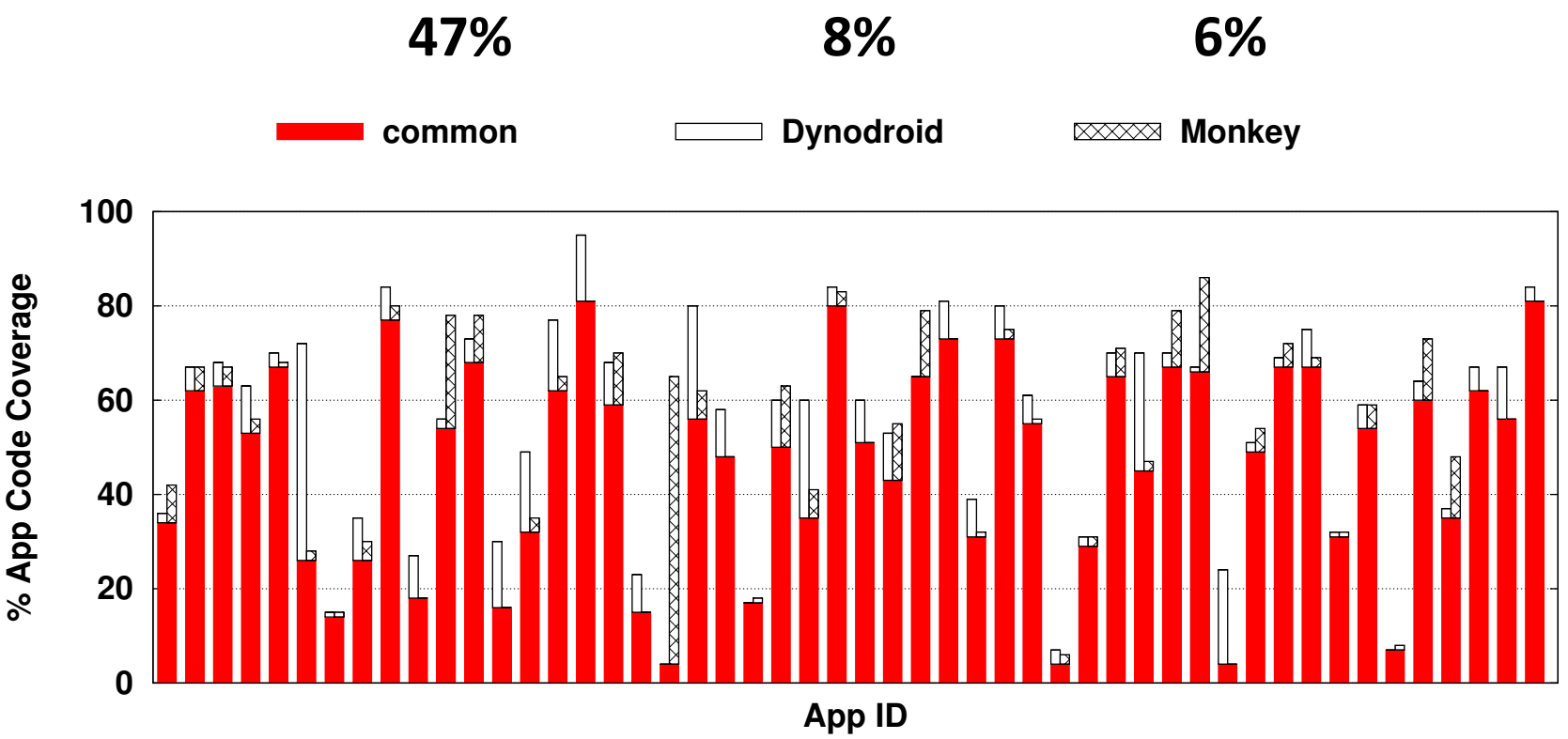
- Local map **L(e)** computed to select the next event from relevant event set **S**
  - Initialize: **L(e)** to 0 for each **e** in **S**
  - Repeat:
    - Pick an **e** in **S** uniformly at random
    - If **L(e) = G(e, S)** increment **G(e, S)** and return **e** else increment **L(e)**

Evaluation 1: App Code Coverage

- 50 open-source apps from F-Droid
  - SLOC ranging from 16 to 22K, mean of 2.7K
- Evaluated Approaches:
  - Dynodroid using three selection strategies
  - Monkey
  - Expert human users

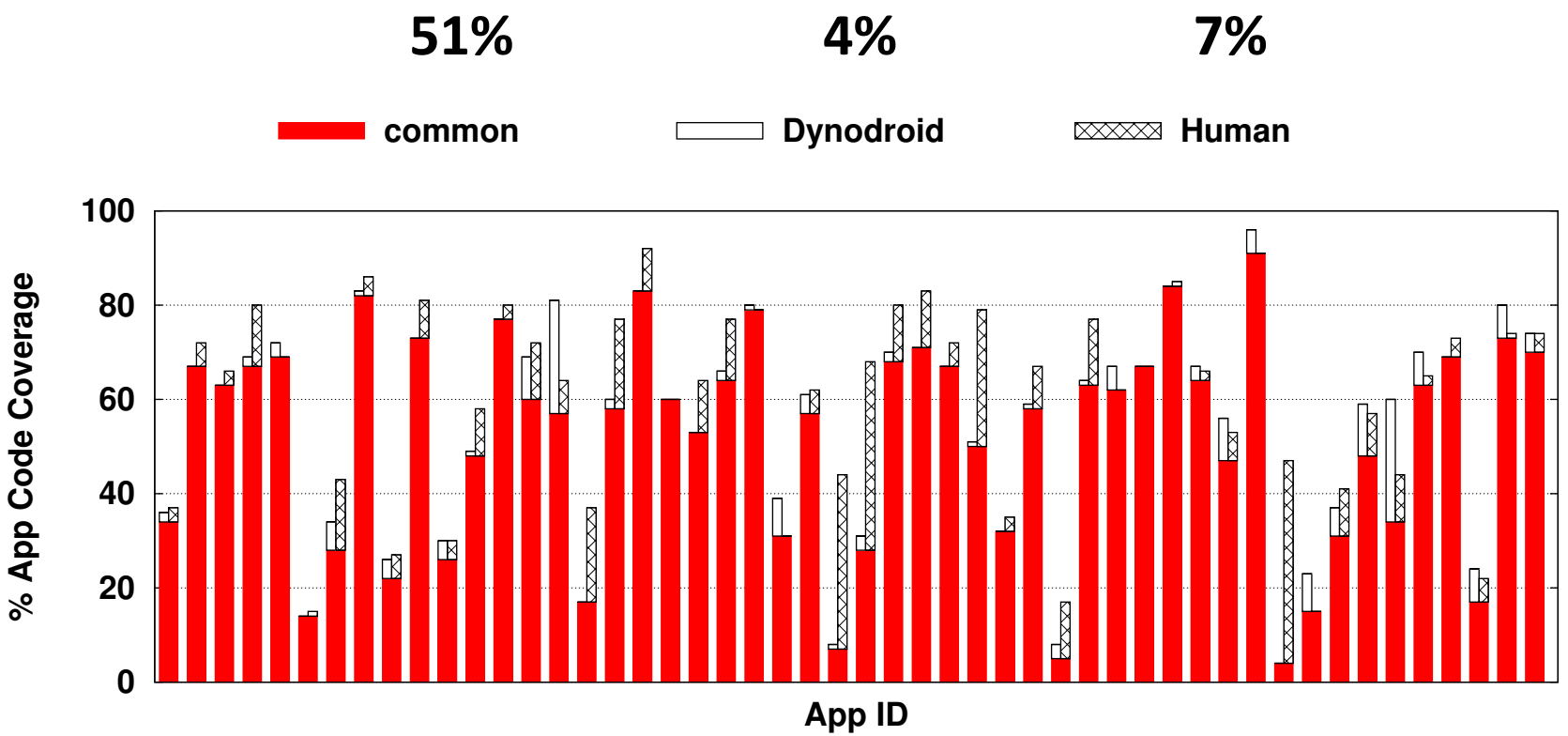
Approach	#Events	#Runs
Dynodroid - Frequency	2,000	1
Dynodroid - UniformRandom	2,000	3
Dynodroid - BiasedRandom	2,000	3
Monkey	10,000	3
Humans	No limit	>= 2

Dynodroid vs. Monkey



Dynodroid achieves higher coverage than Monkey for 30 of the 50 apps

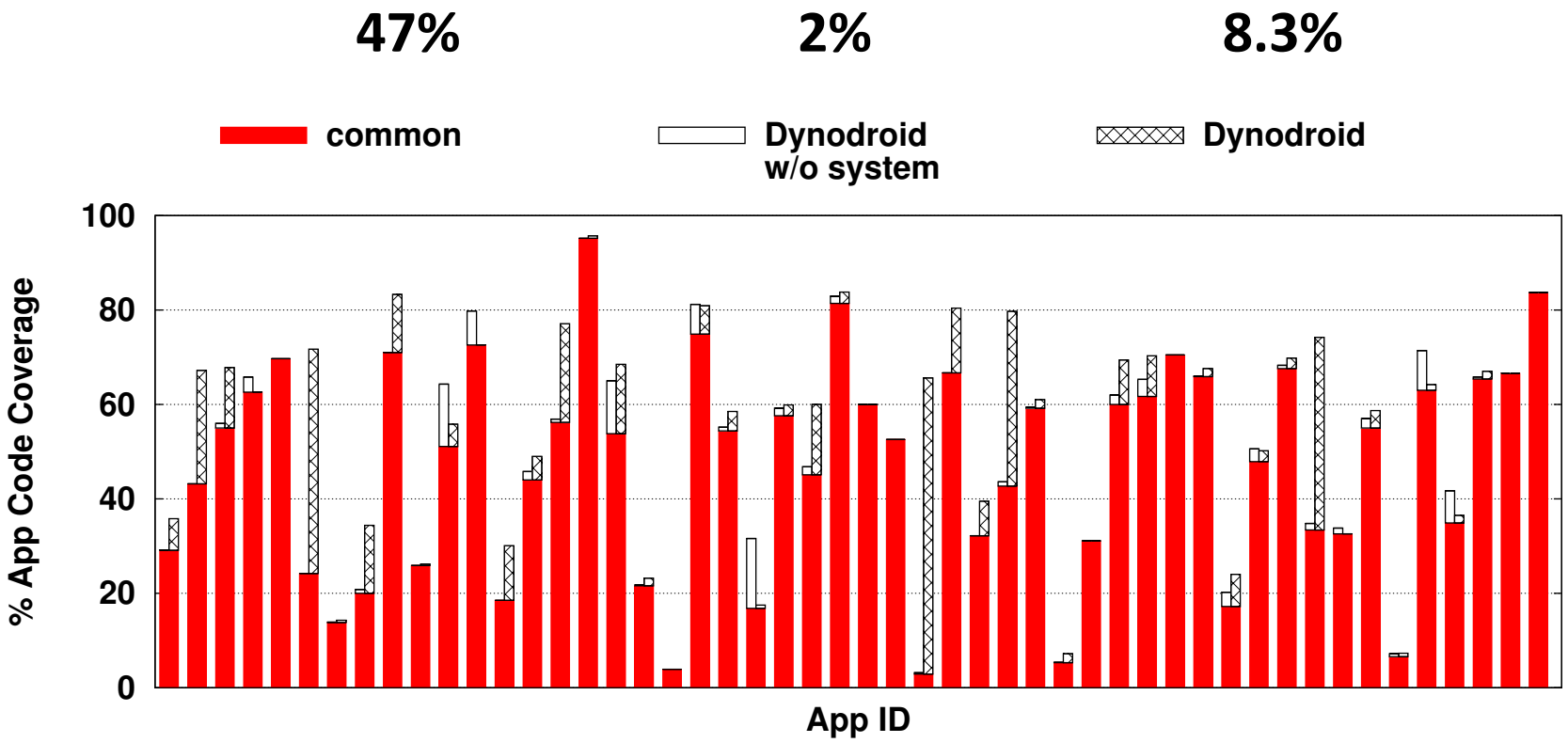
Dynodroid vs. Humans



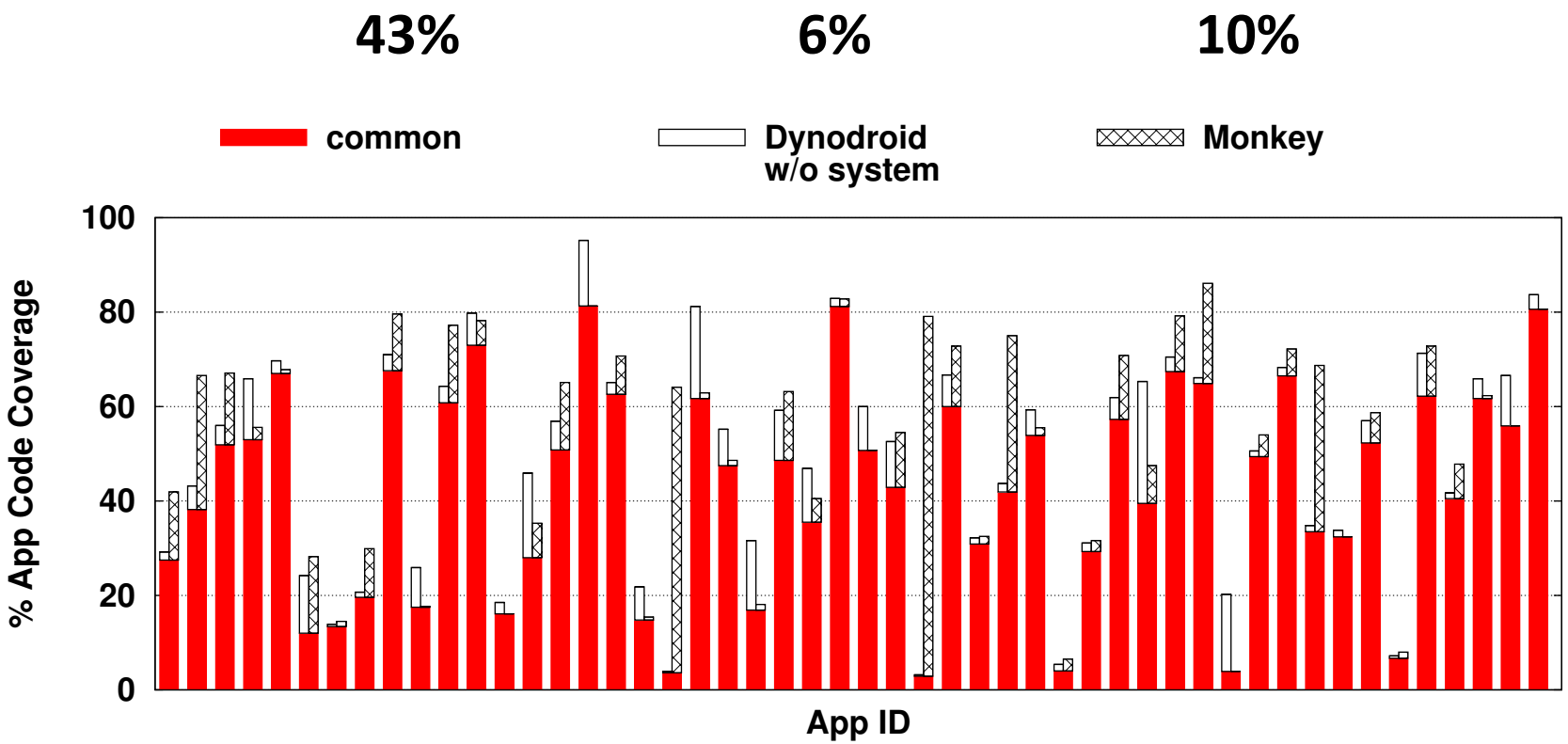
Automation Degree = C(Dynodroid ∩ Human) / C(Human)

Range = **8-100%**, Average = **83%**, Standard Deviation = **21%**

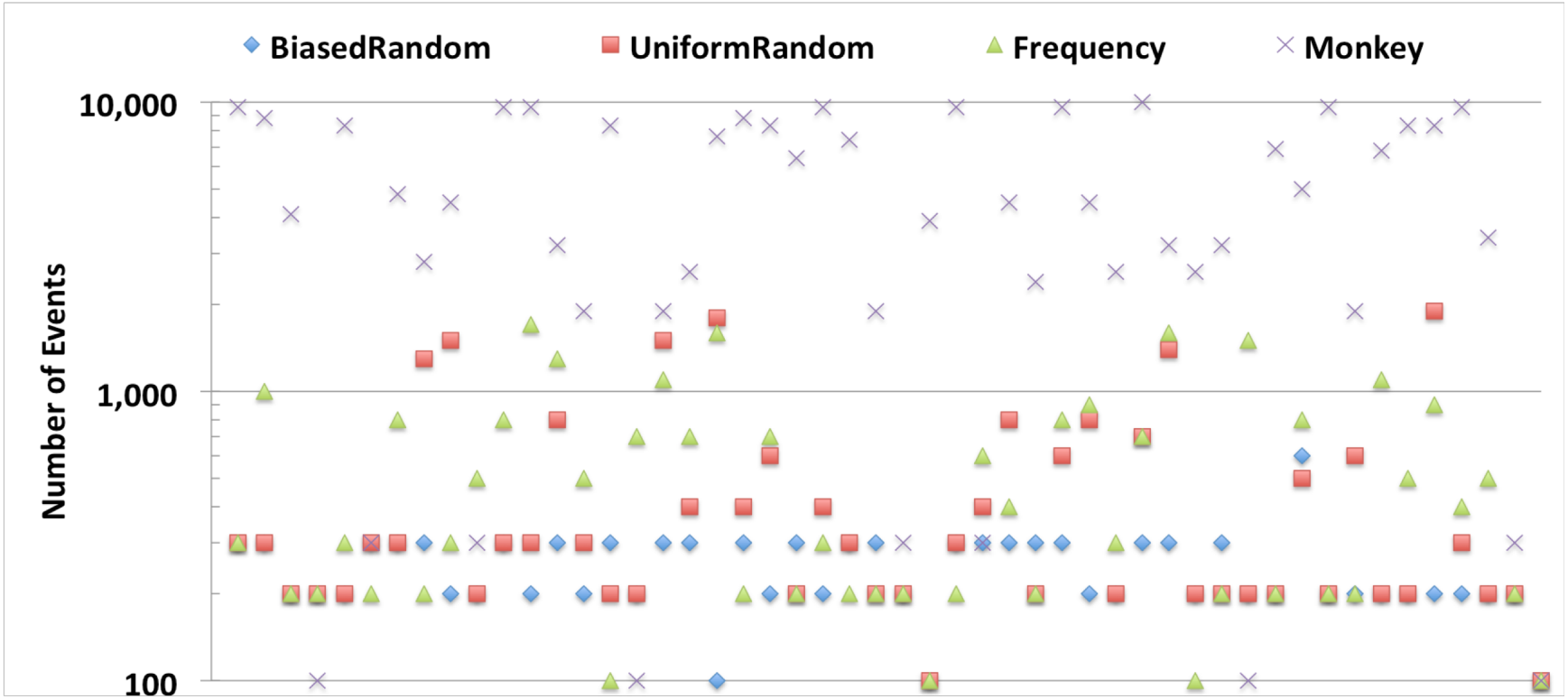
Dynodroid without vs. with System Events



Dynodroid without System Events vs. Monkey



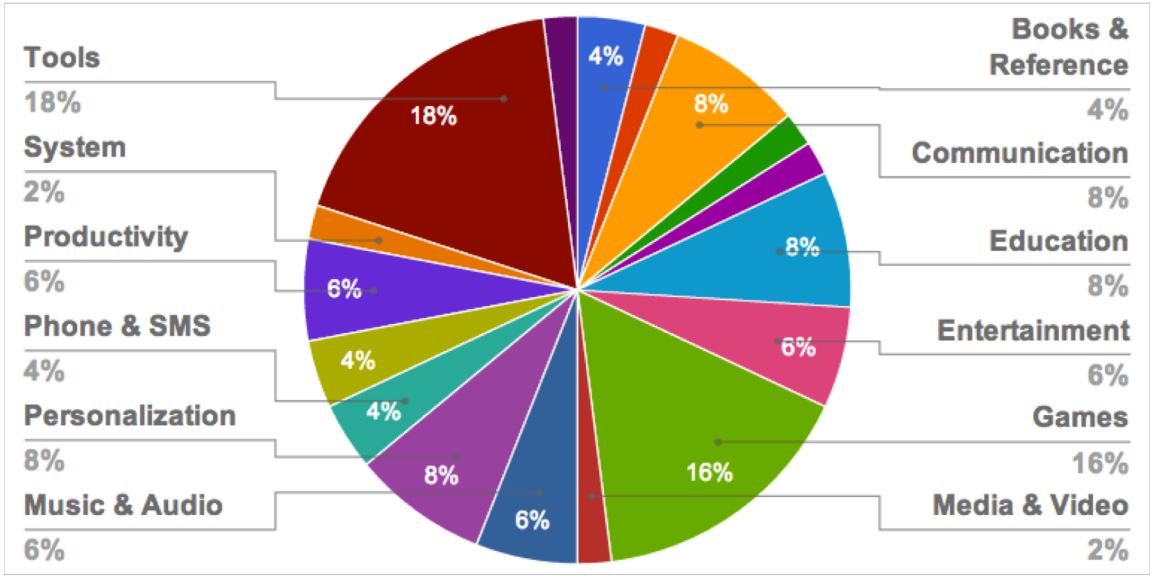
Minimum Number of Events to Peak Coverage



- Monkey requires **20X** more events than BiasedRandom
- Frequency and UniformRandom require **2X** more events than BiasedRandom

Evaluation 2: Bugs Found in Apps

- 1,000 most popular free apps from Google Play



- Conservative notion of bug: FATAL EXCEPTION (app forcibly terminated)

Bugs Found in 50 F-Droid Apps

App Name	Bugs	Kind	Description
PasswordMakerProForAndroid	1	Null	Improper handling of user data.
com.morphoss.acal	1	Null	Dereferencing null returned by an online service.
hu.vsza.adsdroid	2	Null	Dereferencing null returned by an online service.
cri.sanity	1	Null	Improper handling of user data.
com.zoffcc.applications.aagtl	2	Null	Dereferencing null returned by an online service.
org.beide.bomber	1	Array	Game indexes an array with improper index.
com.addi	1	Null	Improper handling of user data.

Bugs Found in 1,000 Google Play Apps

App Name	Bugs	Kind	Description
com.ibm.events.android.usopen	1	Null	Null pointer check missed in onCreate() of an activity.
com.nullsoft.winamp	2	Null	Improper handling of RSS feeds read from online service.
com.almalence.night	1	Null	Null pointer check missed in onCreate() of an activity.
com.avast.android.mobilesecurity	1	Null	Receiver callback fails to check for null in optional data.
com.aviary.android.feather	1	Null	Receiver callback fails to check for null in optional data.

Limitations

- Does not exercise inter-app communication
  - Communication via key-value maps (“Bundle” objects)
  - Could synthesize such maps symbolically
- Uses fixed, concrete data for events
  - E.g., geo-location, touch-screen coordinates, etc.
  - Could randomize or symbolically infer such data
- Requires instrumenting the platform SDK
  - Limited to particular SDK version
  - Lightweight enough to implement for other versions

Dynodroid vs. Monkey

- Advantages compared to Monkey
  - **Consider both UI and System events**
    - Only checks the relevant events for the app
  - **Smarter random selection methods**
    - Frequency: least frequently selected are first selected
    - Biased: events relevant in more contexts are more likely selected
- Limitations compared to Monkey
  - **5× slower than Monkey**
    - The observer needs to query the view hierarchy
    - Uses reflection, which is usually slow
  - **Need a modified Android SDK**
    - Limited to a particular SDK version
    - Migrating to other versions may require code changes

Reducing Combinatorics in GUI Testing of Android Applications

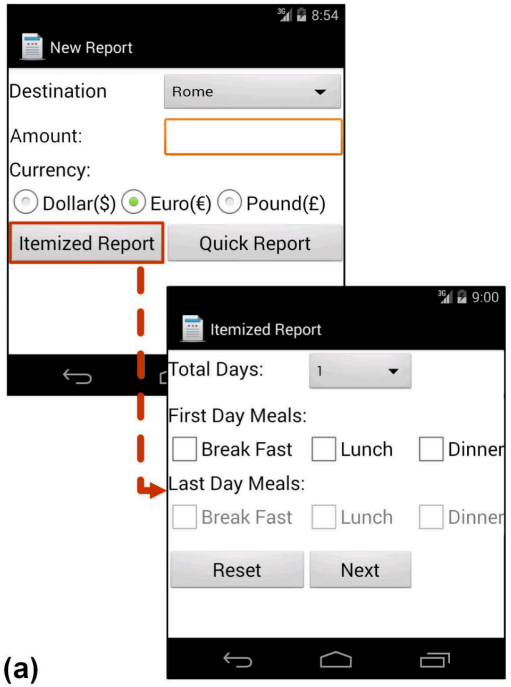
Nariman Mirzaei, Joshua Garcia, Hamid Bagheri, Alireza Sadeghi, Sam Malek

ICSE 2016

https://www.ics.uci.edu/~seal/projects/trimdroid/index.html

A Running Example

- The Expense Reporting System (ERS) app

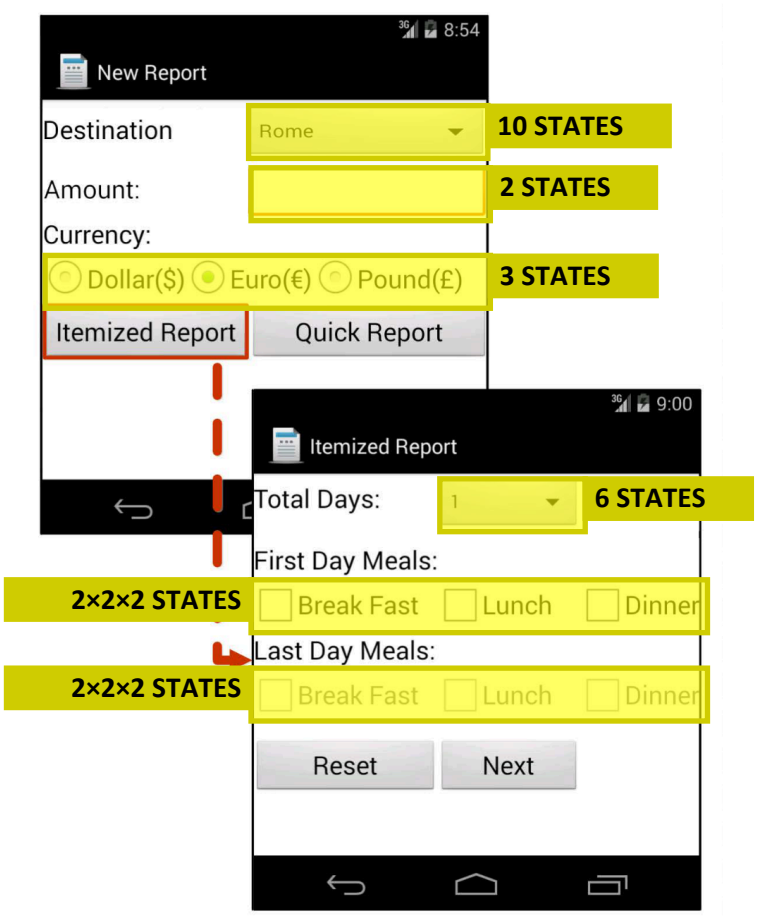


(a)

Contains 5 Activities

- **NewReportActivity**: main/entry screen;
- **ItemizedReportActivity**: to report itemized meal expenses for a day;
- **QuickReportActivity**: to report aggregated meal expenses for a trip;
- **ConfirmationActivity**: to show the expense to the user for confirmation before submitting;
- **SummaryActivity**: To show a summary of the expense after submitting.

Explosion of Combinatorial States

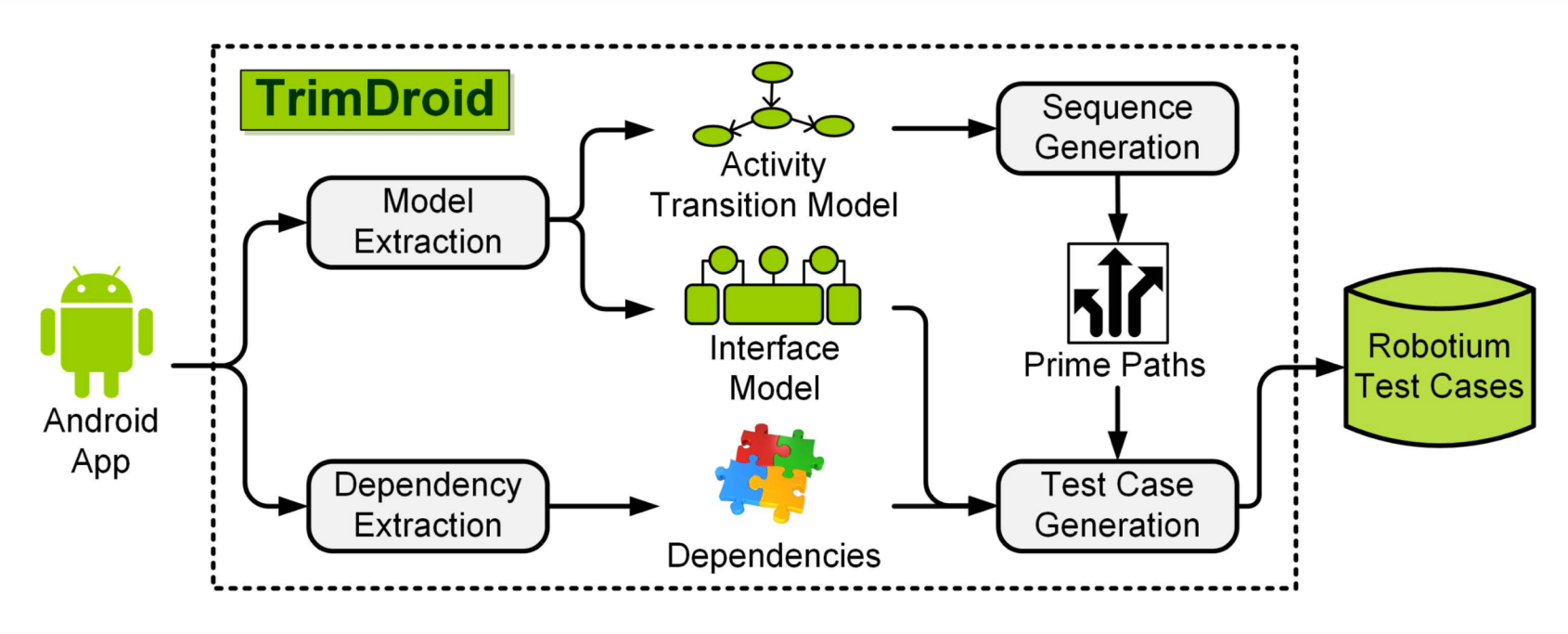


$10 \times 2 \times 3$
$= 60$ STATES

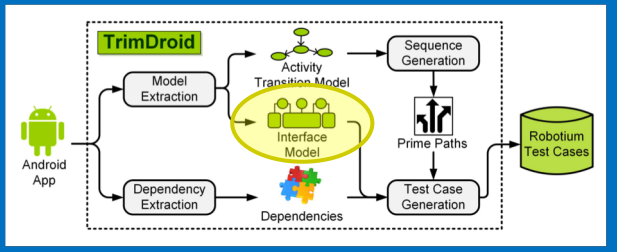
$6 \times 2^3 \times 2^3$
$= 384$ STATES

60×384
$= 23{,}040$ STATES!

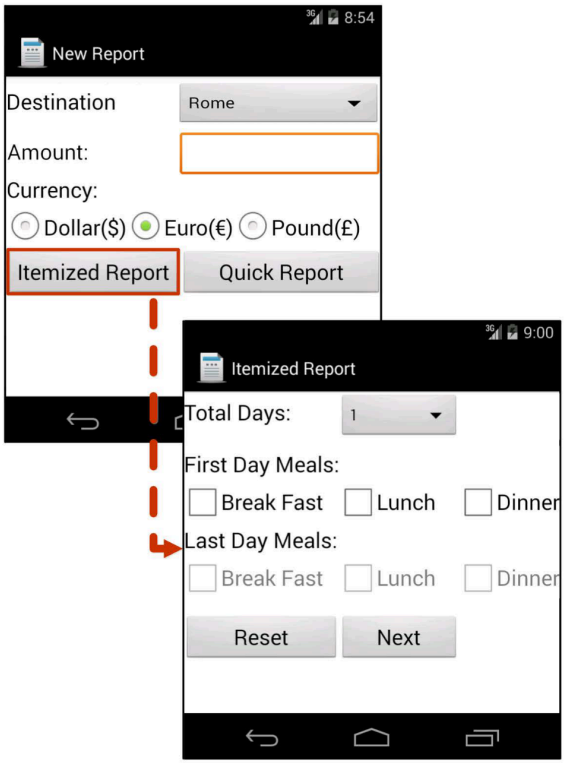
TrimDroid Overview



Interface Model



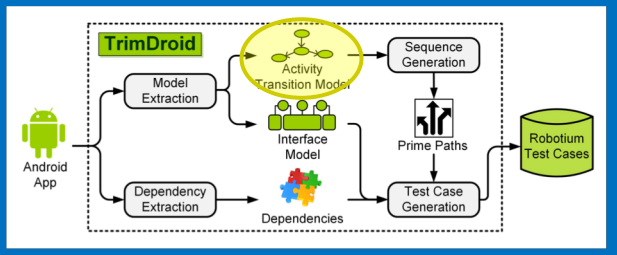
- An **interface model** contains information about all the GUI inputs of an Android app
  - Extract from the manifest and layout XML files in the APK



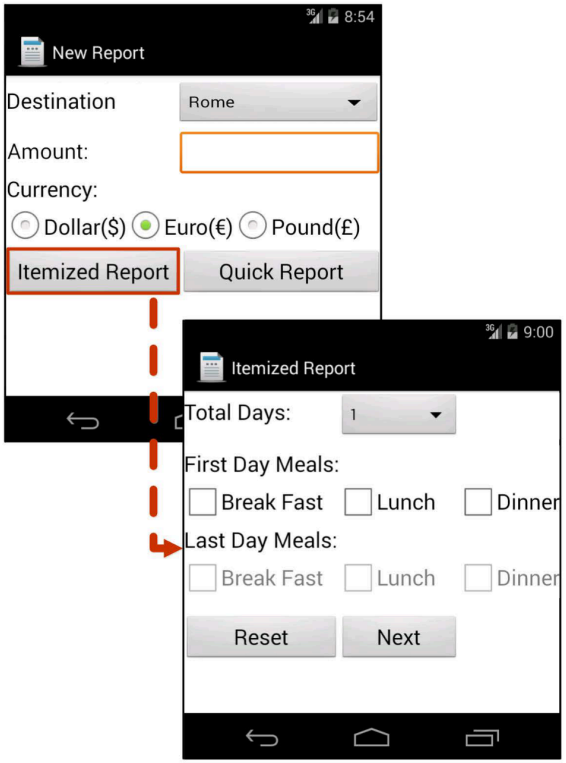
NewReportActivity			
#	$ic(dest)$	$ic(amount)$	$ic(cur)$
1	Rome	100	Euro
2	London	0	Dollar
3			Pound
.	.		
.	.		
10	Berlin		

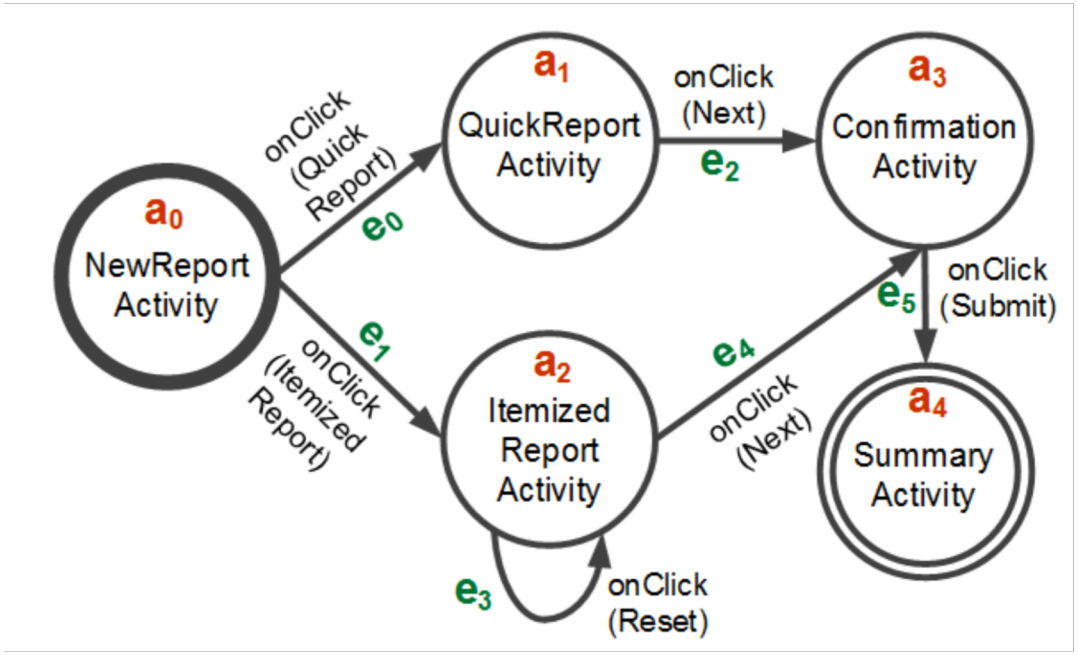
ItemizedReportActivity							
#	$ic(totalDays)$	$ic(fbf)$	$ic(fl)$	$ic(fd)$	$ic(lbf)$	$ic(ll)$	$ic(ld)$
1	1	true	true	true	true	true	true
2	2	false	false	false	false	false	false
3	3						
.	.						
.	.						
6	6						

Activity Transition Model

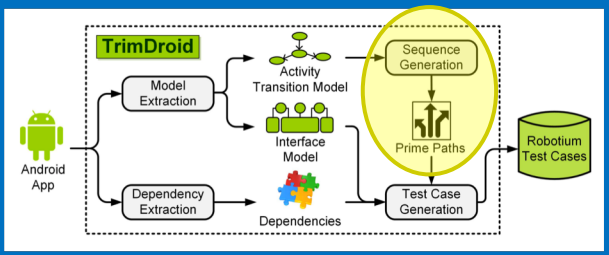


- An **activity transition model** represents the high-level behavior of an app's GUI, i.e., the transitions resulting from invocations of the event handlers in the activities
  - Analyze manifest XML file and traverse the call graph

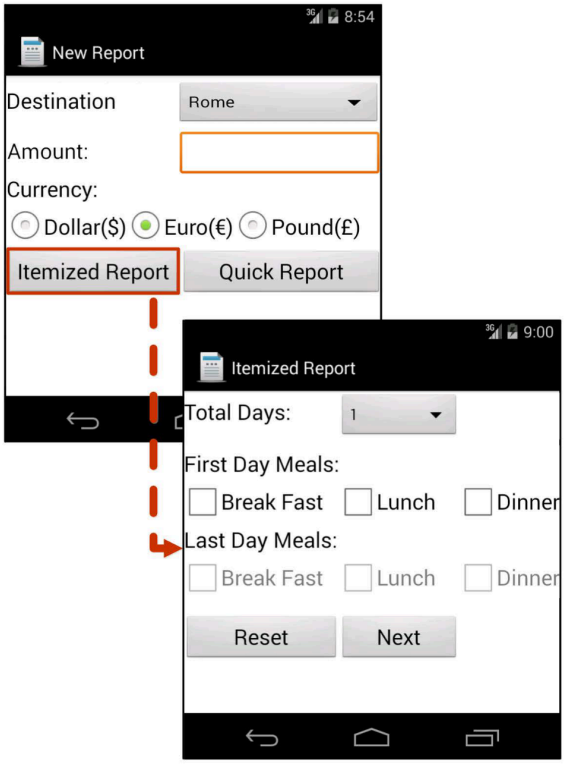


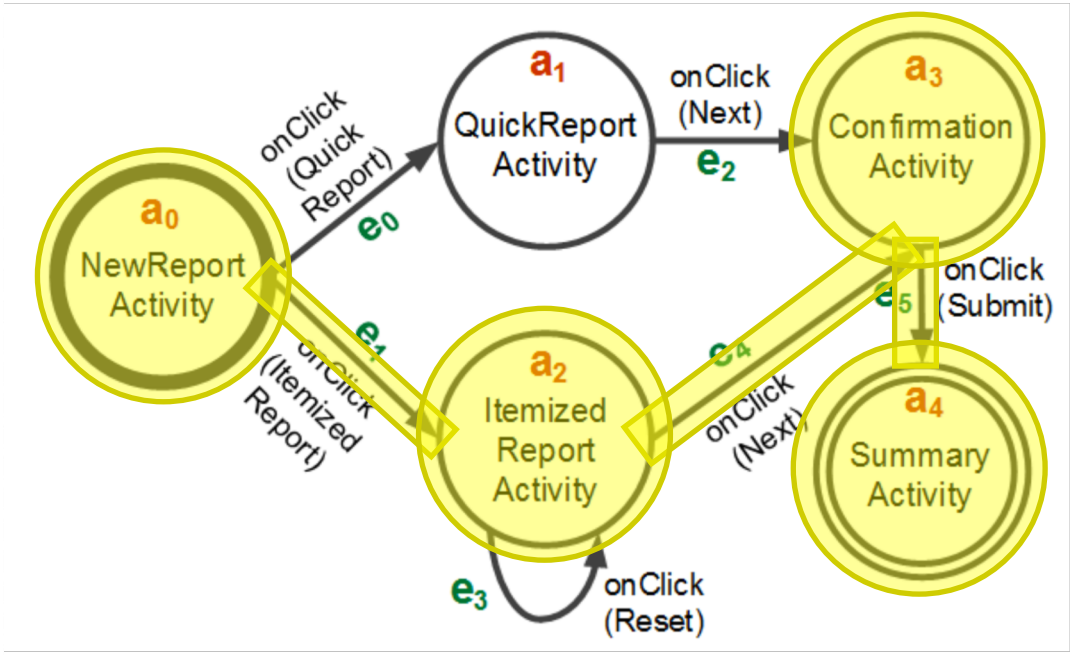


Sequence Generation

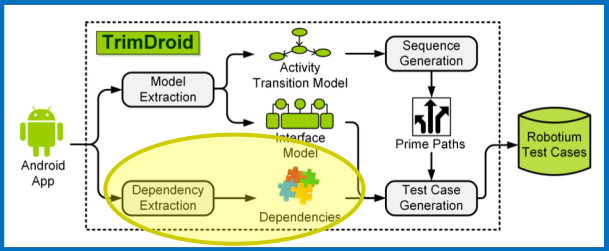


- A **simple path** is a sequence of transitions from the starting activity, where no activity node appears more than once
- A **prime path** is a simple path that does not appear as a sub-path of any other simple path

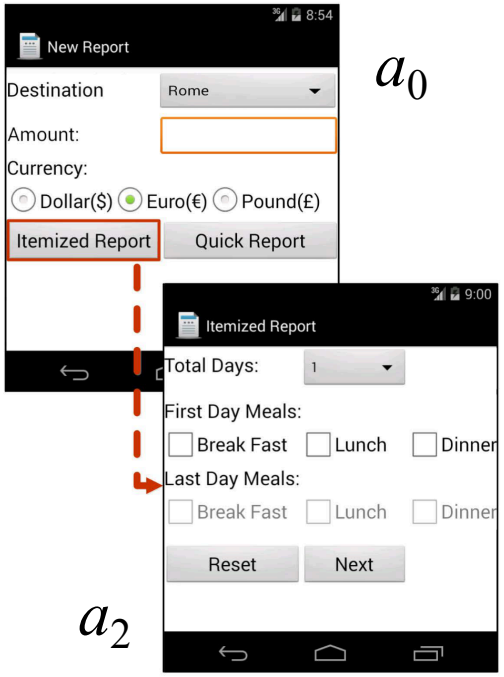




Dependency Extraction



- **Type 1: Widget Dependency**
- Two widgets $\mathbf{w_1}$ and $\mathbf{w_2}$ are dependent if the combinations of the values of $\mathbf{w_1}$ and $\mathbf{w_2}$ affect an app's control- or data-flow

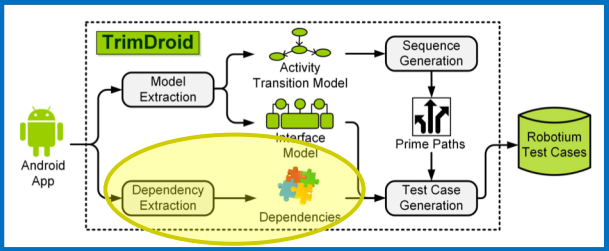


a_0

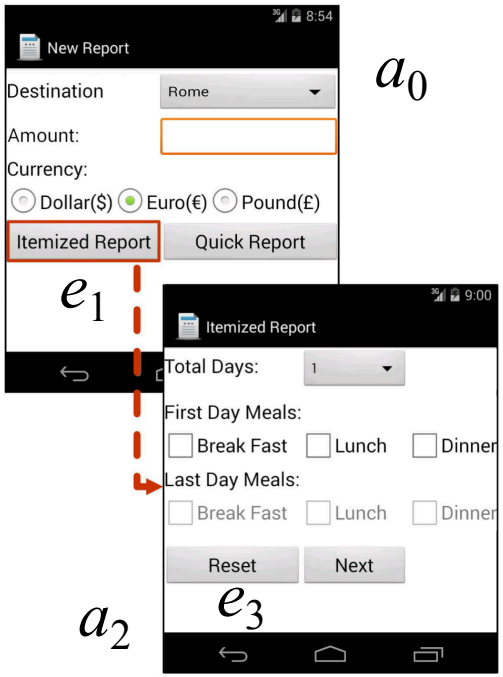
a_2

$$wDep(a_0) = \{\{dest, cur\}, \{amount\}\}$$
$$wDep(a_2) = \{\{totalDays, lbf, ll, ld\}, \{fbf\}, \{fl\}, \{fd\}\}$$

Dependency Extraction

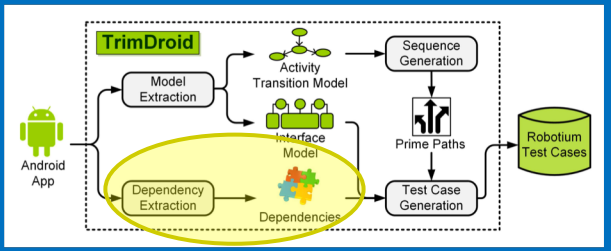


- **Type 2: Handler Dependency**
- A widget **w** and an event handler **e** is dependent if the value of **w** is used in **e**
  - All combinations of the widget values in **w** and the event resulting in the invocation of **e** should be tried

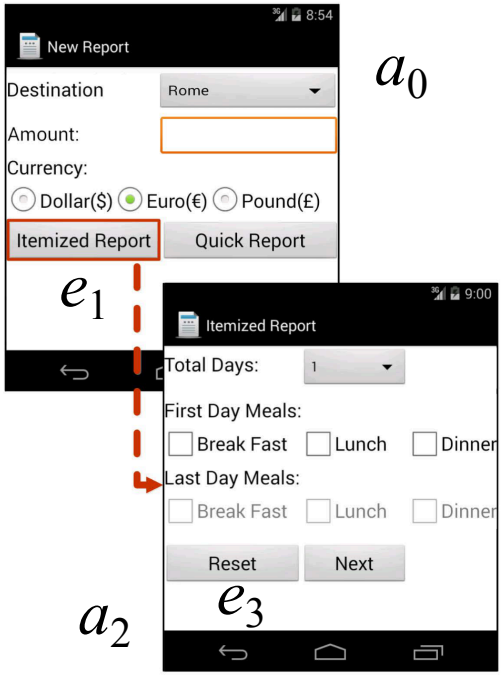


$$hDep(a_0, e_1) = \{\{dest, cur\}, \{amount\}\}$$
$$hDep(a_2, e_3) = \{\}$$

Dependency Extraction

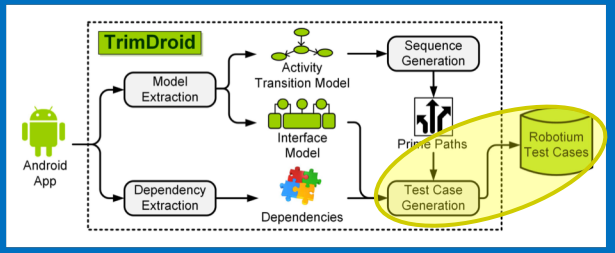


- **Type 3: Activity Dependency**
- An activity $\mathbf{a_2}$ has a dependency to an activity $\mathbf{a_1}$ if the widget values in **w** of $\mathbf{a_1}$ may impact the behavior of $\mathbf{a_2}$
  - All combinations of the widget values in **w** should be tried

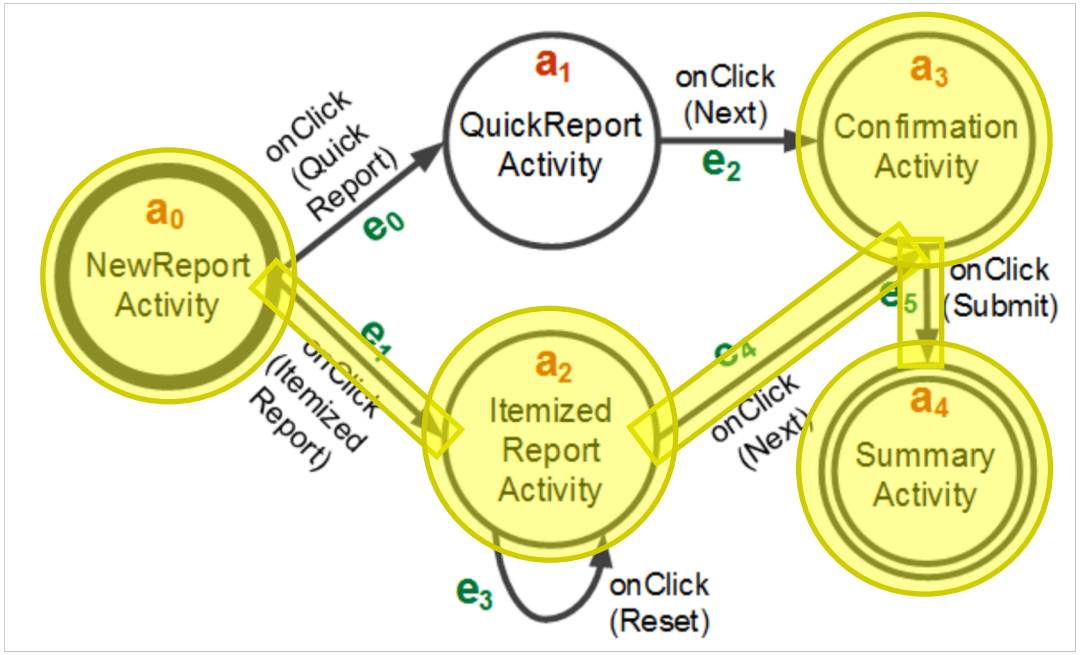


$$aDep(a_0, e_1, a_2) = true$$
$$aDep(a_2, e_3, a_2) = false$$

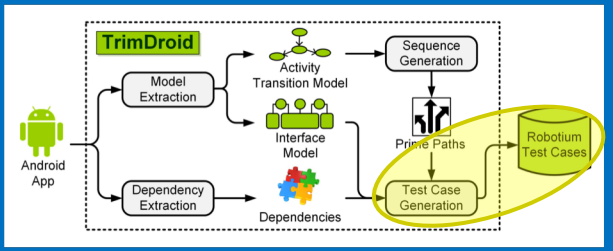
Test Case Generation



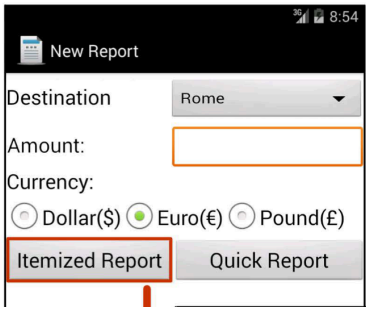
- **Test cases for a prime path** is generated by combining test cases of events in each activity
- **Test cases for an event in an activity** is generated by combining the values of widgets



Test Case Generation



- **Example: Generating test cases for an event in an activity**



Activity a_0 of ERS

NewReportActivity			
#	$ic(dest)$	$ic(amount)$	$ic(cur)$
1	Rome	100	Euro
2	London	0	Dollar
3			Pound
.	.		
.	.		
10	Berlin		

Input classes for each widget in a_0

$$hDep(a_0, e_1) = \{\{dest, cur\}, \{amount\}\}$$

Handler dependency sets of e_1 in a_0

Widget value combinations for each dependent set

#	$WC_{(dest,cur)}$	$WC_{(amount)}$
1	Rome, Euro	100
2	Rome, Dollar	0
3	Rome, Pound	
4	London, Euro	
5	London, Dollar	
.	.	
.	.	
30	Berlin, Pound	

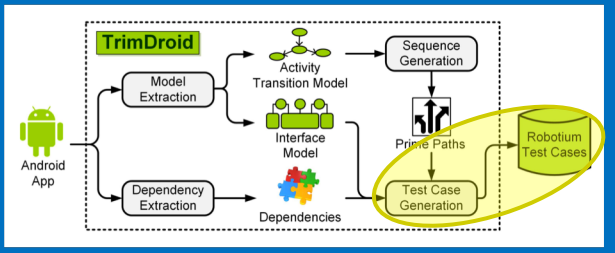
MERGE

Simply make sure every unique value appears

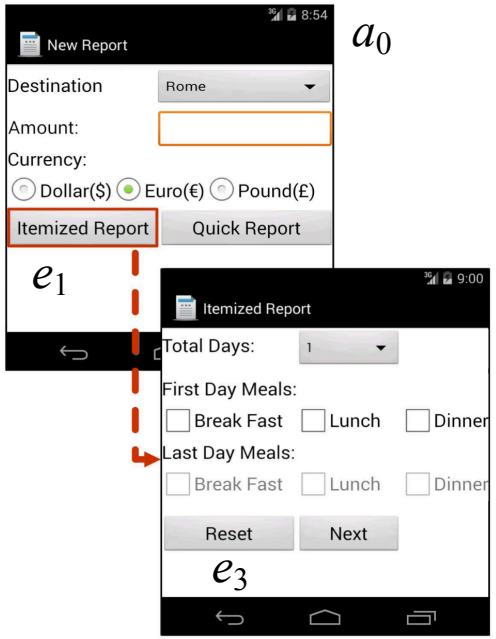
Test cases for event e_1 of activity a_0

#	AT $hDep(a_0, e_1)$
1	{{Rome , Euro, 100}, ItemizedReport}
2	{{Rome , Dollar, 0}, ItemizedReport}
3	{{Rome , Pound, 100}, ItemizedReport}
4	{{London , Euro, 0}, ItemizedReport}
5	{{London , Dollar, 100}, ItemizedReport}
.	.
.	.
30	{{Berlin, Pound, 0}, ItemizedReport}

Test Case Generation



- **Example: Generating test cases for a prime path**



a_0

e_1

e_3

a_2

$a_0 \xrightarrow{e_1} a_2 \xrightarrow{e_3} a_2$

Example path

#	$AT_{hDep(a_0,e_1)}$
1	{{Rome , Euro, 100}, ItemizedReport}
2	{{Rome , Dollar, 0}, ItemizedReport}
3	{{Rome , Pound, 100}, ItemizedReport}
4	{{London , Euro, 0}, ItemizedReport}
5	{{London , Dollar, 100}, ItemizedReport}
.	.
.	.
30	{{Berlin, Pound, 0}, ItemizedReport}

Test cases for event e_1 of activity a_0

#	$AT_{hDep(a2, e3)}$
1	{Reset}

Test cases for event e_3 of activity a_2

$$aDep(a_0, e_1, a_2) = true$$
$$aDep(a_2, e_3, a_2) = false$$

Activity dependencies

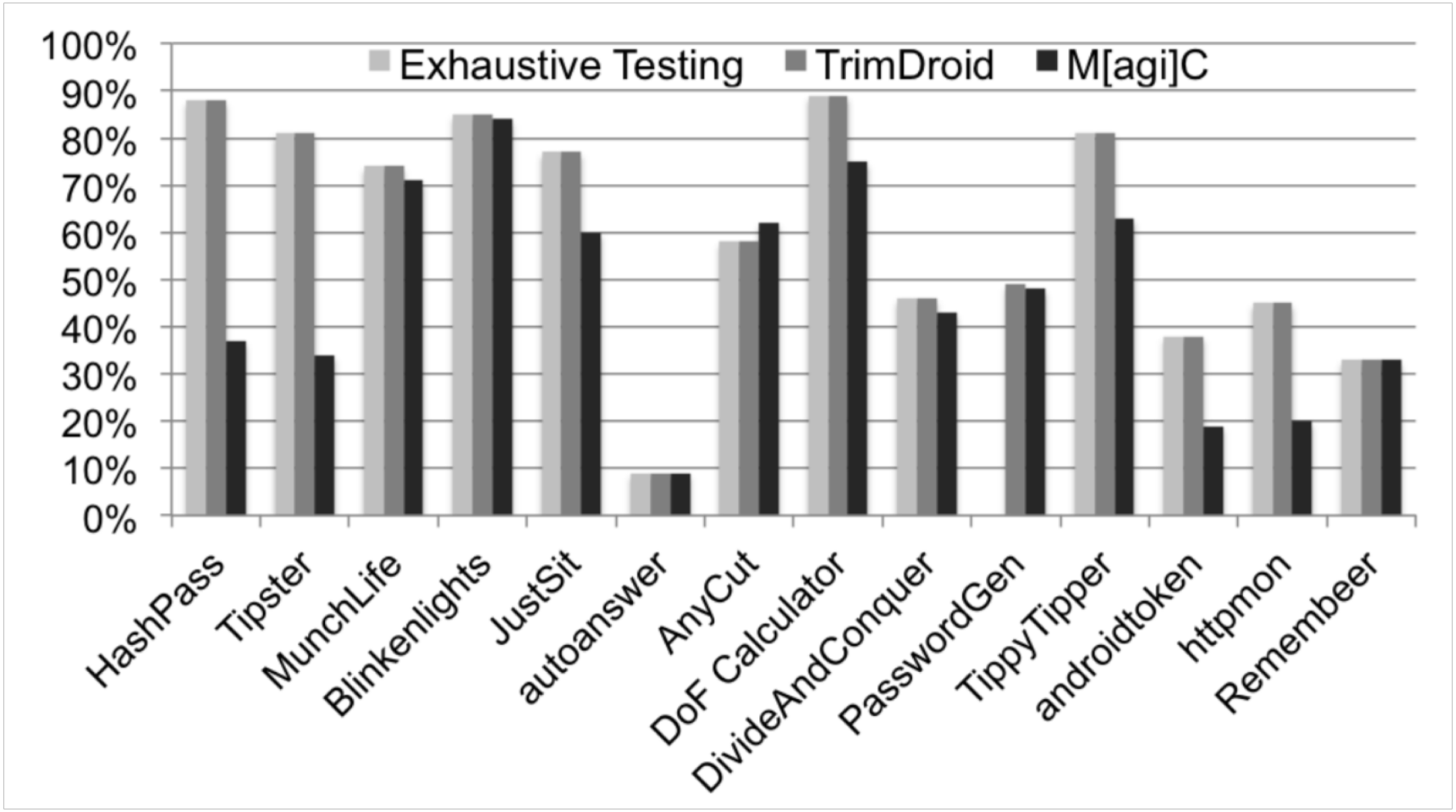
#	$ST_{a_0 \xrightarrow{e_1} a_2 \xrightarrow{e_3} a_2}$
1	$\langle\{\{Rome, Euro, 100\}, ItemizedReport\}, \{Reset\}\rangle$
2	$\langle\{\{Rome, Dollar, 100\}, ItemizedReport\}, \{Reset\}\rangle$
3	$\langle\{\{Rome, Pound, 100\}, ItemizedReport\}, \{Reset\}\rangle$
4	$\langle\{\{London, Euro, 100\}, ItemizedReport\}, \{Reset\}\rangle$
5	$\langle\{\{London, Dollar, 100\}, ItemizedReport\}, \{Reset\}\rangle$
.	.
.	.
30	$\langle\{\{Berlin, Pound, 100\}, ItemizedReport\}, \{Reset\}\rangle$

Test cases for the example path

Evaluation: Test Case Reduction

App	LOC	Exhaustive Testing		M[agi]C		TrimDroid		Reduction
		Test Cases	Time (s)	Test Cases	Time (s)	Test Cases	Time (s)	
HashPass	429	128	515	15	50	32	126	75.00%
Tipster	423	36	243	20	44	24	156	33.33%
MunchLife	631	10	84	9	37	8	56	20%
Blinkenlights	851	54	252	5	36	22	112	59.25%
JustSit	849	50	236	10	42	16	74	68%
autoanswer	999	576	5655	17	196	12	118	97.91%
AnyCut	1095	6	38	4	38	6	38	0%
DoF Calculator	1321	1174	7292	107	953	30	373	97.44%
Divide&Conquer	1824	12	85	5	47	4	32	66.66%
PasswordGene	2824	> 48000	–	58	351	418	1263	99.91%
TippyTipper	2953	26	238	30	172	25	225	3.84%
androidtoken	3680	454	13336	19	189	42	686	90.74%
httpMon	4939	42	407	34	235	28	282	33.33%
Remembeer	5915	48	633	24	194	17	320	64.58%

Evaluation: Code Coverage



Evaluation: Cross Comparison

App	TrimDroid		M[agi]C		Monkey		Dynodroid		EvoDroid	
	Coverage	Time(s)	Coverage	Time(s)	Coverage	Time(s)	Coverage	Time(s)	Coverage	Time(s)
HashPass	88%	126	37%	50	40%	41	57%	33917	57%	23472
Tipster	81%	156	34%	44	67%	104	59%	33825	89%	22813
MunchLife	74%	56	71%	37	49%	75	54%	31421	78%	20965
Blinkenlights	85%	112	84%	36	49%	49	81%	25278	63%	22418
JustSit	77%	74	60%	42	35%	163	53%	41252	84%	20391
autoanswer	9%	118	9%	196	6%	43	10%	59672	8%	21883
AnyCut	58%	38	62%	38	6%	71	66%	21757	84%	19735
DoF Calculator	89%	373	75%	953	43%	70	-	-	36%	20713
DivideAndConquer	46%	32	43%	37	51%	58	83%	33644	-	-
PasswordGen	49%	1263	48%	351	42%	128	33%	31882	62%	26129
TippyTipper	81%	225	63%	172	42%	106	-	-	82%	23108
androidtoken	38%	686	19%	189	10%	67	11%	57813	29%	19188
httpmon	45%	282	20%	235	4%	46	6%	22563	36%	23403
Remembeer	33%	320	33%	194	27%	46	23%	53013	28%	22517

Comparisons

	TrimDroid	Monkey	Dynodroid
Mechanism for reducing test cases	Detect dependencies	None	Find relevant events in each app state
Source Code Needed?	No	No	Need to compile with modified SDK
Ease of Use	Only the APK is needed	Come with the developer toolkit	Need to use their modified version of Android SDK
Code Coverage	Same as exhaustive testing	Could miss cases due to randomness	Better than Monkey due to smarter event selection strategy
Dynamically Generated GUI?	Not Supported	Supported	Supported
Event Types	UI events	UI events	UI and System events
Latest Supported Android Version	No claims. Theoretically 8.1	8.1	2.3 (Old!)

Reading Materials

- Aravind Machiry, Rohan Tahiliani, and Mayur Naik. 2013. Dynodroid: an input generation system for Android apps. In ESEC/FSE 2013, 224–234.
- Riyadh Mahmood,Nariman Mirzaei, and Sam Malek. 2014. EvoDroid: segmented evolutionary testing of Android apps. In FSE 2014, 599–609.
- Ke Mao, Mark Harman, and Yue Jia. 2016. Sapienz: multi-objective automated testing for Android applications. In ISSTA 2016, 94–105.
- Nariman Mirzaei, Joshua Garcia, Hamid Bagheri, Alireza Sadeghi, and Sam Malek. 2016. Reducing Combinatorics in GUI Testing of Android Applications. In ICSE 2016, 559–570.
- Wontae Choi, George C. Necula, and Koushik Sen. 2013. Guided GUI testing of Android apps with minimal restart and approximate learning. In OOPSLA 2013, 623–640.
- Tanzirul Azim and Iulian Neamtiu. 2013. Targeted and depth-first exploration for systematic testing of Android apps. In OOPSLA 2013, 641–660.
- Ting Su, Guozhu Meng, Yuting Chen, Ke Wu, Weiming Yang, Yao Yao, Geguang Pu, Yang Liu, and Zhendong Su. 2013. Guided, Stochastic Model-Based GUI Testing of Android Apps. In ESEC/FSE 2017, 245–256.

Q&A?

Bihuan Chen, Pre-Tenure Assoc. Prof.

bhchen@fudan.edu.cn

https://chenbihuan.github.io