

HANDS ON WITH CRASHSCOPE: AN AUTOMATED ANDROID TESTING TOOL

Kevin Moran,
Assistant Professor
George Mason University





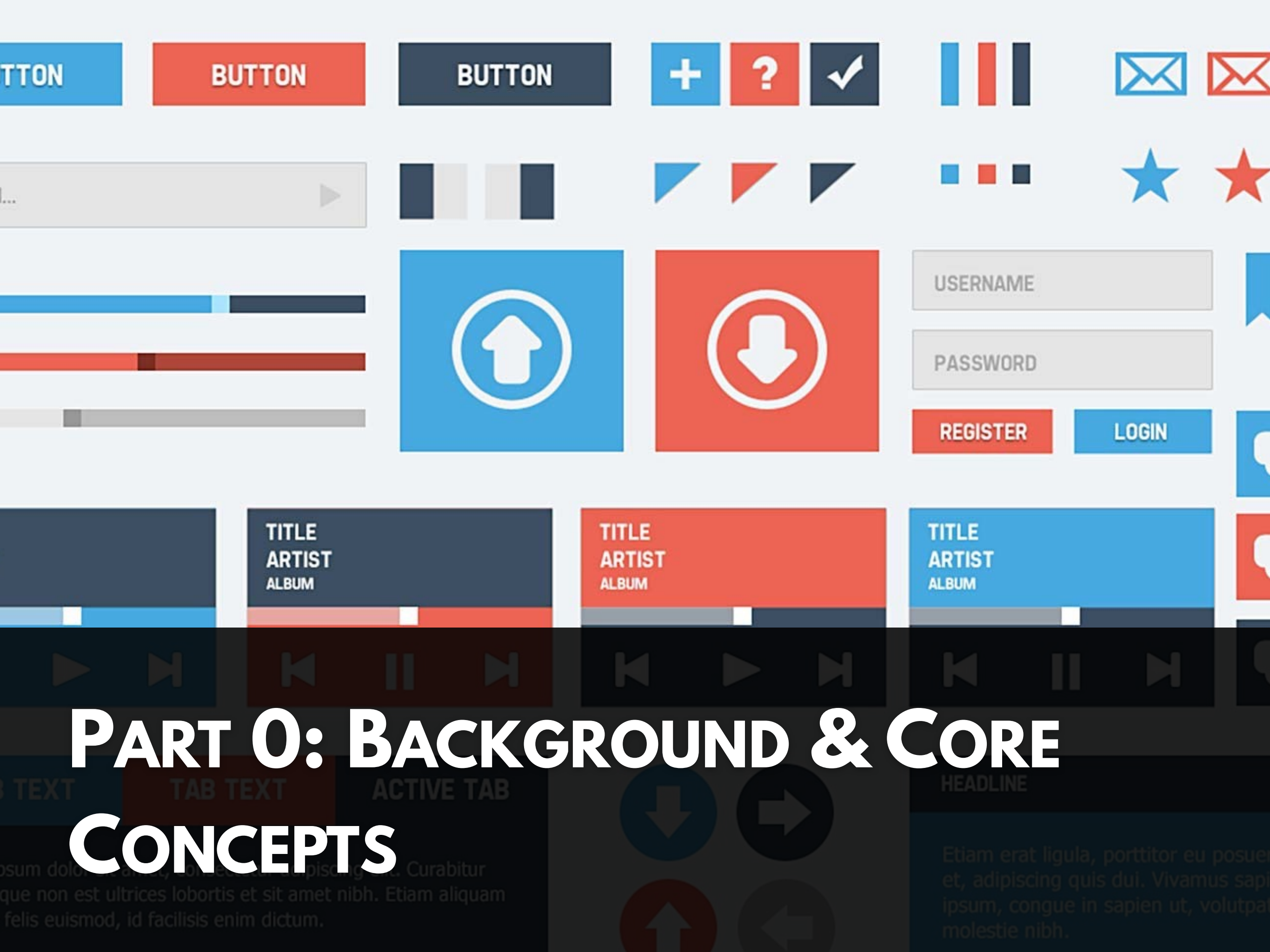
Kevin Moran
Assistant Professor
George Mason University
Fairfax, VA, USA
kpmoran@gmu.edu
<https://www.kpmoran.com>

PART 0: BACKGROUND AND CORE CONCEPTS

PART 1: CURRENT RESEARCH & FUTURE WORK

PART 2: AN OVERVIEW OF CRASHSCOPE

PART 4: HANDS-ON SESSION WITH CRASHSCOPE



PART 0: BACKGROUND & CORE CONCEPTS

The Importance of GUI Testing

- Several different types of testing are important for ensuring software quality:

The Importance of GUI Testing

- Several different types of testing are important for ensuring software quality:

Unit Testing

Performance Testing

Integration Testing

Compatibility Testing

Regression Testing

The Importance of GUI Testing

- For Mobile, GUI-Based Testing subsumes many other types of testing
- GUI-Testing is typically expensive, and test scripts are difficult to maintain
- There is a clear opportunity for automation to Improve development workflows

GUI Testing: The Main Idea



UI Events



GUI Testing: The Main Idea

Oracle

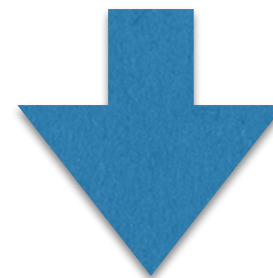
UI Events



GUI Testing: The Main Idea

Oracle

UI Events

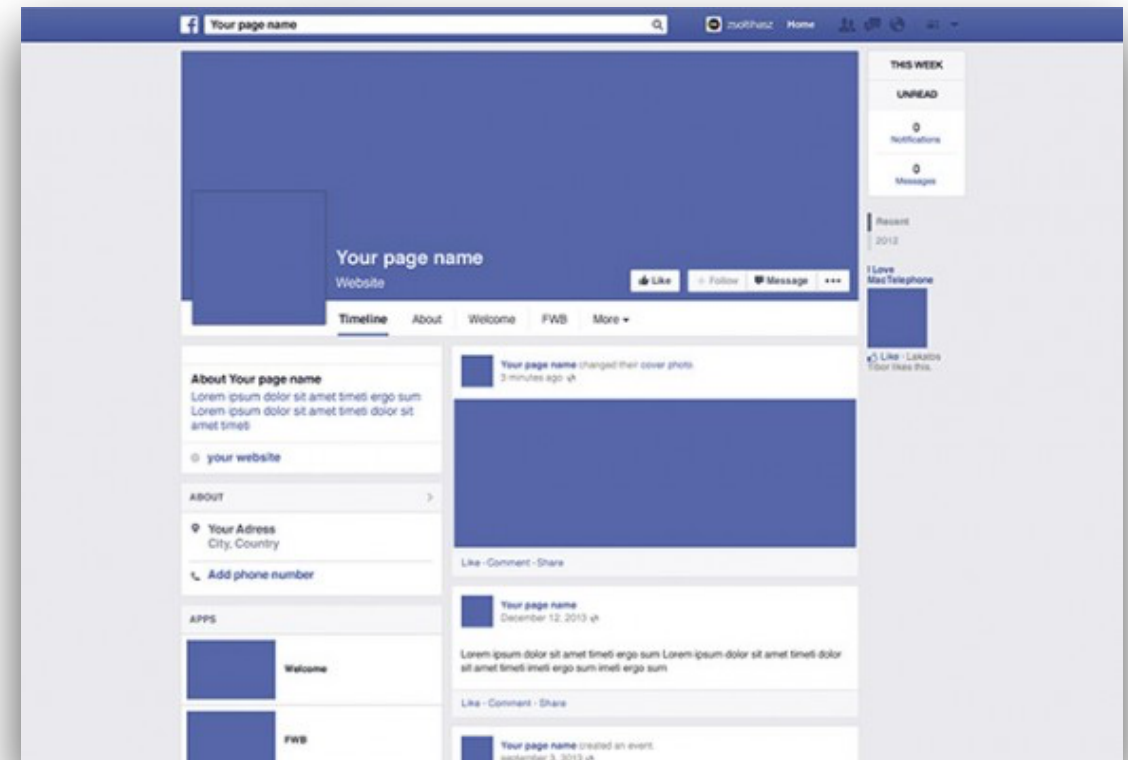
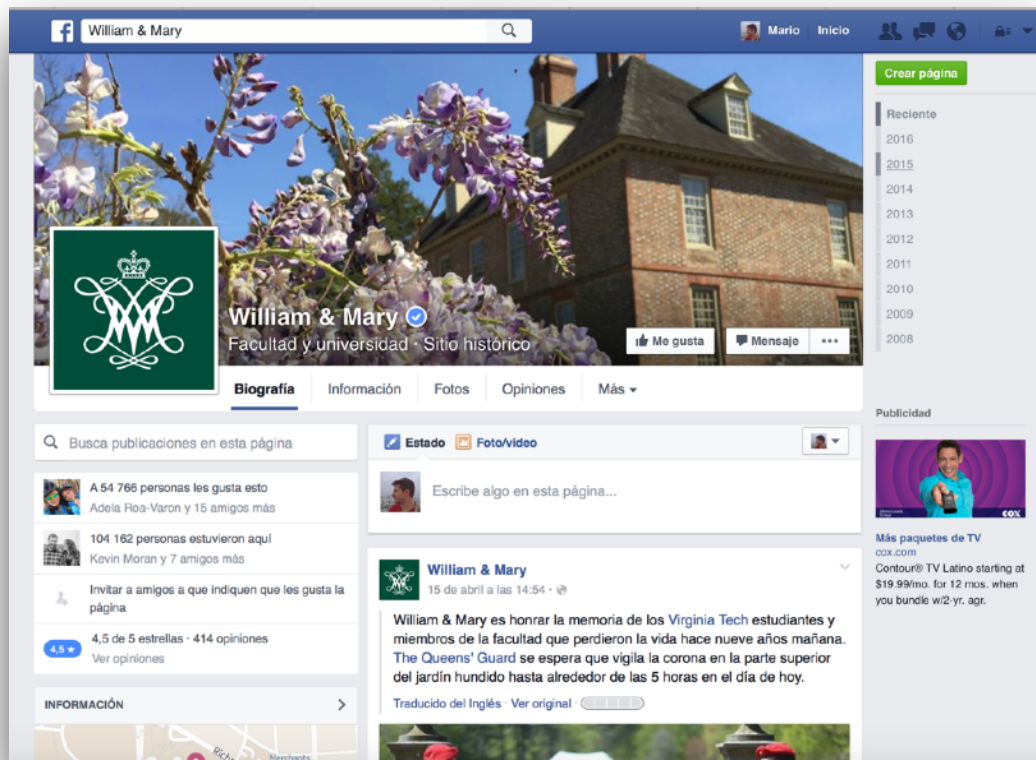


Output, layout, exceptions,
presentation logic, quality attributes, ...

GUI Testing: Core Concepts

Oracle

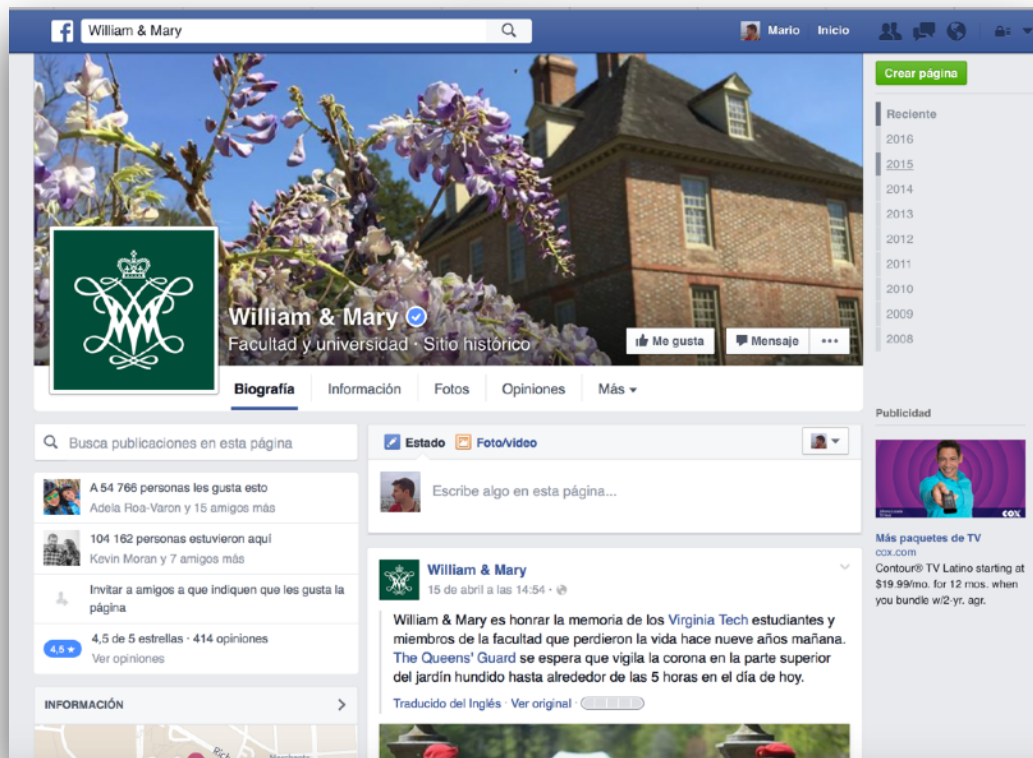
Test
result



GUI Testing: Core Concepts

Oracle

Test
result



500
Unexpected Error :(

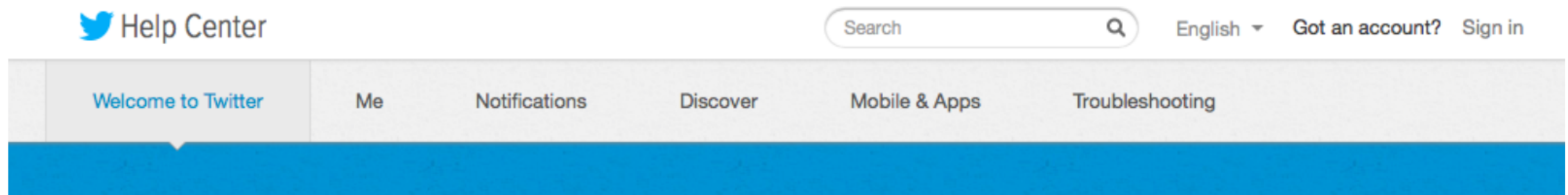
An error occurred and your request couldn't be completed. Please try again.



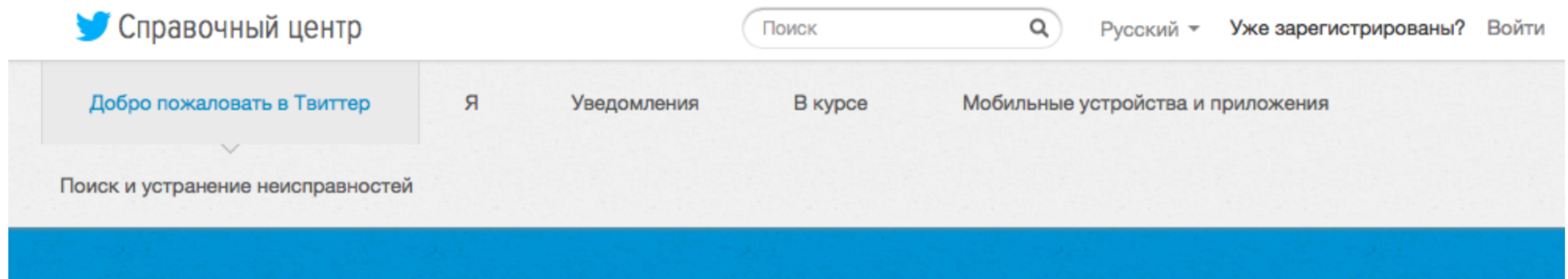
Sorry...
It's not you.
It's us.

We're experiencing an internal server problem.
Please try again later or contact support@fitbit.com

GUI Testing: Example



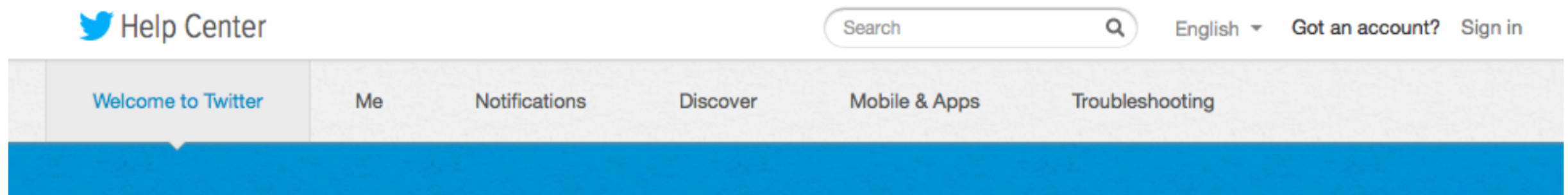
(a) English version



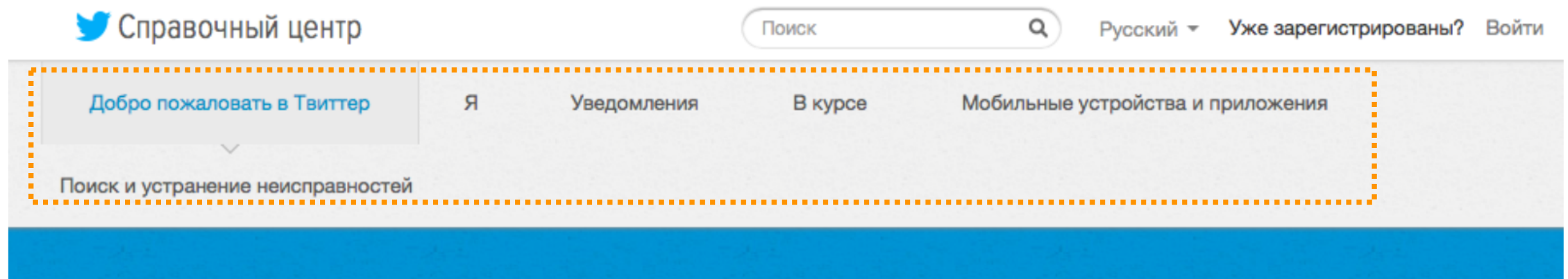
(b) Russian version

Detecting and Localizing Internationalization Presentation Failures in Web Applications. Abdulmajeed Alameer, Sonal Mahajan, William G.J. Halfond. In Proceeding of the 9th IEEE International Conference on Software Testing, Verification, and Validation (ICST). April 2016.

GUI Testing: Example

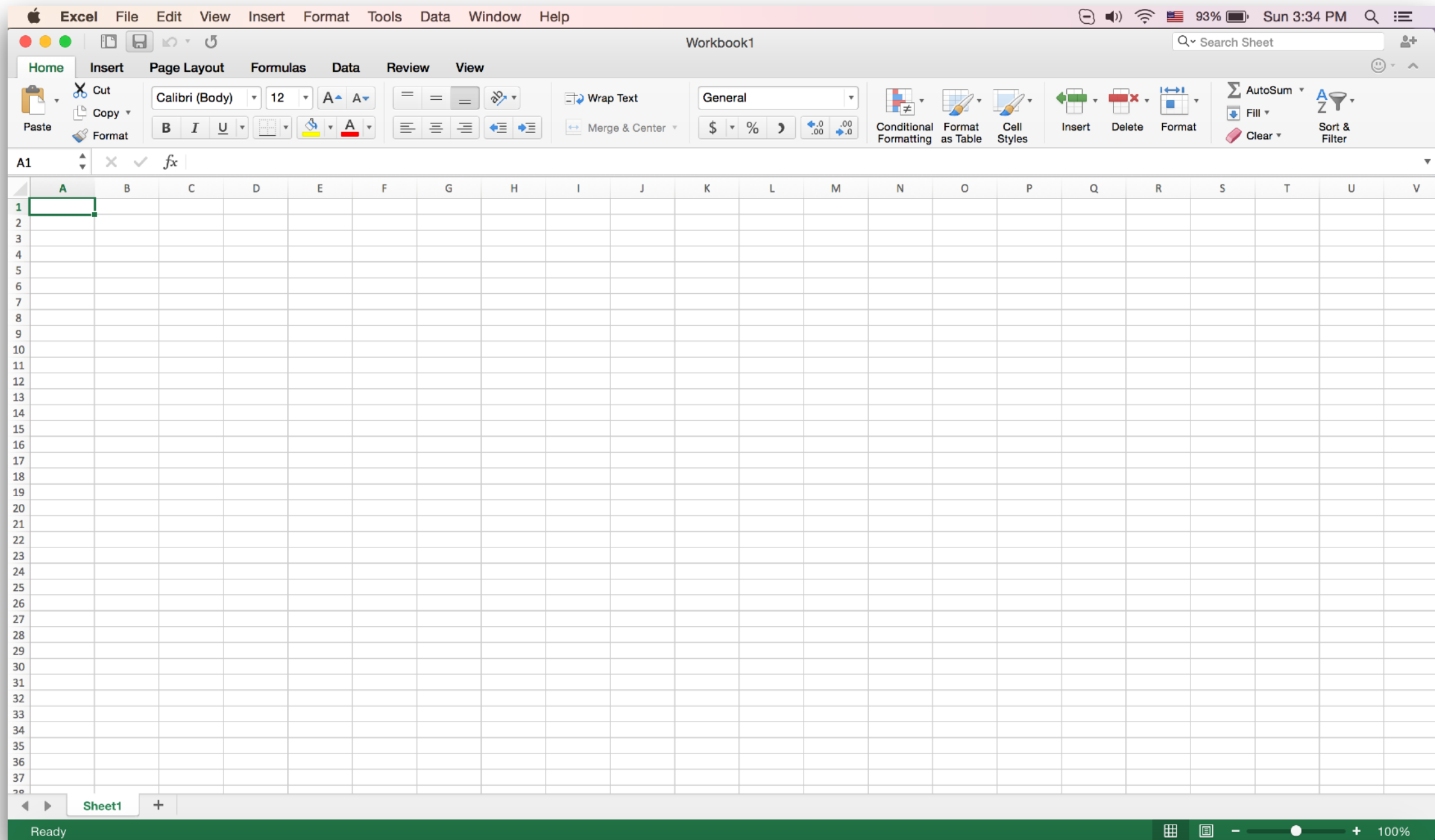


(a) English version

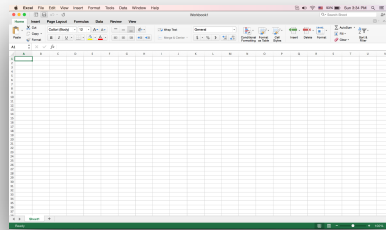


(b) Russian version

Detecting and Localizing Internationalization Presentation Failures in Web Applications. Abdulmajeed Alameer, Sonal Mahajan, William G.J. Halfond. In Proceeding of the 9th IEEE International Conference on Software Testing, Verification, and Validation (ICST). April 2016.



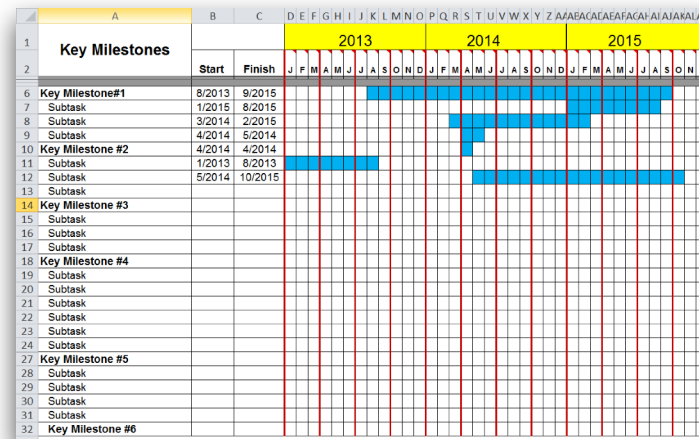
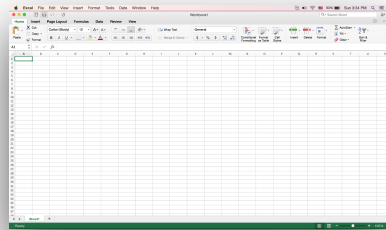
GUI Testing (Challenges)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM		
1	Key Milestones			2013												2014												2015													
2		Start	Finish	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D		
6	Key Milestone#1	8/2013	9/2015																																						
7	Subtask	1/2015	8/2015																																						
8	Subtask	3/2014	2/2015																																						
9	Subtask	4/2014	5/2014																																						
10	Key Milestone #2	4/2014	4/2014																																						
11	Subtask	1/2013	8/2013																																						
12	Subtask	5/2014	10/2015																																						
13	Subtask																																								
14	Key Milestone #3																																								
15	Subtask																																								
16	Subtask																																								
17	Subtask																																								
18	Key Milestone #4																																								
19	Subtask																																								
20	Subtask																																								
21	Subtask																																								
22	Subtask																																								
23	Subtask																																								
24	Subtask																																								
27	Key Milestone #5																																								
28	Subtask																																								
29	Subtask																																								
30	Subtask																																								
31	Subtask																																								
32	Key Milestone #6																																								

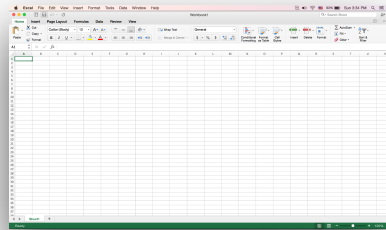
Inputs: combinatorial explosion

GUI Testing (Challenges)



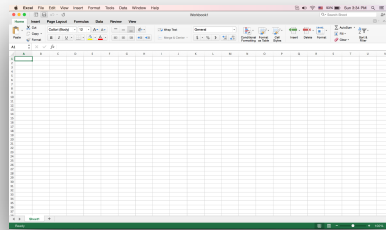
Inputs: combinatorial explosion

GUI Testing (Challenges)

[illegible]

Internationalization

GUI Testing (Challenges)



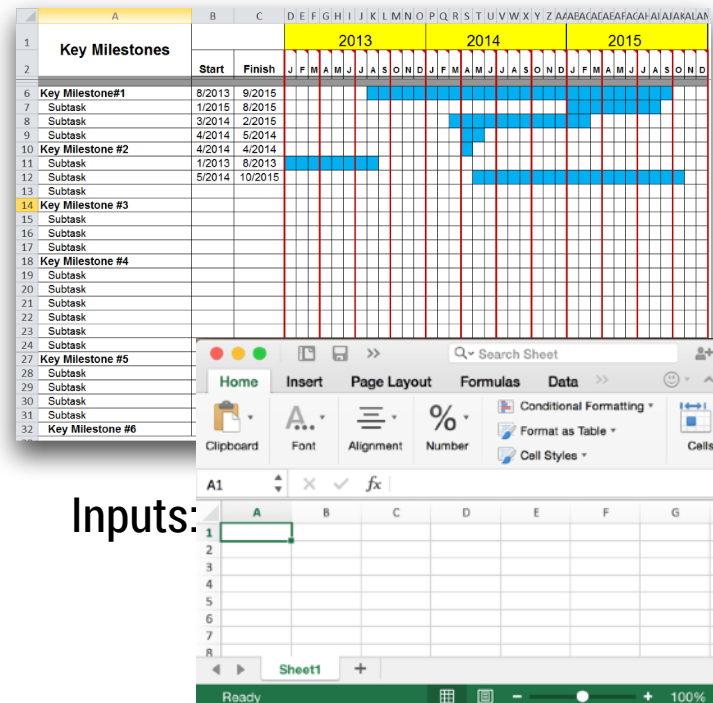
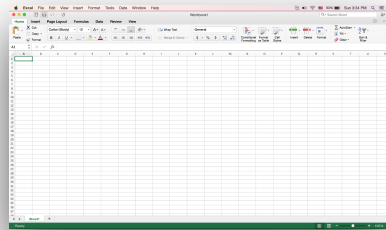
	A	B	C	D E F G H I J K L M N O P Q R S T U V W X Y Z AA AB AC AD AE AF AG AH AI AJ AK AL																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
1	Key Milestones			2013													2014													2015																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
2		Start	Finish	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
6	Key Milestone #1	8/2013	9/2015																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										</

Inputs: combinatorial explosion

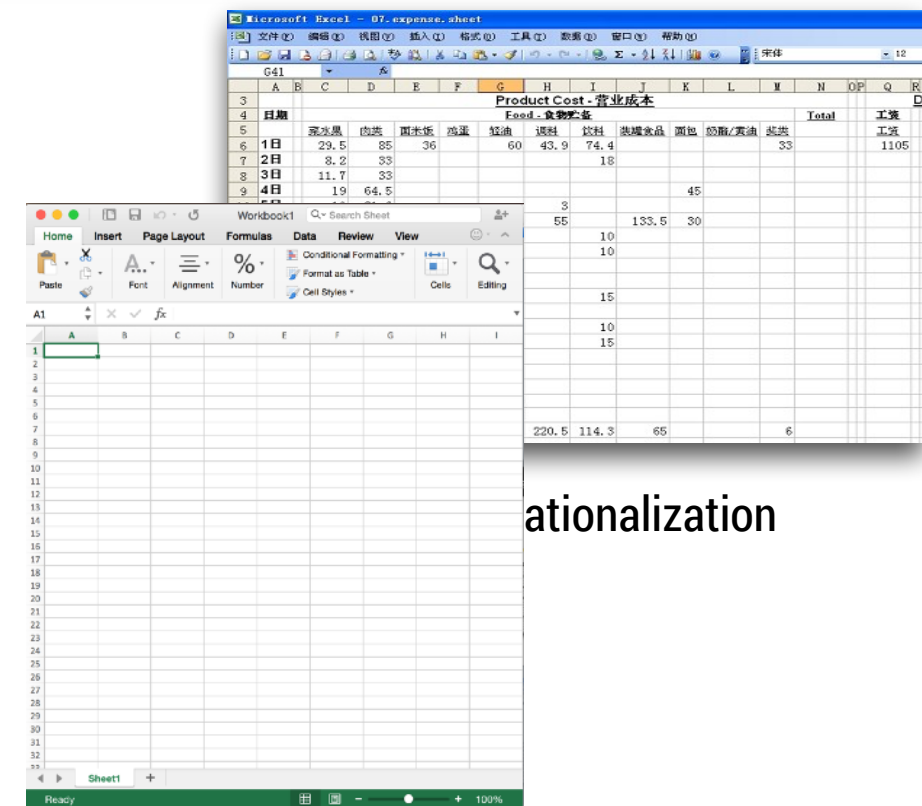
Microsoft Excel - D7.expense.sheet																	
文件(F) 编辑(E) 视图(V) 插入(I) 格式(O) 工具(T) 数据(D) 窗口(W) 帮助(H)																	
G41 12																	
3	A	B	C	D	E	F	G	H	I	J	K	L	M	N	OP	Q	R
4	Product Cost - 营业成本																
5	日期						Feed - 食物喂养								Total		工费
6		深冰墨	肉类	蔬菜饭	鸡蛋		奶油	调料	饮料	洗罐食品	面包	脂肪/黄油	豆类				
7	1日	29.5	85	36			60	43.9	74.4	18				33			1105
8	2日	8.2	33														
9	3日	11.7	33														
10	4日	19	64.5										45				
11	5日	10	21.6					3									
12	6日	5	37		17			55		133.5	30						
13	7日	21.6	33	56					10								
14	8日	10	24.8						10								
15	9日	18			12												
16	10日		22														
17	11日	16.6	20						15								
18	12日		24														
19	13日	21.8	23	35					10								
20	14日	1							15								
21	15日	9	22		12												
22	16日																
23	17日	14.8	20														
24	18日	10	23				25										
25	19日						25										
26	20日	2	25	54	12.8			220.5	114.3	65			6				

Internationalization

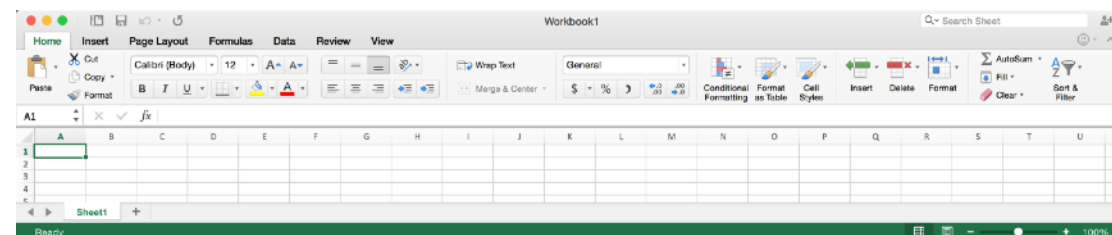
GUI Testing (Challenges)



Inputs:

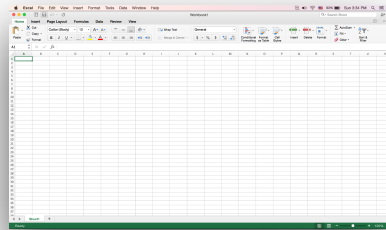


ationalization



Responsive design

GUI Testing (Challenges)

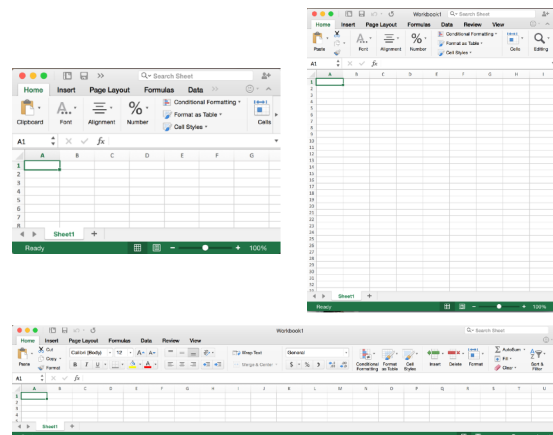


	A	B	C	D E F G H I J K L M N O P Q R S T U V W X Y Z AA AB AC AD AE AF AG AH AI AJ AK AL																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
1	Key Milestones			2013													2014													2015																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
2		Start	Finish	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
6	Key Milestone #1	8/2013	9/2015																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										

Inputs: combinatorial explosion

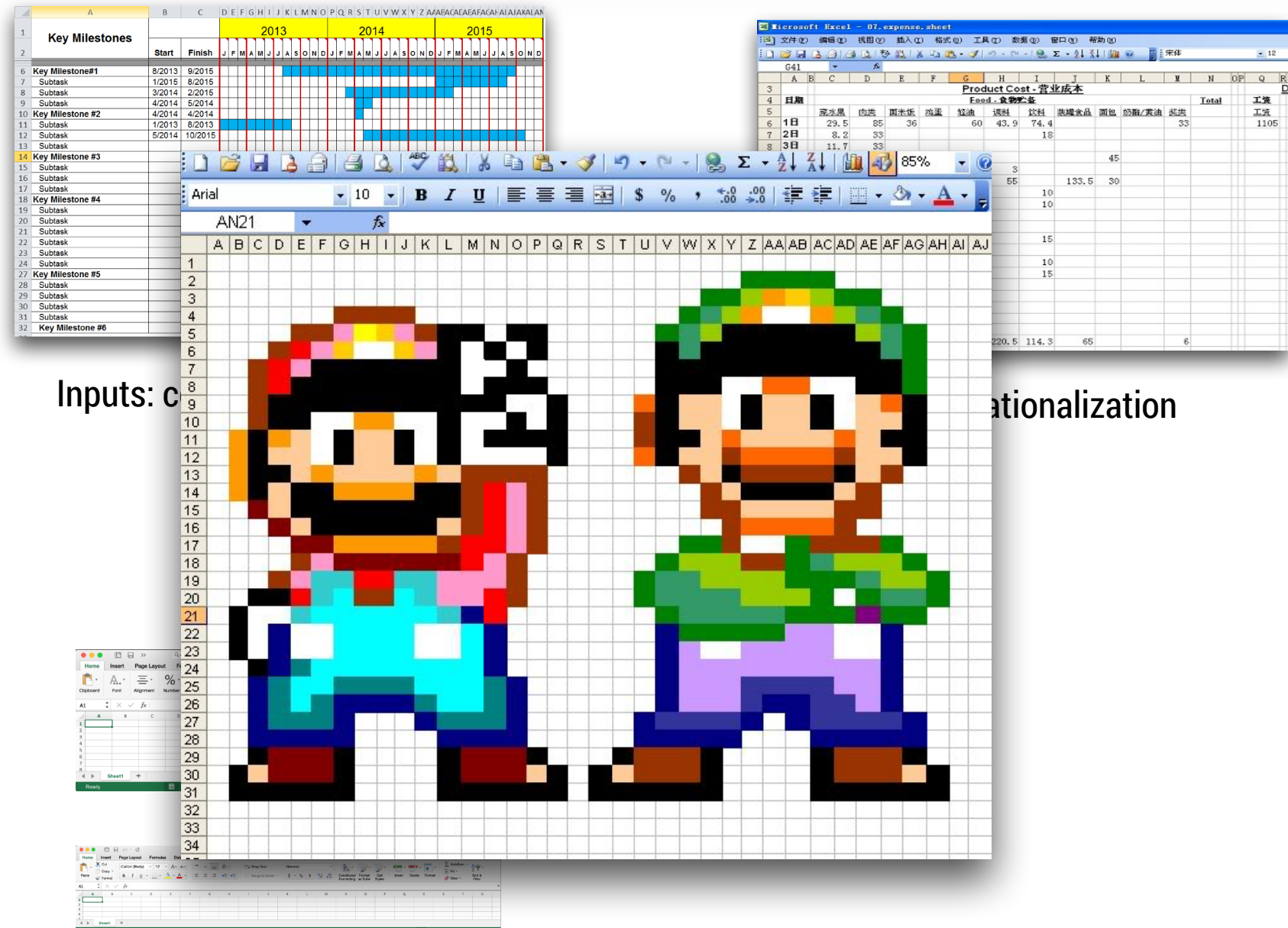
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
3																		
4																		
5																		
6	日期	鸡次星	肉类	蛋类	鸡蛋	鱼油	饲料	饲料	装罐食品	面包	脂肪/黄油	豆类	Total				工资	
7	1日	29.5	85	36		60	43.9		74.4				33				1105	
8	2日	8.2	33					18										
9	3日	11.7	33															
10	4日	19	64.5								45							
11	5日	10	21.6				3											
12	6日	5	37		17		55		133.5	30								
13	7日	21.6	33	56				10										
14	8日	10	24.8					10										
15	9日	18			12													
16	10日		22															
17	11日	16.6	20					15										
18	12日		24															
19	13日	21.8	23	35				10										
20	14日	1						15										
21	15日	9	22		12													
22	16日																	
23	17日	14.8	20															
24	18日	10	23			25												
25	19日					25												
26	20日	2	25	54	12.8		220.5	114.3	65				6					

Internationalization



Responsive design

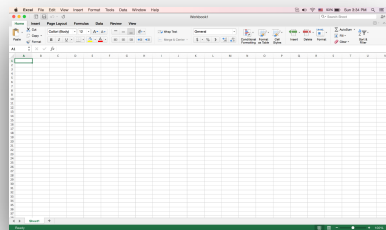
GUI Testing (Challenges)



Unexpected usage scenarios

Responsive design

GUI Testing (Challenges)

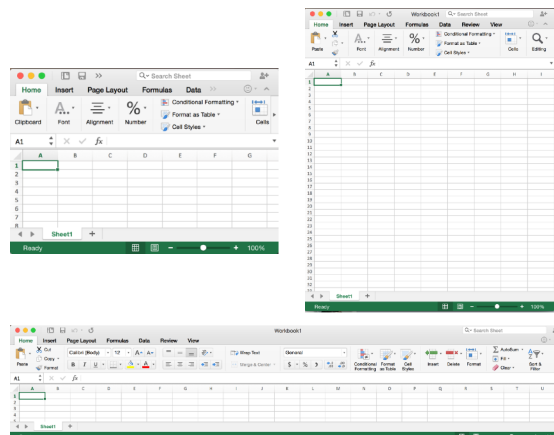


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF	CG	CH	CI	CJ	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT	CU	CV	CW	CX	CY	CZ	DA	DB	DC	DD	DE	DF	DG	DH	DI	DJ	DK	DL	DM	DN	DO	DP	DQ	DR	DS	DT	DU	DV	DW	DX	DY	DZ	EA	EB	EC	ED	EE	EF	EG	EH	EI	EJ	EK	EL	EM	EN	EO	EP	EQ	ER	ES	ET	EU	EV	EW	EX	EY	EZ	FA	FB	FC	FD	FE	FF	FG	FH	FI	FJ	FK	FL	FM	FN	FO	FP	FQ	FR	FS	FT	FU	FV	FW	FX	FY	FZ	GA	GB	GC	GD	GE	GF	GG	GH	GI	GJ	GK	GL	GM	GN	GO	GP	GQ	GR	GS	GT	GU	GV	GW	GX	GY	GZ	HA	HB	HC	HD	HE	HF	HG	HH	HI	HJ	HK	HL	HM	HN	HO	HP	HQ	HR	HS	HT	HU	HV	HW	HX	HY	HZ	IA	IB	IC	ID	IE	IF	IG	IH	II	IJ	IK	IL	IM	IN	IO	IP	IQ	IR	IS	IT	IU	IV	IW	IX	IY	IZ	JA	JB	JC	JD	JE	JF	JG	JH	JI	IJ	JK	KL	KM	KN	KO	KP	KQ	KR	KS	KT	KU	KV	KW	KX	KY	KZ	LA	LB	LC	LD	LE	LF	LG	LH	LI	LJ	LK	LL	LM	LN	LO	LP	LQ	LR	LS	LT	LU	LV	LW	LX	LY	LZ	MA	MB	MC	MD	ME	MF	MG	MH	MI	MJ	MK	ML	MM	MN	MO	MP	MQ	MR	MS	MT	MU	MV	MW	MX	MY	MZ	NA	NB	NC	ND	NE	NF	NG	NH	NI	NJ	NK	NL	NM	NN	NO	NP	NQ	NR	NS	NT	NU	NV	NW	NX	NY	NZ	OA	OB	OC	OD	OE	OF	OG	OH	OI	OJ	OK	OL	OM	ON	OO	OP	OQ	OR	OS	OT	OU	OV	OW	OX	OY	OZ	PA	PB	PC	PD	PE	PF	PG	PH	PI	PJ	PK	PL	PM	PN	PO	PP	PQ	PR	PS	PT	PU	PV	PW	PX	PY	PZ	QA	QB	QC	QD	QE	QF	QG	QH	QI	QJ	QK	QL	QM	QN	QO	QP	QQ	QR	QS	QT	QU	QV	QW	QX	QY	QZ	RA	RB	RC	RD	RE	RF	RG	RH	RI	RJ	RK	RL	RM	RN	RO	RP	RQ	RR	RS	RT	RU	RV	RW	RX	RY	RZ	SA	SB	SC	SD	SE	SF	SG	SH	SI	SJ	SK	SL	SM	SN	SO	SP	SQ	SR	SS	ST	SU	SV	SW	SX	SY	SZ	TA	TB	TC	TD	TE	TF	TG	TH	TI	TJ	TK	TL	TM	TN	TO	TP	TQ	TR	TS	TT	TU	TV	TW	TX	TY	TZ	UA	UB	UC	UD	UE	UF	UG	UH	UI	UJ	UK	UL	UM	UN	UO	UP	UQ	UR	US	UT	UU	UV	UW	UX	UY	UZ	VA	VB	VC	VD	VE	VF	VG	VH	VI	VJ	VK	VL	VM	VN	VO	VP	VQ	VR	VS	VT	VU	VV	VW	VX	VY	VZ	WA	WB	WC	WD	WE	WF	WG	WH	WI	WJ	WK	WL	WM	WN	WO	WP	WQ	WR	WS	WT	WU	WV	WW	WX	WY	WZ	XA	XB	XC	XD	XE	XF	XG	XH	XI	XJ	XK	XL	XM	XN	XO	XP	XQ	XR	XS	XT	XU	XV	XW	XX	XY	XZ	YA	YB	YC	YD	YE	YF	YG	YH	YI	YJ	YK	YL	YM	YN	YO	YP	YQ	YR	YS	YT	YU	YV	YW	YX	YY	YZ	ZA	ZB	ZC	ZD	ZE	ZF	ZG	ZH	ZI	ZJ	ZK	ZL	ZM	ZN	ZO	ZP	ZQ	ZR	ZS	ZT	ZU	ZV	ZW	ZX	ZY	ZZ	AA	AB	AC																							
1	Key Milestones			2013												2014												2015																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
2		Start	Finish	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M

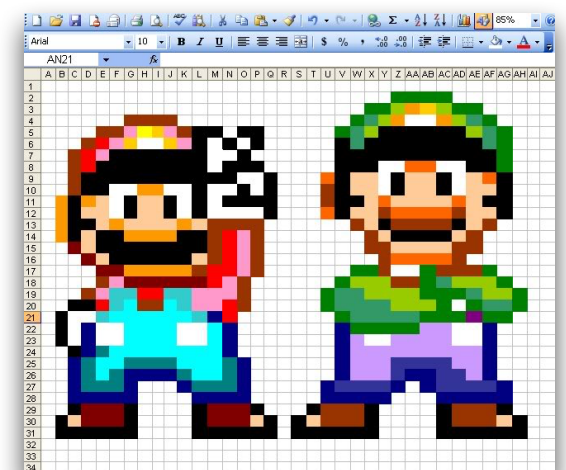
Inputs: combinatorial explosion

Microsoft Excel - 07.expense.sheet																
文件(F) 编辑(E) 视图(V) 插入(I) 格式(O) 工具(T) 数据(D) 窗口(W) 帮助(H)																
G41 + X																
A	B	C	D	E	F	G	H	I	J	K	L	M	N	OP	Q	R
3	Product Cost - 营业成本															
4	Feed - 食物准备															
5	日期	鸡头恩	肉类	玉米饭	鸡蛋	鱼油	饲料	饮料	类罐食品	面包	脂肪/黄油	蔬菜	Total		工费	
6	1日	29.5	85	36		60	43.9	74.4					33		1105	
7	2日	8.2	33					18								
8	3日	11.7	33													
9	4日	19	64.5								45					
10	5日	10	21.6				3									
11	6日	5	37		17		55		133.5	30						
12	7日	21.6	33	56				10								
13	8日	10	24.8					10								
14	9日	18			12											
15	10日		22													
16	11日	16.6	20					15								
17	12日		24													
18	13日	21.8	23	35				10								
19	14日	1						15								
20	15日	9	22		12											
21	16日															
22	17日	14.8	20													
23	18日	10	23			25										
24	19日					25										
25	20日	2	25	54	12.8	220.5	114.3	65				6				

Internationalization



Responsive design



Unexpected usage scenarios

MONKEY TESTING !!



~~MONKEY TESTING !!~~ AUTOMATED TESTING !!

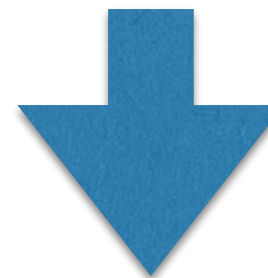
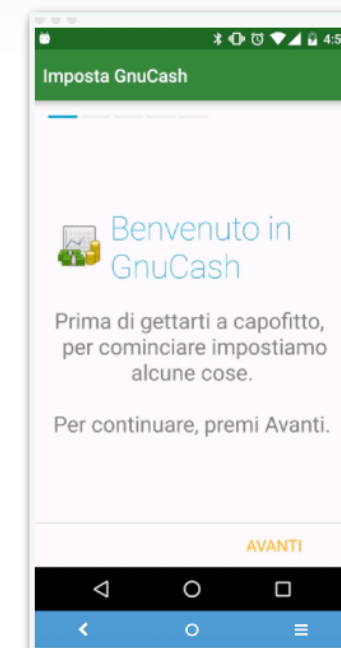


Automated GUI Testing



Monkey

UI Events



Output, layout, exceptions, presentation logic, quality attributes, ...



ANDROID GUI TESTING

Unique Challenges in Mobile Development



Thousands of apps are released and updated every day on the online store




2.8M apps - Google Play


65B downloads - Google Play

25 releases (Android) since 2008


Large volume of crowdsourced requirements and ratings




Benjamin Sanchez September 12, 2015
★★★★★
Keeps skipping my music listening to a song, and it j itself. Is there something I on the settings for that to




Ian Sammut September 14, 2015
★★★★★
Some help maybe? I really like Spotify but recently i cant change song from lock screen or change songs from my Bluetooth car stereo because of this




eva yadmeiri olivares martinez September 12, 2015
★★★★★
Wouldve given 5 but its taking too much internal memory I will give it a 5 if they would allow for app to h saved




April Senchuk September 13, 2015
★★★★★
What happened to Spotify? Once upon a time when I wanted to hear a specific song, I could do that. Now it starts these playlists that start with stuff I like, but then play random stuff I'd never have chosen. Why can't I just listen to the song I have stuck in my



Bosco Wong September 12, 2015
★★★★★
I can't pay The app is great comes to payment, you o option to pay in credit card by visa amet, but most banks in hong Kong uses Union Pay and EPS, I don't have



Saumitra Dixit September 12, 2015
★★★★★
Not logging in. Says offline. First it started with saying no network connection and said "offline". Reinstalled after trying out the settings. Now it won't even log in.



Albert Rinck September 11, 2015
★★★★★
Doesnt have some music It do have any taylor swift , & not the original " God Help The Outca

Fragmentation at device and OS level



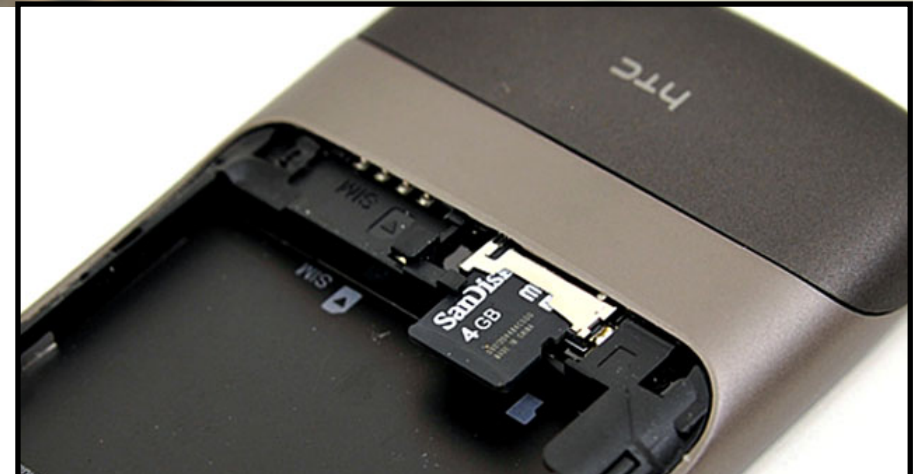
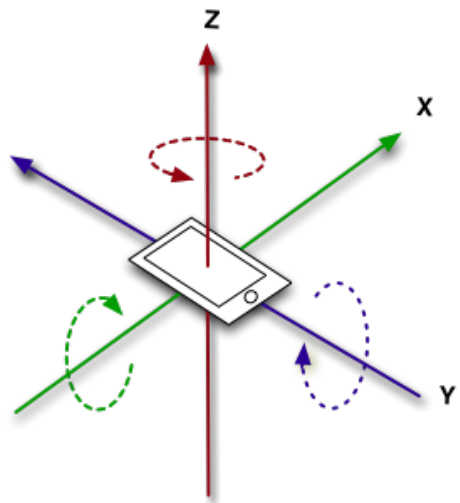
A large, dense crowd of young people, mostly men, is shown at what appears to be a music festival or concert. In the center, a young man is being crowd-surfed, held high above the crowd by several people. He has his mouth open as if shouting or singing. The crowd is filled with many hands raised in the air, some holding up phones to record. The background is a vast sea of people stretching far into the distance. A semi-transparent dark banner is overlaid across the top half of the image, containing the text "Pressure for continuous delivery" in white. The overall atmosphere is one of high energy and excitement.

Pressure for continuous delivery

Manual testing is still preferred



Mobile-specific quality attributes, inputs, and scenarios





PART 1: CURRENT RESEARCH & FUTURE WORK

Overview of Tools & Services

Overview of Tools & Services

- Automation Frameworks & APIs

Overview of Tools & Services

- Automation Frameworks & APIs
- Record & Replay Tools

Overview of Tools & Services

- Automation Frameworks & APIs
- Record & Replay Tools
- Automated Input Generation Tools

Overview of Tools & Services

- Automation Frameworks & APIs
- Record & Replay Tools
- Automated Input Generation Tools
- Bug & Error Reporting

Overview of Tools & Services

- Automation Frameworks & APIs
- Record & Replay Tools
- Automated Input Generation Tools
- Bug & Error Reporting
- Crowdsourced Testing

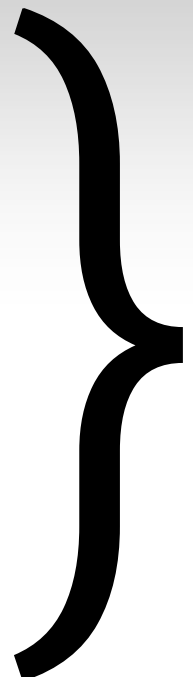
Overview of Tools & Services

- Automation Frameworks & APIs
- Record & Replay Tools
- Automated Input Generation Tools
- Bug & Error Reporting
- Crowdsourced Testing
- Cloud Testing Services

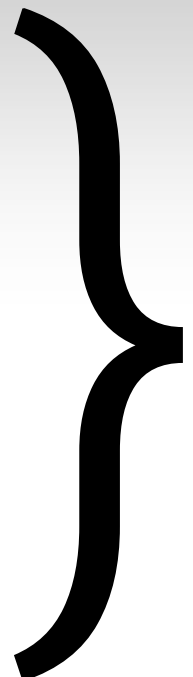
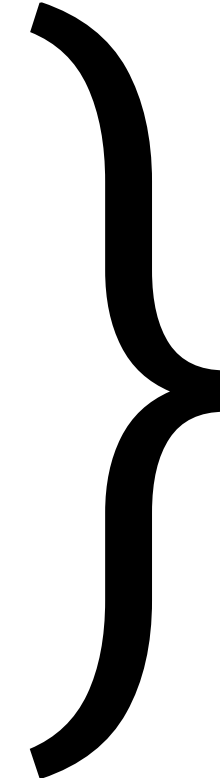
Overview of Tools & Services

- Automation Frameworks & APIs
- Record & Replay Tools
- Automated Input Generation Tools
- Bug & Error Reporting
- Crowdsourced Testing
- Cloud Testing Services
- Device Streaming Tools

Overview of Tools & Services

- Automation Frameworks & APIs
 - Record & Replay Tools
 - Automated Input Generation Tools
 - Bug & Error Reporting
 - Crowdsourced Testing
 - Cloud Testing Services
 - Device Streaming Tools
- 
- Traditional Android Testing
Tools and Approaches

Overview of Tools & Services

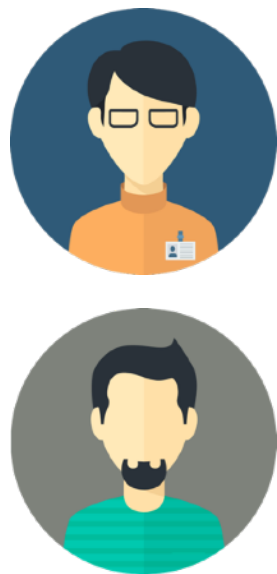
- Automation Frameworks & APIs
 - Record & Replay Tools
 - Automated Input Generation Tools
- 
- Traditional Android Testing
Tools and Approaches
- Bug & Error Reporting
 - Crowdsourced Testing
 - Cloud Testing Services
 - Device Streaming Tools
- 
- Bug Reporting,
Crowdsourcing and Services

ANDROID TESTING TOOLS & APPROACHES



Automation Frameworks/APIs (AF/A)

TESTS



```
@Test
public void autoCompleteTextView_oneSuggestion() {
    // Type "South" to trigger one suggestion.
    onView(withId(R.id.auto_complete_text_view))
        .perform(typeTextIntoFocusedView("South "), closeSoftKeyboard());

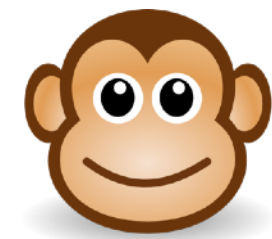
    // Should be displayed
    onView(withText("South China Sea"))
        .inRoot(withDecorView(not(is(mActivity.getWindow().getDecorView()))))
        .check(matches(isDisplayed()));

    // Should not be displayed.
    onView(withText("Southern Ocean"))
        .inRoot(withDecorView(not(is(mActivity.getWindow().getDecorView()))))
        .check(doesNotExist());
}
```

```
@Test
public void checkPreconditions() {
    assertThat(mDevice, notNullValue());
}

@Test
public void testChangeText_sameActivity() {
    // Type text and then press the button.
    mDevice.findObject(By.res(BASIC_SAMPLE_PACKAGE, "editTextUserInput"))
        .setText(STRING_TO_BE_TYPED);
    mDevice.findObject(By.res(BASIC_SAMPLE_PACKAGE, "changeTextBt"))
        .click();

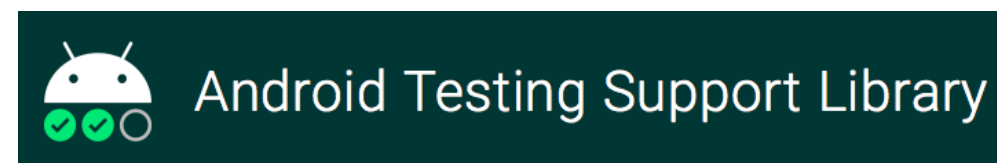
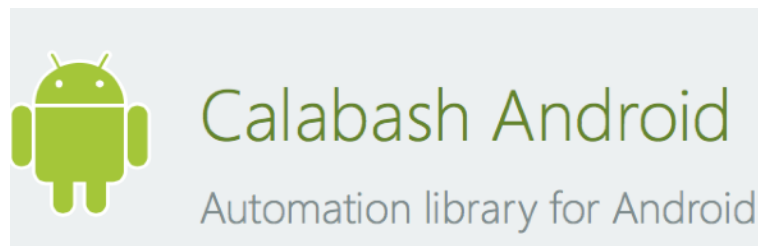
    // Verify the test is displayed in the Ui
    UiObject2 changedText = mDevice
        .wait(Until.findObject(By.res(BASIC_SAMPLE_PACKAGE, "textToBeChanged")),
            500 /* wait 500ms */);
    assertThat(changedText.getText(), is(equalTo(STRING_TO_BE_TYPED)));
}
```



Monkey

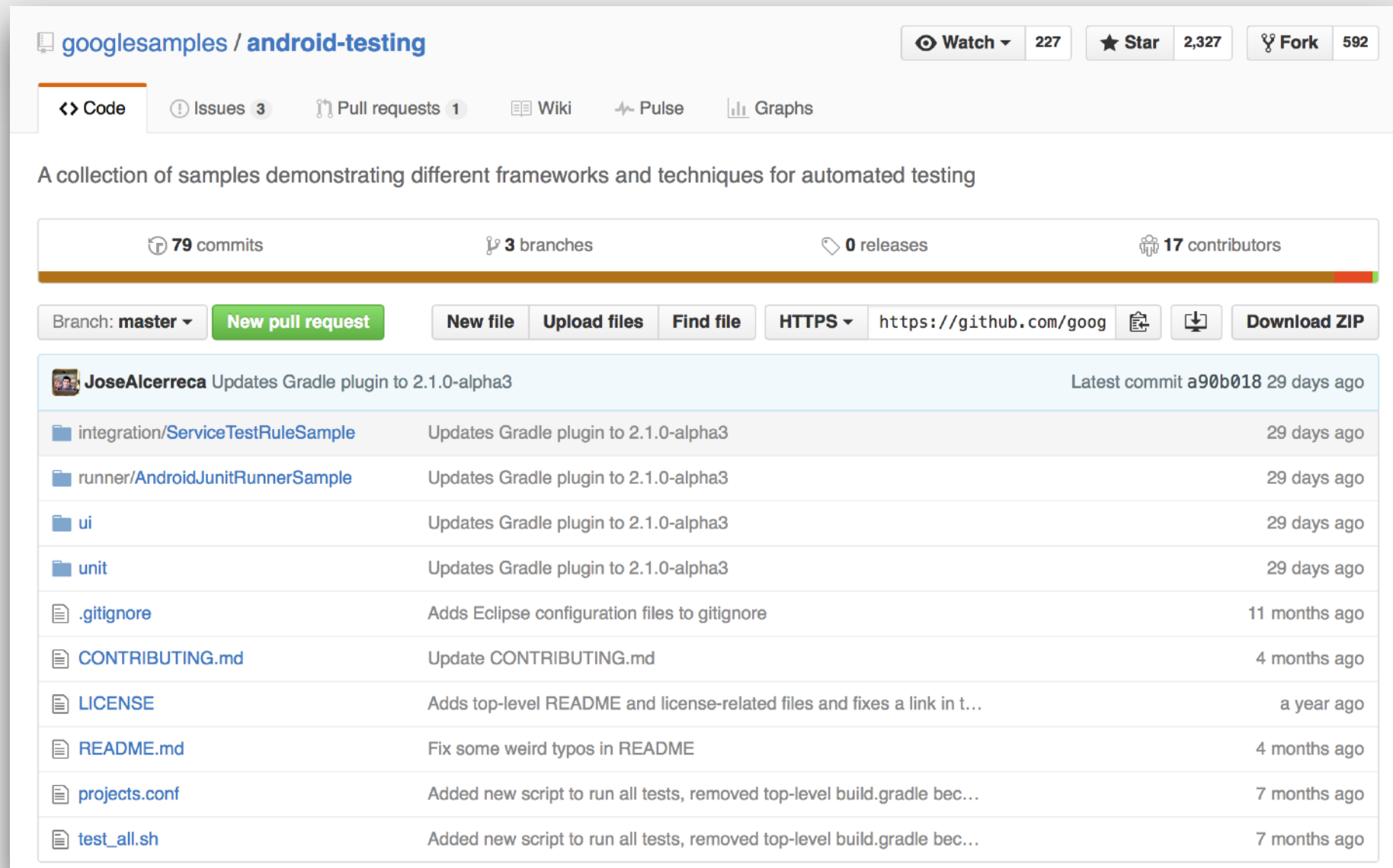
JUnit, Espresso, UI Automator, Robotium

Testing Automation Frameworks/APIs



UI Automator

Testing Automation Frameworks/APIs



googlesamples / android-testing

Watch 227 Star 2,327 Fork 592

Code Issues 3 Pull requests 1 Wiki Pulse Graphs

A collection of samples demonstrating different frameworks and techniques for automated testing

79 commits 3 branches 0 releases 17 contributors

Branch: master New pull request New file Upload files Find file HTTPS https://github.com/goog Download ZIP

JoseAlcerreca Updates Gradle plugin to 2.1.0-alpha3 Latest commit a90b018 29 days ago

integration/ServiceTestRuleSample	Updates Gradle plugin to 2.1.0-alpha3	29 days ago
runner/AndroidJUnitRunnerSample	Updates Gradle plugin to 2.1.0-alpha3	29 days ago
ui	Updates Gradle plugin to 2.1.0-alpha3	29 days ago
unit	Updates Gradle plugin to 2.1.0-alpha3	29 days ago
.gitignore	Adds Eclipse configuration files to gitignore	11 months ago
CONTRIBUTING.md	Update CONTRIBUTING.md	4 months ago
LICENSE	Adds top-level README and license-related files and fixes a link in t...	a year ago
README.md	Fix some weird typos in README	4 months ago
projects.conf	Added new script to run all tests, removed top-level build.gradle bec...	7 months ago
test_all.sh	Added new script to run all tests, removed top-level build.gradle bec...	7 months ago

<https://github.com/googlesamples/android-testing>

Testing Automation Frameworks/APIs

```
@Test
public void changeText_sameActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput))
        .perform(typeText("STRING_TO_BE_TYPED"), closeSoftKeyboard());
    onView(withId(R.id.changeTextBt)).perform(click());

    // Check that the text was changed.
    onView(withId(R.id.textToBeChanged)).check(matches(withText("STRING_TO_BE_TYPED")));
}

@Test
public void changeText_newActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput)).perform(typeText("STRING_TO_BE_TYPED"),
        closeSoftKeyboard());
    onView(withId(R.id.activityChangeTextBtn)).perform(click());

    // This view is in a different Activity, no need to tell Espresso.
    onView(withId(R.id.show_text_view)).check(matches(withText("STRING_TO_BE_TYPED")));
}
```

<https://github.com/googlesamples/android-testing>

Testing Automation Frameworks/APIs

```
@Test
public void changeText_sameActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput))
        .perform(typeText(String.TO_BE_TYPED), closeSoftKeyboard());
    onView(withId(R.id.changeTextBt)).perform(click());

    // Check that the text was changed.
    onView(withId(R.id.textToBeChanged)).check(matches(withText(String.TO_BE_TYPED)));
}

@Test
public void changeText_newActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput)).perform(typeText(String.TO_BE_TYPED),
        closeSoftKeyboard());
    onView(withId(R.id.activityChangeTextBtn)).perform(click());

    // This view is in a different Activity, no need to tell Espresso.
    onView(withId(R.id.show_text_view)).check(matches(withText(String.TO_BE_TYPED)));
}
```

<https://github.com/googlesamples/android-testing>

Testing Automation Frameworks/APIs

```
@Test
public void changeText_sameActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput))
        .perform(typeText(STRING_TO_BE_TYPED), closeSoftKeyboard());
    onView(withId(R.id.changeTextBt)).perform(click());

    // Check that the text was changed.
    onView(withId(R.id.textToBeChanged)).check(matches(withText(STRING_TO_BE_TYPED)));
}

@Test
public void changeText_newActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput)).perform(typeText(STRING_TO_BE_TYPED),
        closeSoftKeyboard());
    onView(withId(R.id.activityChangeTextBtn)).perform(click());

    // This view is in a different Activity, no need to tell Espresso.
    onView(withId(R.id.show_text_view)).check(matches(withText(STRING_TO_BE_TYPED)));
}
```

<https://github.com/googlesamples/android-testing>

Testing Automation Frameworks/APIs

```
@Test
public void changeText_sameActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput))
        .perform(typeText(STRING_TO_BE_TYPED), closeSoftKeyboard());
    onView(withId(R.id.changeTextBt)).perform(click());

    // Check that the text was changed.
    onView(withId(R.id.textToBeChanged)).check(matches(withText(STRING_TO_BE_TYPED)));
}

@Test
public void changeText_newActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput)).perform(typeText(STRING_TO_BE_TYPED),
        closeSoftKeyboard());
    onView(withId(R.id.activityChangeTextBtn)).perform(click());

    // This view is in a different Activity, no need to tell Espresso.
    onView(withId(R.id.show_text_view)).check(matches(withText(STRING_TO_BE_TYPED)));
}
```

<https://github.com/googlesamples/android-testing>

Testing Automation Frameworks/APIs

```
@Test
public void changeText_sameActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput))
        .perform(typeText(StringToBeTyped), closeSoftKeyboard());
    onView(withId(R.id.changeTextBt)).perform(click());

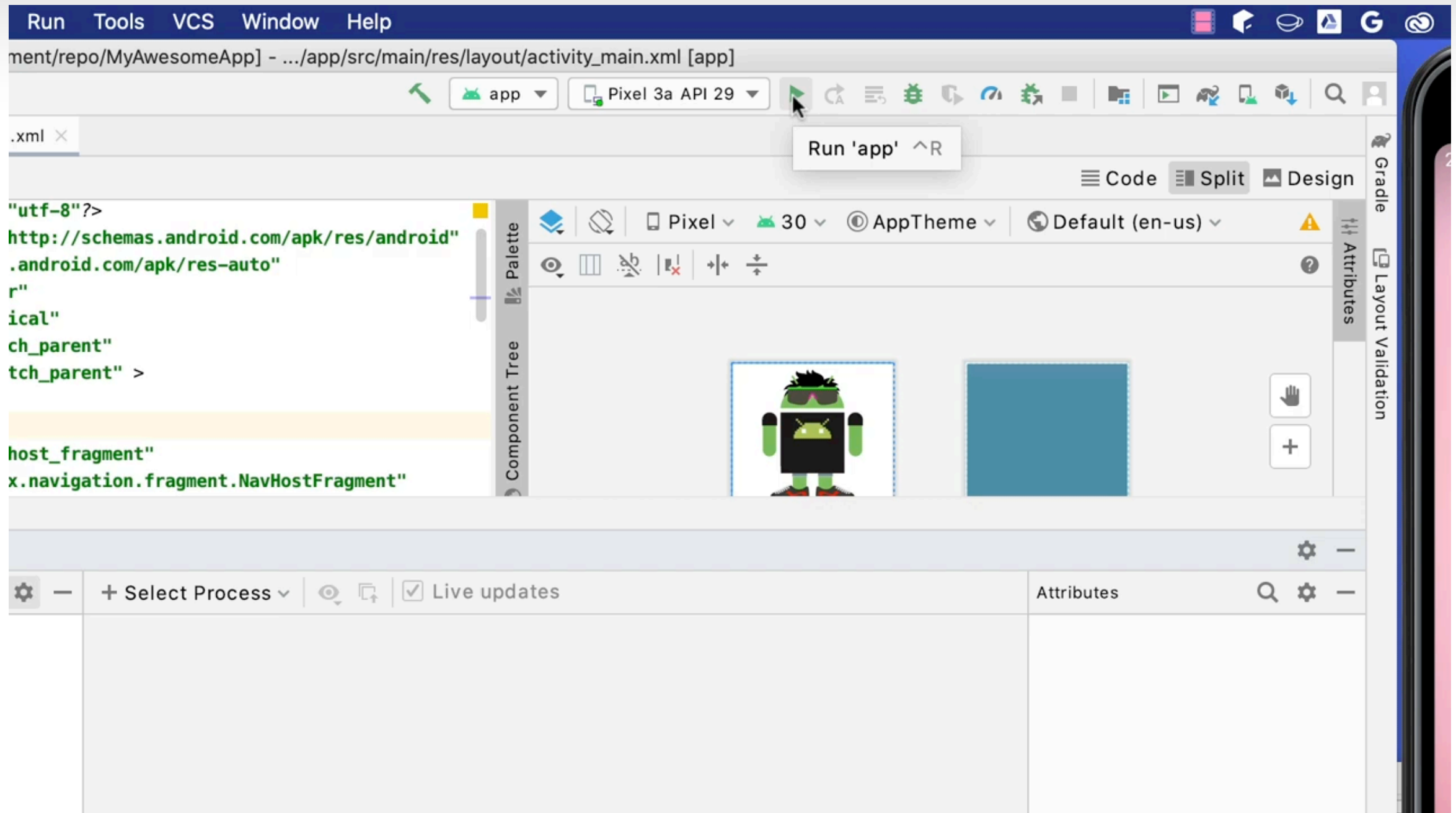
    // Check that the text was changed.
    onView(withId(R.id.textToBeChanged)).check(matches(withText(StringToBeTyped)));
}

@Test
public void changeText_newActivity() {
    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput)).perform(typeText(StringToBeTyped),
        closeSoftKeyboard());
    onView(withId(R.id.activityChangeTextBtn)).perform(click());

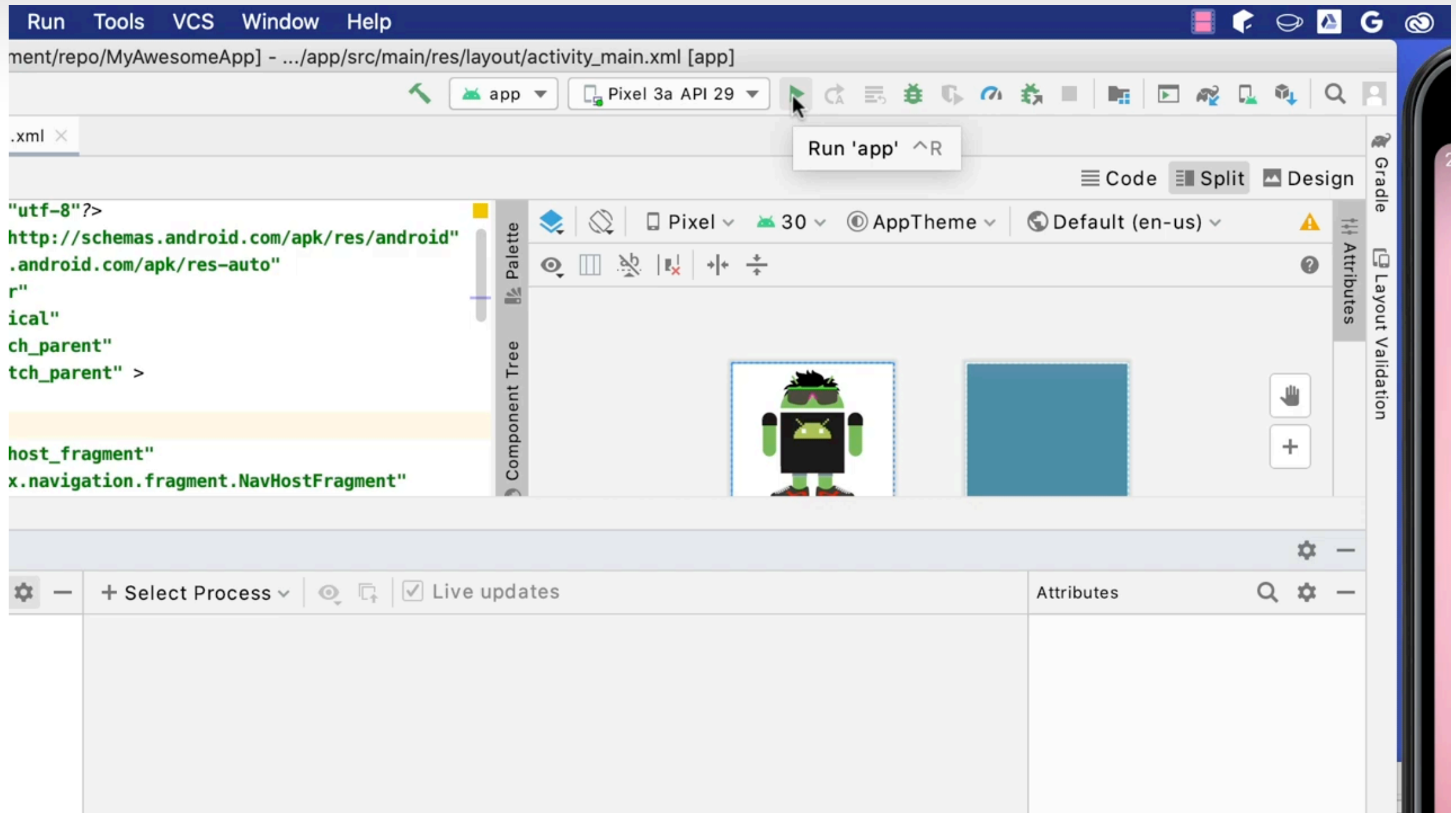
    // This view is in a different Activity, no need to tell Espresso.
    onView(withId(R.id.show_text_view)).check(matches(withText(StringToBeTyped)));
}
```

<https://github.com/googlesamples/android-testing>

Tools: Layout Inspector



Tools: Layout Inspector



Pros and Cons



Automation Frameworks

- ✓ Easy reproduction
- ✓ High level syntax
- ✓ Black box testing



- Learning curve
- User-defined oracles
- Expensive maintenance

Record and Replay (R&R)



UI Events

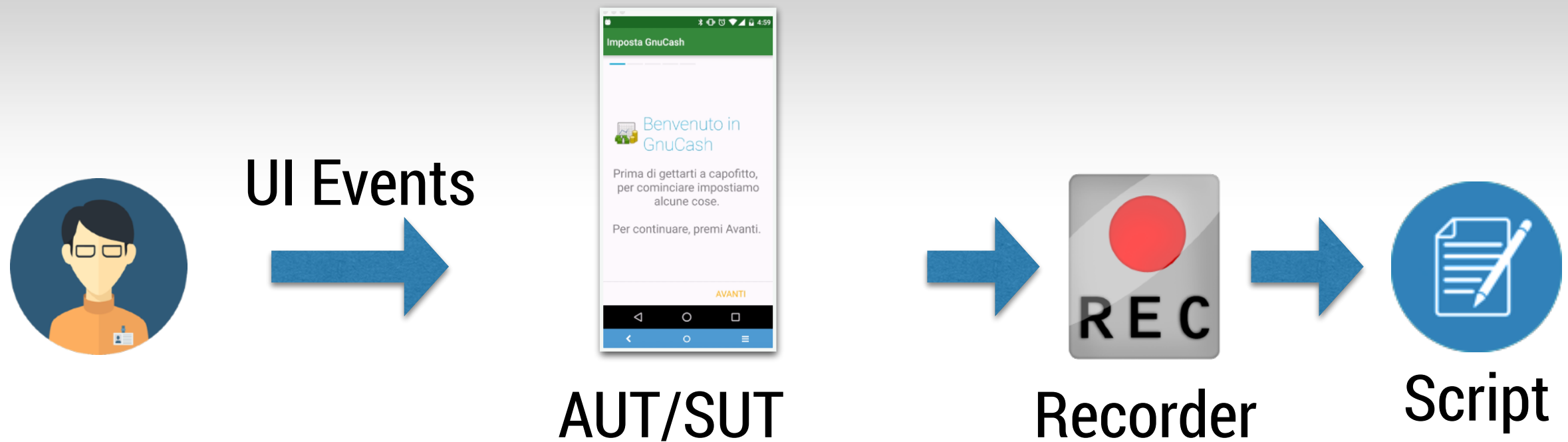


AUT/SUT

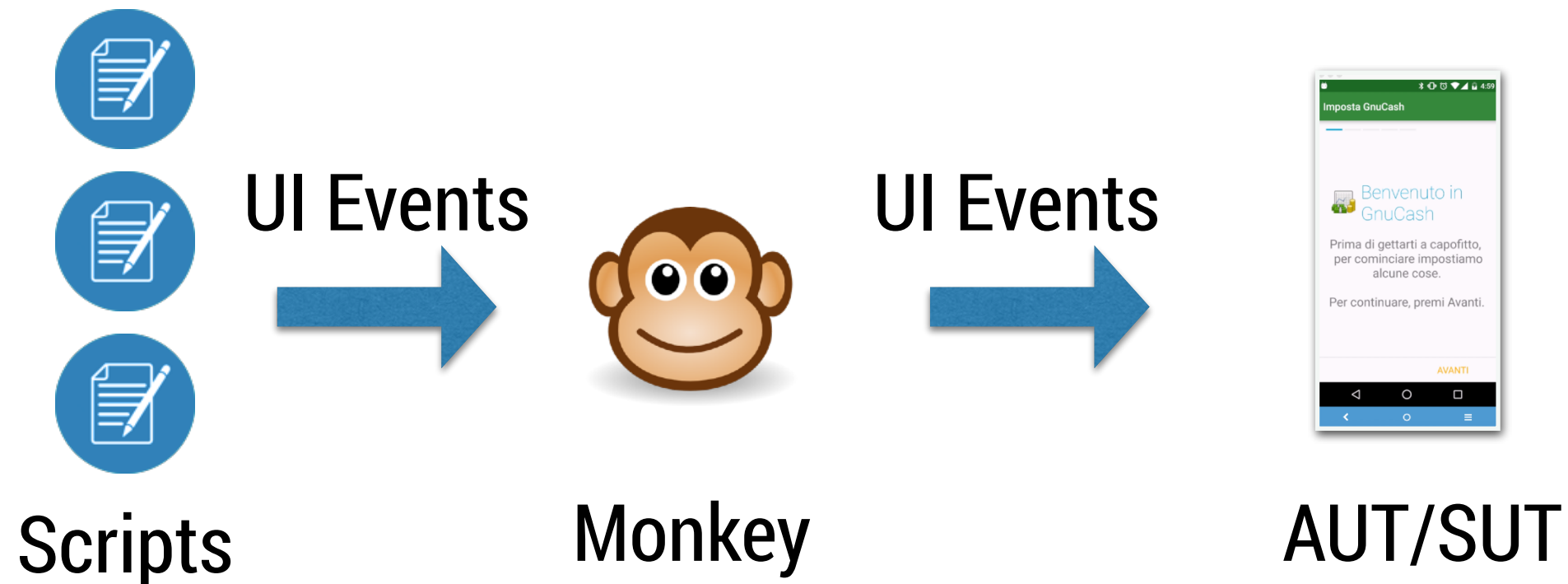
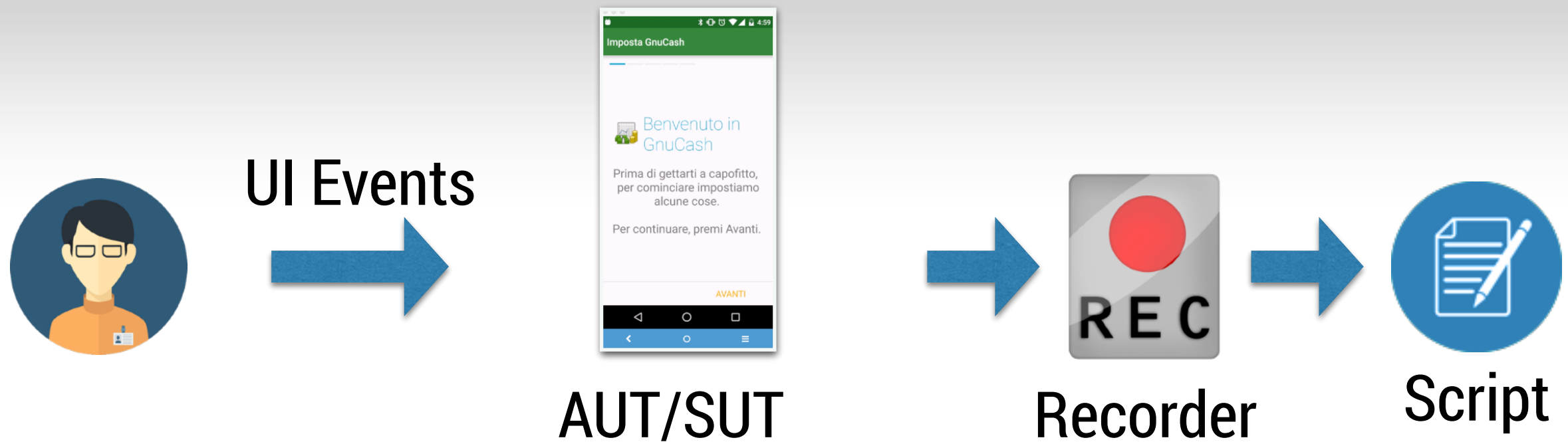
Record and Replay (R&R)



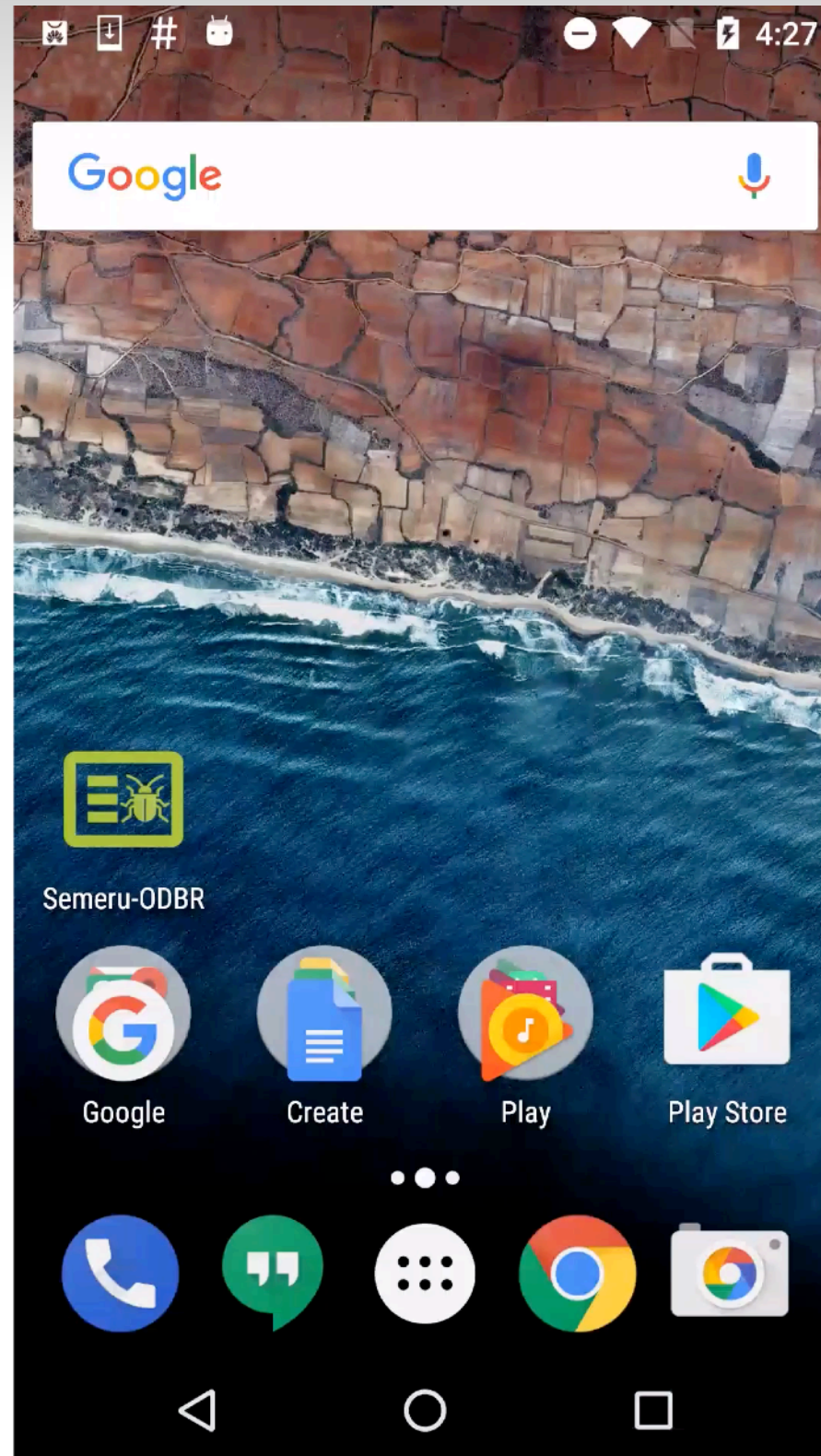
Record and Replay (R&R)



Record and Replay (R&R)

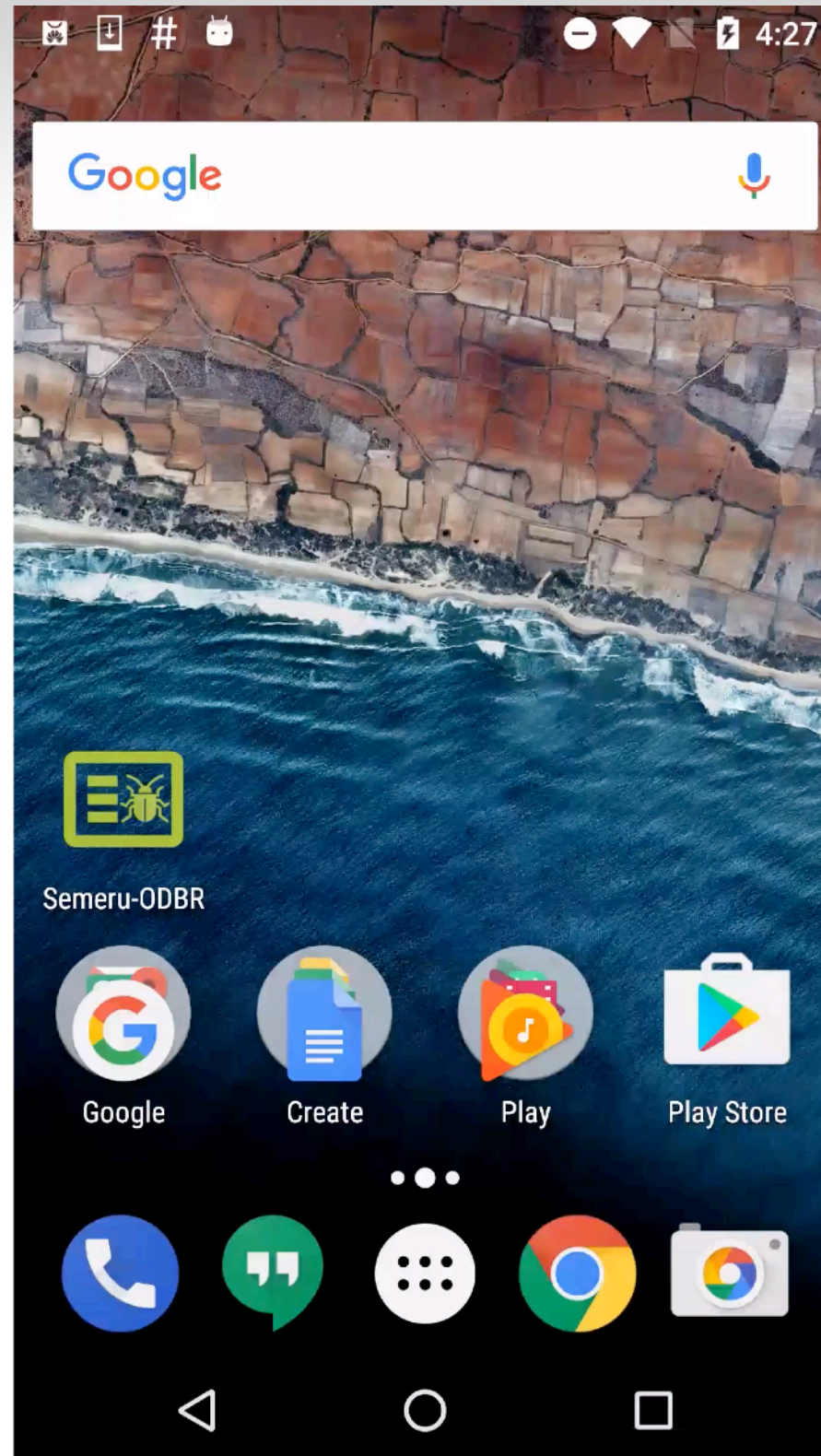


Tools: ODBR



www.android-dev-tools.com/odbr

Tools: ODBR



www.android-dev-tools.com/odbr

Pros and Cons



Automation Frameworks

- ✓ Easy reproduction
- ✓ High level syntax
- ✓ Black box testing



- Learning curve
- User-defined oracles
- Expensive maintenance

Record & Replay

- ✓ Easy reproduction

- Expensive collection and maintenance
- Coupled to locations

Automated Input Generation (AIG) Techniques

Automated Input Generation (AIG) Techniques

- **Differing Goals:**

Automated Input Generation (AIG) Techniques

- **Differing Goals:**
 - *Code Coverage*

Automated Input Generation (AIG) Techniques

- **Differing Goals:**
 - *Code Coverage*
 - *Crashes*

Automated Input Generation (AIG) Techniques

- **Differing Goals:**
 - *Code Coverage*
 - *Crashes*
- **Three Main Types:**

Automated Input Generation (AIG) Techniques

- **Differing Goals:**
 - *Code Coverage*
 - *Crashes*
- **Three Main Types:**
 - *Random-Based*

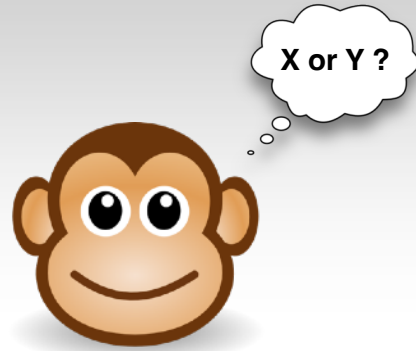
Automated Input Generation (AIG) Techniques

- **Differing Goals:**
 - *Code Coverage*
 - *Crashes*
- **Three Main Types:**
 - *Random-Based*
 - *Systematic*

Automated Input Generation (AIG) Techniques

- **Differing Goals:**
 - *Code Coverage*
 - *Crashes*
- **Three Main Types:**
 - *Random-Based*
 - *Systematic*
 - *Model-Based*

Random/Fuzz Testing (R/FT)



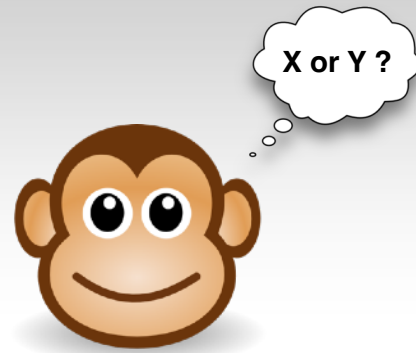
Monkey



AUT/SUT



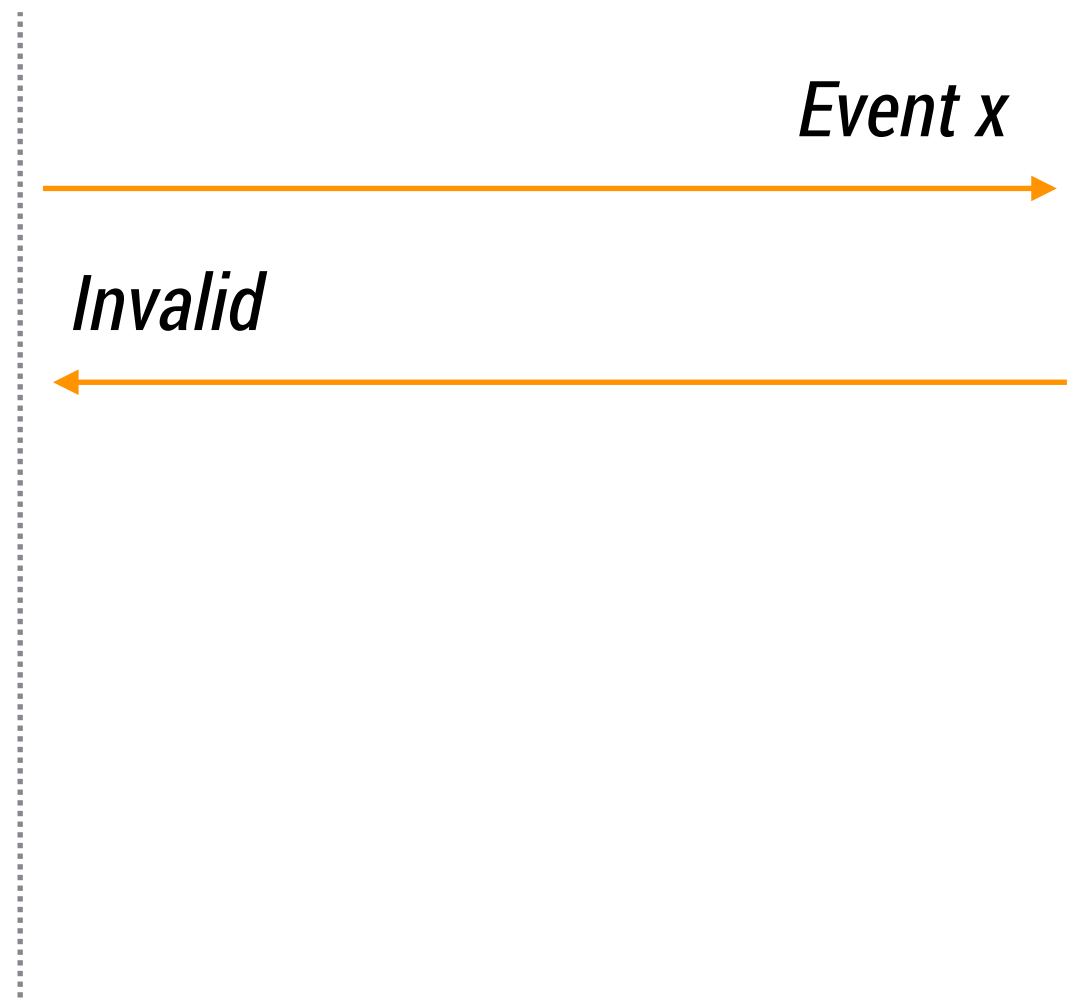
Random/Fuzz Testing (R/FT)



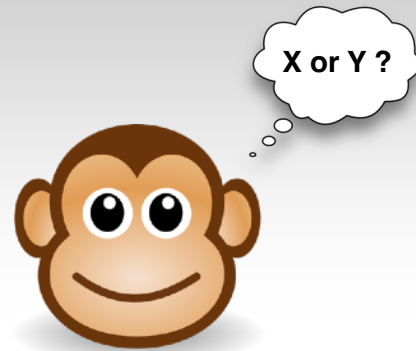
Monkey



AUT/SUT



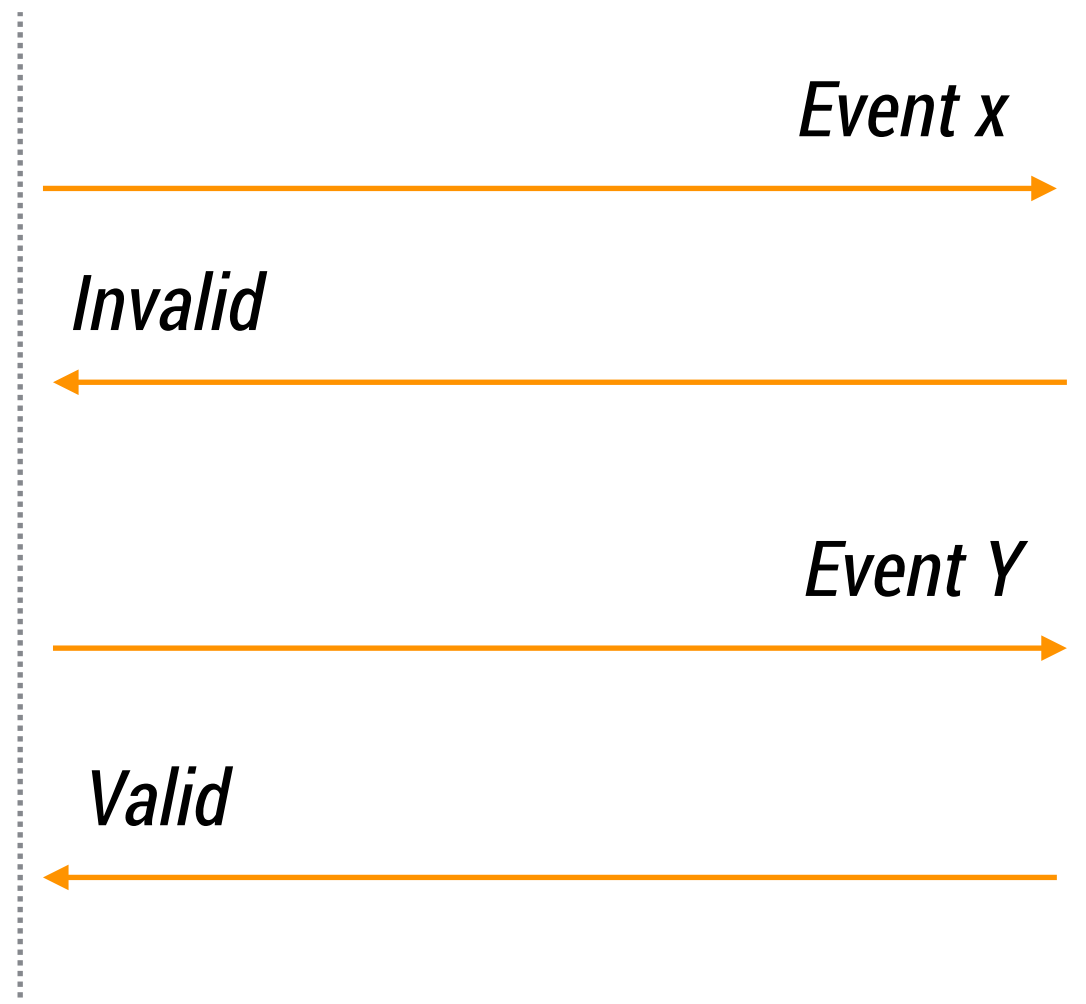
Random/Fuzz Testing (R/FT)



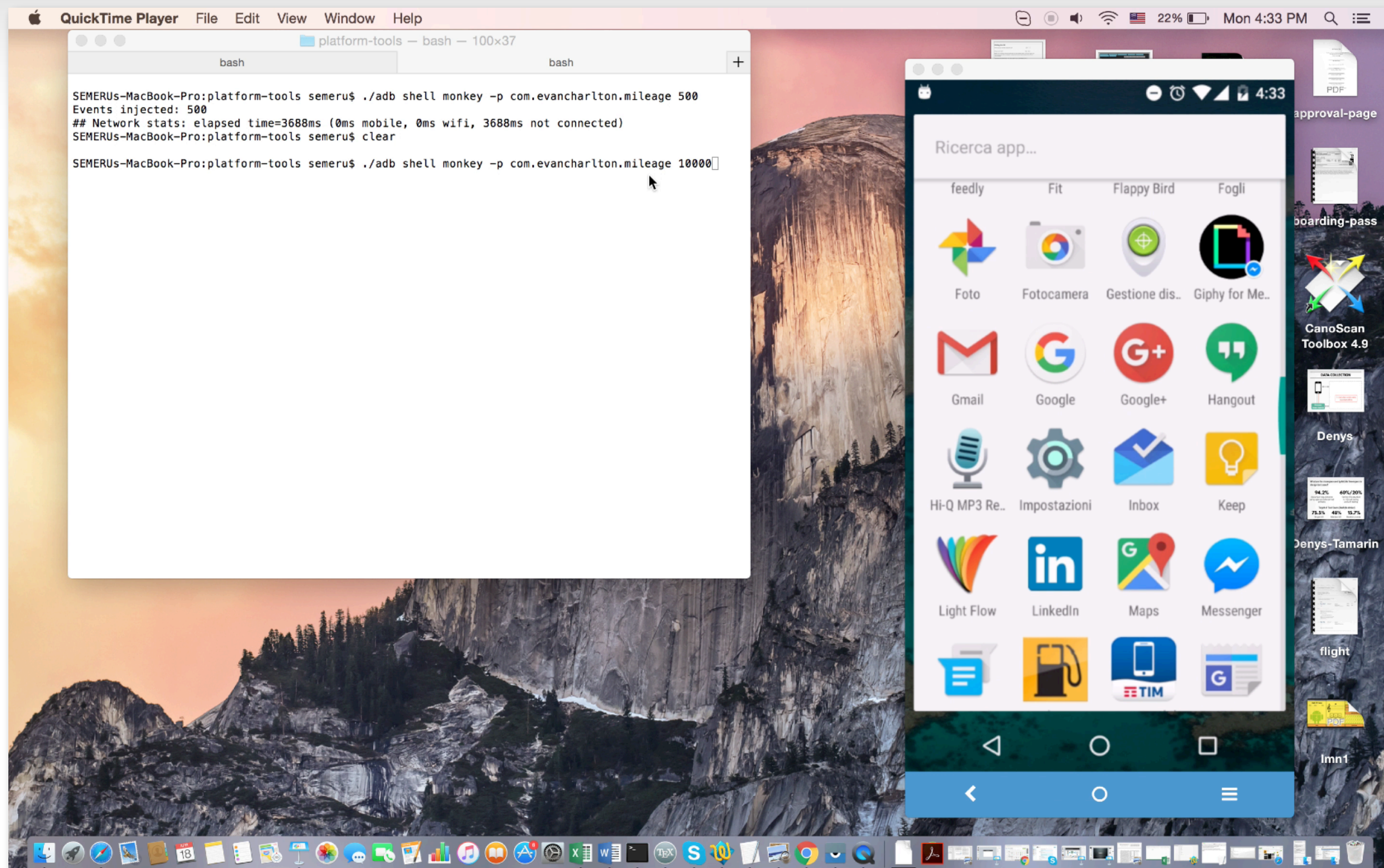
Monkey



AUT/SUT

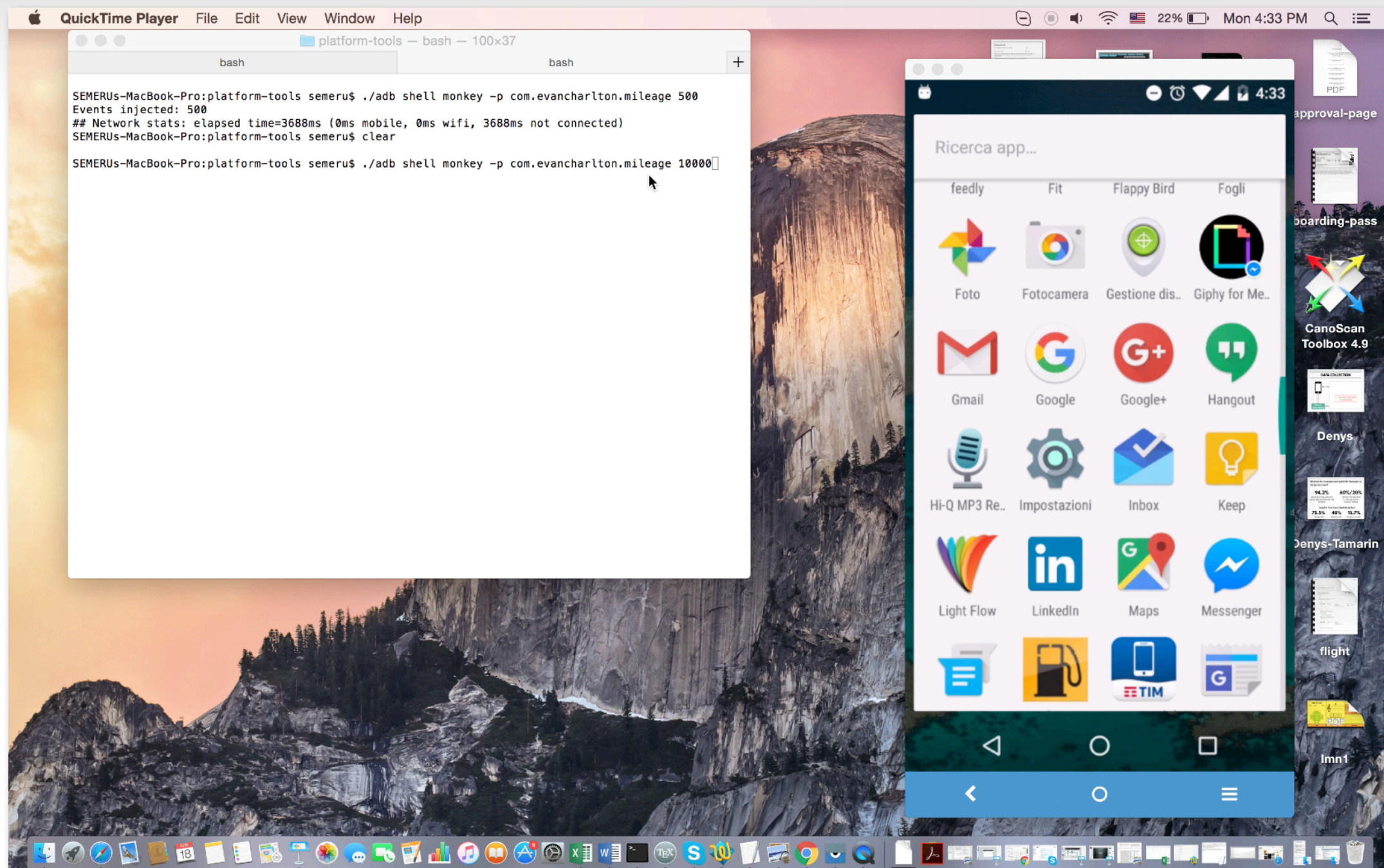


Random/Fuzz Testing (R/FT)



```
./adb shell monkey -p com.evancharlton.mileage 10000
```


Random/Fuzz Testing (R/FT)



```
./adb shell monkey -p com.evancharlton.mileage 10000
```

Pros and Cons



Automation Frameworks

- ✓ Easy reproduction
- ✓ High level syntax
- ✓ Black box testing



- Learning curve
- User-defined oracles
- Expensive maintenance

Record & Replay

- ✓ Easy reproduction

- Expensive collection and maintenance
- Coupled to locations

AIG: Random Based

- ✓ Fast execution
- ✓ Good at finding crashes

- Invalid events
- Lack of expressiveness

Aside: GUI Ripping



AUT/SUT



Ripper/Extractor



Model



Monkey

Aside: GUI Ripping



AUT/SUT



Ripper/Extractor



Model



Monkey

Snapshot 1



Aside: GUI Ripping



AUT/SUT



Ripper/Extractor



Model



Monkey

Snapshot 1



Snapshot 1



Aside: GUI Ripping



AUT/SUT



Ripper/Extractor



Model



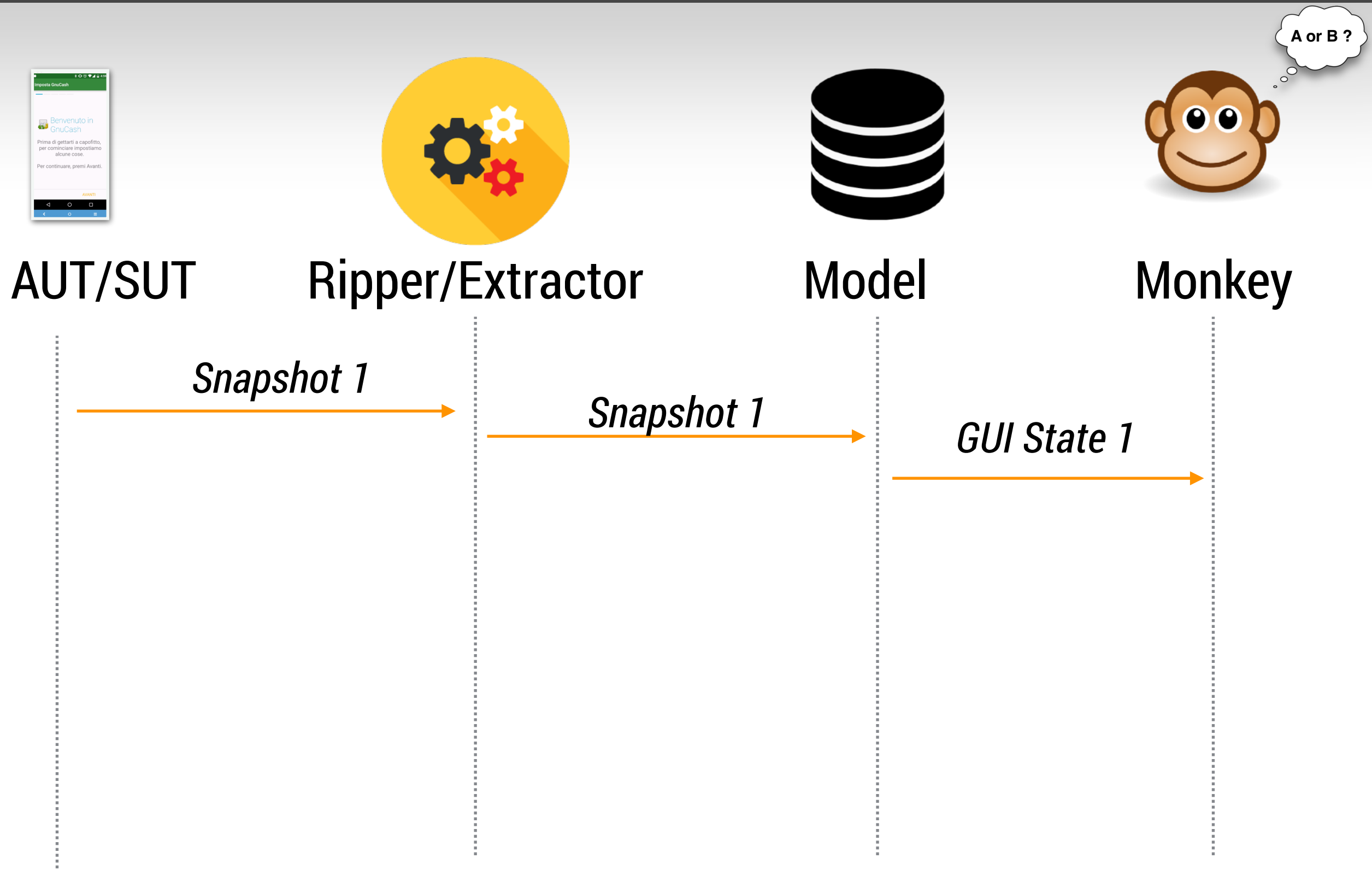
Monkey

Snapshot 1

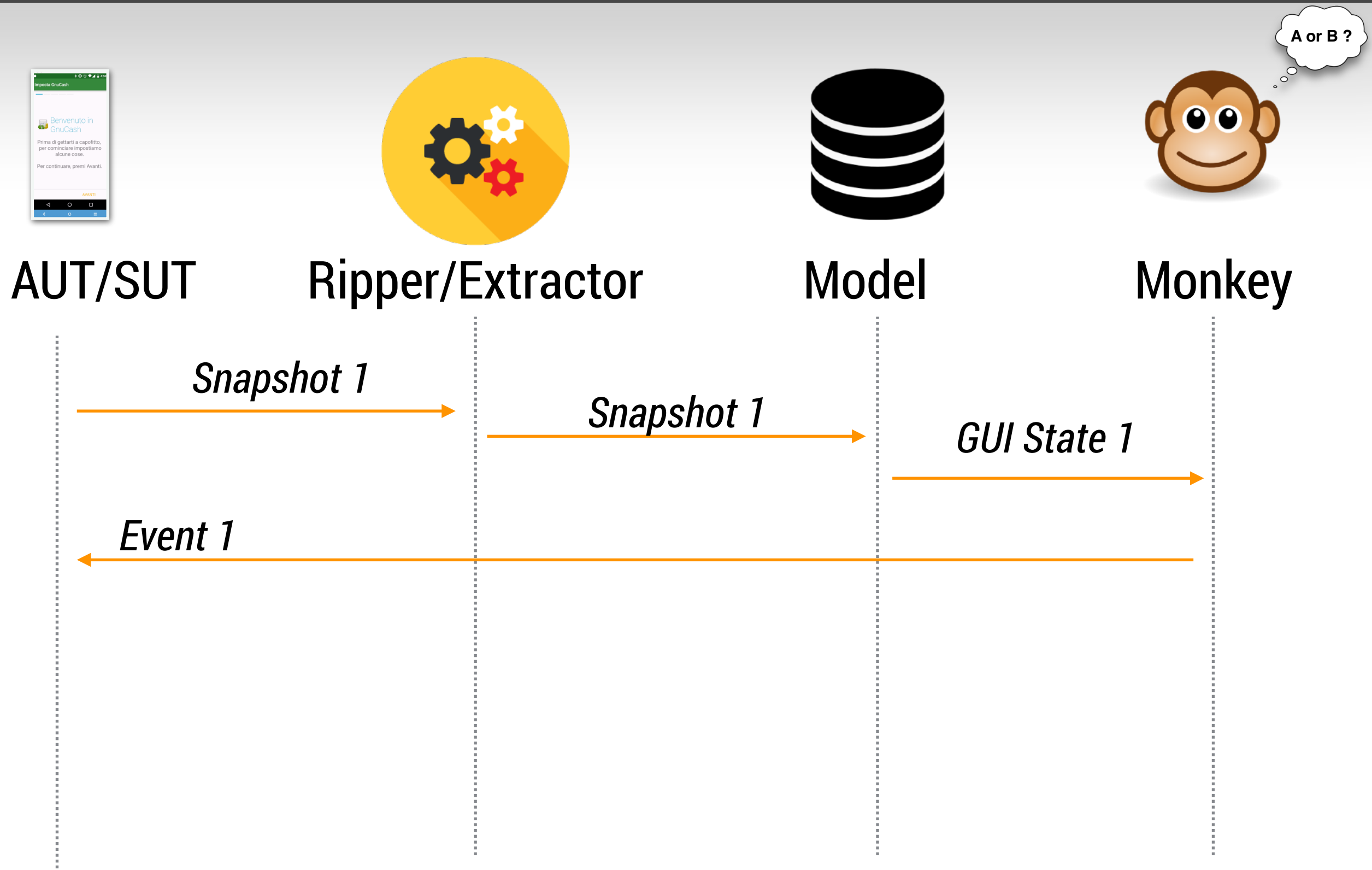
Snapshot 1

GUI State 1

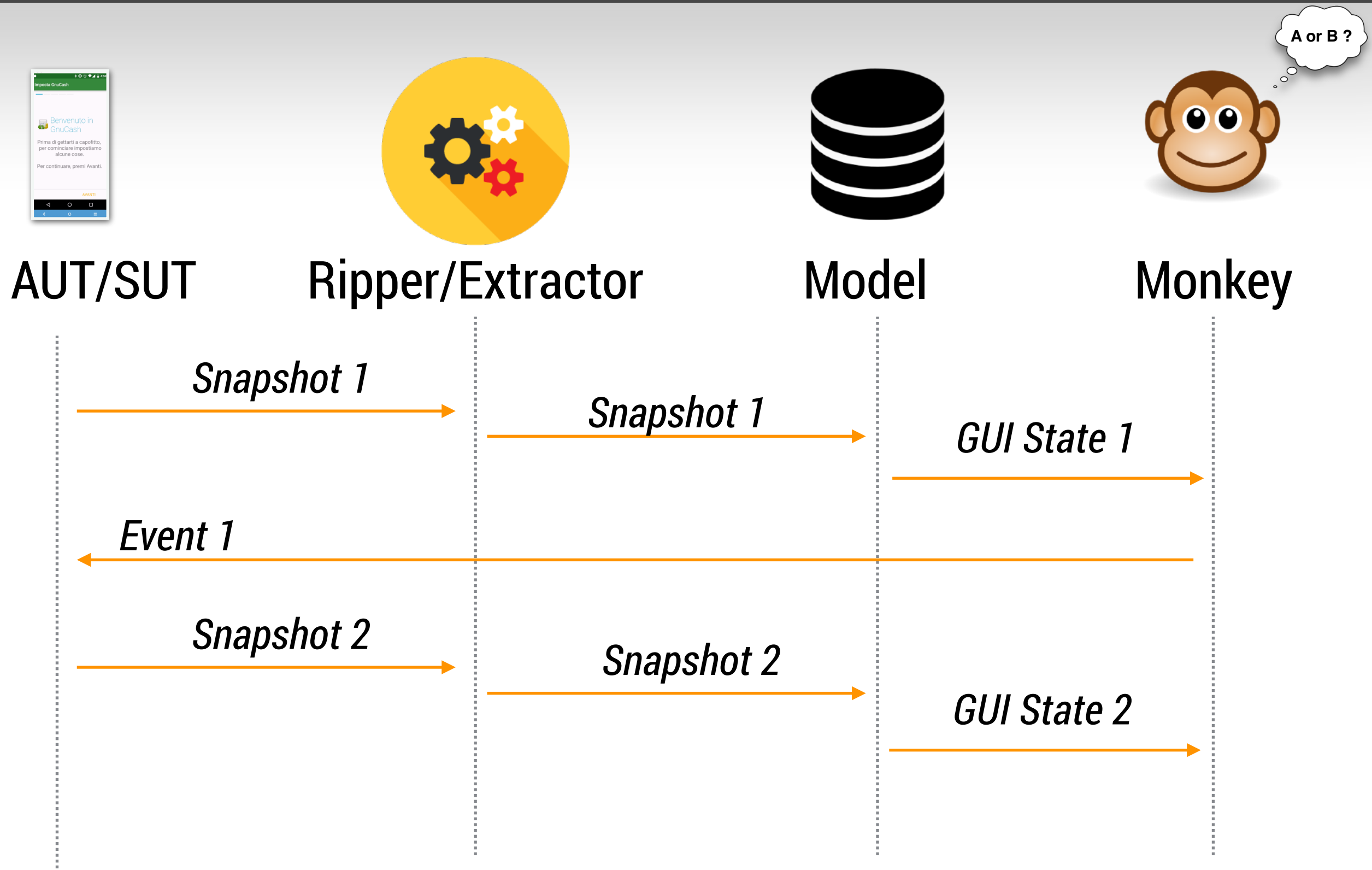
Aside: GUI Ripping



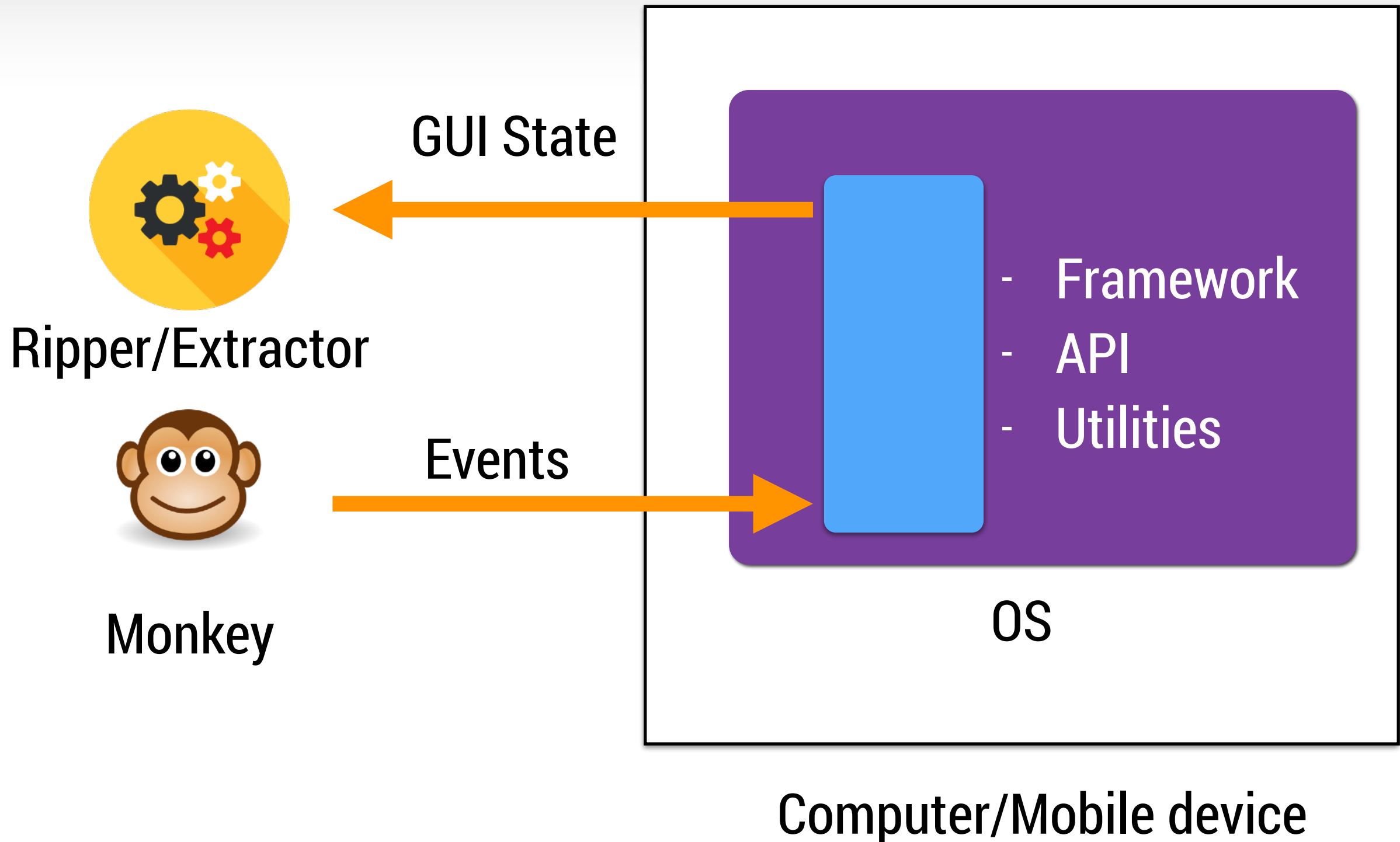
Aside: GUI Ripping



Aside: GUI Ripping



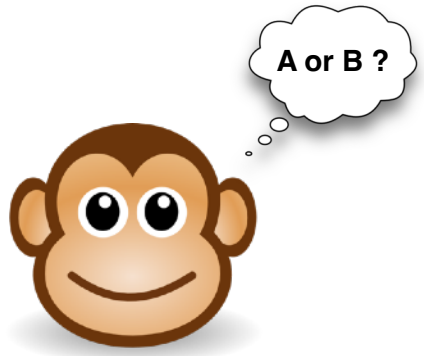
GUI State extraction



GUI State extraction

```
1 <?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
2 <hierarchy rotation="0">
3   <node index="0" text="" resource-id="" class="android.widget.FrameLayout" package="org
4     <node index="0" text="" resource-id="" class="android.widget.LinearLayout" package=
5     <node index="0" text="" resource-id="" class="android.widget.FrameLayout" package=
6     <node index="0" text="" resource-id="org.secuso.privacyfriendlydicer:id/decor_cont
7     <node index="0" text="" resource-id="org.secuso.privacyfriendlydicer:id/action_bar
8       <node index="0" text="" resource-id="org.secuso.privacyfriendlydicer:id/action
9         <node index="0" text="" resource-id="" class="android.widget.ImageButton"
10         <node index="1" text="Settings" resource-id="" class="android.widget.TextV
11       </node>
12     </node>
13   <node index="1" text="" resource-id="android:id/content" class="android.widget.Fra
14     <node index="0" text="" resource-id="" class="android.widget.LinearLayout" pac
15       <node index="0" text="" resource-id="android:id/list" class="android.widge
16         <node index="0" text="Dicing Settings" resource-id="android:id/title"
17         <node index="1" text="" resource-id="" class="android.widget.LinearLay
18           <node index="1" text="" resource-id="" class="android.widget.Relat
19             <node index="0" text="Roll Dice by Shaking" resource-id="andro
20           </node>
21         <node index="2" text="" resource-id="android:id/widget_frame" clas
22           <node index="0" text="" resource-id="android:id/checkbox" clas
23         </node>
24       </node>
```

Systematic Exploration



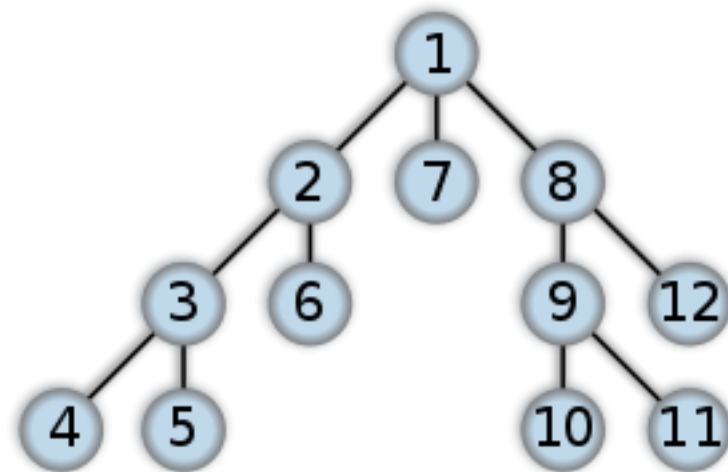
Monkey

Systematic Exploration

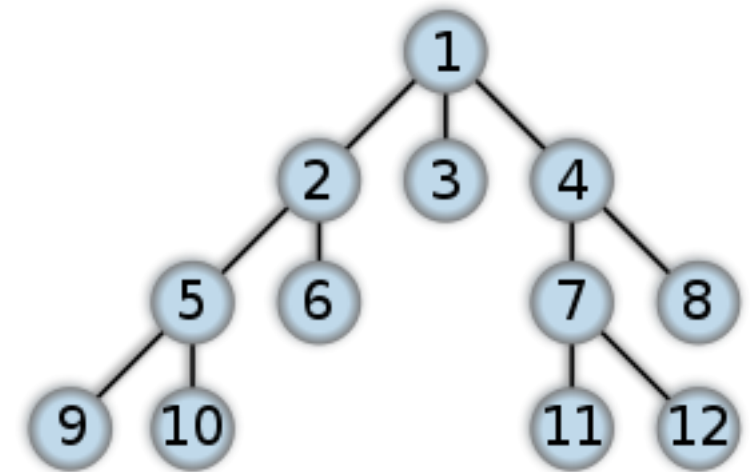


A or B ?

Monkey



Depth-First (DF)

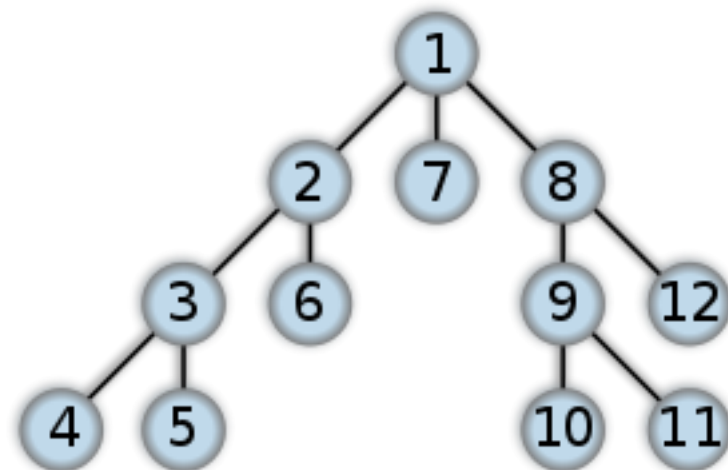


Breadth-First (BF)

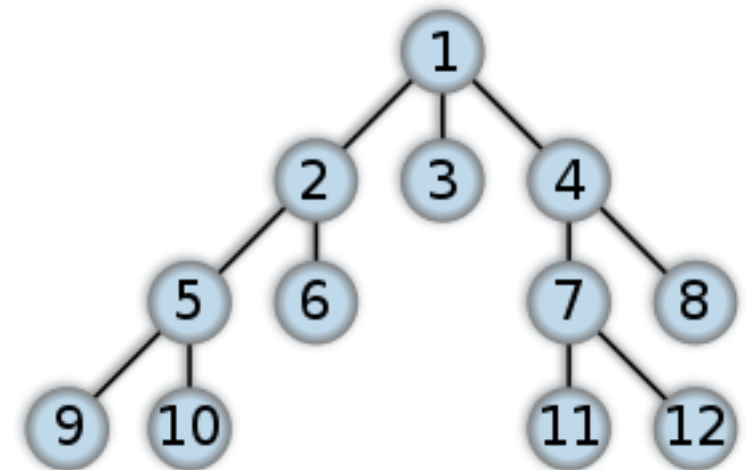
Systematic Exploration



Monkey



Depth-First (DF)



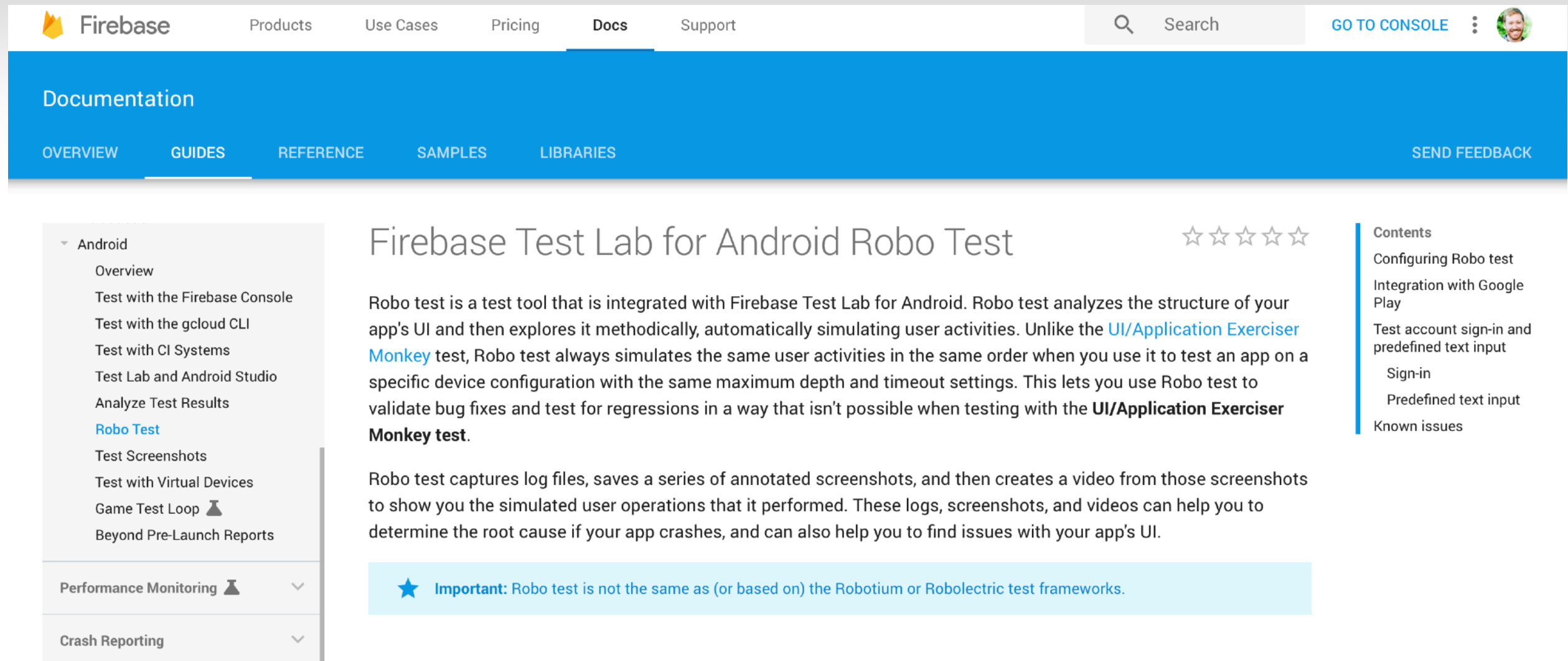
Breadth-First (BF)

Other options (online decision)

Random (Uniform)

Random (A-priori distr.)

Tools: Google Robo Test



The screenshot shows the Firebase documentation website. The top navigation bar includes links for Products, Use Cases, Pricing, Docs (selected), and Support. A search bar and a 'GO TO CONSOLE' button are also present. Below the navigation bar is a blue header with 'Documentation' and sub-navigation links: OVERVIEW, GUIDES (selected), REFERENCE, SAMPLES, and LIBRARIES. A 'SEND FEEDBACK' link is on the right. On the left side, there is a sidebar menu under the 'Android' category, listing various testing and monitoring tools. The main content area is titled 'Firebase Test Lab for Android Robo Test' and includes a star rating. The text describes Robo test as a tool integrated with Firebase Test Lab for Android, which analyzes the app's UI and simulates user activities. It compares Robo test to the UI/Application Exerciser Monkey test, noting that Robo test always simulates the same user activities in the same order. A blue callout box contains an important note. On the right, a 'Contents' section lists the page's structure.

Documentation

OVERVIEW GUIDES REFERENCE SAMPLES LIBRARIES SEND FEEDBACK

Android

- Overview
- Test with the Firebase Console
- Test with the gcloud CLI
- Test with CI Systems
- Test Lab and Android Studio
- Analyze Test Results
- Robo Test**
- Test Screenshots
- Test with Virtual Devices
- Game Test Loop
- Beyond Pre-Launch Reports

Performance Monitoring

Crash Reporting

Firebase Test Lab for Android Robo Test

☆☆☆☆☆

Robo test is a test tool that is integrated with Firebase Test Lab for Android. Robo test analyzes the structure of your app's UI and then explores it methodically, automatically simulating user activities. Unlike the [UI/Application Exerciser Monkey](#) test, Robo test always simulates the same user activities in the same order when you use it to test an app on a specific device configuration with the same maximum depth and timeout settings. This lets you use Robo test to validate bug fixes and test for regressions in a way that isn't possible when testing with the **UI/Application Exerciser Monkey test**.

Robo test captures log files, saves a series of annotated screenshots, and then creates a video from those screenshots to show you the simulated user operations that it performed. These logs, screenshots, and videos can help you to determine the root cause if your app crashes, and can also help you to find issues with your app's UI.

★ **Important:** Robo test is not the same as (or based on) the Robotium or Robolectric test frameworks.

Contents

- Configuring Robo test
- Integration with Google Play
- Test account sign-in and predefined text input
 - Sign-in
 - Predefined text input
- Known issues

<https://firebase.google.com/docs/test-lab/robo-ux-test>

Pros and Cons



Automation Frameworks

- ✓ Easy reproduction
- ✓ High level syntax
- ✓ Black box testing



- Learning curve
- User-defined oracles
- Expensive maintenance

Record & Replay

- ✓ Easy reproduction

- Expensive collection and maintenance
- Coupled to locations

AIG: Random Based

- ✓ Fast execution
- ✓ Good at finding crashes

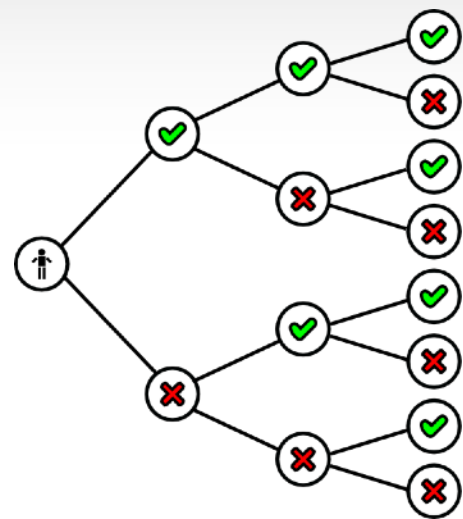
- Invalid events
- Lack of expressiveness

AIG: Systematic

- ✓ Achieves Reasonable Coverage
- ✓ May miss crashes

- Can be time consuming
- Typically cannot exercise complex features

Model-Based Testing (MBT)



Model



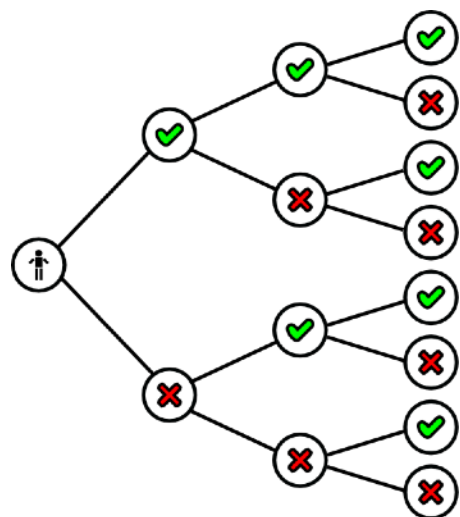
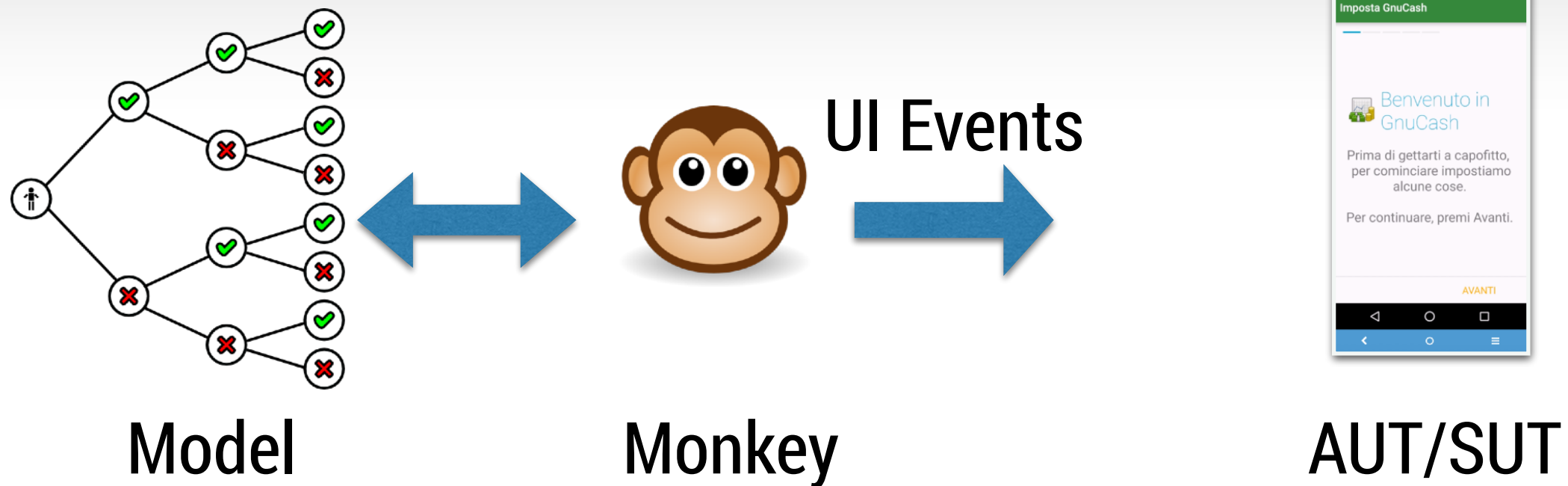
Monkey

UI Events



AUT/SUT

Model-Based Testing (MBT)



- Manually generated
- Automatically generated (source code)
- Ripped at runtime (upfront)
- Ripped at runtime (interactive)

Pros and Cons



Automation Frameworks

- ✓ Easy reproduction
- ✓ High level syntax
- ✓ Black box testing



- Learning curve
- User-defined oracles
- Expensive maintenance

Record & Replay

- ✓ Easy reproduction

- Expensive collection and maintenance
- Coupled to locations

AIG: Random Based

- ✓ Fast execution
- ✓ Good at finding crashes

- Invalid events
- Lack of expressiveness

AIG: Systematic

- ✓ Achieves Reasonable Coverage
- ✓ May miss crashes

- Can be time consuming
- Typically cannot exercise complex features

AIG: Model Based

- ✓ Event sequences
- ✓ Automatic exploration

- Some Invalid sequences
- State Explosion
- Incomplete models

Other Types of AIG Approaches

- **Recently New Approaches have been introduced for AIG:**
 - **Search-Based Approaches¹**
 - **Symbolic/Concolic Execution²**

¹Ke Mao, Mark Harman, and Yue Jia. 2016. Sapienz: multi-objective automated testing for Android applications. In *Proceedings of the 25th International Symposium on Software Testing and Analysis (ISSTA 2016)*

²Nariman Mirzaei, Joshua Garcia, Hamid Bagheri, Alireza Sadeghi, and Sam Malek. 2016. Reducing combinatorics in GUI testing of android applications. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*

Extending Research to XR/AR/VR apps



Extending Research to XR/AR/VR apps

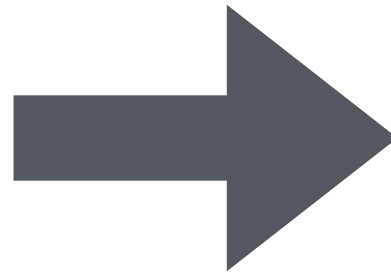


Extending Research to XR/AR/VR apps

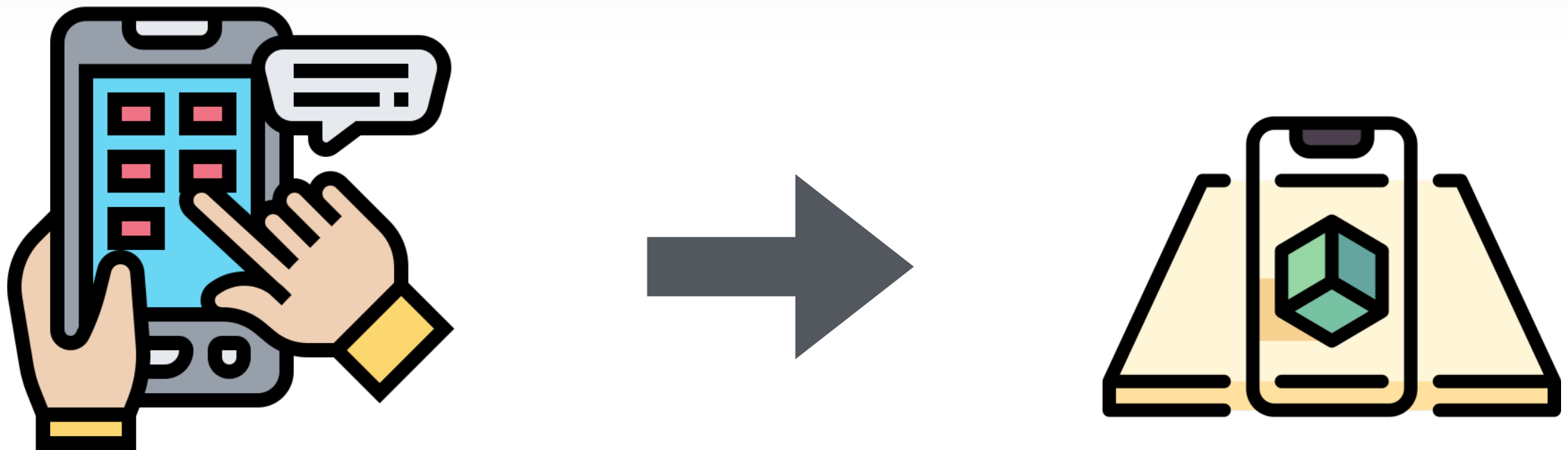


Main Challenge 1: Interfacing with and fetching GUI information

Extending Research to XR/AR/VR apps

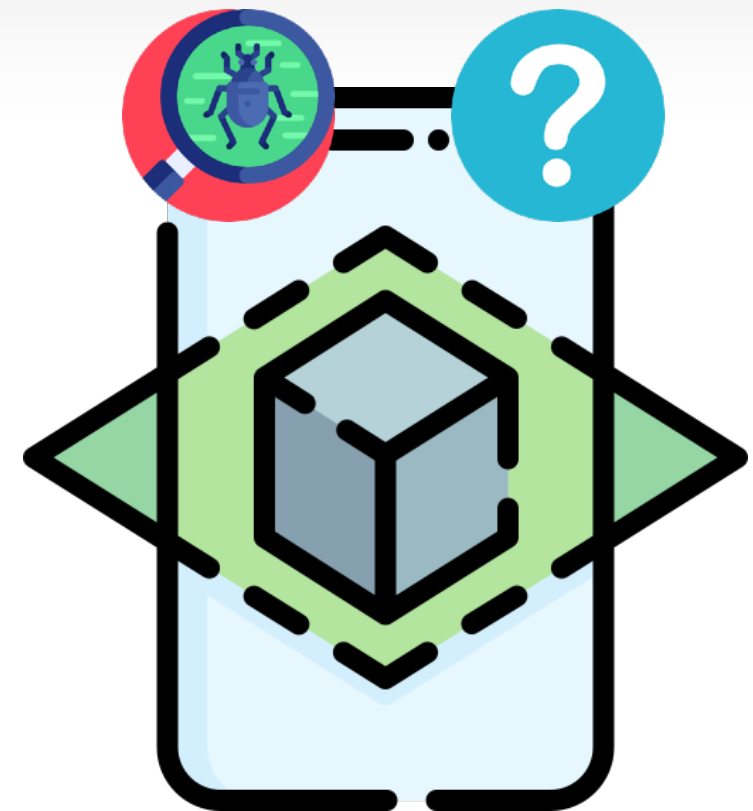
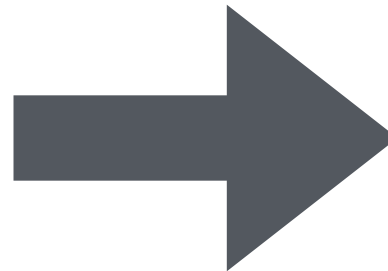
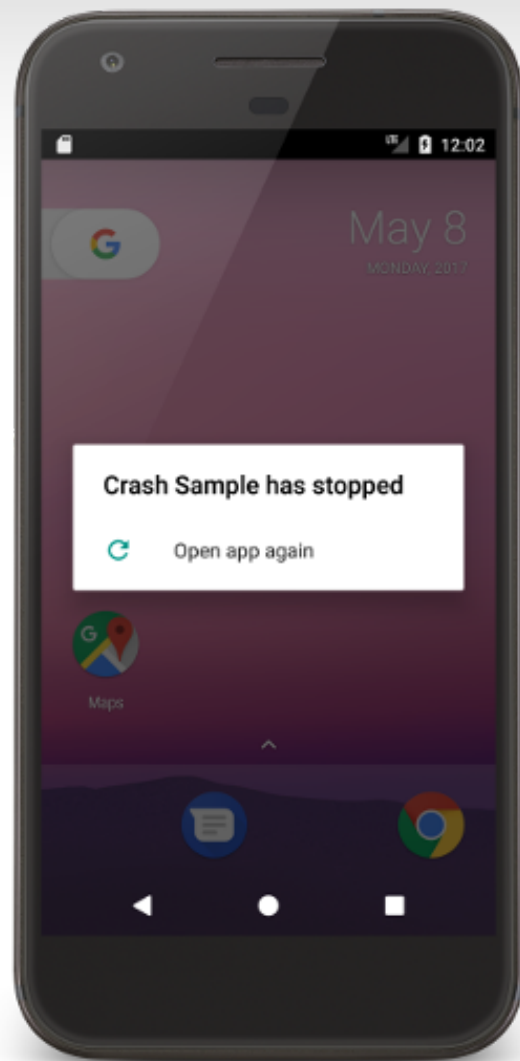


Extending Research to XR/AR/VR apps

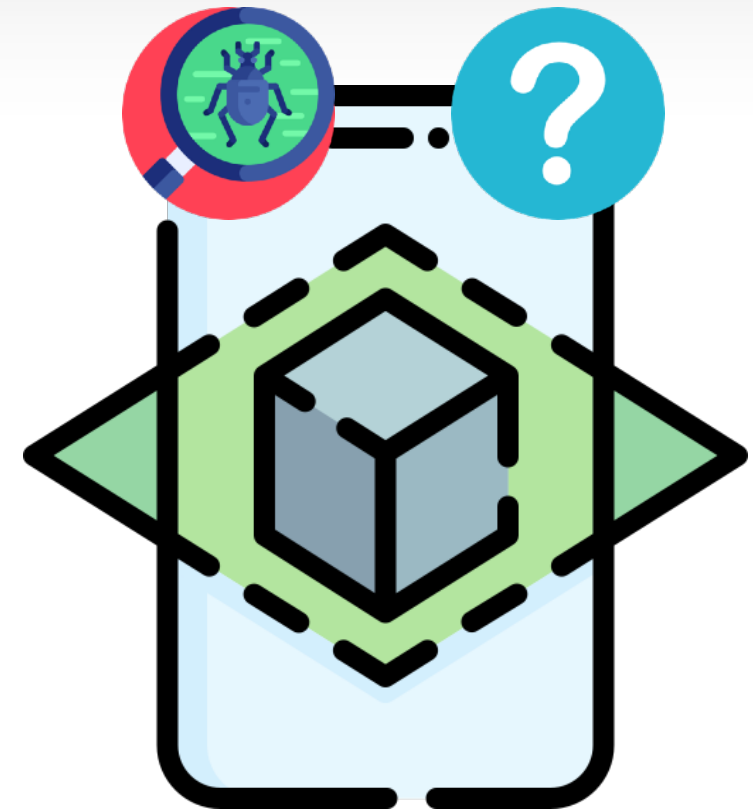
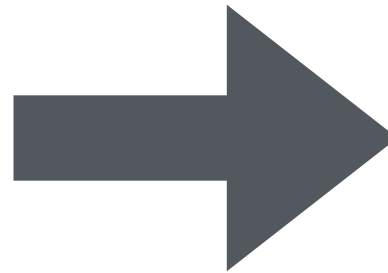
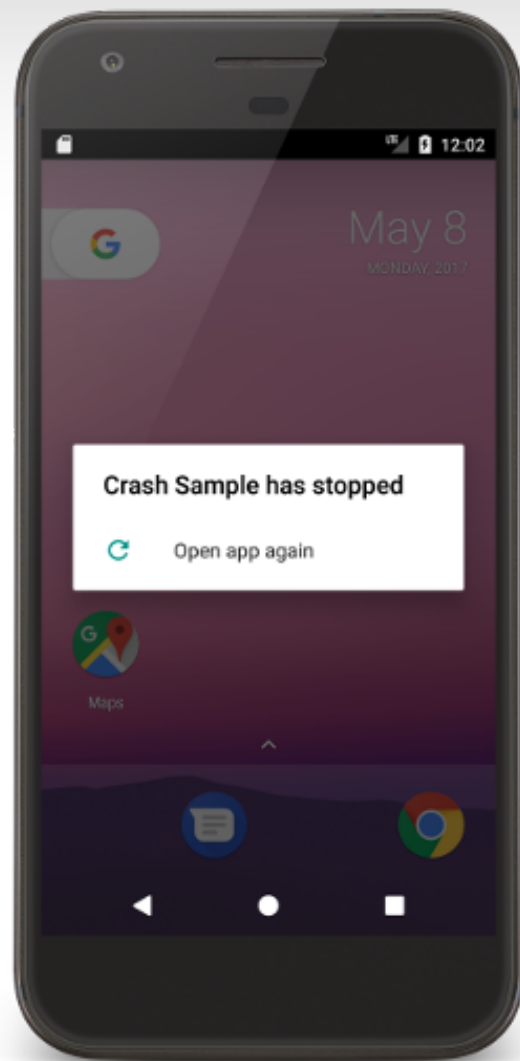


Main Challenge 2: Generating meaningful inputs

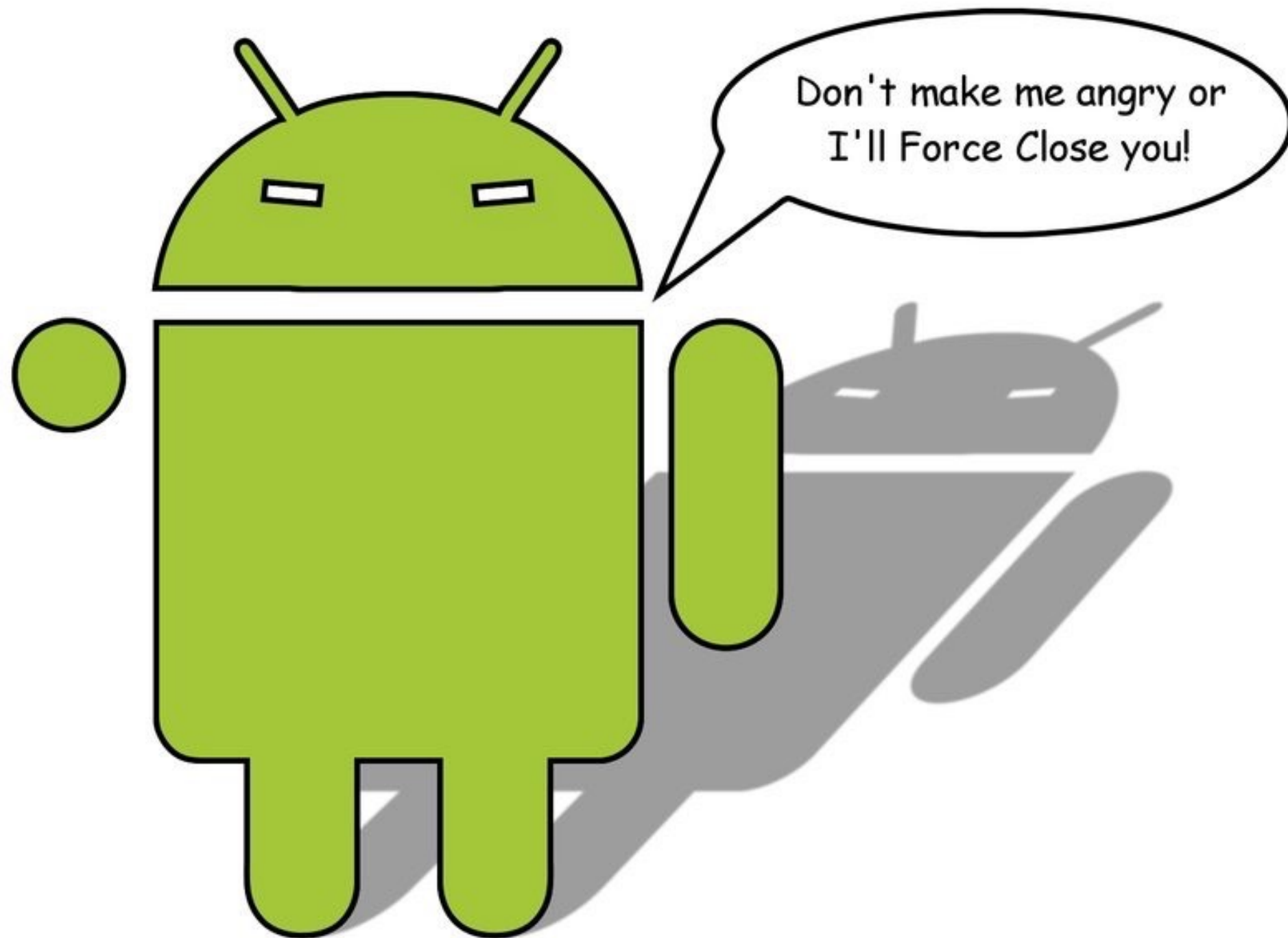
Extending Research to XR/AR/VR apps



Extending Research to XR/AR/VR apps



Main Challenge 3: Understanding & Detecting Failures



Automatically Discovering, Reporting and Reproducing
Android Application Crashes with CrashScope

CrashScope Publication

Automatically Discovering, Reporting and Reproducing Android Application Crashes

Kevin Moran, Mario Linares-Vásquez, Carlos Bernal-Cárdenas, Christopher Vendome, and Denys Poshyvanyk
College of William & Mary
{kpmoran, mlinarev, cebernal, cvendome, denys}@cs.wm.edu

Abstract—Mobile developers face unique challenges when detecting and reporting crashes in apps due to their prevailing GUI event-driven nature and additional sources of inputs (e.g., sensor readings). To support developers in these tasks, we introduce a novel, automated approach called CRASHSCOPE. This tool explores a given Android app using systematic input generation, according to several strategies informed by static and dynamic analyses, with the intrinsic goal of triggering crashes. When a crash is detected, CRASHSCOPE generates an augmented crash report containing screenshots, detailed crash reproduction steps, the captured exception stack trace, and a fully replayable script that automatically reproduces the crash on a target device(s).

We evaluated CRASHSCOPE's effectiveness in discovering crashes as compared to five state-of-the-art Android input generation tools on 61 applications. The results demonstrate that CRASHSCOPE performs about as well as current tools for detecting crashes and provides more detailed fault information. Additionally, in a study analyzing eight real-world Android app crashes, we found that CRASHSCOPE's reports are easily readable and allow for reliable reproduction of crashes by presenting more explicit information than human written reports.

I. INTRODUCTION

Continued growth in the mobile hardware and application marketplace is being driven by a landscape where users tend to prefer mobile smart devices and apps for tasks over their desktop counterparts. The gesture-driven nature of mobile apps has given rise to new challenges encountered by programmers during development and maintenance, specifically with regard to testing and debugging [41]. One of the most difficult [22], [24] and important maintenance tasks is the creation and resolution of bug reports [35]. Reports concerning application crashes are of particular importance to developers, because crashes represent a jarring software fault that is directly user facing and immediately impacts an app's utility and success. If an app is not behaving as expected due to crashes, missing features, or other bugs, nearly half of users are likely to abandon the app for a competitor [12] in a marketplace such as Google Play [10].

Mobile developers heavily rely on user reviews [42], [49], [65], crash reports from the field in the form of stack traces, or reports in open source issue tracking systems to detect bugs in their apps. In each of these cases, the bug/crash reports are typically lacking in information [27], [41], containing only a stack trace, overly detailed logs or loosely structured natural language (NL) information regarding the crash [23]. This is not surprising as previous studies showed that information, which is most useful for a developer resolving a bug report (e.g., reproduction steps, stack traces and test cases), is often

the most difficult information for reporters to provide [33]. Furthermore, the absence of this information is a major cause of developers failing to reproduce bug/crash reports [22]. In addition to the quality of the reports, some other factors specific to Android apps such as hardware and software fragmentation [3], API instability and fault-proneness [21], [48], the event-driven nature of Android apps, gesture-based interaction, sensor interfaces, and the possibility of multiple contextual states (e.g., wifi/GPS on/off) make the process of detecting, reporting, and reproducing crashes challenging.

Motivated by these current issues developers face regarding mobile application crashes, we designed and implemented CRASHSCOPE, a practical system that automatically discovers, reports, and reproduces crashes for Android applications. CRASHSCOPE explores a given app using a systematic input generation algorithm and produces expressive crash reports with explicit steps for reproduction in an easily readable natural language format. This approach requires only an .apk file and an Android emulator or device to operate and requires no instrumentation of the subject apps or the Android OS. The entirety of the CRASHSCOPE workflow is completely automated, requiring no developer intervention, other than reading produced reports. Our systematic execution includes different exploration strategies, aimed at eliciting crashes from Android apps, which include automatic text generation capabilities based on the context of allowable characters for text entry fields, and targeted testing of contextual features, such as the orientation of the device, wireless interfaces, and sensors. We specifically tailored these features to test the common causes of app crashes as identified by previous studies [26], [45], [79]. During execution, CRASHSCOPE captures detailed information about the subject app, such as the inputs sent to the device, screenshots and GUI information, exceptions, and crash information. This information is then translated into detailed crash reports and replayable scripts, for any encountered crash.

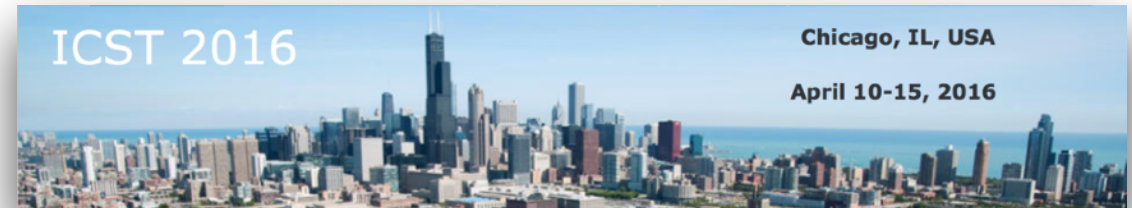
This paper makes the following noteworthy contributions:

- 1) We design and implement a practical and automatic approach for discovering, reporting, and reproducing Android application crashes, called CRASHSCOPE. To the best of the author's knowledge, this is the first approach that is able to generate expressive, detailed crash reports for mobile apps, including screenshots and augmented NL reproduction steps, in a completely automatic fashion. CRASHSCOPE is also one of the only available fully-automated Android testing approaches

ICST 2016

Chicago, IL, USA

April 10-15, 2016







Categories of automated testing approaches for Mobile apps

- Model-based input generation
- Random-based input generation
- Record and replay
- Others (Manual Testing Frameworks)

The Current State of Automated Mobile Testing

Tool Name	Instr.	GUI Exploration	Types of Events	Crash Resilient	Replayable Test Cases	NL Crash Reports	Emulators, Devices
Dynodroid	Yes	Guided/Random	System, GUI, Text	Yes	No	No	No
EvoDroid	No	System/Evo	GUI	No	No	No	N/A
AndroidRipper	Yes	Systematic	GUI, Text	No	No	No	N/A
MobiGUITar	Yes	Model-Based	GUI, Text	No	Yes	No	N/A
A3E DFS	Yes	Systematic	GUI	No	No	No	Yes
A3E Targeted [20]	Yes	Model-Based	GUI	No	No	No	Yes
Swifthand	Yes	Model-Based	GUI, Text	N/A	No	No	Yes
PUMA	Yes	Programmable	System, GUI, Text	N/A	No	No	Yes
ACTEve	Yes	Systematic	GUI	N/A	No	No	Yes
VANARSena	Yes	Random	System, GUI, Text	Yes	Yes	No	N/A
Thor	Yes	Test Cases	Test Case Events	N/A	N/A	No	No
QUANTUM	Yes	Model-Based	System, GUI	N/A	Yes	No	N/A
AppDoctor	Yes	Multiple	System, GUI, Text	Yes	Yes	No	N/A
ORBIT	No	Model-Based	GUI	N/A	No	No	N/A
SPAG-C	No	Record/Replay	GUI	N/A	N/A	No	No
JPF-Android	No	Scripting	GUI	N/A	Yes	No	N/A
MonkeyLab	No	Model-based	GUI, Text	No	Yes	No	Yes
CrashDroid	No	Manual Rec/Replay	GUI, Text	Manual	Yes	Yes	Yes
SIG-Droid	No	Symbolic	GUI, Text	N/A	Yes	No	N/A
CrashScope	No	Systematic	GUI, Text, System	Yes	Yes	Yes	Yes

The Current State of Automated Mobile Testing

Tool Name	Instr.	GUI Exploration	Types of Events	Crash Resilient	Replayable Test	NL Crash Reports	Emulators, Devices
Dr							
A							
A3							
What are the limitations of current automated approaches?							
Cr							
SIG-Dro	No	Systematic	GUI, Text	N/A	Yes	No	N/A
CrashScope	No	Systematic	GUI, Text, System	Yes	Yes	Yes	Yes

Limitations of Automated Mobile Testing and Debugging

- Lack of detailed, easy to understand testing results for faults/crashes¹
- No easy way to reproduce test scenarios¹
- Not practical from a developers viewpoint
- Few approaches enable different strategies capable of generating text and testing contextual features

¹S. R. Choudhary, A. Gorla, and A. Orso. Automated Test Input Generation for Android: Are we there yet? In 30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015), 2015

Past Studies of Mobile Bugs and Crashes

- Many crashes can be mapped to *well-defined, externally inducible* faults¹
- *Contextual features*, such as network connectivity and screen rotation, account for many of these externally inducible faults¹²
- These dominant root causes can affect *many different* user execution paths¹

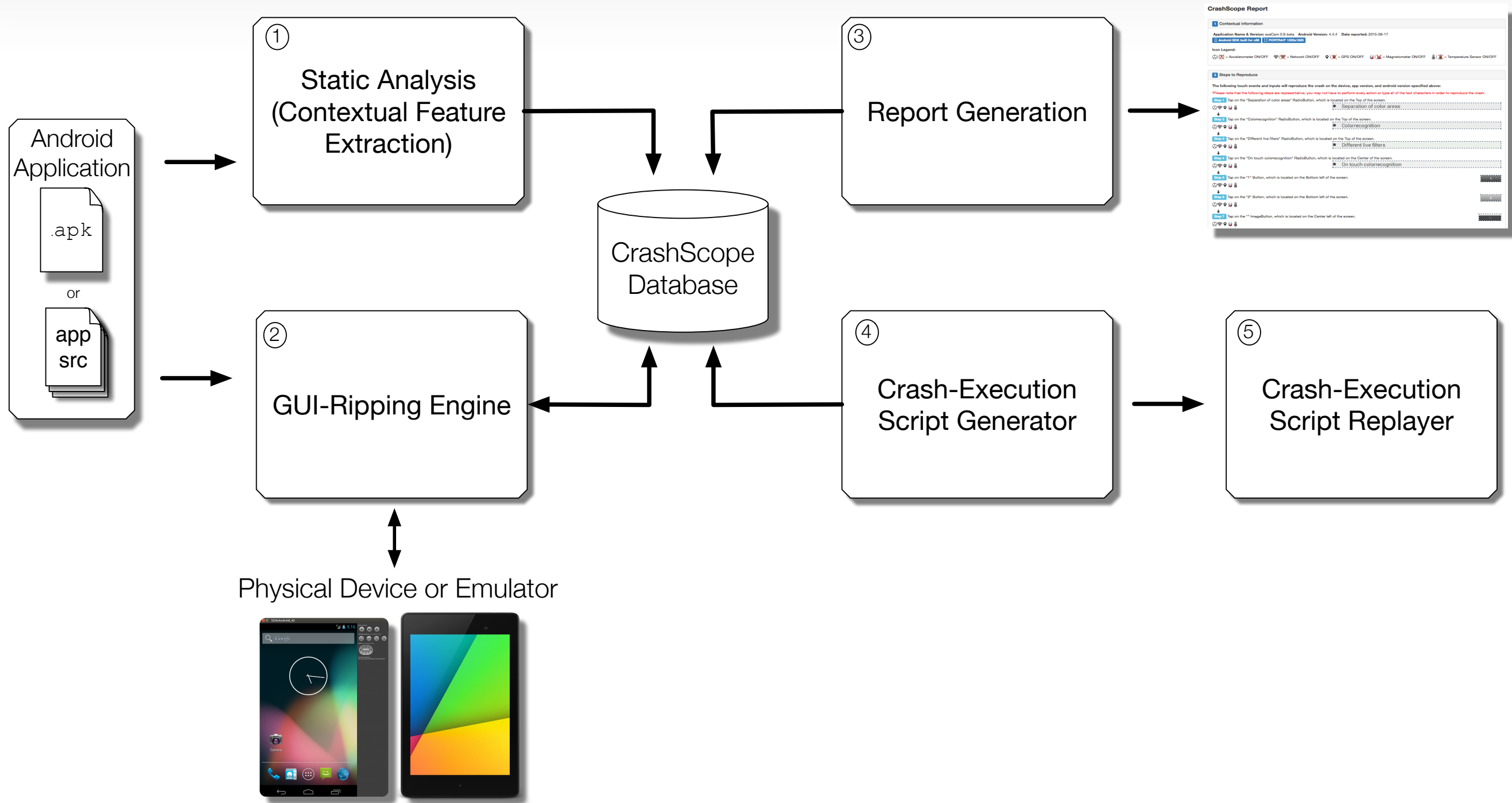
¹L. Ravindranath, S. Nath, J. Padhye, and H. Balakrishnan. Automatic and scalable fault detection for mobile applications. MobiSys '14

²R. N. Zaeem, M. R. Prasad, and S. Khurshid. Automated generation of oracles for testing user-interaction features of mobile apps, ICST '14

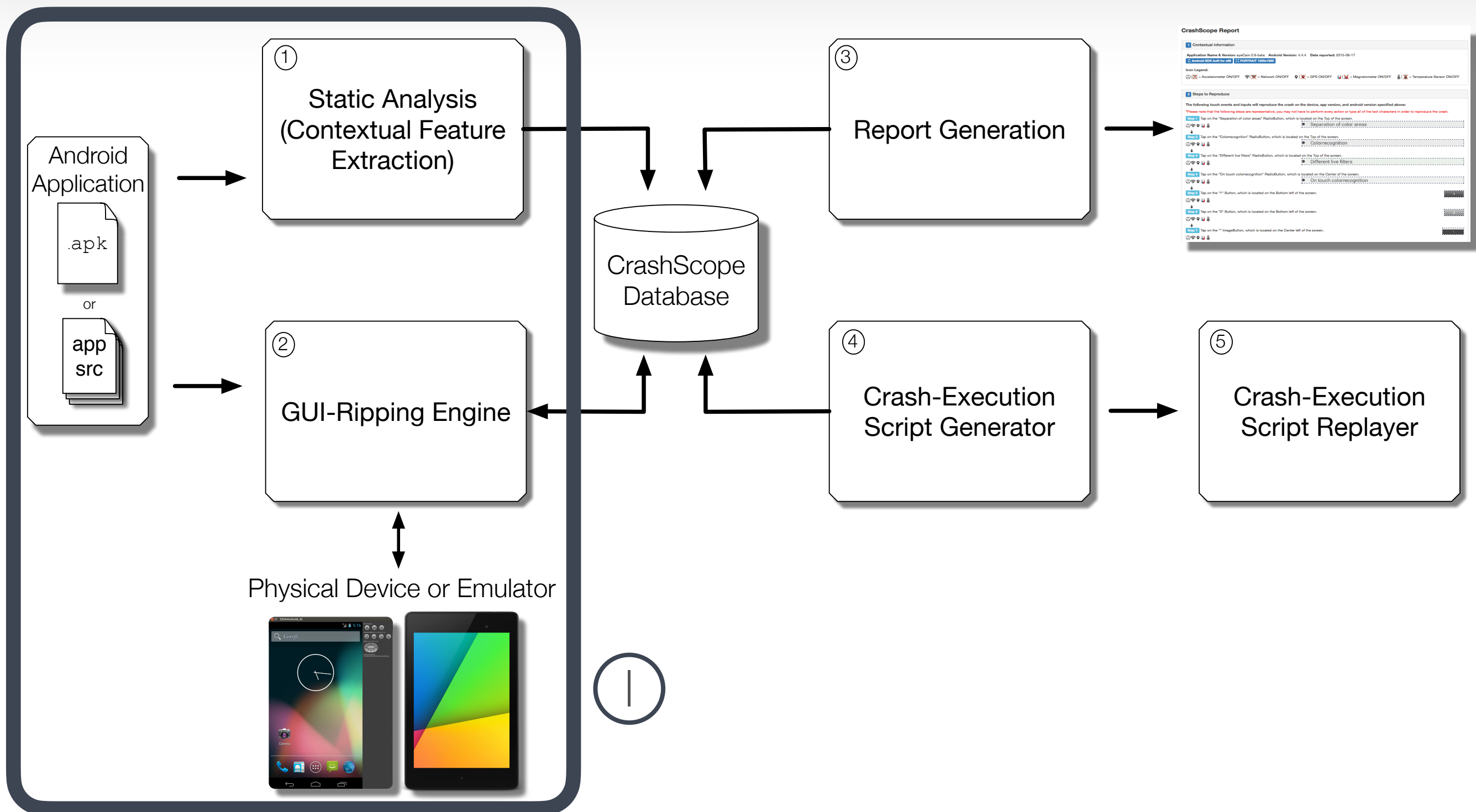
Our Solution: CRASHSCOPE

- Completely automated approach
- Generates detailed, expressive bug reports and replayable scripts
- A practical tool, requiring no instrumentation framework, or modification to the OS or applications
- Capable of running on both physical devices and emulators
- Differing execution strategies able to test contextual features

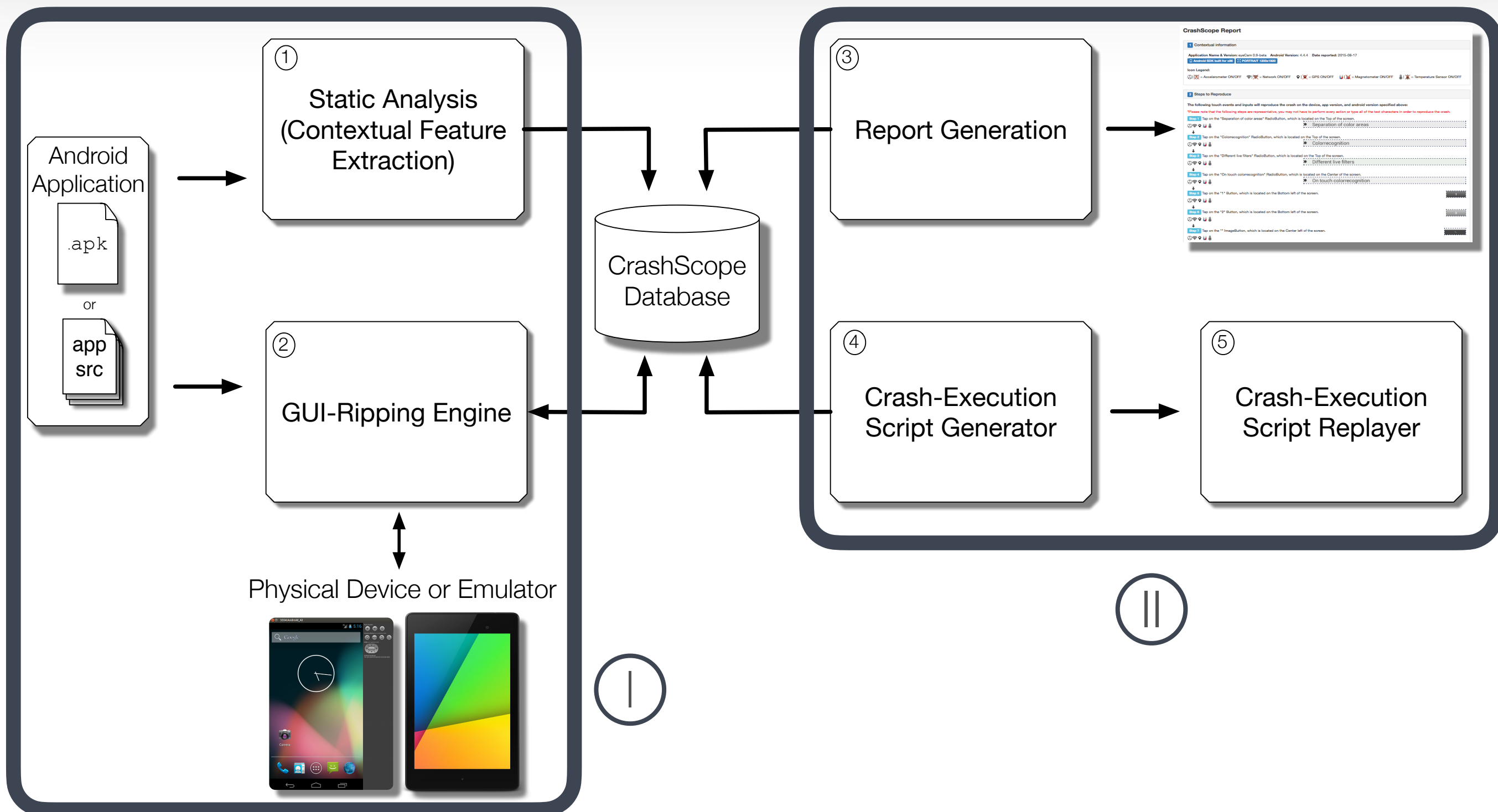
CRASHSCOPE: Design



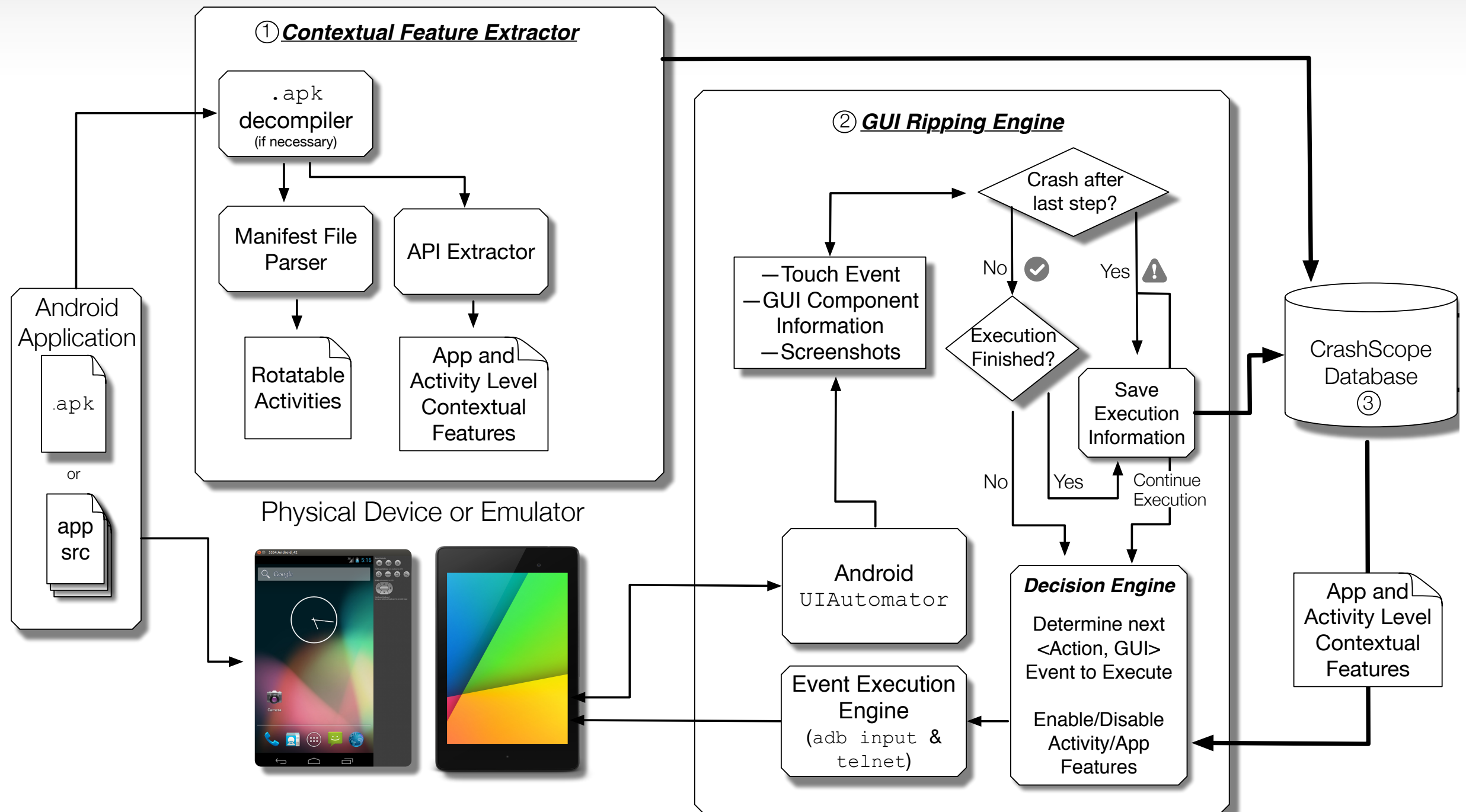
CRASHSCOPE: Design



CRASHSCOPE: Design



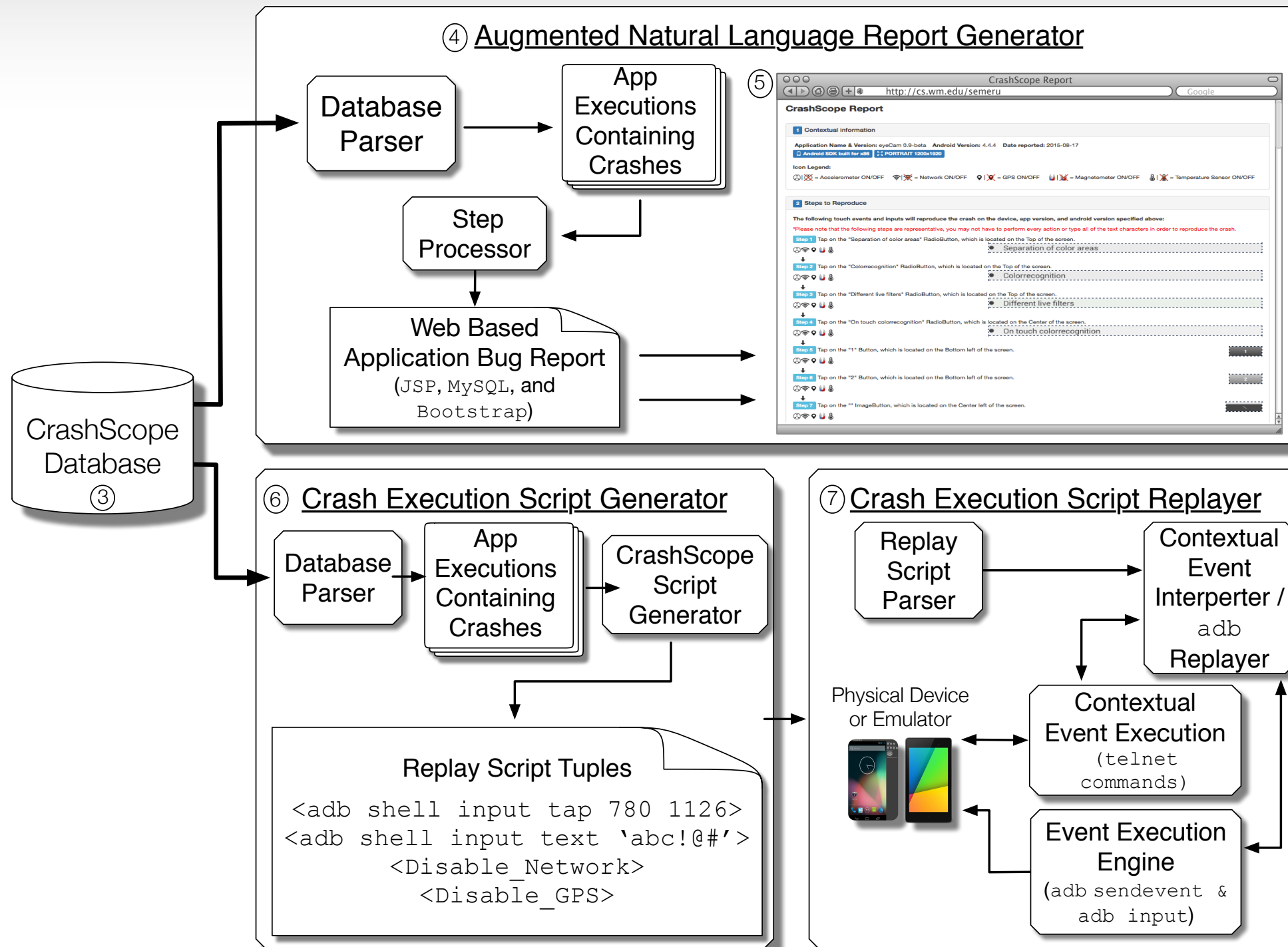
CRASHSCOPE: Analysis



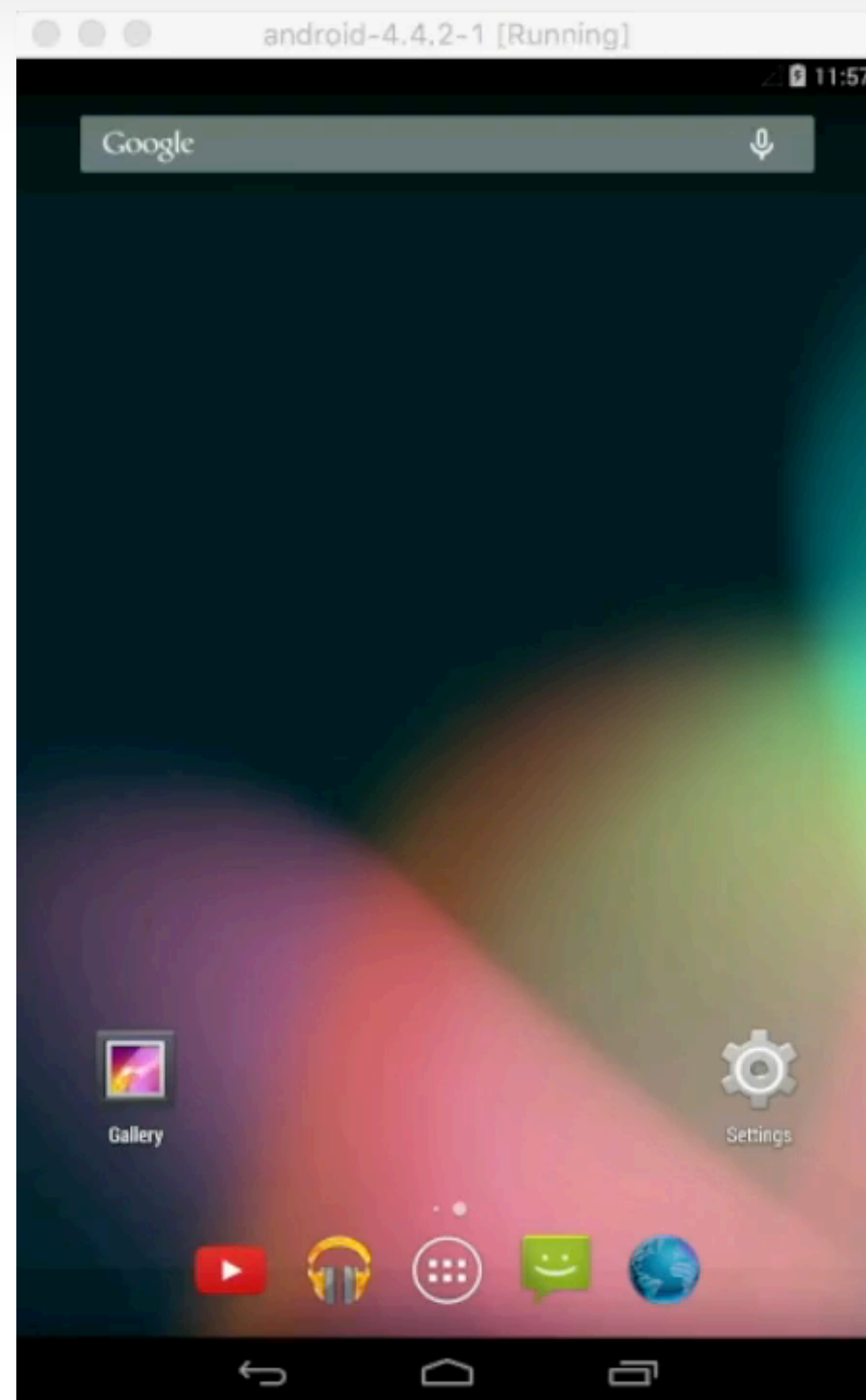
CRASHSCOPE: Exploration

- GUI-Traversal: Top-Down & Bottom Up
- Text Entry: Expected, Unexpected, No Text
- Contextual Features: Enabled or Disabled

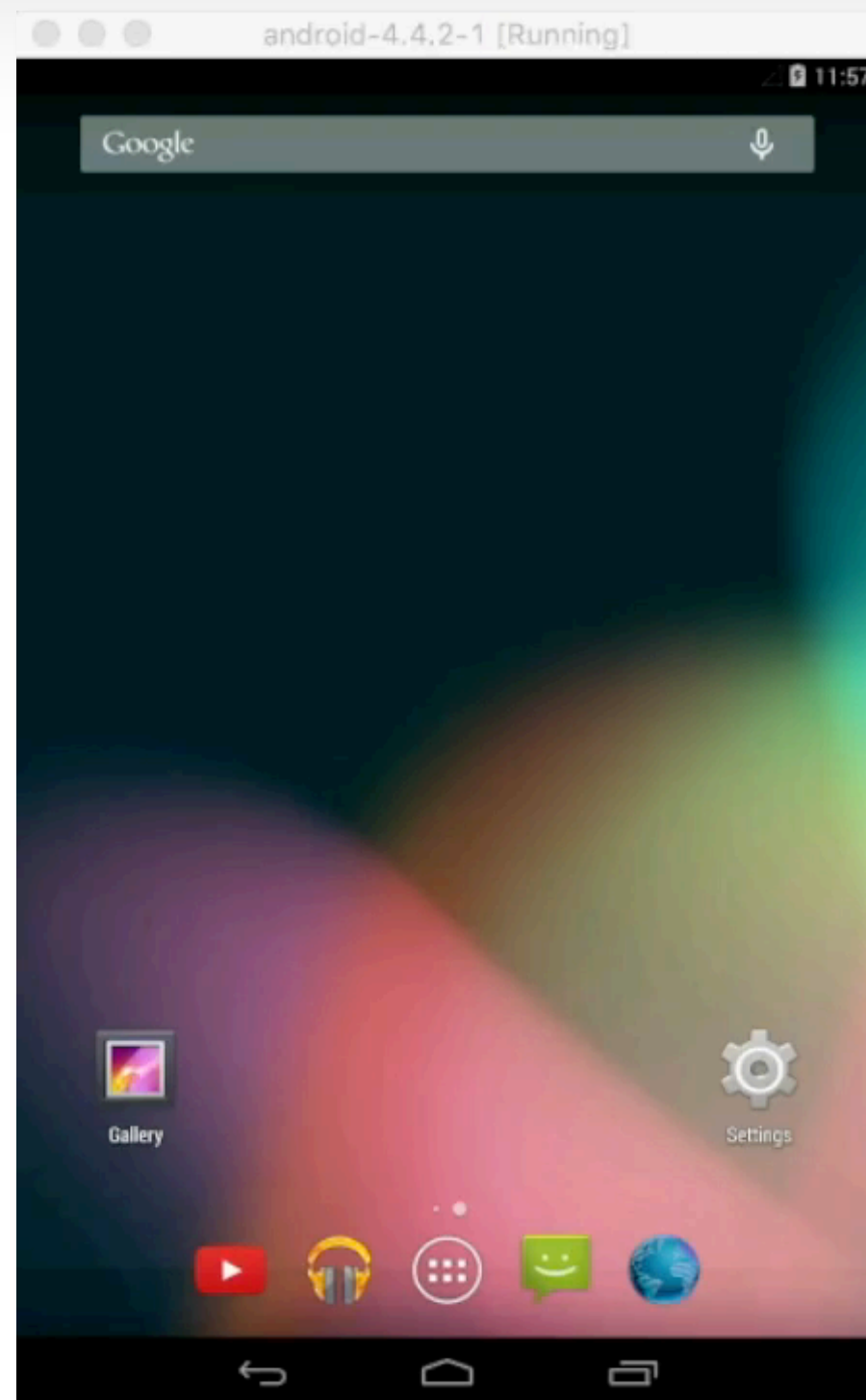
CRASHSCOPE: Report and Script Generation



CRASHSCOPE: Exploration Demo



CRASHSCOPE: Exploration Demo



CRASHSCOPE: Reports











CrashScope Report

1 Contextual information

Application Name & Version: GnuCash 1.0.3 **Android Version:** 4.4.4 **Date reported:** 2015-08-17

 Android SDK built for x86  PORTRAIT 1200x1920

Icon Legend:

  = Accelerometer ON/OFF   = Network ON/OFF   = GPS ON/OFF   = Magnetometer ON/OFF   = Temperature Sensor ON/OFF

2 Steps to Reproduce

The following touch events and inputs will reproduce the crash on the device, app version, and android version specified above:

*Please note that the following steps are representative, you may not have to perform every action or type all of the text characters in order to reproduce the crash.

Step 1 Tap on the "Expenses" CheckedTextView, which is located on the Center of the screen.



Step 2 Tap on the "Income" CheckedTextView, which is located on the Center of the screen.



Step 3 Tap on the "Assets" CheckedTextView, which is located on the Center of the screen.



Step 4 Tap on the "Entertainment" CheckedTextView, which is located on the Center of the screen.



Step 5 Tap on the "Insurance" CheckedTextView, which is located on the Center of the screen.




Step 6 Tap on the "Expenses" CheckedTextView, which is located on the Center of the screen.




Expenses 

Income 

Assets 

Entertainment 

Insurance 

Expenses 

CRASHSCOPE: Reports

Step 7 Tap on the "Income" CheckedTextView, which is located on the Center of the screen.



Step 8 Tap on the "Assets" CheckedTextView, which is located on the Center of the screen.



Step 9 Tap on the "Entertainment" CheckedTextView, which is located on the Center of the screen.



Step 10 Tap on the "Insurance" CheckedTextView, which is located on the Center of the screen.



Step 11 Tap on the "Cancel" Button, which is located on the Center left of the screen.



Step 12 Tap on the "Accounts" LinearLayout, which is located on the Top left of the screen.



Step 13 Tap on the "" TextView, which is located on the Top right of the screen.



Step 14 Type "sphvxfriq%sz" on the "Account name" EditText, which is located on the Center of the screen.



Step 15 Tap on the "" Spinner, which is located on the Center of the screen.



Step 16 Tap on the "US Dollar" CheckedTextView, which is located on the Top of the screen.



Income ☐

Assets ☒

Entertainment ☒

Insurance ☒

Cancel

Accounts

+

inhphsigfi z|

US Dollar

US Dollar

CRASHSCOPE: Reports

↓

Step 17 Type "ltmjfwjqss%sz" on the "Account name" EditText, which is located on the Center of the screen.

📶 📶 📶 📶 📶

↓

Step 18 Tap on the "Cancel" Button, which is located on the Center left of the screen.

📶 📶 📶 📶 📶

↓

Step 19 Long-touch on the "" TextView, which is located on the Top right of the screen.

📶 📶 📶 📶 📶

↓

Step 20 Tap on the "" ImageButton, which is located on the Top right of the screen.

📶 📶 📶 📶 📶

↓

Step 21 Tap on the "Export OFX" LinearLayout, which is located on the Top right of the screen.

📶 📶 📶 📶 📶

↓

Step 22 Tap on the "" Spinner, which is located on the Center of the screen.

📶 📶 📶 📶 📶

↓

Step 23 Tap on the "Share file..." TextView, which is located on the Center of the screen.

📶 📶 📶 📶 📶

↓

Step 24 Tap on the "Export all transactions" CheckBox, which is located on the Center left of the screen.

📶 📶 📶 📶 📶

↓

Step 25 Tap on the "Delete after export" CheckBox, which is located on the Center left of the screen.

📶 📶 📶 📶 📶

↓

Step 26 Tap on the "Cancel" Button, which is located on the Center left of the screen.

📶 📶 📶 📶 📶

ltmjfwjqss%sz

Cancel

+

!

Export OFX

Share file...

Share file...

☐ Export all transactions

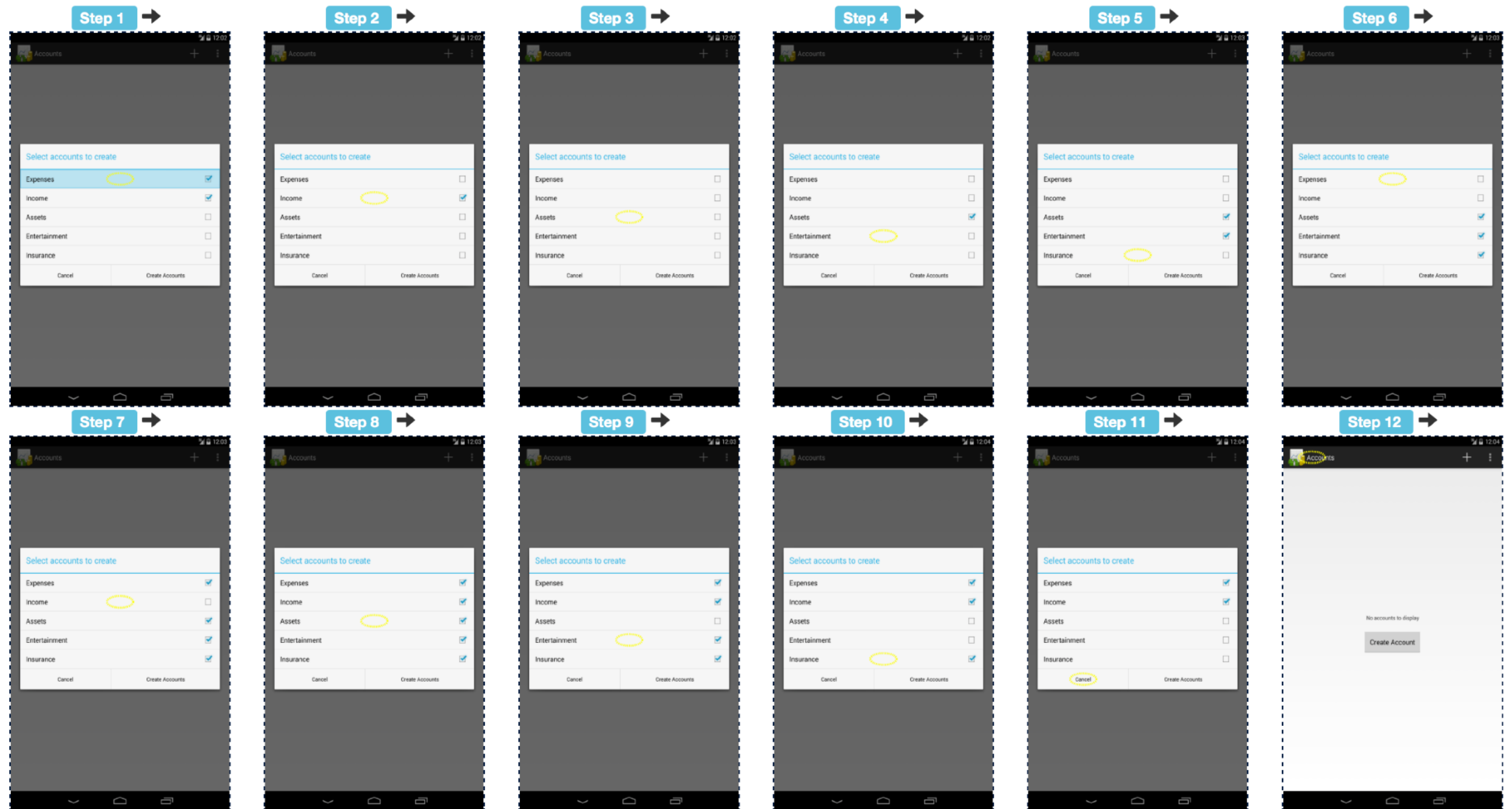
☐ Delete after export

Cancel

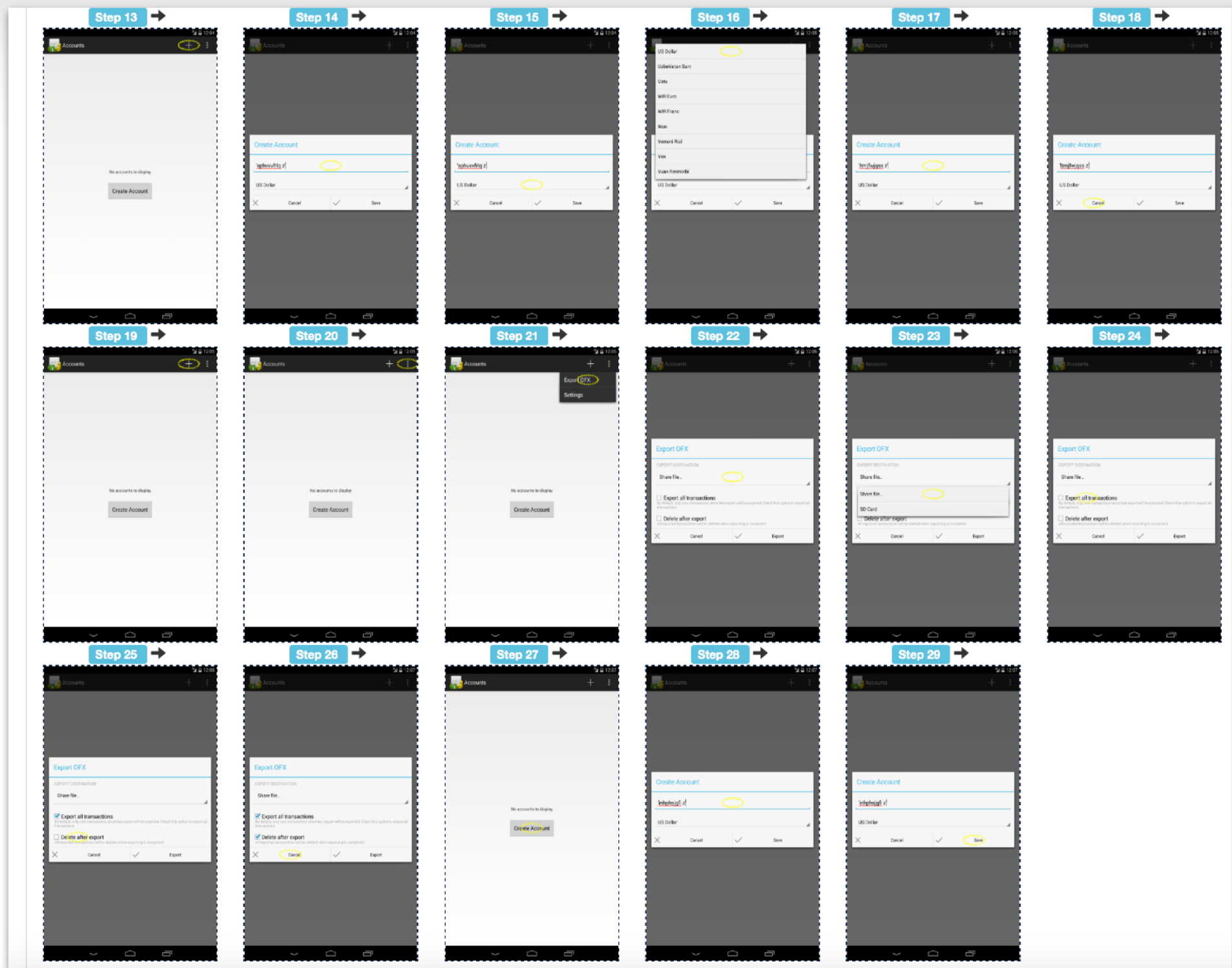
CRASHSCOPE: Reports

3 Crash Application Screen-Flow

[\(Go top\)](#)



CRASHSCOPE: Reports



CRASHSCOPE: Reports

4 Crash Pruned Stack Trace

[\(Go top\)](#)

```
E/SQLiteLog(17653): (1) near "inhphsjgf": syntax error
E/AndroidRuntime(17653): FATAL EXCEPTION: main
E/AndroidRuntime(17653): Process: org.gnucash.android, PID: 17653
E/AndroidRuntime(17653): android.database.sqlite.SQLiteException: near "inhphsjgf": syntax error (code 1): , while compiling: SELECT _id, uid FROM accounts WHERE uid = 'inhphsjgf-d724114f522e'
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteConnection.nativePrepareStatement(Native Method)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteConnection.acquirePreparedStatement(SQLiteConnection.java:889)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteConnection.prepare(SQLiteConnection.java:500)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteSession.prepare(SQLiteSession.java:588)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteProgram.(SQLiteProgram.java:58)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteQuery.(SQLiteQuery.java:37)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteDirectCursorDriver.query(SQLiteDirectCursorDriver.java:44)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteDatabase.rawQueryWithFactory(SQLiteDatabase.java:1314)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteDatabase.queryWithFactory(SQLiteDatabase.java:1161)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteDatabase.query(SQLiteDatabase.java:1032)
E/AndroidRuntime(17653):     at android.database.sqlite.SQLiteDatabase.query(SQLiteDatabase.java:1200)
E/AndroidRuntime(17653):     at org.gnucash.android.db.AccountsDbAdapter.getAccountID(AccountsDbAdapter.java:166)
E/AndroidRuntime(17653):     at org.gnucash.android.db.AccountsDbAdapter.addAccount(AccountsDbAdapter.java:76)
E/AndroidRuntime(17653):     at org.gnucash.android.ui.accounts.NewAccountDialogFragment$1.onClick(NewAccountDialogFragment.java:156)
E/AndroidRuntime(17653):     )
E/AndroidRuntime(17653):     at android.view.View.performClick(View.java:4438)
E/AndroidRuntime(17653):     at android.view.View$PerformClick.run(View.java:18422)
E/AndroidRuntime(17653):     at android.os.Handler.handleCallback(Handler.java:733)
E/AndroidRuntime(17653):     at android.os.Handler.dispatchMessage(Handler.java:95)
E/AndroidRuntime(17653):     at android.os.Looper.loop(Looper.java:136)
E/AndroidRuntime(17653):     at android.app.ActivityThread.main(ActivityThread.java:5001)
E/AndroidRuntime(17653):     at java.lang.reflect.Method.invokeNative(Native Method)
E/AndroidRuntime(17653):     at java.lang.reflect.Method.invoke(Method.java:515)
E/AndroidRuntime(17653):     at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:785)
E/AndroidRuntime(17653):     at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:601)
E/AndroidRuntime(17653):     at dalvik.system.NativeStart.main(Native Method)
```


Evaluation

- Two Empirical Studies
- Study 1: Crash Detection Capabilities
- Study 2: Crash Report Reproducibility and Readability

Study 1: Crash Detection & Coverage

- *RQ₁*: Crash Detection Effectiveness?
- *RQ₂*: Orthogonality of Crashes?
- *RQ₃*: Effectiveness of Individual Strategies?
- *RQ₄*: Does Crash Detection Correlate with Code Coverage?

Study 1: Experimental Setup

Tools Used In The
Comparative Fault
Finding Study

Tool Name	Android Version	Tool Type
Monkey	Any	Random
A3E Depth-First	Any	Systematic
GUI-Ripper	Any	Model-Based
Dynodroid	v2.3	Random-Based
PUMA	v4.1+	Random-Based

- 61 subject applications from the *Androtest*¹ toolset
- Each testing tool was run 5 separate times for 1 hour, whereas CrashScope ran through all strategies
- Monkey was limited by the number of events

¹S. R. Choudhary, A. Gorla, and A. Orso. Automated Test Input Generation for Android: Are we there yet? In 30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015), 2015

Study 1: Crash Results

Unique Crashes Discovered With Instrumented Crashes in Parentheses

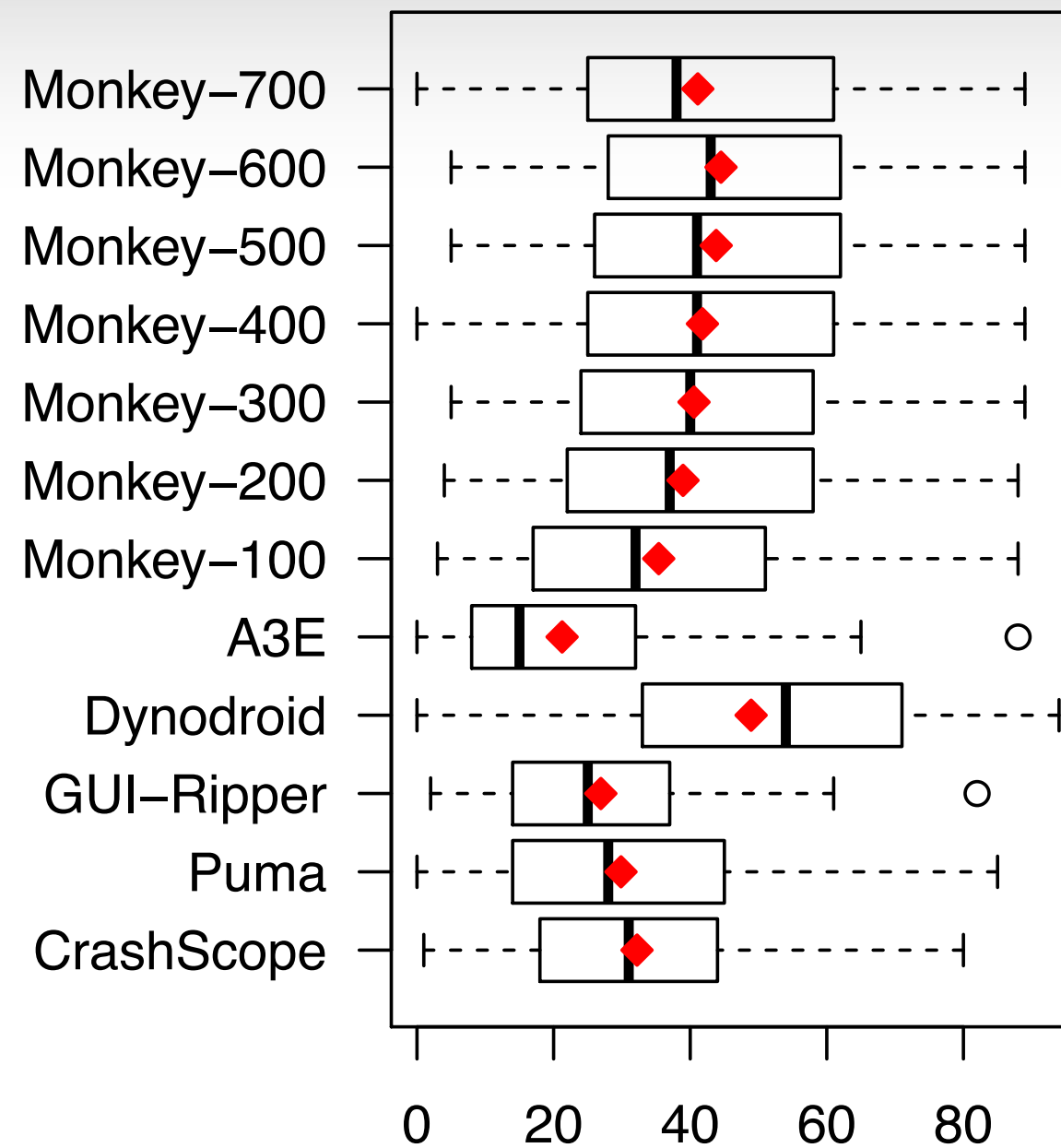
App	A3E	GUI- Ripper	Dynodroid	PUMA	Monkey (All)	CrashScope
A2DP Vol	1	0	0	0	0	0
aagtl	0	0	1	0	1	0
Amazed	0	0	0	0	1	0
HNDroid	1	1	1	2	1	1
BatteryDog	0	0	1	0	1	0
Soundboard	0	1	0	0	0	0
AKA	0	0	0	0	1	0
Bites	0	0	0	0	1	0
Yahtzee	1	0	0	0	0	1
ADSDroid	1	1	1	1	1	1
PassMaker	1	0	0	0	1	1
BlinkBattery	0	0	0	0	1	0
D&C	0	0	0	0	1	0
Photostream	1	1	1	1	1	0
AlarmKlock	0	0	1	0	0	0
Sanity	1	1	0	0	0	0
MyExpenses	0	0	1	0	0	0
Zooborns	0	0	0	0	0	2
ACal	1	2	2	0	1	1
Hotdeath	0	2	0	0	0	1
Total	8 (21)	9 (5)	9 (6)	4 (0)	12 (1)	8 (0)

Study 1: Crash Results

Unique Crashes Discovered With Instrumented Crashes in Parentheses

App	A3E	GUI- Ripper	Dynodroid	PUMA	Monkey (All)	CrashScope
A2DP Vol	1	0	0	0	0	0
aagtl	0	0	1	0	1	0
Amazed	0	0	0	0	1	0
HNDroid	1	1	1	2	1	1
BatteryDog	0	0	1	0	1	0
Soundboard	0	1	0	0	0	0
AKA	0	0	0	0	1	0
Bites	0	0	0	0	1	0
Yahtzee	1	0	0	0	0	1
ADSDroid	1	1	1	1	1	1
PassMaker	1	0	0	0	1	1
BlinkBattery	0	0	0	0	1	0
D&C	0	0	0	0	1	0
Photostream	1	1	1	1	1	0
AlarmKlock	0	0	1	0	0	0
Sanity	1	1	0	0	0	0
MyExpenses	0	0	1	0	0	0
Zooborns	0	0	0	0	0	2
ACal	1	2	2	0	1	1
Hotdeath	0	2	0	0	0	1
Total	8 (21)	9 (5)	9 (6)	4 (0)	12 (1)	8 (0)

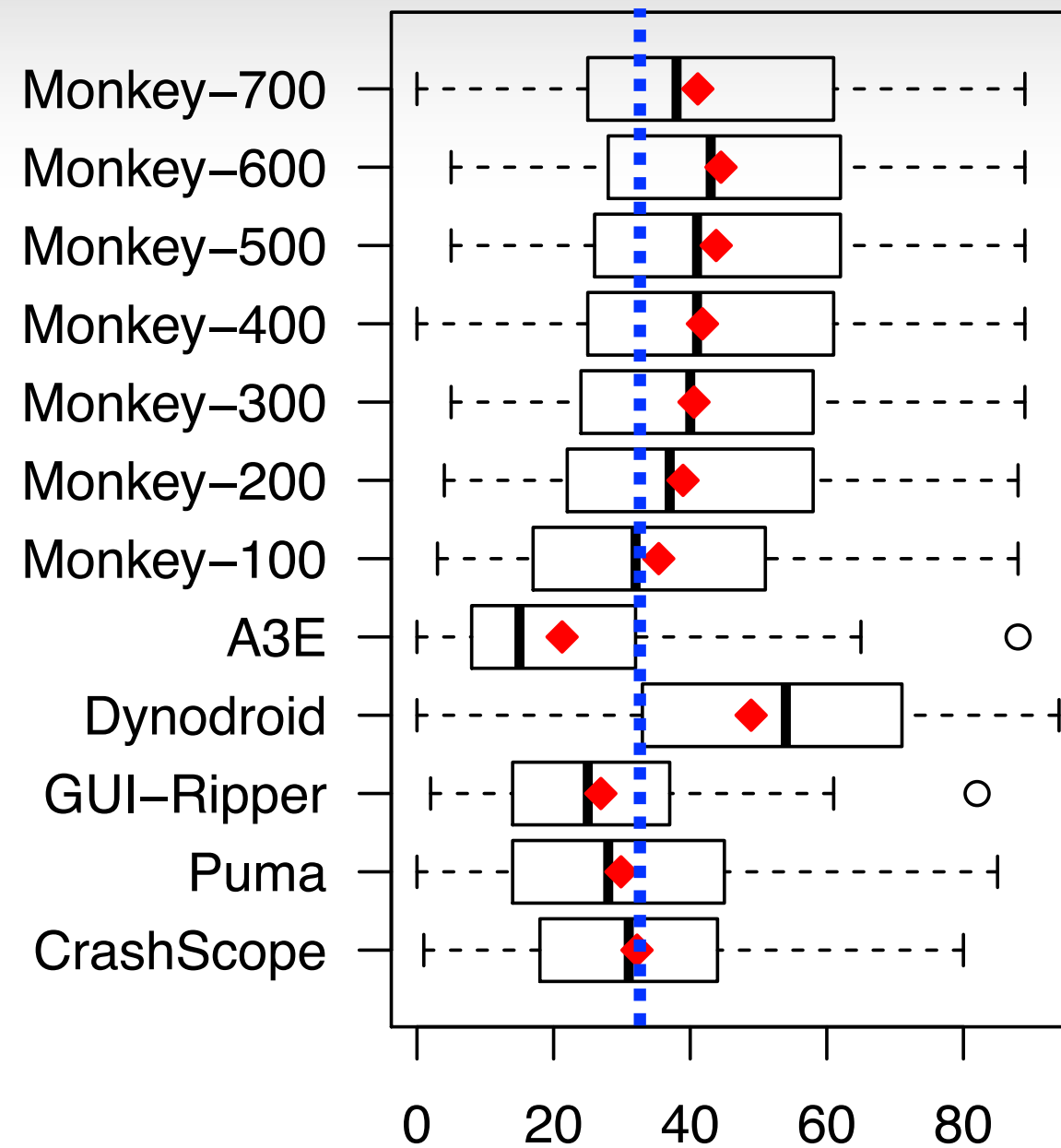
Study 1: Statement Coverage Results



Average Statement Coverage Results for the Comparative Study

Reported in Average %

Study 1: Statement Coverage Results



Average Statement Coverage Results for the Comparative Study

Reported in Average %

Study 1: Summary of Findings

- *RQ₁*: CrashScope is nearly as effective at discovering crashes as the other tools, without reporting crashes caused by instrumentation
- *RQ_{2&3}*: CrashScope's differing strategies led to the discovery of unique crashes
- *RQ₄*: Higher statement coverage does not necessarily correspond with crash detection capabilities

Study 2: Reproducibility & Readability

- *RQ₅*: Reproducibility of CrashScope Reports?
- *RQ₆*: Readability of CrashScope Reports?

Study 2: Experimental Setup

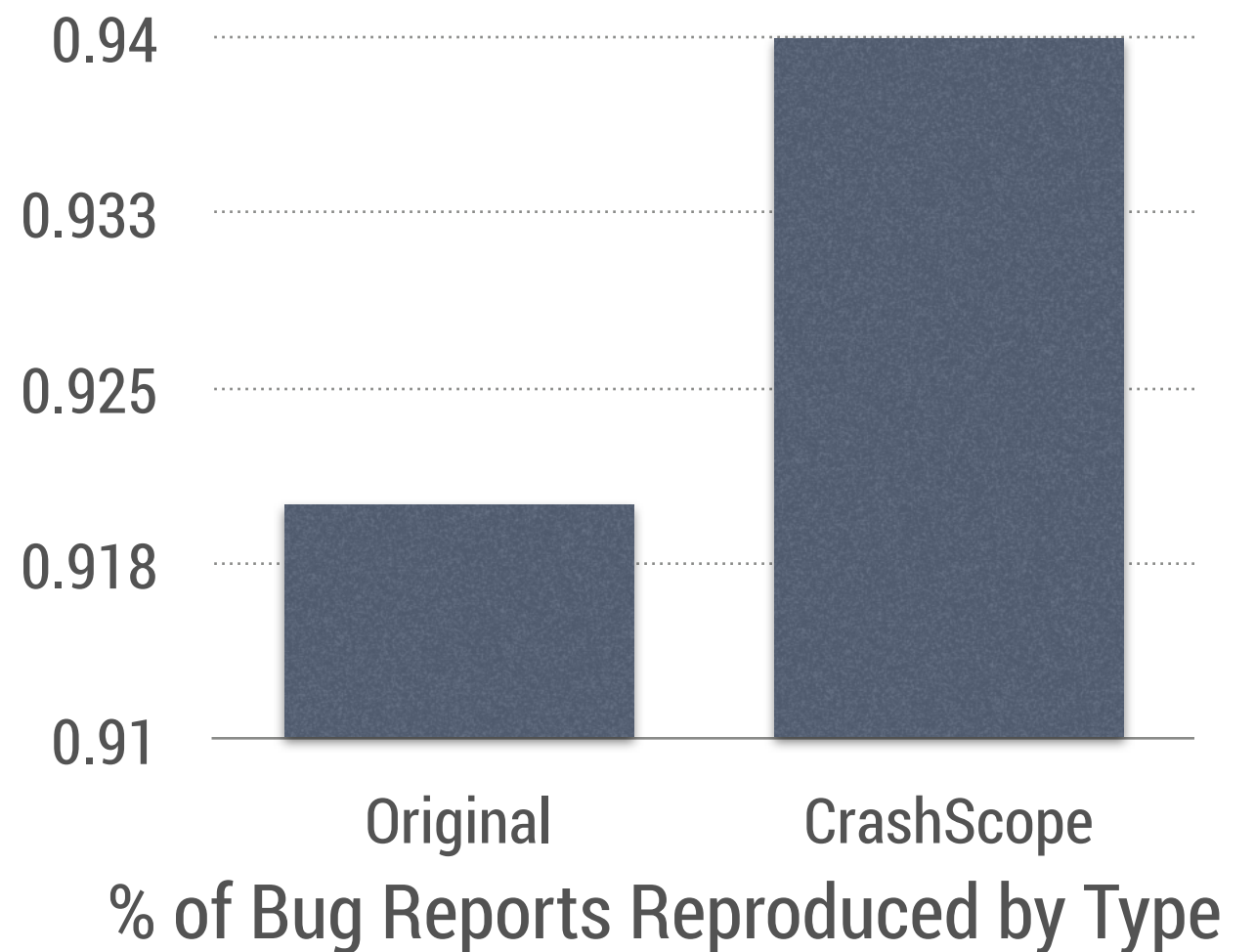
- *8 Real-World Crash Reports from Open Source Apps*
- *16 Graduate Students from the College of William & Mary*

Application Name	# of Reproduction Steps
BMI	4
Schedule	7
adsdroid	2
Anagram-solver	7
Eyecam	14
GNU Cash	29
Olam	2
CardGame Scores	23

- Each student attempted to reproduce 8 bugs: 4 from the original reports, 4 from CrashScope Reports
- Participants used a Nexus 7 tablet for reproduction

Study 2: Reproducibility Results

Type of Crash Report	# of Total/Non-Reproducible Reports
Original Bug Reports	59/64
CrashScope Bug Reports	60/64



Study 2: Readability Results

Question	CrashScope Mean	CrashScope StdDev	Original Mean	Original StdDev
UX1: I think I would like to have this type of bug report frequently.	4.00	0.89	3.06	0.77
UX2: I found this type of bug report unnecessarily complex.	2.81	1.04	2.125	0.96
UX3: I thought this type of bug report was easy to read/understand.	4.00	0.82	3.00	0.97
UX4: I found this type of bug report very cumbersome to read.	2.50	1.10	2.44	0.81
UX5: I thought the bug report was very useful for reproducing the crash.	4.13	0.62	3.44	0.89

Study 2: Readability Results

Question	CrashScope Mean	CrashScope StdDev	Original Mean	Original StdDev
UX1: I think I would like to have this type of bug report frequently.	4.00	0.89	3.06	0.77
UX2: I found this type of bug report unnecessarily complex.	2.81	1.04	2.125	0.96
UX3: I thought this type of bug report was easy to read/understand.	4.00	0.82	3.00	0.97
UX4: I found this type of bug report very cumbersome to read.	2.50	1.10	2.44	0.81
UX5: I thought the bug report was very useful for reproducing the crash.	4.13	0.62	3.44	0.89

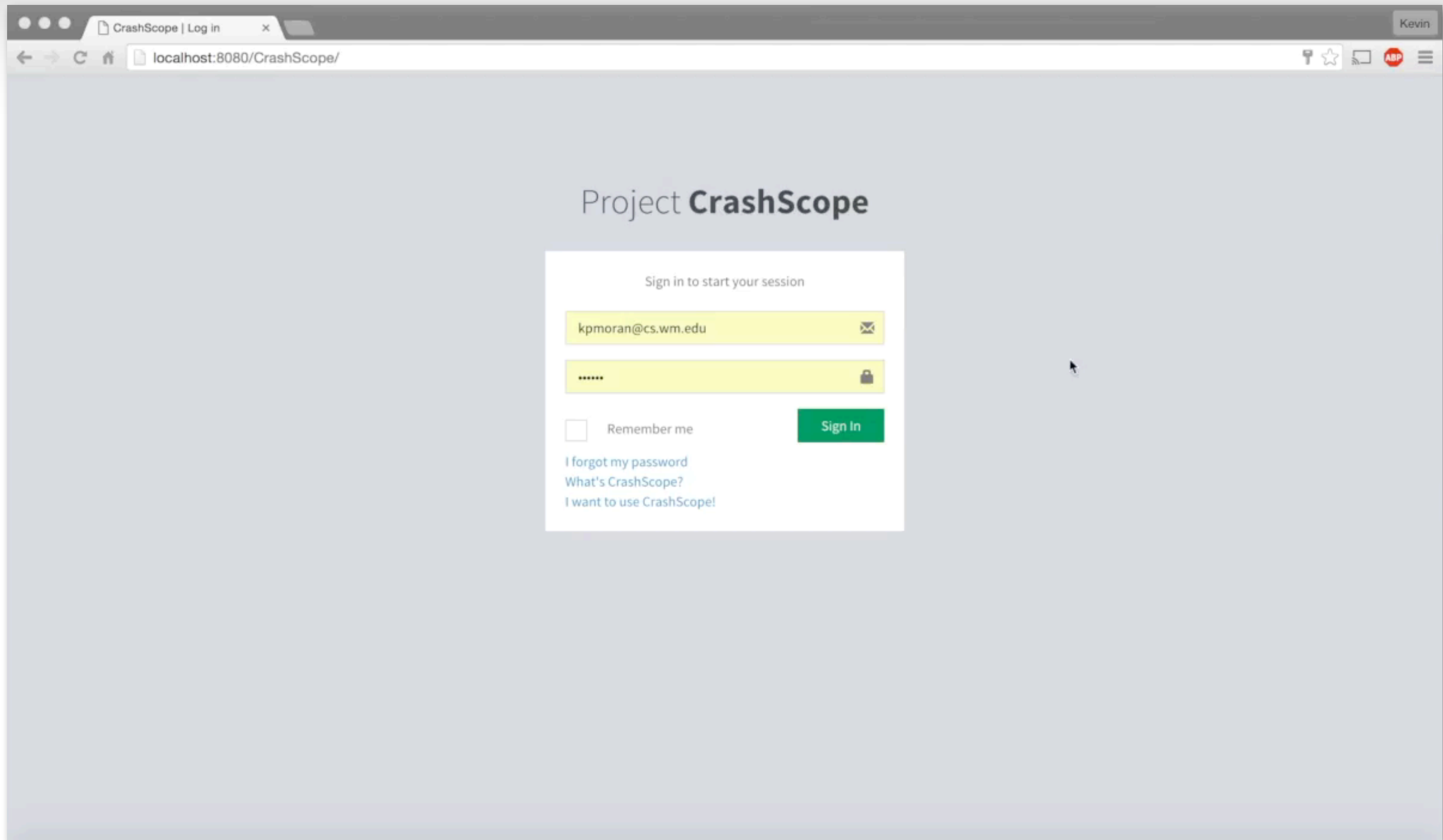
Study 2: Readability Results

Question	CrashScope Mean	CrashScope StdDev	Original Mean	Original StdDev
UX1: I think I would like to have this type of bug report frequently.	4.00	0.89	3.06	0.77
UX2: I found this type of bug report unnecessarily complex.	2.81	1.04	2.125	0.96
UX3: I thought this type of bug report was easy to read/understand.	4.00	0.82	3.00	0.97
UX4: I found this type of bug report very cumbersome to read.	2.50	1.10	2.44	0.81
UX5: I thought the bug report was very useful for reproducing the crash.	4.13	0.62	3.44	0.89

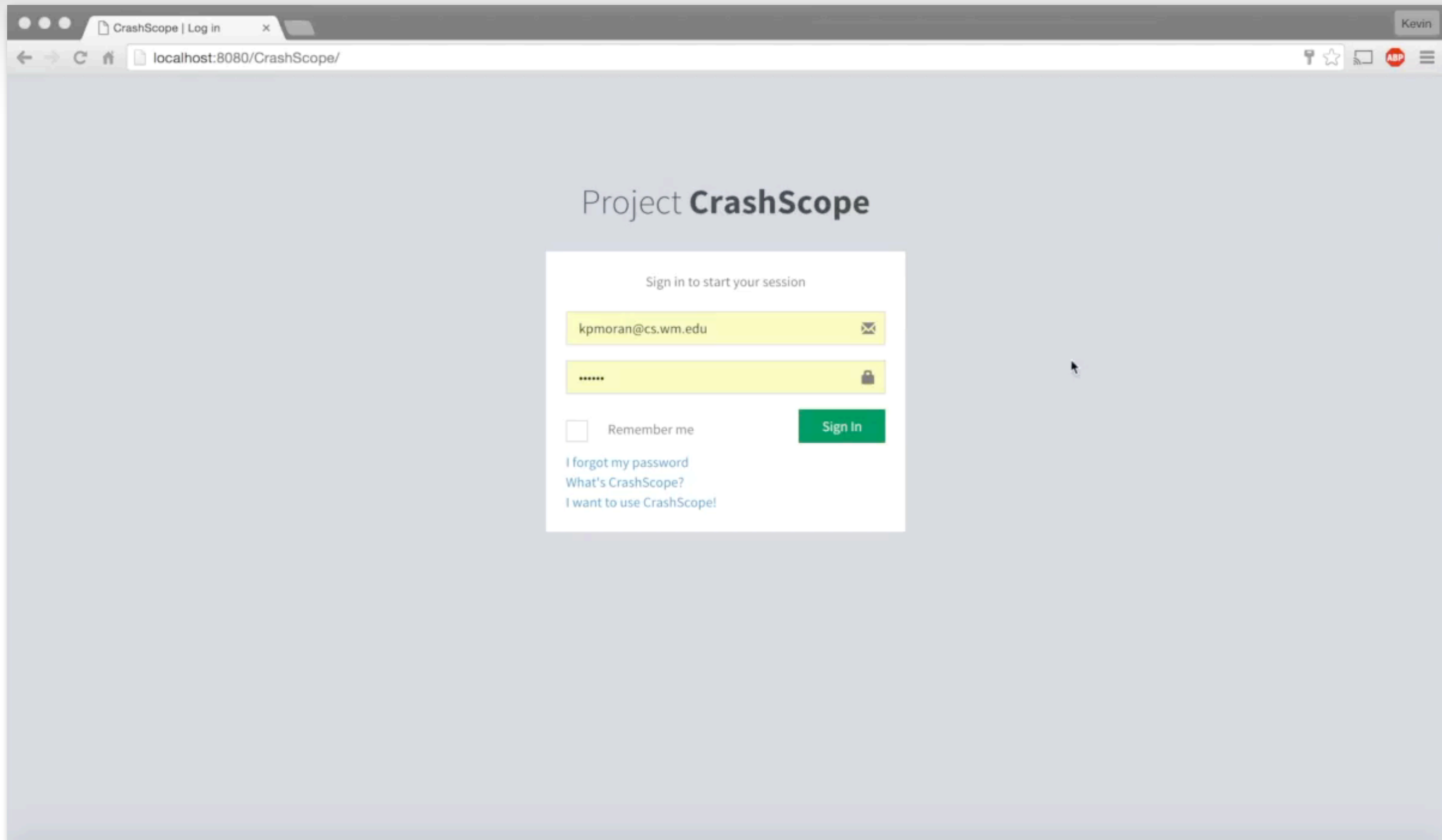
Study 2: Summary of Findings

- *RQ₅*: Reports generated by CrashScope are about as reproducible as human written reports extracted from open-source issue trackers
- *RQ₆*: Reports generated by CrashScope are more readable and useful from a developers' perspective compared to human-written reports.

CRASHSCOPE: A Practical Tool



CRASHSCOPE: A Practical Tool



THANK YOU !!

QUESTIONS/DISCUSSION?



kpmoran@gmu.edu

Hands-On Session

<https://sagelab.io/crashscope-tutorial/>



Discussion Questions

- Potential solutions to challenges we covered?
- Other future research directions?
- How can mobile testing techniques cope with AR/VR environments?