

TODO
2015-2016
TODO

836897 Simone Graziussi

TODO

Abstract

TODO

Contents

I	Introduction	3
1	Abstract	4
2	Testing	4
3	Android	4
4	Android Testing State of the Art	4
5	Introduction to Events	4
6	Background/Motivation	4
II	Lifecycle Testing	4
7	Lifecycle	4
7.1	Component Lifecycle	4
7.2	Lifecycle Problems	4
7.3	Existing Lifecycle Testing	4
8	Static Analysis	4
8.1	Static Program Analysis	4
8.2	Android Lint	4
8.3	Custom Android Lint Checks	4
8.4	Static Lifecycle Checks	4
8.4.1	Target Components	4
8.4.2	Design	4
8.4.3	Implementation	4
8.4.4	Evaluation	4
9	Dynamic Analysis	4
9.1	Design	4
9.2	Implementation	4
9.3	Evaluation	4
III	Event-based Testing	5

10 Event-based Systems	5
10.1 Events in Android	5
10.2 Problems with Events	5
10.3 Existing Event Testing	5
11 Temporal Assertions Language	5
11.1 Consistency Checks	5
11.2 Checks on Single Events	5
11.2.1 Can Happen Only After	5
11.3 Checks on Sets of Events	6
11.3.1 Must Happen After	6
11.4 Checks on the Whole Stream	8
11.5 Connectives between Checks	8
12 Reactive Programming	8
12.1 ReactiveX	8
12.2 Components	8
12.3 RxJava and RxAndroid	8
12.4 Events Observable in Android	8
13 Design	8
14 Implementation	8
15 Evaluation	8
IV Conclusion	8
16 Recap	8
17 Future Work	8

Part I

Introduction

- 1 Abstract
- 2 Testing
- 3 Android
- 4 Android Testing State of the Art
- 5 Introduction to Events
- 6 Background/Motivation

Part II

Lifecycle Testing

- 7 Lifecycle
 - 7.1 Component Lifecycle
 - 7.2 Lifecycle Problems
 - 7.3 Existing Lifecycle Testing
- 8 Static Analysis
 - 8.1 Static Program Analysis
 - 8.2 Android Lint
 - 8.3 Custom Android Lint Checks
 - 8.4 Static Lifecycle Checks
 - 8.4.1 Target Components
 - 8.4.2 Design
 - 8.4.3 Implementation
 - 8.4.4 Evaluation
- 9 Dynamic Analysis
 - 9.1 Design
 - 9.2 Implementation
 - 9.3 Evaluation

Part III

Event-based Testing

10 Event-based Systems

10.1 Events in Android

10.2 Problems with Events

10.3 Existing Event Testing

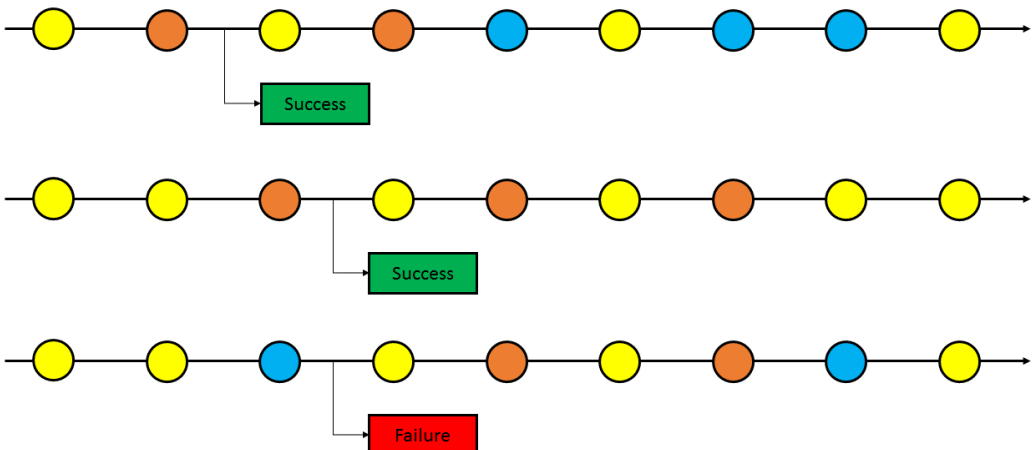
11 Temporal Assertions Language

11.1 Consistency Checks

11.2 Checks on Single Events

11.2.1 Can Happen Only After

Structure	<code>anEventThat(m1).canHappenOnlyAfter(anEventThat(m2))</code>
Description	Checks that the events that match <code>m1</code> happen only after any event that matches <code>m2</code> , i.e. there cannot be an event that matches <code>m1</code> before the first event that matches <code>m2</code> .
FOL	$\forall e1 \left(match(e1, m1) \Rightarrow \exists e2 \left(match(e2, m2) \wedge before(e2, e1) \right) \right)$

Visual	<p>anEventThat(●).canHappenOnlyAfter(anEventThat(●))</p>  <p>Note in particular that in the second case the checks succeeds even if no blue event occurs: we are just saying that blue events <i>can</i> happen after an orange one.</p>
Code Example	<pre>anEventThat(isMarkerPlacement()) .canHappenOnlyAfter(anEventThat(isMapReady()))</pre>

11.3 Checks on Sets of Events

11.3.1 Must Happen After

Structure	<pre>exactly(n).eventsWhereEach(m1).mustHappenAfter(anEventThat(m2))</pre>
-----------	--

Description	<p>Checks that exactly n events that match $m1$ happen exclusively after every event that matches $m2$. “Exclusively” means that there cannot be another event that matches $m2$ before the sequence of n events is completed.</p>
FOL	$\forall e2 \left(match(e2, m2) \Rightarrow \exists_n e1 \left(match(e1, m1) \wedge before(e2, e1) \right. \right. \\ \left. \left. \wedge \neg \exists e2' \left(match(e2', m2) \wedge between(e2, e2', e1) \right) \right) \right)$
Visual	<p>exactly(2).eventsWhereEach(●).mustHappenAfter(anEventThat(●))</p> <p>Note in particular that the third case shows the “exclusively” constraint mentioned before: the check fails because we do not have two blue events after the first orange but before the second one (i.e. the first orange event does <i>not</i> match one of the three blue events that follow the second orange event)</p>

Code Example	<pre>exactly(1).eventsWhereEach(isTextChangeFrom(locationTextView)) .mustHappenAfter(anEventThat(isLocationChange()))</pre>
--------------	---

11.4 Checks on the Whole Stream

11.5 Connectives between Checks

12 Reactive Programming

12.1 ReactiveX

12.2 Components

12.3 RxJava and RxAndroid

12.4 Events Observable in Android

13 Design

14 Implementation

15 Evaluation

Part IV

Conclusion

16 Recap

17 Future Work

References