Android 底層事件轉換為關鍵字驅動測試腳本之設計與實作
Design and Implementation of a Tool for Converting Android Low-Level Events into Keyword-Driven Testing Scripts

劉建宏　陳偉凱　黃映瑞　陳季宣
Chien-Hung Liu , Woie-Kae Chen,　Ying-jui Huang, Chi-Hsuan Chen
國立臺北科技大學資訊工程系
Department of Computer Science and Information Engineering,
National Taipei University of Technology
Email：{cliu, wkchen, t100599006, t101598044}@ntut.edu.tw

摘　要

近年來 Android 應用程式的數量快速成長，這些 Android 應用程式經常使用大量的觸控手勢及各種硬體感測器與操作者互動，因此 Android 應用程式測試方式不同於傳統的應用程式。現有的 Android 錄製與播放測試工具大多無法重播較複雜的手勢或僅能對 GUI 元件進行操作。本論文提出一個錄製及播放的 Android 自動化測試方法，透過分析 Android 底層的輸入事件記錄來辨識使用者的操作。一段 Android 底層事件將被分類及辨識為一個觸控手勢、硬體按鍵操作或是其他的感測器事件。此方法並將辨識結果轉換為相對應的關鍵字並產生關鍵字驅動測試腳本，使得使用者可以簡單的對錄製結果進行修改及插入斷言（assertion）。在播放測試腳本時，腳本內的關鍵字會被重製成在目標裝置上描述原先使用者操作的 Android 底層事件並注入 Android 輸入子系統。透過這個方法，複雜的手勢、非 GUI 元件的操作及感測器事件可以在不同的裝置上被重播。此外，本論文實作了一個工具，藉由此工具的使用案例可以顯示此方法之實用性。透過此工具使用者可以有效降低測試 Android 應用程式所需的時間與成本。

關鍵字： Android 測試、錄製與重播、關鍵字驅動測試

一、前言

近年來行動裝置越來越普及，其中 Android 系統是市佔率最高的平台，因此 Android 應用程式的數量快速成長。這些應用程式的操作方式不同於傳統電腦，大量的使用觸控螢幕、手勢及各種硬體感測器，使得其測試方式也不同於傳統的應用程式。

現有具錄製與重播功能的 Android 測試工具雖然可以幫助我們得到 Android UI 元件資訊，甚至能以關鍵字的方式紀錄 Android UI 元件的操作。但有許多如繪圖、影像處理等 Android 應用程式其操作介面並不使用 Android UI 元件，而是使用其他繪圖引擎所繪製介面讓使用者操作，這些應用程式的操作將無法被正確的錄製及重播。除此之外，這些現有的測試工具幾乎只能紀錄及重播基本的點擊、滑動與拖曳，少數可以支援縮放及旋轉，能支援其他複雜的手勢如不規則滑動、多點觸控及硬體感測器相關事件，只能透過 Android 底層事件錄製使用者操作的 RERAN [1]，但其所錄製的腳本是低階的事件描述，一般使用者難以閱讀維護，且無法於腳本中插入斷言等測試所需之指令。

本論文針對 Android 系統及應用程式，提出一個方法及實作此工具，由 Android 底層的輸入事件轉換為關鍵字驅動測試腳本，讓使用者可以簡單的對錄製結果進行修改及插入斷言。我們對於使用者操作進行分析及辨認，其中較為簡單的手勢（如點擊和滑動）及按鍵事件，將簡化為對應 UiAutomator [2]的高階關鍵字，以利於維護。而對於複雜的手勢（如不規則滑動與多指觸控）、非 GUI 元件的操作及硬體感測器事件，我們的關鍵字資料保留了足以重製低階事件的資訊，使得複雜的操作能完整重播。

本論文組織架構如下：第一節為本論文的簡介。第二節為 Android 底層輸入事件介紹及相關文獻探討。第三節說明 Android 輸入事件轉換為關鍵字之方法。第四節為關鍵字測試腳本的執行及播放方法。第五節說明本論文提出之工具的架構與實作，及案例展示。第六節為結論與未來研究方向。

二、背景介紹及文獻探討

2.1　相關論文研究及工具

Gomez [1]等人提出了一個 Android 錄製與播放方法，並實作一個錄製播放工具 RERAN (Record and Replay for Android)。RERAN 利用 Android SDK 提供的 getevent 工具取得裝置上的輸入事件記錄，將其轉換為 sendevent 工具所需參數的格式後傳送至裝置上，再於裝置上執行重播程式，重建所記錄的輸入事件，藉此重播一連串的使用者操作。

Kuo [3]提出一個 Android 錄製重播之測試方法，並實作測試工具 ACRT (Android Capture and Replay testing Tool)。此工具利用 Android 系統中的穿插(Instrumentation)功能將事件攔截模組插入於 Android 應用程式，藉由攔截模組捕捉應用程式的 UI 元件資訊及攔截使用者的操作事件，達到錄製測試操作行為之目的。

Halpern [4]等人提出了一個跨裝置的 Android 多指觸控的錄製及播放方法，並實作一個錄製及播放工具 Mosaic。Mosaic 分析 Android 底層的觸控事件並將這些底層事件簡化為按下、釋放及移動三種動作的描述的腳本。在播放時 Mosaic 會將腳本內所記錄的移動座標與目標播放裝置的硬體規格作對應，使 RERAN 的播放方法能用於跨裝置播放。

Takala [5]等人提出一個 Android 應用程式的 Model-based 圖形使用者介面測試工具 TEMA。此工具使用 Monkey 隨機產生的使用者事件操作各項 UI 元件，幫助 Android 應用程式建立狀態機模型，再將這些操作過程以關鍵字的方式紀錄，產生直觀易懂的測試腳本。

Pajunen [6]等人將 TEMA 圖形使用者介面測試工具和 Robot Framework 關鍵字驅動測試自動化框架予以整合，使得 TEMA 工具能夠使用 Robot Framework 所支援的 wide-library，因此擴大其可利用之範圍。

Wu [7]等人提出一個 Android 應用程式的關鍵字驅動測試框架 AKDT。此工具基於 Android testing framework [8]及 Robotium [9]測試框架，在其上設計各種可重複使用的關鍵字庫，將測試資料及測試腳本邏輯分開，以簡化測試案例的設計。藉此提高測試腳本的可重用性，減少測試人員編寫測試腳本的成本。

Chen [10]提出了一個適用於Android應用程式的關鍵字測試函式庫及對應的測試工具—RobotDroid，此工具進行測試時將 UI 事件及非 UI 事件的執行方式分開，前者利用 UiAutomator 執行，後者則藉由預先植入待測裝置的 RobotDroid Agent 對裝置進行操作。藉此使關鍵字驅動測試的應用範圍能觸及 GPS 操作及驗證硬體感測器狀態等行為。

MonkeyTalk [11]為 Gorilla Logic 公司所開發的行動裝置測試工具，可運行於 Android 及 iOS 平台。MonkeyTalk 將 MonkeyTalk agent 和應用程式原始程式碼一起建置打包，使應用程式擁有紀錄使用者操作各項元件事件之能力，並可對元件發出操作事件，藉此達到錄製及重播使用者操作之目的。以此種方式錄製使用者操作必須擁有待測程式的原始程式碼，且可錄製的事件也僅限於待測程式內。

Robotium Recorder [12]為 Robotium Tech 公司所推出的商業產品，可讓使用者以錄製的方式產生 Robotium 的測試腳本。Robotium Recorder 利用 Android 提供的 XML 佈局對 Android UI 元件進行定位及操作，並可於錄製中利用 UI 元件或行動裝置的螢幕截圖插入測試斷言。

2.2 Android 輸入事件背景

Android 底層的輸入事件產生基於 Linux Input Subsystem，主要有 Input driver、Input core、Event handler 三個組件。Input driver 負責驅動硬體裝置並且向 Input core 註冊一個 input_dev 以便系統與裝置溝通，Event handler 負責事件的處理，將收到的事件轉換成標準的 event 格式並寫入 user space 相應的 event files，Input core 則是 Input driver 與 Event handler 之間的溝通媒介，負責正確的讓 Input driver 接收到的硬體資料對應到相應的 Event handler 處理。圖 1 表示了 Android 輸入子系統的架構及輸入事件的產生方式。

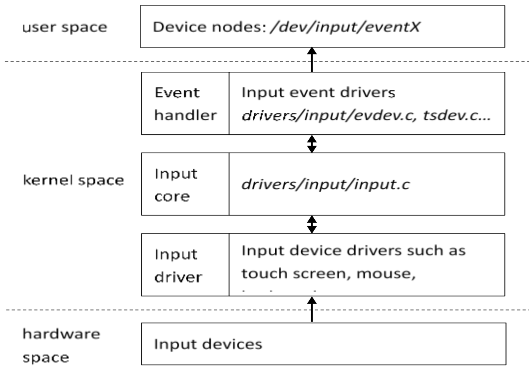


圖 1　Android (Linux) 輸入子系統架構

Android 輸入事件相關定義位於標頭檔 linux/input.h 內，我們可以使用 Android SDK 中所提供的 getevent 工具取得這些事件資料。

從圖 2 我們可以看到一個觸控螢幕的點擊動作是由數個 Android 的底層輸入事件組成，在此層級上，從單一的輸入事件沒有辦法直接得知使用者在觸控螢幕上進行了何種手勢操作。

time (sec.μs)　device file　type　code　value

```
1 [   217231.410036] /dev/input/event0: EV_ABS   ABS_MT_TRACKING_ID   0000834
2 [   217231.410101] /dev/input/event0: EV_ABS   ABS_MT_TOUCH_MAJOR   0000000
3 [   217231.410107] /dev/input/event0: EV_ABS   ABS_MT_PRESSURE      0000002
4 [   217231.410112] /dev/input/event0: EV_ABS   ABS_MT_POSITION_X    0000027
5 [   217231.410116] /dev/input/event0: EV_ABS   ABS_MT_POSITION_Y    0000036
6 [   217231.410123] /dev/input/event0: EV_SYN   SYN_REPORT           0000000
7 [   217231.462517] /dev/input/event0: EV_ABS   ABS_MT_TRACKING_ID   fffffff
8 [   217231.462520] /dev/input/event0: EV_SYN   SYN_REPORT           0000000
```

圖 2　使用 getevent 工具抓取的點擊事件

由 user space 抓取到的每個 Android 底層輸入事件擁有標準的 5 個欄位([timestamp] device: type code value)，timestamp 欄位為從裝置開機以來到事件發生時的時間。device 欄位為事件經 Android Input Subsystem 處理後，在 user space 所對應的 device file，其中的檔案編號會因裝置的不同而有所差異。type 欄位為輸入事件的種類，與 device driver 向 Input core 註冊的 input_dev 有關。code 欄位的功能依 event type 而異，在行動裝置觸控事件 EV_ABS 下用來分辨裝置回報數值的意義，在按鍵事件 EV_KEY 下則是用於顯示按鍵的名稱。最後的 value 欄位則是事件的 16 進位數值。

三、Android 輸入事件轉換與關鍵字

本論文所提出的方法分為錄製及播放兩部分，本節將說明 Android 底層的輸入事件轉換為關鍵字之方法。關鍵字腳本的播放方法則在第 4 節介紹。

為了更容易了解各種Android輸入事件的轉換流程及方法，在介紹事件轉換流程之前我們會先介紹我們的關鍵字分類、設計以及做為轉換輸出的關鍵字測試腳本格式。

3.1 常見使用者操作及其對應之關鍵字

行動裝置通常仰賴觸控螢幕的手勢以及少數實體按鍵操作。對於觸控螢幕操作來說，點擊、長按、滑動及拖曳是其中最基本的元素，任何更複雜的手勢都是由這幾種操作組合而成，因此我們首先為這些操作建立關鍵字。在按鍵操作方面，使用者的輸入可能為短按、長按或是組合鍵輸入，我們也為這些操作建立相應的關鍵字。

基於使用者操作對象及Android底層事件轉換方式的不同，我們將測試關鍵字分為4類，分別為觸控手勢操作、按鍵操作、硬體感測器及流程控制關鍵字。依據關鍵字播放及執行的方式我們又可將關鍵字分為高階操作，低階操作及控制流程這 3 種類型，關於播放類型的差異我們將留至第 4 節說明。表 1 為目前我們的系統所支援的測試關鍵字。

表 1　測試關鍵字一覽

關鍵字	關鍵字類型	播放類型	說明
Click	觸控手勢	高階操作	點擊
LongClick	觸控手勢	高階操作	長按
Swipe	觸控手勢	高階操作	滑動
Drag	觸控手勢	高階操作	拖曳
SingleFreeSwipe	觸控手勢	低階操作	單點不規則滑動
SingleFreeDrag	觸控手勢	低階操作	單點不規則拖曳
MultiTouch	觸控手勢	低階操作	多點觸控
Press	按鍵操作	高階操作	實體按鍵輸入
LongPress	按鍵操作	低階操作	長按一實體按鍵
ComboKey	按鍵操作	低階操作	實體組合鍵操作
LowLevel	硬體感測器	低階操作	尚未支援辨識之低階事件
Sleep	流程控制	流程控制	等待一段時間
Wait	流程控制	流程控制	等待 UI 元件出現
Assert	流程控制	流程控制	測試斷言
Play	流程控制	流程控制	播放一段預錄好的腳本

為了讓使用者易於編輯與閱讀，我們使用JSON格式來描述關鍵字測試腳本。一個測試關鍵字分為action及args兩部分，action為使用者操作或是執行測試時所需的動作名稱，args則為執行時所需的參數及測試資料。

3.2 Android 底層事件轉換關鍵字流程

圖3為Android底層事件轉換為關鍵字之活動圖。首先我們會依序讀入事件記錄檔，並在讀入的同時將底層事件分類為觸控事件、按鍵事件或其他硬體事件，之後將不同類別的底層事件交由不同的eventAnalyzer 處理，eventAnalyzer 在判斷出一段使用者操作終結後便會將這段操作轉換為關鍵字並寫入 keywordList，最後在事件紀錄檔中所有的低階事件皆處理完畢後，我們將 keywordList 序列化，產生 JSON 格式的關鍵字測試腳本。

3.3 觸控手勢關鍵字之轉換

一個觸控操作在Android底層是由數個輸入事件組成，我們無法直接由底層事件資訊得知使用者進行了何種手勢操作，為了使不同的觸控手勢操作能轉換為不同的關鍵字，我們必須對這些底層輸入資訊進行手勢辨認。

表 2　觸控操作相關底層事件

Event Type	Event Code	說明
EV_ABS	ABS_MT_TRACKING_ID	一段觸控軌跡的開始或結束
EV_ABS	ABS_MT_SLOT	觸控軌跡的編號
EV_ABS	ABS_MT_POSITION_X	觸控點的 X 座標
EV_ABS	ABS_MT_POSITION_Y	觸控點的 Y 座標
EV_ABS	ABS_MT_TOUCH_MAJOR	接觸範圍的長軸
EV_ABS	ABS_MT_TOUCH_MINOR	接觸範圍的短軸
EV_ABS	ABS_MT_WIDTH_MAJOR	觸控工具的長軸
EV_ABS	ABS_MT_WIDTH_MINOR	觸控工具的短軸
EV_ABS	ABS_MT_ORIENTATION	接觸範圍的方向
EV_ABS	ABS_MT_TOOL_TYPE	觸控工具的類型
EV_ABS	ABS_MT_BLOB_ID	用以描述接觸範圍多邊形的編號
EV_ABS	ABS_MT_PRESSURE	觸控點的壓力值
EV_ABS	ABS_MT_DISTANCE	觸控面到觸控工具的距離
EV_SYN	SYN_REPORT	同步事件

所有觸控螢幕的相關操作在Android底層的事件處理時都會以 event type 為 EV_ABS 且 event code 為 ABS_MT_*的觸控事件或 event type 為 EV_SYN 且 event code 為 SYN_REPORT 的同步事件回報，表 2 顯示在 Android 底層事件中與觸控操作有關的事件類別，其中 ABS_MT_TRACKING_ID 、 ABS_MT_SLOT 、 ABS_MT_POSITION_X、ABS_MT_POSITION_Y 及 SYN_REPORT 這 5 種類型的低階事件是我們在進行手勢辨認時必須處理的事件類別，其他則是選用或是與觸控操作軌跡無關的事件類別，因此不需對這些事件進行處理。圖 4 為一段觸控手勢操作之底層事件紀錄。

由圖4的底層事件紀錄我們可以看到第1行及第 9 行兩個帶有非 ffffffff 值的 ABS_MT_TRACKING_ID 事件，分別代表兩段觸控軌跡的開始，第 24 行及 26 行帶有 ffffffff 值的 ABS_MT_TRACKING_ID 事件代表兩段軌跡的結束，之後接著 27 行的 SYN_REPORT 事件代表整段操作結束。操作中的 ABS_MT_POSITION_X 及 ABS_MT_POSITION_Y 事件回報觸控操作的點座標資訊，ABS_MT_SLOT 則回報不同段軌跡資訊的變換，我們可以利用這些資訊對這段操作進行手勢辨認。

由於一個觸控手勢操作始於一個帶有非 ffffffff 值的 ABS_MT_TRACKING_ID 事件，終止於一個帶有 ffffffff 值的 ABS_MT_TRACKING_ID 事件再接著一個 SYN_REPORT 的同步資訊，我們可以利用此特性將一連串的Android底層輸入事件組合切割成不同段的手勢操作。

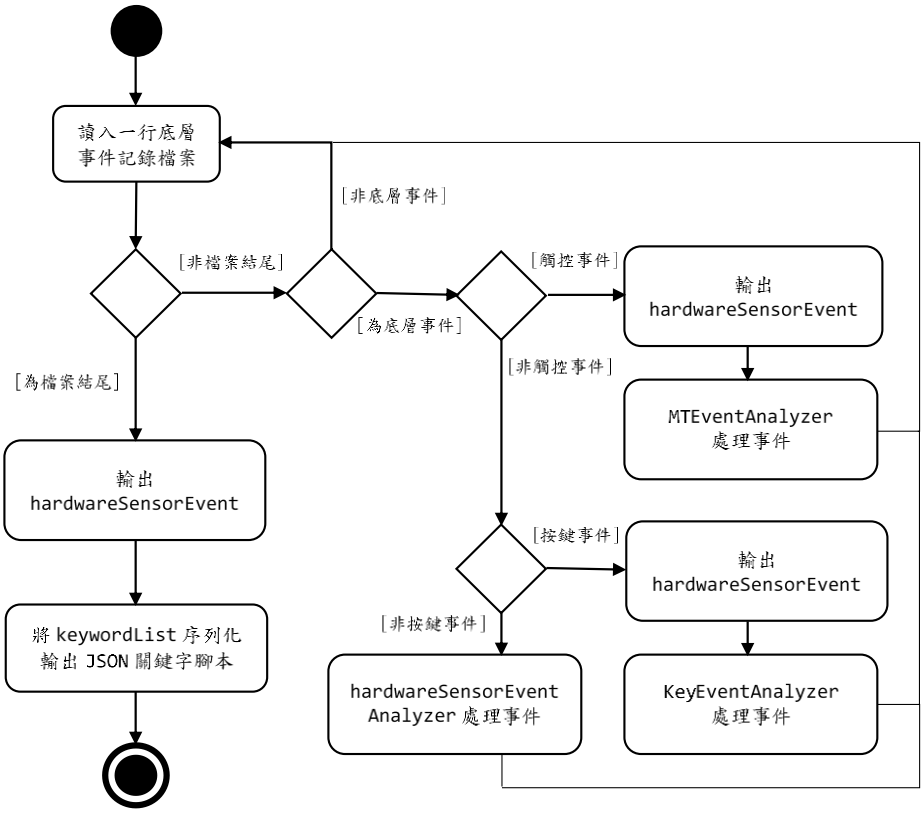
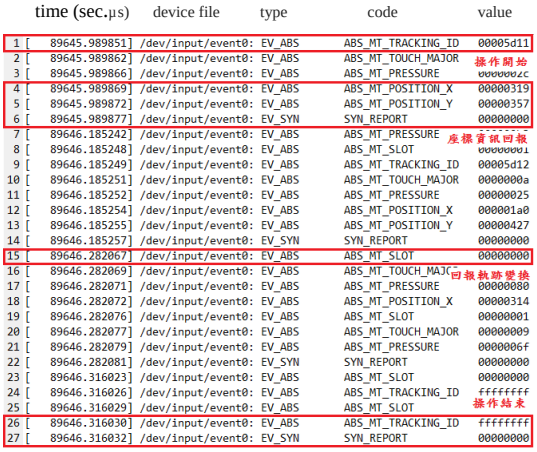
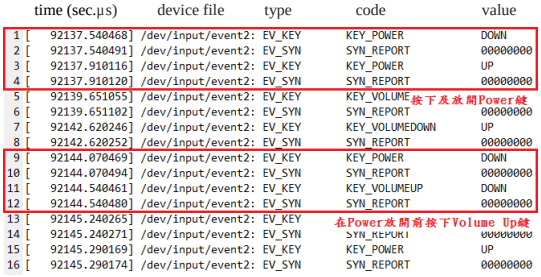


圖 3　Android 底層事件轉換為關鍵字活動圖



	time (sec.µs)	device file	type	code	value
1	[89645.989851]	/dev/input/event0:	EV_ABS	ABS_MT_TRACKING_ID	00005d11
2	[89645.989862]	/dev/input/event0:	EV_ABS	ABS_MT_TOUCH_MAJOR	操作開始
3	[89645.989866]	/dev/input/event0:	EV_ABS	ABS_MT_PRESSURE	000000zc
4	[89645.989869]	/dev/input/event0:	EV_ABS	ABS_MT_POSITION_X	00000319
5	[89645.989872]	/dev/input/event0:	EV_ABS	ABS_MT_POSITION_Y	00000357
6	[89645.989877]	/dev/input/event0:	EV_SYN	SYN_REPORT	00000000
7	[89646.185242]	/dev/input/event0:	EV_ABS	ABS_MT_PRESSURE	座標資訊回報
8	[89646.185248]	/dev/input/event0:	EV_ABS	ABS_MT_SLOT	00000001
9	[89646.185249]	/dev/input/event0:	EV_ABS	ABS_MT_TRACKING_ID	00005d12
10	[89646.185251]	/dev/input/event0:	EV_ABS	ABS_MT_TOUCH_MAJOR	0000000a
11	[89646.185252]	/dev/input/event0:	EV_ABS	ABS_MT_PRESSURE	00000025
12	[89646.185254]	/dev/input/event0:	EV_ABS	ABS_MT_POSITION_X	000001a0
13	[89646.185255]	/dev/input/event0:	EV_ABS	ABS_MT_POSITION_Y	00000427
14	[89646.185257]	/dev/input/event0:	EV_SYN	SYN_REPORT	00000000
15	[89646.282067]	/dev/input/event0:	EV_ABS	ABS_MT_SLOT	00000000
16	[89646.282069]	/dev/input/event0:	EV_ABS	ABS_MT_TOUCH_MAJ	回報執跡變換
17	[89646.282071]	/dev/input/event0:	EV_ABS	ABS_MT_PRESSURE	00000080
18	[89646.282072]	/dev/input/event0:	EV_ABS	ABS_MT_POSITION_X	00000314
19	[89646.282076]	/dev/input/event0:	EV_ABS	ABS_MT_SLOT	00000001
20	[89646.282077]	/dev/input/event0:	EV_ABS	ABS_MT_TOUCH_MAJOR	00000009
21	[89646.282079]	/dev/input/event0:	EV_ABS	ABS_MT_PRESSURE	0000006f
22	[89646.282081]	/dev/input/event0:	EV_SYN	SYN_REPORT	00000000
23	[89646.316023]	/dev/input/event0:	EV_ABS	ABS_MT_SLOT	00000000
24	[89646.316026]	/dev/input/event0:	EV_ABS	ABS_MT_TRACKING_ID	ffffffff
25	[89646.316029]	/dev/input/event0:	EV_ABS	ABS_MT_SLOT	操作結束
26	[89646.316030]	/dev/input/event0:	EV_ABS	ABS_MT_TRACKING_ID	ffffffff
27	[89646.316032]	/dev/input/event0:	EV_SYN	SYN_REPORT	00000000

圖 4　一段觸控手勢操作之底層事件紀錄



	time (sec.µs)	device file	type	code	value
1	[92137.540468]	/dev/input/event2:	EV_KEY	KEY_POWER	DOWN
2	[92137.540491]	/dev/input/event2:	EV_SYN	SYN_REPORT	00000000
3	[92137.910116]	/dev/input/event2:	EV_KEY	KEY_POWER	UP
4	[92137.910120]	/dev/input/event2:	EV_SYN	SYN_REPORT	00000000
5	[92139.651055]	/dev/input/event2:	EV_KEY	KEY_VOLUME	按下及放開Power鍵
6	[92139.651102]	/dev/input/event2:	EV_SYN	SYN_REPORT	00000000
7	[92142.620246]	/dev/input/event2:	EV_KEY	KEY_VOLUMEDOWN	UP
8	[92142.620252]	/dev/input/event2:	EV_SYN	SYN_REPORT	00000000
9	[92144.070469]	/dev/input/event2:	EV_KEY	KEY_POWER	DOWN
10	[92144.070494]	/dev/input/event2:	EV_SYN	SYN_REPORT	00000000
11	[92144.540461]	/dev/input/event2:	EV_KEY	KEY_VOLUMEUP	DOWN
12	[92144.540480]	/dev/input/event2:	EV_SYN	SYN_REPORT	00000000
13	[92145.240265]	/dev/input/event2:	EV_KEY		在Power放開前按下Volume Up鍵
14	[92145.240271]	/dev/input/event2:	EV_SYN	SYN_REPORT	00000000
15	[92145.290169]	/dev/input/event2:	EV_KEY	KEY_POWER	UP
16	[92145.290174]	/dev/input/event2:	EV_SYN	SYN_REPORT	00000000

圖 5　一段按鍵操作之底層事件紀錄

3.4　按鍵操作關鍵字之轉換

按鍵操作的判斷較觸控手勢容易，只有點擊、長按及組合鍵這三種操作。一個按鍵操作在 Android 底層是由一個值為 1 (DOWN)及另一個值為 0 (UP)的 EV_KEY 事件組成，其中 event code 的欄位會說明使用者按下的是什麼按鍵，按鍵的詳細定義位於 linux/input.h 檔案中。圖 5 為一段按鍵操作的底層事件紀錄。

由圖 5 可以看到我們只要利用第 1 行的 DOWN 及第 3 行的 UP 兩個事件的時間差便可判斷使用者所按下的 KEY_POWER 鍵是短按或是長按，若是如同第 9 行及第 11 行連續收到兩個以上 event code 不同但值同為 DOWN 的事件，則可判定使用者是輸入 KEY_POWER 鍵及 KEY_VOLUMEUP 鍵的組合鍵操作。

四、關鍵字測試腳本播放

本論文依照執行方式不同，將所設計的測試關鍵字分為三類，分別為高階關鍵字、低階關鍵字及控制流程關鍵字。在節中我們針對高低階關鍵字不同的播放方式及跨裝置的播放方法進行說明。

4.1　高階操作關鍵字播放

對應到 UiAutomator API 的使用者操作關鍵字我們分類為高階關鍵字，其播放時我們僅需將 JSON 格式的測試關鍵字反序列化，並將測試關鍵字內的 args 對應到 UiAutomator 相關函數所需的參數，UiAutomator 會透過 ADB (Android Debug

Bridge) [13]向行動裝置下達執行這些高階關鍵字所描述動作的指令，幫助我們完成使用者操作的播放。表 3 為高階關鍵字及其對應之 Python UiAutomator API。

表 3　高階操作關鍵字及其對應之 Python UiAutomator API

關鍵字 action	關鍵字 args	Python UiAutomator API
Click	point	device.click(x, y)
LongClick	point	device.long_click(x, y)
Swipe	points	device.swipe(sx, sy, ex, ey)
Drag	points	device.drag(sx, sy, ex, ey)
Press	key	device.press(key)

4.2　低階操作關鍵字播放

無法使用 UiAutomator 等現有的自動化測試框架重播之使用者操作關鍵字我們將其分類為低階關鍵字，這些低階關鍵字的重播方式我們仰賴 RERAN 工具中的重播部分。

RERAN 重播工具為一個直接在待測裝置上執行的 C 程式，他可以透過指定的檔案格式重製一連串的 input_event 結構並寫入 event file。在測試腳本執行前我們會將 RERAN 重播工具上傳至待測裝置，以便執行測試腳本時能透過 ADB 執行。

圖 6 為一個 SingleFreeSwipe 關鍵字經轉換成為 RERAN 重播工具在重播操作時所使用的檔案，其中第 1 行為總底層事件數目，之後是由一行 sleep 時間（μs）及一行 device file, event type, event code, event value 的 10 進位值描述所組成的集合。

關鍵字

```
{
   "action": "SingleFreeSwipe",
   "args": {
      "points": "[(218, 596), (223, 596),
(226, 596), (230, 595), (235, 594), (242,
593), (251, 592), (261, 591), (275, 589),
(288, 587), (297, 586), (308, 585), (323,
583), (337, 581), (349, 580), (358, 579),
(367, 579), (375, 578), (385, 577), (400,
577), (412, 576), (421, 576), (431, 575),
(444, 574), (452, 574), (458, 573), (465,
572), (474, 572), (482, 571), (514, 574),
(509, 579), (504, 582), (497, 589), (489,
595), (481, 601), (471, 609), (464, 616),
(457, 622), (450, 627), (441, 632), (432,
637), (419, 643), (408, 650), (402, 653),
(399, 655), (387, 662)]"
   }
}
```

RERAN 重播檔案

```
142
0
0,3,57,128
5000
0,3,48,20
5000
0,3,53,348
2000
0,3,54,982
2000
0,0,0,0
10000000
0,3,53,356
        ……
2000
0,0,0,0
10000000
0,3,57,4294967295
5000
0,0,0,0
```

圖 6　關鍵字轉換為 RERAN 重播工具所使用的重播檔案範例

為了降低播放測試腳本時的延遲時間，我們在讀入測試腳本後會先為腳本內所有的低階關鍵字建立 RERAN 重播工具所需的播放檔案並將檔案上傳至待測裝置。之後當我們執行到這些低階關鍵字時僅需透過 ADB 呼叫 RERAN 重播工具即可重播這些使用者所錄下的低階操作。

4.3　關鍵字測試案例播放流程

圖 7 為關鍵字測試案例播放之活動圖。在播放開始時，我們先將 JSON 格式的測試腳本反序列化以建立 keywordList，之後對清單內所有的低階關鍵字預先產生 RERAN 播放工具所需檔案並將檔案路徑寫入關鍵字中的參數，最後我們依序對 keywordList 中每個關鍵字呼叫其 action 欄位所對應的函式並將關鍵字中的 args 欄位做為參數傳入，這些函式就會呼叫 UiAutomator 或是 RERAN 執行這些關鍵字所定義的使用者操作，使測試腳本順利執行。

4.4　跨裝置播放測試腳本

本論文使用的測試腳本是由 Android 底層事件轉換而來，其在播放時不同於使用 Android UI 元件錄製的工具，能得知應用程式中的 Android UI 元件狀態。再者在不同的裝置平台對於底層事件的描述不一定相同，所錄製的低階事件具有硬體的相依性，所以當我們所錄製的測試腳本在不同的裝置平台上播放時將有以下挑戰需克服：

- 平台效能及環境不同導致時間同步問題

我們的工具在錄製腳本時並無當下的 Android UI 元件資訊，所以無法得知使用者是對哪個 UI 元件進行操作，所以我們只能利用一定的等待時間，等待裝置的狀態改變成使用者下一段操作時的狀態，而不同的裝置由於效能不同，所需的反應時間也不盡相同，一個在效能較好的裝置上所錄製的腳本在效能較差的裝置上執行時很可能會遭遇狀態同步的問題。

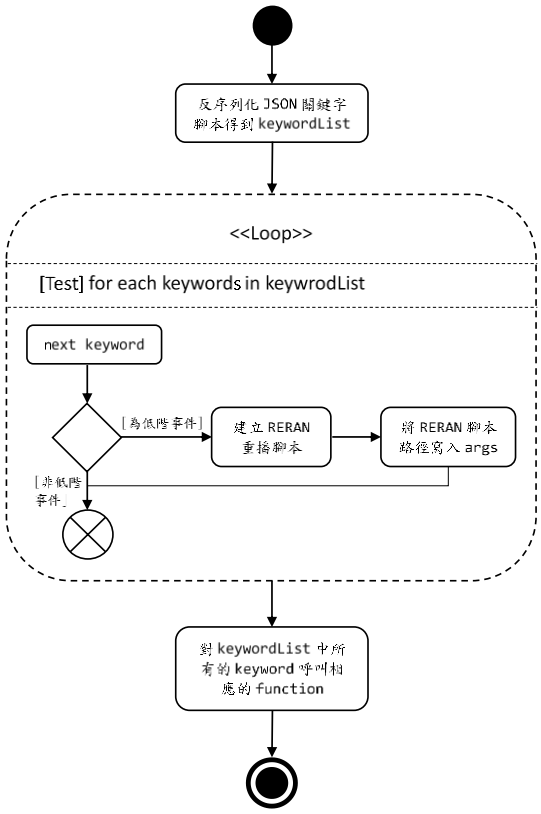


圖 7　關鍵字測試案例播放之活動圖

目前克服此問題需要使用者手動編輯測試腳本，使用者可以更改關鍵字 Sleep 中的 time 參數以調整不同裝置上的等待時間，或是於腳本內插入 Wait 關鍵字等待特定的 Android UI 元件，經過編輯的測試腳本即可順利運行於其他裝置上。

- 不同裝置的 Input device file 編號不同

由於硬體的設計不同，不同的裝置可能有不同的輸入裝置，這些輸入裝置在 Android Input Subsystem 及其對應的/dev/input/eventX 檔案編號也不盡相同，這使得我們在某一行動裝置上所錄製的低階事件在其他裝置上播放時可能無法對應到正確的 Input device file。

克服這個問題的方法則是將 Input device file 編號參數化並寫入裝置設定檔，使用者只需要在播放時於裝置設定檔指定錄製腳本的來源裝置及播放的目標裝置，在測試腳本執行時便可自動的替換測試腳本中的輸入裝置編號。

- 不同裝置的螢幕解析度與觸控範圍座標不同

不同的行動裝置的螢幕尺寸不同，其觸控螢幕驅動程式所描述的觸控區域範圍也不同，這使得使用者在某一裝置上所錄製的觸控座標在另一裝置上播放時無法正確的對應到我們所期望的觸控點。

使用者同樣透過切換裝置設定檔來克服這個問題，我們將裝置的螢幕範圍及觸控區域範圍參數化，並定義數個常數來描述這些範圍的比值。不同裝置的觸控範圍及螢幕大小可以透過 ADB 使用 dumpsys 工具得知，使用者僅需指定錄製腳本的來源裝置及播放的目標裝置，在測試腳本執行時便會自動重新計算腳本內的座標點該如何正確的對應到待測裝置上的座標點。

五、系統開發與工具實作

5.1 系統簡易運作流程

本論文所實作之工具分為錄製及播放兩部分，其簡易運作流程概念如圖 8 所示。在錄製時使用者操作行動裝置並錄下 Android 底層事件紀錄，之後使用轉換工具分析事件紀錄以產生關鍵字測試腳本。播放時使用者利用 Test runner 驅動重播工具讀入關鍵字腳本並對行動裝置進行腳本內的關鍵字所定義的操作。

5.2 系統限制

基於開發版本因素，本論文所實作的系統在錄製及播放上尚有以下限制：

- 播放與錄製的操作時間不完全相同

由於裝置在接收 UiAutomator 的操作指令或是處理 RERAN 重播工具所注入的底層事件時需要通訊及反應的時間，所以在重播測試腳本時所消耗的時間會比錄製腳本時間需要更多，在一些對時間較敏感的應用程式測試上，本論文的工具無法完全的重製使用者的操作。

- 無法在一個關鍵字操作中插入其他操作

由於本論文在轉換關鍵字時會將同類型的使用者操作分段處理，一段 Android 底層事件只會對應到一個關鍵字，所以使用者無法在一個關鍵字操作中插入其他不同種類的操作。例如我們無法在一個滑動手勢操作進行中插入一個實體按鍵的點擊。

- 未使用 Android Input Subsystem 的硬體無法錄製及操作

部分行動裝置的硬體不使用 Android Input Subsystem 與 Android 系統溝通(例如相機、重力感測器…等)，這些硬體事件無法由 getevent 工具錄製也無法利用 RERAN 重播工具重播，目前本論文所提出的工具無法對這些硬體進行錄製及操作。

- 僅限對 UI 元件進行斷言

本論文所提出的工具在斷言時是使用 Android XML layout 中的屬性進行比對，此方法僅能對 UI 元件屬性進行斷言，對於非 UI 元件的圖形狀態或是 Android 系統本身的狀態資訊本論文的工具目前不支援斷言。

5.3 系統操作流程及個案展示

本節我們展示的應用程式為 Writing Order Hiragana [14]，這是一個日文五十音的教學遊戲，以 Unity [15]繪圖引擎寫成，其操作不使用 Android UI 元件。使用者依照遊戲畫面指示於觸控螢幕上

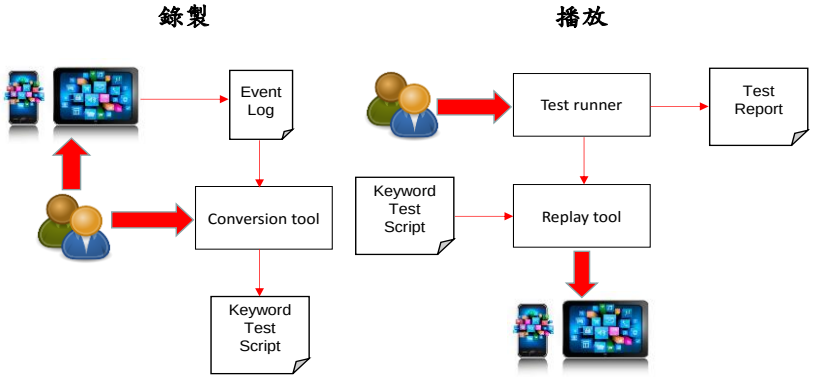


圖 8　系統簡易運作流程概念

輸入日文假名，遊戲會以筆順及筆跡判定使用者輸入是否正確。以下依序說明利用本系統進行錄製操作之各項步驟：

(1) 錄製 Android 底層輸入事件

首先使用者於本機端的終端機使用 adb shell 指令執行 Android SDK 中的 getevent 工具並加入"-t"參數以錄製帶有時間戳記的 Android 底層事件。在此測試情境中我們從點擊 Writing Order Hiragana 主畫面的 Start 開始按鈕，進行了 10 個日文假名的手寫輸入，總費時約 1 分鐘的操作，得到 5731 個 Android 底層輸入事件。

(2) 將 Android 底層輸入事件紀錄檔轉換為 JSON 格式之關鍵字測試腳本

使用者操作錄製完成後，我們使用本系統的關鍵字轉換工具將此 Android 底層輸入事件紀錄檔轉換為 JSON 格式的關鍵字測試腳本。圖 9 為底層輸入事件紀錄檔經本工具轉換後產生的關鍵字測試腳本片段。由此可看到，我們進行手寫輸入操作時所錄製的 Android 底層事件，在經手勢辨認後成為 SingleFreeSwipe 及 Swipe 等觸控手勢。

```
    "action": "SingleFreeSwipe",
    "args": {
        "points": "[(301, 539), (298, 542), (298, 545), (2
    }
},
{
    "action": "Sleep",
    "args": {
        "time": 2.295011
    }
},
{
    "action": "Swipe",
    "args": {
        "points": "[(500, 533), (504, 536), (507, 537), (5
    }
},
{
    "action": "Sleep",
    "args": {
        "time": 3.381024
    }
},
{
    "action": "SingleFreeSwipe",
    "args": {
        "points": "[(158, 678), (162, 677), (167, 676), (1
    }
},
```

圖 9　經轉換後得到的 JSON 格式關鍵字測試腳本（片段）

(3) 編輯 JSON 格式之關鍵字測試腳本

Android 底層事件轉換 JSON 格式測試腳本後我們可對其進行編輯，如更改等待時間、改變原先錄製的座標點、插入使用者操作或測試斷言等。

(4) 指定播放裝置平台

在測試腳本播放之前，使用者必須設定待測裝置的硬體相關資訊，這些參數位於 device_cfg.py 檔案中，藉由編輯此檔案使先前錄製的測試腳本能夠運行於不同的待測裝置。在此案例中我們使用了 ASUS Nexus 7 及 HTC Desire 300 這 2 個不同的行動裝置做為測試腳本播放平台，稍後可以看見我們先前錄製的腳本運行在兩種不同的裝置上。圖 10 為 device_cfg.py 設定檔中的裝置硬體資訊。

```
nexus7_cfg = dict(
    displayW = 800,
    displayH = 1280,
    touchW = 1279,
    touchH = 2111,
    touch_device = 0,
    key_device = 2)

desire300_cfg = dict(
    displayW = 480,
    displayH = 800,
    touchW = 712,
    touchH = 1072,
    touch_device = 5,
    key_device = 1)

source_device = nexus7_cfg
target_device = nexus7_cfg
#target_device = desire300_cfg
```

圖 10　device_cfg.py 中設定的裝置硬體資訊

(5) 使用 pytest 驅動播放工具並得到測試結果

最後使用 pytest 驅動本論文的播放執行工具，播放工具會讀入 JSON 格式測試腳本並依序執行其中各個關鍵字所定義的動作。圖 10 展示先前錄製的測試腳本同時在兩台行動裝置上執行情形。

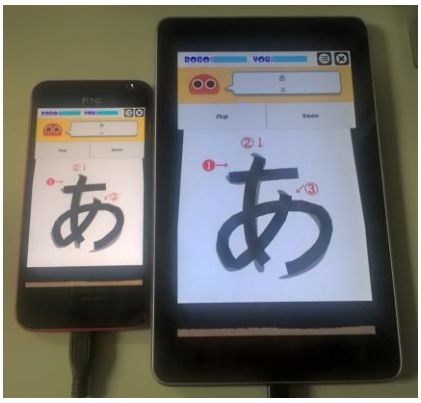


圖 11　測試腳本同時於 2 台行動裝置上執行

六、結論與未來展望

本論文提出一個利用 Android 底層事件紀錄轉換為關鍵字測試腳本的方法並實作相關工具，使用者能以錄製的方式產生 Android 應用程式的自動化測試腳本。透過底層事件錄製使用者操作可以處理現有工具無法處理的非 Android UI 元件及複雜手勢操作，除此之外我們對於不同種類的 Android 底層事件提供不同的事件分析器，這些分析器可以辨認一連串由低階數值所組成的底層事件並使用關鍵字的形式去描述這些動作，使用者可以了解及維護本論文工具產生的測試腳本。

未來本論文持續擴充測試工具，有更好的使用者介面、支援更多使用者操作及關鍵字還有降低腳本對時間的敏感度，使我們的測試工具更可靠且更易用於 Android 自動化測試。

誌謝

本研究由科技部計畫 MOST 104-2221-E-027-009 所補助，特此感謝。

參考文獻

[1] Lorenzo Gomez, Iulian Neamtiu, Tanzirul Azim, and Todd Millstein, "RERAN: Timing- and Touch-Sensitive Record and Replay for Android." In *Proceedings of the International Conference on Software Engineering (ICSE 2013)*, May 2013, pp. 72-81.

[2] UiAutomator, https://developer.android.com/tools/testing-support-library/index.htm, Jun. 1, 2015

[3] 郭柏廷，Android應用程式之GUI自動化測試方法，碩士論文，國立臺北科技大學資訊工程研究所，台北，2013。

[4] Matthew Halpern, Yuhao Zhu, Ramesh Peri and Vijay Janapa Reddi, "Mosaic: Cross-Platform User-Interaction Record and Replay for the Fragmented Android Ecosystem" In *Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, March 2015, pp. 215-224.

[5] Tommi Takala, Mika Katara, and Julian Harty, "Experiences of system-level model-based GUI testing of an Android application." In *Proceedings of the 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, 2011, pp. 377-386

[6] Tuomas Pajunen, Tommi Takala, and Mika Katara, "Model-Based Testing with a General Purpose Keyword-Driven Test Automation Framework." In *Proceedings of the IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2011, pp. 242-251

[7] Zhongqian Wu, Shu Liu, Jinzhe Li, and Zengzeng Liao, "Keyword-Driven Testing Framework for Android Applications." In *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*, 2013, pp. 1096-1102.

[8] Android testing framework, http://developer.android.com/tools/testing/testing_android.html, Jun. 1, 2015

[9] Robotium, https://code.google.com/p/robotium/, Jun. 1, 2015

[10] 陳皇各，Android 關鍵字驅動測試函式庫與工具之設計與開發，碩士論文，國立臺北科技大學資訊工程研究所，台北，2014。

[11] MonkeyTalk, http://www.cloudmonkeymobile.com/monkeytalk, Jun. 1, 2015

[12] Robotium Recorder, http://robotium.com/products/robotium-recorder, Jun. 1, 2015

[13] Android Debug Bridge, http://developer.android.com/tools/help/adb.html, Jun. 1, 2015

[14] Writing Order Hiragana, https://play.google.com/store/apps/details?id=com.robotani.kakijyunlite, Jun. 1, 2015

[15] Unity, https://unity3d.com, Jun. 1, 2015