延伸雲端測試平台以支援 Android 與 Firefox OS 應用程式之測試
Extending Cloud Testing Platform for Supporting Android and Firefox OS Application Testing

劉建宏[1]、林敬文[2]、陳宥名[3]、蔡泳誠[4]
國立臺北科技大學資訊工程系[1,4]
財團法人資訊工業策進會[2,3]
Chien-Hung Liu[1], Ching-Wen Lin[2], Yu-Ming Chen[3] and Yung-Cheng Tsia[4]
Department of Computer Science and Information Engineering,
National Taipei University of Technology[1,4]
Institute for Information Industry[2,3]
Email:{cliu[1], t103598015[4]}@ntut.edu.tw, {tedlin[2], randolphchen[3] }@iii.org.tw

摘　要

雲端測試平台(Cloud Testing Platform，簡稱CTP)是一個支援Android App相容性測試的平台，開發(測試)人員只要將 App 與測試腳本上傳至CTP，即可自動進行 App 與各種裝置之相容性測試，節省測試成本。本論文擴充 CTP，以支援Firefox OS 裝置與其 App 之測試，使得 Firefox OS App 亦能在 CTP 平台上進行相容性測試。由於以ADB 命令操作 Firefox OS 裝置時，必須將該裝置連接至 USB 埠，而 CTP 是以 Wireless 的方式執行ADB 命令，因此，本論文提出重構 CTP 的方法，使其兼容 USB 以及 Wireless 的連接方式，並確保不同的連接方式不影響測試結果。重構時亦考量未來的可擴充性，使 CTP 能夠輕鬆加掛新的測試工具或支援新的其他類似裝置。實驗結果顯示 CTP能同時支援 Android 與 Firefox OS 的應用程式測試，且使用 USB 或 Wireless 連接裝置對於測試結果並沒有影響。

關鍵詞：Android、Firefox、CTP、重構

一、前言

隨著 Firefox OS[1]發展逐漸成熟，越來越多的Firefox OS 裝置與其 App 出現，因此 App 的測試更顯得日益重要，但 Firefox OS App 開發(測試)人員在驗證 App 在各款裝置上的執行狀況時，並沒有合適的測試工具或平台能幫助他們完成測試工作，導致 App 開發(測試)人員需要花費大量的時間在進行測試，造成測試成本增加。

CTP 將 Android[2]的測試工具轉換成雲端服務，利用雲端運算將測試工作分散到多個虛擬機器上平行化執行，模擬開發(測試)人員對應用程式進行測試，App 開發(測試)人員能使用 CTP 平台減少花在相容性測試的時間。利用雲端測試平台平行化執行測試工作的能力，能減少進行 Firefox OS App相容性測試的測試成本，然而雲端測試平台僅支援Android 應用程式之測試，因此本論文擴充 CTP，以支援 Firefox OS 裝置與其 App 之測試，使得Firefox OS App 亦能在 CTP 平台上進行相容性測試。

由於以 ADB[3]命令操作 Firefox OS 裝置時，必須將該裝置連接至 USB 埠，而 CTP 是以 Wireless 的方式執行 ADB 命令，因此要延伸 CTP 以支援Firefox OS App 測試，必須重構系統使其兼容 USB以及 Wireless 的連接方式，這代表 Android 裝置能自由切換 Wireless 與 USB 的連接方式，Firefox OS裝置未來若支援 Wireless 的連接方式時，亦能自由切換連接方式。

雖然目前能支援 Firefox OS App 測試的工具數量不多，僅有官方提供的 Marionette[4]及資策會自行開發的 OrangFuzz[5]兩套，但未來可能開發出其他測試工具，因此 CTP 要能擁有加掛工具的能力，使得 Firefox OS App 的開發(測試)人員能透過CTP 對應用程式作更完整的測試。隨著智慧型裝置的發展，未來可能推出其他與類似 Android 的裝置，只要能以 ADB 命令來操作其裝置，CTP 就能利用相同的重構方法，使得 CTP 支援其測試，基於以上兩點，系統對於新的測試工具與新的其他類似裝置的可擴充性亦為本次重構的重點。

為了驗證重構後的 CTP 能夠同時支援Android 與 Firefox OS 的測試以及使用 Wireless 或USB 來連接裝置對於測試結果沒有影響，我們設計了兩個實驗，實驗結果顯示 Android 與 Firefox OS 應用程式測試能夠同時執行且不互相影響，使用 Wireless 或 USB 來連接裝置進行測試時，除了測試花費的時間有些許差異外，其餘測試結果皆為相同。

本文後續章節組織如下：第二節介紹本論文的相關研究；第三節敘述本論文重構 CTP 的方法；第四節為實驗與結果；第五節為結論及研究展望。

二、相關研究

整合式雲端測試平台[6][7][8]是由臺北科技大

學資訊工程系所開發的系統，支援 Android 與 Web 的相關測試，其中 Android 的部分支援以下測試工具：(1) Monkey[9]是一個 command-line 工具，它可以在 Android 模擬器或是實體機器上使用，藉由產生隨機事件對 Android 應用程式進行壓力測試；(2) MonkeyTalk[10]是一套行動裝置的測試工具，支援 Android 以及 iOS 應用程式的測試，它可進行簡單的冒煙測試(smoke test)到複雜的資料驅動測試(data-driven test suites)；(3) Robot Framework[11]是使用 Python 撰寫的自動化測試框架，主要用於驗收測試(acceptance testing)和驗收測試驅動開發(acceptance test-driven development, ATDD)，支援 KDT (keyword driven testing)方法；(4) Robotium[12]是個開放原始碼的 Android 自動化測試框架，支援 Native[13]與 Web[14]類型的 Android App；(5) Espresso[15] 是 Google 開發的 Android 自動化測試框架，提供了一系列的 API 可以讓測試人員建構 Android 應用程式的介面測試流程；(6) UiAutomator[16]是由 Android 提供的使用者介面測試的測試框架，可以對裝置當前畫面上的元件進行操作。而 Web 則支援 Functional Testing 及 Performance Testing。

WiFi ADB[17]是能在 Google Play 上免費下載的應用程式，提供一個簡單的懶人介面使得應用程式開發人員能以 Wireless 連接 Android 智慧型裝置來進行除錯，通常要使用 Wireless 連接裝置時要先以 USB 埠連接裝置，透過 ADB 指令開啟裝置上特定的埠後，App 開發人員方能透過 Wireless 連接裝置來下達 ADB 命令，一旦裝置重新啟動就要再次進行上述步驟，對於開發人員而言相當不便，使用 WiFi ADB 能輕鬆的開啟/關閉 Wireless 連接裝置的功能。我們利用 WiFi ADB 協助 CTP 透過 Wireless 連接裝置來下達 ADB 命令。

Testdroid[18]是由 Bitbar 開發團隊所開發的雲端平台，提供 Android 與 iOS[19] APP 自動與手動測試，在測試完成後能得到測試結果及裝置硬體資源的使用狀況，其中 Android 的部分支援以下測試工具：(1)Appium[20]是一個開放原始碼的 Android 與 iOS App 自動化測試框架，，對 Native、Web 及 Hybrid[21]等類型的 App 有全面性的支援；(2)Calabash[22]是一個自動化測試框架，支援 Android 與 iOS App 測試，主要用於驗收測試；(3)Espresso；(4)Robotium；(5)UiAutomator。使用此平台能減少應用程式相容性測試所需的測試成本，但此平台不支援 Firefox OS 應用程式的測試，無法驗證 Firefox OS 應用程式執行於各款裝置的狀況。

三、 系統設計與實作

本論文以 CTP 為基礎，擴充並重構其系統架構與設計，以支援 Firefox OS 應用程式測試。CTP 在進行測試工作時，必須以 Wireless 連接至該裝置，才能對裝置下達 ADB 命令，但 Firefox OS 裝置不支援以 Wireless 連接裝置的方式，因此，我們必須以 USB 埠連接 Firefox OS 裝置，才能進行 Firefox OS 應用程式的測試，為了解決這個問題，我們利用周世邦提出的 USB Master/Slave Server[25]，使得 CTP 能夠對連接於 USB Master/Slave Server 上的裝置進行測試。

CTP 平台的部分類別圖，如下圖 1 所示，其中繼承 AndroidTestingBuilder 的類別除了圖中所示之外，尚有 MonkyTalk、Robotium、UiAutomator 以及 Espresso，為避免類別圖過於雜亂予以省略，為了使系統能夠兼容兩種不同的連接方式以及增加系統的可擴充性，我們必須對系統中的 Executor 以及 Builder 進行重構，Executor 為系統下達測試指令的類別，因此，要使 CTP 兼容不同的連接方式與重構此類別有關，Builder 為 CTP 控制測試工作如何進行的類別，根據不同測試工具與裝置連接方式而有不同的型態，因此，測試工具與平台的可擴充性與重構此類別有關，將於下面兩個小節分別介紹這兩個類別如何重構。

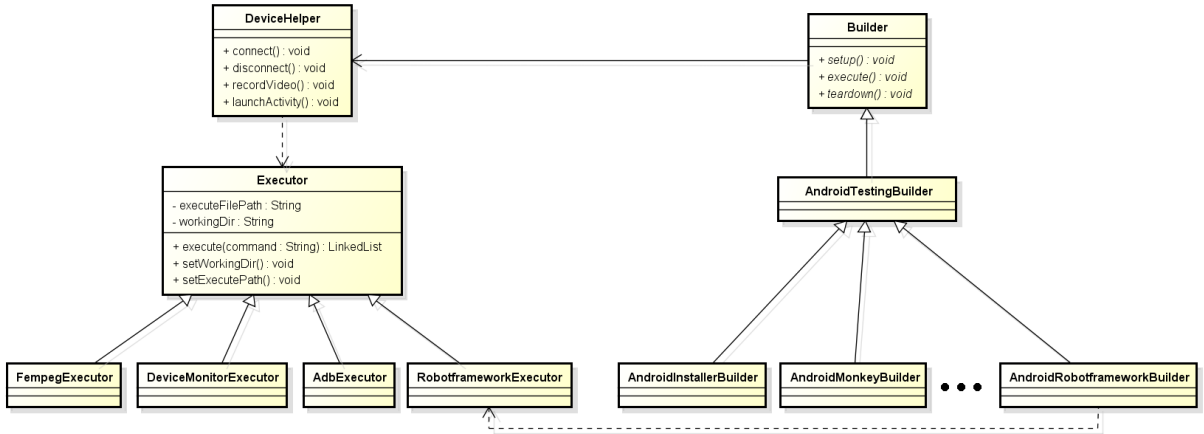

圖 1 CTP 類別圖

3.1 重構 Executor

為了讓 CTP 能夠兼容 Wireless 與 USB 的連接方式，我們必須重構 Executor 這個類別，Executor 是用來執行測試指令的類別，負責組成完整的可執行測試指令、設置測試指令的環境以及測試指令工作目錄。重構前的類別除了不支援 USB 的連接方式外還有以下問題，第一 DeviceHelper 的封裝不完整，Builder 仍要去使用 RobotframeworkExecutor 這個類別，失去了 DeviceHelper 消除介於 Builder 與 Executor 之間相依性的功能；第二 RobotframeworkExecutor 功能並無擴充 Executor 的功能，僅在建構子中設定測試指令的環境；第三 DeviceMonitor 為 CTP 安裝於裝置上的應用程式，負責將裝置由睡眠中喚醒以及紀錄待測 App 的對裝置硬體資源的消耗狀況，DeviceMonitorExecutor 利用 Socket 連線至待測裝置的 DeviceMonitor，傳送特定訊息給 DeviceMonitor 使其運行特定功能，在以 Wireless 連接裝置時我們會配置一個固定 IP 給代測裝置，DeviceMonitorExecutor 能夠藉此 IP 與 DeviceMonitor 進行 Socket 連線，但是使用 USB 連接裝置時，無法傳送訊息給 DeviceMonitor 以運行特定功能，因此要尋求其他解決方式。

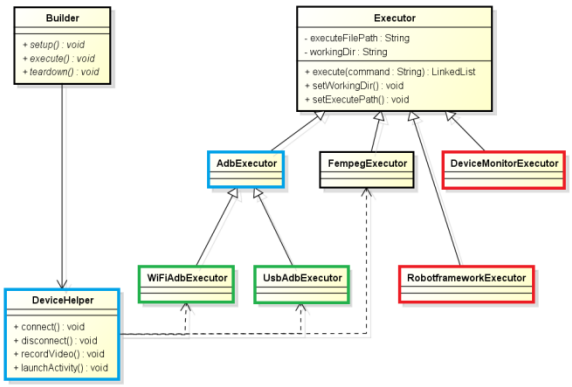

圖 2 重構後的 Executor 類別圖

重構後的類別圖，如圖 2 所示，其中以紅框標示的為刪除的類別，藍框標示的為修改的類別，綠框標示的為新增的類別，AdbExecutor 作為抽象類別將常用的 ADB 命令定義成抽象方法，由 WiFiAdbExecutor 及 UsbAdbExcutor 繼承它，根據不同的連接方式去實作個別的方法，以呼叫 DeviceHelper 中的 connect 方法為例，當我們使用 Wireless 的連接方式時，循序圖如下圖 3 所示，DeviceHelper 會產生 WiFiAdbExecutor 的實體並呼叫它的 connect 方法，這個方法能夠直接對裝置下達 adb connect 的命令以跟裝置取得連線；當我們使用 USB 的連接方式時，循序圖如下圖 4 所示，DeviceHelper 則是產生 UsbAdbExecutor 的實體，同樣呼叫它的 connect 方法，這個方法會透過 RMI 來呼叫 USB Master Server 的 doCommand 方法，USB Master Server 會將測試指令委派給適當的 USB Slave Server，USB Slave Server 則會回覆 DeviceHelper 要求的裝置是否連接於機器上，若有則 connect 成功，反之則失敗。其他的 adb 指令下達方式皆採用同樣的模式，Wireless 為直接對裝置進行操作，USB 則透過 USB Master/Slave Server 來完成。

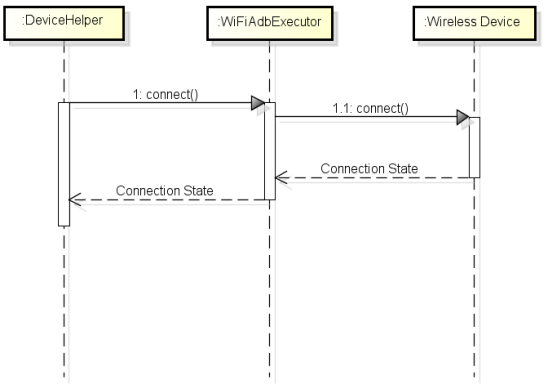

圖 3 操作 Wireless 裝置的循序圖

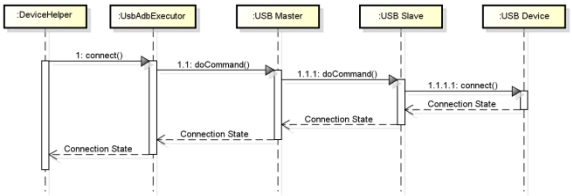

圖 4 操作 USB 裝置的循序圖

RobotframeworkExecutor 為 Executor 的子類別，從繼承的角度來看，它應該要擴充 Executor 的功能，但這個類別與 Executor 的差異只在於建構子設定測試工具的執行檔路徑不同，因此我們認為 RobotframeworkExecutor 不適合作為類別予以刪除，將設定路徑的工作交由 DeviceHelper 來負責，Builder 可以藉由呼叫 DeviceHelper 中 launchRobotframework 這個方法來完成相同的工作，在這個方法中，DeviceHelper 會產生一個 Executor 的實體，並且指定執行檔路徑，如此我們可以產生一個功能與 RobotframeworkExecutor 功能相同的物件。

DeviceMonitorExecutor 需要透過 Socket 連線來啟動 DevieceMonitor 的服務，但是以 USB 連接的裝置則無法使用此服務，因此我們改寫 DeivceMonitor 這個應用程式，使得 CTP 能透過下達 adb Broadcast 這個命令，啟動 DeviceMonitor 喚醒裝置及記錄待測應用程式對裝置硬體資源消耗狀況的服務，由於改為使用下達 adb 命令的方式，DeviceMonitorExecutor 就沒有存在的必要，其功能可以用 AdbExecutor 來取代。

3.2 重構 Builder

Builder 在系統中負責控制測試工作的進行，測試工作共分三個階段，setup 為設定測試環境的階段，例如安裝待測應用程式、開啟錄影功能及開始監控裝置硬體資源消耗情況等；execute 為執行測試，這個階段根據測試工具的不同而有不同的操作模式；teardown 為收集測試結果以及回復裝置環境的階段，例如移除待測程式、關閉錄影功能以及收集測試過程裝置硬體資源的消耗情況等。

在完成 Executor 的重構後，Builder 能夠根據測試工作的需求來決定要使用 USB 還是 Wireless 的連接方式，但是使用不同連接方式時會有些差異，舉例來說，在 teardown 階段要取得測試過程的影片時，需要下達 adb pull 這個命令來取出存放於裝置中的原始影片，當使用 Wireless 連接裝置時不需做額外的工作，然而使用 USB 連接裝置時，因為 adb pull 命令完成之後影片只會存放在 USB Slave 上，因此需額外進行將影片由 USB Slave 傳回到 CTP 的工作。類似的情況存在於各個階段中，為了解決這個問題，我們重構 Builder 使其能夠根據需求來切換 USB 與 Wireless 的操作模式。

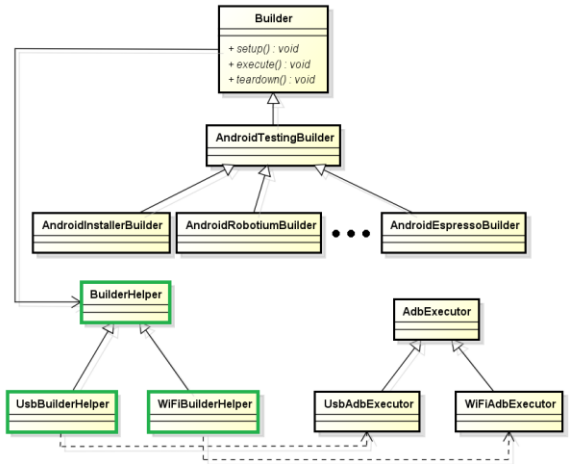

圖 5 重構 Builder 後的類別圖

重構後的類別圖，如圖 5 所示，我們將 DeviceHelper 改為 BuilderHelper 這個抽象類別，分別實作了 USB 以及 Wireless 的 BuilderHelper，這個部分用了 Adapter 樣式[31]的概念，Builder 可以產生不同的 BuilderHelper 物件來達到轉介不同連接方式的功能，因此能夠減少系統中不斷重複去決定要採用何種連接方式的判斷式，程式碼的可讀性也因此提高。根據連接裝置方式的不同需要使用對應的 Executor，UsbBuilderHelper 需使用 UsbAdbExecutor 來下達測試指令，WiFiBuilderHelper 則需使用 WiFiAdbExecutor，這個部分我們套用工廠樣式[25]，讓實作 BuilderHelper 的類別去覆寫 getExecutor 這個工廠方法來產生對應的 AdbExecutor，這樣可以避免產生錯誤的 AdbExecutor 導致程式發生錯誤，也可以省去多餘的判斷式。

由於系統重構後加掛 Firefox OS 的測試變得容易許多，加掛後的類別圖，如圖 6 所示，新增 FirefoxTestingBuilder 去繼承 Builder，覆寫 setup、execute 以及 teardown 這三個方法，分別定義各階段要完成的基本工作，例如安裝待測程式或是解除安裝待測程式等等，新增 FirefoxOrangFuzzBuilder 這個類別來定義本論文加掛的測試工具如何使用，OrangFuzz 是一款類似 Monkey 的測試工具，能夠送出隨機事件來達到測試的目的，這裡套用了 Template Method 樣式[25]，透過實作 doSetup、doExecute、doTeardown 這三個 Template Method 來決定不同的測試工具要如何進行對應的測試工作，未來如果需要擴充支援的測試工具只要新增一個類別並實作這三個方法即可。

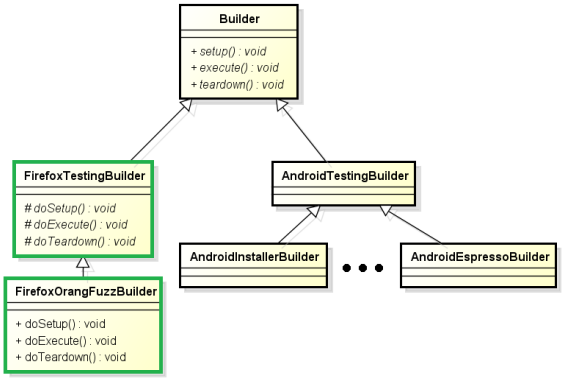

圖 6 加掛 Firefox OS 測試的類別圖

四、 實驗

本實驗目的在於回答以下兩個研究問題(Research Question)，RQ1：本平台是否可同時支援多部 Android 與 Firefox OS 裝置，RQ2：對於 CTP 支援的所有測試工具，使用 Wireless 或 USB 連接裝置是否對測試結果有所影響。

4.1 實驗設計

為了回答兩個研究問題，我們設計以下兩個實驗，實驗 1 我們假設本平台可同時支援多部 Android 與 Firefox OS 裝置進行測試，實驗方法為透過 CTP 同時對 5 部 Android 及 2 部 Firefox OS 裝置進行測試，統計重複執行 10 次之實驗數據後，確認有無錯誤發生，實驗 2 我們假設使用 USB 或 Wireless 連接裝置對於測試結果沒有影響，實驗方法為透過 CTP 對 5 部以 Wireless 連接 Android 裝置進行測試並記錄測試結果(包含測試報表、logcat、效能報表與測試錄影)，接著將待測裝置改用 USB 連接後進行相同的測試工作並記錄測試結果，再統計重複 10 次之實驗數據後，比較兩者之差異。

本實驗所使用的 CTP Server 硬體規格，如表

1 所示。虛擬機器透過 OpenStack 進行管理，其硬體規格如表 2 所示。待測裝置共有 Firefox 裝置 2 部及 Android 裝置 5 部，其規格如表 3 所示。實驗 1 所執行的測試工作中，Android 測試使用的待測程式為為 ezWeight，ezWeight 是軟體測試工廠自行開發的應用程式能夠計算 BMI 以及紀錄使用者每日攝取卡路里量，測試工具為 Monkey，Firefox OS 測試使用的待測程式為 Calendar，測試工具為 OrangFuzz，實驗一中設定的 Input Event 數量為 50 個。實驗二所執行的測試工作中使用的待測程式為 ezWeight，Monkey 測試設定的 InputEvent 數量為 50 個，MonkeyTalk、Robotium、Robotframework、UiAutomator、Espresso 等測試皆使用相同的測試腳本，測試腳本詳細內容如表 4 所示。

表 1 CTP Server 硬體規格

CPU	Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz
RAM	16 GB
OS	Windows 7 Enterprise 64bit
DISK	SSD 100GB

表 2 OpenStack 硬體規格

CPU	Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz x2
RAM	66 GB
OS	Ubuntu 14.04 LTS
DISK	200GB

表 3 待測裝置規格

Phone Type	RAM	OS	Internal Storage
Infocus Saga G1	2GB	B2G 1.4.0.0	16GB
Infocus sun6i	2GB	B2G 2.2.0.0	16GB
Samsung S5	2GB	Android 4.4	16GB
HTC Buttrtfly2	2GB	Android 4.4	16GB
Sony Xperia Z3	3GB	Android 4.4	16GB
ASUS Padfone S	2GB	Android 4.4	16GB
LG GPro2	3GB	Android 4.4	16GB

表 4 實驗 2 測試腳本詳細內容

Test case：計算 BMI	
Description：輸入身高及體重後按下計算按鈕	
Data Requirement： ${身高}：170 ${體重}：65 ${BMI}：22.4	
Step Number	Step Description
1	輸入${身高}
2	輸入${體重}
3	點擊”開始計算”按鈕
4	驗證計算結果與${BMI}相符

4.2 實驗結果

實驗一同時對 5 部 Android 裝置與 2 部 Firefox OS 裝置進行 10 測試，總計 70 次測試中皆無發生錯誤，根據實驗結果，RQ1 的答案為，CTP 平台確實能同時支援多部 Android 與 Firefox OS 裝置。

實驗二結果如下表 5 所示，其中若 USB 與 Wireless 執行結果相同的項目加上✓的符號，接著說明各項目如何算是結果相同，首先是測試報表，測試報表會依據測試工具的不同而有所異同，因此測試報表判斷的方式為測試是否通過，installer 為應用程式是否安裝成功，成功則為通過，反之則否，Monkey 為利用相同的亂數碼對應用程式發送隨機事件，如果應用程式沒有崩潰(crash)則為通過，反之則否，其餘的測試工具需要會測試腳本執行測試工作，這些測試工具我們判斷其測試案例是否通過作為評斷標準，實驗過程中以 Wireless 連接 5 部待測裝置並使用 7 種不同的測試工具皆進行 10 次測試，共計 350 次測試結果皆為通過，接著將 Wireless 連接的裝置改為以 USB 連接，同樣進行 350 次測試，測試結果亦皆為通過，因此，我們認為不同的連接方式對於測試報表沒有影響。

表 5 實驗 2 測試結果

裝置	測試報表	效能報表	測試錄影	logcat
Samsung S5	✓	✓	✓	✓
HTC Buttrtfly2	✓	✓	✓	✓
Sony Xperia Z3	✓	✓	✓	✓
ASUS PadfoneS	✓	✓	✓	✓
LG GPro2	✓	✓	✓	✓

效能報表的結果會根據測試工作進行的狀況有些許的差異，即使是相同的測試工作也僅能得到相似的結果，難以完全相同，因此判斷方式為不同的連接方式能夠得到相似的效能報表，兩種連接方式各 350 次測試中得到效能報表皆相似，因此我們認為連接方式對於效能報表沒有影響，測試錄影與 logcat 也是同樣的道理，在兩種連接方式各 350 次測試中同樣得到相似的結果。綜合以上實驗結果，RQ2 的回答為，使用 USB 或 Wireless 的連接裝置對測試結果沒有影響。

五、 結論與未來研究方向

本論文延伸 CTP 使其相容 Android 與 Firefox OS 的應用程式測試。我們重構 Executor 與 Builder 等類別，套用了 Adapter、Factory 以及 Template Method 等設計樣式，提高系統的彈性、可讀性以及可擴充性，使得 CTP 能對 USB Master/Slave Server [x]上連接的裝置進行測試工作。

實驗結果顯示當測試工作越多時，使用本平台所得到的效益就更多，並且使用 USB 或是 Wireless 連接裝置對於測試結果並沒有影響。未來我們考慮在 CTP 增加以下功能：(1)加掛其他測試工具，目前平台僅提供 OrangFuzz [5]一項 Firefox OS 的測試工具，未來希望能夠加掛官方的 Marionette 測試工具[4]或其他開放源碼的測

試工具，使得 Firefox OS 的功能性測試能夠更加完整。(2)自動適時重置待測裝置，當不斷反覆進行各種測試時，待測裝置不免會因為 App 或作業系統的問題，而進入不穩定的狀態，需要有一套機制讓裝置能夠在適當的時機進行 reboot，以維持穩定。

致謝

本研究依經濟部補助財團法人資訊工業策進會「104 年度智慧手持裝置核心技術攻堅計畫(3/4)」辦理。

This study is conducted under the “The core technologies of smart handheld devices project” of the Institute for Information Industry which is subsidized by the Ministry of Economic Affairs of the Republic of China.

參考文獻

[1] Firefox OS, Available, https://developer.mozilla.org/zh-TW/Firefox_OS
[2] Android, Available, http://www.android.com/
[3] ADB(Android Debug Bridge) , Available, https://developer.android.com/studio/command-line/adb.html
[4] Marionette, Available, https://developer.mozilla.org/en-US/docs/Mozilla/QA/Marionette
[5] OrangFuzz, Available, https://github.com/MozillaSecurity/orangfuzz
[6] 李友文，雲端測試服務平台 Android 測試過程錄影服務設計與實作，碩士論文，國立臺北科技大學資訊工程所，臺北，2014。
[7] 呂建群，雲端測試平台 Android 資源耗用之報表服務設計與實作，碩士論文，國立臺北科技大學資訊工程所，臺北，2015。
[8] 楊杰浩，STF-CTP 的資源利用率改善之探討，碩士論文，國立臺北科技大學資訊工程所，臺北，2015。
[9] Monkey, Available, http://developer.android.com/tools/help/monkey.html
[10] MonkeyTalk, Available, https://www.cloudmonkeymobile.com/monkeytalk
[11] Robot Framework, Available, http://robotframework.org/
[12] Robotium, Available, https://code.google.com/p/robotium/
[13] Native (computing), Available, https://en.wikipedia.org/wiki/Native_(computing)
[14] Web Apps, Available, http://developer.android.com/guide/webapps/index.html
[15] UiAutomator, Available, http://www.android.com/
[16] Espresso, Available, https://developer.android.com/training/testing/ui-testing/espresso-testing.html
[17] WiFi ADB, Available, https://play.google.com/store/apps/details?id=com.ttxapps.wifiadb&feature=search_result
[18] Testdroid, Available, http://calaba.sh/
[19] iOS, Available, http://www.apple.com/
[20] Appium, Available, http://appium.io/
[21] Hybrid app, Available, https://de.wikipedia.org/wiki/Hybrid-App
[22] Calabash, Available, http://calaba.sh/
[23] Scenario-based test, Available, https://en.wikipedia.org/wiki/Scenario_testing
[24] 周世邦，解決 ADB 連接上限之 USB Master/Slave 架構，碩士論文，國立臺北科技大學資訊工程所，臺北，2016。
[25] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.