運用 Google Analytics 報表功能強化軟體自動化測試之研究
A Study of Reporting Enhancement on Automated Software Testing Using Google Analytics

蔡皓羽
國立台北科技大學
Hau-Yu Tsai
feather_1201@hotmail.com

陳英一
國立台北科技大學
Ing-Yi Chen
ichen@mail.ntut.edu.tw

摘　要

隨著行動應用的普及，行動裝置的種類及版本不一而足，如何減少測試成本並同時維持軟體品質是值得探討的議題。軟體自動化測試技術即藉由撰寫測試腳本模擬使用者操作行為，以電腦自動執行程式，達到減少人力成本及使資源有效利用之目的。而自動化測試逐步執行腳本時，會顯示出每個步驟是否執行成功，其測試維度僅為 Pass 或 Fail。本文將探討如何運用 Google Analytics，獲取更多測試維度。

以 Google Analytics 應用於軟體自動化測試，透過其行為事件的設計及載體資訊的追蹤，可取得結構化資料，減少了資料庫設計及資料彙整的開發成本，達到以低成本取得自動化測試的操作行為數據，達到強化軟體自動化測試之目的。

本文從需求分析、系統設計至軟體測試等架構規範皆遵循軟體工程，並著重於軟體測試之研究。最後以電影院購票 APP 實例進行軟體自動化測試，並產出測試報表，以達到強化軟體自動化測試結果之目的。

關鍵字：Google Analytics、軟體自動化測試、報表、驗收測試、軟體測試

一、前言

軟體工程的目的是希望透過工程模組化的方式，有效率的開發出功能與品質兼備的軟體，在軟體的開發流程，一直是軟體工程裡重要的一環，它定義了軟體開發時所經歷的階段，以傳統的「瀑布式開發模式（Waterfall）」為例，其將開發流程分為「分析」、「設計」、「開發」、「整合與測試」，即需求分析完成後進行系統設計，接著開發各個子系統，最後進行整合並測試，以線性的方式前進，如：圖 1[1]。

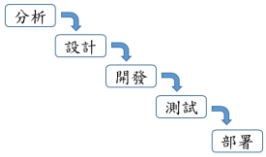

圖 1：Waterfall 示意圖

隨著開發階段越接近部署，其開發所變更的成本會越高，尤在上線階段產生需求變更，變更的成本更會驟升，如圖 2。故在上線前必須謹慎做好測試，以便降低程式變更的風險。

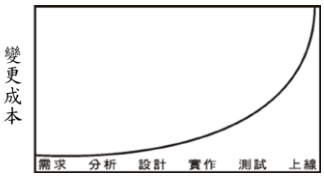

圖 2：需求變更成本示意圖

軟體測試分成了許多階段，以 V 模型為例，如圖 3。

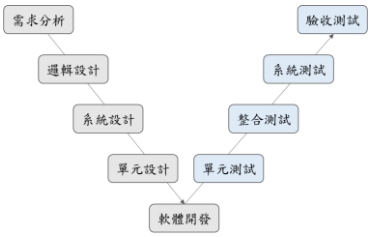

圖 3：V – Model 示意圖

V-Model 是一種瀑布模型改良的開發流程，左半部說明了軟體開發流程需經歷需求分析、系統設計然後進入軟體開發，右半部說明當軟體開發後，緊接著進行一系列的測試設計，從單元測試開始到整合測試，進而到整個系統的測試，最後以使用者操作情境進行驗收測試。V-Model 清楚描述了軟體開發的基本過程及測試行為，透過 V-Model 也知悉測試流程的各個階段。

在測試的最後階段為驗收測試(Acceptance Test)[2]，即由使用者或開發人員，以某一個情境(user story)操作系統，檢驗系統在特定操作下，應該完成什麼樣的功能，以及針對某些輸入，得到怎樣的輸出，其預期輸出與實際輸出是否相符，並紀錄各種測試情境的結果，藉此檢驗功能是否正確。

驗收測試除了檢驗功能是否正常外，在行動應用的普及下，行動裝置的種類也與日俱增，如何確保軟體程式支援各種裝置？一般的方式為在驗收測試時，由人員親自操作，並記錄其測試環境及其結果數據，產出一份紀錄報表，如：圖 4。

專案名稱	電影APP
版本序號	V1.2
測試裝置	HTC One X
作業系統	Android 4.2
正常/錯誤	正常
備註	Non
負責人	Henry

圖 4：傳統測試報告

當欲測試裝置很多時，測試人員須針對相同的測試情境，每個裝置分別執行操作情境流程，藉此檢驗軟體是否相容於裝置的版本；當問題（Bug）修復或是一個功能變更後，則必須再重頭逐一檢驗，確保各種裝置皆能正常運作。故在軟體開發過程中，「測試」需花費不少的時間成本。

隨著專案越來越大，如果要確保系統功能皆正常運作，無可避免地在修改程式之後，欲測試的項目也會越來越多，不斷地重複執行撰寫新功能、測試新功能、測試舊功能...。為了減少以相同測試情境來測試不同裝置及測試舊功能時所花費之重複動作與時間，其解決方案為使用「軟體自動化測試」[3]，藉由撰寫測試腳本，使機器自動幫我們測試，以達到減少重複測試的時間成本。而自動化測試又有哪些不足之處？本文將探討其缺點並以 Google Analytics 報表功能，強化軟體自動化測試結果，達到降低開發成本之目的。

二、研究背景及目的

隨著行動裝置的普及，裝置、版本不一而足，為了確保軟體能夠相容於不同裝置，並減少相同測試情境的重工，自動化軟體測試的技術逐漸發展，即藉由撰寫測試腳本，使電腦自動執行 Test Case 中的步驟，模擬出 User 的操作行為，其優點為（1）同個測試情境(Test Case)可同時應用於不同裝置，減少時間成本（2）若問題修正，則同個腳本可重複利用，減少測試重工。

而軟體自動化測試有哪些可精進之處？其缺乏測試維度及完整報表呈現。常見的做法為於 APP 程式中，撰寫維度資料，並儲存於 Database 中；或是於測試腳本中撰寫程式碼，產生原始碼(.html)、網頁快照(.png)；亦或在測腳本中使用 Plugin 工具，如 Python – HTMLTestRunner、Mocha –Mochawesome。透過第三方工具所得之測試報表，其對於非測試人員來說可視度較低，客製化的彈性度相對較低，難易度也相對較高，如圖 30。

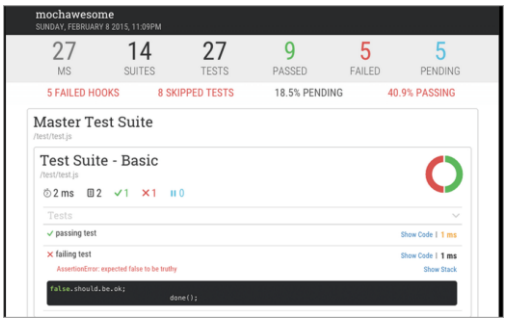

圖 30：Mochawesome 測試圖

若沒有多加處理，軟體自動化測試只是執行測試腳本所設定的行為，讓使用者檢視最後輸出是否為預期結果，其測試維度僅為 Pass / Fail，如圖 5。其他資訊則需手動記錄於圖 4 報告。

圖 5：自動化測試結果

在各式各樣的裝置中，驗收測試須檢驗軟體是否支援裝置，故「裝置版本」、「作業系統」會是所需要紀錄的資訊；而針對不同的裝置版本，若在相同的條件環境下，能瞭解其所執行的「反應時間」，可以知悉在甚麼裝置環境下軟體程式的反應速度，並針對不同的系統進行優化處裡，以達到主流版本裝置皆順暢執行。

而軟體自動化測試腳本[8]其驅動程式碼若是以陣列方式隨機取得操作的行為數據，如陣列[1]、陣列[2]代表不同行為，若能記錄所模擬的行為，對於日後產品的維運工作有很大的幫助。如何擴充測試維度並呈現？開發 APP 的做法為以程式碼取得所需之資訊，以 JavaScript 開發 APP 為例，最簡單方式以程式碼 Log 方式獲取其所需資訊，如欲得到裝置版本，則呼叫 navigator.userAgent，如 console.log(navigator)；欲記錄此函式名稱等備註，則 Log 事件名稱，如 console.log(“事件名稱”)。如此所需的測試維度愈多，所撰寫的程式碼越多，其額外的開發成本也愈高。若要將每筆測試記錄儲存並彙整成測試報告，則必須仰賴資料庫，設計資料庫所需欄位及做資料的結構化，並將每筆記錄儲存下來，再額外設計報表的呈現，如此，不僅開發的時間成本提高，所需的空間成本也提高。因此，本篇論文將繼續探討，如何以低成本擴充測試維度並做呈現。

三、解決方案

以開發 APP 軟體程式為例，裝置的版本對於開發者來說，是有意義的資訊，除了可以透過程式碼取得外，可以利用流量分析工具 – Google

Analytics[4]來獲得其維度資訊。

Google Analytics 為一流量分析工具，其透過追蹤程式碼的設定，來記錄訪客的各種行為及使用環境，並做資料的結構化，產出各式各樣的報表，供開發人員檢驗是否達到其目標效益。而 Google Analytics 不僅提供網站的追蹤，亦支援 Mobile APP 的追蹤，提供 iOS、Android、JS 各種平台之 SDK，其也提供 API 服務，方便我們擴充加值功能，產出適合自己的測試報表。

使用 Google Analytics，可以輕易地收集結構化資料，省去資料庫設計的時間，並獲得結構化資料的呈現，再利用其提供之 API，製作出自己的「測試報告」。故利用 Google Analytics，可以強化軟體自動化測試的結果呈現，並減少程式碼的開發及資料庫的設計，達到降低測試之成本。

四、技術分析

Google Analytics 技術架構如圖 6 所示[5]。

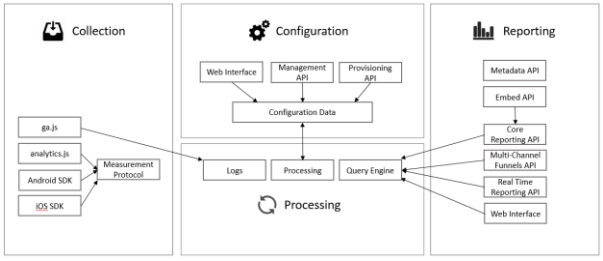

圖 6：Google Analytics 技術架構圖

其架構分為 Collection(收集)、Configuration & Processing（設定＆運算)、 Reporting（報表）三大部分。

Collection：以 JavaScript 追蹤碼(或是 Native SDK)收集使用者行為及瀏覽量，如點閱事件或瀏覽頁面等，並收集載體的資料，例如瀏覽器、手機裝置等。

Configuration & Processing：以管理者所設定的條件來處理原始資料，如篩選器（filter），透過設定使每個資料檢視（View）有各自不同的數據。

Reporting：Google Analytics 分成五大類報表，每類報表有各自意義，其亦提供眾多 API 服務，如 Core Reporting API，供使用者於自己的應用程式內取得 Google Analytics 數據。

Google Analytics 收集資料及追蹤原理如圖 7 所示。

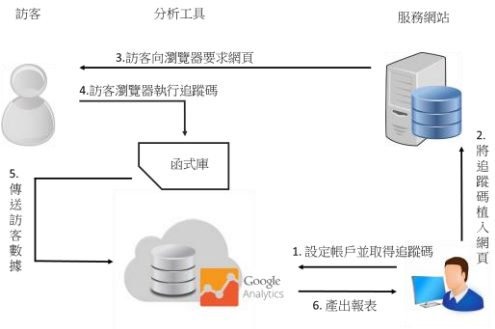

圖 7：Google Analytics 原理圖

（1）設定取得追蹤碼(T.C)

（2）將追蹤碼植入網頁(Mobile)

（3）訪客向瀏覽器要求網頁

（4）訪客瀏覽器執行追蹤碼

（5）建立 Cookie，獲取並傳送訪客數據

（6）產出報表

Google Analytics 其報表架構分為即時報表(Real)、目標對象(Audience)、客戶開發(Acquisition)、站內行為(Behavior)、轉換(Conversion)，自訂（Customization）每類報表所帶呈現的數據皆不同，維度、指標的依據也不相同。

Real：即時報表為當下訪客之瀏覽行為，以 30 分鐘內為依據；

Audience：目標對象記錄了訪客的資訊，其維度如地區、語言、國家、瀏覽器、作業系統等；

Acquisition：客戶開發即訪客的來源，此報表說明了訪客是從何管道到訪，如廣告、關鍵字等媒介；

Behavior：站內行為即訪客的行為記錄，包含了到達頁面、離開頁面，甚至是開發人員自行定義的操作行為；

Conversion：轉換報表則可設定追蹤目標轉換率、轉換價值、電子商務成效用以評估網站價值等等。

Customization：自訂報表，其維度指標可依使用者需求做設定，產出使用者需求之報表。

由以上說明得知，測試所需的維度如裝置版本、事件紀錄，其資訊分別記錄於 Google Analytics 之目標對象（Audience）及行為報表（Behavior）中，並依自訂報表（Customization），設計出符合測試之報表。以下將繼續說明如何操作 Google Analytics，將其應用於軟體自動化測試。

Google Analytics 管理結構如圖 8 所示，一個 Google 帳號至多可以管理 100 個帳戶(Account)，每個帳戶為獨立營運單位；而一個帳戶至多可以追蹤 50 個資源(Property)，其資源可以為網站或 APP，各個資源含各自的程式追蹤碼，供開發人員安裝設定追蹤其網站或 APP 之流量；每個資源至多可以劃分 25 個資源數據(View)，每一個資源數據可以有不同的設定，用以做不同的數據分類及分析。

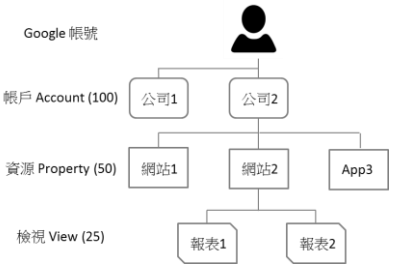

圖 8：Google Analytics 帳戶管理架構

一般安裝追蹤碼的方式為設定 Property，取得一組追蹤編號及追蹤程式碼，將追蹤程式碼安裝置網站中，通常為一段 JavaScript；若追蹤的資源為行動應用程式，可以透過提供之 iOS/Android SDK，於原生程式碼中安裝設定；若為 Hybrid APP 開發

模式，其亦提供 JS 函式庫供使用。

追蹤碼安裝後，當訪客來到該網站存取網頁，其會載入函式庫，建立 Cookies，並傳送訪客行為數據、使用環境等資訊至 Google Analytics，若欲記錄訪客之非瀏覽行為，如自行定義的點擊事件，Google Analytics 提供「事件追蹤碼」，透過設定事件參數，即可收集、分析非瀏覽之互動，其事件參數定義如表 1。

參數	條件	格式	說明
類別 Category	必填	String	事件的名稱
動作 Action	必填	String	事件的動作
標籤 Label	選填	String	事件的說明
值 Value	選填	Number	量化價值

表 1：事件參數設定

透過事件參數的設定，使用者可以自行定義事件，記錄額外的資訊，以達到 Log 之效果，本文接下來將以 Hybrid App 開發為例，說明如何將 Google Analytics 設定於程式，並應用於軟體自動化測試。

五、系統設計

以電影院購票 APP 為例，一個電影院的基本購票流程為(1)選擇電影(2)選擇影城(3)選擇日期時間(4)輸入卡號(5)選擇張數(6)劃位選座(7)購票成功，其功能如圖 9 所示

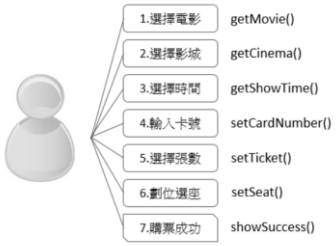

圖 9：購票流程示意圖

定義 7 個 Function，分別處理購票的基本流程，並利用 Google Analytics 事件追蹤作為 Log 紀錄，其事件設計如下：

類別 - 購票紀錄
動作 - 選擇電影、選擇影城、選擇時間等等
標籤 - 選擇項目
程式碼如下：

```
ga_storage.trackEvent("購票紀錄", "1.選擇電影",movie );
ga_storage.trackEvent("購票紀錄", "2.選擇影城",cinema);
ga_storage.trackEvent("購票紀錄", "3.選擇時間",time);
ga_storage.trackEvent("購票紀錄", "4.輸入卡號",cardNo);
ga_storage.trackEvent("購票紀錄", "5.選擇張數",num);
ga_storage.trackEvent("購票紀錄", "6.劃位選座",seat);
ga_storage.trackEvent("購票紀錄", "7.購票成功",success);
```

將其程式碼（追蹤碼）嵌入至於各個 Function，藉由 Log 各個階段的資訊，檢驗程式是否正確進入函式，如圖 10。

圖 10：Google Analytics 即時事件報表

於「行為報表」檢視事件行為紀錄，如圖 11。

圖 11：Google Analytics 行為報表

於「目標對象報表」檢視載體的資料，如裝置版本及作業系統版本，如圖 12。

圖 12：Google Analytics 目標對象報表

若要將「事件」及「裝置」等維度一同顯示，由「自訂報表」設定，產出符合使用者之測式報表，其用於 APP 開發測式，所需維度設定為「行動裝置資訊」、「作業系統版本」、「事件類別」，指標設定為「平均工作時間」，用以紀錄成式的工作時間，設定及報表如圖 13、圖 14。

圖 13：Google Analytics 自訂報表-維度設定

行動裝置資訊	作業系統版本	事件類別	事件動作	活動標籤	平均工作階段時間長度
1. Asus Z00LD Zenfone 2 Laser	5.0.2	購票紀錄	1選擇電影	玩命關頭 (6	00:01:56
2. Asus Z00LD Zenfone 2 Laser	5.0.2	購票紀錄	2選擇影城	信義影城	00:01:56
3. Asus Z00LD Zenfone 2 Laser	5.0.2	購票紀錄	3選擇時間	20130511-1:00	00:01:56
4. Asus Z00LD Zenfone 2 Laser	5.0.2	購票紀錄	4輸入卡號	4311	00:01:56
5. Asus Z00LD Zenfone 2 Laser	5.0.2	購票紀錄	5選擇張數	1	00:01:56
6. Asus Z00LD Zenfone 2 Laser	5.0.2	購票紀錄	6劃位選座	D2	00:01:56
7. Asus Z00LD Zenfone 2 Laser	5.0.2	購票紀錄	7購票成功	成功	00:01:56

圖 14：Google Analytics 自訂報表

六、實作成果

接下來將以電影院 APP 為例，探討 Google Analytics 應用於自動化測試之成效及成果，。

自動化測試即藉由撰寫測試腳本，以程式呼叫待測 APP，執行於手機裝置或瀏覽器，模擬使用者操作行為如點擊、滑動等動作。目前自動化工具眾多，本文以 Appium 為例，展示 Google Analytics 與其應用之成果。

Appium 為跨行動平台之測試工具，應用於手機 APP 方面之自動化測試，可測試原生 APP 或混合式 APP，並支援多種程式語言如 JAVA、JavaScript，其工作原理如圖 15 所示[6]。

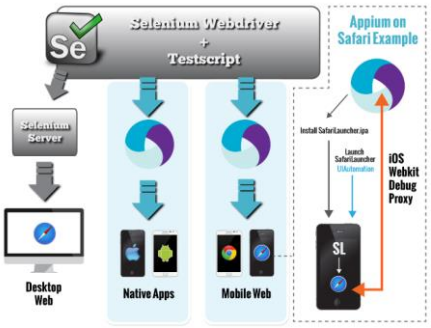

圖 15：Appium 原理圖

以電影購票 APP 為例，畫面如下（圖 16、圖 17、圖 18、圖 19）：

圖 16：(1)選擇電影

圖 17：(2)選擇影城
(3)選擇時間
(4)輸入卡號
(5)選擇張數

圖 18：(6) 選座位

圖 19：(7)確認購票

測試腳本以 Mocha[7]開發，列舉部分程式碼（圖 20、圖 21、圖 22）

```
it("1 選電影", function() {
  var timeouts = [];
  _.each(drivers, function(driver) {
    var result = driver
      .sleep(6000)
      .execute(function() {
        tapClick({
          target: $('.movie-Info')[2]
        });
      }, [], null)
      .sleep(2500);
    timeouts.push(result);
  });
  return q.allSettled(timeouts);
});
```

圖 20：測試腳本 - 選擇電影

```
xit("填卡號 ", function() {
  var timeouts = [];
  _.each(drivers, function(driver) {
    var result = driver
      .sleep(1500)
      .execute(function() {
        tapClick({
          target: $('[ng-model="cardNum1"]')[0]
        });
      }, [], null)
      .sleep(500)
      .elementByCss('[ng-model="cardNum1"]').sendKeys('4311').sleep(1000)
      .sleep(500)
      .elementByCss('[ng-model="cardNum2"]').sendKeys('7854').sleep(1000)
      .sleep(500)
      .elementByCss('[ng-model="cardNum3"]').sendKeys('2222').sleep(1000)
      .sleep(500)
      .elementByCss('[ng-model="cardNum4"]').sendKeys('2226').sleep(1000)
      .execute(function() {
        $('[ng-model="cardNum4"]').empty().blur();
      }, [], null)
      .sleep(3000);
    timeouts.push(result);
  });
```

圖 21：測試腳本 - 卡號輸入

```
it('滑', function() {
  _.each(driver, function(driver) {
    driver.executeScript(function() {
      $('[delegate-handle="MainScroll"]').trigger('goDown', [400]);
    });

  });
});
```

圖 22：測試腳本 –
滑至最底部（為了點擊底部確認按鈕）

測試情境為一個完整之購票流程，並於三個手機裝置（iOS 8、Android 4.2、Android 5）執行 Appium 自動化測試，以相同的測試腳本，檢視其所得到之結果報告，如圖 23：

	行動裝置資訊	作業系統版本	行動裝置型號	事件動作	活動標籤	平均工作階段時間長度
1.	HTC PJ83100 One X	4.2.2	PJ83100	1選擇電影	玩命關頭 (6	00:17:23
2.	HTC PJ83100 One X	4.2.2	PJ83100	2選擇影城	信義影城	00:17:23
3.	HTC PJ83100 One X	4.2.2	PJ83100	3選擇時間	20130511-1:00	00:17:23
4.	HTC PJ83100 One X	4.2.2	PJ83100	4輸入卡號	4311	00:17:23
5.	HTC PJ83100 One X	4.2.2	PJ83100	5選擇張數	1	00:17:23
6.	Apple iPhone	8.4.1	iPhone	1選擇電影	玩命關頭 (6	00:17:17
7.	Apple iPhone	8.4.1	iPhone	2選擇影城	信義影城	00:17:17
8.	Apple iPhone	8.4.1	iPhone	3選擇時間	20130511-1:00	00:17:17
9.	Apple iPhone	8.4.1	iPhone	4輸入卡號	4311	00:17:17
10.	Apple iPhone	8.4.1	iPhone	5選擇張數	1	00:17:17
11.	Apple iPhone	8.4.1	iPhone	6劃位選座	D2	00:17:17
12.	Apple iPhone	8.4.1	iPhone	7購票成功	成功	00:17:17
13.	Asus Z00LD Zenfone 2 Laser	5.0.2	Z00LD	1選擇電影	玩命關頭 (6	00:15:17
14.	Asus Z00LD Zenfone 2 Laser	5.0.2	Z00LD	2選擇影城	信義影城	00:15:17
15.	Asus Z00LD Zenfone 2 Laser	5.0.2	Z00LD	3選擇時間	20130511-1:00	00:15:17
16.	Asus Z00LD Zenfone 2 Laser	5.0.2	Z00LD	4輸入卡號	4311	00:15:17
17.	Asus Z00LD Zenfone 2 Laser	5.0.2	Z00LD	5選擇張數	1	00:15:17
18.	Asus Z00LD Zenfone 2 Laser	5.0.2	Z00LD	6劃位選座	D2	00:15:17
19.	Asus Z00LD Zenfone 2 Laser	5.0.2	Z00LD	7購票成功	成功	00:15:17

圖 23：自動化測試結果之報表

由 Google Analytics 報表得知，iOS 8 及 Android 5 皆完成購票流程，Android 4.2 則中斷於劃位選座的階段，此時再去檢驗 Android 4.2 裝置，並依照所紀錄之行為去實際操作一次(一般自動化測試無記錄諸參數)。依序選擇玩命關頭 6 → 信義影城 → 20130511-1:00 → 4311 → 1 張，其畫面如圖 24。

圖 24：Android 4.2 畫面圖

發現下方 Menu 選單遮蔽住「劃位選座」之按鈕，導致 Android 4.2 流程無法順利進行，進一步測試追蹤，得知 Android 4.2 並不支援某些 CSS 語法，此例為 Margin 語法失作用，導致頁面產生遮蔽問題。

七、結論

自動化測試只是使機器自動操作軟體程式，其缺乏報表資訊儲存所有的執行情況，若要存取各個操作紀錄及載體資訊，則需要仰賴資料庫、程式碼開發或第三方工具，若所需的維度資料愈多，其所需之程式碼也會愈多，增加開發成本如表 2。以 Google Analytics 應用於自動化軟體測試，可以很快結構化所需資料，並產出測試報告，減少設計資料庫時間成本及存取空間成本。以上述為例，以自動化測試三個手機裝置，從 Google Analytics 報表來看，僅一個裝置無法完成整個流程，此時測試人員可以很快得知該裝置之手機型號、作業系統、於哪個階段錯誤等資訊，並針對該測試報告，進行程式碼的追蹤，分析是軟體與裝置的相容問題，或是程式碼支援的問題等等。

	Non	Google Analytics
程式碼	n 行 (取決於維度數量)	1 行(僅需追蹤碼)
開發成本	高	低
空間成本	高	低
報表	Non	多

表 2：成本比較表

運用 Google Analytics，將原本的測試報告由圖 25、圖 26，強化為圖 27。並再利用 Google Analytics 之 Core Reporting API，將 Google Analytics 結構化資料取出，如圖 31，製作成客製化測試報表，如圖 28、圖 29，供主管參考。

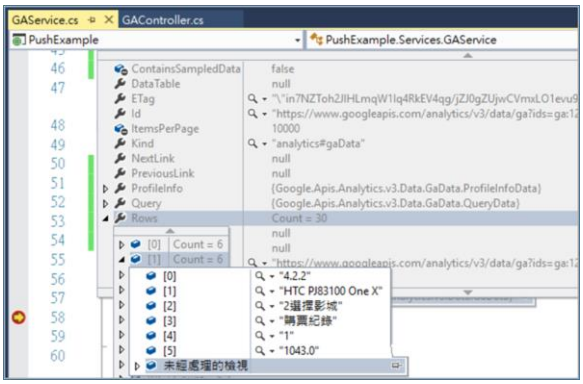

圖 31：取得結構化資料圖

運用 Google Analytics 減少了資料庫設計及資料彙整的開發成本，達到降低開發測試成本之效益，其產生的效益，值得開發人員嘗試。

專案名稱	電影APP
版本序號	V1.2
測試裝置	HTC One X
作業系統	Android 4.2
正常/錯誤	正常
備註	Non
負責人	Henry

圖 25：傳統測試報告

圖 26：自動化測試結果

圖 27：Google Analytics 測試報告

圖 28：客製化測試報表

圖 29：客製化測試報表

八、參考文獻

[1] Muzaffar Iqbal; Muhammad Rizwan,” Application of 80/20 rule in software engineering Waterfall Model”, 2009

[2] Karl R. P. H. Leung; W. L. Yeung,“Generating User Acceptance Test Plans from Test Cases”,2007

[3] Zhenyu Liu; Mingang Chen; Lizhi Cai , “A Novel Automated Software Test Technology with Cloud Technology”,2014

[4] Daniel Amo Filvà; María José Casany Guerrero; Marc Alier Forment,”
Google analytics for time behavior measurement in Moodle”,2014

[5] Justin Cutroni, O'Reilly Media,” Google Analytics 網頁分析：了解您的網站訪客”,2011

[6] Ville-Veikko Helppi ,”The Basics of Mobile Web Testing on Real Devices Using Selenium”, http://testdroid.com/tech/the-basics-of-mobile-web-testing-using-appium-selenium,2015

[7] K. V. Aiya, Hemdutt Verma,” Keyword driven automated testing framework for web application”,2014

[8] Daniel Amo Filvà ; María José Casany Guerrero ; Marc Alier Forment,” Google analytics for time behavior measurement in Moodle”,2015