

擴充 Selenium IDE 網頁測試案例錄製工具 用以自動識別未命名之視窗與內嵌框架

Extending Selenium IDE Test Case Recording Tool for Automatically Locating Unnamed Windows and Inner Frames

李信杰^{1,2}、游傑麟²

¹ 國立成功大學計算機與網路中心

² 國立成功大學資訊工程學系

Shin-Jie Lee, Chieh-Lin Yu

Email: jielee@mail.ncku.edu.tw, ygl0118@gmail.com

摘要

Selenium 是一套可以直接驅動真實瀏覽器以進行網頁功能性測試之熱門軟體，而 Selenium IDE 工具更提供了測試者以操作網頁的方式錄製(Record)測試腳本，並提供了測試腳本自動播放功能(Playback)。然而，此工具其中一個重要的問題為無法正確的選取未命名(Unnamed)之彈出視窗(Window)與內嵌框架(Inner Frame)，因而無法自動產生相對應的測試腳本，而此問題卻經常發生於現今許多網站中。在此研究中，我們嘗試了解此工具程式架構、運作規則到測試腳本生成，進而提出一個擴充 Selenium IDE 方法以解決此問題，協助使用者在錄製測試腳本時，自動識別未命名之視窗與內嵌框架，並產生相對應之測試腳本，解決使用者手動新增與修正測試案例之情形。實驗結果顯示，在 36 種已命名與未命名視窗與內嵌框架 Pairwise 測試案例中，此擴充皆能正確地自動錄製，相較於原工具僅能錄製到 4 種測試案例，此擴充方法有著顯著效果。

關鍵字：Selenium IDE, Test Case Recording, Unnamed Window, Unnamed Inner Frame.

1. 前言

在雲端蓬勃發展過程中，近年來出現測試即服務(Testing as a Service, TaaS)為導向的雲端時代[1], [2]。以網頁測試為例，目前有各式各樣的網頁測試軟體，也因此產生出許多線上網頁測試平台，對於網頁工程師在開發各種網站的過程中，於開發的最後階段通常需經過一道軟體測試的檢驗，方能將系統上線使用。軟體測試檢驗有許多方法，如 Visual Studio 壓力與網頁測試商業軟體[3]，不僅成本高昂且往往僅能相容自家或某一程式語言所開發的網站作各種測試，也因此開源碼網頁測試軟體逐漸成熟並且廣泛的被網站測試人員使用。

跨平台測試主要為針對不同平台間的交互作

用做測試[7]，這裡的跨平台通常是指「異質」的平台，像是作業系統有 Windows、Mac OS 與 Linux 等，每個作業系統都有各自的系統架構與不同表現，若開發程式的過程中沒有注意到其他平台應注意的條件，很容易就因為平台不同而產生開發上的問題[8]，所以對於程式開發者來說，跨平台測試是非常重要的事情[9]。在開源的世界中常見的跨平台測試工具包含自動化功能測試框架工具(如 Robot Framework)、Web 介面 GUI 自動化測試工具(如 Selenium)等，都是目前相當熱門的測試工具。

Selenium [4][6] 是一套可以直接驅動真實瀏覽器以進行網頁功能性測試之熱門軟體，而 Selenium IDE 工具更提供了測試者以操作網頁的方式錄製(Record)測試腳本，並提供了測試腳本自動播放功能(Playback)。然而，現在的 Selenium IDE 在錄製的過程中，其中一個重要的問題在於對於網頁上視窗(Window)與內嵌框架(Inner Frame)的錄製容易出現錯誤，原因是其錄製方式是以視窗或內嵌框架是否有對應的 Name 為準，而如果一開始沒有賦予 Name 的話，就會發生選取錯誤的情形。

雖然 Selenium IDE 是目前相當熱門的網頁 UI 測試軟體，尤其是該軟體的錄製功能更進進了實用性，讓使用者能夠隨時錄製測試腳本並且加以運用，但是在錄製腳本的過程中往往還是會遇到許多問題，使錄製功能有所限制，雖然說可以手動更改測試腳本使它達到使用者的需求，但這反而增加了使用者的不便性，費時費力、欠缺便利性[5]。

在此研究中，我們嘗試了解此工具程式架構、運作規則到測試腳本生成，進而提出一個擴充 Selenium IDE 方法以解決此問題，協助使用者在錄製測試腳本時，自動識別未命名之視窗與內嵌框架，並產生相對應之測試腳本，解決使用者手動新增與修正測試案例之情形。實驗結果顯示，在 36 種已命名與未命名視窗與內嵌框架 Pairwise 測試案例中，此擴充皆能正確地自動錄製，相較於原工具僅能錄製到 4 種測試案例，此擴充方法有著顯著效果。

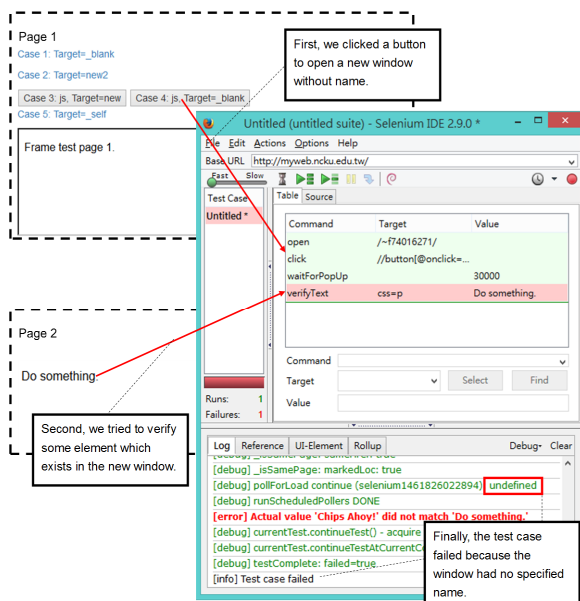


圖 1 彈出視窗沒有 Name 的失敗案例

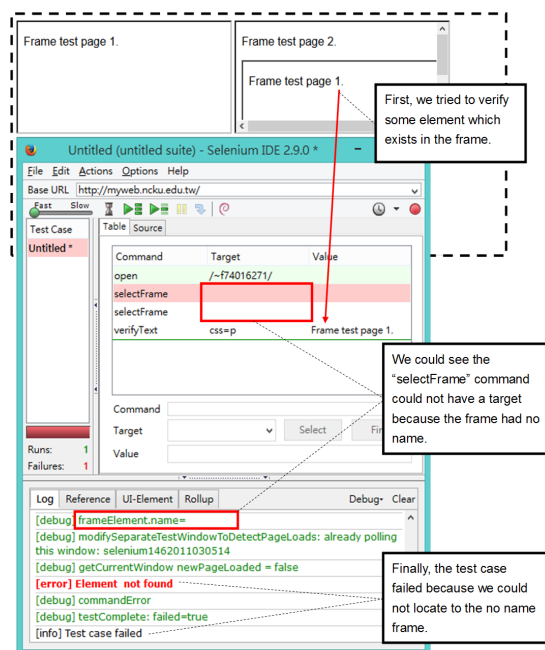


圖 2 內嵌框架沒有 Name 的失敗案例

2. Selenium IDE 介紹

Selenium [4] 是一套可以直接用真實的瀏覽器來測試頁面上 UI 的網頁 UI 測試軟體，建立測試案例的方式可以使用 Selenium IDE 工具來執行錄製的工作，IDE 目前是以 Firefox 外掛工具的方式來實作，因此只能用在 Firefox 瀏覽器上，透過工具可以以操作網頁的方式錄製操作行為並且產生腳本，而產生出來的腳本可以指定其他類型的瀏覽器來執行測試，也因此 Selenium 已被廣泛的用來做跨平台網頁自動測試[10]。

Selenium 可以透過遠端傳送指令的方式來對不同作業系統或是不同的瀏覽器執行測試案例[11],[12]，它還更支援多種程式語言格式的輸出，提供網頁有不同的需求時有更多的選擇。除了以直接設定然後執行測試案例的方式以外，Selenium 能夠透過連線的方式與作為節點的機器互相連結，並且將測試案例的腳本傳送給節點，節點機器在收到測試案例以後會自動執行測試並且把報告結果回傳給主機，讓測試軟體的實用彈性更廣。本研究欲得知以 Selenium IDE 錄製腳本時會遇到的問題，先從多次測試中尋找可能出現的問題，彙整問題的情況以後再加以解決。經過多次實驗與測試結果，目前 Selenium IDE 其中一個很重要的議題為解決未命名視窗造成的問題，因此此研究試圖提出一個擴充方法以解決此問題。

3. 針對未命名視窗與內嵌框架之 Selenium IDE 擴充方法

以圖 1 為例，使用者欲在 Page 1 點擊按鈕以開啟 Page 2，並且驗證 Page 2 中的文字是否能夠正確存取。錄完測試腳本並播放後，IDE 模擬動

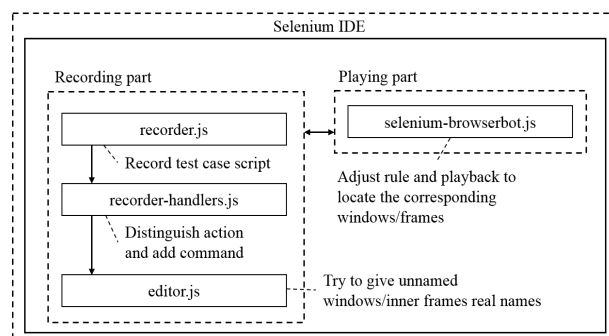


圖 3 Selenium IDE Extension 系統架構圖

作點擊按鈕，由於目標為_blank，因此開啟了一個不具有 Name 的彈出視窗，但測試案例最後失敗了，原因在於視窗沒有明確的 Name，如圖中紅框處所示，彈出視窗的 Name 被設定成 Undefined，因此無法正確的選取該視窗，使得驗證指令失敗。

以圖 2 為例，使用者欲驗證某個內嵌框架中的文字是否能夠正確存取。錄完測試腳本並播放後，IDE 執行驗證動作，由於目標在不具有 Name 的內嵌框架，測試案例最後失敗了，原因在於內嵌框架沒有明確的 Name，如圖中紅框處所示，因此並無法選取正確的視窗，使得驗證指令失敗。

3.1 系統架構

我們將先從 Firefox 的附加元件開始著手，並且深入了解 Selenium IDE 的構成方式以及理解腳本錄製的方法，從中套用各種不同的方式，或者是新增一套新的功能，去改良原本 Selenium IDE 遇到或是無法解決的問題，使錄製腳本能夠更加順利。本研究提出的系統架構如圖 3 所示，Selenium

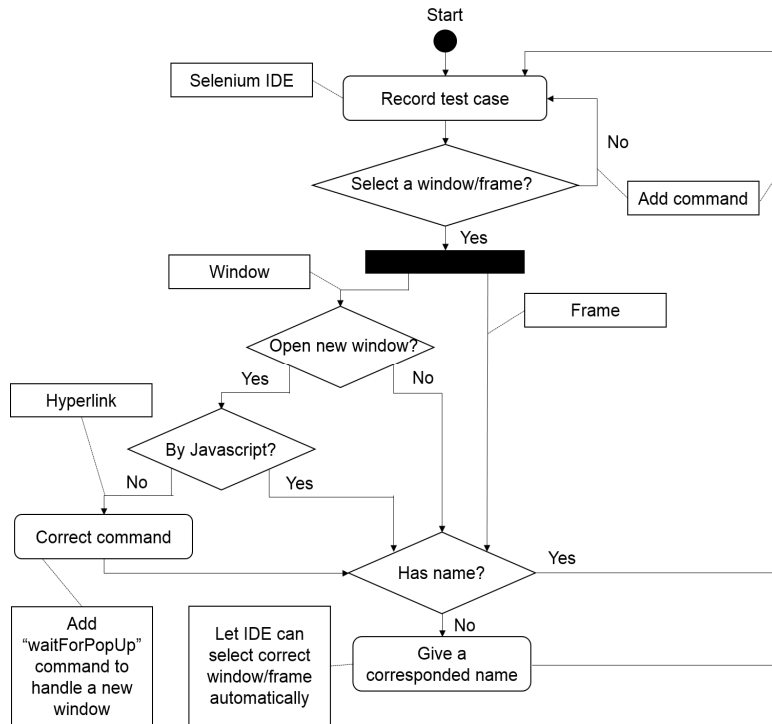


圖 4 研究步驟

IDE 整體構成大致上可以把它視為三個部分，分別是 IDE 的 UI 樣板、錄製指令與播放指令，主要的問題皆在於錄製功能的差異與播放時對指令的辨別，因此理解腳本錄製方式是非常重要的。錄製的部分主要有三份檔案，recorder.js 用來執行最基礎的錄製功能，recorder-handlers.js 則是對於使用者錄製時的動作作為指令辨別的依據，editor.js 則是判斷最後應為哪一項指令並且添加到測試腳本中；播放指令主要的檔案為 selenium-browserbot.js，用來針對不同的指令執行不一樣的動作。

本研究主要在於修正各項檔案中所遺漏的條件，及添加所需的元素，讓使用者在錄製與播放測試腳本不再因為 Unnamed Windows/Inner Frames 所造成的錄製問題而感到困擾，而且能夠順利自動化完成測試錄製。

3.2 問題分析與研究步驟

未命名視窗(Window)的問題可分成兩個部分，分別是 Window 與 Frame 兩方面的問題，而 Window 又可以細分成兩部分，分別是使用超連結(<a>)以及使用 Javascript(window.open())兩種方式所產生的視窗問題，而所有問題主要發生的癥結點皆在於 Name 的有無，因此便從解決 Name 的問題開始著手。

圖 4 為研究步驟。Selenium IDE 在錄製指令時，判斷的方式通常是使用 Javascript 的功能去作指令的判別，因此當錄製到一個彈出視窗的時候，如果是使用 window.open()的方式的話，IDE 可以

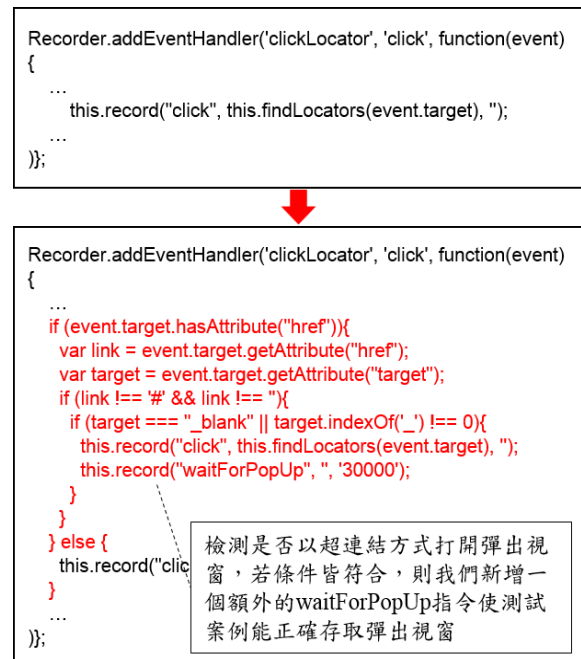


圖 5 在 recorder-handlers.js 中修正彈出視窗問題

正確的辨識為新視窗出現，但如果是使用點選超連結的方式，IDE 本身並不認為這個動作是開啟一個新視窗，因此在對新視窗做動作的時候，執行測試案例會卡住，停止在開啟新視窗的動作以前。

以圖 5 做示範，若要解決這個問題，就是在點擊超連結，也就是指令為 Click 的時候判斷這一

```
function Editor(window) {
  ...
}

Editor.prototype.addCommand =
function (command, target, value, window, insertBeforeLastCommand) {
  ...
  if (command != 'open' && command != 'selectWindow' && command != 'selectFrame') {
    ...
    if (!this.safeLastWindow.isSameWindow(window)) {
      if (this.safeLastWindow.isSameTopWindow(window)) {
        ...
      } else {
        // popup
        var windowName = window.name;
        if (windowName == "") {
          this.addCommand('selectWindow', 'null', "", window);
        } else {
          this.addCommand('selectWindow', "name=" + windowName, "", window);
        }
      }
    }
    ...
  }
}
```

```
function Editor(window) {
  ...
  this.count = 1;
}

Editor.prototype.addCommand =
function (command, target, value, window, insertBeforeLastCommand) {
  ...
  if (command != 'open' && command != 'selectWindow' && command != 'selectFrame') {
    ...
    if (!this.safeLastWindow.isSameWindow(window)) {
      if (this.safeLastWindow.isSameTopWindow(window)) {
        ...
      } else {
        // popup
        var windowName = window.name;
        if (windowName == "") {
          window.name = "win_ser_" + this.count;
          this.count += 1;
          this.addCommand('selectWindow', "name=" + window.name, "", window);
        } else {
          this.addCommand('selectWindow', "name=" + windowName, "", window);
        }
      }
    }
    ...
  }
}
```

在Editor中自定義count變數，
以作為後續為視窗命名之依據

若彈出視窗不具有Name時，我們賦予這個視窗一個win_ser_(Window Serial Number)序列號碼的Name當作固定標籤，為之後選取視窗的依據，每賦予一次Name序列號碼就會加1，因此每個視窗的Name皆會有所不同。

圖 6 在 editor.js 中修正視窗沒有 Name 的部分程式碼

個 Click 的動作是否需要開啟一個新視窗，因此當 IDE 偵測到點擊動作的時候，先檢查該元素是否存在 href (為一個超連結的方式)，再來檢查 href 所連結到的地方是否為一個合法的網域，接下來再檢查 target 的目標視窗是否為開啟一個新視窗，如果上述條件皆符合，則為 IDE 新增一個 waitForPopUp 的指令在 click 的後方，測試案例便能順利通過。

當 Selenium IDE 在錄製的過程中開啟了新的視窗，便會想辦法取得該視窗的 Name 以作為 selectWindow 指令對應視窗的依據，當目標視窗並沒有指定的 Name 時，IDE 會自動幫該視窗產生一個亂數的 Name 以達到錄製的目的，但是在執行測試案例的時候，開啟新視窗的時候 IDE 又為該視窗產生了一個新的亂數 name，因此當執行到 selectWindow 指令時需要選取對應 Name 的視窗，新視窗的 Name 與錄製時的 Name 的亂數並不相同，以致於無法正確的選取彈出視窗。

```
var BrowserBot = function(topLevelApplicationWindow) {
  ...
  this.modalDialogTest = null;
  this.recordedAlerts = new Array();
  this.recordedConfirmations = new Array();
  this.recordedPrompts = new Array();
  this.openedWindows = {};
  ...
};
```

```
var BrowserBot = function(topLevelApplicationWindow) {
  ...
  this.modalDialogTest = null;
  this.recordedAlerts = new Array();
  this.recordedConfirmations = new Array();
  this.recordedPrompts = new Array();
  this.openedWindows = {};
  this.openedWindows["win_ser_local"] = this.topWindow;
  this.count = 1;
  ...
};
```

在BrowserBot中自定義count變數，以作為後續為視窗命名之依據，並且將初始視窗的Name從空初始化為win_ser_local

圖 7 在 selenium-browserbot.js 中修正初始化視窗的部分程式碼

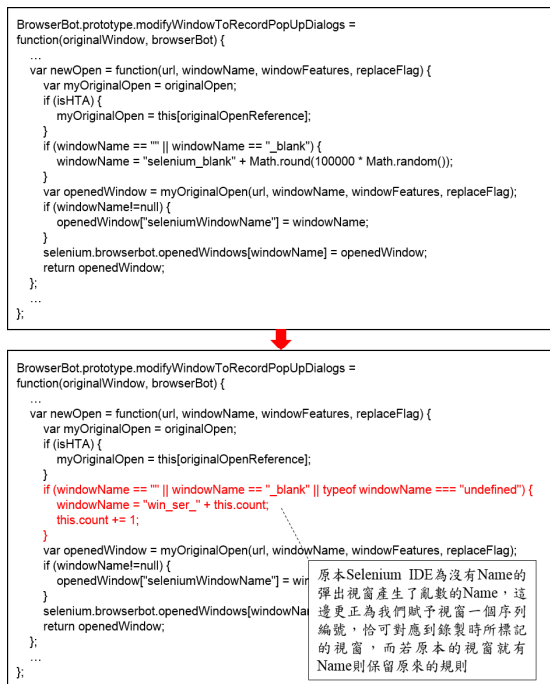
```
BrowserBot.prototype._modifyElementTarget = function(element) {
  if (element.target) {
    if (element.target == "_blank" || /^selenium_blank/.test(element.target)) {
      var tagName = getTagName(element);
      if (tagName == "a" || tagName == "form") {
        var newTarget = "selenium_blank" + Math.round(100000 * Math.random());
        this.browserbot.openWindow("", newTarget);
        element.target = newTarget;
      }
    }
  }
};
```

```
BrowserBot.prototype._modifyElementTarget = function(element) {
  if (element.target) {
    if (element.target == "_blank" || /^selenium_blank/.test(element.target)) {
      var tagName = getTagName(element);
      if (tagName == "a" || tagName == "form") {
        var newTarget = "win_ser_" + this.count;
        this.count += 1;
        this.browserbot.openWindow("", newTarget);
        element.target = newTarget;
      } else {
        this.browserbot.openWindow("", element.target);
      }
    }
  }
};
```

原本Selenium IDE為沒有Name的彈出視窗產生了亂數的Name，這邊更正為我們賦予視窗一個序列編號，恰可對應到錄製時所標記的視窗，而若原本的視窗就有Name則保留原來的規則

圖 8 在 selenium-browserbot.js 中修正超連結對應彈出視窗的部分程式碼

以圖 6做示範，若要解決這個問題，必須先把 IDE 所產生亂數問題解決掉，因此在錄製時開啟了一個新視窗，而這個視窗的 Name 原本就為空的時候，便想辦法把 Name 更換為另一組容易辨識的名稱，這裡所要使用的方法為固定標籤名稱，我們使用 win_ser_(Window serial number)作為固定標籤，並且在後方加入 Count 變數從 1 開始往上加，因此如果開啟若干個沒有 Name 的新視窗



```

BrowserBot.prototype.modifyWindowToRecordPopUpDialogs =
function(originalWindow, browserBot) {
    ...
    var newOpen = function(url, windowName, windowFeatures, replaceFlag) {
        var myOriginalOpen = originalOpen;
        if (isHTA) {
            myOriginalOpen = this[originalOpenReference];
        }
        if (windowName == "" || windowName == "_blank") {
            windowName = "selenium_blank" + Math.round(100000 * Math.random());
        }
        var openedWindow = myOriginalOpen(url, windowName, windowFeatures, replaceFlag);
        if (windowName!=null) {
            openedWindow["seleniumWindowName"] = windowName;
        }
        selenium.browserbot.openedWindows[windowName] = openedWindow;
        return openedWindow;
    };
    ...
};

BrowserBot.prototype.modifyWindowToRecordPopUpDialogs =
function(originalWindow, browserBot) {
    ...
    var newOpen = function(url, windowName, windowFeatures, replaceFlag) {
        var myOriginalOpen = originalOpen;
        if (isHTA) {
            myOriginalOpen = this[originalOpenReference];
        }
        if (windowName == "" || windowName == "_blank" || typeof windowName === "undefined") {
            windowName = "win_ser_" + this.count;
            this.count += 1;
        }
        var openedWindow = myOriginalOpen(url, windowName, windowFeatures, replaceFlag);
        if (windowName!=null) {
            openedWindow["seleniumWindowName"] = win
        }
        selenium.browserbot.openedWindows[windowName] = openedWindow;
        return openedWindow;
    };
    ...
};

```

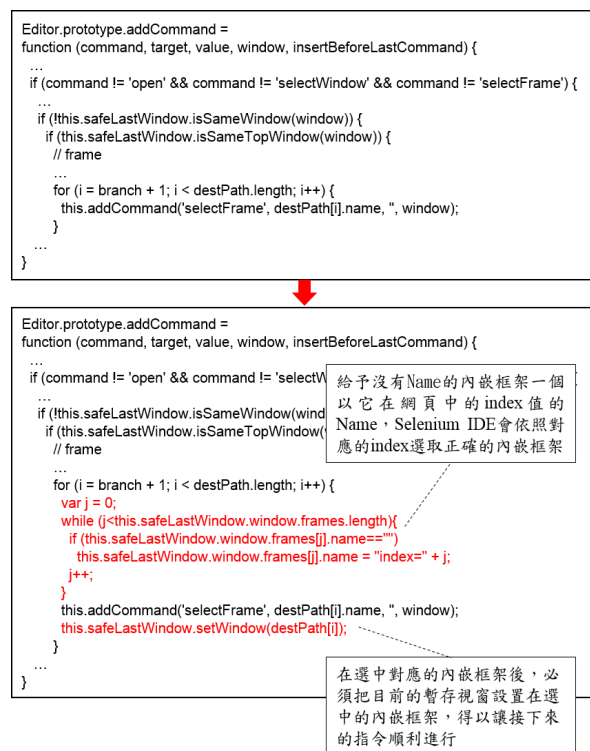
原本Selenium IDE為沒有Name的彈出視窗產生了一個亂數的Name，這邊更正為我們賦予視窗一個序列編號，恰可對應到錄製時所標記的視窗，而若原本的視窗就有Name則保留原本的規則

圖 9 在 selenium-browserbot.js 中修正 Javascript 對應彈出視窗的部分程式碼

時，便會使用這套方法來賦予 name。

另一方面則是在執行測試案例的時候，為了讓 selectWindow 可以選到正確的對應視窗，因此執行到開啟新視窗指令的同時，也使用了同一套方法賦予相對應的 Name，因此 selectWindow 便能選取到正確的視窗。以圖 7 為例，原本的視窗預設是沒有 Name 的，因此我們先給它一個 win_ser_local 的 Name 來做辨識，接著與錄製時一樣初始化一個 Count 變數。接下來在執行測試案例後遇到同一個沒有 Name 的視窗時，以圖 8 為例，IDE 便會自動為它產生一個我們定義的變數，並且對應到錄製時的同一個視窗的 Name，因此 selectWindow 便能選取到正確的視窗。以圖 9 為例，以 Javascript 產生的彈出新視窗所遇到的問題一樣為 Name 的有無，因此解決方法與超連結的方式相同。

當 Selenium IDE 在錄製的過程中錄製到 Frame 的視窗，一樣會想辦法要去取得 Frame 的 Name 以作為對應 Frame 的依據，因此如果 Frame 沒有 Name 的話，就沒有辦法順利取得對應的依據而造成指令錯誤。以圖 10 做示範，若要解決這個問題，就要先想辦法賦予 Frame 一個 Name 作為選取的依據，這邊我們使用的方式為取得 frame 在該網頁之中的 index 值，因為 IDE 在播放的過程中可以依據 index 來選取對應的 Frame，因此在錄製時如果能夠給予正確的 index，如此一來測試案例便能順利通過。



```

Editor.prototype.addCommand =
function (command, target, value, window, insertBeforeLastCommand) {
    ...
    if (command != 'open' && command != 'selectWindow' && command != 'selectFrame') {
        ...
        if (!this.safeLastWindow.isSameWindow(window)) {
            if (this.safeLastWindow.isSameTopWindow(window)) {
                // frame
                ...
                for (i = branch + 1; i < destPath.length; i++) {
                    this.addCommand('selectFrame', destPath[i].name, "", window);
                }
            }
        }
    }
    ...
};

Editor.prototype.addCommand =
function (command, target, value, window, insertBeforeLastCommand) {
    ...
    if (command != 'open' && command != 'selectWindow' && command != 'selectFrame') {
        ...
        if (!this.safeLastWindow.isSameWindow(window)) {
            if (this.safeLastWindow.isSameTopWindow(window)) {
                // frame
                ...
                for (i = branch + 1; i < destPath.length; i++) {
                    var j = 0;
                    while (j < this.safeLastWindow.window.frames.length) {
                        if (this.safeLastWindow.window.frames[j].name == "") {
                            this.safeLastWindow.window.frames[j].name = "index=" + j;
                            j++;
                        }
                    }
                    this.addCommand('selectFrame', destPath[i].name, "", window);
                    this.safeLastWindow.setWindow(destPath[i]);
                }
            }
        }
    }
    ...
};

```

給予沒有Name的內嵌框架一個以它在網頁中的index值的Name，Selenium IDE會依照對應的index選取正確的內嵌框架

在選中對應的內嵌框架後，必須把目前的暫存視窗設置在選中的內嵌框架，得以讓接下來的指令順利進行

圖 10 在 editor.js 中修正內嵌框架沒有 Name 的部分程式碼

4. 實驗評估

本研究擴充 Selenium IDE，協助使用者在使用 Selenium IDE 錄製測試腳本時，自動識別未命名之視窗(Window)與內嵌框架(Inner Frame)，以降低使用者手動修正測試案例之情形，提供更簡易之自動化測試案例錄製操作環境。

4.1 實驗設計

在本實驗設計中，將所有對於視窗以及內嵌框架的可能情形交互測試，以達到使用者錄製彈出視窗與內嵌框架的所有可能情形。圖 11 為實驗驗證模擬情境，首先會在模擬的第一個頁面中看到六種狀況，分別為使用超連結開啟指定彈出視窗(Named window by hyperlink)、使用超連結開啟未命名彈出視窗(Unnamed window by hyperlink)、使用 Javascript 開啟指定彈出視窗(Named window by Javascript)、使用 Javascript 開啟未命名彈出視窗(Unnamed window by Javascript)、選取指定的內嵌框架(Named frame)與選取未命名之內嵌框架(Unnamed frame)，在對應的狀況執行之後，皆連結到一樣含有六種情況的第二個頁面，並且模擬同樣六種狀況，在第二個頁面對應的狀況執行之後，皆連結到相同的第三個頁面。

錄製的過程為點選第一個頁面的一種情況連結到第二個頁面，再從第二個頁面點選其中一種情況到達第三個頁面，連續錄製彈出視窗與內嵌框架

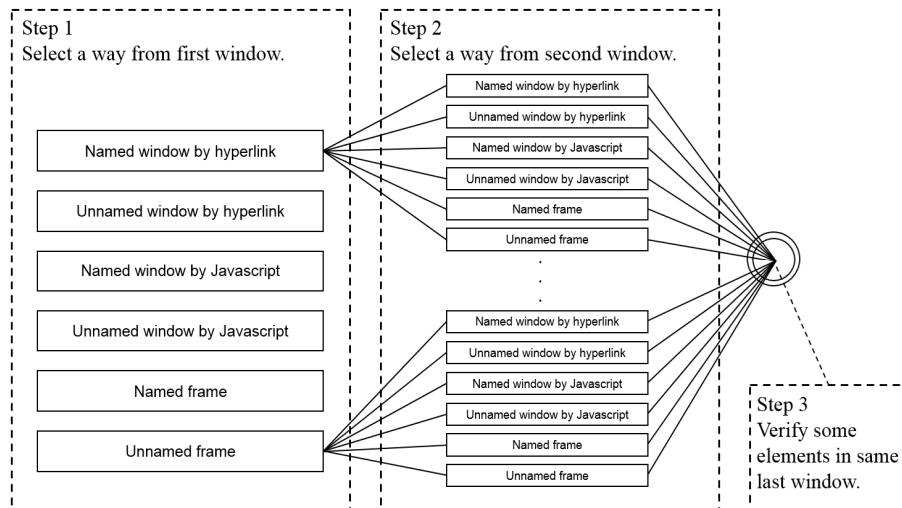


圖 11 Pairwise 實驗驗證測試案例

等情況，因此能夠產生出 6×6 ，也就是 36 個不同的測試案例，以用來模擬所有實際網頁中可能會發生的情況。第三個頁面僅存在一個元素，而此頁面用來作驗證，使用 `verifyText` 指令驗證網頁元素作為測試案例成功與否的依據，若是能夠正確的驗證第三個頁面中的元素存在，即表示錄製案例的指令成功，指令皆能順利執行並且選取正確的視窗或內嵌框架，若其中有錯誤則為失敗，表示在錄製的過程中有可能是 Selenium IDE 內部存在的問題而需要人工修正測試案例。

另外則是真實案例的情況，從真實的網站中列舉幾個熱門的網站並且含有彈出視窗或者內嵌框架的問題，使用原來的版本與修正過後的版本去錄製同一個網站的彈出視窗與內嵌框架，來驗證選取視窗在真實世界的需求與改善後的情況。

4.2 實驗結果

在本實驗中，使用原本的 Selenium IDE 錄製的測試案例並無法全部都順利進行，如表 1 所示，在全部 36 個案例之中，甚至只有四個能夠通過。如圖 1 與圖 2 的情況所示，錄製過程中若選取到未命名的視窗或內嵌框架，測試案例容易造成失敗而無法達到完全自動化的狀態，這意味著這些測試案例的指令在自動化錄製測試案例方面仍然需要改善或是得以人為方式添加所需要的指令。

另一方面，本研究提出的方法在改善 Selenium IDE 的錄製與播放方式以後，如表 2 所示，全部 36 個測試案例皆能順利通過。如圖 12 與圖 13 所示，原因在於以本研究所提出的方法，賦予原本沒有 Name 的彈出視窗與內嵌框架一個序列的標籤

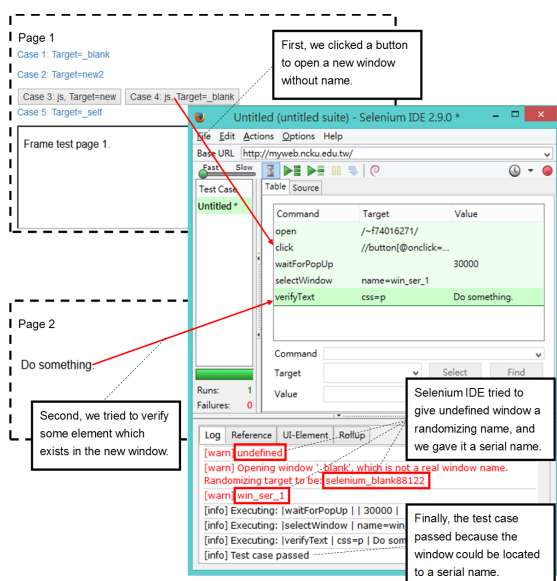


圖 12 彈出視窗沒有 Name 的成功案例

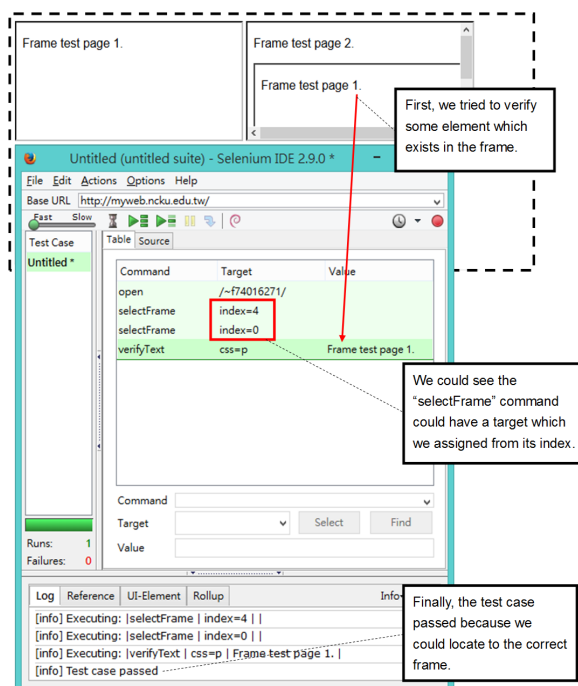


圖 13 內嵌框架沒有 Name 的成功案例

表 1 使用原本的 Selenium IDE 執行 36 種 Pairwise 測試案例之結果。

Step 1 \ Step 2	Named window by hyperlink(T1)	Unnamed window by hyperlink(T2)	Named window by Javascript(T3)	Unnamed window by Javascript(T4)	Named frame(T5)	Unnamed frame(T6)
Named window by hyperlink(T1)						
Unnamed window by hyperlink(T2)						
Named window by Javascript(T3)			V		V	
Unnamed window by Javascript(T4)						
Named frame(T5)				V	V	
Unnamed frame(T6)						

表 2 使用此研究擴充的 Selenium IDE 執行 36 種 Pairwise 測試案例之結果

Step 1 \ Step 2	Named window by hyperlink(T1)	Unnamed window by hyperlink(T2)	Named window by Javascript(T3)	Unnamed window by Javascript(T4)	Named frame(T5)	Unnamed frame(T6)
Named window by hyperlink(T1)	V	V	V	V	V	V
Unnamed window by hyperlink(T2)	V	V	V	V	V	V
Named window by Javascript(T3)	V	V	V	V	V	V
Unnamed window by Javascript(T4)	V	V	V	V	V	V
Named frame(T5)	V	V	V	V	V	V
Unnamed frame(T6)	V	V	V	V	V	V

表 3 使用此研究擴充的 Selenium IDE 執行真實網站測試案例之結果

Web name	Website	Test Case	Original Selenium IDE	Extended Selenium IDE
Twitter	twitter.com	T6	X	V
Tumblr	www.tumblr.com/explore/trending	T2 to T6 to T5	X	V
reddit	www.reddit.com/	T2	X	V
Digg	digg.com/	T2 to T2	X	V
A pop up window example in github	rip747.github.io/popupwindow/	T1	V	V
A pop up window example in javascript.info	javascript.info/tutorial/popup-windows	T2	X	V
A pop up window example in codylindley	swip.codylindley.com/popupWindowDemo.html	T4	V	V
A pop up window example in pageresource	www.pageresource.com/jscrip/jwinopen.htm	T3	V	V
A frame example in w3schools	www.w3schools.com/tags/tryit.asp?filename=tryhtml_iframe	T6 to T6	X	V
A frame example in tutorialspoint	www.tutorialspoint.com/html/html_iframes.htm	T6 to T1	X	V
A frame example in nunzioweb	nunzioweb.com/iframes-example.htm	T5 to T2	X	V
A frame example in cs.tut.fi	www.cs.tut.fi/~jkorpela/html/iframe.html	T6	X	V

或是特定名稱，使得在錄製與播放的情況下都能夠正確的選取到正確的視窗，這在錄製自動化測試案例的過程中，大大的減少了錄製上的不便利以及人為添加指令的時間，使得錄製測試案例可以更加順利的進行。

除此之外，在真實網頁測試案例的情況下，結果如表 3 所示，在全部 12 個測試案例中，前 4 個測試案例為一般網頁，中間 4 個測試案例為開啟彈出視窗範例的教學網頁，後 4 個測試案例為使用內嵌框架範例的教學網頁，其中包含在模擬測試中所涵蓋的測試內容。最後測得的結果用原本的 Selenium IDE 測試後，12 個測試案例中只有 3 個能通過，而使用本研究修正版的 IDE 則能夠全數通過，這樣的結果顯示出在真實網頁上仍然經常會遇到許多彈出視窗與內嵌框架等問題，但是由於原本的 Selenium IDE 無法正確錄製這些視窗造成自動化的失敗，而使用本研究所修正過後的 IDE 錄製測試的話，大大的減少了錄製上的不便利以及人為添加指令的時間，使得錄製測試案例可以更加順利的進行。

5. 結論

本論文中，我們提出了一個改良 Selenium IDE 在自動化錄製過程中的方法，透過擴充 Selenium IDE 的程式碼，將原本在執行錄製測試案例時會遇到無法正確選取彈出視窗與內嵌框架的問題，經過解析後使用對應的方式加以改善，並且使用我們的方法來賦予這些未命名的彈出視窗與內嵌框架一個有序列的 Name，使得不論是在錄製或者播放測試案例方面都能夠正確的選取視窗，改善原本需要人工添加指令的需求以及節省更多時間，本研究主要有以下三項貢獻：i) 使用 Selenium IDE 錄製彈出視窗或內嵌框架能夠不受原本存在的錯誤干擾，使得自動化測試案例更加順利執行。ii) 透過修正 Selenium IDE 在錄製或播放時的錯誤情況，可以讓測試案例更加自動化。iii) 研究結果經模擬與真實網站的驗證，本研究方法比起原本的方法，在測試案例的正確率上有明顯的提升，可大幅減少了錄製上的不便以及人為添加指令的時間。

6. 致謝

感謝科技部編號 MOST 103-2221-E-006-218 之計畫對本研究之經費提供與技術支援，由於科技部的支持，使本研究得以順利進行，特此致上感謝之意。

參考文獻

- [1] Gao, J.; Xiaoying Bai; Wei-Tek Tsai; Uehara, T., “Testing as a Service (TaaS) on Clouds,” *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium*, pp. 212-223, 25-28 March 2013.
- [2] Lian Yu; Wei-Tek Tsai; Xiangji Chen; Linqing Liu; Yan Zhao; Liangjie Tang; Wei Zhao, “Testing as a Service over Cloud,” *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium*, pp. 181-188, 4-5 June 2010.
- [3] microsoft.com , “Testing Performance and Stress Using Visual Studio Web Performance and Load Tests”, Retrieve from: [http://msdn.microsoft.com/en-us/library/vstudio/dd293540\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/vstudio/dd293540(v=vs.110).aspx)
- [4] Selenium. <http://www.seleniumhq.org/>
- [5] Selenium Frequently Asked Questions. <http://wiki.openqa.org/display/SEL/FAQ>
- [6] Antawan Holmes and Marc Kellogg, “Automating Functional Tests Using Selenium,” *AGILE 2006 Conference, IEEE*, ISBN 0-7695-2562-8/2006
- [7] CrossBrowserTesting. <http://www.crosbrowsertesting.com>
- [8] J. Dolson. What is “Cross-browser compatibility?” <http://www.joedolson.com/articles/2008/03/what-is-cross-browser-compatibility/>
- [9] Ali Mesbah ;Mukul R. Prasad, “Automated cross-browser compatibility testing,” *the 33rd International Conference on Software Engineering*, May 21-28, 2011.
- [10] Harpreet Kaur; Dr.Gagan Gupta, “Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete,” Harpreet kaur et al *Int. Journal of Engineering Research and Applications*, pp.1739-1743, Sep-Oct 2013
- [11] SeleniumHQ. “Selenium WebDriver.” <http://docs.seleniumhq.org/projects/webdriver/>
- [12] Nidhika Uppal; Vinay Chopra, “Design and Implementation in Selenium IDE with Web Driver,” *International Journal of Computer Applications* (0975 – 8887), Volume 46– No.12, May 2012