應用及改善網頁物件樣式之系統化網頁測試方法
A Systematic Approach for Web Testing with Page Object Pattern Application and Enhancement

Xin Hung Chen and Nien Lin Hsueh
Department of Information Engineering and Computer Science
Feng Chia University
Email: nlhsueh@mail.fcu.edu.tw

Abstract---在軟體專案開發中，軟體測試是一項非常重要的環節，而在網頁應用程式開發中亦是如此，在網頁測試中最常見的問題就是當網頁新版本發佈之後，進行回歸測試時，因為某些介面的更改或是網頁標籤的變更，造成測試案例的失效，無法正常測試，此時就要對測試案例進行維護。 使用網頁物件樣式(page object pattern) 可以提高測試案例的可維護性，它將網頁物件(page object)與測試腳本(test script)分離，降低彼此的耦合性，但是它仍有幾個問題：(1)僅將網頁模組化；(2)網頁物件(page object)並無抽象化的概念；(3)沒有用到多型的好處。 本文將提出一個系統化的網頁測試方法，說明從建立網頁物件的原則，並加入若干個測試樣式(testing pattern)解決網頁測試時會遇到的問題與困難，到將物件導向的概念與技巧融入網頁物件樣式中，進一步的提升測試案例的可維護性與彈性。我們選擇Open edX作為實驗對象，先針對較舊的版本分別使用網頁物件樣式與我們的方法建立測試案例，再將Open edX版本更新後進行回歸測試，並對測試案例進行維護，我們將記錄兩種版本的維護時間與修改的地方，檢驗我們的方法是否可以改善及加強網頁物件樣式。

Keywords---網頁測試, 回歸測試, 網頁物件樣式, 可維護性

I. 緒論

在傳統軟體專案開發的過程中，軟體測試是一項非常重要的環節，在網頁應用程式開發中亦是如此，網頁的測試方法分為三種：(1)手動測試；(2)自動化測試；(3)半自動化測試，本研究將以自動化測試方法為主軸，來探討其測試案例之可維護性的問題。

在網頁自動化測試方法中，使用網頁物件樣式(Page Object Pattern)可以提高測試案例的可維護性[1]，此方法將網頁與測試腳本分離，降低兩者之間的耦合性。但是僅將網頁與測試腳本分離我們認為效果有限，在每個網頁物件(page object)中還是有許多重複的程式碼或屬性，本研究將提出一個方法來應用及加強網頁物件樣式，提升測試案例之可維護性。

當網頁應用程式更新至新版本時，除了新功能的測試之外，為了確保原本的功能可以正常的運作，就必須進行回歸測試，而網頁測試一個很直觀的方法就是手動的來做測試，藉由人為的操作來檢查網頁的功能是否正常，但是手動測試非常的浪費時間而且成本也很高，所以自動化測試是必要的。

而自動化測試最常見的問題就是當網頁更新至新版本時，常常因為一些使用者介面的變動或是網頁標籤的更改，使得原有的測試案例會失效，如果測試案例沒有經過良好的規劃與設計，維護起來將會非常的耗時且費力。 網頁物件樣式將網頁與測試腳本分離，藉此提高測試案例的可維護性，但是它仍有下列問題：

- 僅將網頁模組化
- 網頁物件並無抽象化的概念
- 沒有用到多型的好處

如圖2所示，雖將兩個網頁模組化，但在兩個網頁中有重複的屬性與程式碼，若這些屬性在網頁版本更新後有變動，就必須回來更改這些地方，明明都是類似的程式碼，卻要更改很多次，增加了維護上的負擔，這就是上述所說的問題存在。

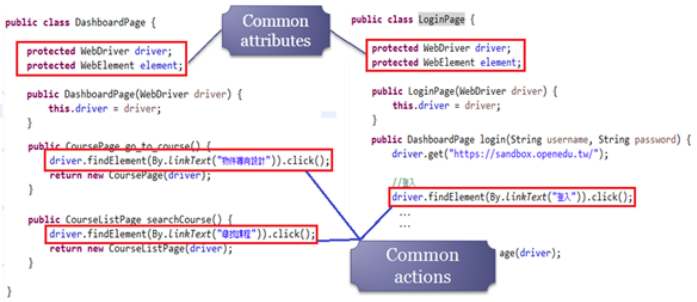


圖 1. 網頁物件中重複的屬性與動作

我們主要的目的是將這些重複的地方抽象化，將測試案例的建立加入物件導向的概念，並應用若干個測試樣式(Testing Pattern)來提高測試案例之可維護性。

本研究將提出一個系統化的網頁測試方法，應用網頁物件樣式來解決網頁測試時的各種狀況並改善網頁物件樣式在維護上的問題，我們將針對網頁物件樣式與我們提出的方法做一個實驗來比較我們的方法是否可以加強及改善網頁物件樣式。

本論文章節結構如下：第二節介紹本研究應用到的背景技術；第三節針對網頁測試的相關研究做一個討論與比較；第四節說明本論文所提出改善網頁物件樣式的方法與實驗介紹；第五節敘述整個實驗的過程；第六節為研究結論。

II. 背景技術介紹

A. 網頁物件樣式

網頁物件樣式(Page Object Pattern)的核心概念是將測試腳本與網頁分離，降低彼此的耦合性與重複的程式碼，提高其可維護性。一個網頁物件(page object)可視作是一個要測試的網頁，在網頁物件中會包含該網頁上有的屬性及動作，提供介面讓測試腳本呼叫並使用。

圖2為尚未使用網頁物件樣式來測試登入功能，它有兩個問題存在：(1)測試方法和抓取網頁物件的定位器寫一起，如果未來登入的標籤更改了，就必須回來修改測試方法；(2)若系統很多地方都會用到登入功能，要修改的地方就會非常多。

```
pulbic class Login {

    pulbic void testLogin() {
        web.type("userID", "testUser");
        web.type("password", "myPassword");
        web.click("sign-in");
        Assert.assertEquals(web.findElement("UserID"), "testUser");
    }
}
```

圖 2. 沒有使用網頁物件樣式來測試登入功能

接著我們套用網頁物件樣式來測試登入功能，將測試方法與抓取網頁的定位器分開撰寫，登入頁面如圖3；首頁如圖4所示，建立其網頁物件，登入頁面包含一個登入功能，正確登入之後會返回首頁，將兩個網頁物件建立完成之後就可以撰寫測試腳本，如圖5所示，測試腳本改用建立物件的方式來測試，如果未來登入標籤更改了，只需要修改SingInPage中的方法即可，就算很多地方都會用到登入功能，也只需要修改一個地方。

```
public class SignInPage {

    public SignInPage() {
        if(!web.getTitle().equals("Sign in page")) {
            throw new IllegalStateException("This is not sign in page");
        }
    }

    /**
     * Login as valid user
     *
     * @param userName
     * @param password
     * @return HomePage object
     */
    public HomePage loginValidUser(String userName, String password) {
        web.type("userName", userName);
        web.type("password", password);
        web.click("sign-in");

        return new HomePage(selenium);
    }
}
```

圖 3. 登入網頁物件

B. Selenium

Selenium[2]為一個網頁的自動化測試工具，它有兩個種類：(1)Selenium IDE；(2)Selenium WebDriver，前者為火狐瀏覽器(FireFox)的外掛程式，其提供自動錄製測試腳本的功能，紀錄使用者在網頁上的行為，藉此建立測試案例，可日後藉由重播的方式來進行回歸測試；後者為一個程式框架，它支援許多當今熱門的程式語言(ex: Java, Python, Ruby)，藉由寫程式的方式並使用它提供的API來建立測試案例，達到自動化測試的目的。

```
public class HomePage {

    public HomePage() {
        if (!web.getTitle().equals("Home Page of logged in user")) {
            throw new IllegalStateException("This is not Home Page of
            logged in user");
        }
    }

    public HomePage manageProfile() {
        // Page encapsulation to manage profile functionality
        return new HomePage(selenium);
    }
}
```

圖 4. 首頁網頁物件

```
public class TestLogin {

    public void testLogin() {
        SignInPage signInPage = new SignInPage();
        HomePage homePage = signInPage.loginValidUser("XinHung", "123456");
        Assert.assertTrue(web.findElement("UserID"), "XinHung");
    }
}
```

圖 5. 使用網頁物件樣式來測試登入功能

我們可將Selenium IDE視為錄製與重播(capture-replay)的測試方法，而Selenium WebDriver為可程式化(programmable)的測試方法，已有學者實驗證實可程式化的測試方法在網頁新版本發佈後測試案例的維護成本是比較低的[4]，所以本研究將以可程式化的測試方法為主軸，並使用Selenium WebDriver的程式框架來撰寫測試案例。

C. JUnit

JUnit[3]是一個Java程式語言為基礎的單元測試框架，它提供了許多便利的API來幫助我們做單元測試，本研究將以Java這個程式語言來開發測試案例，結合Selenium WebDriver與JUnit來撰寫測試案例。

D. Polymorphism

多型(polymorphism)是物件導向中的一項特性，它藉由父類別與子類別間繼承的關係讓父類別可以當成子類別的通用型態，直到程式在執行時才會知道要使用哪一個子類別，使得系統具備動態連結(dynamic binding)的特性和彈性。

III. 相關研究

關於自動化回歸測試，主要涉及的相關研究可以分為自動化測試方法、測試案例之建立與測試案例的維護三個部分，以下將分為三小節來介紹與討論相關研究的方法與比較。

A. 自動化測試方法

一般來說自動化回歸測試的方法有三個種類：(1)座標式；(2)錄製與重播(capture-replay)；(3)程式化(programmable)，第一種座標式的自動化測試方法已普遍被認為不適用於現今的網頁應用程式，Leotta等人[4]對後兩種的測試方法做了實驗比較，他們找了若干個開源網站進行實驗，先在舊版本撰寫測試案例，再把網站版本更新之後進行回歸測試，紀錄修復測試案例的時間，經由

表 I.　自動化測試方法比較

自動化測試方法	可維護性	執行效能
座標式	差	適中
錄製與重播	差	很好
可程式化	良好	很好
圖片辨識定位器	適中	略差
DOM-based定位器	良好	很好

實驗結果證實可程式化的測試方法，在維護成本上是較占優勢的，圖6可簡單說明隨著版本的演進，錄製與重播式的測試方法在維護成本上會越來越高。

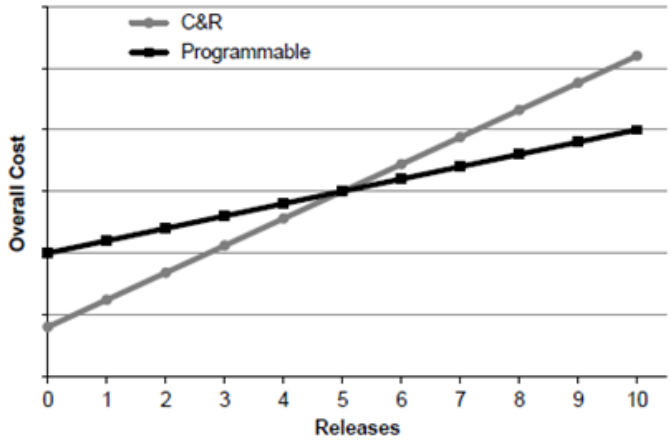


圖 6.　測試案例維護成本示意圖

Leotta等人[5]比較了在測試案例中定位網頁物件的兩種方法：(1)圖片辨識(visual-based)；(2) 文件物件模型(Document Object Model，縮寫DOM)，圖片辨識的測試方法雖然較為新穎，但是由於圖片辨識效能負擔較重，相較於DOM-based的方法，測試時執行測試案例所花的時間較多，但是在維護成本上他們並沒有得到哪一種方法絕對比較好的結論，但是大部分的實驗數據顯示DOM-based的方法成本是比較低的。他們還使用了多種定位器的方法[6]，想藉此提高測試案例的強度，經由比較一個和多個定位器，並給予定位器的權重值來判斷應該要使用哪一個定位器，實驗結果發現確實是可以增加強度，而且系統負擔也不大。

B. 測試案例之建立

Deursen[7]將狀態物件(state object)的概念融入在建立測試案例時所面臨到的問題，藉由狀態圖(state diagram)的輔助來模擬系統的各種狀態，在狀態圖中加入錯誤處理的轉換，並利用轉換樹(transition tree)來延伸每一個狀態，依照我們所設計的狀態圖來建立測試案例，可以減少在建立測試案例時所面臨的問題。

Felderer等人[8]將UML中的狀態機(state machine)和活動圖(activity diagram)應用在建立測試案例上，他們找了兩個系統分別使用狀態機和活動圖來建立測試案例，藉此想了解利用這兩種UML建立測試案例會有什麼樣的問題產生，實驗結果顯示在狀態機中最常見的問題是條件的判斷被當成是一個指令(condition as operation call)；而活動圖中最常見的則是指令被當成結果(operation call as result)，雖然活動圖的方式在實驗中被認為是較好理解的，但是在建立測試案例上他出錯的數量卻比應用狀態機來的高。

Carino等人[9]點出了在自動化測試中，建立測試案例時一個常見的問題，就是我們往往都建立了脆弱的測試案例，造成日後維護上的困難，於是他們在建立測試案例時加入了額外的機制像是自動化的追蹤程式碼(trace code)和自動化的產生測試案例，藉由這些額外的機制來建立較好的測試案例，增加測試案例的強度，降低日後維護上的困難。

C. 測試案例的維護

Collins等人[10]將自動化測試應用在敏捷開發(agile development)上，他們一開始盡可能的想用Selenium IDE達到自動化測試，但由於敏捷開發中網頁上的變化過於快速，所以他們必須一直修復或重新撰寫測試案例，但是這樣花費在測試上的時間太多了，於是他們僅將穩定的網頁來做自動化測試，不過我們認為若是採用Selenium WebDriver的方式，可以大幅地降低日後維護上的困難和成本。

Victor Hurdugaci等人[11]指出了軟體測試可以提高整體的品質，但隨著軟體的演進，若測試案例沒有跟著演進，那軟體的品質就會隨之下降，Katja Karhu等人[12]發現使用自動化測試可以提升專案的品質，但是隨之也有新的成本產生，測試案例的維護成本就是其中之一，如果測試案例的維護被忽略的話，可能整個系統維護的成本就會上升，因為可能要額外的手動測試，可見測試案例的維護是非常重要的。

IV.　研究方法

為了提高測試案例之可維護性，我們應用了網頁物件樣式(Page Object Pattern)來撰寫測試案例，但網頁物件樣式仍有上述問題，因此我們將網頁物件樣式加入了一些物件導向的技巧與應用若干個測試樣式(Testing Pattern)，想藉此解決上述問題。

本研究所提出的方法論可分為下列三個階段，如圖7所示：

- 基礎網頁物件之建立，
- 測試樣式應用，
- 網頁物件程式碼重整，

本章各小節將一一說明上述階段的詳細過程。

A. 階段一：基礎網頁物件建立

首先我們要建立需要測試的網頁物件(page object)，我們可以根據下列原則來建立：

- 以一個頁面為一個網頁物件；
- 網頁上有哪些功能就建立方法(method)來模擬該功能。

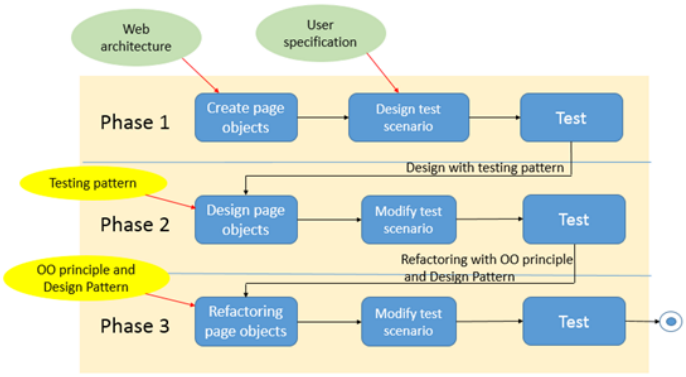


圖 7. 研究方法架構圖

接著要設計測試流程，我們可以根據下列原則來建立：

- 自己對該網站的操作經驗；
- 從使用者規格書中來設計測試流程。

從自己對該網站的操作經驗來規劃測試流程可能會不夠詳細，所以我們建議從使用者規格書中來設計，但是使用者規格書往往是以較容易理解的話語來描述網頁的功能，可能不是那麼容易抓住重點功能，所以我們可以從他的目錄中了解他有那些主要功能和次要功能，從這些地方來了解有那些功能必須要測試。

在設計測試流程中，我們可以藉由UML的輔助來幫助我們規劃測試案例，像是使用案例圖(use case diagram)、活動圖(activity diagram)與狀態機(state machine)都是常見的方法，當我們的網頁物件和測試流程都設計完之後，就可以藉由剛剛建立的UML來開發測試案例，之後去執行它確保我們設計的網頁物件和測試流程都正確無誤。

B. 階段二：測試樣式應用

在前一小節中我們建立了基本的網頁物件與測試案例，但是階段一中的方法尚未考慮到一些自動化測試時常見的問題，本小節將加入測試樣式(Testing Pattern)來解決這些問題，讓我們的網頁物件更加完整。

1)TP1: Wait and Check: 在網頁測試時，並不是所有的回應都是立即可以得到的，可能受到網路環境的限制(例：網路頻寬、網路速度)，所以我們必須等待一段時間，直到取得回應才繼續下一個步驟的測試。

以Selenium為例，我們可以設定最大等待時間，如圖8所示，若超過我們設定的時間都尚未取得回應，就會跳出例外。

```
driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(30,  TimeUnit.SECONDS);
```

圖 8. 設定最大等待時間(秒)

而在ajax的應用程式中亦是如此，ajax的特點是不用重新整理網頁，就可以進行局部的更新，所以當我們在測試ajax應用程式時，必須等待ajax引擎收到回應後，才能對它進行結果的驗證(assertion)。

2)TP2: Screen Capture: 在進行自動化測試時，偶爾會遇到無法利用程式判斷的情況，例如表單的送出(form submission)或圖片位置的判斷，我們難以用程式的方法來判斷是否有正確送出或是圖片出現在我們想要的位置，此時我們可以藉由畫面的擷取，再經由人為的判斷，判斷它是否有送出成功。

Selenium有提供畫面擷取的API使用，我們可以寫一個擷取畫面的方法來進行畫面截圖的功能，如圖9所示。

```
private void ScreenShot(String fileName) {
    try {
        screenshot = ((TakesScreenshot)augmentedDriver).getScreenshotAs(OutputType.FILE);
        bi = ImageIO.read(screenshot);
        outputfile = new File("C://your location" + fileName +".png");
        ImageIO.write(bi, "png", outputfile);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

圖 9. 畫面擷取方法

3)TP3: Location Strategy: 在抓取網頁物件時，有非常多種方式可以使用，如：ID、Name、Link Text、CSS、XLocator等。 一般來說最常用的是ID和Name，但是當某些物件它並沒有標示ID或名稱時，我們就必須使用CSS或XLocator來抓取物件。

4)TP4: External System Integration: 在測試過程中，可能會遇到需要第三方軟體的支援來達到自動化測試的目的，例如需要大量假帳號來註冊，而註冊大量帳號需要非常多的電子郵件地址，我們就可以利用10分鐘信箱(10 Minutes Mail)[13]來取得大量電子郵件地址來註冊帳號。

C. 階段三：網頁物件程式碼重整

在為每個網頁物件加入測試樣式之後，雖然解決了自動化測試時常見的問題，但是我們尚未思考每一個網頁物件應該擁有哪些屬性和行為，每個網頁物件應該要包含：

- 屬性：網頁上擁有的物件；
- 行為：網頁上可執行的動作；
- 自我檢查機制：確保此網頁已經正確地進入。

每個網頁上會擁有自己的物件，像是登入後通常會看到自己的暱稱，在程式碼中有了屬性就應該要有其對應的方法來取得和設定此屬性，一般稱為Getter method和Setter method。

- Getter method：回傳物件的值(ex: 使用者名稱)，通常用來和我們的預期結果做比較，如圖10所示。
- Setter method：用來設定物件的值或狀態來幫助我們做測試，如圖11所示，也可讓我們的測試案例更有彈性。

網頁上也會有許多可執行的動作，例如：超連結、按鈕、下拉式選單、文字輸入…等，我們必須建立方法來代表這些行為，一般稱為Trigger methods。

- Trigger method：用來模擬使用者的行為，將測試流程帶到一個新的狀態，如圖12所示。

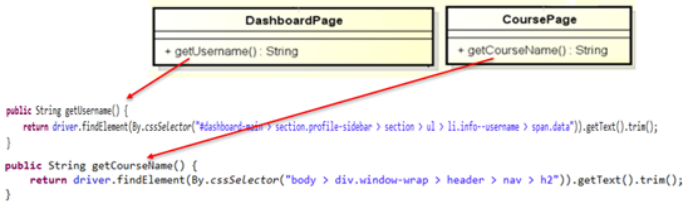


圖 10. Getter methods

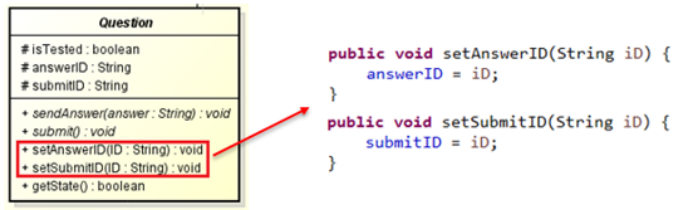


圖 11. Setter methods

把網頁物件的屬性和行為加入之後，會發現其中有許多相同的屬性或是類似的動作重複出現，於是我們想將物件導向的概念加入其中，對網頁物件進行抽象化的動作，並且應用若干個設計模式來加強它的可維護性與擴充性。

抽象化的過程如下：

- 尋找共同的屬性及動作
- 建立父類別將共同的屬性及動作抽象化
- 繼承父類別把剩餘的功能建立完成

如此一來就將重複的程式碼減少了許多，而且利用繼承的方式可達到多型的目的，程式的彈性也就跟著上升了。

V. 實驗方法

本研究提出一個可維護性高的系統化測試方法，我們挑選Open edX來進行實驗。

A. Open edX介紹

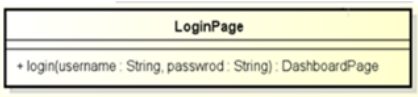
Open edX[14]為開源的磨課師(MOOCs)的線上學習平台，他有兩個主要的平台分別是教師使用的課程編輯平台(studio)與學生瀏覽課程的學習管理系統(Learning Management System)。

edX平台專案規模龐大如表II所示，主要由Python程式語言與JavaScript動態網頁語言撰寫而成。

edX平台功能繁多且複雜，一門課程底下有三層結構，分別為「章節」、「小節」以及「單元」，其中單元例面包含許多課程內容的物件，像是影片播放、討論區、習題等，習題又分為很多種類，像是單選題、多選題、數字與文字輸入、同儕互評、圖片點選題…等等。

B. 實驗方法

功能如此繁多的Open edX，需要一完整的測試過程，來確保其功能正常，保障服務的品質，我們撰寫兩種版本的測試案例：(1)單純使用網頁物件樣式；(2)使用本文提出的方法，藉此比較兩者之間的差異性。




```
public DashboardPage login(String username, String password) {
    driver.get("https://sandbox.openedu.tw/");

    //登入
    driver.findElement(By.linkText("登入")).click();
    //等待帳號輸入出現後，輸入帳號
    element = driver.findElement(By.id("email"));
    element.sendKeys(username);

    //等待密碼輸入出現後，輸入密碼
    element = driver.findElement(By.id("password"));
    element.sendKeys(password);

    //點擊登入按鈕
    driver.findElement(By.id("submit")).click();
    return new DashboardPage(driver);
}
```

圖 12. Trigger Method

表 II. Open edX平台專案規模

語言	程式碼行數	註解行數
JavaScript	285,073	59,368
Python	268,437	108,327
HTML	27,619	30,270
CSS	14,307	1,102
SQL	4,806	18
XML	3,464	32
總和	603,706	199,117

圖13為使用本文方法的測驗題測試案例架構圖，每個package中的類別都有使用到繼承的關係來提高程式碼的重用性，在課程網頁物件中(CoursewarePage)，因當中包含了許多測驗題，所以我們將測驗題獨立成一個抽象類別，而每一種測驗題都應該有自己測試的方法，所以在測驗題抽象類別(Question)中有一個抽象方法testQuestion()，在子類別中必須要定義屬於自己的測試方式。

圖14為一個使用網頁物件樣式的課程網頁物件程式碼，一門課程中會有要測試的章節(程式碼第14行)、小節(程式碼第18行)與影片(程式碼第22行)等等，也會有許多測驗題型(程式碼第4-8行)，我們可以從中發現有許多程式碼都是寫死的，這將不利於維護，圖15為使用本文方法經過程式碼重構之後的課程網頁物件程式碼，我們將trigger method改為參數化的方式帶入需要測試的章節、小節與影片，提高網頁物件的彈性，這些設定應在測試腳本做設定而不是寫在網頁物件當中，我們也把測驗題型進行抽象化的動作，讓程式碼看起來更簡潔，更易於閱讀。

圖16為上述課程網頁物件中使用網頁物件樣式測試測驗題的方法，我們仍然可以發現有許多寫死的程式碼，圖17為使用本文方法測試測驗題的方法，我們將測驗題型進行抽象化之後，不僅可以提高程式碼的重用性(若許多測試案例都會測試到相同題型)，也利用到了多型的好處，每一種題型測試的方式都不盡相同，他們各自的測試方法都寫在各自的testQuestion方法中(程式碼第11、23行)。

如果將來有新的測試案例或是測驗題產生，我們只

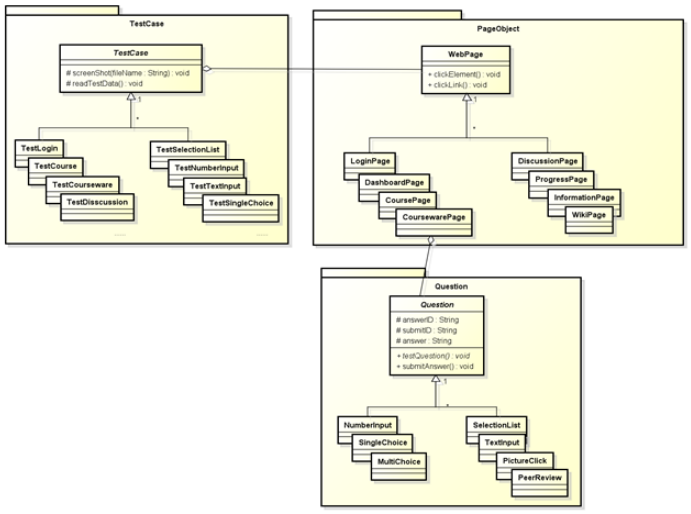


圖 13.　測驗題測試案例類別圖

```java
public class CoursewarePage {

    private WebDriver    driver;
    private MultiChoice  multiChoice;
    private SingleChoice singleChoice;
    private TextInput    textInput;
    private NumberInput  numberInput;
    private SelectionList selectionList;

    public CoursewarePage(WebDriver driver) {
        this.driver = driver;
    }

    public void selectChapter() {
        driver.findElement(By.linkText("Chapter 2")).click();
    }

    public void selectSection() {
        driver.findElement(By.linkText("Section 2.2")).click();
    }

    public void playVideo(String css) {
        driver.findElement(By.cssSelector("#video_i4x-Apple-CS2014-video-" +
            "5f96cda61c884551b1811a655b591bd0 > div.tc-wrapper > article > " +
            "section.video-controls > div:nth-child(2) > div.vcr > button > " +
            "span > span.icon.fa.fa-play")).click();
    }
    ...
    ...
}
```

圖 14.　課程網頁物件(使用網頁物件樣式)

需要繼承測試案例的類別或是測驗題的類別，在當中建立新的測試步驟和流程即可，以擴充代替修改，亦可提高測試案例的品質。

C. 實驗結果

我們利用網頁物件樣式與本研究的方法分別設計了10個測試項目相同的測試案例，然後額外再擴充10個測試案例，藉此比較本研究的方法是否可提高網頁物件樣式之可維護性，實驗結果如表III。

由於我們的方法著重在改進網頁物件上，所以從實驗結果可以看到在測試案例上的程式碼行數是相同的，而在擴充性上面使用本文的方法所擴充的程式碼行數是較少的，花費的時間也比較少，因為我們的方法有針對測驗題的部分進行抽象化的動作所以網頁物件的個數會比較多，但是在擴充的程式碼行數上卻是比較少的，擴充的部分僅僅在於測試案例上，網頁物件並不用去更改他，而單純的網頁物件樣式如同前面提到的，有許多地方是寫死得所以

```java
public class CoursewarePage extends WebPage {

    protected Question question;

    public CoursewarePage(WebDriver driver) {
        super(driver);
    }

    public void selectChapter(String name) {
        clickLink(name);
    }

    public void selectSection(String name) {
        clickLink(name);
    }

    public void playVideo(String css) {
        clickElementByCss(css);
    }
    ...
    ...
}
```

圖 15.　課程網頁物件(使用本文方法)

```java
public class CoursewarePage {

    private WebDriver    driver;

    public boolean testSelectionList(String answer) {

        Select select = new Select(driver.findElement(By.cssSelector("#input_" +
            "_i4x-Apple-CS2014-problem-47371a24899d4a1c9285a3902ef8a80a_2_1");
        select.selectByVisibleText(answer);
        driver.findElement(By.cssSelector("#problem_i4x-Apple-CS2014-problem" +
            "-47371a24899d4a1c9285a3902ef8a80a > div.problem > div.action >" +
            "button").click();

    }

    public boolean testNumberInput(String answer) {

        WebElement element = driver.findElement(By.cssSelector("#input_i4x-Apple" +
            "-CS2014-problem-2c6e89a1762d4e7ea6af7861fd5207fd_2_1");
        element.sendKeys(answer);
        driver.findElement(By.cssSelector("#problem_i4x-Apple-CS2014-problem-" +
            "2c6e89a1762d4e7ea6af7861fd5207fd > div.problem > div.action >" +
            "button").click();

    }
    ...
    ...
}
```

圖 16.　測試方法(使用網頁物件樣式)

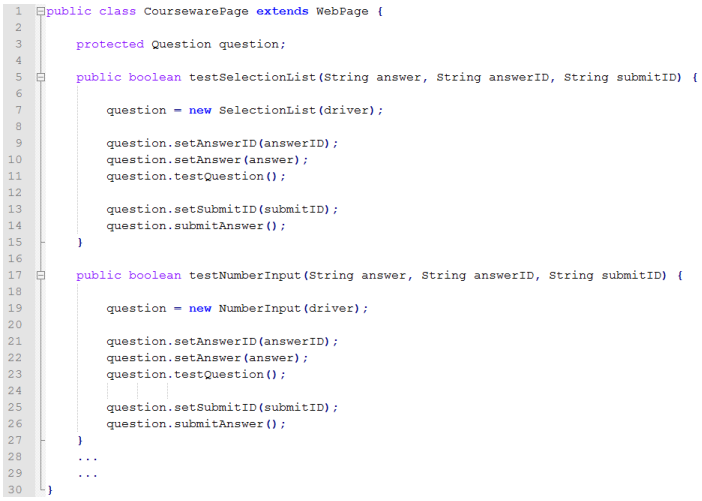
擴充的程式碼行數會來的比較多。

VI.　結論與未來研究

本研究提出一個系統化的網頁測試方法，應用了網頁物件樣式來增加測試案例的可維護性，並且加入了測試樣式與物件導向的概念解決它原有的問題，而從實驗結果來

表 III.　實驗結果

	網頁物件樣式	我們的方法
測試案例程式碼行數	695	695
網頁物件個數	5	13
擴充測試案例增加的程式碼行數	80	30
擴充測試案例花費時間(分鐘)	12	5



```
public class CoursewarePage extends WebPage {

    protected Question question;

    public boolean testSelectionList(String answer, String answerID, String submitID) {

        question = new SelectionList(driver);

        question.setAnswerID(answerID);
        question.setAnswer(answer);
        question.testQuestion();

        question.setSubmitID(submitID);
        question.submitAnswer();
    }

    public boolean testNumberInput(String answer, String answerID, String submitID) {

        question = new NumberInput(driver);

        question.setAnswerID(answerID);
        question.setAnswer(answer);
        question.testQuestion();

        question.setSubmitID(submitID);
        question.submitAnswer();
    }
    ...
    ...
}
```

圖 17.　測試方法(使用本文方法)

看本研究提出的方法確實可以縮短測試案例維護的成本。

未來我們將針對Open edX做一個更完整的實驗，藉由版本之間的更新來比較網頁物件樣式與本文提出的方法，檢驗是否能夠增加網頁物件樣式的可維護性。

致謝

本研究承蒙國家科學委員會研究計畫編號: MOST 103-2221-E-035-095 之經費補助得以順利完成，特此感謝。

References

[1] M. Leotta, D. Clerissi, F. Ricca, and C. Spadaro, ``Improving test suites maintainability with the page object pattern: An industrial case study,'' in Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on. IEEE, 2013, pp. 108--113.

[2] Selenium, ``Selenium - web browser automation.'' Available: http://www.seleniumhq.org/.

[3] JUnit, ``Junit - about.'' Available: http://junit.org/.

[4] M. Leotta, D. Clerissi, F. Ricca, and P. Tonella, ``Capture-replay vs. programmable web testing: An empirical assessment during test case evolution,'' in 2013 20th Working Conference on Reverse Engineering (WCRE). IEEE, 2013, pp. 272--281.

[5] ------, ``Visual vs. dom-based web locators: An empirical study,'' in Web Engineering. Springer, 2014, pp. 322--340.

[6] M. Leotta, A. Stocco, F. Ricca, and P. Tonella, ``Using multi-locators to increase the robustness of web test cases,'' in Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on. IEEE, 2015, pp. 1--10.

[7] A. Van Deursen, ``Testing web applications with state objects,'' Communications of the ACM, vol. 58, no. 8, pp. 36--43, 2015.

[8] M. Felderer and A. Herrmann, ``Manual test case derivation from uml activity diagrams and state machines: A controlled experiment,'' Information and Software Technology, vol. 61, pp. 1--15, 2015.

[9] S. Carino, J. H. Andrews, S. Goulding, P. Arunthavarajah, and J. Hertyk, ``Blackhorse: creating smart test cases from brittle recorded tests,'' Software Quality Journal, vol. 22, no. 2, pp. 293--310, 2014.

[10] E. F. Collins et al., ``Software test automation practices in agile development environment: An industry experience report,'' in Proceedings of the 7th International Workshop on Automation of Software Test. IEEE Press, 2012, pp. 57--63.

[11] V. Hurdugaci and A. Zaidman, ``Aiding software developers to maintain developer tests,'' in Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on. IEEE, 2012, pp. 11--20.

[12] K. Karhu, T. Repo, O. Taipale, and K. Smolander, ``Empirical observations on software testing automation,'' in Software Testing Verification and Validation, 2009. ICST'09. International Conference on. IEEE, 2009, pp. 201--209.

[13] ``10 minutes mail.'' Available: http://10minutemail.com/10MinuteMail/index.html.

[14] edX, ``edx - platform.'' Available: https://github.com/edx/edx-platform.