國立臺北科技大學

資訊工程系碩士班

碩士學位論文

Robot Framework 測試腳本重構工具的改善：增加重構方法之多元選擇

Further Improvements of a Robot Framework Test Scripts Refactoring Tool

研究生:吳俊青

指導教授：鄭有進教授、謝金雲教授

中華民國一百一十年六月

摘 要

論文名稱：Robot Framework 測試腳本重構工具的改善: 增加重構方法之多元選擇

頁數：四十四頁

校所別：國立台北科技大學 資訊工程系碩士班

畢業時間：一百零九學年度第二學期

學位：碩士

研究生：吳俊青

指導教授：謝金雲、鄭有進教授

關鍵字：Robot Framework、自動化驗收測試、重構關鍵字驅動測試腳本、Eclipse

針對本實驗室開發的一個 Robot Framework 測試腳本重構工具 RF Refactoring，本論文提出兩種功能延伸，除了保有原先所提供的三種重構功能，重新命名關鍵字、重新命名變數及修改關鍵字介面之外，新增了抽取重複步驟成為新關鍵字、移動關鍵字宣告的功能，其能夠使工具更加完備，讓開發人員進行相關重構時，不再只能利用搜尋取代的方式進行重構，且不需進行不必要的人工檢查，進而提升重構效率，而在重構方法的選擇上也更加多元。

ABSTRACT

Title:Further Improvements of a Robot Framework Test Scripts Refactoring Tool

Pages: 44

School: National Taipei University of Technology

Department: Computer Science and Information Engineering

Time: June,2021

Degree: Master

Researcher:JUN-QING WU

Advisor: C.-Y. Hsieh and Y.C.Cheng

Keywords: Robot Framework, Automated acceptance tests, Refactoring keyword-driven test script, Eclipse

Aiming at RF Refactoring, a Robot Framework test script refactoring tool developed by this laboratory, the thesis proposes two extensions of functions. In addition to retaining the three original refactoring functions provided, renaming keyword, renaming variable, and modifying the keyword interface, two new function have been added to extract duplicate steps to a new keywords, and to move keyword declaration. It can make the tool more complete, so that when developers carry out relevant refactoring, they can no longer only use search and replace to refactor, and there is no need to perform unnecessary manual inspections, thereby improving the efficiency of refactoring, and in the refactoring method the choices are also more diverse.

誌 謝

致謝.....

目錄

表目錄

圖目錄

程式碼目錄

第一章　緒論

1.1　研究背景與動機

Robot Framework [2] 是一個自動化測試框架，其中的關鍵字可被視為一個測試步驟，透過將多個關鍵字包裹成一個更高層級的關鍵字時，便可將其視為一個測試流程。

當多個團隊一起開發測試腳本時，關鍵字能夠盡量被重複使用是一個常見的目標。而部份情況中我們可以在撰寫測試腳本時就判斷出撰寫的測試步驟是可以被重複使用，因而將其提前包裹成一個新關鍵字，例如當我們在同一個測試腳本中，有多個步驟需要重複使用時，我們便會提前將其抽取成一個關鍵字提供多處使用，這是可以預知的；但大部分的情況中，我們無法預期目前所撰寫的測試步驟是否會被其他測試腳本再次使用，以兩個測試團隊為例：其中一個測試團隊需要以創立一個項目的流程作為測試腳本中的主要步驟，另外一個測試團隊也需要相同的流程做為測試資料的準備，平時撰寫測試腳本時，兩個組別無法隨時互相溝通，只能以現有的情況做為判斷，因此無法透過提前判斷而去抽取成一個關鍵字，或者其中一個團隊剛好發現其他團隊已有撰寫好之流程，直接拿取做使用且當下未立刻進行抽取關鍵字之重構，導致後續時常需要對現有的程式碼進行重構。

進行重構時，重複的測試步驟往往都是存在於不同的檔案中，一不小心就會有部分程式碼未修改，直到後續執行時才發現錯誤，這些經常都是人為錯漏所導致的。為了避免以上所提及的錯誤不斷發生，導致測試團隊必須再次花費時間針對缺漏的錯誤進行修正，因此在測試團隊重構時，需要一個能夠避免人為錯漏且更加方便的重構工具。

1.2 研究目標

劉冠志論文 [3] 中提供了測試腳本重構工具 RF Refactoring，此工具提供了三種 Robot Framework 測試腳本的重構方法，分別為重新命名關鍵字、重新命名變數，以及修改關鍵字介面，但仍然無法解決目前所遭遇之問題，因此本論文將針對此工具進行改善，否則開發人員只能使用目前用來撰寫 Robot Framework 測試腳本的整合式開發環境 (RED [4]、Visual Studio Code [5]) 中的搜尋取代工具進行重構，因而造成重構效率的降低，以及人工檢查的疏忽，導致重構的錯誤和缺漏等等。為此工具增加重構方法之多元選擇，使其可根據測試人員的需求重構測試腳本，並且在重構過程中能夠避免人為錯漏的問題發生。

1.3 論文組織架構

本論文共有六章節，第二章將介紹背景知識及使用工具，第三章介紹擴充重構工具之方法設計，其中介紹了欲擴充之重構方法及其流程，並且介紹如何將重構方法擴充至重構工具中，第四章將會以第三章所設計之方法進行實作，其中包含了重構流程及外掛程式之實作，第五章將以實際 Robot Framework 測試腳本介紹重構案例，並且邀請測試團隊成員分別使用 Visual Studio Code 搜尋取代工具及本論文擴充後之重構工具進行重構，後續比較使用之差異。

第二章　背景知識

2.1 Refactoring

Refactoring [6] 通常被定義為「在不改變程式碼外在行為的前提下，改善既有程式碼內部架構的過程」，而重構測試程式碼與重構一般程式碼也有所不同，在重構一般程式碼時，會著重在內部架構的設計，讓其未來被使用及維護上都能夠更加地容易；而重構測試程式碼時，則是根據每個測試當下的狀況，判斷其設計是否符合當時所需來進行調整，或是改變測試的順序讓整體運作上能夠更加地順暢。儘管兩者有不同之處，但其最終目的皆是希望讓整體程式碼的設計更加完善，進而提升程式碼的品質。

2.2 Robot Framework

Robot Framework 是一個基於 Python [7] 所開源的自動化框架，通常被使用於開發驗收測試及驗收測試驅動開發 (Acceptance test-driven development) [8]、行為驅動開發 (Behavior-driven development) [9]。由於關鍵字驅動測試 (Keyword-driven testing) [2] 的特色，開發人員能夠使用簡單且較容易理解的關鍵字撰寫測試腳本，讓整體具備極高的可讀性及降低開發的門檻，並且開發人員能夠將多個關鍵字包裝成更高階的關鍵字，使測試腳本閱讀起來更加地接近自然語言。

Robot Framework 同時也擁有極佳的可擴充性，除了能夠使用官方及外部函式庫所提供的關鍵字以外，當開發人員遭遇現有關鍵字無法滿足開發需求時，也能夠透過 Python 或 Java [10] 自行擴充所需的函式庫，滿足新關鍵字的需求。此外 Robot

Framework 提供了公開的 API(Application Programming Interface) [11]，例如：針對現有的測試腳本進行解析、調整測試腳本執行的結果等等，讓開發人員能夠依照需求自行製作其他的功能，使設計測試腳本的過程能夠更加彈性。程式碼2.1為 Robot Framework 測試腳本之實例

程式碼 2.1　Robot Framework 實例腳本

```
*** Settings ***
Library     SeleniumLibrary
Resource    ../microsoft.txt

Test Setup    Go To English Microsoft

*** Variables ***
@{welcomeTaipei} =    Welcome    To    Taipei

*** Test Cases ***
Go To "Windows Security" Page And Log Welcome Text
    Go To Windows Page
    Open Windows 10 Menu
    Go To "Windows Security" Page
    "Windows Security" Page Should Be Visble
    FOR    ${var}    IN    @{welcomeTaipei}
        Log Double Text    ${var}
    END
    [Teardown]    Close Browser

*** Keywords ***
Go To English Microsoft
    Go To Microsoft
    Open Language Option
    Select English Language
```

2.3　抽象語法樹

抽象語法樹 (Abstract Syntax Trees) [12] 是程式碼結構的一種抽象表達，利用樹狀結構展現出程式中的語法關係，每個節點分別表示程式碼中的一種結構，但不會將語法中的細節完全呈現，例如：Java 中利用括號所表示的領域區分，在節點中並不會被呈現、條件式判斷則以節點下的分支進行表示，並非表示在節點之中。

在 Python 中，開發人員能夠透過 AST 套件 [13] 開發與抽象語法樹相關的程式碼，

對於當前程式碼在語法層面的操作及檢視都提供了十分大的幫助。套件中提供了 parse 相關的輔助函數，可以讓開發人員更簡單地取得所需的語法樹，此外也提供了方便開發人員訪問各節點的可繼承類別，例如：NodeVisitor、NodeTransformer，讓開發人員能夠依照需求，自行新增含有不同功能的類別，運用上擁有較大的彈性。當其運用在 Robot Framework 上時，其可搭配 Robot Framework 提供的公開 API，將測試檔案解析成抽象語法樹 (AST) 模型，讓開發人員在對於測試腳本及測試資源的存取上都更加地簡單。

2.4 Eclipse

團隊開發測試腳本所使用之整合式開發環境 (RED) 是基於 Eclipse 所開發出來的，而 Eclipse [14] 是一個跨平台的開源整合式開發環境，起初是以開發 Java 作為主要目的，後續透過外掛程式的擴展，可使其成為其他語言的開發工具，例如：C++、PHP、Python 等等。Eclipse 僅以圖形 API、Java 開發環境、外掛程式開發環境等作為基礎核心，其餘功能則一律以外掛程式的方式附加於核心之上，例如：圖形編輯器、複雜重構、版本控制等等功能。

由於 Eclipse 除了核心以外，其餘皆為外掛程式，使其具有良好的擴充性，更重要的是，平台定義了十分明確的機制，讓不同的外掛程式可以利用延伸點 (extension points) 及貢獻 (contribute) 相互合作，既存的功能能夠被新功能使用，而新功能也可以輕鬆地整合於平台之中。圖2.1為 Eclipse 平台之架構 [1]

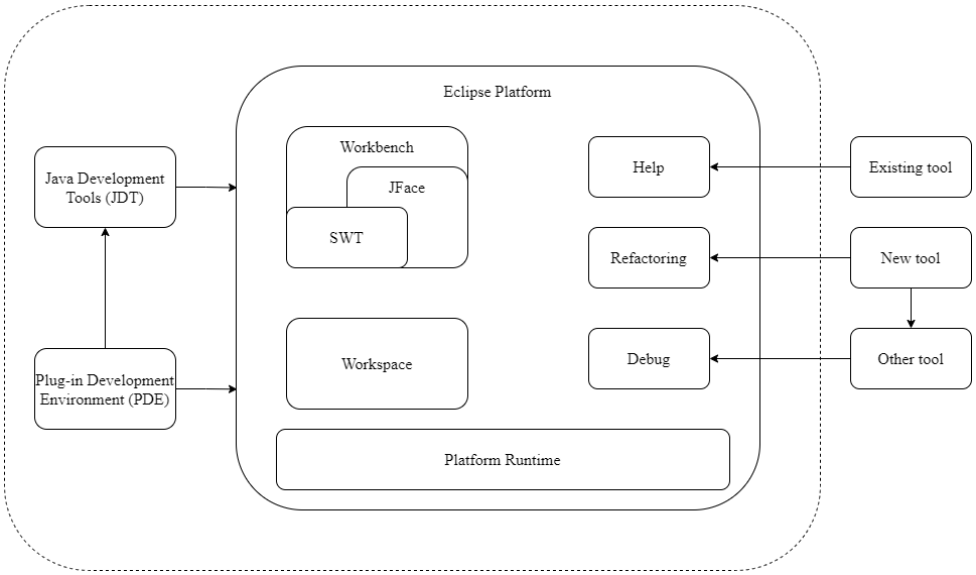

圖 2.1　Eclipse 外掛式架構 [1]

2.5　Jython

Jython [15] 是一個利用 Java 所撰寫的 Python 直譯器，在任何支援 JVM [16](Java virtual machine) 的環境上皆可使用，也代表著能夠使用 JVM 中的 Java 函式庫。Jython 提供開發人員在開發過程中能夠自由地混用 Java 及 Python 兩種不同的語言，使開發能夠更加地彈性。

程式碼 2.2　Jython 於 Python 語言環境中執行 Java 之實例

```
from java.lang import System

current_java_version = System.getProperty('java.version')
strInPython = "Current java version is " + current_java_version
System.out.println(strInPython)
```

程式碼 2.3　Jython 於 Java 語言環境中執行 Python 之實例

```
import org.python.util.PythonInterPreter;

public class RunPythonInJava{
        public static void main(String []args){
                PythonInterPreter pyRunner = new PythonInterPreter();
                pyRunner.exec("print('I am from python world!')");}}
```

第三章　RF Refactoring 延伸功能之設計

本論文為增加 RF Refactoring 的功能，以及避免手動重構測試腳本及測試資源時，所造成的搜尋缺漏和取代錯誤，因而提出此設計。透過實作新的重構功能，將其與 Eclipse 中既有的外掛程式結合，團隊使用此工具選擇所需的重構方法進行重構時，其將利用 Eclipse 中的視圖 (Views) 及視窗 (Dialogs) 提供使用者做所需的選擇及輸入，進而完成重構的流程。

3.1　抽取重複步驟成為新關鍵字之流程

在抽取重複步驟之前，必須提前解析測試專案下的所有測試檔案，後續則抽取指定的重複步驟成為一個新關鍵字，並搜尋所有重複步驟提供使用者以新關鍵字取代，最後協助其中未引入所需測試資源的測試腳本進行引入。圖3.1為此重構的全部活動，以下將分別介紹各項活動。

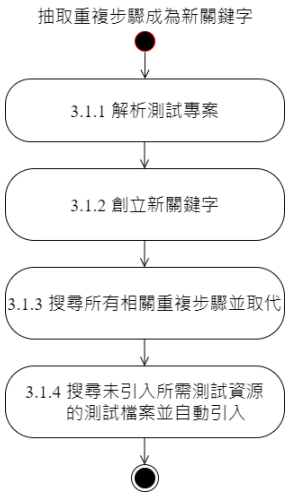

圖 3.1　抽取重複步驟成為新關鍵字之活動圖

3.1.1 解析測試專案

在抽取重複步驟成為新關鍵字時，需要在不同的測試檔案搜尋重複步驟，因此需要先利用2.2節所提到的公開 API 解析測試檔案，並將其儲存於記憶體中，使其在進行重構時，能夠方便且快速地被使用。圖3.2為解析測試專案的詳細活動，專案中的每個測試檔案解析成 AST 模型後，於其他重構活動上即可利用2.3節所提及的抽象語法樹應用，根據不同的需求來完成測試檔案的重構。

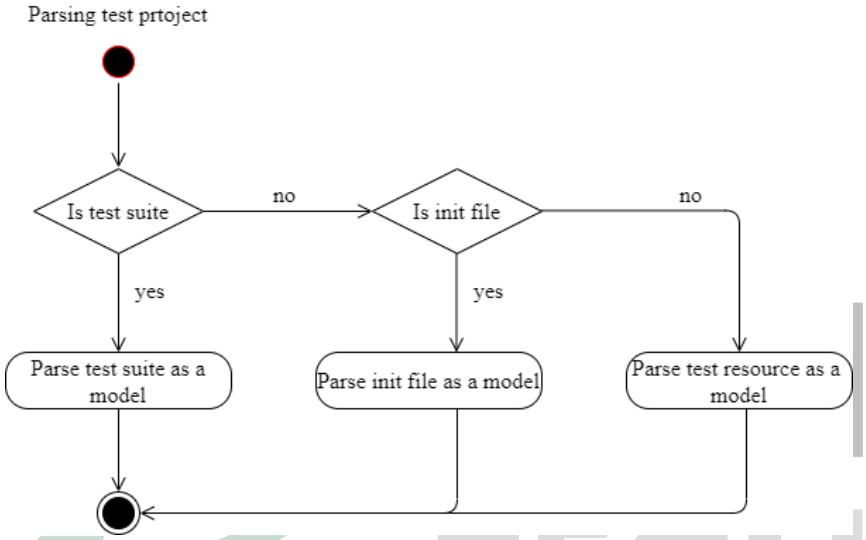

圖 3.2　解析測試專案之活動圖

3.1.2 創立新關鍵字

透過3.1.1節所得到的 AST 模型，可從其中指定測試檔案中的步驟範圍，並且將其做為新關鍵字的測試步驟，以此創立新關鍵字。在決定所要抽取的測試步驟範圍時，必須為其檢查是否有使用未宣告在被抽取步驟的變數，一旦有此類型之變數，則需將其做為新關鍵字之參數，最後在指定的檔案中創立新關鍵字。圖3.3為創立新關鍵字的詳細活動。

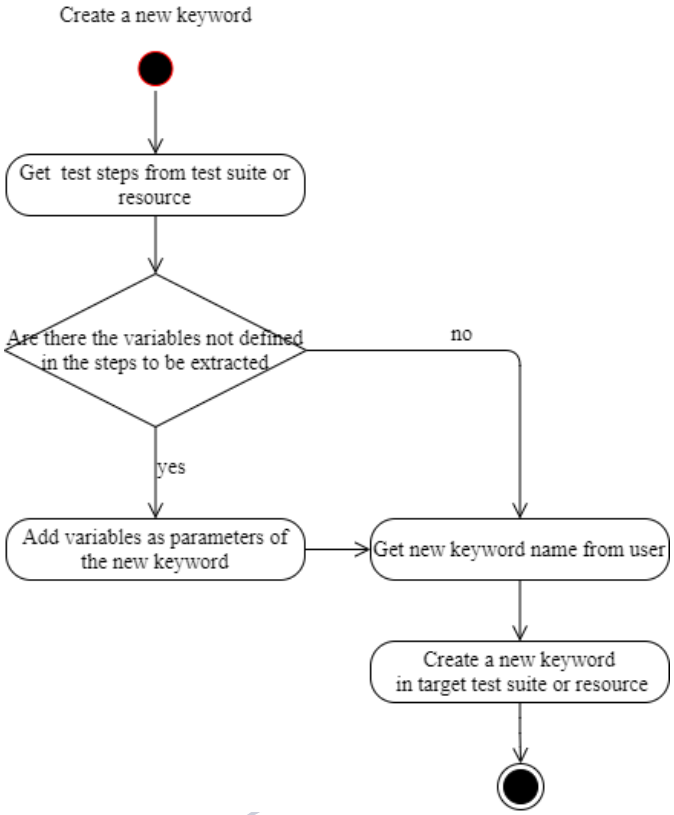

圖 3.3　創立新關鍵字之活動圖

3.1.3　搜尋所有相關重複步驟並取代

決定新關鍵字中的測試步驟時，將從3.1.1節取得的 AST 模型搜尋重複步驟。首先確認檔案是否為測試套件，如是則檢查其測試案例內及關鍵字宣告內的測試步驟順序是否相同、使用的關鍵字是否命名相同、參數數量是否相同，三種條件皆符合才加入搜尋結果；反之則認定為測試資源，檢查其關鍵字宣告內的測試步驟是否符合上述條件，並且加入搜尋結果。此外，一旦發現符合條件的測試步驟位於不同的關鍵字或測試案例中時，需將其判別為非重複步驟，進而得出重複步驟的清單。

取得重複步驟清單後，必須提供選取需被取代的重複步驟之功能，藉此使用者可依照不同測試腳本的狀況進行挑選，使此重構能夠更加準確。圖3.4為搜尋重複步驟並取代的詳細活動。

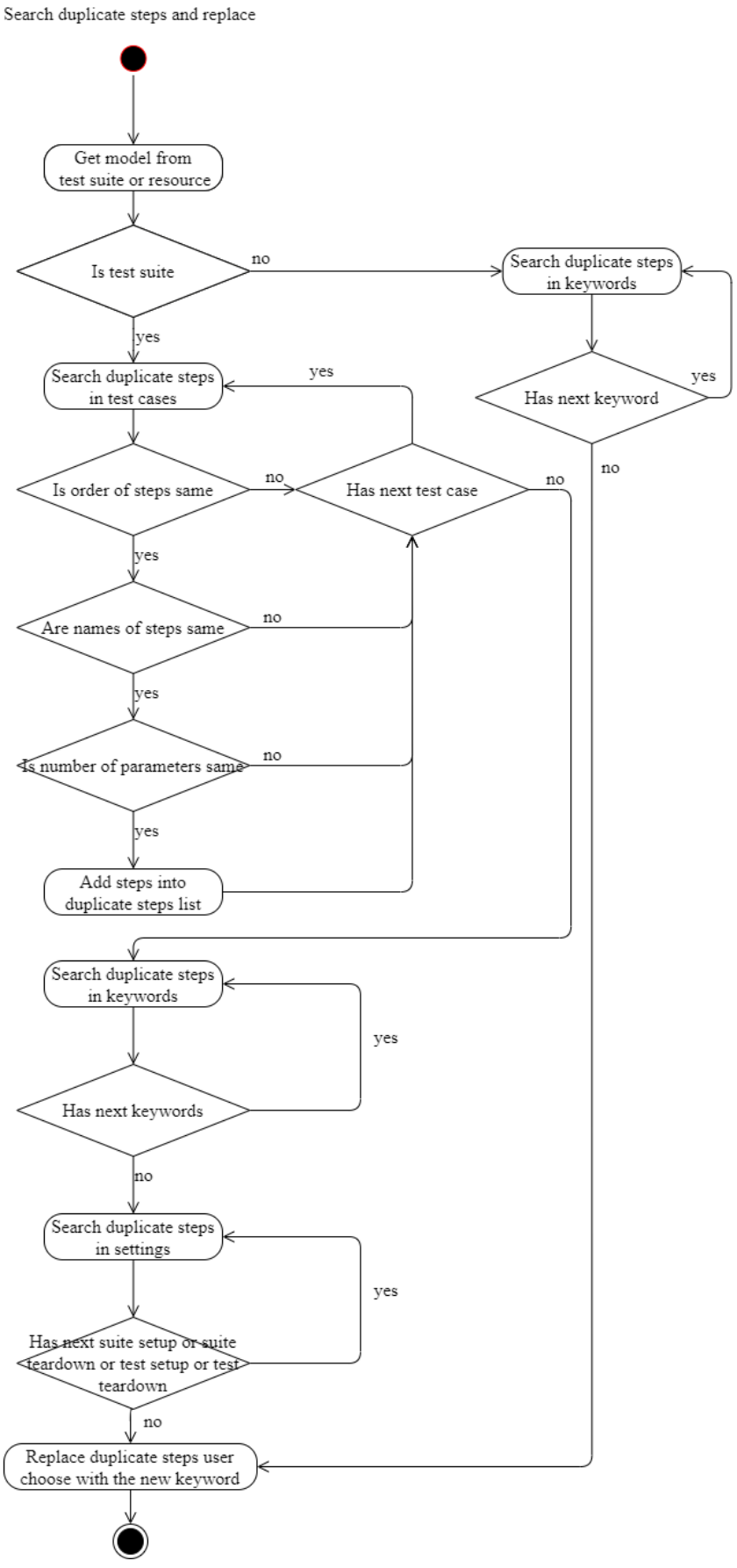

圖 3.4　搜尋重複步驟並取代之活動圖

3.1.4 搜尋未引入所需測試資源的測試檔案並自動引入

在3.1.3節進行重複步驟的取代時，將會保留有使用新關鍵字的 AST 模型，透過留存的 AST 模型不必再次從專案全部檔案中進行搜尋，即可檢測是否都有確實引入新關鍵字所需的測試資源，能有效地提升重構效率。根據新關鍵字宣告所在檔案的位置及需要引入測試資源的檔案位置，便能為其引入所需測試資源的相對位置。圖3.5為搜尋未引入所需測試資源的測試檔案並自動引入的詳細活動。

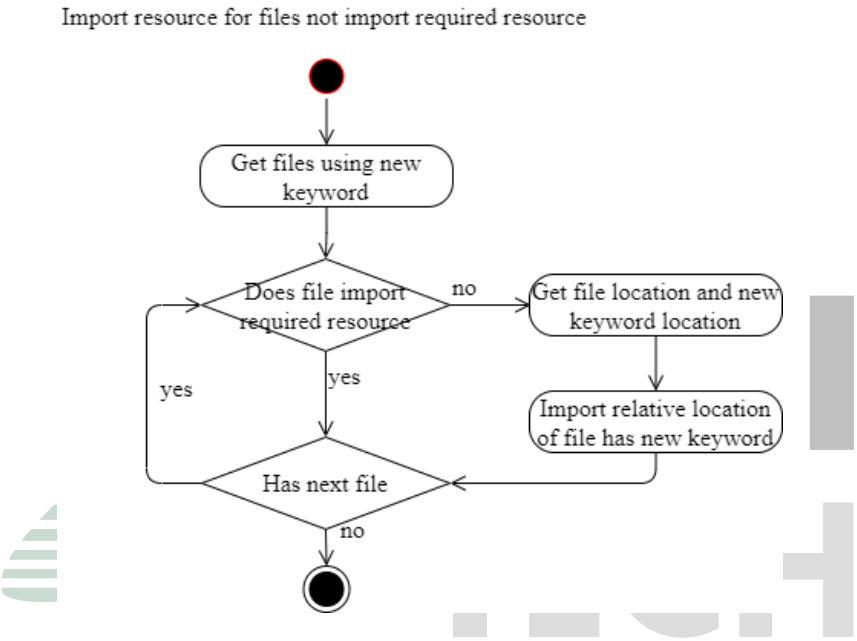

圖 3.5　搜尋未引入所需測試資源的測試檔案並自動引入之活動圖

3.2 移動關鍵字宣告之流程

在移動關鍵字宣告之前，必須與3.1節相同，提前解析測試專案下的所有測試檔案。移動關鍵字宣告至指定的測試資源後，必須檢查有使用被移動關鍵字的測試檔案，是否有引入所需之測試資源，最後進行引入之動作。圖3.6為移動關鍵字宣告之全部活動，以下將分別介紹各項活動。

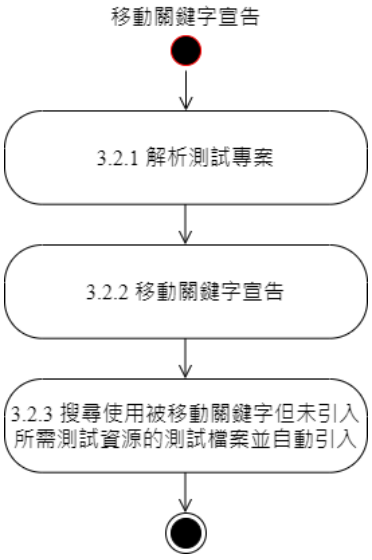

圖 3.6　移動關鍵字宣告之活動圖

3.2.1　解析測試專案

此活動將與3.1.1節介紹相同，於移動關鍵字宣告之前解析測試專案中的測試檔案，並將其解析完成的 AST 模型儲存於記憶體，後續即可用來完成重構需求。

3.2.2　移動關鍵字宣告

在解析完測試專案後，指定所要移動的關鍵字宣告及新的測試資源位置，並將原先的關鍵字宣告進行移除，且將被移除的關鍵字宣告新增至新的測試資源中，以此完成關鍵字宣告之移動。圖3.7為移動關鍵字宣告的詳細活動。

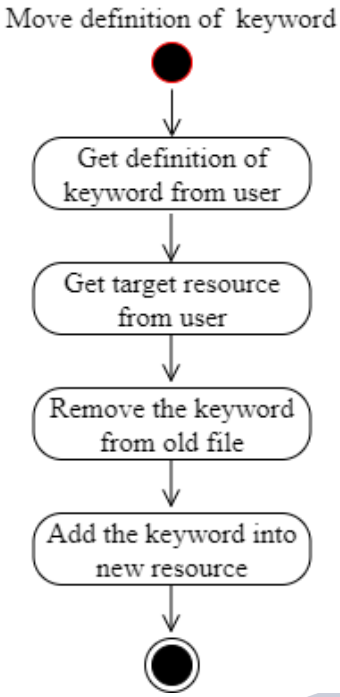

圖 3.7　移動關鍵字宣告之活動圖

3.2.3　搜尋使用被移動關鍵字但未引入所需測試資源的測試檔案並自動引入

移動關鍵字宣告後，首先要搜尋有引入原先關鍵字宣告所在之檔案位置，且使用與關鍵字宣告同名關鍵字的測試檔案，藉此取得原先已使用被移動關鍵字的測試檔案，後續檢查其是否有引入被使用關鍵字所需的新測試資源，如果未引入所需測試資源則自動為其引入與新測試資源之相對路徑。圖3.8為搜尋使用被移動關鍵字但未引入所需測試資源的測試檔案並自動引入的詳細活動。

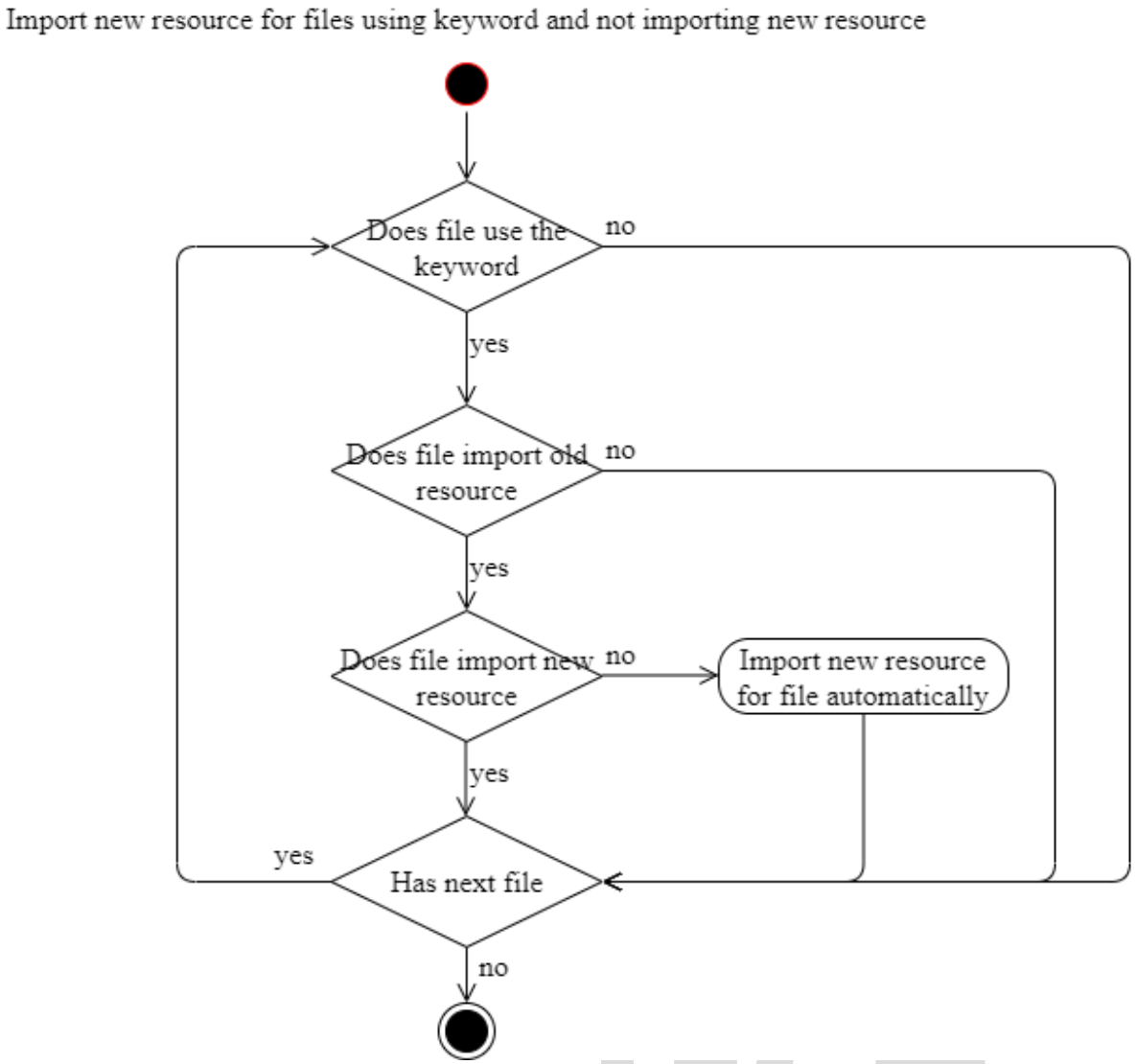

圖 3.8　搜尋使用被移動關鍵字但未引入所需測試資源的測試檔案並自動引入之活動圖

3.3　將新重構功能結合至 Eclipse 既有之外掛程式

RF Refactoring 是 Eclipse 中的外掛程式，因此本論文將參考劉冠志論文 [3] 中所介紹之 Eclipse 外掛程式開發方法，將新重構功能結合至 RF Refactoring 中。

第四章　RF Refactoring 延伸功能之實作

本章將對第三章所敘述之設計進行實作，其將針對 RF Refactoring 的兩個子系統進行擴充，分別為重構功能子系統及外掛程式子系統，後續小節將針對擴充之系統架構與子系統進行介紹。

4.1　系統架構

圖4.1為研究方法實作的系統架構圖，粗框元件代表擴充後的子系統，下列為各元件之介紹：

- **Eclipse**：除基礎核心外皆為外掛程式之開源整合式開發環境
- **Expanded Refactoring Plugin**：擴充後提供使用者進行重構之 Eclipse 外掛程式
- **Expanded Refactoring Function**：以 Python 實作之擴充後外掛程式
- **Robot Framework Parsing API**： Robot Framework 提供解析測試檔案的應用程式介面
- **Test Script**：測試腳本及測試資源

黑粗框元件為本論文主要之元件，其中 Expanded Refactoring Plugin 於 Eclipse 上針對 Robot Framework 測試檔案提供使用者更加多元的重構功能。使用者利用 Expanded Refactoring Plugin 中的使用者介面進行重構時，其會透過 Jython 函式庫呼叫 Expanded Refactoring Function 進行重構。重構功能將新增兩種，分別為抽取重複步驟成為新關鍵字及移動關鍵字宣告，而其流程將如3.1節及3.2節中所敘述，透過自動搜尋所需的資訊

後，於 Eclipse 上的視圖及視窗提供使用者進行選取，並自動確保關鍵字所需測試資源都有確實被引入，不會導致未知關鍵字之錯誤。

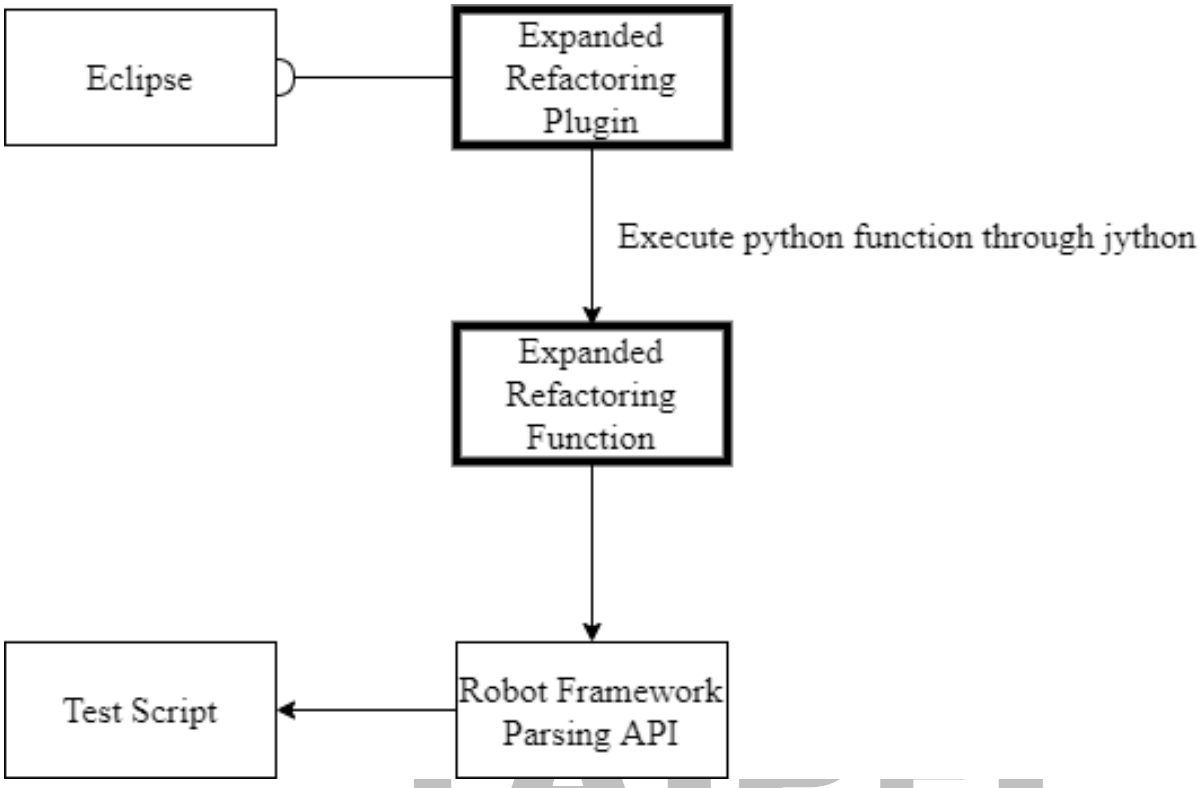

圖 4.1　研究方法實作之系統架構圖

4.2 重構功能之擴充

圖4.2為擴充的重構功能子系統類別圖，下列為圖中各類別之介紹：

- **NodeVisitor**：位於 AST 套件中的類別，其提供針對抽象語法樹的拜訪，並且提供部分方法可被覆寫，可依照需求做調整。
- **NodeTransformer**：位於 AST 套件中的類別，與 NodeVisitor 很像，但額外提供了對於抽象語法樹的修改，可依照需求修改語法樹的內容。
- **TestModelBuilder**：用於解析測試專案中的測試檔案，且將解析後的 AST 模型提供給其他類別使用。
- **LineKeywordsHelper**：針對使用者選取出的測試步驟，進行資料處理，例如：取得未定義於測試步驟中的變數、將測試步驟字串化以提供顯示等等。
- **KeywordCreator**：繼承 NodeTransformer 類別，主要用於建立新關鍵字中的測試步驟、創立一個新關鍵字及以新關鍵字取代重複步驟。
- **KeywordMoveHelper**：繼承 NodeTransformer 類別，主要用於移動關鍵字宣告、移動關鍵字所需測試資源。
- **FileChecker**：繼承 NodeVisitor 類別，主要用於搜尋未引入關鍵字所需測試資源的測試檔案、搜尋使用特定關鍵字宣告的測試檔案及搜尋含有重複步驟的測試檔案等等。
- **KeywordFinder**：繼承 NodeVisitor 類別，主要用於搜尋關鍵字宣告、搜尋被使用的關鍵字。
- **NewRefactoringFacade**：負責提供新重構功能給 Eclipse 外掛程式使用的類別，其提供解析測試專案、創立新關鍵字、搜尋所有重複步驟、移動關鍵字宣告、搜尋未引入所需測試資源的測試檔案等等功能。

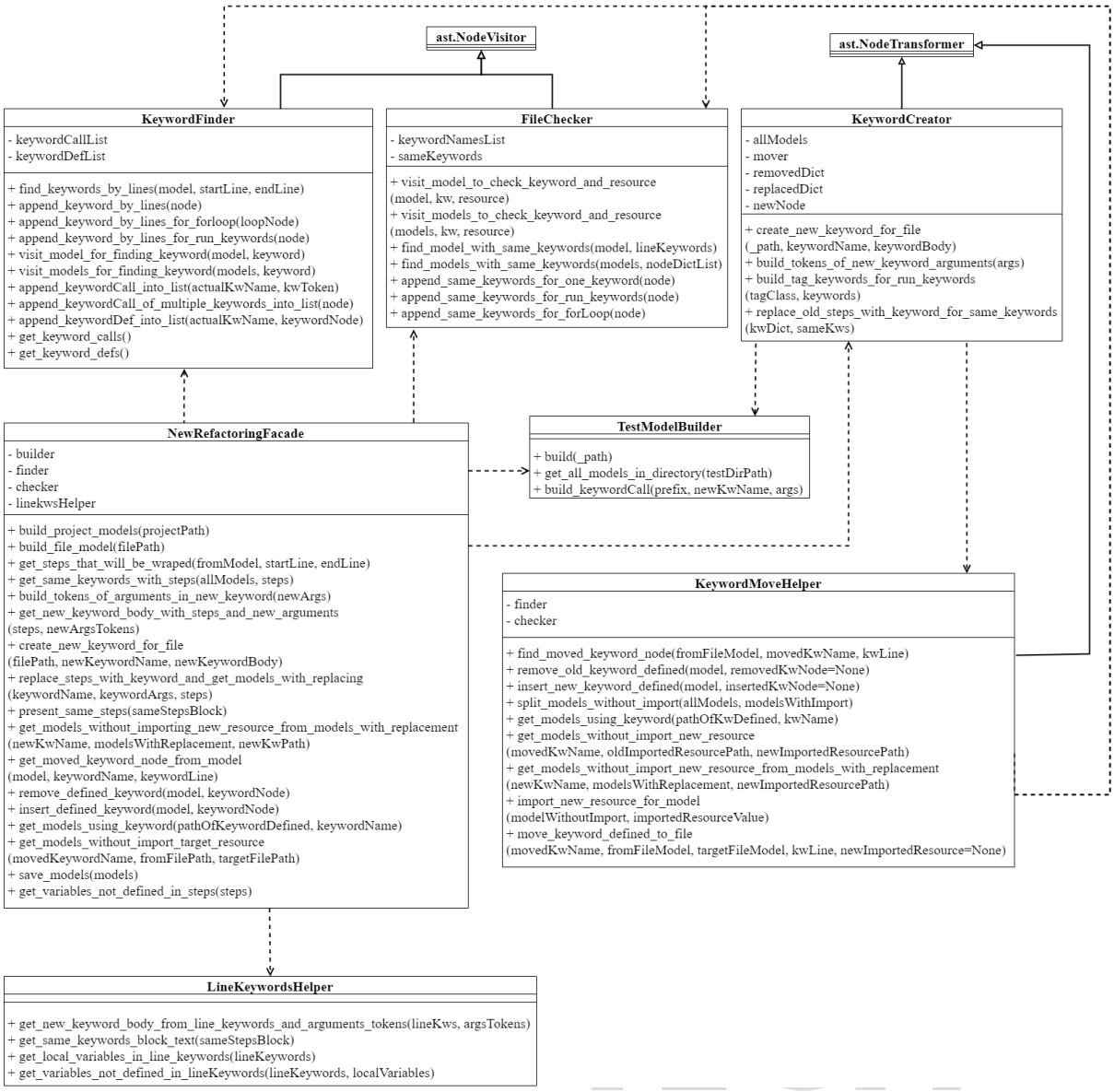

圖 4.2　重構功能擴充之類別圖

重構功能子系統將依3.2節及3.3節所敘述之流程，擴充其中之功能，下列小節將針對各功能之實作進行說明。

4.3　預先解析測試專案之實作

在 Robot Framework 中，測試檔案分為測試套件及測試資源兩種，本論文利用 Robot Framework Parsing API 套件之函式解析測試套件及測試資源，其參數都是檔

案路徑。圖4.3為預先解析測試專案之循序圖，取得測試專案的路徑後，將其傳入 TestModelBuilder 類別之函式，其將會透過遞迴方式搜尋是否有測試套件及測試資源，如搜尋到的檔案為測試套件，則利用 api 套件中的 get model 取得其 AST 模型，反之則利用 get resource model，進而解析出專案下全部的測試套件及測試資源，提供其他實作功能使用。

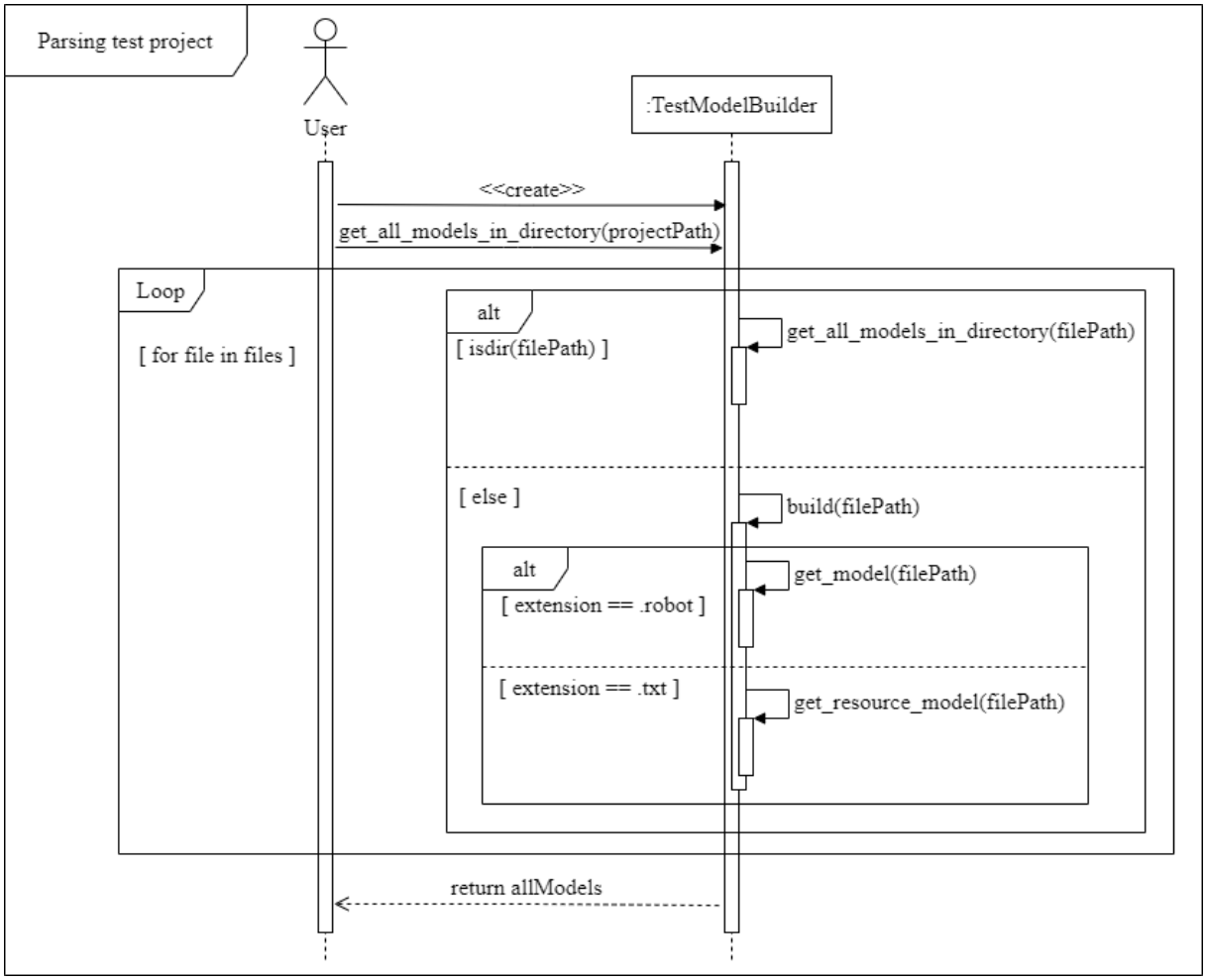

圖 4.3　預先解析測試專案之循序圖

4.4　抽取重複步驟成為新關鍵字之實作

本論文將依照3.1節所敘述之流程進行實作，透過 KeywordFinder、FileChecker 及 KeywordCreator 等類別完成下列各功能，進而達到抽取重複步驟成為新關鍵字之目的。

4.4.1 創立新關鍵字

如3.1.2節所敘述，本論文將依照所決定之步驟作為新關鍵字中的測試步驟，進而創立新關鍵字，圖4.4為此功能之循序圖，根據 AST 模型及步驟所在的行數區間呼叫 KeywordFinder 類別中的函式，即可取得其指定的測試步驟。透過 LineKeywordsHelper 類別之函式，在測試步驟中搜尋是否有變數之宣告，如有則將其視為區域變數，其餘皆認定為未宣告之變數。當未宣告變數數量大於零時，將透過 KeywordCreator 類別將其作為新關鍵字之參數，後續將依照指定的檔案路徑、新關鍵字名稱及經過處理的新關鍵字內容，完成新關鍵字之創立並將被修改的 AST 模型透過儲存函式回存至檔案系統中。

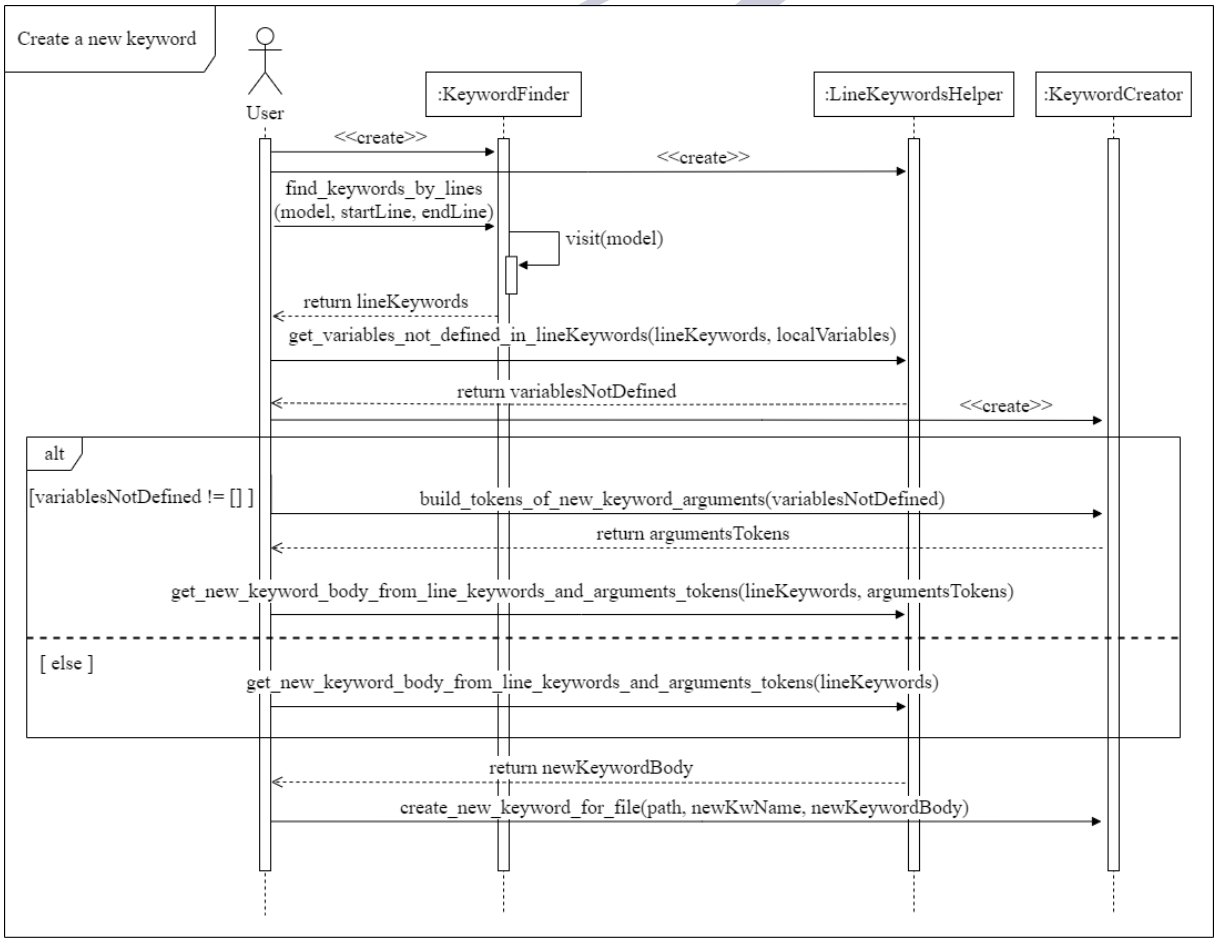

圖 4.4　創立新關鍵字之循序圖

4.4.2 搜尋所有相關重複步驟並取代

根據3.1.3節之流程，將4.4.1節所決定之測試步驟作為重複步驟判定之依據，得出專案中含有重複步驟之測試檔案，最後將選擇的重複步驟使用新關鍵字進行取代。圖4.5為搜尋所有相關重複步驟並取代之循序圖，FileChecker 類別將拜訪所有 AST 模型，藉由判定多個步驟與依據步驟的排序、參數數量和關鍵字名稱是否相同，確認其是否為重複步驟，如是則存入搜尋結果中，最後得到重複步驟清單。重複步驟清單經過選擇後，將會透過 KeywordCreator 類別進行取代，首先判斷步驟是否為重複步驟之最後一步，若不是，將會從 AST 模型中移除；反之則以新關鍵字取代，每組重複步驟皆進行上述處理，最後完成該功能。

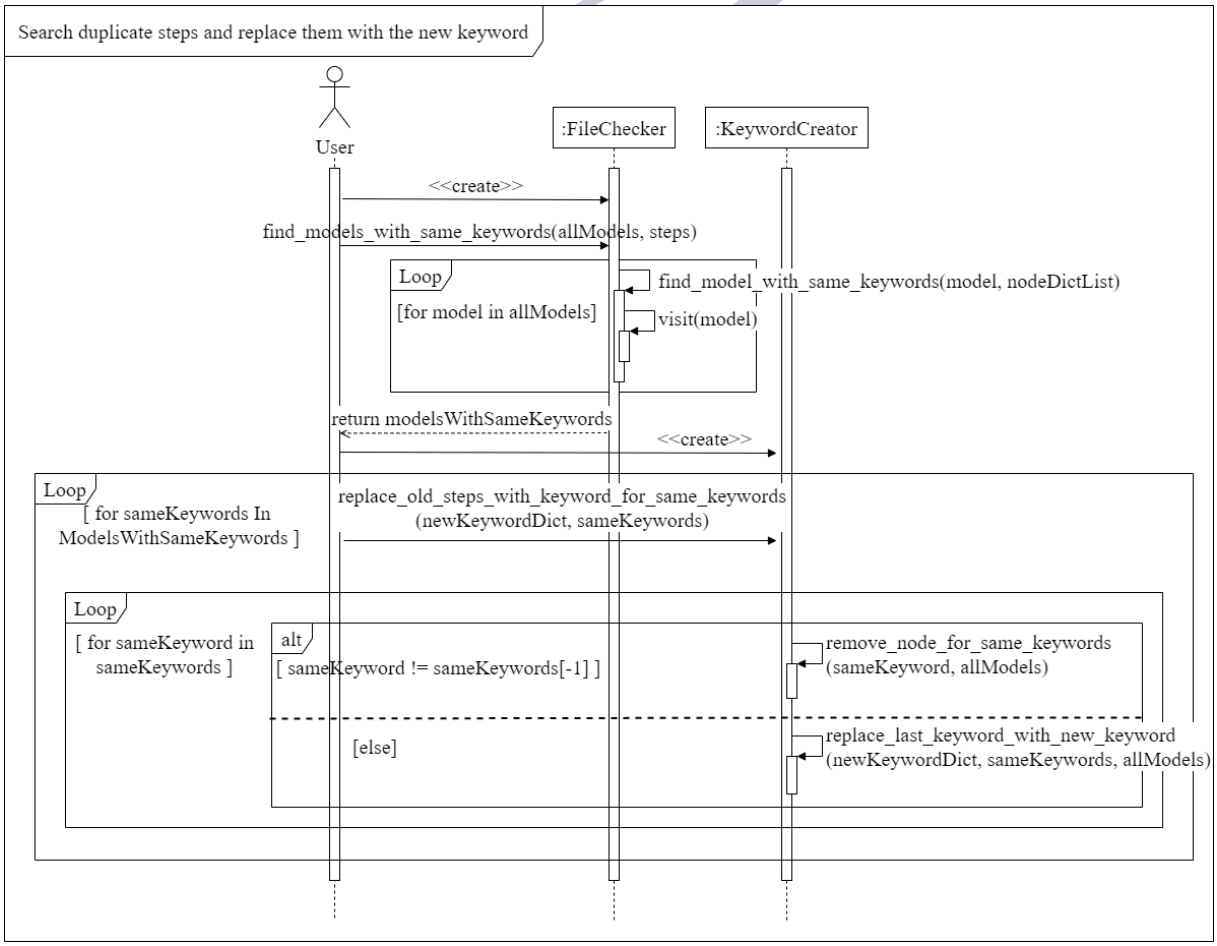

圖 4.5　搜尋所有相關重複步驟並取代之循序圖

4.4.3 搜尋未引入所需測試資源的測試檔案並自動引入

依據3.1.4之流程，保留4.4.2節使用新關鍵字之 AST 模型，且修正其中未引入新關鍵字所需測試資源的測試檔案。圖4.6為搜尋未引入所需測試資源的測試檔案並自動引入之循序圖，KeywordMoveHelper 類別之函式將透過 FileChecker 類別，比對每個 AST 模型所引入的測試資源是否含有新關鍵字所需測試資源，最後與所有 AST 模型比對得出需要修正之測試檔案。需要修正的測試檔案，將透過 KeywordMoveHelper 類別之函式，為其引入所需的測試資源，最後更新修正後的 AST 模型至全部 AST 模型中，確保後續進行其他重構時，檔案內容皆為正確。

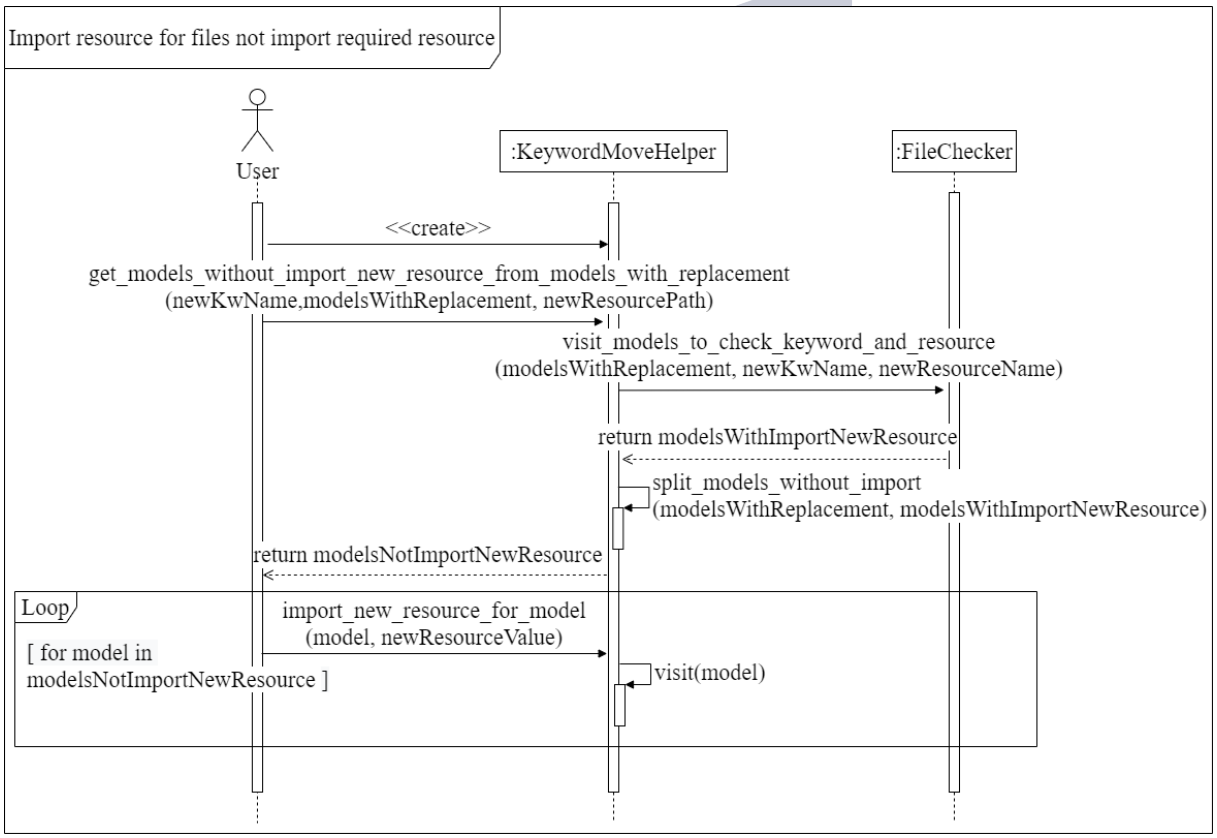

圖 4.6　搜尋未引入所需測試資源的測試檔案並自動引入之循序圖

4.5 移動關鍵字宣告之實作

本論文將依照3.2節所敘述之流程進行實作，透過 KeywordFinder、FileChecker 及 KeywordMoveHelper 等類別完成下列各功能，進而達到移動關鍵字宣告之目的。

4.5.1 移動關鍵字宣告

藉由3.2.2節之流程，將會把需被移動之關鍵字宣告從其所屬檔案中移除，並將其完整複製至指定檔案的關鍵字列表中。圖4.7為移動關鍵字宣告之循序圖，透過 KeywordFinder 類別之函式，比對關鍵字宣告名稱是否與所要移動的關鍵字名稱符合，如是則將其加入搜尋結果，最後得出關鍵字宣告清單。在關鍵字宣告清單中，如果數量只有一個則將其移動；反之則檔案中可能已存在關鍵字重複宣告之錯誤或此檔案中並無此關鍵字宣告，因此必須中止此重構。最後利用 KeywordMoveHelper 類別之函式，將欲移動之關鍵字宣告從 AST 模型中移除，並將其移動到目標檔案中。

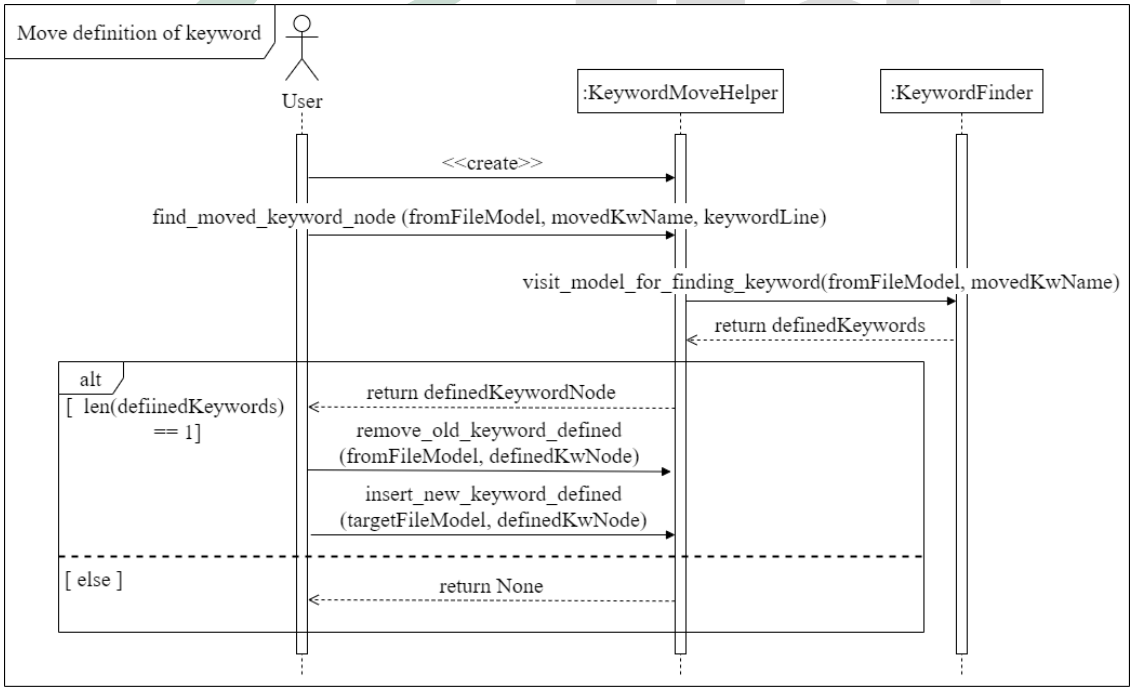

圖 4.7 移動關鍵字之循序圖

4.5.2 搜尋使用被移動關鍵字但未引入所需測試資源的測試檔案並自動引入

根據3.2.3節之流程，搜尋使用被移動關鍵字之測試檔案，並修正其中未引入所需測試資源的測試檔案。圖4.8為搜尋使用被移動關鍵字但未引入所需測試資源的測試檔案並自動引入之循序圖，透過 FileChecker 類別之函式，於檔案中比對關鍵字名稱與被移動關鍵字宣告之名稱是否相同，並驗證其是否有引入原先所需測試資源，如果兩種條件皆符合，則將其列入搜尋結果，以此取得使用被移動關鍵字之測試檔案。最後 KeywordMoveHelper 類別將藉由 FileChecker 類別之函式，得出有引入所需測試資源之 AST 模型，並與全部 AST 模型比對得到未引入所需測試資源的測試檔案，且為其引入所需測試資源。

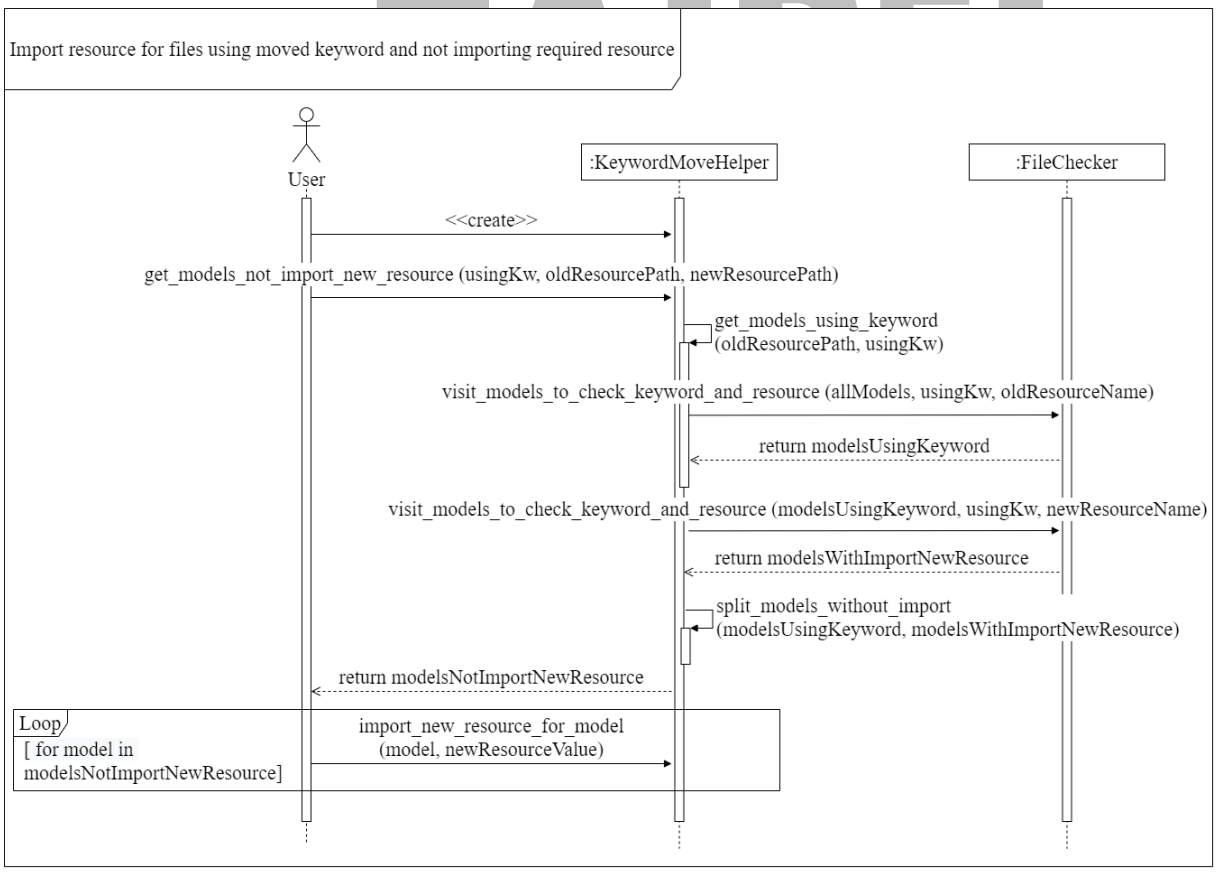

圖 4.8　搜尋使用被移動關鍵字但未引入所需測試資源的測試檔案並自動引入之循序圖

4.6 外掛程式擴充之實作

圖4.9為外掛程式擴充後之類別圖，下列將針對因擴充所新增之類別：

- **NewRefactorHelper**：用於呼叫重構功能的類別。
- **WrapStepsAsANewKeywordHandler**：用於抽取重複步驟成為新關鍵字的 handler 元素，其定義 WrapStepsAsANewKeywordCommand 的實際行為。
- **MoveKeywordDefinedToAnotherFileHandler**：用於移動關鍵字宣告的 handler 元素，其定義 MoveKeywordDefinedToAnotherFileCommand 的實際行為。
- **CreateANewKeyword**：用於提供創立新關鍵字的使用者介面。
- **AddArgumentsForKeywordReplacingSameSteps**：用於提供取代重複步驟時，可新增關鍵字參數的使用者介面。
- **FileSelectionView**：用於顯示檔案樹狀結構及選擇檔案目標的視圖元素。
- **SameKeywordsSelectionView**：用於顯示重複步驟清單及選擇取代對象的視圖元素。
- **SameStepsBlock**：用於儲存重複步驟資訊的類別。
- **Keyword**：用於儲存單一關鍵字資訊的類別。
- **Folder**：用於儲存資料夾資訊的類別。
- **Model**：用於儲存 AST 模型資訊的類別。

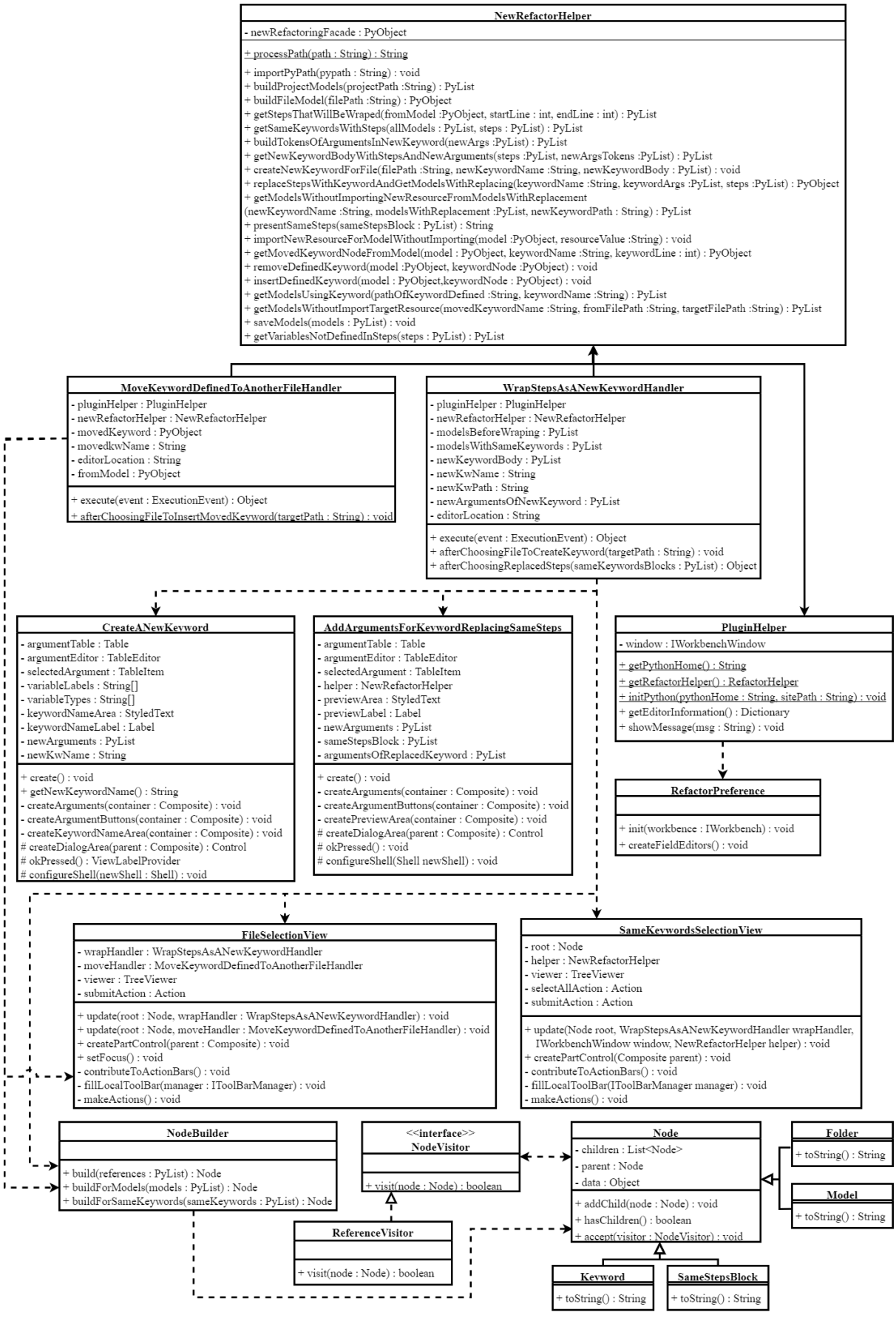

圖 4.9　外掛程式擴充後之類別圖

根據3.3節所敘述之方法，本論文將4.2節所實作之重構功能與劉冠志論文 [3] 提供之 Eclipse 外掛程式進行結合，以此進行擴充，其中提供了抽取重複步驟為新關鍵字及移動關鍵字宣告兩種方法，下列小節將分別介紹其實作。

4.6.1 抽取重複步驟成為新關鍵字

抽取重複步驟成為新關鍵字功能提供使用者於測試腳本中抽取步驟成為新關鍵字，並搜尋重複步驟以新關鍵字進行取代，最後為其引入所需測試資源，此功能實作共可分為三部分進行講解。第一部分為抽取步驟成為新關鍵字，圖4.10為其循序圖，當 WrapStepsAsANewKeywordCommand 被執行時，Eclipse 會呼叫定義其實際行為的 WrapStepsAsANewKeywordHandler，WrapStepsAsANewKeywordHandler 會透過 PluginHelper 取得使用者所選取的步驟、該檔案之路徑及該檔案所屬專案位置，並利用 NewRefactorHelper 解析專案路徑下的所有測試檔案提供後續使用。解析完成後其會從選取步驟中取出未宣告於步驟中的變數，並將其列入新關鍵字之參數，利用 CreateANewKeyword 提供使用者確認新關鍵字架構及輸入新關鍵字名稱，送出資訊後透過 NodeBuilder 將解析完成的測試檔案建置成樹狀結構，並顯示於 FileSelectionView 提供使用者選擇創立新關鍵字之位置，圖4.11為選擇新關鍵字位置之視圖實例。

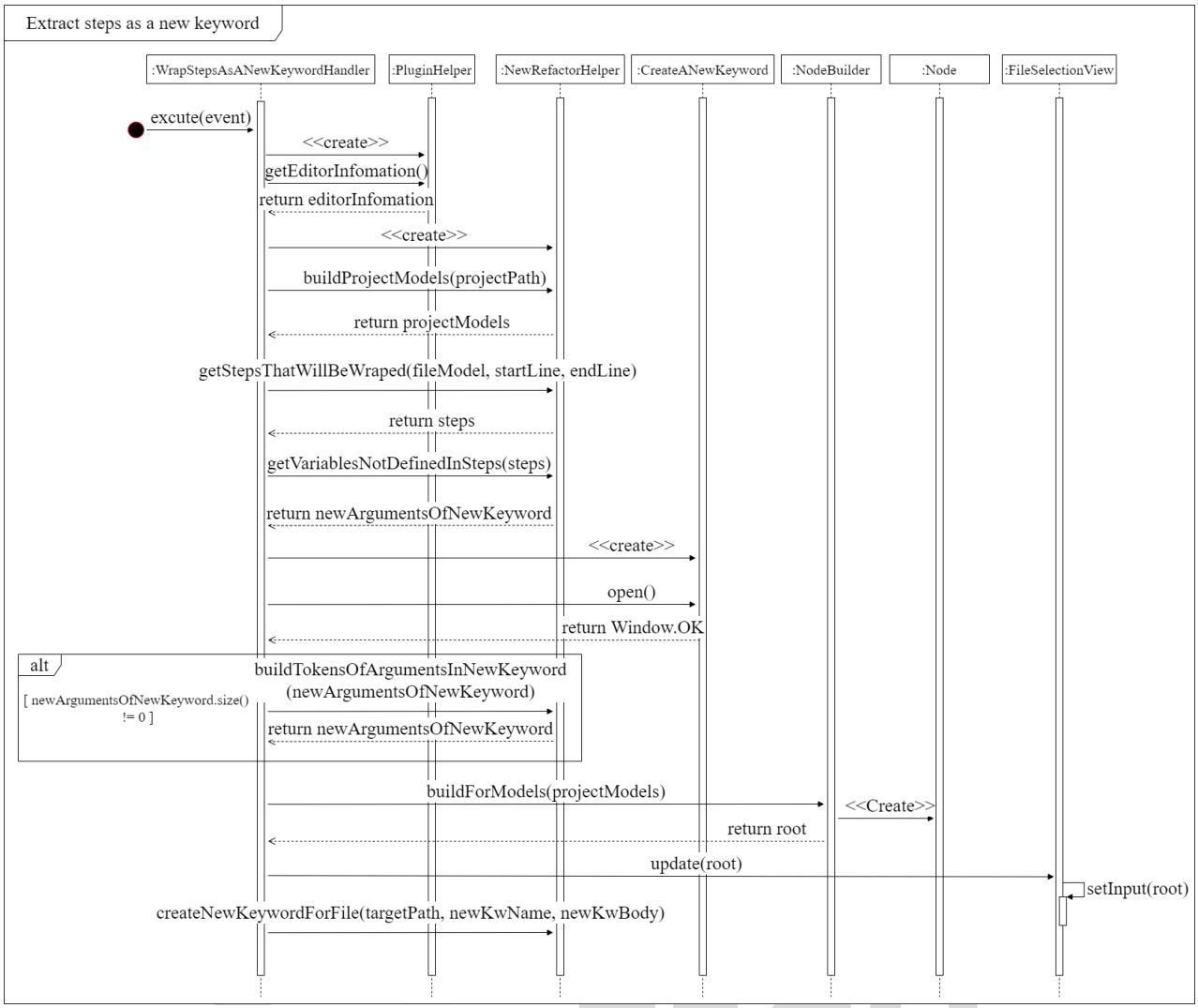

圖 4.10　抽取步驟成為新關鍵字之循序圖

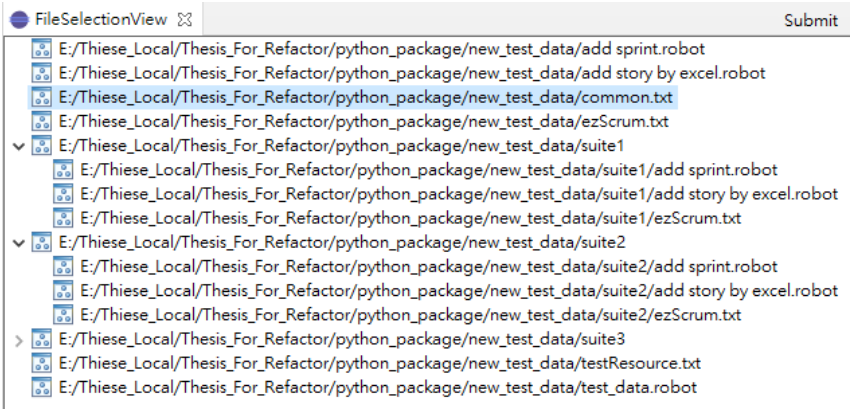

圖 4.11　選擇新關鍵字位置之視圖實例

第二部分為搜尋重複步驟並以新關鍵字進行取代，圖4.12為其循序圖，WrapStepsAsANewKeywordHandler 利用 NewRefactorHelper 搜尋重複步驟，並透過 NodeBuilder 將所有重複步驟建置成樹狀結構，後續顯示於 SameKeywordsSelectionView 並提供使用者選擇欲取代之重複步驟，圖4.13為選擇欲取代之重複步驟的視圖實例。當使用者送出選取之重複步驟後，將透過 AddArgumentsForKeywordReplacingSameSteps 提供使用者為取代重複步驟之新關鍵字加入參數實際資料的視窗，其會顯示重複步驟內容及參數資料的輸入介面，圖4.14為其視窗實例。

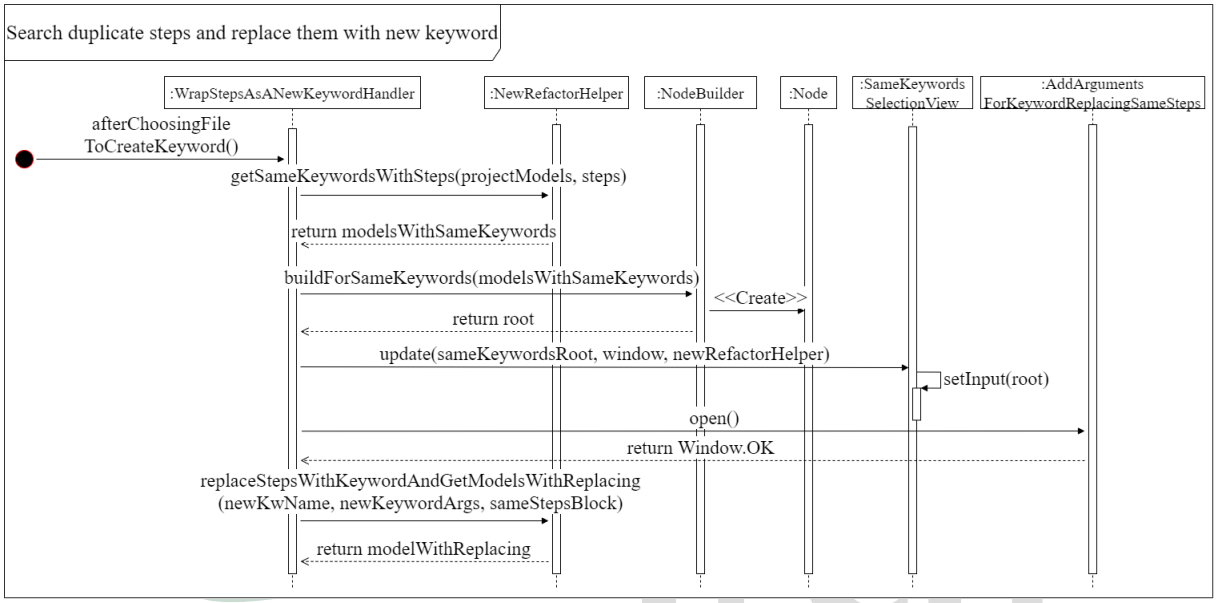

圖 4.12　搜尋重複步驟並以新關鍵字進行取代之循序圖

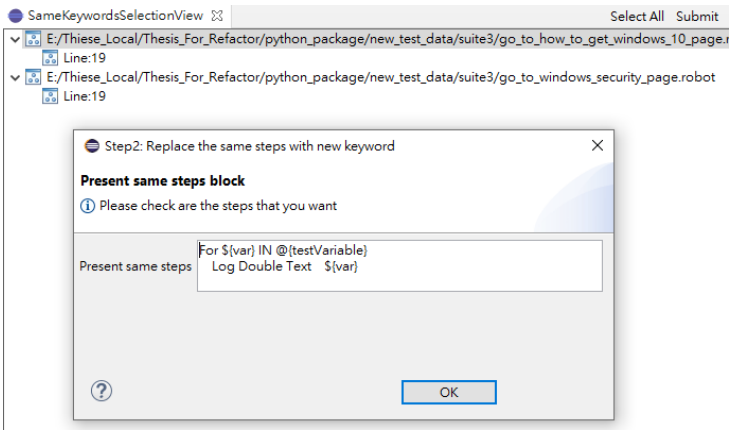

圖 4.13　選擇需以新關鍵字取代之重複步驟的視圖實例

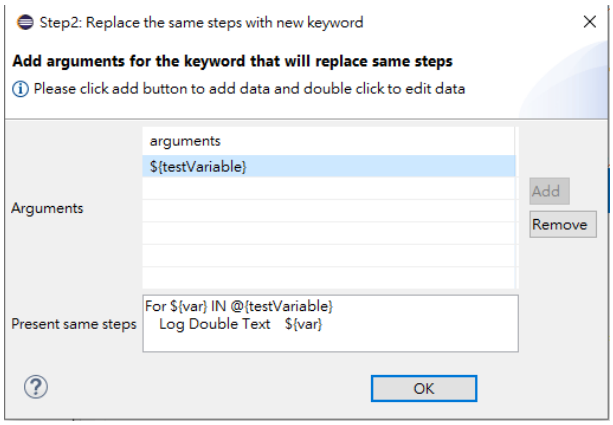

圖 4.14　取代重複步驟之新關鍵字加入參數實際資料之視窗實例

第三部分為引入新關鍵字所需之測試資源，圖4.15為其循序圖，透過 NewRefactorHelper 於使用新關鍵字之測試檔案中，搜尋未引入所需測試資源的檔案，並將所需測試資源及使用新關鍵字之檔案資訊進行路徑比對，最後為其引入所需之相對路徑。

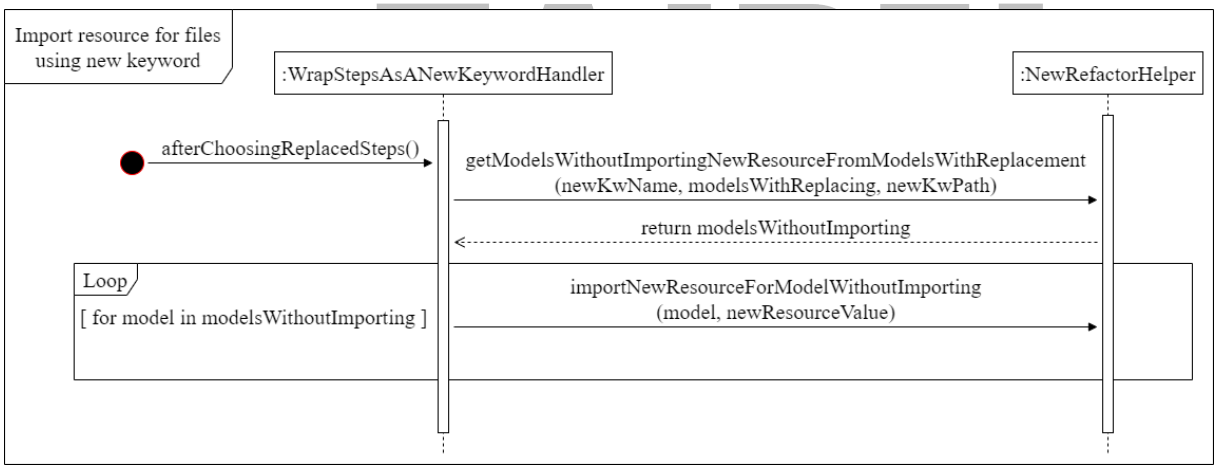

圖 4.15　引入新關鍵字所需測試資源之循序圖

4.6.2　移動關鍵字宣告

使用者可於測試檔案中移動關鍵字宣告，並於移動後為原先已使用此關鍵字之測試檔案引入其所需測試資源。圖4.16為其循序圖，當 MoveKeywordDefinedToAnotherFileCommand 被執行時，Eclipse 會呼叫定義其實際行為的 MoveKeywordDefinedToAn-

otherFileHandler，其透過 PluginHelper 取得選取之關鍵字宣告、該檔案之路徑及該檔案所屬專案位置，並利用 NewRefactorHelper 解析專案路徑下的所有測試檔案。透過 NodeBuilder 將解析完成的測試檔案建置成樹狀結構，並顯示於 FileSelectionView 提供使用者選擇移動的目標檔案。提交目標檔案後，透過 NewRefactorHelper 將關鍵字宣告移動到目標檔案上，並搜尋所有使用此關鍵字但未引入所需測試資源的測試檔案，最後為其引入所需測試資源之相對路徑。

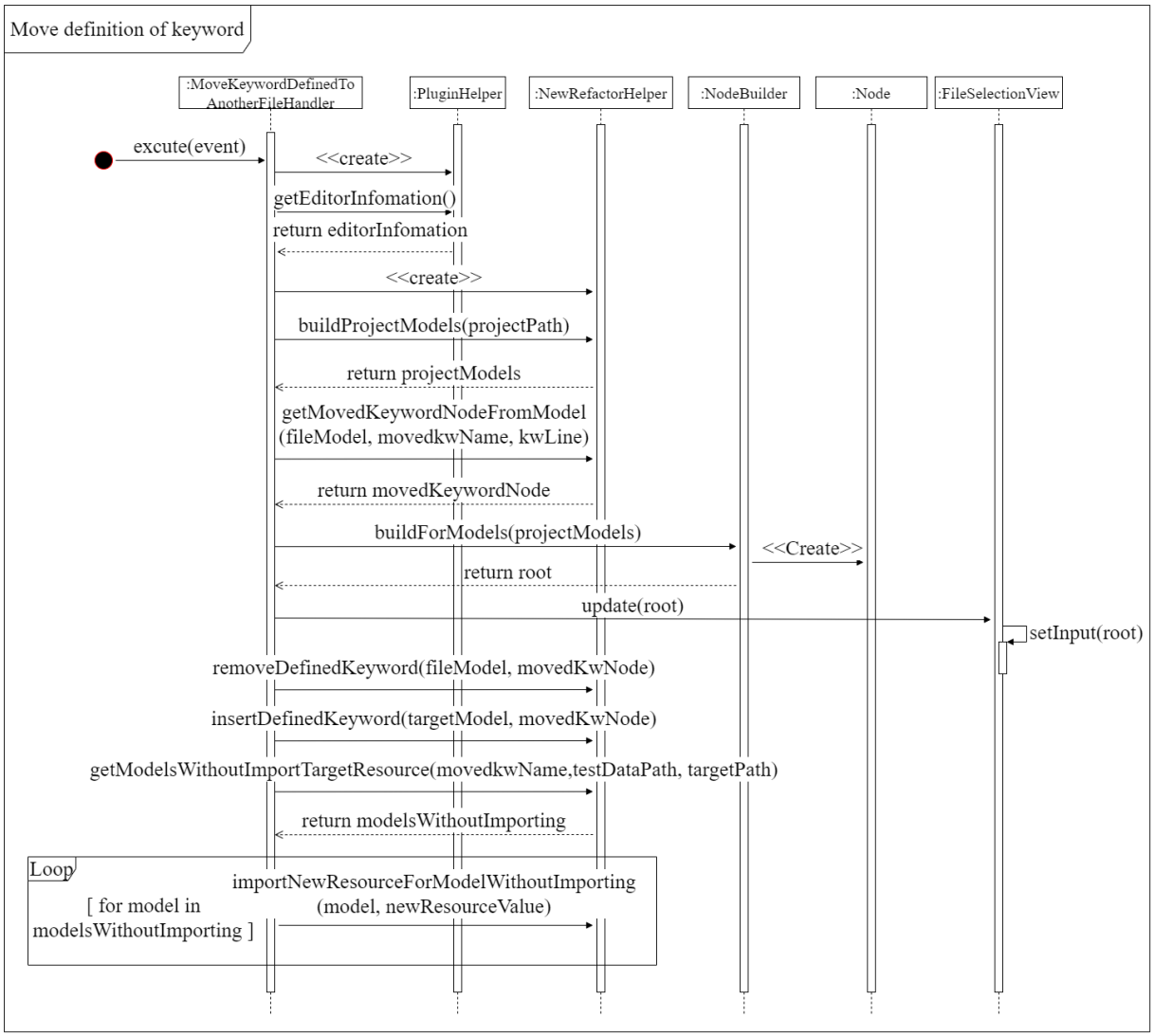

圖 4.16　移動關鍵字宣告之循序圖

第五章　案例分析

本論文將以自行設計之 Microsoft 網頁相關測試專案作為實例，其中含有四個測試套件、兩個測試資源，分別以團隊最常使用的 Visual Studio Code 搜尋取代工具及4.6節所擴充後之 Eclipse 外掛程式進行重構，其重構方法為抽取測試腳本之重複步驟成為新關鍵字及移動關鍵字宣告兩種，並且邀請國立台北科技大學軟體系統實驗室之測試團隊成員協助使用兩種工具進行重構並比較其時間差異，以此確認擴充後之重構工具對於團隊是否有實質之幫助。

5.1　案例一：抽取測試腳本中的重複步驟成為新關鍵字並引入所需測試資源

程式碼5.1為測試專案的測試套件之一，其含有一個測試案例，用來測試前往 Windows 安全性頁面之功能，並且測試其功能正常後，於報表中印出歡迎相關之訊息。程式碼5.2同為測試專案中的測試套件，同樣含有一個測試案例，用來測試前往 Windows10 功能頁面之功能，於功能驗證完成後，在報表中印出歡迎相關之訊息。

比對程式碼5.1(18-20 行) 及程式碼5.2(18-20 行) 可以發現，其在於功能驗證後都會印出歡迎相關之訊息，因此可以將它們認定為重複步驟並進行重構。此測試專案中共有四個測試案例都擁有重複之步驟，需要將其抽取成新關鍵字並取代後，引入其所需之測試資源，因此本論文邀請五位測試團隊成員依照下列小節之方式，進行相對應之重構，並於後續比較其使用差異。

程式碼 5.1　前往 Windows 安全性頁面之測試套件

```
1   *** Settings ***
2   Library     SeleniumLibrary
3   Resource    ../microsoft.txt
4
5   Test Setup    Run Keywords    Go To Microsoft
6   ...    AND    Open Language Option
7   ...    AND    Select English Language
8
9   *** Variables ***
10  @{welcomeTaipei} =    Welcome    To    Taipei
11
12  *** Test Cases ***
13  Go To "Windows Security" Page And Log Welcome Text
14      Go To Windows Page
15      Open Windows 10 Menu
16      Go To "Windows Security" Page
17      "Windows Security" Page Should Be Visble
18      FOR    ${var}    IN    @{welcomeTaipei}
19          Log Double Text    ${var}
20      END
21      [Teardown]    Close Browser
```

程式碼 5.2　前往 Windows10 功能頁面之測試套件

```
1   *** Settings ***
2   Library     SeleniumLibrary
3   Resource    ../microsoft.txt
4
5   Test Setup    Run Keywords    Go To Microsoft
6   ...    AND    Open Language Option
7   ...    AND    Select English Language
8
9   *** Variables ***
10  @{welcomeTainan} =    Welcome    To    Tainan
11
12  *** Test Cases ***
13  Go To "Windows 10 features" Page And Log Welcome Text
14      Go To Windows Page
15      Open Windows 10 Menu
16      Go To "Windows 10 features" Page
17      "Windows 10 features" Page Should Be Visble
18      FOR    ${var}    IN    @{welcomeTainan}
19          Log Double Text    ${var}
20      END
21      [Teardown]    Close Browser
```

5.1.1 使用 Visual Studio Code 搜尋取代工具

首先利用 Visual Studio Code(VSCode) 將重複步驟複製至目標測試資源中，以其做了什麼為新關鍵字之名稱，且將未宣告於步驟內之變數作為新關鍵字之參數，藉此創立新關鍵字，結果如程式碼5.3。後續於搜尋工具中，以重複步驟之關鍵字為搜尋目標，搜尋結果如圖5.1所示，其中一個搜尋結果為關鍵字之宣告，非所需之重複步驟故不取代，其餘四個結果分別與重複步驟之參數數量、關鍵字順序及關鍵字名稱相同，因此以新關鍵字進行取代。

後續於已進行重複步驟取代之測試套件，檢查其是否有引入新關鍵字所需之測試資源，為未引入者進行引入之動作。

程式碼 5.3　新關鍵字架構

```
Log Welcome Text
    [Arguments]    ${welcomeTexts}
    FOR    ${var}    IN    @{welcomeTexts}
        Log Double Text    ${var}
    END
```

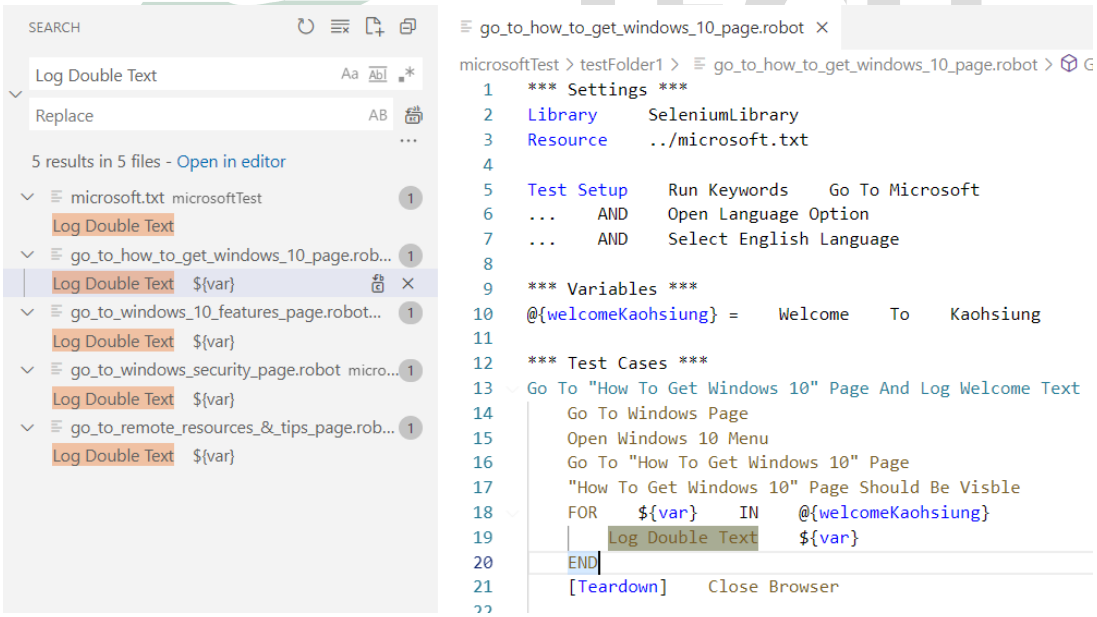

圖 5.1　重複步驟搜尋結果

5.1.2 使用擴充後之 Eclipse 外掛程式

後續使用擴充後之外掛程式的 Wrap Steps As A New Keyword 功能重構5.1節所提及之重複步驟，選取其中一個測試套件中的重複步驟，其將會自動把未宣告於步驟中的變數加入新關鍵字的參數，如圖5.2所示，於此視窗可以同時決定新關鍵字之名稱，並且創立新關鍵字於圖5.3視圖中所選擇之檔案。根據專案下所檢查出含有重複步驟的測試套件，其可在圖5.4之視圖中選擇要以新關鍵字取代之重複步驟，並且為每個要使用的新關鍵字決定其參數實際值，以此完成重複步驟之取代，最後外掛程式將自動引入新關鍵字所需之測試資源。

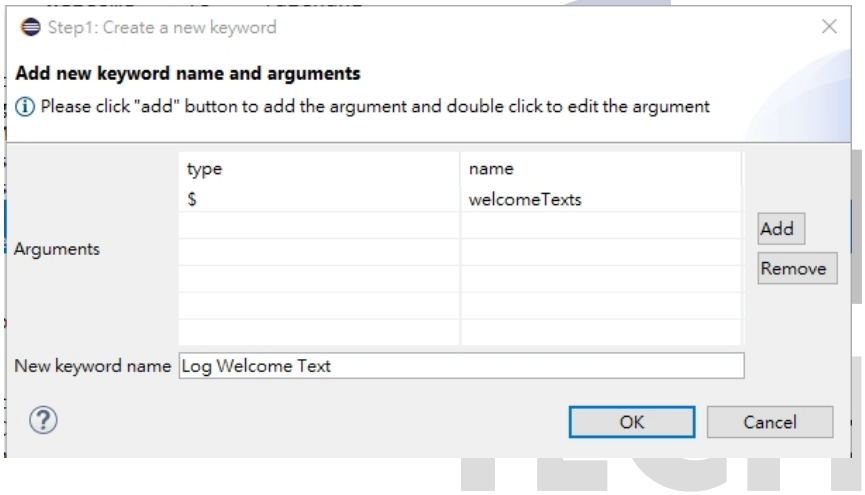

圖 5.2　創立新關鍵字視窗結果

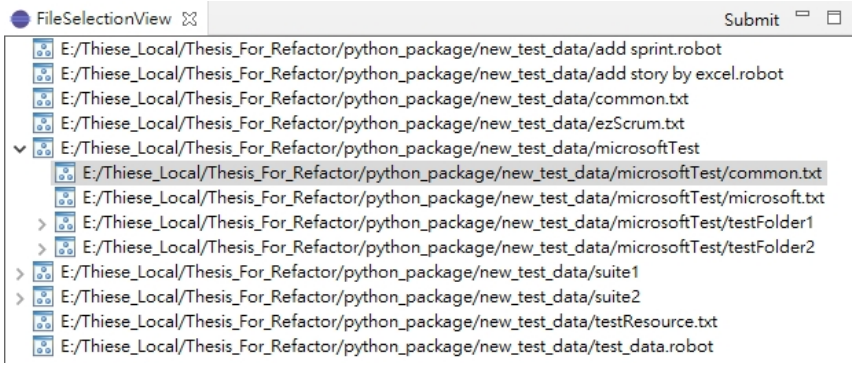

圖 5.3　專案下檔案顯示視圖結果

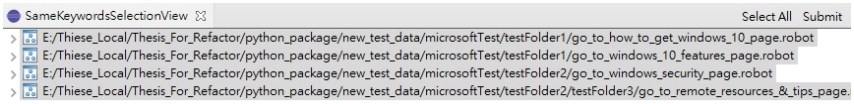

圖 5.4　搜尋重複步驟結果之視圖

5.1.3　重構工具使用之比較

表 5.1　使用兩種工具包裹重複步驟成為新關鍵字並引入所需測試資源之比較

	使用 VSCode 重構花費之時間	使用擴充後之 Eclipse 外掛程式重構花費之時間
測試人員 1	06m02s	03m24s
測試人員 2	05m12s	04m20s
測試人員 3	07m31s	04m20s
測試人員 4	06m20s	04m59s
測試人員 5	06m29s	06m07s
平均時間	**06m18s**	**04m38s**

表5.1為紀錄團隊測試人員使用兩種不同工具進行此重構之比較，從表中可見每一位測試人員根據經驗之不同，使用 VSCode 進行重構所花費之時間也略微不同，平均大約花費了 6 分 18 秒，並且重構過程中，測試人員有三位不小心忽視了新關鍵字所需之測試資源，導致測試失敗；而使用擴充後之 Eclipse 外掛程式重構時，平均大約花費了 4 分 38 秒，並且未有測試資源沒被引入之狀況發生。由此比較可發現，使用擴充後之 Eclipse 外掛程式進行此重構時，因為不需要手動搜尋重複步驟以及檢查所需之測試資源是否引入，能夠花費較少的時間，其大約減少了 26.4% 的時間，並且較不容易發生錯誤。

5.2 案例二：移動測試資源中的關鍵字宣告並引入所需測試資源

程式碼5.4為測試專案的測試套件之一，其中第 5 行之 Go To Microsoft 關鍵字於其他測試套件剛好也需要使用，因此必須將其之宣告移動至共用之測試資源，以便其他測試套件使用。在移動到目標測試資源後，必須檢查原先已使用此關鍵字之四個測試套件是否都有引入其所需之測試資源，避免發生未宣告關鍵字之錯誤。

根據此案例之重構，本論文邀請與5.1節相同之五位測試團隊成員使用下列小節之方式，進行相對應之重構，最後比較其使用差異。

程式碼 5.4　前往如何取得 Windows10 頁面之測試套件

```
1   *** Settings ***
2   Library     SeleniumLibrary
3   Resource    ../microsoft.txt
4
5   Test Setup    Run Keywords    Go To Microsoft
6   ...    AND    Open Language Option
7   ...    AND    Select English Language

8   *** Variables ***
9   @{welcomeKaohsiung} =    Welcome    To    Kaohsiung
10
11  *** Test Cases ***
12  Go To "How To Get Windows 10" Page And Log Welcome Text
13      Go To Windows Page
14      Open Windows 10 Menu
15      Go To "How To Get Windows 10" Page
16      "How To Get Windows 10" Page Should Be Visble
17      Log Welcome Text    ${welcomeKaohsiung}
18      [Teardown]    Close Browser
```

5.2.1 使用 Visual Studio Code 搜尋取代工具

首先利用 VSCode 將需要移動之關鍵字宣告於其所在檔案中移除，並在共用測試資源上完整複製遭移除之關鍵字宣告，後續於搜尋工具之視窗，以被移動之關鍵字名稱

進行搜尋，搜尋結果如圖5.5所示，其中一個搜尋結果為關鍵字之宣告的，故不需檢查，其餘結果皆為原先已使用此關鍵字之測試套件。接著於每個測試套件中搜尋是否有引入共用之測試資源，如否則為其引入以避免測試錯誤。

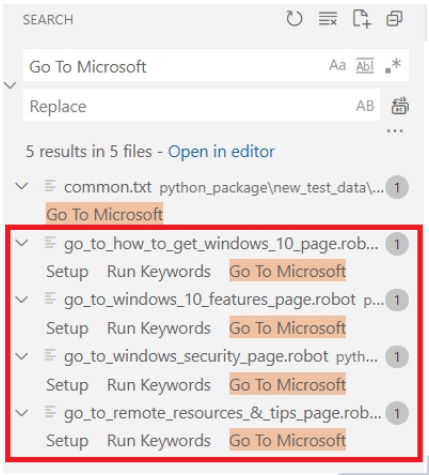

圖 5.5　搜尋使用被移動關鍵字之測試套件結果

5.2.2　使用擴充後之 Eclipse 外掛程式

接著使用擴充後之外掛程式的 Move Keyword Defined To Another File 功能重構5.2節所提及的需移動之關鍵字宣告，選取其關鍵字名稱後，即可將關鍵字宣告移動至圖5.6視圖中所選取之目標檔案，後續外掛程式將自動為原先已使用此關鍵字之測試套件引入所需測試資源。

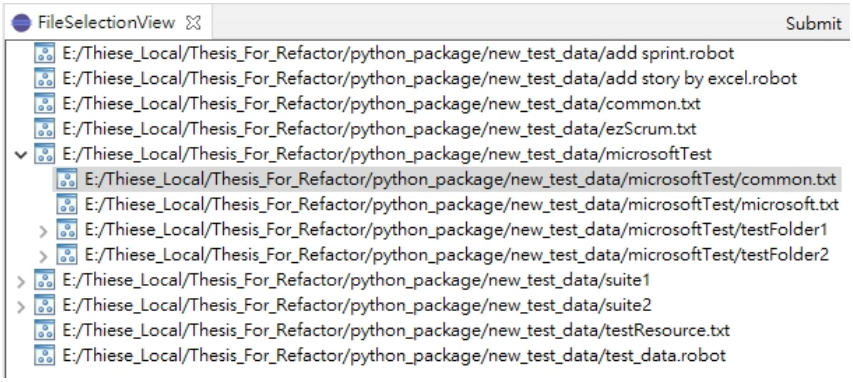

圖 5.6　專案下全部測試檔案之視圖

5.2.3 重構工具使用之比較

表 5.2 使用兩種工具移動關鍵字宣告並引入所需測試資源之比較

	使用 VSCode 重構花費之時間	使用擴充後之 Eclipse 外掛程式重構花費之時間
測試人員 1	01m14s	00m26s
測試人員 2	01m35s	00m18s
測試人員 3	01m17s	00m22s
測試人員 4	01m00s	00m26s
測試人員 5	02m02s	00m41s
平均時間	**01m25s**	**00m26s**

表5.2為紀錄團隊測試人員使用兩種不同工具進行此重構之比較，從表中可發現，使用 VSCode 進行重構時，平均花費了 1 分 25 秒，且有兩位測試人員對於部分測試套件有錯誤引入關鍵字所需測試資源之相對路徑，導致測試錯誤；而使用擴充後之 Eclipse 外掛程式重構時，平均花費了 26 秒，且引入所需測試資源之路徑皆為正確。從此比較可發現，使用擴充後之 Eclipse 外掛程式進行此重構時，因為不需要自我檢查測試套件是否都有引入所需之測試資源，能夠花費較少的時間，其大約減少了 69.4% 的時間，並且較不容易因人為檢查缺漏而發生錯誤。

第六章　結論與未來展望

6.1　結論

本論文擴充了劉冠志論文 [3] 於 Eclipse 實作之外掛程式，增加了兩種重構方法，不只讓團隊於選擇重構方法時能夠更加多元，且能夠減少搜尋的缺漏及取代的錯誤。根據第三章的研究方法，使擴充後的外掛程式在搜尋重複步驟時，能夠確實地得到需要進行新關鍵字取代之重複步驟，而不是得到與團隊需求不同之步驟；此外移動關鍵字宣告後，其搜尋原先使用關鍵字之測試腳本時，也能夠確實地得到真正有使用被移動之關鍵字的測試腳本，而不是得到使用類似名稱之關鍵字的測試腳本等等。

第四章中將擴充之重構功能結合至原先已存在之外掛程式後，使用者不必進行搜尋，即可得知其需要被取代的重複步驟，並且可以依照需求進行選擇，而移動關鍵字宣告後，也不需要自行引入測試腳本所需之測試資源，即可完成重構。由第五章中可得知，當團隊遇到實際案例時的重構過程，並且使用擴充後之外掛程式，確實可以減少重構之時間，因而提升整體重構效率。

6.2 未來方向

本論文擴充之測試腳本重構工具仍有待改善之處，以下將分為三點說明：

- **執行重構之效能：**本論文擴充之測試腳本重構工具在執行任一重構方法前，都必須先行解析測試專案下的所有測試資料，而使用者通常必須花費較長時間等待其解析完成，並且專案中的資料越多時，則等待時間也會相對增加。若能在啟動重構工具時，利用其他執行緒事先於背景解析專案下之測試資料，且資料有任何新增、修改、刪除等操作時，則再次重新解析相對應資料，確保檔案內容之正確，即可消除執行重構方法時的等待時間，使重構效率更加提升。
- **重構方法之新增：**測試腳本重構工具經本論文擴充後，其重構方法目前有五個，重新命名關鍵字、重新命名變數、修改關鍵字介面、包裹重複步驟成為新關鍵字及移動關鍵宣告，重構方法擁有十分多種，而測試人員於開發時，也會有遭遇其他重構需求的時候，因此如果能夠增加更多種類之重構方法，其能減少測試人員於手動重構時錯誤及缺漏的產生，表6.1為未來可擴充之重構方法。
- **其他功能：**本論文所擴充之測試腳本重構工具是參考其他 IDE 重構功能所開發，因此於重構完成後必須手動執行被重構之測試腳本，才可知道其是否測試功能與未重構前相同，若能與其他測試腳本變更偵測工具結合，例如：邱文煜論文 [17] 所提供之測試選擇工具，於重構完成後，即可執行有所變更之測試腳本，能夠更快地知道重構後之結果是否如測試人員所需，縮短測試人員於重構後之確認時間。

表 6.1　未來可擴充之重構方法

可擴充之重構方法	重構方法之描述
抽取變數	搜尋出多個重複數值並建立新變數進行取代。
移動變數	移動變數宣告至其他測試資源中，並確保引入所需測試資源。
內聯化 (Inline) 關鍵字	刪除關鍵字宣告並將使用其的參考以關鍵字的實作取代。
內聯化 (Inline) 變數	刪除變數宣告並將使用其的參考以變數的實作取代。
重新命名測試檔案	重新命名測試檔案，且確保原先引入此檔案之資訊正確。

參考文獻

[1] Eclipse Platform architecture. https://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html.

[2] Pekka Laukkanen. Data-driven and keyword-driven test automation frameworks. Master's thesis, Helsinki University of Technology, 2006.

[3] 劉冠志. Rf refactoring：一個 robot framework 測試腳本之重構工具. Master's thesis, 國立台北科技大學, 2020.

[4] RED - Robot Editor. http://nokia.github.io/RED/help/.

[5] Visual Studio Code - code editing. https://code.visualstudio.com/.

[6] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA, USA, 1999.

[7] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*, volume 620. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[8] Markus Gärtner. *ATDD by example : a practical guide to acceptance test-driven development*. Addison-Wesley, Upper Saddle River, NJ, 2013.

[9] J.F. Smart. *BDD in Action: Behavior-driven development for the whole software lifecycle*. Manning Publications, 2014.

[10] Paul J Deitel. *Java how to program*. Pearson Education India, 2002.

[11] Robot framework api documentation. https://robot-framework.readthedocs.io/en/v3.2.2/.

[12] Adnan Masood. *Learning F# Functional Data Structures and Algorithms*. Packt Publishing Ltd, 2015.

[13] ast —Abstract Syntax Trees. https://docs.python.org/3.6/library/ast.html.

[14] J Wiegand et al. Eclipse: A platform for integrating development tools. *IBM Systems Journal*, 43(2):371–383, 2004.

[15] Samuele Pedroni and Noel Rappin. *Jython Essentials: Rapid Scripting in Java*. " O'Reilly Media, Inc.", 2002.

[16] Robert F Stärk, Joachim Schmid, and Egon Börger. *Java and the Java virtual machine: definition, verification, validation*. Springer Science & Business Media, 2012.

[17] 邱文煜. 基於 robot framework 之測試選擇工具. Master's thesis, 國立台北科技大學, 2021.