私立東海大學資訊工程與科學研究所

碩士論文

指導教授：周忠信

# 軟體設計相似度的計算

# Measuring Similarity in Software Design



研究生：洪士軒

中華民國九十二年七月二十八日

# 摘要

近幾年來，隨著物件導向設計的技術快速發展，軟體架構的設計乃逐漸受到廣泛的重視。各種幫助使用者分析、比較並探查系統架構的研究與工具也因應而生。本論文透過軟體設計相似度的研究，提出一個計算軟體設計相似度的演算法，幫助設計者經由對同類型軟體設計間的比較，以及透過觀察軟體系統設計上的關鍵點，讓設計者能方便比較兩個軟體設計上的差異與相似處，進而分析設計上的優劣並據此改善軟體設計。本研究我們實作此演算法用以分析以 Java 技術為主所發展的軟體系統設計。

關鍵字：物件導向設計、軟體設計相似度、Sink、Source、Java

# Abstract

Recently, object-oriented design has become the major trend in software development. Many research issues are raised especially for improving the quality of the software design. Among these issues, by comparing the similarity of two software is a good direction to improve the design of the software. In this research, a similarity solution for object-oriented design is proposed. It can help identify the similar and different portions of the designs. Based on the information obtained from this solution, software designers can easily study the difference and improve their design. In order to show the superiority of our solution, we further apply this technology to the Java-based software development. It shows that by observing the similarity of the design, it really can help designer identify the critical portions of the design and improve the quality.

Key words：Object-Oriented Design、Software Design Similarity、Sink、Source、Java

# 目次

# 表目錄

# 圖目錄

# 第一章 緒論

## 1.1 研究目的

隨著物件導向設計的技術快速發展，物件導向設計逐步成為軟體系統開發的重要技術之一[12][13][14]。而軟體設計的優缺點因此越來越受重視，各種相關議題的研究也隨之快速發展，這其中包括如元件的再利用(reuse)[6]、各種軟體架構(Framework) 的探討[2][3][11][18]、以及物件導向設計的設計樣式(Design Patterns)研究[16]等，皆是軟體發展的重要課題。在此同時，除了協助塑模軟體設計的 modeling 工具外[8]，各種幫助使用者探討軟體設計問題的工具，也成為重要的研究方向之一，這其中如 The SmallWorld[17]，乃是一個典型的代表。經由這些工具，使用者能輕易分析、比較並探查系統架構的設計問題，並找出設計上的優劣部份。這些相關研究及工具，無非希望透過事先分析軟體設計，進而利用分析結果改善設計並減低當軟體系統實作時所可能出現的缺陷。本論文主要目的，即在透過對「軟體設計相似度」(Similarity in Software Designs)的研究，去探討並比較同類型軟體系統設計間的相似與差異地方，進而分析出設計上的優劣處並據此改善軟體設計。

本研究首先提出一個計算軟體設計相似度的演算法，此法可有效計算出兩軟體系統設計上之相似度，並進一步指出其間的異同點。同時我們也實作此演算法用以分析以 Java 技術為主的軟體系統設計。本工具除可計算出兩 Java 技術所做的軟體設計之相似度外，同時經由分析設計上的關鍵點，本工具可進一步指出兩軟體設計間的相似部份以及差異性較高之所在，進而有效幫助 Java 軟體設計人員分析軟體設計上的優劣處。同時當需要進行軟體更新時，亦可由這些資料來分析各版本軟體系統間的差異處，以評估這項更動對整體系統的影響。另外對於軟體研發人員而言，則可經由對同類系統的比較，有效找出良好的設計方式，以便解決可能遭遇之問題。

## 1.2 論文架構

本文在第二章中，介紹本研究所使用到之技術，包含 UML/XMI、XML Parser、以及解決多對多 Matching 之作法。在第三章裡，我們將介紹本研究所提之相似度計算演算法，並以一個簡單的範例來說明整個演算法流程。本文第四章，則以兩個開放原始碼(Open Source)， httpunit 1.3 及 httpunit 1.4[7]之軟體設計作為範例，展示本研究之實際應用方式，並在此過程中，說明演算法在各個步驟中產出的實用價值。最後，第五章為本論文結論，並說明本研究未來可行之研究與發展方向。

# 第二章 相關之技術

## 2.1 UML/XMI

隨著軟體工程技術的進步，以及各種軟體開發流程評鑑、管理機制的盛行，記錄需求、軟體設計架構及描述開發過程的文件已受到高度重視，而UML(Uniform Modeling Language)便是一個廣為使用的文件紀錄語法。透過UML所紀錄的各類文件，包括需求(requirement)、軟體分析(SA, Software Analysis)以及軟體設計(SD, Software Design)等，對於往後系統的維護、修改或更新，將即有助益。

### 2.1.1 XMI 簡介

XMI(XML Metadata Interchange)[15]由OMG於1999年提出，為一個以XML為基楚的軟體設計交換標準。提出這個標準的主要目的，在於能方便且有效率的讓軟體設計在不同modeling tools以及metadata repositories間作交換。它整合了三個重要的標準：

- XML - eXtensible Markup Language；
- UML - Unified Modeling Language；
- MOF - Meta Object Facility，由OMG所提出，用於metamodeling以及metadata repository的標準。

透過整合這三項包含資料交換格式以及軟體工程技術的標準，分散於各地的使用者可在網路上透過XMI來交換物件模型(object model)及其他metadata。

為要了能夠在各種不同modeling tools間交換資料，XMI必需能夠完整記錄所有之modeling資訊。因此，透過分析XMI中的資料，將能取得軟體設計中的各種設計資訊。

由於不同的modeling tools以及metadata repositories間存在著不小的差異，要能在這些格式中順利作出轉換，XMI必需具備足夠的彈性，它的規格主要包含兩個部份：

- XML DTD Production Rules：用於產生任何以MOF為基楚的metamodel所對應之DTD。
- XML Document Production Rules：用於產生以MOF為基楚的metadata之XML文件。

因此，XMI可以整合任何以MOF描述之mode或metamodel資訊，以便於之後的資料交換或搜尋。

本節中，我們將對XMI的各種定義以及其DTD之結構，作一概略介紹。

### 2.1.2 DTD於XMI之應用

XMI亦為一種XML文件，文件內容受到DTD之規範。在XMI中，DTD之使用標準有下列幾點：

- XMI可產生任何以MOF為基楚的metamodel所對應之DTD；
- 一份XMI文件可經由內部DTD(internal DTD)之作法來定義其中所有標記；
- XML之validation動作，可用於檢查metamodel中extension之部份；
- XMI使用之DTD有下列限制：
  - 所有在XMI中定義之XML element都必需在DTD中宣告；
  - 所有的metamodel construct，如：class、attribute、association，都必需有對應的XML element；
  - 所有用於metalmodel extension之XML element都必需在DTD中宣告。

### 2.1.3 Class於Metalmodel中之呈現方式

XMI為了能完整的表達出一個metalmodel資訊，因此透過一些規則，來規範這些metalmodel class、attribute以及association表示方式：

- 在metalmodel中的每個class都需要有一個對應的XML element來表示，而此element之name即為該class名稱；
- 該XML element中會列出此metalmodel class所有的attribute；
- 每個attribute皆有一對應XML element，而此element之name即為該attribute名稱；

- 在metalmodel classes間的association則以兩個XML element表示，這兩個element分別代表此association的兩個association ends。

### 2.1.4 Metamodel Extension Mechanism

為了便於各類不同的modeling tools記錄所需資訊，以及使用者在特定的需求下，會需要記錄除了metamodel外的其它資料，XMI定義了一個機制，使得這些資料得以完整的存放於文件中。此機制稱之為extension mechanism。使用extension mechanism的規範如下：

- 透過DTD中之XMI.extension這個XML element，來存放extension資料；
- 此XML element可包含所有種類的XML element(ANY)；
- 一份XMI文件中，可擁有一個或一個以上之XMI.extension。

### 2.1.5 XMI DTD Structure

一個完整的XMI DTD應包含下列部份：

- XMI必要之DTD宣告，包含：
  - Link：XMI為了能構完整描述metamodel包含之資訊，使用了數種linking機制，如單一文件內部使用的ID與ID reference，以及跨文件使用的XLink。此部份的DTD定義如下：

```
<!ENTITY % XMI.element.att ’        xmi.id ID
#IMPLIED
          xmi.label CDATA #IMPLIED
          xmi.uuid CDATA #IMPLIED ’ ><!ENTITY %
XMI.link.att '
          href CDATA #IMPLIED
          xmi.idref IDREF #IMPLIED' >
```

  - XMI文件標記：此標記用於標示本文件為一XMI文件，並記錄文件中使用之XMI版本、文件之有效期限、以及在Parsing過程中是否執行verify動作等資訊。其DTD定義如下：

```
<!ELEMENT XMI (XMI.header?,
          XMI.content?,
          XMI.difference*,
          XMI.extensions*) >
<!ATTLIST XMI
          xmi.version CDATA #FIXED "1.2"
          timestamp CDATA #IMPLIED
          verified (true|false) #IMPLIED>
```

- XMI header：**此部份用於存放該**XMI**文件的相關資訊，此部份其**DTD**定義如下：**

```
<!ELEMENT XMI.header (XMI.documentation?,
          XMI.model*,
          XMI.metamodel*,
          XMI.metametamodel*,
          XMI.import*) >
```

**而這些資訊又分為下列幾個部份：**

- XMI.document：**其中包含了文件擁有者、文件相關說明等資料，其**DTD**定義如下：**

```
<!ELEMENT XMI.documentation (#PCDATA |
          XMI.owner | XMI.contact |
          XMI.longDescription |
          XMI.shortDescription | XMI.exporter |
          XMI.exporterVersion | XMI.notice)* >
<!ELEMENT XMI.owner ANY >
<!ELEMENT XMI.contact ANY >
<!ELEMENT XMI.longDescription ANY >
<!ELEMENT XMI.shortDescription ANY >
<!ELEMENT XMI.exporter ANY >
<!ELEMENT XMI.exporterVersion ANY >
<!ELEMENT XMI.exporterID ANY >
<!ELEMENT XMI.notice ANY >
```

- XMI.model、XMI.metamodel、XMI.metametaodel：**記錄此份**XMI

文件是用於存放何種類型之model、metamodel、及metametamodel資訊。其DTD定義如下：

```
<!ELEMENT XMI.model ANY>
<!ATTLIST XMI.model
          %XMI.link.att;
          xmi.name CDATA #REQUIRED
          xmi.version CDATA #REQUIRED><!ELEMENT
XMI.metamodel ANY>
<!ATTLIST XMI.metamodel
          %XMI.link.att;
          xmi.name CDATA #REQUIRED
          xmi.version CDATA #REQUIRED><!ELEMENT
XMI.metametamodel ANY>
<!ATTLIST XMI.metametamodel
          %XMI.link.att;
          xmi.name CDATA #REQUIRED
          xmi.version CDATA #REQUIRED>
```

- XMI.import：記錄此文件所包含的其他文件，並透過XLink作連結的動作。其DTD定義如下：

```
<!ELEMENT XMI.import ANY>
<!ATTLIST XMI.import
          %XMI.link.att;
          xmi.name CDATA #REQUIRED
          xmi.version CDATA #REQUIRED>
```

- XMI.content：此部份用於存放model、metamodel、及metametamodel之內容，所有相關於這三者而定義出之標記皆需為於此標記內。其DTD定義如下：

```
<!ELEMENT XMI.content ANY >
```

- XMI.extension：用於存放文件作者或是modeling tool所需的資訊，其DTD定義如下：

```
<!ELEMENT XMI.extensions ANY >
<!ATTLIST XMI.extensions
```

- XMI.deference：在XMI中，為了節省不同版本所造成的空間浪費，乃定義deference機制。透過這個機制，可標示出兩份XMI文件的差異，並於其間進行轉換動作。其DTD定義如下：

```
<!ELEMENT XMI.difference (XMI.difference| XMI.add |
XMI.delete | XMI.replace)* >
<!ATTLIST XMI.difference
          %XMI.element.att;
          %XMI.link.att;>
<!ELEMENT XMI.add EMPTY >
<!ATTLIST XMI.add
          %XMI.element.att;
          %XMI.link.att;>
<!ELEMENT XMI.delete EMPTY >
<!ATTLIST XMI.delete
          %XMI.element.att;
          %XMI.link.att;>
<!ELEMENT XMI.replace ANY >
<!ATTLIST XMI.replace
          %XMI.element.att;
          %XMI.link.att;
          xmi.position CDATA "-1">
```

- XMI.reference：記錄與此XMI文件相關之文件。其DTD定義如下：

```
<!ELEMENT XMI.reference ANY >
<!ATTLIST XMI.reference %XMI.link.att; >
```

- 對特定metamodel所定義的XML element宣告：如UML DTD中針對UML格式定義出的XML element，如上一點所述，在XMI文件中，這些XML element將被包含於XMI.content中；

透過這些定義，使得XMI具有良好的彈性，也讓一份XMI文件能完整記錄軟體設計之資訊。

擁有上列特性及優點，目前市面上大多數的modeling tools皆已支援XMI標

準。透過這些工具，使用者可將設計之model及metamodel以XMI的格式來存放，便於日後處理工作。

**2.2 XML Parser**

由於擁有便於資料交換、可自定標記(tag)、可使用 XSL 作格式轉換等優點，XML 格式在近幾年內發展迅速，受到許多應用軟體及企業系統的支援。但 XML 是一種文件而非程式，軟體系統必需要具有分析 DTD 及文件本身的能力，才能快速的從 XML 文件中取出資料。因此，為了便於 XML 文件的處理，XML Parser 的發展便成了一種必然的趨勢。

目前的 XML Parser 大致上可分為兩大類：DOM Parser 及 SAX Parser，其特性分述如下：

- SAX Parser：根據 W3C 定義之 SAX(Simple API for XML parser)規格實作之 XML Parser。此種 Parser 僅對輸入之 XML 文件作一次 Scan 動作，並根據軟體開發人員定義之 Handler 處理過程中 Scan 到之資料，具有下列優點：
  - 節省記憶體資源：在 Parsing 過成中不會產生暫存之資料結構，因此對於記憶體之需求量較少；
  - 效率較高：只作一次 Scan 動作，並於 Scan 過程中取出所需資料，因此 Parsing 速度較快。

  而其缺點為：
  - 只能對文件作一次處理動作：因為只 Scan 一次，且不產生任何暫存資料結構，因此只能根據 Handler 物件之內容對文件作一次處理動作，若需重複操作，則要作多次的 Parsing 方可辦到。
- DOM Parser：此類 Parser 根據 W3C 定義的 DOM(Document Object Mode)[4] 結構，將 XML 文件轉換為一個可由軟體處理的樹狀資料結構，文件中每個標記皆以一物件型式存放於記憶體中。此類 Parser 有下列優點：
  - 文件結構完整性佳：透過 DOM 的樹狀結構，記憶體中存放之物件可完整包含 XML 文件之結構性，便於軟體系統根據此結構進行資料處理；

- 可對資料作多次操作：由於產生之 DOM 結構存放於記憶體中，因此，可對這些資料進行反覆操作。

但此作法也有下列缺點：

- 記憶體需求量較大：由於整份 XML 文件皆會以 DOM 格式存放在記憶體中，所有的標記將轉為對應之物件，因此需要使用較多的記憶體；
- 一次 Parsing 的動作所需時間較長：DOM Parser 基本上也是使用了 SAX Parser 的元件，除了 SAX 中單純的 Scan 動作外，它還需建立出該樹狀結構，因此，在一次 Parsing 的動作中會使用較多的時間。

根據這些特性，系統開發人員在開發的過程中，需要針對 XML 文件的特性來選擇使用何種 XML Parser。

目前 XML Parser 的領域裡，有數個不同單位分別以上述兩類型實作出 XML Parser，最常被使用的有下列幾種：

- Apache Xerces/IBM xml4j：此兩個 API 皆以 Xerces 為核心，分別實作出 DOM Parser 及 SAX Parser；
- Sun Crimson：此 API 皆以 Sun 開發出之 Crimson 為核心，分別實作出 DOM Parser 及 SAX Parser；
- JDOM：此 API 提供了 Java 版之 DOM 結構實作方案，以 Java 中之對應物件取代 W3C 定義之 DOM Interface，實作出 DOM Parser，對於 Java 程式開發人員來說，此 API 操作較前兩者方便許多，但此 API 僅提供 DOM Parser 之實作，無 SAX Parser 的供能。

**2.3 Hungarian Algorithm**

Hungarian Algorithm 是一個解決加權(weighted)多對多 Matching 問題之演算法[5]，目的在於找出 edge weighted bipartite graph 中 cost 總合最小之 maximum matching。Maximum matching 為一個 bipartite graph 中包含最多 edge 之 matching，對於這類問題，它提供了完整的解決方案及良好的效率。此演算法之步驟如下：

START：由 $X=\{x_1, x_2, \ldots\}$ 及 $Y=\{y_1, y_2, \ldots\}$ 兩個點集合 (vertices)所構成的 bipartite

graph，初步並不存在任何 matching。

Part A：建立 initial partial graph。

步驟一：付予圖中所有點一個權重值，方法如下：

- 將 X 集合中的每個點，皆付予一權重，此權重之值等於該點所擁有之最低權重連線(edge)；
- 將每個點所擁有的每個連線之權重，減去該點之權重值；
- 對 Y 集合中之所有點重複上述步驟。

步驟二：將此時圖形中權重值不為 0 的連線去除，假設此時之連線關係為 M。

Part B：在圖形中作標記

步驟三：將 X 中沒有任何屬於 M 之連結的點標上(*)符號，此時可能出現兩種情況：

- 若此時圖形中沒有此類點，則停止演算法之執行；
- 若此時圖形中有此類點，則繼續進行步驟四。

步驟四：

- 選擇 X 中一個剛標記之點，假設其為 $x_i$，找出 Y 中所有目前沒有任何標記，且與 $x_i$ 有不屬於 M 之連結的點，標上($x_i$)符號；
- 重複此步驟直到 X 中所有剛標記上之點皆經過處理，之後進行步驟五。

步驟五：

- 選擇 Y 中一剛標記上之點，假設其為 $y_j$，找出 X 中所有目前沒有任何標記，且與 $y_j$ 有屬於 M 之連結的點，標上($y_j$)；
- 複此步驟直到 Y 中所有剛標記上之點皆經過處理；

重複步驟四及步驟五，直到下列兩個條件中有一者成立為止：

- 某個已標記的點之標記動作無法完成，此種情況稱之為 *breakthrough*，此時演算法跳至 Part C 步驟六繼續進行；
- 無法於圖形中進行進一步之標記動作，此時演算法跳至 Part D 步驟八繼續

進行。

Part C：Matching Improvement

步驟六：找出一組 alternating path，方式如下：

- 以發生 breakthrough 之點為起始點，找出標記為此點之其他點；
- 在這些點中，再進一步找出標記為這些點的點；
- 遞迴執行上步驟，直到找到標記為(*)之點；

這個過成中經過的點及連結，稱為一組 alternating path，以 P 表示之。

步驟七：經由符合下列條件連結找出一組新的 matching：

- 屬於目前之 M，但不屬於 P；
- 屬於 P，但不屬於目前之 M；

移除所有標記，演算法跳回步驟三繼續進行。

Part D：Modification of the Partial Graph

步驟八：找出符合下列條件，且其權重值最低的連結：

- 以已標記且屬於 X 之點為起點者；
- 以未標記且屬於 Y 之點為終點者；
- 其目前的權重值不等於 0；

將此權重值稱為δ。

步驟九：

1. 將 X 中所有已標記之點的權重值加上δ；
2. 將 Y 中所有未標記之點的權重值減去δ；
3. 將所有”一端為已標記且屬於 X 之點，一端為未標記且屬於 Y 之點，且目前權重值不等於 0”的連結之權重值減去δ；
4. 將所有”一端為未標記且屬於 X 之點，一端為已標記且屬於 Y 之點，且目前權重值不等於 0”的連結之權重值加上δ；

在上述第 3 個部份中，將在圖形中增加至少一個權重值為 0 之連結。將這些連結加入 partial graph 中，除去所有標記，演算法跳回 Part B 步驟

三繼續進行。

以上為 Hungarian Algorithm 之詳細步驟。經由這個演算法，可快速且正確的找出一個權重值總合最小之加權多對多 Matching 組合。

# 第三章 軟體相似度之計算

由上述兩個章節的概念及技術為基礎，我們提出了一套軟體設計相似度計算方法。經由此演算法，可計算出兩個軟體系統設計上的相似度，從巨觀的角度來判斷這兩個系統間相似與否。而除了計算出此數值外，這個方法同時也可以微觀的角度，指出兩個軟體系統設計間結構相似的部份，以及兩個系統中差異較大之處。藉由這些資訊，將可提高系統開發人員分析及評估軟體設計的關鍵處。

由於現代的軟體系統是由許多元件(components)所構成，因此，在發展軟體設計相似度方法時，也需要能夠分析系統中各元件間的相似度。由於軟體系統設計中包含的元件數量可能非常龐大，若單純去計算各原件間的相似度，勢必會因為其複雜度而造成執行效率低落。因此，許多軟體相似度相關的研究乃使用如 Parser Tree[9][10]、Pattern Matching[12]、分析 Java Bytecode[1]等做法，來減少這個問題所造成的影響。而這些研究的應用，大多著眼於程式碼的抄襲以及版本控制之上，甚少利用於軟體設計的相似度。

### 3.1 Class Signature

在計算軟體設計相似度時，為了讓比對元件相似度更有效率，我們使用 Class Signature 來代表一元件的特徵。一個 Class Signature 是由一組 elements 所構成。每個 element 代表該元件的某項特徵點。經由比對這些特徵點，便能快速且正確的計算出兩元件的相似度。

而為了便於實作相似度計算，我們以 XML 來儲存每個元件的 Class Signature。透過 XML parser 操作，此種格式將能快速取出資料進行比對。這些 element 及其定義如下：

- access_flags，用以區別此原件、method、或 field 的操作限制。共有三大類：
  - Class 或 interface 使用之 access flags 及其在 XML 文件中表示用的值如表 3.1 所示：

表 3.1 Class 與 interface 之 access flags

| Flag Name | Value |
|---|---|
| ACC_PUBLIC | 0x0001 |
| ACC_FINAL | 0x0010 |
| ACC_SUPER | 0x0020 |
| ACC_INTERFACE | 0x0200 |
| ACC_ABSTRACT | 0x0400 |

- Field 使用之 access flags 及其在 XML 文件中表示用的值如表 3.2 所示：

表 3.2 field 之 access flags

| Flag Name | Value |
|---|---|
| ACC_PUBLIC | 0x0001 |
| ACC_PRIVATE | 0x0002 |
| ACC_PROTECTED | 0x0004 |
| ACC_STATIC | 0x0008 |
| ACC_FINAL | 0x0010 |
| ACC_VOLATILE | 0x0040 |
| ACC_TRANSIENT | 0x0080 |

- Method 使用之 access flags 及其在 XML 文件中表示用的值如表 3.3 所示：

表 3.3 method 之 access flags

| Flag Name | Value |
|---|---|
| ACC_PUBLIC | 0x0001 |
| ACC_PRIVATE | 0x0002 |
| ACC_PROTECTED | 0x0004 |
| ACC_STATIC | 0x0008 |
| ACC_FINAL | 0x0010 |
| ACC_SYNCHRONIZED | 0x0020 |
| ACC_NATIVE | 0x0100 |
| ACC_ABSTRACT | 0x0400 |
| ACC_STRICT | 0x0800 |

- FieldType 值包含三種主要類型：
  - BasicType，為 Java 中定義之基本型態(primary types)，共有九種：
    - B：Java 中定義之 byte 型態；
    - C：Java 中定義的 char 型態；
    - D：Java 中定義的 double 型態；
    - F：Java 中定義的 float 型態；
    - I：Java 中定義的 int 型態；
    - J：Java 中定義的 long 型態；S：Java 中定義的 short 型態；
    - Z：Java 中定義的 boolean 型態；
  - [：表示此變數為一陣列型式。
  - ArrayType，Java 中定義的陣列。
    - [ field type
  - ObjectType
    - L<classname>，<classname>為一個 class 之完整名(full name)

- super_class，用以標示出該 class 或 interface 之 super class，在 Java 中，class 必需遵守單一繼承的規定，而 interface 可作多重繼承。

  以下為其 DTD 之定義：

```
<!ELEMENT SuperClass (#PCDATA)>
```

- interfaces，此元件所實作(implement)之 interface，在 Java 中，一個 class 可實作一個以上之 interface。以下為其 DTD 之定義：

```
<!ELEMENT Interface (#PCDATA)>
```

- fields_count，此元件中包含的變數個數。
- fields_info[fields_count]，此元件中所有變數之資訊，包含操作限制、變數類型等。

  以下為變數部份在 DTD 中之定義：

```
<!ELEMENT Field (Count, FieldInfo+)>
<!ELEMENT Count (#PCDATA)>
<!ELEMENT FieldInfo (FieldAccessFlags, FieldType)>
```

- methods_count，此元件中包含的 method 個數。
- methods_info[methods_count]，此元件中所有 method 之資訊，包含操作限制、參數個數、類型以及傳回值等。

  以下為 method 部份在 DTD 中之定義：

```
<!ELEMENT Method (Count, MethodInfo+)>
<!ELEMENT MethodInfo (MethodAccessFlags, Descriptor)>
```

- Library components used，此元件中使用到的外部元件。

  其 DTD 定義為：

```
<!ELEMENT Library (#PCDATA)>
```

圖 3.1 為一個完整之 Java Class 原始程式碼範例，而圖 3.2 則為該 Java Class 對應之 Class Signature 的 XML 檔案內容。

```
import java.io.File;
import java.util.Hashtable;
public class ExampleFileView extends FileView {
    private Hashtable icons = new Hashtable(5);
    private Hashtable fileDescriptions = new Hashtable(5);
    private Hashtable typeDescriptions = new Hashtable(5);
    public String getName(File f) {
    return null;
    }
    public void putDescription(File f, String fileDescription) {
    fileDescriptions.put(fileDescription, f);
    }
    public String getDescription(File f) {
    return (String) fileDescriptions.get(f);
    }
    public void putTypeDescription(File f, String typeDescription) {
    putTypeDescription(getExtension(f), typeDescription);
    }
    public String getTypeDescription(File f) {
    return (String) typeDescriptions.get(getExtension(f));
    }
    public Boolean isTraversable(File f) {
    if(f.isDirectory()) return Boolean.TRUE;
    else return Boolean.FALSE;
    }
}
```

**圖** 3.1 Java Class **範例原始碼**

```
<Class>
    <name>ExampleFileView</name>
    <SuperClass>FileView</SuperClass>
    <Field>
        <Count>3</Count>
        <FieldInfo>
            <AccessFlag>0x0002</AccessFlag>
            <FieldType>Ljava/util/Hashtable</FieldType>
        </FieldInfo>
        <FieldInfo>
            <AccessFlag>0x0002</AccessFlag>
            <FieldType>Ljava/util/Hashtable</FieldType>
        </FieldInfo>
        <FieldInfo>
            <AccessFlag>0x0002</AccessFlag>
            <FieldType>Ljava/util/Hashtable</FieldType>
        </FieldInfo>
    </Field>
    <Method>
        <Count>7</Count>
        <MethodInfo>
            <AccessFlag>0x0001</AccessFlag>

    <Descriptor>(Ljava/io/File)Ljava/lang/String</Descriptor>
        </MethodInfo>
        <MethodInfo>
            <AccessFlag>0x0001</AccessFlag>
            <Descriptor>(Ljava/io/File,
Ljava/lang/String)</Descriptor>
        </MethodInfo>
        <MethodInfo>
            <AccessFlag>0x0001</AccessFlag>

    <Descriptor>(Ljava/io/File)Ljava/lang/String</Descriptor>
        </MethodInfo>
        <MethodInfo>
            <AccessFlag>0x0001</AccessFlag>

    <Descriptor>(Ljava/io/File)Ljava/lang/String</Descriptor>
        </MethodInfo>
        <MethodInfo>
            <AccessFlag>0x0001</AccessFlag>

    <Descriptor>(Ljava/io/File)Ljava/lang/String</Descriptor>
        </MethodInfo>
        <MethodInfo>
            <AccessFlag>0x0001</AccessFlag>
            <Descriptor>(Ljava/io/File)B</Descriptor>
        </MethodInfo>
    </Method>
    <Library>java/io/File</Library>
    <Library>java/util/Hashtable</Library>
</Class>
```

**圖** 3.2 Java Class Class Signature XML **範例**

### 3.2 Software Association Network

**在物件導向的設計裡，一個軟體系統是由許多元件所組成，每個元件負責**

提供所需服務。以下為本研究中對於軟體系統符號定義的說明：

一軟體系統 $P$ 由數個元件所構成，包含：$x_1, x_2, \cdots x_n$，則 $P$ 可以集合$\{x_1, x_2, \cdots x_n\}$表示之，其中 $x_i$ 屬於 P，$1 \leqq i \leqq n$。

一軟體系統 $Q$ 由數個元件所構成，包含：$y_1, y_2, \cdots y_m$，，則 $Q$ 可以集合$\{y_1, y_2, \cdots y_m\}$表示之，其中 $y_j$ 屬於 Q，$1 \leqq j \leqq m$。

這些軟體設計中，元件間的關係可用一個網路結構描述之，我們稱之為 Software Association Network。元件間參數傳遞、繼承、傳回值、實作等交互動作皆簡簡化稱之為 association。如圖 3.3 所示，每個 association 連結兩個元件，這些 association 在圖形中以箭頭表示，起始之元件為 start，結述者為 end。沒有任何 Association 連結之元件稱之為孤立元件(isolated component)。

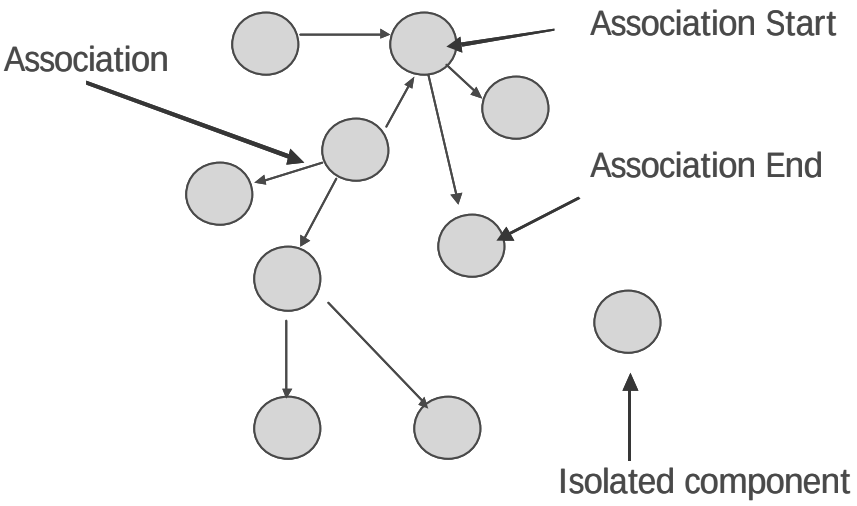
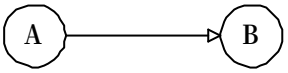



圖 3.3 Software Association Network 示意圖

透過這個網路架構，我們可以概略描繪出一個軟體系統的設計結構，以及其內部各元件間的關聯。但若要完整表現出軟體設計時的架構，單靠這簡單的圖形仍嫌不足，在接下來的小節中，我們進一步提出一套有效的描述方法，充份表達軟體設計的概念。在一個軟體系統設計中，元件彼此間的關係包含了繼承、實作、參數傳遞等許多種類，為了清楚表達這些關係，我們將 association 分為四類：

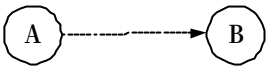
- 繼承(Extends)：此為 Java 中定義的繼承關係。在 Java 中，Class 必需遵守單一繼承的規定。圖 3.4 為一繼承關係示意圖，A、B 為兩元件，B 繼承 A。

- **實作**(Implements)：Java **中定義的實作關係。一個** Java Class **可以實作一個或多個** interface。**圖** 3.5 **為一實作關係示意圖，** A、B **為兩元件，** B **實作** A。
- **包含**(Contain)：Java **中定義的包含關係；在** Java **裡，** Class **經由** new **的動作，來達到包含這項關係；圖** 3.6 **為一包含關係示意圖，** A、B **為兩元件，** B **包含** A。
- **使用**(Use)：**在** Java **中的參數傳遞以及傳回值，皆為使用關係；圖** 3.7 **為一使用關係示意圖，** A、B **為兩元件，** B **使用** A。



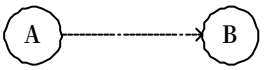

A Extend B

**圖** 3.4 **繼承關系示意圖**



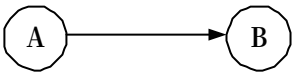

A Implement B

**圖** 3.5 **實作關系示意圖**




A Use B

3.6 **使用關系示意圖**




A Contain B

**圖** 3.7 **包含關系示意圖**

**一個元件所連結到的** associations **數量稱之為** Dependency。**依照** association **種類以及** start、end **之關係分類，共有下列八種** Dependency：

- $E_S(x_i)$：**繼承元件** $x_i$ **的** association **數量；**
- $I_S(x_i)$：**實作元件** $x_i$ **的** association **數量；**
- $C_S(x_i)$：**包含元件** $x_i$ **的** association **數量；**
- $U_S(x_i)$：**使用元件** $x_i$ **的** association **數量；**$E_e(x_i)$：**元件** $x_i$ **繼承其他元件的** association **數量；**$I_e(x_i)$：**元件** $x_i$ **實作其他元件的** association **數量；**
- $C_e(x_i)$：**元件** $x_i$ **包含其他元件的** association **數量；**$U_e(x_i)$：**元件** $x_i$ **使用其他元件的** association **數量。**

**在一個軟體系統設計** $P$ **中，共存在有四種** Average Dependency：

- $E_P$：**系統中各元件繼承關係的平均數**

$$E_P = \frac{\sum_{i=1}^{n} E_s(x_i)}{n}$$

- $I_P$：系統中各元件實作關係的平均數

$$I_P = \frac{\sum_{i=1}^{n} I_s(x_i)}{n}$$

- $C_P$：系統中各元件包含關係的平均數

$$C_P = \frac{\sum_{i=1}^{n} C_s(x_i)}{n}$$

- $U_P$：系統中各元件使用關係的平均數

$$U_P = \frac{\sum_{i=1}^{n} U_s(x_i)}{n}$$

一個元件的 Dependency 數量越多，當它出現問題或需要修改、更新時，整個軟體系統受到的影響可能性也越高。因此，這些元件在軟體系統的架構中佔有關鍵地位。

根據 association 箭頭的方向，可以將系統中 Dependency 數量較多的元件分為兩類：

- Sink：在一個軟體系統中受到許多其他元件影響的部份，當系統需要作出修改或調整時，這些元件很可能因為其他部份的更動而受到影響；
- Source：在一個軟體系統中，影響許多其他元件的部份，當這些元件被修改或調整時，將會影響到許多其他的元件。

這兩者可能代表了軟體設計上的特徵，但這種分類方式太過粗略，由此著手比較將會因為這些元件數量過多而造成效率不佳。因此，我們更進一步根據 Dependency 的種類，將 Sink 和 Source 又各分為四類，定義如下：

- Extend Sink：若 $E_e(x_i) \geqq E_P$ 則 $x_i$ 為一 Extend Sink，由於 Java 中 Class 單一繼承的設定，Extend Sink 必定為一個 Interface；
- Implements Sink：若 $I_e(x_i) \geqq I_P$ 則 $x_i$ 為一 Implements Sink；
- Contain Sink：若 $C_e(x_i) \geqq C_P$ 則 $x_i$ 為一 Contain Sink；
- Use Sink：若 $U_e(x_i) \geqq U_P$ 則 $x_i$ 為一 Use Sink；
- Extend Source：若 $E_S(x_i) \geqq E_P$ 則 $x_i$ 為一 Extend Source；
- Implements Source：若 $I_S(x_i) \geqq I_P$ 則 $x_i$ 為一 Implements Source；
- Contain Source：若 $C_S(x_i) \geqq C_P$ 則 $x_i$ 為一 Contain Source；
- Use Source：若 $U_S(x_i) \geqq U_P$ 則 $x_i$ 為一 Use Source；

**Lemma 3.1** 一軟體系統設計中必存在一個或一個以上任意類型之 Sink。

Proof：假設存在一軟體系統 *P*，其中不包含任何 Sink，根據 Sink 之定義，則可推論系統 *P* 中所有元件的任意種類 dependency 數量皆小於該種類之 average dependency；此種情況不可能存在，故得證。

**Lemma 3.2** 一軟體系統中必存在一個或一個以上任意類型之 Source

Proof：假設存在一軟體系統 *P*，其中不包含任何 Source，根據 Source 的定義，則可推論系統 *P* 中所有元件的任意種類 dependency 數量皆小於該種類之 average dependency；此種情況不可能存在，故得證。

### 3.3 Software Design Similarity

上一節定義的 Sink 及 Source，可以協助找出軟體系統內較特別的元件，由這些元件作為相似度比對的切入點。元件間的相似度定義如下：

$$C_{sim}(x,y) = \frac{|\{a_i \mid (a_i,b_j) \in Cs\}| + |\{b_j \mid (a_i,b_j) \in Cs\}|}{n+m}$$

令 *x*、*y* 兩元件，*x* 的 class signature 中有 n 個 element，分別為 $a_1$、$a_2$…、$a_n$，*y* 的 class signature 中有 m 個 element，分別為 $b_1$、$b_2$…、$b_m$。令 *Cs* 為兩者中相等之 element 構成的集合，則 *x*、*y* 之相似度 $C_{sim}(x,y)$為。

由於 $|\{a_i \mid (a_i,b_j)\in C_s\}|\le n$ 且 $|\{b_j \mid (a_i,b_j)\in Cs\}|\le m$，故 $0\leqq C_{sim}(x,y)\leqq 1$。

若 *P*、*Q* 為兩軟體系統，當兩元件夠相似時，它們便被稱為一個 Match。至於何謂夠相似，則必須根據下面我們所發展的演算法來決定之。兩系統間所有 Match 構成的集合為 *M*，其定義如下：

$$M=\{(x,y)\mid x\in P, y\in Q\}$$
$$M_P=\{x\mid (x,y)\in M\}$$
$$M_Q=\{y\mid (x,y)\in M\}$$

而 *M'* 則為 *P*、*Q* 中 non-match components 之 maximum matching 結果，其產生方式由下面的演算法定義。*P*、*Q* 兩軟體系統設計間的相似度由一組向量組成：

$$\left(\frac{\sum_{(x,y)\in M} C_{sim}(x,y)}{|M|},\frac{\sum_{(a,b)\in M'} C_{sim}(a,b)}{|M'|}\right)$$

兩系統之相似度值為：

$$S(P,Q)=\sqrt{\frac{\left(\frac{\sum_{(x,y)\in M} C_{sim}(x,y)}{|M|}\right)^2+\left(\frac{\sum_{(a,b)\in M'} C_{sim}(a,b)}{|M'|}\right)^2}{2}} \qquad \textbf{(3.1)}$$

緊接著我們提出一個演算法，來決定兩元件間是否夠相似，以及有效計算兩軟體系統設計上的相似度。此演算法分為四個步驟：

- Preprocessing：
  - 輸入 *P*、*Q* 兩軟體系統設計之 XMI 文件，並以 XML Parser 處理之。
  - 取出其中包含的元件，產生這些元件對應之 Class Signature 文件。
  - 計算每個元件之各種 Dependency 數量，以及系統的各類 Average Dependency 資訊。
  - 使用者可決定何元件為關鍵元件並將之紀錄以便於後續進一步分析用。
- Refining：
  1. 產生一變數 Level，其初始值為 1。

2. 輸入兩軟體系統 XMI 文件之 DOM 結構，以及對應之 Class Signature 文件。

3. 根據 Lemma 3.1 及 Lemma 3.2，一軟體系統中必存在有一個或一個以上任意類型之 Sink 及 Source，因此，由兩系統目前的結構中找出、儲存並移除 Sink 及 Source，其原則如下：

   - 根據 Dependency 判斷元件是否為 Sink 或 Source，以 Dependency 與 Average Dependency 差距最大者為首選；
   - 以此元件在 XMI 文件中之 ID 為 key 值，將其 class signature、level 以及 Type(何種類形之 Sink 或是 Source)存放至 Hashtable 中，如圖 3.8 所示；

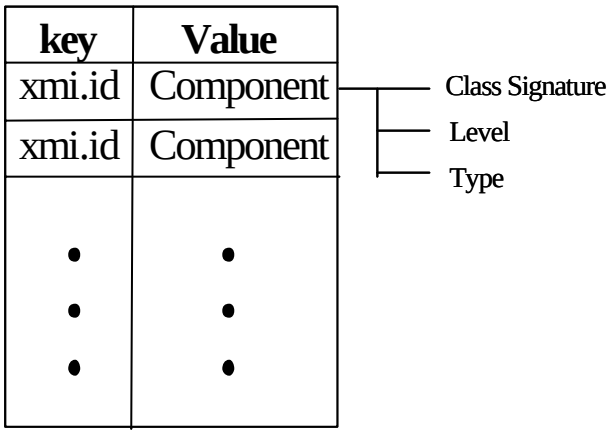



圖 3.8 元件存放之資料節構示意圖

   - 將此 Source 或 Sink 元件移出原系統結構；

4. 若此時系統中存在有獨立元件，亦將其 class signature、level 以及 Type(Isolated Component)儲存於該資料結構中；

5. Level = Level + 1；

6. 重複 2~5，直到其中一個系統中無任何元件存在；另一個尚存有元件的系統中之所餘元件則全數放入獨立元件資料結構中。

- Matching：

1. 重設變數 Level 值為 1；

2. 根據目前的 Level 值，取出對應之 Sink、Source 以及獨立元件；

3. 以兩軟體系統中同 Level 元件建立出 bipartite graph 如下：
   - 同類型之 Sink 或 Source 元件歸於同一 bipartite graph，例如全為 level 2 之 Extend Source 的 bipartite graph；
   - 計算出該 bipartite graph 中任兩組元件間所有組合之相似度值；
   - 去除相似度值小於 k 值的組合；
   - 使用(1-相似度值)為權重，建立出此 bipartite graph 的權重線 (weighted edges)；
4. 使用 Hungarian Algorithm 找出各 bipartite graph 中的 maximum matching；
5. 若存在無法被 matched 到的元件時，將其 Level 值加一；
6. Level = Level + 1；
7. 重複 2~4，直到處理完所有元件；
8. 最後剩下的剩餘元件，同步驟 3 方法，建立剩餘元件之 bipartite graph，並使用 Hungarian Algorithm 找出相似度大於 k 之 Match；
9. 此時若還有元件不屬於任何 Match，稱之為 non-match component，以兩系統中之 non-match component 建立 bipartite graph，透過 Hungarian Algorithm 找出其中相似度最高之 maximum matching，此即為 *M'*；
10. 若考慮關鍵元件時：
    - 以兩系統中的關鍵元件建立一 bipartite graph，並找出其中之 maximum matching。
11. 輸出包含：
    - 所有元件之 Matching 結果；
    - 關鍵元件時之比較結果。

- Similarity Calculation：根據上一步驟產生之 matching 所構成之集合 M，利用式子 3.1 計算出兩軟體系統設計相似度之值。

以下透過一個簡單的例子以圖表解說演算法流程：

假設存在兩軟體系統 *P*、*Q*。在 Preprocessing 中可以得到兩系統中各元件之資訊。之後在 Refining 中找出兩軟體系統各 Level 的 Sink、Source 及 Isolated Components。流程如圖 3.9 所示，此處為簡化範例以便於瞭解，所有 dependency 皆不分類。

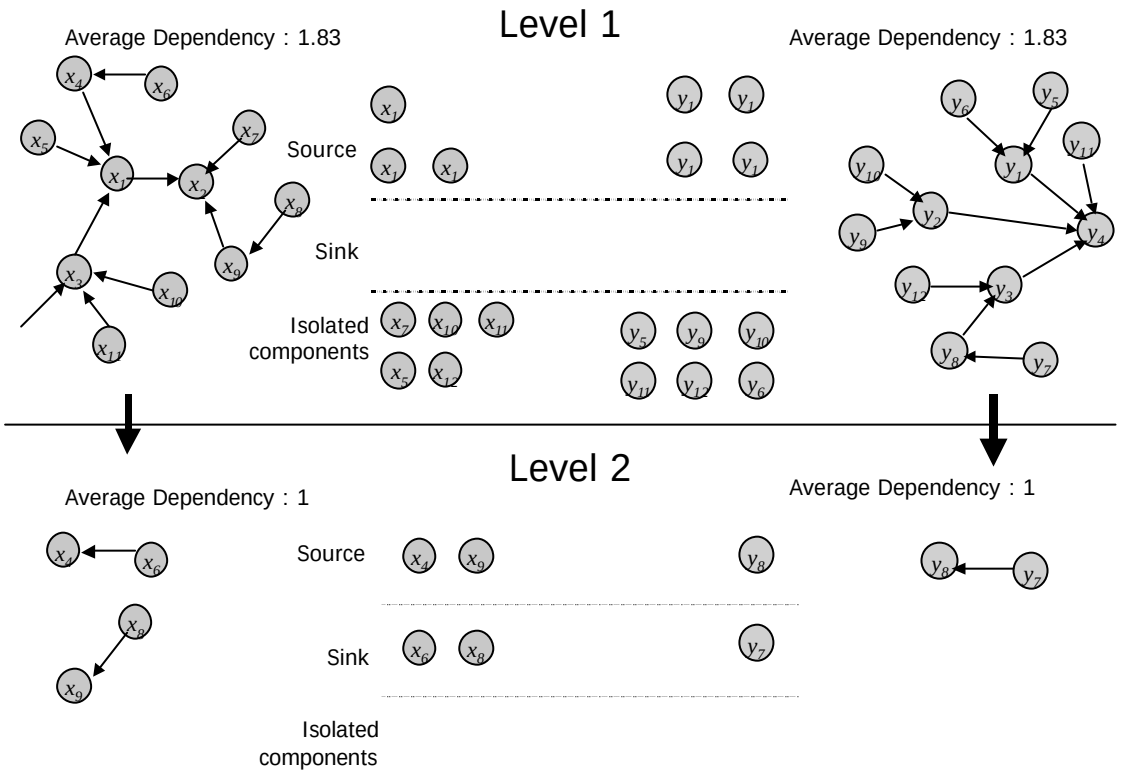



圖 3.9 Refining 步驟示意圖

Matching 過程以兩系統中 Level 1 的 Source 為例，軟體系統 *P* 中共有 $x_1$、$x_2$、$x_3$ 三個元件為 Level 1 之 Source，而軟體系統 *Q* 中有 $y_1$、$y_2$、$y_3$ 以及 $y_4$ 共四個 Level 1 之 Source 元件。如圖 3.10 所示。

其 Matching 過成如下：

1. 假設計算出所有元件間之相似度，如表 3.4 所示；

表 3.4 level 1 中 Source 元件之相似度列表

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| $y_1$ | 80 | 67 | 6 |

| | | | |
|---|---|---|---|
| $y_2$ | 65 | 44 | 33 |
| $y_3$ | 57 | 28 | 15 |
| $y_4$ | 68 | 58 | 70 |

2. **根據演算法的常數** k **值，刪除其中元件相似度小於** k **值的部份，以確保這些相似度過低的組合不會對** matching **過程造成影響，假設此處** k **之值為** 66%**，其結果如表** 3.5 **所示 ；**

**表** 3.5 **剔除相似度小於** k **之組合後之列表**

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| $y_1$ | 80 | 67 | |
| $y_2$ | | | |
| $y_3$ | | | |
| $y_4$ | 68 | | 70 |

3. **以**(1-Csim($x_i$,$y_j$))**作為兩元件間** Association **之權重；**

4. **根據** Hungarian Algorithm**：**

   i. **建立** cost matrix**，如表** 3.6**；**

**表** 3.6 **起始狀態之** cost matrix

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| $y_1$ | 20 | 33 | |
| $y_2$ | | | |
| $y_3$ | | | |
| $y_4$ | 32 | | 30 |

   ii. **以完全沒有任何** matching **之情況為起始狀態，如圖** 3.10**；**

   iii. **建立** first revised cost matrix**，如表** 3.7**：**

**表** 3.7 first revised cost matrix

| | | | |
|---|---|---|---|
| | 0 | 12 | 0 |

| | $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| 20 $y_1$ | 0 | 1 | |
| $y_2$ | 0 | | |
| $y_3$ | 0 | | |
| 30 $y_4$ | 2 | | 0 |

iv. **建立** initial partial graph**，如圖** 3.11**；**

v. **找出一組** maximum matching**，如圖** 3.12 **所示；**

vi. **計算這組** maximum matching **之** second revised cost matrix**，如表** 3.8**；**

**表** 3.8 second revised cost matrix

| | 0<br>$x_1$ | 12<br>$x_2$ | 0<br>$x_3$ |
|---|---|---|---|
| 20 $y_1$ | 0 | 1 | |
| 30 $y_4$ | 2 | | 0 |

vii. **重複上述步驟，直到達成停止條件；**

5. **最後結果如圖** 3.13 **所示；**

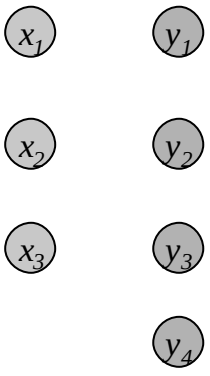



**圖** 3.10 **起始狀態示意圖**

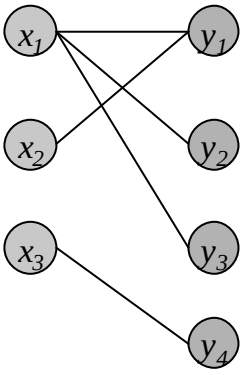



**圖** 3.11 initial partial graph

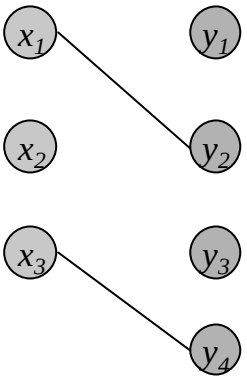



圖 3.12 maximum matching

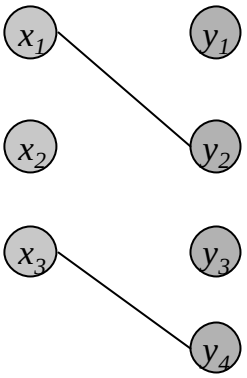



圖 3.13 matching 結果

6. 將沒有 Matching 到之元件放入下一個 level 的初始元件中, 得到圖 3.14 之結果。

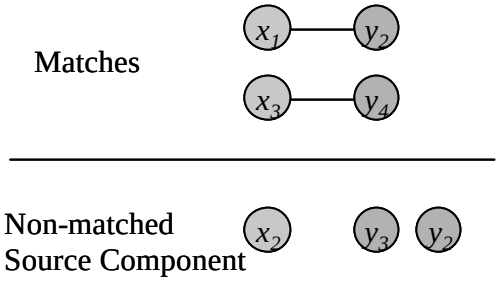



圖 3.14 level 1 matching 結果

Sink 及獨立元件間的 matching 方式亦同。經由這個演算法, 可以找出各 level 中 Sink、Source 以及獨立元件間的 match, 最後留下無法找到任何 matching 的元件即為兩軟體系統中差異性最高的部份。

透過這個演算法, 除了計算出兩軟體系統設計上的相似度之外, 尚可取得系統中相似的部份、差異性較大的部份、以及幾乎完全不同的部份。較之單純計算出相似度數質的做法, 此演算法對軟體開發及管理的人員來說, 能更有效的幫助瞭解兩軟體系統在設計上的異同, 以進行更進一步的工作。

下一個章節裡, 本文將詳細說明此演算法於實際案例中之操作方式, 以及各種應用。

# 第四章 演算法之實作與實驗

根據第三章的定義，在此章節中，本文將以開放原始碼(open source)之軟體：httpunit 的兩個版本作為例子，說明軟體設計相似度演算法之運作方式，以及最後產生的結果。範例中使用的軟體是 httpunit 1.3 以及 httpunit 1.4 版，使用此軟體這兩個版本的原因有以下兩點：

- 為開發過成穩定可靠之開放原始碼軟體
- 具有足夠的結構性，足以表現出完整的軟體設計架構

經由這個範例，本文詳細介紹演算法中各步驟的目的及產出物，並分析在最後的結果中，軟體開發、管理人員所能得到的幫助。

## 4.1 軟體相似度計算結果數據之分析

在實驗的一開始，我們使用 Rational Rose，產生出 httpunit 1.3 及 1.4 對應之 XMI 文件，透過演算法分析此兩份文件，取得相關資訊，並產生對應於各步驟之結果，此處我們使用的 k 值為 0.7。各步驟之詳細產出結果請參照附錄。在完成演算法的各步驟之後，根據這些結果進行對這兩個軟體的分析。

首先，在經過相似度計算之後，httpunit 1.3 及 httpunit 1.4 之軟體設計相似度值為 82%；match 之部份為 93%，non-match 的部份則為 54%。兩軟體在 match 的部份相似度極高，由此可以推斷，這次版本更新中部份原本的設計仍繼續被延用。而 non-match 的部份也仍有 54%之相似度，此現象可能是由於某些原有的元件遭到中大更動之故，更詳細的情況需經由分析 Match 結果方可得知。

接下來，我們觀察演算法產出之 Match 以及元件間之相似度值。圖 4.1 為兩軟體 Source 部份 Match 之長條圖，圖 4.2 為 Sink 部份之 Match，而圖 4.3 則為剩餘元件 matching 後之結果。

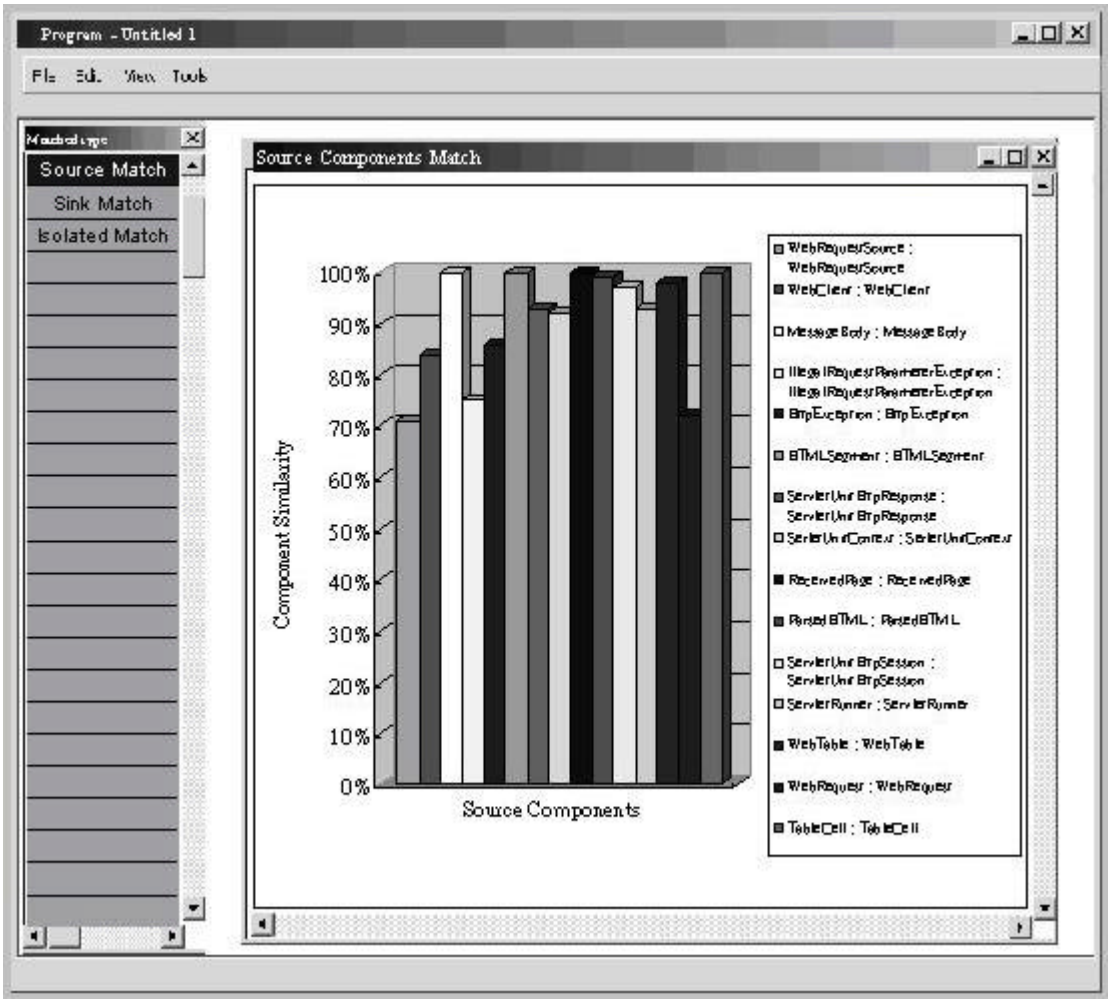



**圖** 4.1 Source **元件之** Match

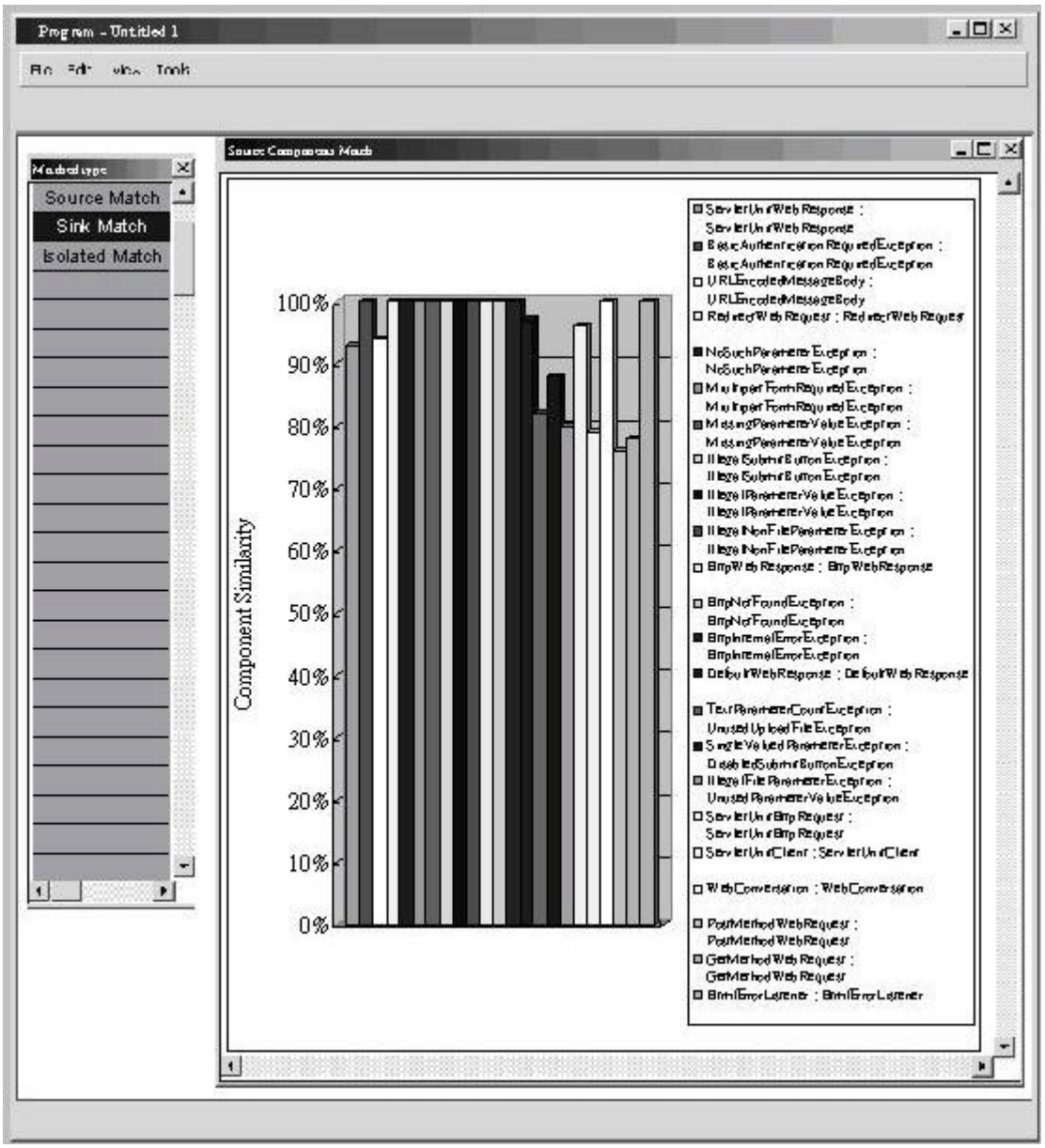



**圖** 4.2 Sink **元件之** Match

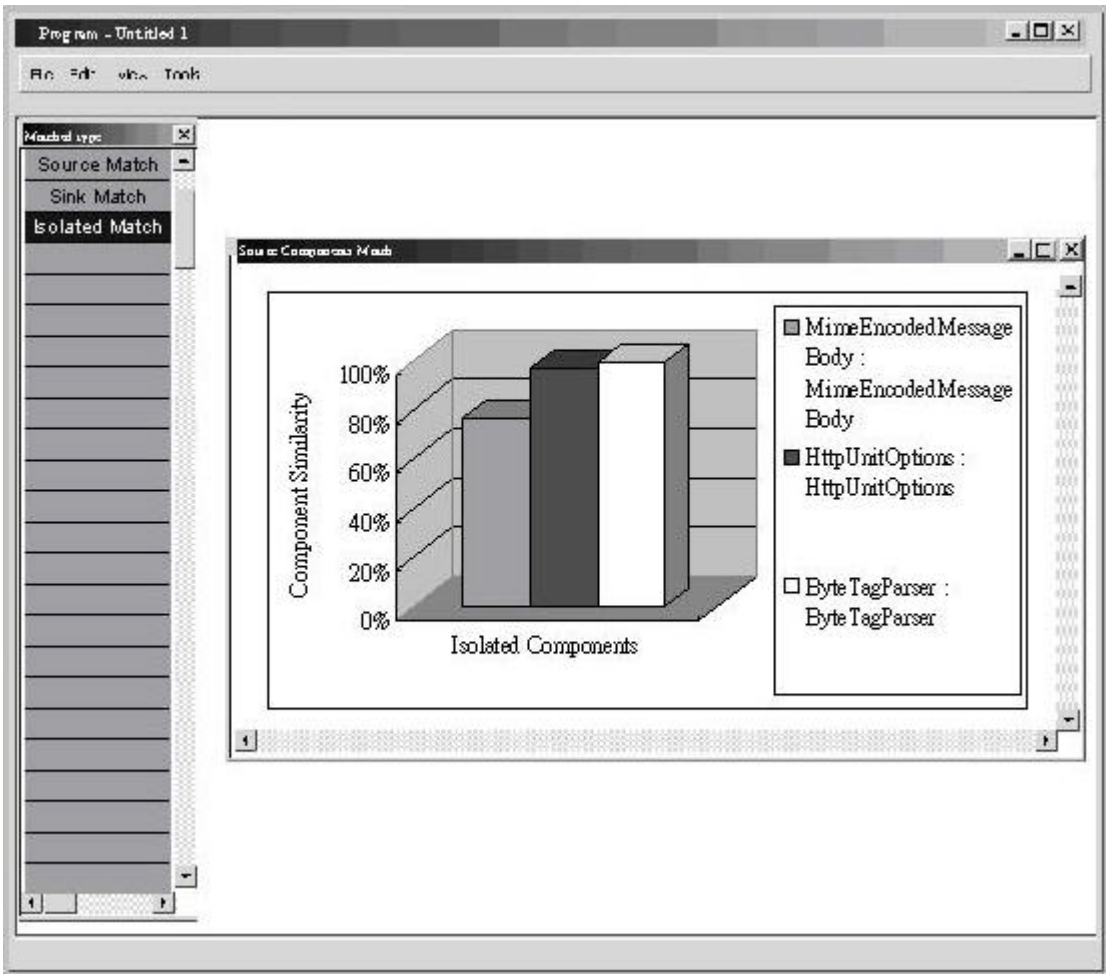



**圖 4.3 剩餘元件之 Match**

**由這些 Matching 的結果，我們可以看出，在 Source 元件的部份，所有 1.4 版中的 Match 元件，皆與 1.3 版中同名之元件相對應。而在 Sink 的 Matching 中，也僅有三組 Match 不屬於上述情況，分別為：**

- HttpInternalErrorException **與** UnusedUploadFileException；
- SingleValuedParameterException **與** DisabledSubmitButtonException；
- IllegalFileParameterException **與** UnusedParameterValueException。

**其共同點在於皆為** Exception **元件，且六個元件皆繼承於** com.meterware.httpunit.IllegalRequestParameterException**，因此在演算法的過程中被判定為** Match。

**由上述** Match **之結果可以推論，原本的設計的許多部份，在此次改版的過程中得到保留。而目前發現在設計上改變較明顯的部份有四：**

- MimeEncodedMessageBody 由 Source 轉為 Sink；
- WebFrame 則由 Use Source 轉為 Contain Source；
- HttpUnitOption 由 level 1 Use Sink 轉為 level 2 Use Sink；
- ByteTagParser 則由 level 1 Use Sink 轉變成為 level 1 isolated component。

若與演算法各步驟產出的結果一併觀察，則可發現，MimeEncodedMessageBody 因為與其它元件間互動關係的改變，而使得它在整個軟體中的地位由 Source 轉為 Sink；而關於 WebFrame、HttpUnitOption 及 ByteTagParser 這三個元件，不論是在元件本身，或是與其它元件的互動關係，都沒有因版本更改而變動，其角色之轉換是因為系統 Average Dependency 數值改變之故，因此可將此三元件視為完全未更動的部份。

MessageBodyWebRequest、InvocationContext、WebForm 及 WebLink 則是保留下來的元件中較特別的部份，以軟體元件間互動的情況而言，它們的地位並沒有出現大更動，但就單一元件的角度來看，其內部的修改則非常明顯，因此雖兩個版本的系統中皆包含同名之元件，卻因元件相似度過低而無法 match。此更動可在 httpunit 1.4 之 release note 中得到 MessageBodyWebRequest 部份的解釋，此元件中許多部份的 access_flag 遭到改變，因而使兩個版本中同名元件間的 class signature 差異過大而無法 match。這些元件之相似度如表 4.1。

表 4.1 內部更動明顯之元件列表

| httpunit 1.3 | httpunit 1.4 | 相似度 |
|---|---|---|
| MessageBodyWebRequest | MessageBodyWebRequest | 57% |
| WebLink | WebLink | 37% |
| WebForm | WebForm | 56% |
| InvocationContext | InvocationContext | 67% |

其餘無法 match 之元件皆為 1.4 版中新增的部份，包含了 FormControl 相關元件及 WebApplication 等，詳細列表請參照附錄 D。

## 4.2 關件元件之相似度分析

在進一步觀察 Preprocessing 及 Refining 產生的結果之後，我們將 dependency 數量大於 average dependency 三倍之元件定為關鍵元件。在兩個軟體系統中分別存在數個與其它元件互動頻繁的關鍵元件。在 httpunit 1.3 中的幾個關鍵元件為：

- Extend Source：MessageBody 及 IllegalRequestParameterException
- Implement Source：HTMLSegment
- Contain Source：ParsedHTML
- Use Source：WebTable、WebRequest、WebForm 以及 SubmitButton
- Contain Sink：ServletUnitHttpRequest 及 InvocationContext

而在 httpunit 1.4 中擁有類似地位的元件則為：

- Extend Source：MessageBody、IllegalRequestParameterException 及 FormControl
- Implement Source：HTMLSegment
- Contain Source：WebApplication 及 ParsedHTML
- Use Source：WebTable、WebRequest、WebForm、UploadFileSpec、ParameterProcessor 以及 SubmitButton
- Contain Sink：ServletUnitHttpRequest、ServletRunner 及 InvocationContextImpl

由於與其它元件擁有充份的互動關係，這些元件極可能是軟體設計上的關鍵之處，因此，我們將這些元件取出進行比較。圖 4.4 為此部份之比較圖。由圖中可以看出，除了因新增而無法對應的部份之外，這些元件間之相似度大多在 70% 以上。說明了在此次改版的動作中，並未針對這些在設計上處於關鍵位置的元件作出大幅度更動，而是在保留原本設計的前題下加入新的部份。

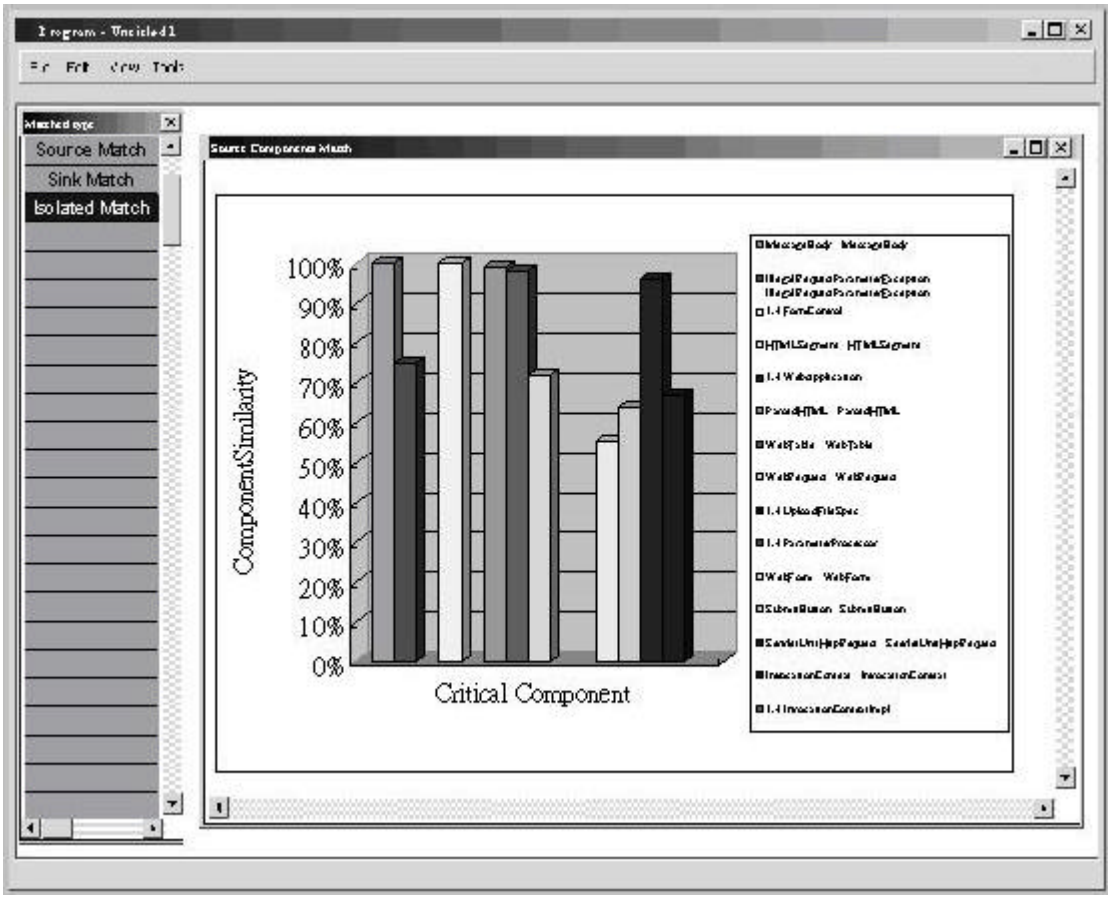



**圖 4.4 關鍵元鍵比較圖**

**除了單一元件間的比較之外，對於相關聯的數個元件，亦可經由以一個元件為中心的** Association Network **來進行分析、比較。以兩系中的** WebForm **為例。以其為中心之** Association Network **如圖** 4.5 **及** 4.6 **所示。**

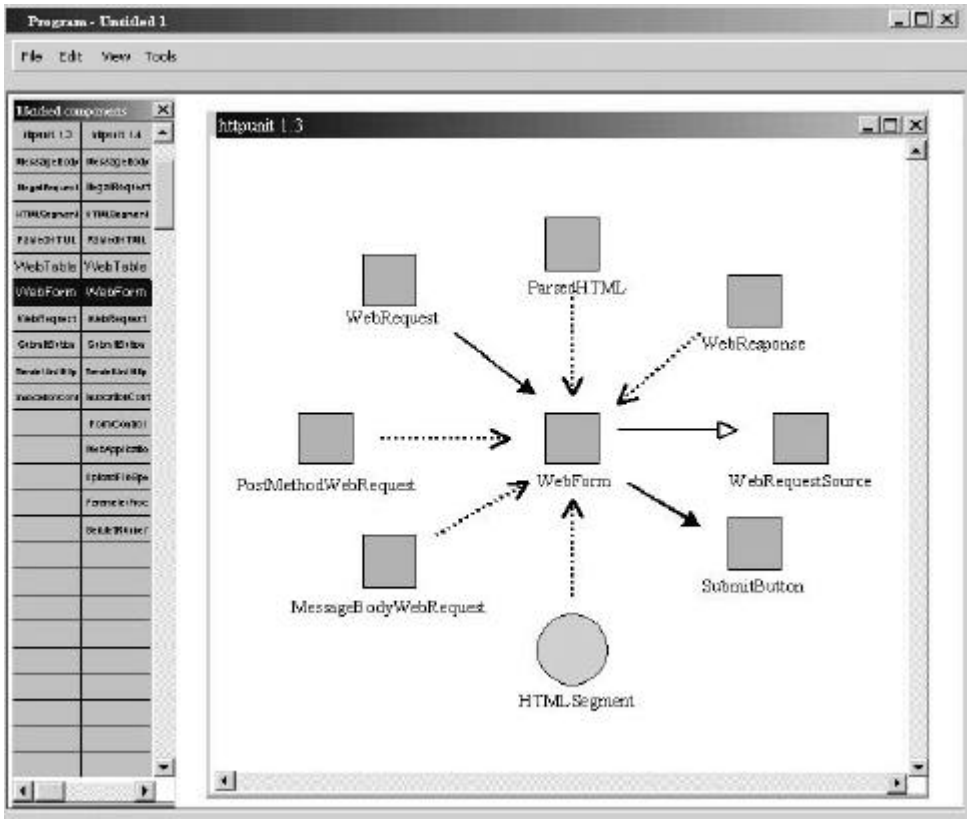



**圖** 4.5 httpunit 1.3 WebForm **元件示意圖**

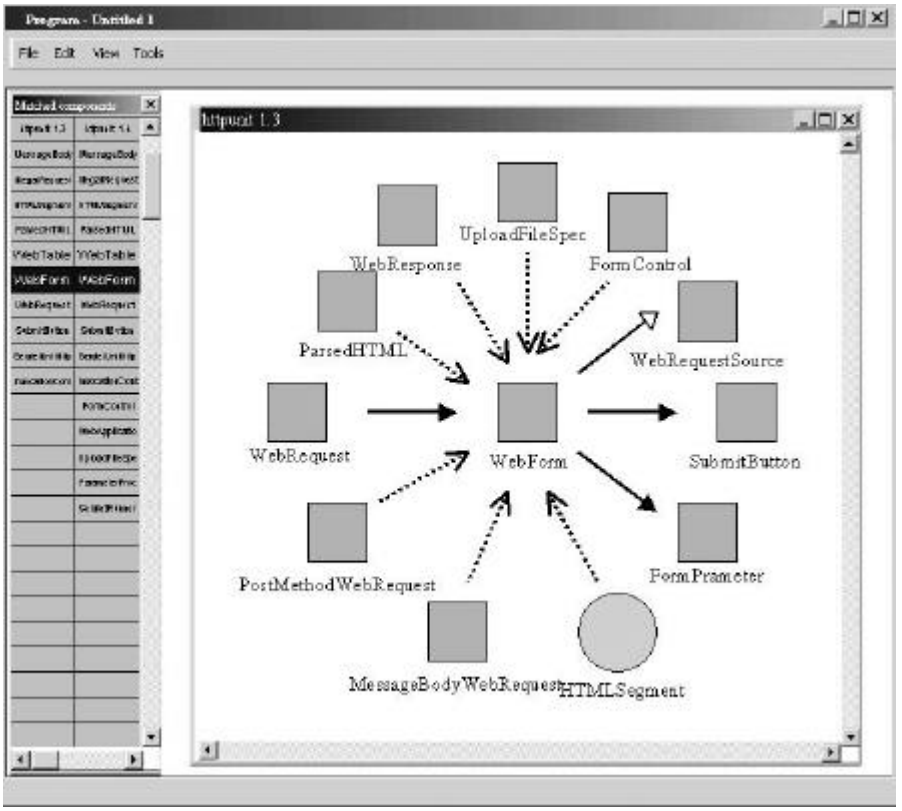



**圖** 4.6 httpunit 1.6 WebForm **元件示意圖**

**透過這兩個圖，我們可以看出，在這一**

組 Match 中，新增了數個元件，使得 1.4 版中的 WebForm 擁有更多與其他元件的 Association，但除此之外，此部份原有之設計並未更動。圖 4.7 為兩系統中以 WebForm 為中心之相關元件相似度比較圖。

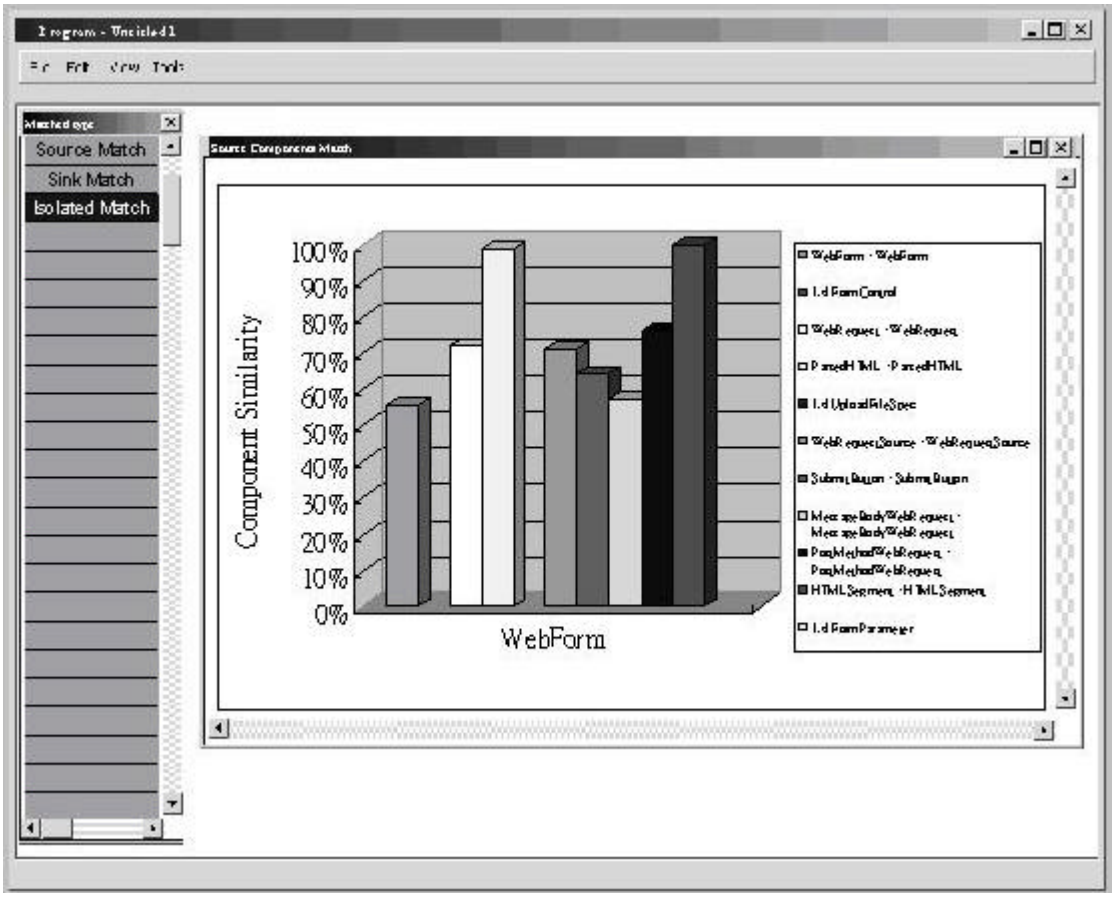



圖4.7 以WebForm為中心之相關元件相似度比較圖

由此圖可以看出，在以 WebForm 為中心之相關元件中，除了新增而無法對應之元件外，其餘部份僅有兩組對應元件之相似度低於 60%。說明了在這兩個版本的 httpunit 中，即便是關鍵元件裡相似度較低的部份，在軟體設計上之地位亦無重大的改變。

最後，我們可以經由無法 Matching 的元件中得知，哪些部份的差異性最高，甚至是新加入的部份。

經過上述的分析之後可以看出，在 httpunit 1.3 中多數的設計未遭到大幅修改，即使是 bug 的修正，也大多為單一元件內部出現的問題而非整個軟體架構所造成。在 httpunit 1.4 中加入了多組新的設計，但也多為附加上之功能或一些參

數處理方式的改變，對於原有的設計結構沒有明顯的影響。因此，我們可推論，大多數在 httpunit 1.3 中使用的設計架構，具有一定程度以上的穩定性，在開發類似的軟體時可考慮參考這些設計方式；而原本使用 httpunit 1.3 的使用者在作版本更新時，由於系統的設計大致上都保留了下來，也僅需注意新增的功能是否符合需求，便能進行更新的動作，在相容性上應無重大的問題。

### 4.3 總結

在本章節中，本文以軟體實例說明演算法之運作過成以及最後產出之結果，並證明此演算法成效良好，除了幫助使用者瞭解軟體間設計上的相似度外，亦能指出其間之異同，對於軟體的開發、分析及管理有顯著的幫助。

# 第五章 結論與未來展望

## 5.1 結論

本文提出一套演算法，能針對兩個以 Java 技術開發之軟體系統，正確且有效率的計算其設計上的相似度，以便於使用者瞭解兩軟體系統在架構上之相似度。除了計算出相似度的值以外，演算法亦可經由系統架構中之關鍵點，指出其中相似的部份、以及差異性高的部份之所在，有效幫助軟體設計人員作出對此兩系統之分析與判斷，找出其間良好的設計方式，以及可能造成問題的設計架構。當系統管理人員需要作軟體的更新或更換時，亦可由這些資料來分析各軟體系統間的異同，或是不同版本的軟體間差異之所在，以評估這項更動對整體系統的影響。而對系統管理維護人員來說，則可以經由同類系統的比對，在需要對系統作出調整時，有效的找出良好的設計方式，以便於解決目前遭遇的問題。

而除了提出演算法之外，在第四章中，本文亦透過以實際軟體系統為分析對向的範例，來證明這個演算法在面對實際軟體時的運作過程、可行性、以及其在分析軟體系統上的幫助，說明了此作法於實際案例中的實用性。

雖然在本文中此演算法以 Java 技術開發之軟體系統為對象，但由於是經由 UML/XMI 中包含之架構資訊來作分析，因此我們相信，只要在其中針對 Java 技術而設計的部份稍作修改，這個演算法亦可有效的應用於 C++等其它物件導向語言開發之軟體系統。

## 5.2 未來展望

本文提出的演算法中，能計算出兩軟體系統之相似度，並找出其間的異同點，但若將此項技術應用於軟體工程的設計樣式(Design Pattern)領域之中，尚有不少發展空間：

- 除了元件間直接的 Association 之外，亦可進一步找出軟體系統中與多個其它元件間接產生關聯的部份，以提升相似度計算及特徵點分析的準確性；

- 經由軟體系統之 XMI 文件與已知 Pattern 之 XMI 文件的比對，找出系統中符合已知 Pattern 精神的部份；
- 經由軟體系統之 XMI 文件與已知 Pattern 之 XMI 文件的比對，找出系統中與 Pattern 類似之設計，並評估其修改為該 Pattern 對整體系統的影響及可行性；
- 根據同質性軟體系統的分析，找出解決特定問題使用之 Pattern。

而在顯示介面的部份，結合視覺化之精神，使得使用者能更方便的根據圖形找出所需的資訊，也是一個值得作更深入探討的課題。

# 參考文獻


[1]. B. Baker and U. Manber, "Deducing Similarities in Java Sources from Bytecodes," 1998 USENIX Technical Conference.

[2]. J. Bosch, P. Molin, M. Mattsson, PO Bengtsson, M. Fayad,"Object-Oriented Frameworks - Problems & Experiences," Object-Oriented Foundations of Framework Design, M. E. Fayad, D. C. Schmidt, R. E. Johnson (eds.), Wiley & Sons, 1999.

[3]. E. Chow and M. A. Heroux. "An object-oriented framework for block preconditioning," ACM Transactions on Mathematical Software, 1998.

[4]. Document Object Model(DOM) http://www.w3.org/DOM/ .

[5]. A. Dolan, J. Aldous, "Networks and Algorithms An Introductory Approach," Wiley, pp. 284 – 334, 1993.

[6]. M. L. Griss, J. Favaro, M. d'Alessandro, "Integrating feature modeling with the RSEB," Fifth International Conference on Software Reuse (Cat. No.98TB100203). IEEE Comput. Soc, Los Alamitos, CA, USA, xiii+388 pp.76-85, 1998.

[7]. httpunit, http://sourceforge.net/projects/httpunit/

[8]. IBM Rational Rose http://www.rational.com/products/rose/index.jsp

[9]. T. Kamiya, S. Kusumoto, , and K. Inoue, "A token-based code clone detection tool - ccfinder and its empirical evaluation," Techinal report, Osaka University, Department of Information and Computer Scineces, Inoue Laboratory, 2000.

[10]. T. Kamiya, S. Kusumoto, , and K. Inoue, "A token-based code clone detection technique and its evaluation," *Technical report of IEICE, SS2000-42-52*, vol. 100, no. 570, pp. 41– 48, 2001.

[11]. B. Karin; B Francois. "An Object-Oriented Framework-Based Architecture for

Decision Support Systems," CLEI Electronic Journal, Santiago - Chile, v. 1, n.1, pp. 1-20, June 1998.

[12]. K. Kontogiannis, M. Galler, R. DeMori, "Detecting Code Similarity Using Patterns," Third Workshop on AI and Software Engineering : Breaking the Toy Mold, pp. 68-73, August 1995.

[13]. B. B. Kristensen. "Object-Oriented Modeling with Roles," 2nd International Conference on Object Oriented Information Systems (OOIS' 95). 1995.

[14]. O. Nierstrasz, S. Gibbs, and D. Tsichritzis, "Component-oriented software development," Communications of the ACM, 35(9), pp.160-165, 1992.

[15]. OMG XML Metadata Interchange(XMI) Specification http://cgi.omg.org/docs/formal/02-01-01.pdf

[16]. Gerson Sunye, Alain Le Guennec, and Jean-Marc Jezequel, "Design Patterns Application in UML," ECOOP 2000, LNCS 1850, pp. 44-62, 2000

[17]. The Small Worlds, http://www.thesmallworlds.com

[18]. Carl A. Waldspurger and William E. Weihl, "An object-oriented framework for modular resource management," Fifth Workshop on Object-Orientation in Operating Systems (IWOOOS ' 96), October 1996.

# 附錄 A：Preprocessing 之結果列表

表 A-1 httpunit 1.3 元件 dependency 資料表

| Class | $C_S$ | $U_S$ | $E_S$ | $I_S$ | $C_e$ | $U_e$ | $E_e$ | $I_e$ |
|---|---|---|---|---|---|---|---|---|
| **Average Dependency** | **0.43** | **0.96** | **0.34** | **0.12** | **0.43** | **0.96** | **0.34** | **0.12** |
| WebApplication | 1 | 2 | 0 | 0 | 2 | 1 | 0 | 0 |
| ServletUnitWebResponse | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ServletUnitServletContext | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ServletUnitServletConfig | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ServletUnitOutputStream | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ServletUnitHttpSession | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| ServletUnitHttpResponse | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| ServletUnitHttpRequest | 4 | 3 | 0 | 0 | 1 | 1 | 0 | 0 |
| ServletUnitContext | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| ServletUnitClient | 1 | 4 | 1 | 0 | 0 | 0 | 0 | 0 |
| ServletRunner | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| InvocationContext | 4 | 3 | 0 | 0 | 1 | 2 | 0 | 0 |
| BasicAuthenticationRequiredException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| WebTable | 1 | 3 | 0 | 0 | 2 | 10 | 0 | 0 |
| WebResponse | 3 | 7 | 0 | 1 | 1 | 5 | 3 | 0 |
| WebRequestSource | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 0 |
| WebRequest | 1 | 2 | 0 | 0 | 2 | 10 | 3 | 0 |
| WebLink | 0 | 1 | 1 | 0 | 1 | 3 | 0 | 0 |
| WebFrame | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| WebConversation | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| WebClient | 1 | 2 | 0 | 0 | 0 | 0 | 2 | 0 |
| URLEncodedMessageBody | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| TextParameterCountException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| TableCell | 0 | 0 | 1 | 1 | 1 | 2 | 0 | 0 |
| SubmitButton | 0 | 0 | 0 | 0 | 1 | 6 | 0 | 0 |
| SingleValuedParameterException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| RedirectWebRequest | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| ReceivedPage | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| PutMethodWebRequest | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| PostMethodWebRequest | 1 | 3 | 1 | 0 | 0 | 2 | 0 | 0 |
| ParsedHTML | 1 | 4 | 0 | 0 | 4 | 4 | 2 | 0 |
| NotHTMLException | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

表 A-2 續表 A-1

| Class | $C_S$ | $U_S$ | $E_S$ | $I_S$ | $C_e$ | $U_e$ | $E_e$ | $I_e$ |
|---|---|---|---|---|---|---|---|---|
| **Average Dependency** | **0.43** | **0.96** | **0.34** | **0.12** | **0.43** | **0.96** | **0.34** | **0.12** |
| NodeUtils | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NoSuchParameterException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| NoSuchFrameException | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MultipartFormRequiredException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| MissingParameterValueException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| MimeEncodedMessageBody | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| MessageBodyWebRequest | 0 | 3 | 1 | 0 | 1 | 2 | 2 | 0 |
| MessageBody | 1 | 1 | 0 | 0 | 2 | 3 | 3 | 0 |
| JTidyPrintWriter | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IllegalUnnamedSubmitButtonException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| IllegalSubmitButtonException | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| IllegalRequestParameterException | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 |
| IllegalParameterValueException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| IllegalNonFileParameterException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| IllegalFileParameterException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| HttpWebResponse | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| HttpUnitUtils | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| HttpUnitOptions | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| HttpNotFoundException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| HttpInternalErrorException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| HttpException | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| HtmlErrorListener | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| HTMLSegment | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 2 |
| GetMethodWebRequest | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| DefaultWebResponse | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ByteTagParser | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ByteTag | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| Base64 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AuthorizationRequiredException | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

表 A-3 httpunit 1.4 元件 dependency 資料表

| Class | $C_S$ | $U_S$ | $E_S$ | $I_S$ | $C_e$ | $U_e$ | $E_e$ | $I_e$ |
|---|---|---|---|---|---|---|---|---|
| **Average Dependency** | **0.43** | **0.96** | **0.34** | **0.12** | **0.43** | **0.96** | **0.34** | **0.12** |
| WebApplication | 3 | 2 | 0 | 0 | 4 | 3 | 0 | 0 |
| ServletUnitWebResponse | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ServletUnitServletContext | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ServletUnitServletConfig | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ServletUnitOutputStream | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ServletUnitHttpSession | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| ServletUnitHttpResponse | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| ServletUnitHttpRequest | 4 | 3 | 0 | 0 | 1 | 1 | 0 | 0 |
| ServletUnitContext | 0 | 1 | 0 | 0 | 2 | 2 | 0 | 0 |
| ServletUnitClient | 1 | 4 | 1 | 0 | 1 | 1 | 0 | 0 |
| ServletRunner | 4 | 4 | 0 | 0 | 1 | 2 | 0 | 0 |
| RequestDispatcherImpl | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ParsedPath | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| JUnitServlet | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| InvocationContextImpl | 4 | 3 | 0 | 1 | 1 | 1 | 0 | 0 |
| InvocationContextFactory | 0 | 2 | 0 | 0 | 3 | 2 | 0 | 1 |
| InvocationContext | 0 | 1 | 0 | 0 | 0 | 3 | 0 | 1 |
| BasicAuthenticationRequiredException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| WebTable | 1 | 3 | 0 | 0 | 2 | 10 | 0 | 0 |
| WebResponse | 4 | 8 | 0 | 1 | 1 | 9 | 3 | 0 |
| WebRequestSource | 0 | 1 | 1 | 0 | 0 | 2 | 2 | 0 |
| WebRequest | 2 | 5 | 0 | 0 | 2 | 14 | 3 | 0 |
| WebLink | 0 | 4 | 1 | 0 | 1 | 4 | 0 | 0 |
| WebFrame | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| WebForm | 3 | 6 | 1 | 0 | 0 | 9 | 0 | 0 |
| WebConversation | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
| WebClientListener | 0 | 3 | 0 | 0 | 0 | 1 | 0 | 0 |
| WebClient | 1 | 3 | 0 | 1 | 0 | 1 | 2 | 0 |
| UploadFileSpec | 0 | 0 | 0 | 0 | 1 | 9 | 0 | 0 |
| UnusedUploadFileException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| UnusedParameterValueException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| UncheckedParameterHolder | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 0 |

表 A-4 續表 A-3

| Class | $C_S$ | $U_S$ | $E_S$ | $I_S$ | $Ce$ | $Ue$ | $Ee$ | $Ie$ |
|---|---|---|---|---|---|---|---|---|
| **Average Dependency** | **0.43** | **0.96** | **0.34** | **0.12** | **0.43** | **0.96** | **0.34** | **0.12** |
| URLEncodedString | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| URLEncodedMessageBody | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| TextFormControl | 0 | 1 | 1 | 0 | 0 | 0 | 2 | 0 |
| TextFieldFormControl | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| TextAreaFormControl | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| TableCell | 0 | 0 | 1 | 1 | 1 | 2 | 0 | 0 |
| SubmitButton | 0 | 2 | 1 | 0 | 2 | 10 | 0 | 0 |
| SelectionFormControl | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| RedirectWebRequest | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| ReceivedPage | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| RadioGroupFormControl | 1 | 2 | 1 | 0 | 1 | 1 | 0 | 0 |
| RadioButtonFormControl | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| PutMethodWebRequest | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| PresetFormParameter | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| PostMethodWebRequest | 1 | 3 | 1 | 0 | 0 | 2 | 0 | 0 |
| ParsedHTML | 1 | 4 | 0 | 0 | 4 | 4 | 2 | 0 |
| ParameterProcessor | 0 | 1 | 0 | 0 | 0 | 12 | 0 | 2 |
| ParameterHolder | 0 | 3 | 0 | 0 | 1 | 1 | 2 | 0 |
| NotHTMLException | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NodeUtils | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NoSuchParameterException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| NoSuchFrameException | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MultipartFormRequiredException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| MissingParameterValueException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| MimeEncodedMessageBody | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| MessageBodyWebRequest | 0 | 3 | 1 | 0 | 1 | 2 | 2 | 0 |
| MessageBody | 1 | 1 | 0 | 0 | 2 | 3 | 3 | 0 |
| LinkParameter | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| JTidyPrintWriter | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IllegalUnnamedSubmitButtonException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| IllegalSubmitButtonException | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| IllegalRequestParameterException | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 |

表 A-5 續表 A-4

| Class | $C_S$ | $U_S$ | $E_S$ | $I_S$ | $C_e$ | $U_e$ | $E_e$ | $I_e$ |
|---|---|---|---|---|---|---|---|---|
| **Average Dependency** | **0.43** | **0.96** | **0.34** | **0.12** | **0.43** | **0.96** | **0.34** | **0.12** |
| IllegalParameterValueException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| IllegalNonFileParameterException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| IllegalLinkParametersRequest | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| IllegalButtonPositionException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| HttpWebResponse | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| HttpUnitUtils | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| HttpUnitOptions | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| HttpNotFoundException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| HttpInternalErrorException | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| HttpException | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| HtmlErrorListener | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| HiddenFieldFormControl | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| HTMLSegment | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 2 |
| GetMethodWebRequest | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 0 |
| FrameHolder | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| FormParameter | 2 | 3 | 0 | 0 | 1 | 1 | 0 | 0 |
| FormControl | 0 | 2 | 0 | 0 | 2 | 2 | 7 | 0 |
| FileSubmitFormControl | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| DisabledSubmitButtonException | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| DefaultWebResponse | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| CheckboxFormControl | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ByteTagParser | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ByteTag | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| BooleanFormControl | 0 | 1 | 1 | 0 | 0 | 0 | 2 | 0 |
| Base64 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AuthorizationRequiredException | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

# 附錄 B：Refining 節果列表

表 B-1 httpunit 1.3 Source

| Source | Type | Level |
|---|---|---|
| WebRequestSource | Extend Source | 1 |
| WebClient | Extend Source | 1 |
| MessageBodyWebRequest | Extend Source | 1 |
| MessageBody | Extend Source | 1 |
| IllegalRequestParameterException | Extend Source | 1 |
| HttpException | Extend Source | 1 |
| AuthorizationRequiredException | Extend Source | 2 |
| HTMLSegment | Implements Source | 1 |
| ServletUnitHttpResponse | Contain Source | 1 |
| SerletUnitContext | Contain Source | 1 |
| ReceivedPage | Contain Source | 1 |
| ParsedHTML | Contain Source | 1 |
| ServletUnitHttpSession | Use Source | 1 |
| ServletRunner | Use Source | 1 |
| WebTable | Use Source | 1 |
| WebRequest | Use Source | 1 |
| WebLink | Use Source | 1 |
| WebFrame | Use Source | 1 |
| WebForm | Use Source | 1 |
| TableCell | Use Source | 1 |
| SubmitButton | Use Source | 1 |
| HtmlErrorListener | Use Source | 2 |

表 B-2 httpunit 1.3 Sink

| Sink | Type | Level |
|---|---|---|
| ServletUnitWebResponse | Extend Sink | 1 |
| BasicAuthenticationRequiredException | Extend Sink | 1 |
| URLEncodedMessageBody | Extend Sink | 1 |
| TextParameterCountException | Extend Sink | 1 |
| SingleValuedParameterException | Extend Sink | 1 |
| RedirectWebRequest | Extend Sink | 1 |
| NoSuchParameterException | Extend Sink | 1 |
| MiultipartFormRequiredException | Extend Sink | 1 |
| MissingParameterValueException | Extend Sink | 1 |
| MimeEncodedMessageBody | Extend Sink | 1 |
| IllegalSubmitButtonException | Extend Sink | 1 |
| IllegalParameterValueException | Extend Sink | 1 |
| IllegalNonFileParameterException | Extend Sink | 1 |
| IllegalFileParameterException | Extend Sink | 1 |
| HttpWebResponse | Extend Sink | 1 |
| HttpNotFoundException | Extend Sink | 1 |
| HttpInternalErrorException | Extend Sink | 1 |
| DefoultWebResponse | Extend Sink | 1 |
| ServletUnitHttpRequest | Contain Sink | 1 |
| InvocationContext | Contain Sink | 1 |
| ServletUnitClient | Use Sink | 1 |
| WebConversation | Use Sink | 1 |
| PostMethodWebRequest | Use Sink | 1 |
| HttpUnitOption | Use Sink | 1 |
| GetMethodWebRequest | Use Sink | 1 |
| ByteTagParser | Use Sink | 1 |

表 B-3 httpunit 1.4 Source

| Source | Type | Level |
|---|---|---|
| WebRequestSource | Extend Source | 1 |
| WebClient | Extend Source | 1 |
| TextFormControl | Extend Source | 1 |
| TextFieldFormControl | Extend Source | 1 |
| ParameterHolder | Extend Source | 1 |
| MessageBodyWebRequest | Extend Source | 1 |
| MessageBody | Extend Source | 1 |
| IllegalRequestParameterException | Extend Source | 1 |
| HttpException | Extend Source | 1 |
| FormControl | Extend Source | 1 |
| BooleanFormControl | Extend Source | 1 |
| AuthorizationRequiredException | Extend Source | 1 |
| HTMLSegment | Implement Source | 1 |
| FrameHolder | Implement Source | 1 |
| WebApplication | Contain Source | 1 |
| ServletUnitHttpResponse | Contain Source | 1 |
| ServletUnitContext | Contain Source | 1 |
| InvocationContextImpl | Contain Source | 1 |
| InvocatinContextFactory | Contain Source | 1 |
| WebFrame | Contain Source | 1 |
| ReceivedPage | Contain Source | 1 |
| RadioGroupFormControl | Contain Source | 1 |
| RadioButtonFormControl | Contain Source | 1 |
| ParsedHTML | Contain Source | 1 |
| MimeEncodedMessageBody | Contain Source | 1 |
| FormParameter | Contain Source | 1 |
| ServletUnitHttpSession | Use Source | 1 |
| ServletRunner | Use Source | 1 |
| WebTable | Use Source | 1 |
| WebResponse | Use Source | 1 |

表 B-4 續表 B-3

| Source | Type | Level |
|---|---|---|
| WebRequest | Use Source | 1 |
| WebLink | Use Source | 1 |
| WebForm | Use Source | 1 |
| UploadFileSpec | Use Source | 1 |
| TableCell | Use Source | 1 |
| SubmitButton | Use Source | 1 |
| PostMethodWebRequest | Use Source | 1 |
| ParameterProcessor | Use Source | 1 |
| ByteTag | Use Source | 1 |
| HtmlErrorListener | Use Source | 2 |

表 B-5 httpunit 1.4 Sink

| Sink | Type | Level |
|---|---|---|
| ServletUnitWebResponse | Extend Sink | 1 |
| BasicAuthenticationRequiredException | Extend Sink | 1 |
| UnusedUploadFileException | Extend Sink | 1 |
| UnusedParameterValueException | Extend Sink | 1 |
| URLEncodedMessageBody | Extend Sink | 1 |
| TextAreaFormControl | Extend Sink | 1 |
| SelectionFormControl | Extend Sink | 1 |
| RedirectWebRequest | Extend Sink | 1 |
| PutMethodWebRequest | Extend Sink | 1 |
| PresetFormParameter | Extend Sink | |
| NoSuchParameterException | Extend Sink | 1 |
| MultipartFormRequiredException | Extend Sink | 1 |
| MissingParameterValueException | Extend Sink | 1 |
| IllegalUnnamedSubmitButtonException | Extend Sink | 1 |
| IllegalSubmitButtonException | Extend Sink | 1 |
| IllegalParameterValueException | Extend Sink | 1 |
| iLLegalNonFileParameterException | Extend Sink | 1 |
| IllegalLinkParametersRequest | Extend Sink | 1 |
| IllegalButtonPositionException | Extend Sink | 1 |
| HttpWebResponse | Extend Sink | 1 |
| HttpNotFoundException | Extend Sink | 1 |
| HttpInternalErrorException | Extend Sink | 1 |
| HiddenFieldFormControl | Extend Sink | 1 |
| FileSubmitFormControl | Extend Sink | 1 |
| DisabledSubmitButtonException | Extend Sink | 1 |
| DefaultWebResponse | Extend Sink | 1 |
| CheckboxFormControl | Extend Sink | 1 |
| URLEncodedString | Implement Sink | 1 |
| ServletUnitServletContext | Contain Sink | 1 |
| ServletUnitHttpRequest | Contain Sink | 1 |

表 B-6 續表 B-5

| Sink | Type | Level |
|---|---|---|
| JUnitServlet | Contain Sink | 1 |
| InvocationContext | Contain Sink | 1 |
| ServletUnitClient | Use Sink | 1 |
| WebConversation | Use Sink | 1 |
| WebClientListener | Use Sink | 1 |
| UncheckedParameterHolder | Use Sink | 1 |
| PostMethodWebRequest | Use Sink | 1 |
| GetMethodWebRequest | Use Sink | 1 |
| ByteTagParser | Use Sink | 2 |
| HttpUnitOption | Use Sink | 2 |

# 附錄 C：Matching 結果列表

表 C-1 LV1 Extend Source Match

| 1.3 Extend Source | 1.4 Extend Source | 相似度 |
|---|---|---|
| WebRequestSource | WebRequestSource | 71% |
| WebClient | WebClient | 84% |
| MessageBody | MessageBody | 100% |
| IllegalRequestParameterException | IllegalRequestParameterException | 75% |
| HttpException | HttpException | 86% |

表 C-2 LV1 Implement Source Match

| 1.3 Implement Source | 1.4 Implement Source | 相似度 |
|---|---|---|
| HTMLSegment | HTMLSegment | 100% |

表 C-3 LV1 Contain Source Match

| 1.3 Contain Source | 1.4 Contain Source | 相似度 |
|---|---|---|
| ServletUnitHttpResponse | ServletUnitHttpResponse | 93% |
| SerletUnitContext | SerletUnitContext | 92% |
| ReceivedPage | ReceivedPage | 100% |
| ParsedHTML | ParsedHTML | 99% |

表 C-4 LV1 Use Source Match

| 1.3 Use Source | 1.4 Use Source | 相似度 |
|---|---|---|
| ServletUnitHttpSession | ServletUnitHttpSession | 97% |
| ServletRunner | ServletRunner | 93% |
| WebTable | WebTable | 98% |
| WebRequest | WebRequest | 72% |
| TableCell | TableCell | 100% |

表 C-5 LV2 Use Source Match

| 1.3 use source | 1.4 use source | 相似度 |
|---|---|---|
| HtmlErrorListener | HtmlErrorListener | 100% |

表 C-6 LV1 Extend Sink Match

| 1.3 Extend Sink | 1.4 Extend Sink | 相似度 |
|---|---|---|
| ServletUnitWebResponse | ServletUnitWebResponse | 93% |
| BasicAuthenticationRequiredException | BasicAuthenticationRequiredException | 100% |
| URLEncodedMessageBody | URLEncodedMessageBody | 94% |
| RedirectWebRequest | RedirectWebRequest | 100% |
| NoSuchParameterException | NoSuchParameterException | 100% |
| MiultipartFormRequiredException | MiultipartFormRequiredException | 100% |
| MissingParameterValueException | MissingParameterValueException | 100% |
| IllegalSubmitButtonException | IllegalSubmitButtonException | 100% |
| IllegalParameterValueException | IllegalParameterValueException | 100% |
| IllegalNonFileParameterException | IllegalNonFileParameterException | 100% |
| HttpWebResponse | HttpWebResponse | 100% |
| HttpNotFoundException | HttpNotFoundException | 100% |
| HttpInternalErrorException | HttpInternalErrorException | 100% |
| DefoultWebResponse | DefoultWebResponse | 97% |
| TextParameterCountException | UnusedUploadFileException | 82% |
| SingleValuedParameterException | DisabledSubmitButtonException | 88% |
| IllegalFileParameterException | UnusedParameterValueException | 80% |

表 C-7 LV1 Contain Sink Match

| 1.3 Contain Sink | 1.4 Contain Sink | 相似度 |
|---|---|---|
| ServletUnitHttpRequest | ServletUnitHttpRequest | 96% |

表 C-8 LV1 Use Sink Match

| 1.3 Use Sink | 1.4 Use Sink | 相似度 |
|---|---|---|
| ServletUnitClient | ServletUnitClient | 79% |
| WebConversation | WebConversation | 100% |
| PostMethodWebRequest | PostMethodWebRequest | 76% |
| GetMethodWebRequest | GetMethodWebRequest | 78% |

表 C-9 剩餘元件之 Match

| 1.3 | 1.4 | 相似度 |
|---|---|---|
| MimeEncodedMessageBody | MimeEncodedMessageBody | 77% |
| HttpUnitOptions | HttpUnitOptions | 97% |
| ByteTagParser | ByteTagParser | 100% |

# 附錄 D：non-match 元件列表

表 4.x non-Match Components

| **1.3 non-Matching components** | **1.4 non-Matching components** |
|---|---|
| MessageBodyWebRequest | TextFormControl |
| WebLink | TextFieldFormControl |
| WebFrame | ParameterHolder |
| WebForm | FormControl |
| InvocationContext | BooleanFormControl |
| | AuthorizationRequiredException |
| | MessageBodyWebRequest |
| | WebApplication |
| | InvocationContextImpl |
| | InvocatinContextFactory |
| | RadioGroupFormControl |
| | RadioButtonFormControl |
| | FormParameter |
| | WebResponse |
| | UploadFileSpec |
| | PostMethodWebRequest |
| | ParameterProcessor |
| | ByteTag |
| | SubmitButton |
| | WebLink |
| | WebForm |
| | UnusedUploadFileException |
| | UnusedParameterValueException |
| | TextAreaFormControl |
| | SelectionFormControl |
| | PutMethodWebRequest |
| | PresetFormParameter |
| | IllegalUnnamedSubmitButtonException |
| | IllegalLinkParametersRequest |
| | IllegalButtonPositionException |
| | HiddenFieldFormControl |

表 D-2 non-Match Components

| **1.3 non-Matching components** | **1.4 non-Matching components** |
|---|---|
| | FileSubmitFormControl |
| | DisabledSubmitButtonException |
| | CheckboxFormControl |
| | InvocationContext |
| | WebClientListener |
| | UncheckedParameterHolder |
| | InvocationContext |