極致開發法的兩個延伸：設計草圖及英詞中句虛擬碼
Design Sketching and Chinese Pseudo-coding: Two Extensions to Extreme Programming

趙英瑾　劉致灝　鞠一文　陳振炎
國立中央大學 資訊工程學系
945402024@cc.ncu.edu.tw

摘要

程式師要直接寫出精確的程式碼,是一大困難與挑戰，更不用說初學者了。本文擴充極致開發法，增加兩個實務：設計草圖與英詞中句虛擬碼，用來改善程式設計的困難，並探討這兩個實務對初學的中學生有何影響。

本方法先用設計草圖表達解題概念,再用英詞中句虛擬碼表現程式邏輯，最後將虛擬碼轉成程式碼。

經過上述方法的輔助，學生對程式設計的恐懼感降低了，學習動機提高了，寫出的程式碼也更精準了。

關鍵詞：極致開發法,設計草圖, 英詞中句虛擬碼。

Abstract

It has been difficult for programmers to write precise source code directly, not to say beginners. This paper extends extreme programming with two practices: 1) design sketching and 2) Chinese pseudo-coding to tackle this. The former expresses solution idea roughly and graphically, and the latter specifies solution logic precisely. The new method has been applied to high school students. The results show less fear and more motivation to programming, along with more precise source code.

Keywords: extreme programming, design sketching, Chinese pseudo-coding

1. 簡介

程式設計是一門不易學習的科目，它像寫文章，同樣意思可用不同的文字表達，如何表達得簡明扼要又正確，需要不斷練習與長期經驗，電腦程式比文章更嚴謹，注重邏輯與因果，錯誤邏輯會導致錯誤結果，甚至執行失敗，因此初學者往往不知如何著手。

本文應用極致開發法(eXtreme Programming, XP) [1]，增加兩個實務：設計草圖 (Design Sketch) 和英詞中句虛擬碼(Pseudo Code)來解決程式設計不易學習的問題;即先使用 Design Sketch 以表達解題概念與步驟，再利用英詞中句 Pseudo Code 針對 Design Sketch 用可讀性高的文字作細部邏輯撰寫(相當於演算法)，最後再將其轉成程式碼，讓以中文閱讀思考寫作的國人提升工作速度與品質。

本文使用上述方法於中學生程式設計教學，使其容易寫出可讀性高且精確的程式碼。另外，XP 的 Pair Programming 論點,與 D. W. Johnson 提出的合作學習 (cooperative learning) 理論，強調透過小組內合作學習方式以精熟學習內容 [2] 不謀而合，在同一時空裡，彼此良好、有效的溝通，能提高學習動機，使學習者將其創意發揮至極。

2. 設計草圖與英詞中句虛擬碼

論文撰寫中英文皆可，本文新增兩個實務到 XP：(1)繪製設計草圖(Sketching)來表達解題概念，(2)用英詞中句 Pseudo code 描述程式邏輯架構，開發步驟如圖一，分述如下：

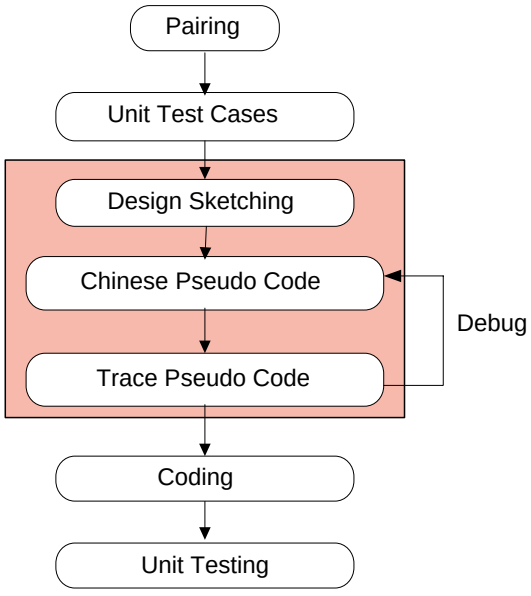


圖 1、XP extended with Sketch and Chinese Pseudo Code

Step 1 - Pairing: 兩位初學者一組，共同使用一部電腦進行解題。

Step 2 - Unit Test Cases：思考可能的測試案例，並

列表說明。

Step 3 - Design Sketching：將腦中的構思利用圖形呈現。

Step 4 -英詞中句 Pseudo Code：根據 Design Sketch，用中文撰寫完整語意的 pseudo code，其中程式碼使用到的名詞(如 variable name)以英詞表示。

Step 5 - Trace Pseudo Code：利用 Step 2 寫出的 Unit Test Cases 來追查(Trace) Step 4 產生的 Pseudo code，進行除錯(Debug)，如有錯誤，立即改寫 Pseudo code。

Step 6 - Coding：將最終產生的英詞中句 Pseudo Code 輸入程式編譯工具中，進行 Coding。版面編排方式以英詞中句 Pseudo Code 為主，code 則用縮排、小字體呈現在 Pseudo code 之後不顯著的地方，因為人眼容易閱讀 Pseudo Code。

Step 7 - Unit Testing：將 Coding 後的 Code 用 Step 2 的 Unit Test Cases 進行 testing。

3. 實驗

本節說明利用本方法開發一個 Bubble Sort [3] 程式，依撰寫 Unit Test Cases (3.1)、Design Sketching (3.2)、撰寫與 trace「英詞中句」 Pseudo Code(3.3)、Coding (3.4)、Unit Testing (3.5)的順序完成。

3.1 測試案例 (Unit Test Cases)

首先根據問題的輸入，列出可能出現具有意義的所有 test cases ，如表一所示：

表 1 Unit Test Cases

編號	Input					測試案例描述
1	1	2	3	4	5	正整數
2	1	2	0	4	5	0、正整數
3	-2	-1	-5	-4	-3	負整數
4	-2	-1	-5	0	-3	0、負整數
5	-2	-1	5	2	3	正整數、負整數
6	-2	-1	0	5	2	負整數、0、正整數

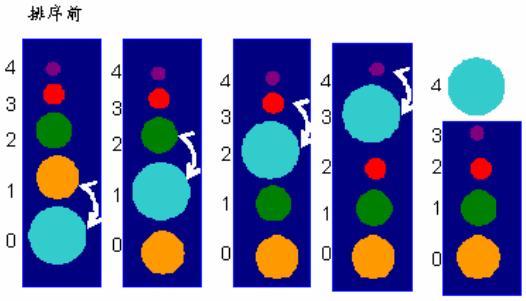
3.2 繪製設計草圖 (Design Sketching)

列出所有可能的測試狀況後，接著進行繪製 Design Sketch，表達解題概念，描述解決問題的步驟與方法。

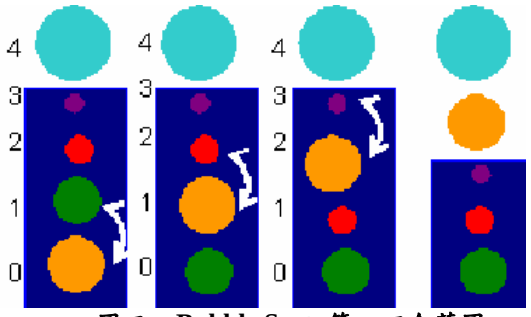
圖二至圖四中，深色的部分代表水，圖中的數字代表水的高度，當中有五個大小不等的 Bubble，Bubble 體積越大，浮力就越大，即浮出水面的速度越快，因此，圖中的五個 Bubble 會陸陸續續地浮出水面。

假設排序前，氣泡由小到大排序，第一回合的 Bubble Sort，由最底部的氣泡開始與其相鄰的氣泡比較，若下方的氣泡大於上方的氣泡則交換位置，再將比較的位置向上提一層，重覆上述步驟，直到最大的氣泡浮出水面為止，如圖二所示：

完成第一回合後，需要排序的資料就會少一個，接著進行第二回合的 Bubble Sort，浮出第二大的氣泡，解題過程如圖三所示：



圖二、Bubble Sort 第一回合草圖



圖三、Bubble Sort 第二回合草圖

完成第二回合後，需要排序的資料就會再少一個，接著進行第三回合和第四回合，最後得到排序的結果，如圖四所示：

圖四、Bubble Sort 的排序結果

3.3 撰寫與 Trace 英詞中句 Pseudo code

完成 3.2 中圖二~圖四的 Design Sketch 之後，根據 Design Sketch 撰寫英詞中句的 Pseudo Code，

並以人腦的思維模式表達，使解題步驟與邏輯在一次又一次整理過程中，漸次明朗，當 Pseudo code 完成後，即利用 test case 進行 Debug，直到通過所有 test case。

以下說明 Design Sketch 轉換成精準的 Pseudo code 的過程，Bubble Sort 第一版的英詞中句 Pseudo code 如下：

(list[j]代表 Design Sketch 中位置 j 的 Bubble)

1.第一回合
  (1)若 List[0]比 List[1]大，則
    將 List [0]和 List [1]的資料交換
  (2)若 List[1]比 List[2]大，則
    將 List [1]和 List [2]的資料交換
  (3)若 List[2]比 List[3]大，則
    將 List [2]和 List [3]的資料交換
  (4)若 List[3]比 List[4]大，則
    將 List [3]和 List [4]的資料交換
2.第二回合
  (1)若 List[0]比 List[1]大，則
    將 List [0]和 List [1]的資料交換
  (2)若 List[1]比 List[2]大，則
    將 List [1]和 List [2]的資料交換
  (3)若 List[2]比 List[3]大，則
    將 List [2]和 List [3]的資料交換
3.第三回合
  (1)若 List[0]比 List[1]大，則
    將 List [0]和 List [1]的資料交換
  (2)若 List[1]比 List[2]大，則
    將 List [1]和 List [2]的資料交換
4.第四回合
  若 List[0]比 List[1]大，則
  將 List [0]和 List [1]的資料交換

由上述第一版的 Pseudo code 可以整理以下四點觀察：

1.前提：假設共有 n (n = 5)筆資料存放於 List[0]、List[1]、List[2]、List[3]、List[4] 中。假設 List 中儲存位置編號為 j，j = 0,1,2,3,4 (即 j=0~n-1)。
2.由 List[0]開始，進行兩兩比較：
  1.List[0]與 List[1]比較
  2.List[1]與 List[2]比較
  3.List[2]與 List[3]比較
  4.List[3]與 List[4]比較
  (即：List [j]與 List[j+1]比較)
3.每一回合找到最大數後，下一回合進行排序的資料就少一個，因此第 i 回合須比較 n-i 次。
4.List 中的資料完成排序共需進行 n-1 回合。

由以上四點，將 Pseudo Code 簡化成第二版的 Pseudo Code 如下：

1.設 List 中有 n 筆資料存放於 List [j]中(j=0~n-1)。
2.由 j=0 開始進行兩兩比較，即
  List [j]與 List[j+1]比較
  若 List [j]比 List[j+1]大，則將 List [j]和 List [j+1]的資料交換。
3.j 值遞增 1，重覆 2.，直到 j=n-1-i 的比較完成為止。(i 為回合數)
4.共進行 n-1 回合的比較：i(回合數)從 1 開始，每次遞增 1，當進行到 n-1 回合時，List 中的資料完成由大到小的排列順序。

由上述第二版的 Pseudo code 可觀察到三點：

1.List[j]中，j=0~n-1，每次遞增 1。
2.每一回合可找出欲排序資料中最大的。
3.每完成一回合的排序，等待排序的資料就少一個。

整理第三版的 Pseudo Code 如下：

1.假設存放了 n 筆資料於 List [j]中 (j=0~n-1)
  j=0，每次遞增 1。
2.List [j]與 List[j+1]比較大小
  若 List [j]比 List[j+1]大，則
  將 List [j]和 List [j+1]的資料交換。
3.每回合兩兩比較直到 j= (n-1-i)時，完成第 i 回合的排序。
4.i=1，每次遞增 1，直到完成 n-1 回合的比較為止。

由上述第三版的 Pseudo code 可觀察到兩點：

1.分別進行 0,1,2,…,(n-1)，共四回合的比較次數。
2.每增加一回合，排序的資料就少一個。

整理第四版的 Pseudo Code 如下：

1.進行 i 回合的排序(i=1~n-1)。
2.每一回合資料比對次數 j，隨回合數之增加而遞減(j=n-1-i)。
3. List [j]與 List [j+1]內的數值比較，判斷儲存內容是否交換? 若是，則將 List [j]和 List [j+1]內的資料交換。

完成 Pseudo code 之後，將一個 test case 代入其中，進行 debug，如果此 test case 順利通過，則進行 Coding，如果無法通過，則找出問題所在，並修改 Pseudo code。

3.4 撰寫程式 (Coding)

本節將 3.3 的第四版「英詞中句」Pseudo Code 逐行編寫成程式碼，一行 Pseudo Code 後，緊接者程式碼，編排方式以 Pseudo Code 為主，程式碼為輔，如下：

```
'進行 i 回合的排序(i=1~n-1)
For i = 1 To n - 1
  '每一回合資料比對次數 j，隨回合數之增加而遞減(j=n-1-i)
  For j = 0 To n-1- i
  'List [j]與 List [j+1]內的數值比較，判斷儲存內容是否交換?
  If List(j) > List(j + 1) Then
  '若是，則進行互換
      Temp = List(j)
      List(j) = List(j + 1)
      List(j + 1) = Temp
```

```
    End If
    Next j
Next i
```

如此就完成將 Pseudo code 轉換成 source code 了。

3.5 單元測試 (Unit Testing)

完成 Code 後，進行 Unit Testing，將所有的 Test cases 進行測試，提高程式的品質，表二為所有試結果紀錄。

表 2 Unit Testing

Input					Output					Test Result
1	2	3	4	5	5	4	3	2	1	OK
1	2	0	4	5	5	4	2	1	0	OK
-2	-1	-5	-4	-3	-1	-2	-3	-4	-5	OK
-2	-1	-5	0	-3	0	-1	-2	-3	-5	OK
-2	-1	5	2	3	5	3	2	-1	-2	OK
-2	-1	0	5	2	5	2	0	-1	-2	OK

4. 評估

當學習者初學程式語言時，同儕之間的先備知識相同，在同一時間、空間，先了解抽象的解題架構，再用最簡單的方式表達思考邏輯，將有助於即時釐清盲點，修正錯誤。

應用 New Agile Method[4]，繪製解題概念的 Sketch，並根據 Design Sketch 用英詞中句撰寫 Pseudo code，而 Pair 可以在 Programming 過程中利用 Design Sketch，彼此交流、進行即時討論、溝通，使學習寫程式的過程中、變得更有趣，學習動機提高[5][6]，最後在龍潭農工進行實驗，學生對 XP 的兩個延伸方法的看法，歸納如表三所示：

表 3 學習者對設計草圖及英詞中句之看法

特色 / 步驟	優點	缺點
繪製草圖	• 可以促進溝通。 • 抽象問題容易用圖表達。 • 解題過程可加速釐清問題。	• 畫出詳盡的 Sketch 很費時。
英詞中句虛擬碼	• 不用考慮如何用字、拼字。 • 閱讀時，能馬上了解意思。 • 容易寫出程式。 • 節省時間。	• 僅中國人適用。 • 抽象概念難以文字描述。 • 重整略有困難（如形成迴圈之思考）。

本研究以 Pair 為主體，程式設計的過程強調雙向溝通和主動參與，尊重個人看法及他人意見，重視解題過程，程式編譯過程與結果並重，以激發創造力，表四為本法與傳統程式開發方法之比較。

表 4 學習者對設計草圖及英詞中句之看法

項目 / 方法	參與者	解題過程	互動性	溝通性	除錯效率
本研究	Pair	腦力激盪	高	高	高
傳統	1	獨力苦思	低	低	低

5. 結論

程式開發是一連串提升抽象層次的活動，藉由 XP 與本研究延伸的 Design Sketch 與英詞中句 Pseudo Code 方法，可將抽象的程式，有方法、有步驟的開發出來，並能幫助程式開發者撰寫精準的程式碼，testing 可以提高程式碼的品質，本法之優點如下：

1. Design Sketch 可將抽象解題概念，用圖形捕捉、表達、溝通、檢查、修改，能夠迅速呈現出解題的邏輯。
2. 英詞中句 Pseudo Code 的中文敘述容易了解，同時 Pseudo Code 讓同學由 High Abstract Level 往 Low Abstract Level 發展， 形成 Top-Down 的思考，所以可很清楚的表達出解題細節，並可以利用測試案例追查(Trace)之，這樣可確保邏輯上的正確無誤。
3. 兩人一組(Pair Programming)不斷溝通，降低溝通的成本，減少遇到困難時的逃避心態，且易於激發創意，並可避免一個人思考時的盲點，減少 debug 的時間。

參考文獻

[1] Kent Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, 2000

[2] D. W. Johnson, E. J. Holubec, R. T. Johnson, Circles of Learning：cooperation in the classroom, Association for Supervision and Curriculum, 1990.

[3] Richard F. Gilberg, Behrouz A. Forouzan, Data Structure: a Pseudocode Approach with C++, Brooks/cole, 2001.

[4] Jen-Yen Chen, A New Agile Method for High Quality Software, http://ape.csie.ncu.edu.tw/ds/myAgile2.ppt, March 2004.

[5] Kim Man Lui, Keith C. C. Chen, When Does a Pair Outperform Two Individuals? , 2003.

[6]Tim Menzies, Jefferey Smith Lane, David Raffo, When is Pair Programming Better?, 2003.