106年 02月 16日 修正替換頁

【發明說明書】

【中文發明名稱】

建立個人化程式設計模型的方法

【英文發明名稱】

APPROACH TO CONSTRUCT PERSONAL CODING ACTIVITY MODELS

【技術領域】

【0001】 本發明係有關於一種建立個人化程式設計模型的方法，尤其是指一種除了可以提供程式設計人員於程式設計的過程中選取適切的程式設計活動來提高程式碼的品質外，並可以藉由不建議的程式設計活動來降低程式碼的瑕疵，以能提供程式設計人員更便捷的開發技術，而在其整體施行使用上更增實用功效特性之建立個人化程式設計模型的方法創新設計者。

【先前技術】

【0002】 按，目前程式設計進行軟體開發的方式對於程式的正確性是透過於程式設計〔Ｃｏｄｉｎｇ〕與執行〔Ｅｘｅｃｕｔｉｏｎ〕來驗證程式的正確性〔Ｃｏｒｒｅｃｔｎｅｓｓ〕，然而程式設計的能力則因人而異，所開發出來的軟體可能因為不當的設計〔Ｉｍｐｒｏｐｅｒ ｄｅｓｉｇｎ〕而增加錯誤產生的機會，目前軟體開發人員對於此問題則是利用程式碼檢視〔Ｃｏｄｅ Ｉｎｓｐｅｃｔｉｏｎ〕的方式來確認程式碼的品質〔Ｑｕａｌｉｔｙ ｏｆ

Ｃｏｄｅ〕，然而不論是透過執行或是程式碼檢視來確認程式碼的品質皆在程式碼完成後，這種檢視〔Ｉｎｓｐｅｃｔｉｏｎ〕的方式屬於靜態的檢視〔Ｓｔａｔｉｃ Ｉｎｓｐｅｃｔｉｏｎ〕，也就是針對已完成的程式碼進行分析，在程式設計的過程中除了程式的架構〔Ｓｔｒｕｃｔｕｒｅ〕外，設計步驟也是影響程式品質〔Ｑｕａｌｉｔｙ〕的一大要素。

【0003】 其中，就目前用於程式偵錯〔Ｄｅｂｕｇｇｉｎｇ〕與瑕疵預防〔Ｄｅｆｅｃｔ Ｄｅｔｅｃｔｉｏｎ〕之技術而言，請參閱公告於９２年６月１日之第５３５０５３號「電腦系統中搜尋錯誤程式碼的方法」，其提供一種電腦系統，其包含有一輸入系統及一輸出系統，該電腦系統中需除錯的程式碼有複數個指令，該輸入系統係用來指出程式碼中的錯誤變數，而該錯誤變數有一與期待值不同的錯誤值；該錯誤變數的錯誤集包含每一個與該錯誤變數相關的指令，且該錯誤集中的指令均給予一優先值來指出造成該錯誤變數產生的可能性；最後，該輸出系統依據優先值依序將該錯誤集中每一個指令顯示出來。該「電腦系統中搜尋錯誤程式碼的方法」主要是分析程式碼來找出程式碼中的錯誤，以分析變數〔Ｖａｒｉａｂｌｅｓ〕為主，而分析的方式是透過自動比對變數的輸出值〔Ｏｕｔｐｕｔ〕與預期值〔Ｅｘｐｅｃｔａｔｉｏｎ〕，來了解程式碼的正確性，如果比對不正確，則依此來找出導致不正確的程式碼。

【0004】 請再參閱公告於９３年５月２１日之第５８８２３８號「程

式除錯方法」，其是先在一待測程式中對應複數事件設置複數中斷點後，執行該待測程式以輸出其中之一中斷點的診斷碼，並由該診斷碼重設其相對應之事件的參數，最後則依據該重設後之參數執行該事件，使該事件進行錯誤處理，藉此來進行程式之錯誤處理功能及其完整性的測試。該「程式除錯方法」是在待測程式中安排多個中斷點〔ｂｒｅａｋｐｏｉｎｔｓ〕，接著執行該待測程式以輸出其中一個中斷點的診斷碼，再依照所輸出的診斷碼來重設〔Ｒｅｓｅｔ〕對應的事件參數，再以該參數執行對應的處理程式。這個方式也是需要於所完成的程式碼中插入中斷點。

【0005】 請再參閱公告於９３年５月２１日之第５８８２６３號「應用於分散式網際網路的爪哇腳本語言程式錯誤處理方法」，分別對客戶端、伺服端以及傳輸通信協定提供一種錯誤處理程式以及執行程式的撰寫格式。客戶端、伺服端乃至於遠端呼叫程式於執行發生錯誤時，對應的錯誤處理程式均能以一延伸標記語言格式字串的形式，將發生錯誤的程式功能與檔案名稱以程式呼叫的逆順序傳遞至客戶端主程式錯誤處理堆疊。該「應用於分散式網際網路的爪哇腳本語言程式錯誤處理方法」主要是處理Ｊａｖａ Ｓｃｒｉｐｔ的除錯方法，主要的概念仍然是透過執行欲測試的程式，當錯誤發生時，再將程式呼叫的逆順序傳至處理程式，以進行處理，這種處理方式也是針對已完成的程式碼進行執行的測試。

【0006】 請再參閱公告於９６年１月２１日之第Ｉ２７１６１７號

「自動識別電腦程式中的程式錯誤之技術」，其在一實施例中，一種用以自動地識別在一電腦程式中的一程式錯誤之方法包括接收一程式錯誤訊息，以指出在程式執行時已發生一新的程式錯誤，並產生新的程式錯誤之一程式堆疊軌跡，指出在程式中新的程式錯誤之源頭。一第一操作應用至新的程式錯誤之程式堆疊軌跡的至少第一部份，以產生與新的程式錯誤相關之一第一數字碼。比較與新的程式錯誤相關之第一數字碼與相關於先前已識別出的程式錯誤之一個或數個儲存的第一數字碼，以決定是否新的程式錯誤與該先前識別出的程式錯誤的至少之一錯誤相同。若是，備置與解決先前識別出的程式錯誤有關的儲存資料以用於解決新的程式錯誤。該「自動識別電腦程式中的程式錯誤之技術」乃著重於建立錯誤識別與處理的方法，主要係為建立程式執行時所產生的錯誤之關聯性，每次當新錯誤產生時，系統會傳出一個數字碼（Ｓｔｒｉｎｇ．ｈａｓｈＣｏｄｅ（）），此數字碼會與處理的方式產生關連性並儲存，當後續有新的錯誤產生，而新產生的數字碼存在於資料庫中，則使用者可以由其中找出用於解決之前錯誤的處理方式。

【0007】 請再參閱公告於９８年６月１日之第Ｉ３１０４９４號「用於電腦系統中解決錯誤的方法、系統及物件」，係為一種用於找出並解決一叢集環境中的一錯誤的方法與系統。對該叢集（ｃｌｕｓｔｅｒ １００）設置至少一多重主控（ｍｕｌｔｉ－ｈｏｍｅｄ）節點（１１０）以及用於每個網路介面（１１２、１１４）的至少一閘道器（１４０）。心跳（ｈｅａｒｔｂｅａｔ）訊息於預定週

期間隔中在對等節點與該閘道器之間傳送（２０２）。在任何節點或閘道器遺失一心跳訊息的情況中（２０４），一ＩＣＭＰ顯示指令（ｅｃｈｏ）被發出至每個網路介面之該叢集中的每個節點與閘道器（２０６）。如果用以回應該ＩＣＭＰ顯示指令之一節點遺失與一網路遺失皆未被確認，一應用程式層級封包查詢指令（ｐｉｎｇ）被送出（２２４）以判定與缺少該心跳訊息有關之錯誤是否為一暫時錯誤狀況（２２８）或一應用程式錯誤（２３０）。該「用於電腦系統中解決錯誤的方法、系統及物件」主要處理資訊傳送中遺失的錯誤，其方法並非直接檢查程式的錯誤，而是透過應用程式訊息的回報來診斷發生問題的地方，其中包含網路的錯誤或是應用程式本身的錯誤。

【0008】 然而，上述各種方式雖可達到驗證程式正確性之預期功效，但也在其整體實際操作施行上發現，現行做法的主要缺點是程式碼設計完成後再檢視〔或執行〕，若發現設計不當時，則對於已完成的程式需要投入修改的成本〔Ｃｏｓｔ〕，其次是無法於設計前予以提醒，以致拖延設計的時程〔Ｓｃｈｅｄｕｌｅ〕，致令其在整體操作施行上仍存在有改進之空間。

【0009】 緣是，發明人有鑑於此，秉持多年該相關行業之豐富設計開發及實際製作經驗，針對現有之技術方法及缺失再予以研究改良，提供一種建立個人化程式設計模型的方法，以期達到更直覺的程式設計流程導引與更佳的程式碼品質之目的者。

【發明內容】

【0010】 本發明之主要目的在於提供一種建立個人化程式設計模型的方法，其主要係除了可以提供程式設計人員於程式設計的過程中選取適切的程式設計活動來提高程式碼的品質外，並可以藉由不建議的程式設計活動來降低程式碼的瑕疵，以能提供程式設計人員更便捷的開發技術，而在其整體施行使用上更增實用功效特性者。

【0011】 本發明建立個人化程式設計模型的方法之主要目的與功效，係由以下具體技術手段所達成：

【0012】 其主要係包括有程式設計活動資訊蒐集〔Coding Activity Collection，CAC〕、個人化程式設計樣式〔Personal Coding Activity Patterns，PCAP〕與個人化的程式設計模型〔Personal Coding Activity Models，PCAM〕；其中：

【0013】 該程式設計活動資訊蒐集〔Coding Activity Collection，CAC〕，主要蒐集與處理程式設計人員的程式設計活動與其所產出的程式碼之品質資訊〔Quality of Codes〕，所蒐集的程式設計活動包含修改的程式碼片段〔Modified Codes Fragments，MCF〕、編譯結果〔Compilation Results〕與執行結果〔Execution Results〕；

【0014】 該個人化程式設計樣式〔Ｐｅｒｓｏｎａｌ Ｃｏｄｉｎｇ Ａｃｔｉｖｉｔｙ Ｐａｔｔｅｒｎｓ，ＰＣＡＰ〕，其係利用由該該程式設計活動資訊蒐集步驟所選定的交易集〔Ｔｒａｎｓａｃｔｉｏｎ Ｓｅｔｓ〕來找出程式設計人員在程式設計過程中較常出現的程式樣式〔Ｐａｔｔｅｒｎｓ〕；

【0015】 該個人化的程式設計模型〔Ｐｅｒｓｏｎａｌ Ｃｏｄｉｎｇ Ａｃｔｉｖｉｔｙ Ｍｏｄｅｌｓ，ＰＣＡＭ〕，主要結合由程式設計人員的修改程式碼中找出的產出個人化程式設計樣式〔ＰＣＡＰ〕與其執行結果〔Ｒｅｓｕｌｔｓ〕來建立個人化的程式設計模型〔ＰＣＡＭ〕，並利用該模型來評估後續程式碼產出的品質〔Ｑｕａｌｉｔｙ ｏｆ Ｃｏｄｅｓ〕，提供程式設計人員建議的程式設計活動〔Ｓｕｇｇｅｓｔｅｄ Ｃｏｄｉｎｇ Ａｃｔｉｖｉｔｙ〕與不建議的程式設計活動〔Ａｖｏｉｄｅｄ Ｃｏｄｉｎｇ Ａｃｔｉｖｉｔｙ〕。

【0016】 本發明建立個人化程式設計模型的方法，其中，該程式設計活動資訊蒐集所蒐集之該修改的程式碼片段與該編譯結果是由開發工具〔ＩＤＥ〕於使用者執行編譯時自動蒐集，而該執行結果則是由程式設計人員於執行後指定。

【0017】 本發明建立個人化程式設計模型的方法，其中，該修改的程式碼片段〔ＭＣＦ〕定義為一段包含修改過程式碼的連續程式碼〔Ｃｏｄｅｓ〕，該編譯結果〔Ｃｏｍｐｉｌａｔｉｏｎ Ｒｅｓｕ

ｌｔｓ〕則利用ＩＤＥ工具對修改的程式進行編譯的結果，且僅蒐集程式碼修改片段〔ＭＣＦ〕中之編譯成功的部份。

【0018】 本發明建立個人化程式設計模型的方法，其中，該程式設計活動資訊蒐集係於使用者的開發工具〔ＩＤＥ〕中設置一個外掛程式〔Ｍｏｄｉｆｉｅｄ Ｃｏｄｅ Ｒｅｔｒｉｅｖｅｄ Ｐｌｕｇ－ｉｎ，ＭＣＲＰ〕，該外掛程式用於蒐集程式設計師的程式設計活動資訊，所蒐集的資訊則傳送到遠端的伺服器〔Ｓｅｒｖｅｒ〕中；該外掛程式於每次程式設計人員完成修改〔Ｍｏｄｉｆｉｅｄ〕與儲存〔Ｓａｖｅｄ〕時，會自動比對修改後與修改前的程式碼，並將修改的部分擷取出來，而所找出的修改的部分則包含新增〔Ｉｎｓｅｒｔｅｄ ｃｏｄｅ，ＩＮＳ〕、刪除〔Ｄｅｌｅｔｅｄ ｃｏｄｅ，ＤＥＬ〕與修改〔Ｍｏｄｉｆｉｅｄ，ＭＯＤ〕。

【0019】 本發明建立個人化程式設計模型的方法，其中，該程式設計活動資訊蒐集在修改的程式碼片段〔ＭＣＦ〕的蒐集上使用３個參數來簡化蒐集的程序，分別為關鍵詞〔ｋｅｙｗｏｒｄｓ〕、識別字〔ｉｄｅｎｔｉｆｉｅｒｓ〕以及範圍〔Ｒａｎｇｅ〕，該關鍵詞〔ｋｅｙｗｏｒｄｓ〕為程式語言的關鍵字，一組關鍵詞集合〔ｋｅｙｗｏｒｄ ｓｅｔ〕預先定義於系統中，用於過濾出要蒐集的修改的程式碼片段；該識別字則是程式碼中所用到的變數〔ｖａｒｉａｂｌｅｓ〕與類別〔ｃｌａｓｓ〕、函數〔ｍｅｔｈｏｄ〕。

【0020】 本發明建立個人化程式設計模型的方法，其中，該程式設計

活動資訊蒐集用來產生交易〔Ｔｒａｎｓａｃｔｉｏｎ〕的演算法如下〔範圍為 r〕：

【0021】 A1.對於每個修改的程式碼 c_i，找到包含該程式碼的第一層方法〔Ｍｅｔｈｏｄ〕或函式；

【0022】 B1.建立ｐｒｏｇＭｓｇ（）的程式結構樹〔Ｐｒｏｇｒａｍ Ｓｔｒｕｃｔｕｒｅ Ｔｒｅｅ，ＰＳＴ〕，並移除其中非出現在關鍵詞集合 S^K 與識別字集合 S^I 中的關鍵詞與識別字，並令依此所建的ＰＳＴ為 pst_i；

【0023】 C1.將此 pst_i 以前序表示法表示成一有序字串的集合，並令此有序字串集合為 p_i；

【0024】 D1.如果此有序字串集合為 p_i 的節點數〔ｎｏｄｅｓ〕（不包含－１）小於（或等於 r），則 p_i 即選為交易〔Ｔｒａｎｓａｃｔｉｏｎ〕，否則由此有序字串集合中挑出 r 個節點數〔ｎｏｄｅｓ〕作為交易〔Ｔｒａｎｓａｃｔｉｏｎ〕；

【0025】 E1.此步驟一直持續到所有的修改程式碼皆涵蓋在所選的交易中為止。

【0026】 本發明建立個人化程式設計模型的方法，其中，於該步驟Ｄ１中，所述之修改的程式碼 c_i 係位於交易的中間節點。

【0027】 本發明建立個人化程式設計模型的方法，其中，於找出該個人化程式設計樣式需先指定最小支持度〔ｍｉｎｉｍｕｍ ｓｕｐ

ｐｏｒｔ〕s，假設所有關鍵詞與識別字的集合以I_1表示，則$I_1 = S^K_1 \cup S^I_1$，用來產生關鍵詞與識別字的頻繁集〔ｌａｒｇｅ ｉｔｅｍｓｅｔｓ〕的演算法如下所示：

【0028】 A2.計算I_1中每個單一項目的支持度（不包含－１），計算方式為出現於ｔｒａｎｓａｃｔｉｏｎ中的次數〔每個ｔｒａｎｓａｃｔｉｏｎ僅計算一次〕，僅保留支持度大於s的項目，並以L_1表示第１階頻繁項目集〔ｆｉｒｓｔ－ｌｅｖｅｌ ｌａｒｇｅ ｉｔｅｍｓｅｔ〕；

【0029】 B2.產生第２階有序的項目集〔ｓｅｃｏｎｄ－ｌｅｖｅｌ ｏｒｄｅｒｅｄ ｉｔｅｍｓｅｔ〕$I_2 = L_1 \times L_1$，其中每個項目（I_i，I_j）的支持度則是視（I_i，＊，I_j）出現在ｔｒａｎｓａｃｔｉｏｎ中的次數，其中”＊”表示I_i與I_j之間可以初現一個或是多個其他項目；

【0030】 C2.由 A2 與 B2 所產生第i組有序的項目集依照$I_i = L_{i-1} \times L_1$的規則來產生，而第i階頻繁項目集L_i則是由I_i選出〔大於s者〕，此步驟一直重複到沒有頻繁項目集產生為止〔亦即L_{i+1}為空集合〕。

【0031】 本發明建立個人化程式設計模型的方法，其中，修改程式碼後的執行結果利用 K-means 分群法依屬性分成數個錯誤群〔ｄｅｆｅｃｔ ｃｌｕｓｔｅｒ〕。

【0032】 本發明建立個人化程式設計模型的方法，其中，用於產生該

個人化的程式設計模型的演算法如下所示，其中用於挑選規則的最小信賴度〔ｍｉｎｉｍｕｍ　ｃｏｎｆｉｄｅｎｃｅ〕為c：

【0033】 A3.計算錯誤群內〔ｄｅｆｅｃｔ　ｃｌｕｓｔｅｒ〕E每個元素的支持度，並挑選支持度大於s的項目，並表示為L_E；

【0034】 B3.計算$L_n \times L_E$〔L_n表示ＰＣＡＰ〕中所有項目的信賴度，並挑選信賴度大於c的項目，則此項目為所產生的規則〔ｒｕｌｅ〕，令其表示為R；

【0035】 其中$r_i \in L_n \times L_E$的支持度是由計算（l_1，・・・，l_n）出現於交易集合中的次數而得，若一個規則r_i的前項〔ｌｅｆｔ－ｈａｎｄ　ｓｉｄｅ〕出現次數為n_t，而總交易數〔ｔｈｅ　ｎｕｍｂｅｒ　ｏｆ　ｔｒａｎｓａｃｔｉｏｎｓ〕為m，且此規則r_i出現於無錯誤交易的次數〔ｐｏｓｉｔｉｖｅ　ａｐｐｅａｒａｎｃｅｓ〕為n_p，則此規則r_i的支持度為n_t / m，且正面預測〔預測為無錯誤〕的信賴度為n_p / n_t，而所得到的規則可以表示為$R = \{r_1, \cdots, r_n\}$可視為個人程式設計模型。

【圖式簡單說明】

【0036】 第一圖：本發明之程式設計活動資訊蒐集與分析架構示意圖

【0037】 第二圖：本發明之程式設計活動資訊蒐集元件架構示意圖

【0038】 第三圖：本發明之程式設計活動資訊自動蒐集介面示意圖

【0039】 第四圖：本發明之修改程式碼片段示意圖

【實施方式】

【0040】 為令本發明所運用之技術內容、發明目的及其達成之功效有更完整且清楚的揭露，茲於下詳細說明之，並請一併參閱所揭之圖式及圖號：

【0041】 本發明主要係包括有程式設計活動資訊蒐集〔Coding Activity Collection，CAC〕、個人化程式設計樣式〔Personal Coding Activity Patterns，PCAP〕與個人化的程式設計模型〔Personal Coding Activity Models，PCAM〕；其中：

【0042】 一、該程式設計活動資訊蒐集〔Coding Activity Collection，CAC〕：

【0043】 其主要蒐集與處理程式設計人員的程式設計活動與其所產出的程式碼之品質資訊〔Quality of Codes〕，請參閱第一圖本發明之程式設計活動資訊蒐集與分析架構示意圖所示，所蒐集的程式設計活動包含修改的程式碼片段〔Modified Codes Fragments，MCF〕、編譯結果〔Compilation Results〕與執行結果〔Execution Results〕，其中修改的程式碼片段與編譯結果則是由開發工具〔IDE〕於使用者執行編譯時自動蒐集，而執行結果則是由程式設計人員於執行後指定。

【0044】 該修改的程式碼片段〔ＭＣＦ〕定義為一段包含修改過程式碼的連續程式碼〔Ｃｏｄｅｓ〕，該編譯結果〔Ｃｏｍｐｉｌａｔｉｏｎ Ｒｅｓｕｌｔｓ〕則利用ＩＤＥ工具對修改的程式進行編譯的結果，該編譯結果包含編譯失敗〔Ｆａｉｌ〕與編譯成功〔Ｓｕｃｃｅｓｓｆｕｌ〕，而本發明所蒐集的程式碼修改片段〔ＭＣＦ〕則僅包含編譯成功的部份，該造成編譯失敗的錯誤則可以由編譯器〔Ｃｏｍｐｉｌｅｒ〕直接找出並修正。

【0045】 而請再一併參閱第二圖本發明之程式設計活動自動資訊蒐集元件架構示意圖所示，其係於使用者的開發工具〔ＩＤＥ〕中設置一個外掛程式〔Ｍｏｄｉｆｉｅｄ Ｃｏｄｅ Ｒｅｔｒｉｅｖｅｄ Ｐｌｕｇ－ｉｎ，ＭＣＲＰ〕，該外掛程式用於蒐集程式設計師的程式設計活動資訊，所蒐集的資訊則傳送到遠端的伺服器〔Ｓｅｒｖｅｒ〕中；該外掛程式於每次程式設計人員完成修改〔Ｍｏｄｉｆｉｅｄ〕與儲存〔Ｓａｖｅｄ〕時，會自動比對修改後與修改前的程式碼，並將修改的部分擷取出來，而所找出的修改的部分則包含新增〔Ｉｎｓｅｒｔｅｄ ｃｏｄｅ，ＩＮＳ〕、刪除〔Ｄｅｌｅｔｅｄｃｏｄｅ，ＤＥＬ〕與修改〔Ｍｏｄｉｆｉｅｄ，ＭＯＤ〕。

【0046】 請再一併參閱第三圖本發明之程式設計活動資訊自動蒐集介面示意圖所示，當程式設計人員進行程式修改前先輸入欲修改的專案資訊，其 User Information 視窗，如果為瑕疵修正則選取瑕疵

編號〔Ｄｅｆｅｃｔ〕，若為需求則選取需求編號〔ＲＥＱ〕，如此則當程式設計人員於編寫程式碼完成並按下儲存鍵〔Ｓａｖｅ〕，而編譯器檢查無語法錯誤時，程式設計活動資訊自動蒐集外掛程式〔ＭＣＲＰ〕便會自動擷取修改的部分，並傳送此資訊到所設定的Ｓｅｒｖｅｒ中，該次修改後的執行結果亦由程式設計人員進行輸入與紀錄。

【0047】 而在修改的程式碼片段〔ＭＣＦ〕的蒐集上使用３個參數來簡化蒐集的程序，分別為關鍵詞〔ｋｅｙｗｏｒｄｓ〕、識別字〔ｉｄｅｎｔｉｆｉｅｒｓ〕以及範圍〔Ｒａｎｇｅ〕，該關鍵詞〔ｋｅｙｗｏｒｄｓ〕為程式語言的關鍵字，例如ｉｆ、ｆｏｒ、ｗｈｉｌｅ等，一組關鍵詞集合〔ｋｅｙｗｏｒｄ ｓｅｔ〕預先定義於系統中，用於過濾出要蒐集的修改的程式碼片段；該識別字則是程式碼中所用到的變數〔ｖａｒｉａｂｌｅｓ〕與類別〔ｃｌａｓｓ〕或是函數〔ｍｅｔｈｏｄ〕，例如基本型別〔Ｐｒｉｍｉｔｉｖｅ ｔｙｐｅ〕的變數ｉｎｔ或類別〔ｃｌａｓｓ〕型別Ｖｅｃｔｏｒ。此步驟先將修改的程式碼片段依照前述的參數過濾出關鍵詞與識別字。

【0048】 請參閱下列修改程式碼片段表單所示，如果修改行數為９，且關鍵詞集合為$S^K=\{if, for, while\}$，且識別字集合$S^I=\{InfoMerge, PaoUtil, Vector\}$，則其過濾後的字集，請再一併參閱第四圖本發明之修改程式

碼片段示意圖所示，為了便於產生個人化程式設計樣式〔ＰＣＡＰ〕，將所蒐集到的程式碼片段〔ＭＣＦ〕表示成一個關鍵詞與識別字的有序集合〔Ｏｒｄｅｒｅｄ　Ｓｅｔ〕，而為了表示程式碼片段〔ＭＣＦ〕中各個元素的架構，在此集合中的S^K元素〔Ｋｅｙｗｏｒｄｓ〕表示向下一層〔如第四圖中的*ｉｆ*與*ｆｏｒ*〕，並利用“－１”表示往上一層，因此若範圍〔Ｒａｎｇｅ〕*r*取為７，則所得到的修改的程式碼片段可以表示為pt_i＝｛*Ｐａｏｕｔｉｌ，ｆｏｒ，Ｖｅｃｔｏｒ，ＰａｏＵｔｉｌ，ｉｆ，ＩｎｆｏＭｅｒｇｅ，－１，ＰａｏＵｔｉｌ，－１*｝，其中“－１”表示往上一層，而此pt_i則可視為用來找出個人化程式設計樣式〔ＰＣＡＰ〕的紀錄或是交易〔Ｔｒａｎｓａｃｔｉｏｎ〕。

```
    public void procMsg() {
      ...
01    InfoMerge merge = new InfoMerge();
02    PaoUtil paoutil = new Paoutil();
03    if (NumAttrMerge==2){
04        String item1 = "-:-";
05        String [] itemarr1 = paoutil.StrArr(item1,":");
06        String item2 = "*:*";
07        for (int i=0; i<tmpvec.size()-1; i++){
08            item2 = (String)tmpvec.elementAt(i+1);
09            String [] itemarr2 = paoutil.StrArr(item2,":");
10            if (itemarr2[0].equals(itemarr1[0]))
11                merge.addItem(item1,item2);
12            item1 = item2;
13        }
14    }
15    ...
    }
```

修改程式碼片段表單

【0049】 用來產生交易〔Ｔｒａｎｓａｃｔｉｏｎ〕的演算法如下〔範圍為*ｒ*〕：

【0050】 A1.對於每個修改的程式碼 c_i，找到包含該程式碼的第一層方法〔Ｍｅｔｈｏｄ〕或函式，如修改程式碼片段表單中的ｐｒｏｇＭｓｇ（）。

【0051】 B1.建立ｐｒｏｇＭｓｇ（）的程式結構樹〔Ｐｒｏｇｒａｍ Ｓｔｒｕｃｔｕｒｅ Ｔｒｅｅ，ＰＳＴ〕，並移除其中非出現在關鍵詞集合 S^K 與識別字集合 S^I 中的關鍵詞與識別字，並令依此所建的ＰＳＴ為 pst_i。

【0052】 C1.將此 pst_i 以前序表示法〔ｐｒｅｏｒｄｅｒ ｔｒａｖｅｒｓａｌ〕表示成一有序字串的集合〔ａ ｓｅｑｕｅｎｃｅ ｏｆ ｓｔｒｉｎｇｓ〕，並令此有序字串集合為 p_i。

【0053】 D1.如果此有序字串集合為 p_i 的節點數〔ｎｏｄｅｓ〕（不包含－１）小於（或等於 r），則 p_i 即選為交易〔Ｔｒａｎｓａｃｔｉｏｎ〕，否則由此有序字串集合中挑出 r 個節點數〔ｎｏｄｅｓ〕作為交易〔Ｔｒａｎｓａｃｔｉｏｎ〕。

【0054】 E1.此步驟一直持續到所有的修改程式碼〔ｍｏｄｉｆｉｅｄ ｃｏｄｅｓ〕皆涵蓋在所選的交易中為止。

【0055】 其中，在上述步驟Ｄ1中，為確保所挑選的交易〔Ｔｒａｎｓａｃｔｉｏｎ〕能包含修改的程式碼，因此在挑選時要確定位於修改的程式碼 c_i 的前面與後面的節點數要相近，亦即修改的程式碼 c_i 能儘量位於交易的中間節點。

【0056】 經由上述演算法所挑選出的修改程式碼交易〔Ｔｒａｎｓａｃｔｉｏｎｓ〕範例，如下所示，其中*ｌｉｎｅ*表示修改的行數。

No	*obtained transaction*	*line*
pst_1	{*InfoMerge*, *PaoUtil*, *if*}	244
pst_2	{*for*, *PaoUtil*, *if*}	251
pst_3	{*InfoMerge*,-1,*PaoUtil*,-1,*InfoMerge*}	255

包含修改紀錄的交易(Transaction)範例表

【0057】 二、個人化程式設計樣式〔Ｐｅｒｓｏｎａｌ Ｃｏｄｉｎｇ Ａｃｔｉｖｉｔｙ Ｐａｔｔｅｒｎｓ，ＰＣＡＰ〕：

【0058】 其係利用由該該程式設計活動資訊蒐集〔Ｃｏｄｉｎｇ Ａｃｔｉｖｉｔｙ Ｃｏｌｌｅｃｔｉｏｎ，ＣＡＣ〕步驟所選定的交易集〔Ｔｒａｎｓａｃｔｉｏｎ Ｓｅｔｓ〕來找出個人化程式設計樣式〔ＰＣＡＰ〕，其目的是透過找出程式設計人員在程式設計過程中較常出現的程式樣式〔Ｐａｔｔｅｒｎｓ〕。其用於找出該個人化程式設計樣式〔ＰＣＡＰ〕的演算法主要由資料探勘技術中的關聯性規則改進而來，因此需要先指定最小支持度〔ｍｉｎｉｍｕｍ ｓｕｐｐｏｒｔ〕s，假設所有關鍵詞與識別字的集合以I_1表示，則$I_1 = S^K{}_1 \cup S^I{}_1$，用來產生關鍵詞與識別字的頻繁集〔ｌａｒｇｅ ｉｔｅｍｓｅｔｓ〕的演算法如下所示：

【0059】 A2.計算I_1中每個單一項目的支持度〔ｓｕｐｐｏｒｔ〕(不包含－１)，計算方式為出現於修改程式碼交易〔ｔｒａｎｓａｃｔｉｏｎ〕中的次數〔每個ｔｒａｎｓａｃｔｉｏｎ僅計算一次〕，僅保留支持度大於s的項目，並以L_1表示第１階頻繁項目集〔ｆ

irst－level large itemset〕。

【0060】 B2.產生第2階有序的項目集〔second－level ordered itemset〕$I_2=L_1\times L_1$，其中每個項目（l_i，l_j）的支持度〔support〕則是視（l_i，＊，l_j）出現在修改程式碼交易〔transaction〕中的次數，其中”＊”表示l_i與l_j之間可以初現一個或是多個其他項目。

【0061】 C2.由A2與B2所產生第i組有序的項目集〔i－th ordered itemset〕依照$I_i=L_{i-1}\times L_1$的規則來產生，而第i階頻繁項目集〔i－th largest itemset〕L_i則是由I_i選出〔大於s者〕，此步驟一直重複到沒有頻繁項目集產生為止〔亦即L_{i+1}為空集合〕。

【0062】 由上述包含修改紀錄的交易(Transaction)範例表所示，其中的$S^K_1=\{if，for\}$〔同時出現2次〕，$S^I_1=\{InfoMerge，PaoUtil\}$〔同時出現2次〕，則$I_1$包含4個項目{$InfoMerge，PaoUtil，if，for$}，而$I_1$每個項目的支持度〔support〕分別為2／3、3／3、2／3與1／3，若$s=0.6$，則由於for的支持度僅1／3，故可以移除，因此$L_1=\{InfoMerge，PaoUtil，if\}$。

【0063】 而在該步驟B2中L_1主要是來自$I_2=L_1\times L_1$，請參閱下

列 I_2 項目集所示，其係為由 L_1 所產生的項目集〔itemsets〕I_2 所示，其中每個 I_2 項目所出現的次數表示有多少個交易〔transactions〕包含該項目，其中兩個項目（I_i 與 I_j）的相對階層位置〔relative position〕也要相同，用於比對兩項目的相對階層位置是計算兩者之間向下〔down level〕與向上〔up one level〕層級抵銷後的層級要相同，例如在{if，a_1，－1，for，a_2，－1，a_3}與{if，a_1，a_2，－1，a_3}中的 if 與 a_3 的相對位置為相同，而所找出的程式設計樣式〔pattern〕可以表示為{if，－1，a_3}，當然樣式{if，－1，a_3}樣式{if，a_3}為不同的樣式。因此 I_2 項目若以 $s = 0.6$，則僅有（PaoUtil，＊，if）保留，其他則移除，所找出的最後頻繁項目集 L_n 即是程式設計人員在修改程式過程中頻繁出現的樣式〔patterns〕，此頻繁出現的樣式則可以視為個人化程式設計樣式〔PCAP〕。

No	I_2	count
1	(*InfoMerge*, *, *InfoMerge*)	0
2	**(*InfoMerge*, *, *PaoUtil*)**	1
3	(*InfoMerge*, *, *if*)	0
4	(*PaoUtil*, *, *InfoMerge*)	1
5	(*PaoUtil*, *, *PaoUtil*)	0
6	**(*PaoUtil*, *, *if*)**	**2**
7	(*if*, *, *InfoMerge*)	0
8	(*if*, *, *PaoUtil*)	0
9	(*if*, *, *if*)	0

I_2 項目集

【0064】 三、個人化的程式設計模型〔Personal Codi

ng Activity Models，PCAM〕：

【0065】 其主要結合由程式設計人員的修改程式碼中找出的產出個人化程式設計樣式〔PCAP〕與其執行結果來建立個人化的程式設計模型〔PCAM〕，並利用該模型來評估後續程式碼產出的品質，提供程式設計人員建議的程式設計活動與不建議的程式設計活動。

【0066】 如該個人化程式設計樣式〔Personal Coding Activity Patterns，PCAP〕步驟所述，修改程式碼後的執行結果由程式設計人員進行紀錄的工作，而此執行結果可作為修改程式碼交易〔Transaction〕的執行結果〔Execution Results〕，而此執行結果可以指出當次修改活動是否沒問題，請參閱下列用於評估修改程式碼的屬性〔attributes〕表所示，其中嚴重性〔*severity*〕於發現問題時由程式設計人員決定，而問題形式〔*defect_type*〕則表示邏輯錯誤〔logic error〕或是資源錯誤〔resource error〕。

Attributes	Description
severity	The severity of the defect, denoted as minor or major (from 1 to 5).
defect_type	The type of defect, such as logic or device error.
deffort	The effort required to detect the defect.
reffort	The effort required to remove the defect.

用於評估修改程式碼的屬性表

【0067】 該個人化程式設計樣式〔PCAP〕可視為個人化程式設計

規則〔ｒｕｌｅ〕的前項〔ｌｅｆｔ－ｈａｎｄ ｓｉｄｅ〕，而執行結果〔Ｅｘｅｃｕｔｉｏｎ Ｒｅｓｕｌｔｓ〕可視為後項〔ｌｅｆｔ－ｈａｎｄ ｓｉｄｅ〕，兩者可以產生個人化程式設計模型〔ＰＣＡＭ〕，要產生規則需要先將執行結果利用 K-means 分群法將其分成數個群〔ｃｌｕｓｔｅｒｓ〕，例如 $D=\{d_1,\cdots,d_n\}$ 表示修改程式碼所產生的錯誤〔ｄｅｆｅｃｔｓ〕，而這些錯誤可以分成 $E=\{e_1,\cdots,e_m\}$ 等群〔ｄｅｆｅｃｔ ｃｌｕｓｔｅｒ〕，則每個修改程式碼交易〔ｔｒａｎｓａｃｔｉｏｎ〕可以表示成 $pst_i=(a_{i1},\cdots,a_{in};E_i)$，其中 $E_i\subseteq E$，表示該修改程式碼所產生的錯誤群〔ｄｅｆｅｃｔ ｃｌｕｓｔｅｒｓ〕，請參閱下列交易與所產生的錯誤表列所示，其中 d_1 與 d_3 屬於群 e_1，而 d_2 則屬於群 e_2。

No	*obtained transaction*	*defect*	*defect cluster*
pst_1	{*InfoMerge, PaoUtil, if*}	*None*	*None*
pst_2	{*for, PaoUtil, if*}	d_1,d_2	e_1, e_2
pst_3	{*InfoMerge*,-1,*PaoUtil*,-1, *InfoMerge*}	d_3	e_1

交易與所產生的錯誤表列

【0068】 用於產生該個人化的程式設計模型〔ＰＣＡＭ〕的演算法如下所示，其中用於挑選規則的最小信賴度〔ｍｉｎｉｍｕｍ ｃｏｎｆｉｄｅｎｃｅ〕為 c：

【0069】 A3.計算錯誤群內〔ｄｅｆｅｃｔ ｃｌｕｓｔｅｒ〕E每個元素的支持度〔ｓｕｐｐｏｒｔ〕，並挑選支持度大於 s 的項目，並表示為 L_E。

【0070】 B3.計算$L_n \times L_E$〔L_n表示ＰＣＡＰ〕中所有項目的信賴度，並挑選信賴度大於c的項目，則此項目為所產生的規則，令其表示為R。

【0071】 其中$r_i \in L_n \times L_E$的支持度是由計算（l_1，・・・，l_n）出現於交易集合中的次數而得，若一個規則〔ｒｕｌｅ〕r_i的前項〔ｌｅｆｔ－ｈａｎｄ　ｓｉｄｅ〕出現次數為n_t，而總交易數〔ｔｈｅ　ｎｕｍｂｅｒ　ｏｆ　ｔｒａｎｓａｃｔｉｏｎｓ〕為m，且此規則r_i出現於無錯誤交易的次數〔ｐｏｓｉｔｉｖｅ　ａｐｐｅａｒａｎｃｅｓ〕為n_p，則此規則r_i的支持度為n_t／m，且正面預測〔預測為無錯誤〕的信賴度為n_p／n_t，而所得到的規則可以表示為$R=\{r_1, \cdots, r_n\}$即可視為個人程式設計模型〔Ｐｅｒｓｏｎａｌ　Ｃｏｄｉｎｇ　Ａｃｔｉｖｉｔｙ　Ｍｏｄｅｌｓ，ＰＣＡＭ〕。

【0072】 因此，如上述交易與所產生的錯誤表列所示，若令$s=0 \cdot 3$，且$c=0 \cdot 6$，而錯誤群〔ｄｅｆｅｃｔ　ｃｌｕｓｔｅｒ〕為$E=\{e_1, e_2\}$，則因為e_1與e_2的支持度〔會產生e_1與e_2的交易〕分別為２／３與１／３〔大於０．３〕，而ＰＣＡＰ為$L_n=\{(PaoUtil, if)\}$，因此$L_n \times L_E$所產生的規則為（$PaoUtil, if; e_1$）與（$PaoUtil, if; e_2$），其信賴度分別為２／２與１／２，因此僅有（$PaoUtil, if; e_1$）得以保留，因為其信賴度大於０．６。

【0073】 以下將本發明應用於財政部高雄關稅局所開發差勤管理系統（Attendance Management System for Kaohsiung Customs，AMS-KC）進行驗證實用性，此AMS-KC系統發展自2012／10至2013／03，請參閱下列修改程式碼的紀錄〔modified code fragments〕表所示，其中修改程式碼交易交易〔transaction〕的長度設為11，所使用的識別字〔identifiers〕包含類別名稱〔classes〕，若conn是類別DBConn，則conn.execsql(”insert ･･･”）可表示為DBConn.execsql（S），其中S表示參數〔String〕，而execsql則為方法。

mno	mid	modified code fragment
1	1	if return -1 if Leave.LeaveOver(S,S,S) return -1 if return -1 if Leave.addBill(S) if DBConn.execsql(S) -1 -1
2	1	while Integer.parseInt(S) if Leave.LeaveOver(S,S,S) return -1 if getFieldStr() if DBConn.execsql(S) -1 -1 Leave.chgVals(S,S) -1 amsutil.udpOneRow(DBConn,S)
3	2	parcls.getItem(S) if parcls.getItem(S) parcls.getItem(S) -1 RecordSet.Open(DBConn,S) if return -1 Leave.getFieldStr() parcls.addItem(S,S) parcls.addItem(S,S) amsutil.udpOneRow(DBConn,S)
4	3	if paotime.TimeStrN(S) if if if Leave.LeaveOver(S,S,S) if Leave.chgVals(S,S) paotime.TimeStrN(S) DBConn.execsql(S) -1 -1 -1 Leave.chgVals(S,S)
5	3	while LeaveOver(S,S,S) RecordSet.get(S) if RecordSet.getInt(S) RecordSet.getInt(S) -1 if parcls.getItem(S) RecordSet.getInt(S) -1 DBConn.execsql(S) Integer.toString(int)

修改程式碼的紀錄表

【0074】 請參閱下列修改所產生的錯誤〔defects〕表所示，

其中ｄｎｏ表示編號，ｍｎｏ表示產生此錯誤的ＭＣＦ〔ｍｏｄｉｆｉｅｄ ｃｏｄｅ ｆｒａｇｍｅｎｔ〕，ｓｅｖｅｒｉｔｙ表示嚴重性，所蒐集的資料並經過標準化〔ｓｔａｎｄａｒｄｉｚｅｄ〕與加權〔ｗｅｉｇｈｔｅｄ〕以及分群處理。

dno	mno	severity	deffort (min)	reffort (min)
1	1	4	8	6
2	3	1	5	5
3	5	2	5	6
4	7	1	5	5
5	9	4	12	15
6	9	1	5	5
7	10	5	10	6

修改所產生的錯誤表

【0075】 請參閱下列所得到的交易〔ｉｄｅｎｔｉｆｉｅｄ ｔｒａｎｓａｃｔｉｏｎｓ〕表所示，Ｔｒａｎｓａｃｔｉｏｎｓ １的執行結果為錯誤e_1或e_2，而Ｔｒａｎｓａｃｔｉｏｎｓ ３與５則沒有產生錯誤。而本發明以０．８為最小信賴度〔ｍｉｎｉｍｕｍ ｃｏｎｆｉｄｅｎｃｅ〕以及０．０５為最小支持度〔ｍｉｎｉｍｕｍ ｓｕｐｐｏｒｔ〕所建立的規則〔ｒｕｌｅｓ〕，請參閱下列執行結果規則表所示，其中ＰＣＡＰ表示規則的前項〔ａｎｔｅｃｅｄｅｎｔ〕，而執行結果〔Ｒｅｓｕｌｔ〕則表示規則的後項〔ｃｏｎｓｅｑｕｅｎｔ〕，也就是修改程式碼片段如果符合Ｒｕｌｅ １中的個人化程式設計樣式〔ＰＣＡＰ〕，則可能會產生如e_1與e_2的錯誤，而修改程式碼片段若符合Ｒｕｌｅ ３〔ｏｒ ５〕則可能會產生e_1或e_2的錯誤，主要是因為Ｒｕｌｅｓ ３與５已經合併為規則｛ｉｆ＊－１＊ｉｆ＊Ｌｅａｖｅ．ｃｈｇＶ

als(S，S)}→{e_1 | e_2}〔the consequent is e_1 or e_2〕。

tranid	mno	dno	results
1	1	{1}, {2}	e1, e2
2	2	{3}	e1
3	3	-	-
4	4	{4, 5}, {6, 7}	e2
5	5	-	-

所得到的交易表

rid	PCAP	results
1	if * Leave.LeaveOver(S,S,S) * if * DBConn.execsql(S)	e1, e2
2	if * for * -1 * if Leave.LeaveOver(S,S) * parcls.addItem(S,S)	e2
3	if * -1 * if * Leave.chgVals(S,S)	e1
4	while * -1 * RecordSet.getInt(S) * -1 * if * RecordSet.getInt(S)	e1
5	if * -1 * if * Leave.chgVals(S,S)	e2

執行結果規則表

【0076】 藉由以上所述說明本發明之使用實施可知，本發明與現有技術手段相較之下，本發明主要係除了可以提供程式設計人員於程式設計的過程中選取適切的程式設計活動來提高程式碼的品質外，並可以藉由不建議的程式設計活動來降低程式碼的瑕疵，以能提供更直覺的程式設計流程導引，令程式設計人員更便捷的開發新技術，而在其整體施行使用上更增實用功效特性者。

【0077】 雖然本發明已利用上述較佳實施例揭示，然其並非用以限定本發明，任何熟習此技藝者在不脫離本發明之精神和範圍之內，相對上述實施例進行各種更動與修改仍屬本發明所保護之技術範疇。

【0078】 綜上所述，本發明實施例確能達到所預期之使用功效，又其所揭露之具體方法，不僅未曾見諸於同類產品中，於申請前文獻中

亦未發現有相同技術存在在先，誠已完全符合專利法之規定與要求，爰依法提出發明專利之申請，懇請惠予審查，並賜准專利，則實感德便。

【符號說明】

【0079】 無