國 立 交 通 大 學
資 訊 工 程 學 系

碩 士 論 文

程式設計人才能力管理系統之設計

The Design of Programmer's Capability Management System

研 究 生：莊志良

指導教授：鍾乾癸

中 華 民 國 九 十 三 年 六 月

程式設計人才能力管理系統之設計

The Design of Programmer's Capability Management System

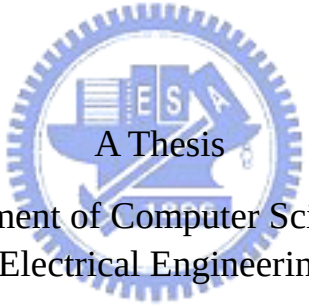
研 究 生：莊志良 Student :Jih -Liang, Juang

指導教授：鍾乾癸 Advisor : Chyan-Goei, Chung

國 立 交 通 大 學

資 訊 工 程 學 系

碩 士 論 文



A Thesis

Submitted to Department of Computer Science and Information Engineering College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science and Information Engineering

June 2004

Hsinchu, Taiwan, Republic of China

中華民國 九十三 年 六 月

程式設計人才能力管理系統之設計

研究生：莊志良　　指導教授：鍾乾癸

國立交通大學資訊工程系碩士班

摘要

軟體專案的開發是高度知識與人力密集之工作。軟體工程師的能力是影響軟體開發之重要關鍵。企業缺乏對於程式設計師能力了解，常會產生在專案的工作分配上無法適才適用，而導致專案失效，成本過高，進度落後等問題。

軟體工程師的能力可由其工作成果的表現來評估，在軟體開發平台盛行的今日，軟體工程師的工作成果應可由開發平台的執行過程取得，此種取得方式可使軟體人才專長能力資料更精確與即時，廖元誠學長依此而提出一套具知識管理與能力管理之軟體開發平台，惟此平台中強調工作成果結果之取得方式，而對專長能力資料庫之建立不夠嚴謹，本研究旨在此平台下建構專長能力管理子系統，並以程式設計師之專長能力管理為研究對象

程式設計師的能力可依程式語言及應用領域分成兩種類別。並利用效率、品質及工作經驗作為該能力成熟度的判定，並更進一步地利用分級的概念將每項能力分級來幫助能力管理與應用，如此一來則可建立一個程式設計師能力模型並可依據此而得知各人員之專長所在。

由程式設計師之能力精進分析，發現程式偵錯是影響程式設計師工作效率與能力提升的重要關鍵，且程式設計師所犯的錯誤大都是前人曾犯過，因此提出錯誤知識庫之構想。

本研究依此構想，在廖元誠所開發的軟體開發平台建立專長能力管理資料庫，錯誤知識庫及相關介面，而構建出此平台之專長／能力管理子系統。

此子系統除可協助軟體企業更精確其開發團隊成果之能力，以便以最經濟方式及時完成專案外，錯誤知識庫可減少程式錯誤的發生及降低偵錯時間，並可提供訓練教材精進之用，提升軟體企業之出產力。

The Design of Programmer's Capability Management System

Student: Jih-Liang, Juang Advisor: Chyan-Goei, Chung
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Chiao-Tung University

Abstract

The development of the software project is a knowledge- and labor-intensive job. By the improving of the efficacy and low price of information related product, the requirements of variant software becomes more and more, and the function of software is being more and more complicated as well. This situation cause problem of the long development days, late time-to-market, bad productivity and low quality during the software development. The concept of integrating the software knowledge management and the auxiliary tools into software development process helped us solve many problems during the software development, but during our research, we found the programmer's capability is the key point that effects the software development. If the cooperation is lack of understanding the capability of the programmer, some problem might occurs, such as we cannot assign the right job to the right man, and we cannot find experts to help solve our problems and the expert dose exist. In present, the mostly popular capability model is PSP and P-CMM. If an software engineer was granted by the PSP or P-CMM, It represent the engineer is capable of processing software development process, but it cannot help us to picture the person's whole skill. In order to improve the software development productivity and quality, it is necessary to provide a capability model with capability and skill management, so the thesis will present a programmer's capability model and system to help solve this problem.

In the part of the capability model establishment, programmer's capabilities can be separated into two categories, program language and application domain. The maturity of each capability can be judged by the development productivity, software quality and work experiences; furthermore according to the maturity, we will rank the capability into four levels. In this way, we can establish a programmer's capability model and in accordance with the model, we can figure the whole skills a programmer has.

The thesis will use the way of implement to integrate the capability model into the software workbench by establishing a defect knowledge base and capability database. The defect knowledge base helps programmers to improve their capability through defect detecting, defect solving. The capability database aid project manager picking the right person for the project development, and in the same time, it also provides an expert query user interface to help the programmer solve their difficulties during coding.

To bring up the concept of programmer's capability model can provide improvement of the software development productivity and software quality from the point of point of view of an engineer, and this is the main contribution of this thesis.

誌謝

這兩年多來，鍾乾癸老師的教導讓我學到了很多。老師淵博的學識，每每可以在研究發生困難時，幫我們指引出正確的方向，但除此之外我覺得老師的嚴以厲己及寬以待人的做事態度給我的影響更大，老師的身教讓我了解了無論在學習和處事上，惟有「用心」、「盡力」才可能有所成就。因此在這一篇論文完成時，第一個要感謝的即是我最尊敬的鍾老師。

其次要感謝的是靜汶學姐及祖望、偉聖、鼎新、大任、衍谷等好同學及實驗室的學弟們，在我失意、需要幫助的時候，他們總是不吝於伸出援手，如果少你們的幫忙，我也沒有辦法完成我的碩士論文。

還要感謝我的家人們，謝謝你們這些日子以來在背後不斷地給我鼓勵，這也是支持我不斷成長的動力。僅以此論文獻給所有我有提到及沒有提到幫助我的人們，也謝謝各位。

目錄

圖目錄

第1章　緒論

1.1. 研究背景與動機

資訊硬體產品的效能及功能大幅提升及價格大眾化，導致各類軟體需求愈來愈多，功能也漸趨複雜及多樣化，因而造成發展軟體時常遭遇開發過程費時、交貨延遲、產能不易提升，且品質不易確保等問題，甚至產生了**軟體危機**[1]，

為了解決這些問題，我們在過去已體認到軟體知識管理的重要性，並展開一系列的研究對軟體知識種類與型態、軟體知識螺旋[2]、軟體重用元件[3][4][5][6][7]等，此方法的提出，解決了許多軟體開發過程中之困難，如：公司必須決定所要發展的產品、專案經理要挑選適當的成員，決定使用的發展程序、方法、時間與可應用的技術等、設計師要決定需使用的演算法、程式撰寫人員要決定使用的函式、變數、測試員要設計一系列的測試個案等問題。除此之外，還進而提出整合知識管理[8]、整合工具、流程及能力管理之軟體開發平台[9]的概念。

在研究過程中發現，軟體發展本質為以人為本之工業，軟體開發人員的能力是軟體開發中最重要的部分，每一個成功的專案一定是由一群具有專業知識的人員共同合作的成果。因此將基於上述之背景，針對軟體開發人才之能力之研究是未來不變的驅勢。然而有幾個問題一直困擾著每一個軟體專案的開發：

- 每一個專案之性質均不相同，需要的專業知識也不同，如何知道哪些程式設計師具有這些專長，以及如何挑選具備適當專業知識的人員加入團隊並指派可勝任的工作。
- 每位專案成員擁有相同之軟體開發能力，但是卻無法得知那一位成員之能力較好，缺乏人才能力比較的機制也是一個嚴重之問題。

- 專案執行過後，如何評估每位成員的表現，並且為其不足較弱的專業領域安排適當的訓練課程，以提升其能力。

目前研究中關於軟體人員能力管理之方法，最常使用的有 PSP[10]與 P-CMM[12]，通過 PSP 與 P-CMM 等級認證，代表一位軟體人才在對於開發流程之熟悉已有一定之程度以上，但是它的缺點為不能告訴我們這一位軟體人才的專長所在。也有人提出將人員的專業能力紀錄在資料庫中或者是以建立社群的方式[13][14]來分享知識，然則這些研究卻又缺少了合理公正的統計數據來作為能力評鑑有力的佐證。

因此為了解決上述三個問題，提出一個能幫助專長管理的軟體人才能力模型是必要的，不過在因為時間與人力之限制，本研究無法完成所有的軟體開發角色之能力模型之設計，因此本研究之能力模型將先由程式設計師著手，於日後再推廣至其它的開發人員。

本研究進行之方法如下：

1. **程式設計師能力之分類：**找出一位程式設計師之能力分為那些方面，而那些能力對於程式設計師而言是重要的，這是我們所要探討之重點。

2. **程式設計師能力成熟度定義:** 在評估的方法上，常常以等級來區分評估的結果，這是一種簡明有效表示方式，也廣泛運用於軟體評估上如 CMMI 與 PSP，在能力模型中我們也採用相同的方法來進行，我們將以四種不同的能力等級來區分程式設計師之專長的成熟度(Maturity)，分別是新進 、一般、菁英與專家級。以 Java 程式語言為例：

    - 新進:對於 Java 程式設計為初學或是僅了解入門的人士，即了解 Java 語言，但尚未使用 Java 語言進行專案開發。
    - 一般:對於 Java 程式已有充分了解，並已運用此能力開發一般程式

的人員。

- 菁英: 對於 Java 程式設計已有相當的經驗，已可解決 Java 程式大部份的困難問題，並可衍生開發新的技術或是應用。
- 專家: 對於 Java 程式設計已有相當豐富的資歷，可供其它人員諮詢並提供指導的人員。

3. **程式設計師能力之管理：**有了能力模型之後，我們即可以客觀的評估每一位專案成員的能力成熟度，所謂能力的管理，就是在專案不同的階段，都能依其工作評估其能力，進而提出適當的能力提升計畫，如透過缺失管理、和培訓課程等，來幫助程式設計師能力之提升。

4. **錯誤知識庫之建立：**藉由討論程式設計師能力之精進，我們發現對於軟體缺失的管理為增進程式設計師能力的良方。因此在本研究中提出一錯誤知識庫之建立。

5. **整合專案人才挑選於 Software Workbench：**將能力模型之概念整合入 Software Workbench 之中，並提供專案經理新增專案及幫助專案挑選人員等機制。

1.2. 章節介紹

本論文章節安排如下，首先在第二章介紹能力管理之相關研究。第三章則進行程式設計師能力模型之建立。將從能力的分類、成熟度指標和分級三個部分來建立起我們的能力模型。第四章則為與程式設計師能力息息相關之錯誤知識庫之設計，並說明使用介面與實作應用範例。第五章，則為程式設計師能力庫之設計與實作。第六章則是本研究之結論及未來之研究方向。

第2章　能力管理相關研究

本章首先介紹相關之軟體人才能力管理之相關研究。2.1 節首先介紹本研究之基礎一個結合知識管理與能力管理之軟體發展平台（Software Workbench）之架構。2.2 節則將範圍聚焦在軟體人才能力管理之相關研究上。

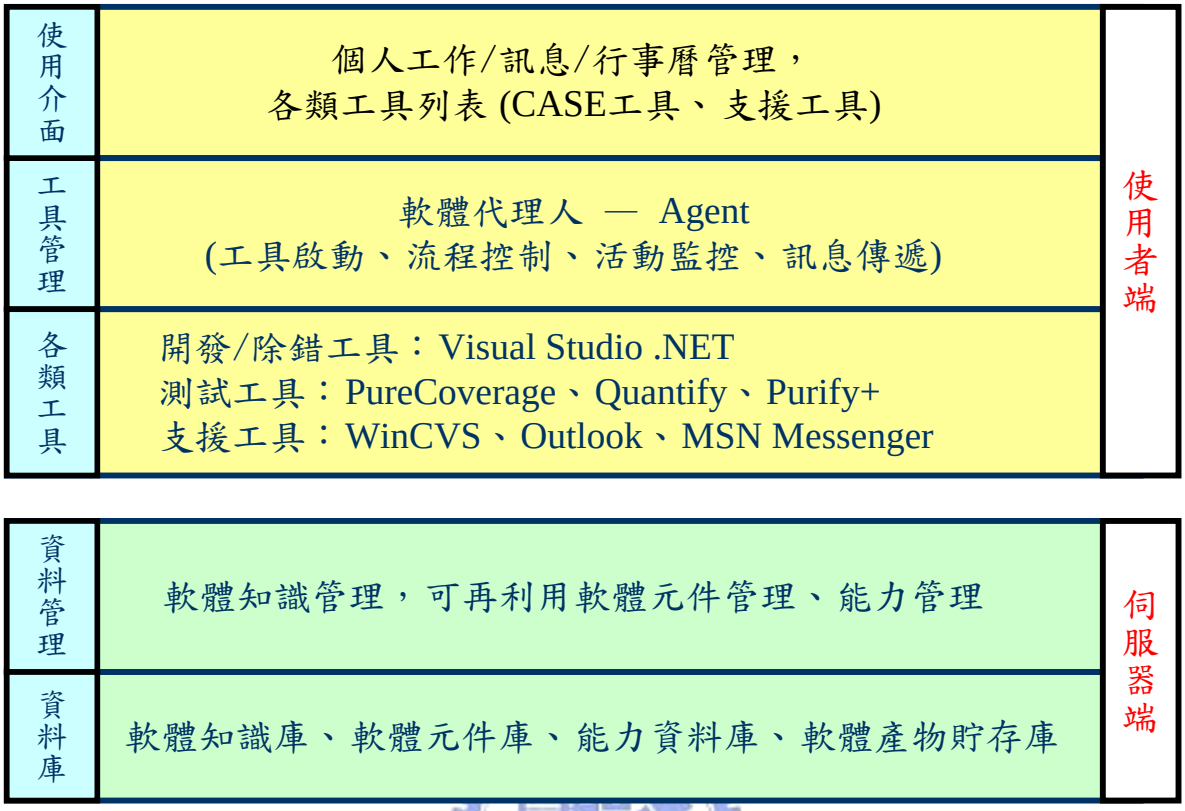
2.1. 知識管理與能力管理之軟體發展平台簡介

軟體產業是一項人力與知識密集的產業 (Birk, et. al., 1999)，軟體發展更是一個知識不斷創新的過程，因此造就軟體知識的多樣化且快速的成長。軟體發展的過程需經歷多個不同的階段 (phase)、活動 (activity) 與產物 (artifact)，需要眾人的參與分工合作才得以完成，不同於生產與製造的程序，一旦決定了生產與製造的流程，生產線的工程人員只要依照規定執行，毋須任何進一步的決策；軟體發展過程中的許多決策需仰賴眾人的集思廣益，例如：公司得選擇要開發何種產品、專案管理者得挑選合適的成員並選擇適用的發展程序、方法與應用的技術、設計師得依需求選擇合適的演算法、程式撰寫人員得決定使用的函數與變數、測試人員得設計一系列的測試個案...等，因此軟體發展可說是一項設計類的程序 (design type process)。

上述對於選擇的決定與問題的解決，大多仰賴個人所累積的知識與經驗。當開發的軟體專案越來越龐大且複雜時，團隊成員的溝通與分工合作就顯得格外重要。每個成員提供自己的知識與經驗也敞開心胸學習他人的知識與經驗，正如同俗諺所云：三個臭皮匠勝過一個諸葛亮。因此，如何做好知識的管理，加速知識傳遞與學習的過程，縮短軟體開發的時程，提高軟體的品質，進而降低軟體開發的成本並加快產品上市的時間...等，便成為軟體產業一項重要的議題。

然而解決上述之問題，交通大學系統模擬實驗室廖元誠先生於 2003 年提出

一套軟體開發平台(Software Workbench)[9]，此平台之架構圖如圖 2-1 所示:



圖表 2-1 軟體開發平台架構圖

然而在各種工具的使用與開發流程緊緊地結合在一起來為軟體開發進行輔助。其分別之系統所能提供之功能如下:

- **個人工作管理子系統：**

  負責使用者身份 (帳號與密碼) 的確認；提供個人工作、訊息與行事曆管理的使用者介面與內容更新；配合軟體發展流程，提供適當的輔助工具執行，啟動軟體代理人負責系統與開發工具間的訊息傳遞、使用者介面整合、即時的軟體產物分析與回報等。

- **輔助工具子系統：**

  搜尋本地端可用的輔助工具、產生工具選單以及工具執行的使用者介面，負責傳遞訊息給下層功能單元等；提供團隊成員溝通、協調的相關

支援工具集，軟體發展者撰寫程式碼、編譯與連結的相關開發工具集，軟體發展者與測試人員測試程式單元的相關測試工具集，軟體發展者偵測、排除程式碼錯誤的相關除錯工具集。

- **軟體知識管理子系統：**

  負責提供軟體知識管理執行平台 (支援 HTTP 通訊協定的瀏覽器) 的使用者介面，傳遞訊息給下層功能單元、與個人知識管理子系統進行訊息溝通等；負責軟體知識搜尋、評估、儲存與管理的機制。

- **可再利用軟體元件子系統：**

  負責提供可再利用軟體元件管理執行平台 (支援 HTTP 通訊協定的瀏覽器) 的使用者介面，傳遞訊息給下層功能單元、與個人知識管理子系統進行訊息溝通等；負責可再利用軟體元件搜尋、取用、評估、新增、刪除與管理機制。

- **人員能力管理子系統：**

  負責提供人員能力管理執行平台 (支援 HTTP 通訊協定的瀏覽器) 的使用者介面，傳遞訊息給下層功能單元、與個人知識管理子系統進行訊息溝通等；負責個人能力資訊統計分析、各領域知識/實作經驗/工作效率/問題解決能力等方面專家的搜尋、能力資訊統計分析，以及專案相關資訊統計分析的機制；提供個人能力資訊更新至資料庫，或從資料庫取出的功能。

而 Software Workbench 的完成下列兩項重要的幫助：

- **提出整合開發流程、CASE 工具、軟體知識、再利用軟體元件以及能力管理的整合軟體發展環境架構**

  結合軟體開發流程、CASE 工具、軟體知識、可再利用軟體元件、能力管理，以及自動化輔助等概念，我們提出軟體發展環境的構想，協助軟

體發展者與專案管理者以更效率的方式進行軟體專案開發與人員專長能力管理。

- **以實作階段為對象，設計及製作此整合軟體發展環境之雛形系統**

  根據上述的研究結果，我們以實作階段為對象，設計並製作此整合軟體發展環境之雛形系統，在軟體發展的過程中提供軟體發展者所需的軟體知識、再利用軟體元件，以及監控軟體發展者的活動，提供發展者與管理者相關專長及能力之分析與建議。

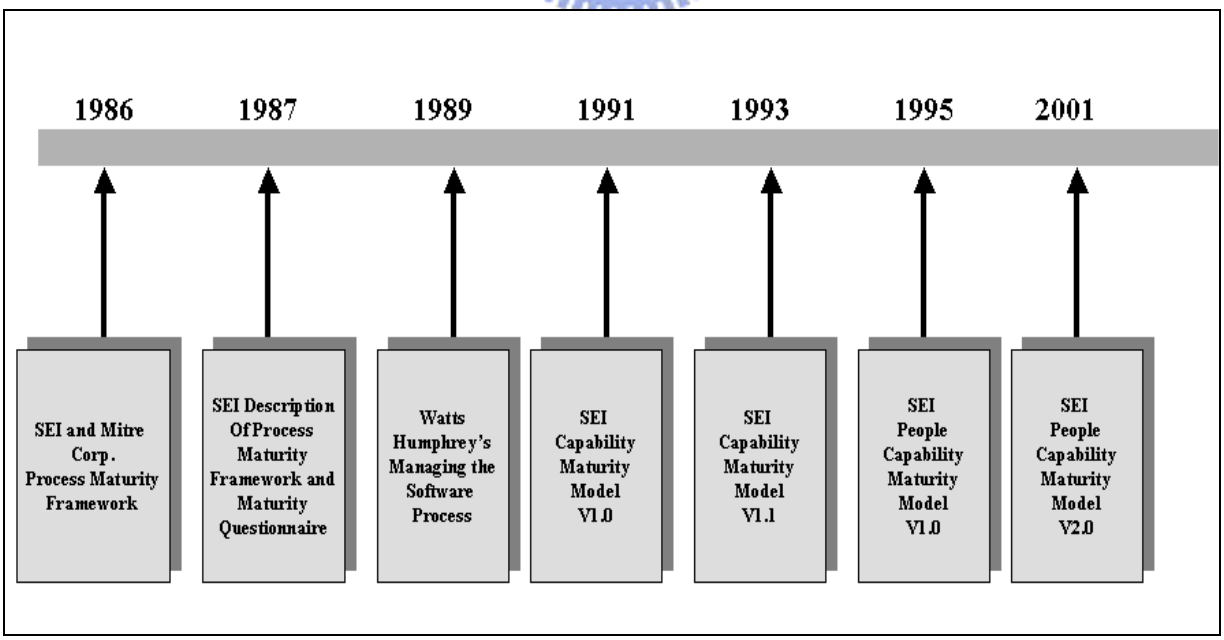
然而 Software Workbench 在整合工具與軟體開發流程上有極大的供獻，也針對了軟體開發的「人員能力」進行了初步之管理，但在 Software Workbench 之部分只有簡單之人員能力評鑑，因此若在多專長的搜尋上並無法達到效果，且除此之外，Software Workbench 只提供了一個開發平台，若今天人員之能不足以完成工作，則平台也無相關之工作或是機制來幫助人員能力的提昇。因此本研究希望能在軟體開發平台(Software Workbench)之上再建構新增「更完整的能力評鑑機制」及「專長管理機制」及「人員能力提昇的輔助機制」等以更實質的剛度來幫助軟體系統的開發。

2.2. 軟體人才能力管理相關研究

軟體產業是一高度知識與人力密集的產業，人力素質 (能力管理) 的良窳或直接或間接地影響所發展軟體的成敗，能力管理的相關研究主要有Bill Curtis, William E. Hefley, Sally Miller等人提出的P-CMM(Capability Maturity Model for Software) 以及Watts S. Humphrey提出的Personal Software Process – PSP 與Team Software Process – TSP ，及由Philips的Ronald Begeer與Niloy Banerjee所提出的Leadership Approach。而接下來我們將分別針對此四者來進行討論。

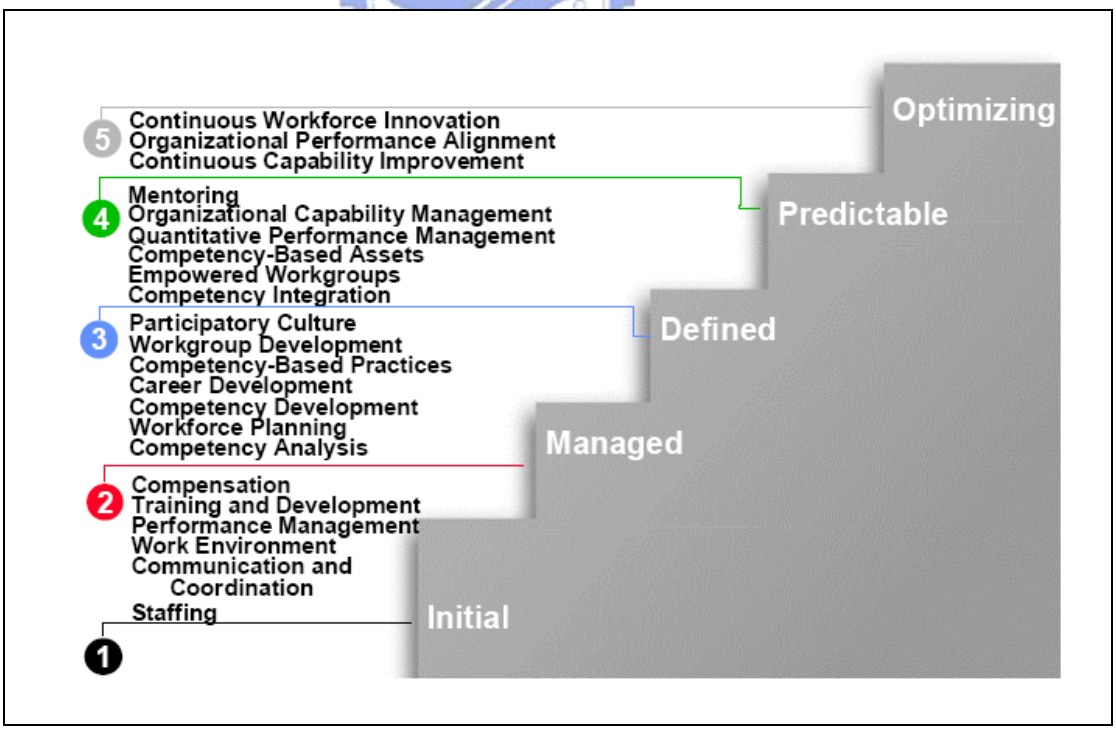
2.2.1. P-CMM

1986 年 11 月，美國卡內基美隆大學（Carnegie-Mellon University）的軟體工程學院（Software Engineering Institute, SEI），在 Mitre 公司的協助下開始發展一個可以幫助軟體開發者改善軟體流程的流程成熟架構（Process Maturity Framework）。1987 年 6 月，SEI 發表了軟體流程成熟架構的簡要描述，接著於當年 9 月出版了基本成熟度之問卷，提供一個工具用以指出軟體業者軟體流程需要改善的領域。經過了四年的使用經驗與努力，於 1991 年 SEI 正式發表 CMM1.0，並於 1992 年 4 月辦理座談會，綜合超過 400 位軟體專家意見，於 1993 年發表 CMM1.1 修正版，歷史沿革如圖 2-2 所示。SW-CMM 主要是針對軟體生產流程發展出來，作為全面品質管理與流程改善的架構；換言之，軟體能力成熟模式（Software Capability Maturity Model, SW-CMM）主要將全面品質管理應用到軟體開發與維護，用以提昇組織的軟體開發的管理能力以達到成本、時程、功能與品質等目標。



圖表 2-2P-CMM 演進圖

軟體組織為了要持續改善他們的開發效能，可以把所有的改善工作集中在三個相關方面，亦即「人員」、「流程」以及「技術」。過去這些軟體組織藉由 SW-CMM 的幫助，已經在他們的軟體開發流程過程當中，達成各種具成本效益與持續改善之實務工作。然而，許多的軟體組織卻發現到，在他們實行這些持續改善工作時，同時也必須要對他們在管理、培養及運用其軟體開發維護人員的方式上，作一些重要的改變，而這些改變是在軟體能力成熟度模式所沒有完全考慮到的。迄今，這些軟體組織所做的改善工作仍是著重在「流程」或者是「技術」上而不是在「人員」身上。1988 年 SEI 首次於學術研討會中提出對評估人員能力成熟度模式（People Capability Maturity Model, P-CMM）之研究議題，1993 年 7 月正式成立 P-CMM 的諮詢委員會，著手起草 P-CMM 模式內容，並於 1995 年 9 月正式公布 P-CMM 1.0 版，並於 2001 年 7 月發佈 P-CMM 2.0。P-CMM 和 SW-CMM 類似，均是一個循序漸進的改善方向，讓一個軟體組織由混亂、不一致地執行工作，轉變為成熟、有制度並持續地改善人員知識技能發展的境界。



圖表 2-3P-CMM 分級圖

圖 2-3 所示是 P-CMM 模式之五個成熟度等級與 20 個關鍵程序領域(Key

Process Areas, KPAs)。而在此將先針對 P-CMM 各個等級進行介紹:

1. **初始等級(Initial):**處於此一成熟度等級的組織，其各種勞動力活動的成效時常是「不一致的」，也就是說組織雖有提供各式活動的正式表格，例如績效評等表或職位需求表，但在導入支援這些表格的活動上，卻是提供相當少的訓練或指引；這類組織的人並不相信，勞動力實務對他們的實際工作及對組織貢獻的程度有很大的關係，故不會去重視這些實務活動；管理階層通常是根據以往的經驗或是個人的特質來管理員工，並無一套有系統的方法來培養其勞動力之競爭能力。此外，「人員的高流動率」是這類組織的另一項特徵，這是因為當人員覺得在別的公司有較好的工作條件或成長潛力時所造成的。因此，隨著時間流逝在組織內可用的知識與技能並無相對的成長，需再次花費時間把這些知識與技能的職缺給補上，使人無法得知這類組織的勞動力能力等級為何。
2. **重覆等級(Repeatable):**在此一成熟度等級的組織，其主要的目標就是建立一套去執行基本勞動力實務的責任及訓練，確保在每一個單位中的人員，都具有達成目前工作所需的知識與技能。當這些實務活動已經制度化後，這類組織就具有能夠去建立一些改善方法的基本能力；此外這類組織的人員漸漸重視他們在各種勞動力活動中被賦予的責任，為了能更有效率地去執行這些活動，就某些活動開始去發展可重覆利用的方法，例如面談或是建立績效標準等等。
3. **定義等級(Defined):**在此一成熟度等級的組織，已經能夠找出在各單位間共同需求的知識與技能，並利用機會將軟體開發能力「標準化」；因為能夠清楚瞭解其勞動力可用的知識與技能程度，分析後再根據組織營運功能而確認出「核心能力」，並詳細訂出有系統的核心能力發展方案；同時為了將焦點擺在這些核心能力，組織也建立一個參與文化的環境，讓個人或群體能夠參與有關於其工作之決策。此外，這類

組織也會為每個人去規劃其生涯發展策略。

4. **管理等級(Managed):**屬於此一成熟度等級的組織，重要的特徵就是能夠去「量化」得知其勞動力目前的能力，進而去「預測」出未來在開發力能力與各種績效調整之趨勢；同時也發展出一套機制，藉由具有擁有高軟體開發能力的團隊，更有效率地去部署其各種能力。

5. **最佳等級(Optimizing):**強調持續地個人能力改進是處於此一成熟度等級組織的主要特徵，提供的顧問輔導訓練更可以進一步幫助發展個人或團隊之能力。在此同時軟體開發所收集之資料已具有有效性，因此可以利用收集到的資料來找出各種需要創新的勞動力實務或技術，並在試驗成功後提供各種創新的實務或技術給全組織來使用。

每個成熟度層級內部各包含了數目不等的 KPA。所謂 KPA 是指從某一層級要提昇之時所需改進的關鍵行動。P-CMM 在設計改善人員流程之時，將欲改進人員能力之流程分為 4 個主題(Theme)：

1．**Develop capabilities**:此類別之 KPA 主題為改善人員發展程式之能力，建立完整訓練、學習、溝通及顧問機制，來幫助提昇使用者之能力。

2．**Building team & Culture**:包括小組之溝通及小組的建立機制等和工作分配等團隊合作的機制與流程。

3．**Motivating and managing performance**：此部分之 KPA 主要是進行軟體人員量化能力的管理及如為達成量化所建立的機制的實施等。

4．**Shaping the workforce**：此部分則是幫助建立整個工作團隊(workforce)機制的建立，其包括組織的能力管理，團隊建立的計劃，及人員挑選的機制等。

在此將 P-CMM 之 KPA 整理成表 2-1:

	Process Categories

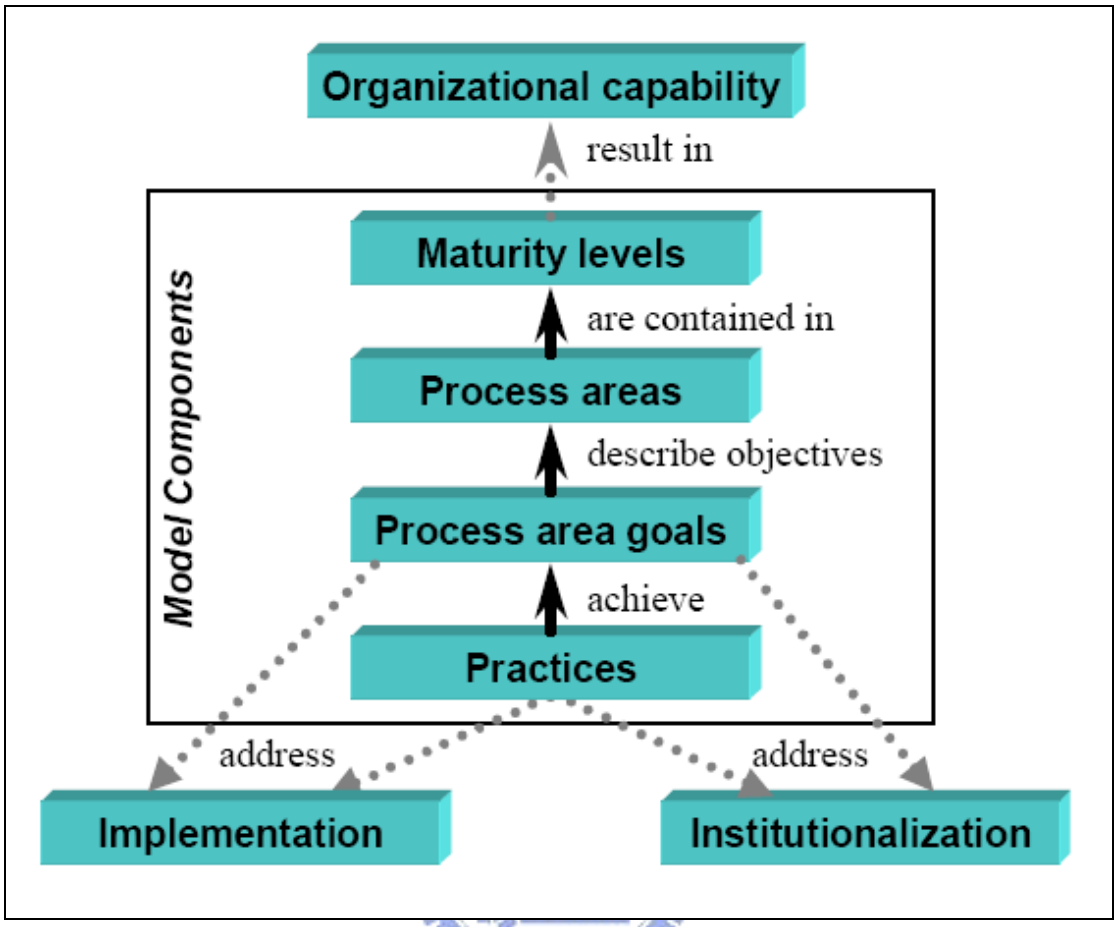
Maturity Levels	Theme 1: Developing Capabilities	Theme 2: Building Teams and Culture	Theme 3: Motivating and managing performance	Theme 4: Shaping the workforce
MATURITY LEVEL 5: Optimizing	Coaching Personal Competency Development	Continuous Workforce Innovation		
MATURITY LEVEL 4: Managed	Mentoring	Team Building	Quantitative Performance management Team-Based Practices	Organizational Capability Management
MATURITY LEVEL 3: Defined	Competency Development Knowledge and Skills Analysis	Participatory Culture	Competency-Based Practices Career Development	Workforce Planning
MATURITY LEVEL 2: Repeatable	Training Communication	Communication Coordination	Compensation Performance Management Work Environment	Staffing
MATURITY LEVEL 1: Initial				

表格 2-1P-CMM 人員能力成熟度表

每一個 KPA 均有不同之目標（Goal）和達成目標所需之工作（Practice），因如表 2-2 示。而在達到了所有的目標之後則完成了此 KPA，而人員在完成了等級中所有的 KPA 之後將會達到下一個能力成熟度等級。而所有人員之能力將會被視為整個組織之能力。(如圖 2-4)。

STAFFING			
Goal 1	Individuals or workgroups in each unit are involved in making commitments that balance the unit's workload with approved staffing.	**P1**	Responsible individuals plan and coordinate the staffing activities of their units in accordance with documented policies and procedures.
		P2	Each unit analyzes its proposed work to determine the effort and skills required.
		P3	Individuals and workgroups participate in making commitments for work they will be accountable for performing.
		P4	Each unit documents work commitments that balance its workload with available staff and other required resources.
Goal 2	Candidates are recruited for open positions.	**P6**	Position openings within a unit are analyzed, documented, and approved.
		P7	Position openings within the organization are widely communicated.
		P8	Units with open positions recruit for qualified individuals.
		P9	External recruiting activities by the organization are planned and coordinated with unit requirements.
Goal 3	Staffing decisions and work assignments are based on an assessment of work qualifications and other valid criteria.	**P10**	A selection process and appropriate selection criteria are defined for each open position.
		P11	Each unit, in conjunction with its human resources function, conducts a selection process for each position it intends to fill.
		P12	Positions are offered to the candidate whose skills and other qualifications best fit the open position.
		P15	Representative members of a unit participate in its staffing activities.
Goal 4	Individuals are transitioned into and out of positions in an orderly way.	**P5**	Individual work assignments are managed to balance committed work among individuals and units.
		P13	The organization acts in a timely manner to attract the selected candidate.
		P14	The selected candidate is transitioned into the new position.
		P16	Workforce reduction and other outplacement activities, when required, are conducted according to the organization's policies and procedures.
		P17	Discharges for unsatisfactory performance or other valid reasons are conducted according to the organization's policies and procedures.
		P18	Causes of voluntary resignation from the organization are identified and addressed.

表格 2-2 Staffing 之 Goal 與 Practice 定義表



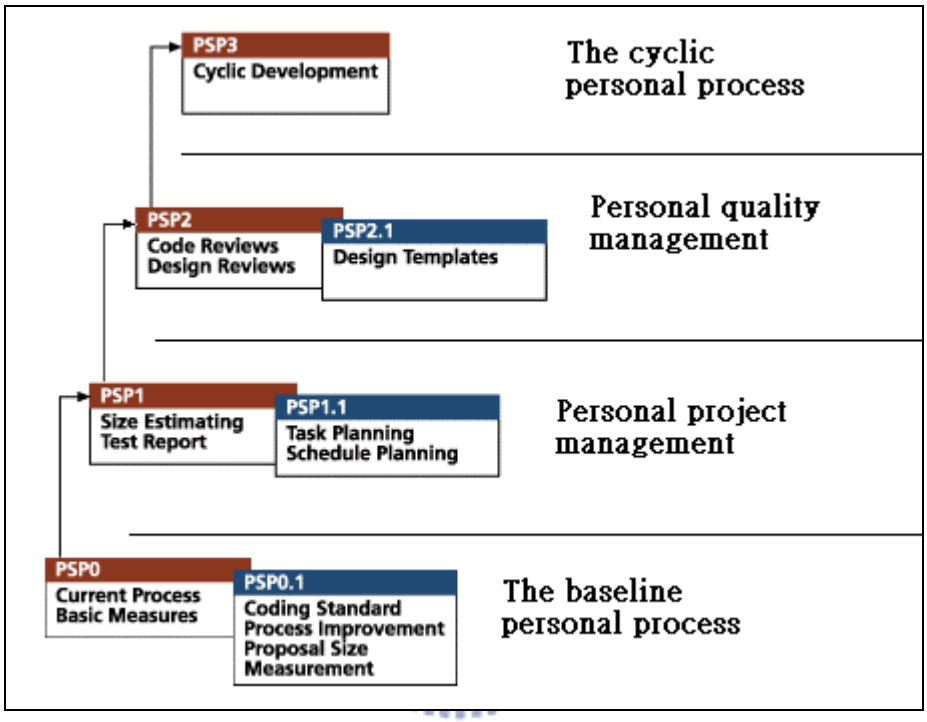
圖表 2-4P-CMM 能力提昇機制

透過 P-CMM 之幫助，企業可以獲得之優點有：(1)描述出人才在軟體開發上之成熟度 (2)持續的對人員能力提升發展提供明確的指示 (3)設定當前改善活動的優先順序 (4)整合人才能力提升發展與開發流程之改善 (5)建立優質之軟體工程文化。

2.2.2. PSP

PSP (Personal Software Process) 是一種可用於控制、管理和改進軟體開發流程之自我持續改進過程，是一個包括軟體發展表格、指南和規程的結構化框架。PSP 與 具體的技術(如不同程式設計語言或工具或者設計方法)是相互獨立，PSP

的原則能夠應用到幾乎任何的軟體工程任務之中。PSP 為提供了下列原則:(1)幫助軟體工程師作出更精確的軟體開發計劃。(2)提供軟體工程師爲改善軟體品質所要採取的步驟。(3)建立度量軟體流程改善的基準。(4)提出流程的改變與軟體工程師能力的關係。同樣的 PSP 也類似 P-CMM 以 KPA 之方式來進行改善。PSP 之流程主要分為 4 種。



圖表 2-5PSP 分級圖

(1) 基本流程建立 PSP0/PSP0.1:

PSP0 的目的是建立個人軟體流程之基礎,工程師學習使用 PSP 的各種表格採集專案的有關資料，通常包括計劃、開發以及維護三個階段，並作一些必要的記錄，如軟體發展時間，缺失類別數和排除缺的失數等，用作爲測量人員在 PSP 過程的基準。

PSP0.1 增加了程式寫作標準、程式規模度量和流程改善建議等三個 KPA，其中流程改善建議表格用於隨時記錄過程中存在的問題、解決問題的措施以及改進過程的方法,以幫助軟體發展人員了解軟體品質和流程之重要。

(2) 個人專案管理 PSP1/PSP1.0:

PSP1 的重點是專案管理，引用 PROBE（PROxy Based Estimating）之估算方法，利用過往之專案資料來預測新專案之大小及所需開發時間，並利用線性回歸方法計算估計參數，確定預測的可信程度。PSP1.1 增加了對專案工作之計劃和專案進度之掌控。

在 PSP1 階段應該學會軟體專案管理，這不僅對大型軟體系統之開發十分重要，即使是小型軟體系統也是不可或缺的。因爲，只有對自己的能力有客觀的評價，才能作出更加準確的計劃，才能實事求是地完成專案之開發。

(3) 個人品質管理 PSP2/PSP2.1:

PSP2 的重點是軟體品質管理，根據程式的缺失建立檢測表，按照檢測表進行設計覆查(design review)及程式覆查(code review)，以便及早發現缺失，以減少缺失造成之損失。

PSP2.1 則論述設計模板(design template)，介紹設計方法，並提供了設計模板、但 PSP 並不強調選用什麼設計方法，而強調設計準則和驗證技術。實施 PSP 的一個重要目標就是學會在開發軟體的早期實際地、客觀地處理由於人們的疏忽所造成的缺失問題。PSP2 引入並著重強調設計覆查和程式覆查技術，一個軟體發展人員所必須掌握的兩項重要技術。

(4) 開發流程循環 PSP3:

PSP3 的目標是把個人開發小程式所能達到的生產效率和生產質量，延伸到大型程式；其方法是採用螺旋式上升過程，即叠代式的開發方法，首先把大型程式分解成小型的模組，然而對每個模組按照 PSP2 及 1 所描述的過程進行開發，最後把這些模組逐步集成爲完整的軟體產品。

由上規納可知 PSP 在人在人才能力評鑑方法方面，提供了 P-CMM 相似的流程分級制度，例如使用者達成 PSP 1 則具備了，「個人專案管理」之能力； 而若達到 PSP2 則擁有「品質管理」部分；而若是 PSP 3 則指工程師擁有開發大型系統之能力。但除此之外 PSP 引進了許多軟體人才能力評鑑之重要概念：

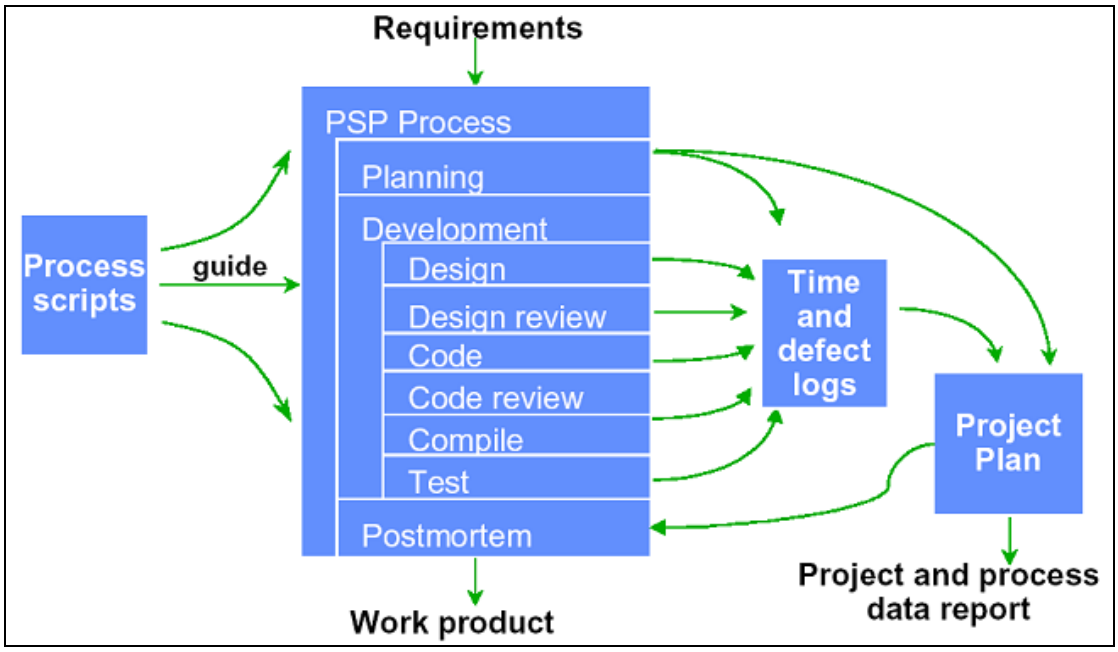
（一）個人專案管理能力：

PSP 制定了一套完整之專案資料記錄格式，其包括，專案設計、實作的文件、程式行數、時間等記錄。讓使用者能針對自我的表現來進行能力調整之工作。

（二）軟體品質控制機制：

PSP 針對軟體品質來進行控制，包括程式碼覆查(code review)機制的建立，和軟體缺失之記錄及統計來量化軟體品質，因此使用者之品質能力表現有了一定之比較標準。

而為了完成上述兩種概念，因此，PSP 也設計了其專案流程機制及專案資料回饋機制，見圖 2-6：



圖表 2-6 PSP process flow

雖然圖中之流程為 Waterfall 但 PSP 並不是一個 Waterfall 的流程，PSP 的流程可以被整合入 TSP 之流程之中，而 PSP 規定其軟體開發所必需之流程，並在對於每個流程所要進行之工作有詳盡的定義和工作指示，及要完成之資料收集。而在各個統計資料下，PSP 也可以幫助使用者評鑑自己各方面的表現情況。

PSP 注重於個人的技能，能夠指導軟體工程師如何保證自己的工作品質，估計和規劃個人的工作，度量和追蹤自我的表現，管理自身的軟體過程和產品質量。經過 PSP 學習和實踐的正規訓練，軟體工程師們能夠在他們參與的專案工作之中充分利用 PSP，從而保證了專案整體的進度和質量。

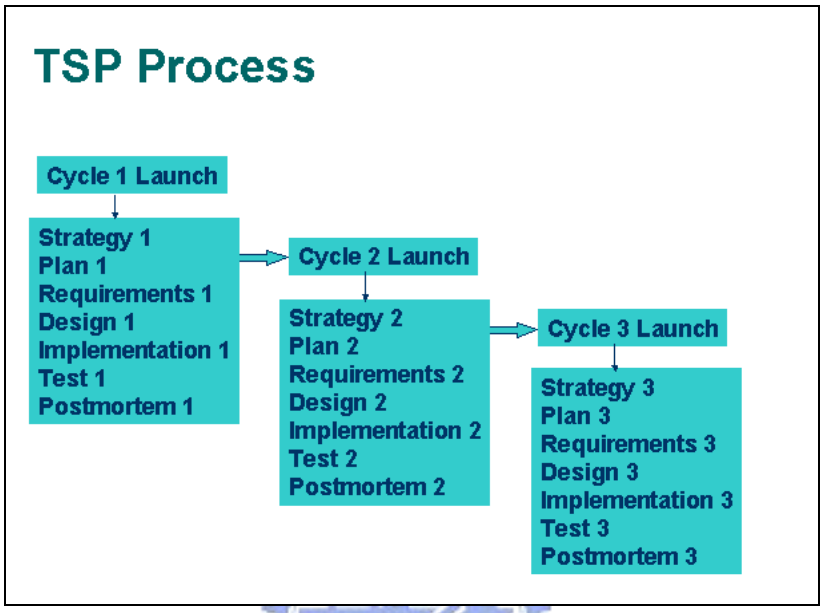
2.2.3. TSP

TSP 是針對團隊軟體過程的定義、度量和改革提出了一整套原則、策略和方法，把 CMM 要求實施的管理與 PSP 要求開發人員具有的技巧結合起來，以按時交付高質量的軟體，並把成本控制在預算的範圍之內。在 TSP 中，講述了如何創建高效且具有自我管理能力的工程小組，工程人員如何才能成爲合格的專案組成員，管理人員如何對群組提供指導和支援，如何保持良好的工程環境使專案組能充分發揮自己的水平等軟體工程管理問題。

TSP 基於以下四條基本原理：(一) 應該遵循一個確定的、可重覆之過程並迅速獲得回饋，這樣才能使學習和改革最有成效；(二) 一個團隊是否有效率，是由明確的目標、有效的工作環境、有能力的指導與領導等四方面因素的綜合作用所確定的，因此應在這四個方面同時努力，而不能偏廢其中任何一個方面；(三) 應注意及時總結經驗教訓，當人員在專案中面臨各種各樣的實際問題並尋求有效的解決問題方案時，就會更深刻地體會到 TSP 的威力；(四) 應注意借鑒前人和他人的經驗，在已經可資利用的工程、科學和教學法經驗的基礎上來規定過程改進的指令。

TSP 一般將一個軟體專案的開發工作分成數個階段，如圖 2-7。任何一個應用 TSP 的專案可以只包括其中的一個階段，也可以包括幾個連續的階段。在專案開始之前，專案團隊應該執行啓動(Launch)過程，對整個任務進行全面地規劃和組織。在每個階段之前，專案團隊應該執行重啓過程，對下一個階段的任務進行規劃。一般來說，如果專案團隊的成員經過了 PSP 的培訓，專案團隊的啓動

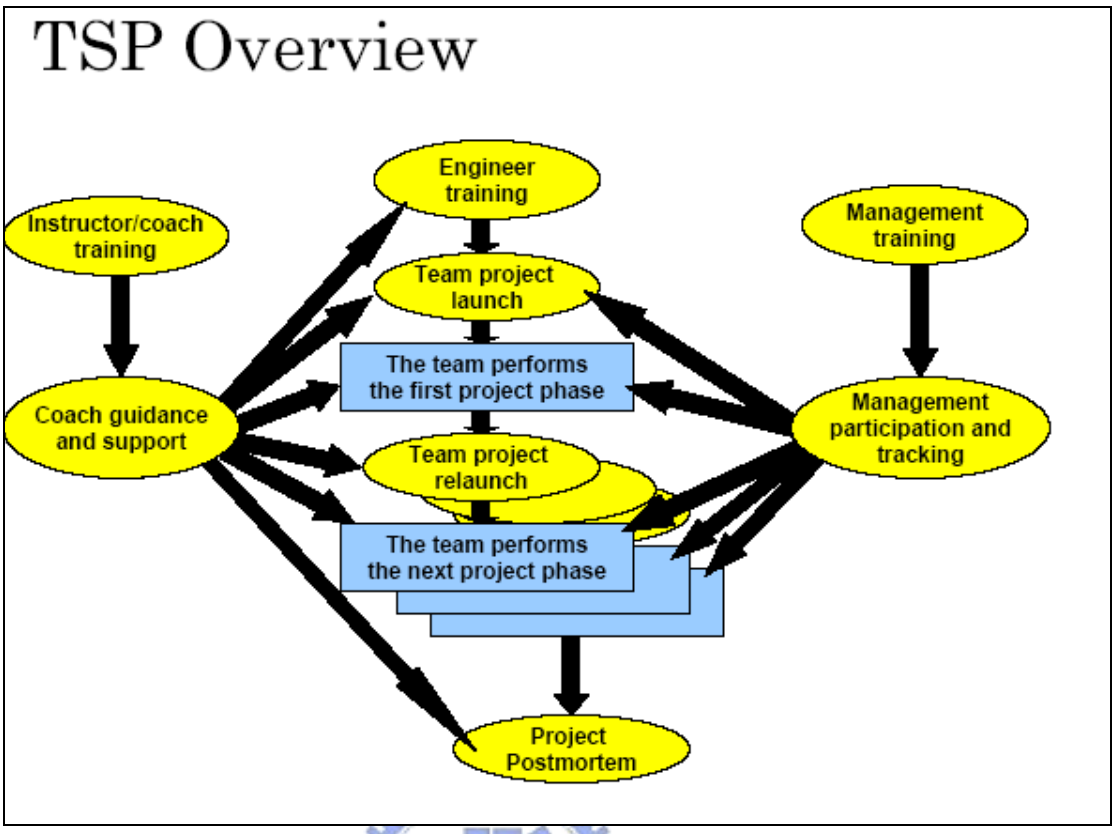
過程約需 3 天時間，重啓過程(Re-launch Process)約需兩天時間。此時，專案同管理人員一起評審專案計劃和分析關鍵風險。在專案已經啓動之後，專案組應每周進行一次專案進展討論會，另外還應及時向有關主管和客戶報告專案的進展情況。



圖表 2-7TSP 專案流程圖

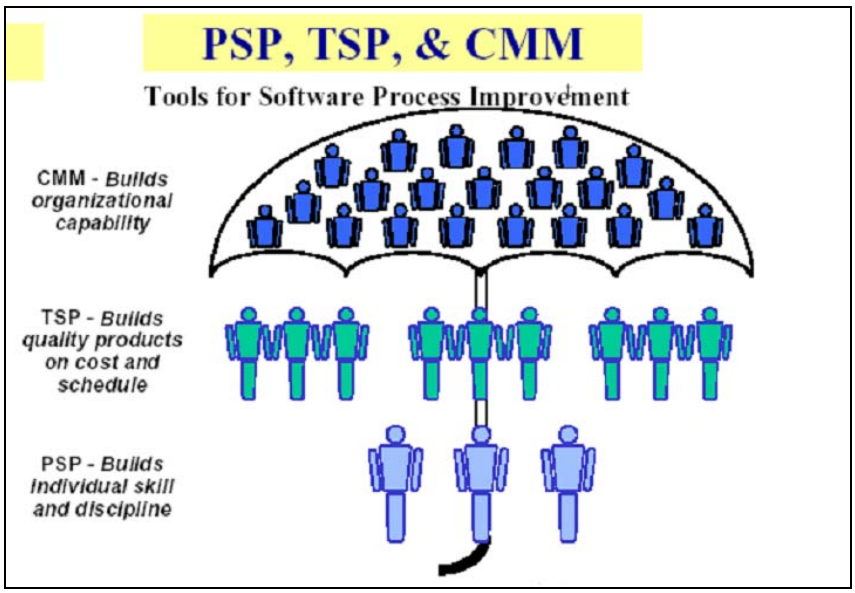
目前 TSP 使用 21 個流程指南、21 個資料表格。在這些流程指南中定義了 173 個啓動和開發步驟。每一個步驟的描述都非常詳細，開發人員能夠清楚地知道下一步應該做什麼，應該怎樣去做。這些過程指南可用來指導專案組來完成啓動過程和一步步地完成整個專案。

專案啓動過程之後，專案團隊應該有以下結果：專案團隊的目標、專案團隊各成員的明確角色、開發流程計劃、專案之品質計劃、全面的開發計劃和進度計劃、下一階段每個成員的詳細工作計劃、專案的風險分析結果以及專案的狀態報告。



圖表 2-8TSP 之架構圖

TSP 之整體架構包括三個部分，第一為針對管理人員之專案管理能力訓練及其管理時應盡之義務與工作之規定，即圖 2-8 之左半部，而第二為上段所述之 TSP 之專案流程，其位於圖 2-8 之中間部分，第三部分為對於指導者之訓練與團隊指導工作之規定，其位於圖 2-8 之左半部。



圖表 2-9CMU/SEI CMM 模型

PSP、TSP 和 CMM 爲軟體產業提供了一個三維的軟體流程改革框架。CMM 建立了組織能力之框架，PSP 則規範一位軟程師之軟體能力技術和軟體開發之紀律。進而 TSP 將接受過 PSP 訓練之工程師們結合成一個團隊提供開發的流程及學習指導以完成高品質之軟體之開發。

2.2.4. Leadership approach

Philips 在全球有超過 6000 位軟體工程師。然而在電子、電機及多媒體之軟體需求日較增加之驅勢之下，Philips 不得針對旗下軟體的軟體工程師們的能力進行管理，以提昇其在上述領域中的競爭力。而這一個以領域為主的能力評鑑方法 Leadership approach[15]也因應而生。

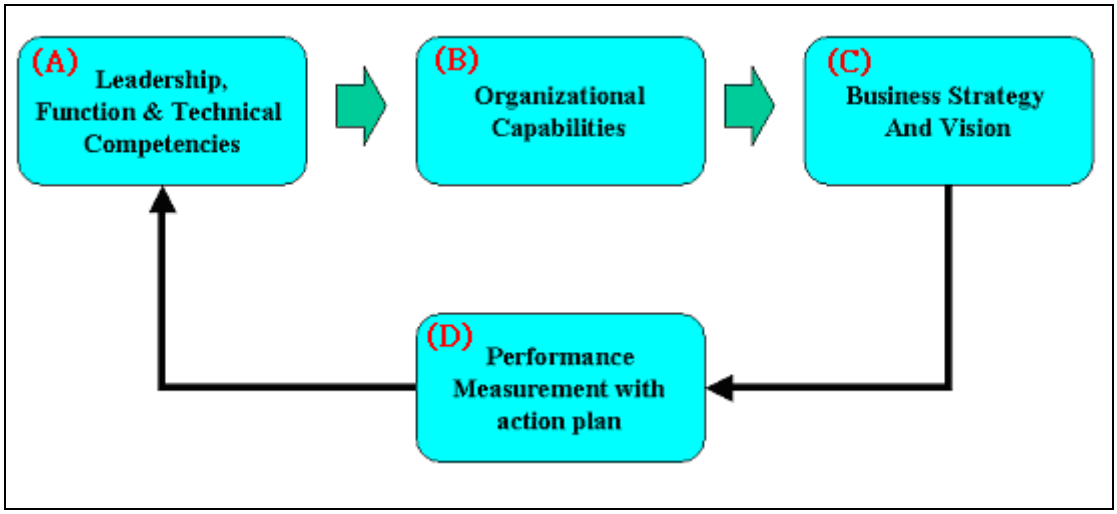
Leadership approach 將人員的能力依分成「領導力」(Leadership competence)、「功能技巧」(Functional Skill)及「專門知識」(Technical Knowledge)三方面來進行人員能力之評鑑。而接下來將提出 Leadership approach 如何確認所需能力之流程：(圖 2-10)

(A) 個人可對此次應用的知識、技巧和行為應如何貢獻？

(B) 判定對於工作而言，這些能力是否是被需要的。

(C) 列出此工作的主要的挑戰、價值及完成目標。

(D) 確認是否這些能力表現是足夠完成工作，如果不足的話應列出改善計劃。



圖表 2-10 Leadership approach 流程

然而在能力表現部分，Leadership approach則採和CMM系統不同之分級機制。其主要是以對專業能力之熟練度之定義為主，而非對流程之使用。在此列出Leadership approach之能力成熟度定義：

(1) **基本（Foundation ．level 1）**：

對於知識有基本的了解，可以完全處理與此知識相關的日常及一般的工作；遵循定義好之方法和工作流程；有基本之專業技術處理每天的工作

(2) **熟練（Practice ．level 2）**：

擁有較深或較廣之知識，可以處理非標準之工作；可根據環境的不同，針對特殊之專業領域，實行較可靠之流程與方法。

(3) **專家（Expert ．level 3）**：

擁有廣泛的專業知識，可以處理處理各式各樣之組織及事務上之問題；可以從策略之角度從長程的規劃中將專門之問題解決。

(4) **理想的領導者（Thought Leader ．level 4）**：

擁有遠見之思考，可以提出世界級之方法來改變目前的範例及，或重新定義特定領域之規則。

在此提出一個分級之範例：

Leadership Competencies					Functional Skills					Technical Knowledge				
Level*	I	II	III	IV	Level*	I	II	III	IV	Level*	I	II	III	IV
Shows determination					Knowledge management					Electrical				
Focuses on the market					Creativity and innovation					Mechanical				
Finds better ways					Problem solving					Materials				
Demands Top performance					Architectural style thinking					Computer				
Inspires commitment										Application				
Develops self and others										Other				

圖表 2-11 Leadership 分類範例

Leadership approach 利用知識、技能和領導三項指標來進行人才能力評鑑，對於如特定專家之尋找或挑選專案開發成員等有很大之助益。

第3章　程式設計師能力模型

軟體的開發需要許多的人員，如：Project Manager、Architect、System Analyst、Designer、Programmer、Tester 等。但為了完成軟體的開發，每個人員之所需擁有之能力均不相同，若要設計一個符合所有軟體開發人員之能力管理模型是不切實際的。因此本研究將能力管理先從 Programmer 著手，日後再推廣至其它軟體開人員。且由程式設計師著手之優點有二：

（１）Software Workbench 平台上可得一位 Programmer 實作之完整的專案資料，對 Programmer 能力分析資料之搜集，較易得到正確之結果。

（２）程式設計師之數目為所有專案角色中最多的，因此在能力管理的部分可以由程式設計師開始提出一個概念，之後再將此概念衍伸至其它軟體開發人員。

本章介紹本研究所提出之程式設計師能力模型，首先在 3.1 節介紹程式設計師之能力分類方式，進而在 3.2 節介紹能力成熟度的概念。3-3 節介紹能力模型之應用。

3.1. 程式設計師能力分類方式

程式設計師之主要之任務為軟體模組的製作，模組製作需靠程式設計語言完成，各種程式之特性不同，因此需依應用系統之特性選用最合適的語言，如 C 或 JAVA 等，因此程式設計師應具備對一到多種程式語言的撰寫能力。對程式語言的了解外，程式設計師還需對熟悉某些系統軟體(如，DBMS)及中介軟體(Middleware)才能以較快速的方式開發出高效率之應用軟體。除此之外，對於程式開發工具與平台的了解也是十分的重要，因為不同作業環境上執行之程式的寫作法與偵錯方式也不盡相同。依據上述分析，程式設計師欲開發程式模組，需具

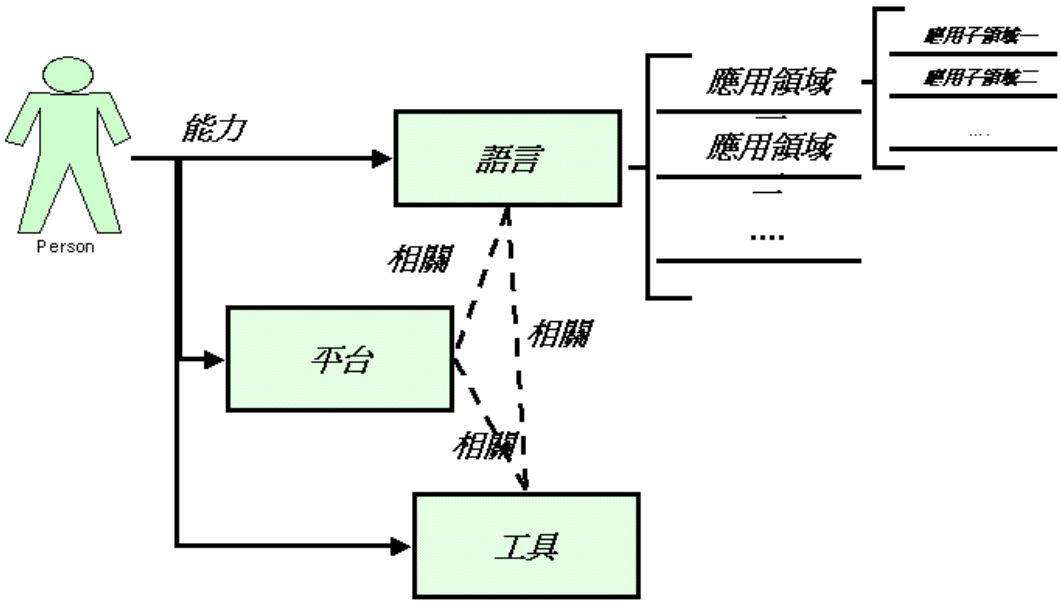
備以下能力：

(一) **熟悉使用之程式語言：** 對於程式語言不了解，則無法進行該程式的開發。此外，對語言的熟悉也包括各種系統軟體和中介軟體的了解。

(二) **程式開發工具之使用熟悉：**開發工具影響程式寫作之品質、效率，對開發的工具的熟悉，更能利用其偵錯功能來幫助提昇軟體的品質。

(三) **開發平台之使用熟悉：**開發平台包括：(1)程式開發環境。(2)程式執行平台。

依據上面分析，Programmer 的能力分類可以圖 3-1 來表示。



圖表 3-1Programmer 之能力分類圖

除了語言之外，於平台和工具的能力部分。例如一公司有 C++與 JAVA 之開發平台與開發工具可以用表 3-1 的分類法式來表示其熟悉處。因此程式設計師應具備在各平台上開發程式的能力及使用工具的熟悉度。才得以幫助軟體之開發工作。

	平台	開發工具	測試工具	偵錯工具
C++	Windows	Borland C++ Visual C++	BoundsChecker	WINDBG Spy++
	Linux	DEV C++ KDevelop	Cantata++	GDB HEX
JAVA	Windows	Jbuilder JavaCreator	JCheck	JDB
	Linux	JDE JDK	JTest	JDB

表格 3-1C++與 JAVA 之平台、工具表

(四)應用領域方面:以 ODMS 軟體實作為例，又可包括 Oracle、SyBase 及 SQL 等子應用領域。因此一位程式設計師之能力，可以從「程式語言」、「應用領域」、「開發平台」及「開發工具」等四項來進行分類。

3.2. 能力成熟度

3-1 節的能力分類，只能表示程式設計師熟悉之程式語言、應用領域知識、開發平台和開發工具四項但無法表示他們在某能力之熟悉程度，而熟悉程度特別影響程式模組開發之效率與品質，因此有必要進一步建立其能力指標(稱為能力成熟度指標)作為我們評鑑一位程式設計師能力標準，而為了應用運於專案開發當中，更進一步地，建立一個程式設計師能力分級制度來幫助程式設計師能力之管理。

3.2.1. 能力成熟度指標

一位程式設計師的能力應用某種程式語言及子應用領域來開發程式模組，其能力之高低可用工作經驗、工作效率與軟體品質來評估，分述如下：

（一）實作經驗方面：實作經驗之多寡與查技術手冊（manual）次數、偵錯時間長短有密切之關，這些都將反應在日後程式開發之速度與品質之上。程式設計師之實作經驗可用其已發展完成程式模組之**行數**(LOC， Line Of Code) [17]來表示。除了用行數之外可用「模組複雜度」(Complexity)，來表示其工作難度，在效率與品質相近之基準下，擁有平均模組複雜度高之的 LOC 應該比平均模組複雜度低的人員能力好。因此可將實作經驗之計算公式（Ci）表示如下：

$$Ci = \sum_{Module} Complexity * LOC$$

（二）效率：一模組完成後，LOC 乘以模組複雜度，再除以工作之天數（LOC*Complexity/Day）。經驗的判定利用累積的專案資料來觀察的是正確的，但若採用同樣的標準來衡量「效率」及「品質」的能力似乎是不恰當的。對於「效率」的評鑑應該採一段一段的時間來觀察。例如：一位程式設計師從新進人員到成為一位菁英的工程師，其工作效率可能從一開始為 20 行一天與後來的能力 100 行一天，這些資料若採用累計平均的標準來觀察，人員工作效率將會被低估。

本研究採用「**一季**」(Quarter)作為人員能力評鑑之時間區間，(組織可以依據其需求改變時間區間)。用季內的平均程式寫作效率之計算公式（Pi）如下：

$$Pi = \frac{\sum LOC\_During\_Quarter * Complexity}{Work\_Day\_During\_Quarter}$$

公式 1

此外，觀每察季 Pi 也可了解一程式設計師之效率精進程度。

（三）品質：在品質方面可以利用模組在 Code Review 及測試時發生之 Defect 來評估。Defects 的資料可利用 Software Workbench 所提供之程式設計師之工作資料來分析。一模組之「**缺失產生密度**」即記錄平均每千行程式會產生多少缺失(Defect/KLOC)[17]來判斷每一模組之品質，不過由於每種實作模組之困難度不一，因此在缺失必度之部分則應考慮模組複雜度此變因，即 $(\frac{Defect}{LOC * Complexity / 1000})$。

缺失密度是判定軟體品質之重要關鍵，但若是單從缺失數目來判讀軟體品質其實是不夠的。Lowell Jay Arthur 所提出的 Substantial Quality Improvement[16]方法中將軟體缺失分為三種等級「嚴重」(Serious)、「中等」(Moderate)、「輕微」(Minor)三種等級。因此在導入每個錯誤的等級之後可以建立起模組品質評鑑(Di)公式：

$$Dl = \frac{Ws * Si + Wm * Mi + Wt * Ti}{LOC * Complexity / 1000}$$

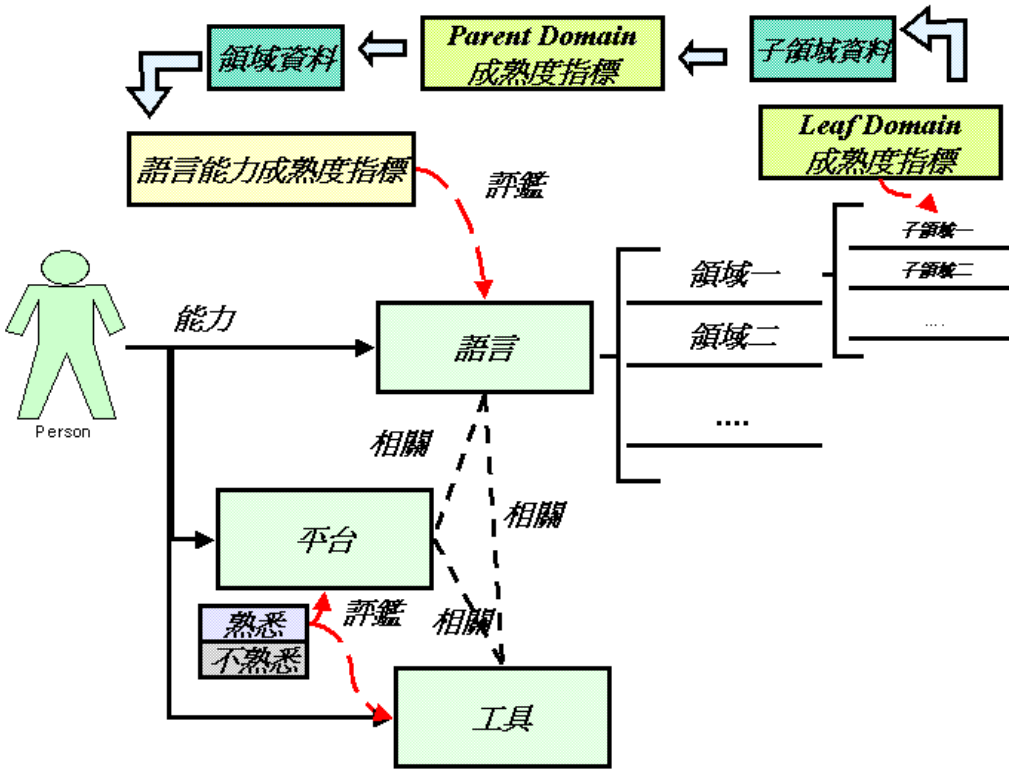
Si： 嚴重缺失數 Mi： 中等缺失數。

Ti： 輕微缺失數

Ws： 嚴重錯誤比重 10。Wm： 中等錯誤比重 3。Wt： 輕微錯誤比重 1

同樣地，也使用每季為單位來衡量程式設計師之品質能力。

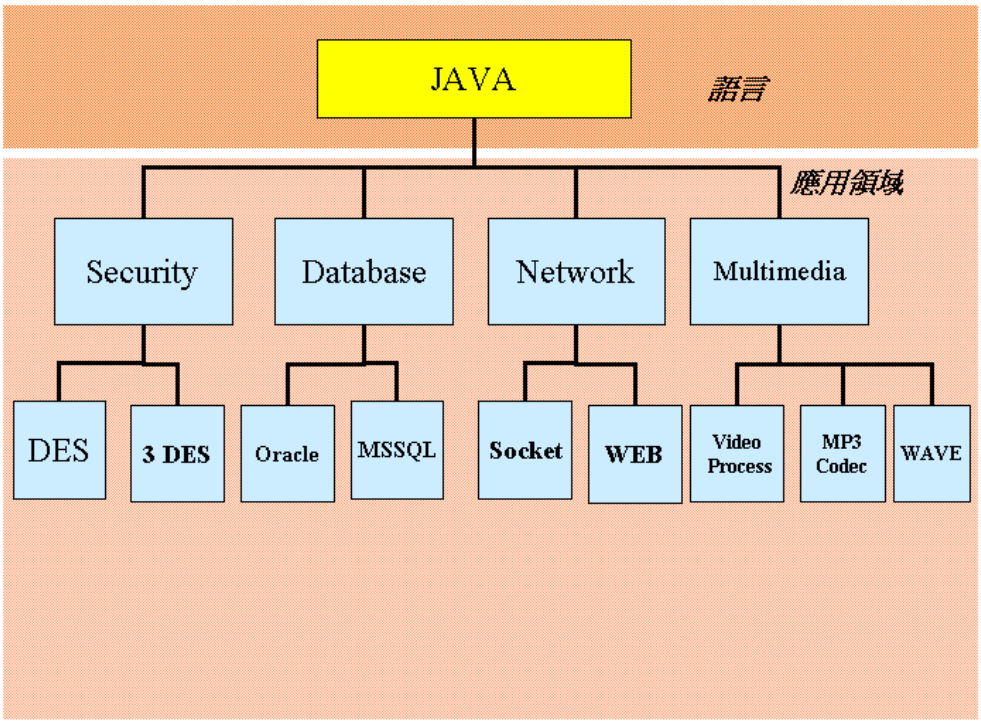
程式設計師最重要之能力為語言的成熟度。而語言的成熟度的判定則是由實作之各領域專案資料所匯集而來的，如圖 3-2。因此接下來將會分別定義語言及應用領域成熟度指標。



圖表 3-2　Programmer 之能力成熟度指標示意圖

領域知識成熟度指標定義：

Programmer 在程式的實作當中，均會使用不同的語言來開發軟體模組，而每個軟體模組均有其所需要解決之問題及工作。因此每個模組一定會是屬於某種知識領域之下的。然而因為需求的不同，知識領域的分類可能會再被區分成更小的子知識領域。



圖表 3-3 語言、領域知識關係圖

以圖 3-3 為例，某公司中將一位程式設計師對於 JAVA 語言分成 Wave、MP3 Codec、Video Process、Socket、WEB、MSSQL、Oracle、3DES 和 DES 等應用領域，而一 Programmer 實作 Java/Socket 模組程式，也視為在 Network 之應用領域內之資料。而對於各應用領域之成熟度的判定可以利用下表之能力指標進行。

Domain 能力成熟度指標	
實作經驗	∑實作模組行數*模組複雜度 $Ci = \sum Complexity * Module\_LOC$
工作效率	∑(每季實作模組行數*模組複雜度/工作天數) $Pi = \frac{\sum LOC\_During\_Quarter * Complexity}{Work\_Day\_During\_Quarter}$

產物品質	AVG(Σ錯誤之等級*錯誤之比重)/(實作模組行數*模組複雜度/1000) $Di = AVG\ (\frac{Ws * Si + Wm * Mi + Wt * Ti}{LOC * Complexity} / 1000)$

表格 3-2Leaf Domain 能力成熟度指標

語言成熟度指標定義

由圖 3-2 及圖 3-3 可知「語言」能力成熟度，應由各應用知識之能力來評量，而評鑑應從對語言能力的「深度」與「廣度」來進行評量。語言能力的「深度」為該語言之實作經驗、效率、品質，而語言的深度評定則應與子領域的平均能力來觀察才公允，因此在經驗、效率、品三分面，均是以所有的子領域之資料的平均來計算：

$$\text{經驗} = \sum_{i=1}^{N} \text{應用領域之經驗}\ (Ci)$$

$$\text{效率} = \frac{\sum_{i=1}^{N} \text{應用領域之效率}\ (Pi)}{\text{總應用領域}\ (N)}$$

$$\text{品質} = \frac{\sum_{i=1}^{N} \text{應用領域之品質}\ (Di)}{\text{總應用領域}\ (N)}$$

「廣度」即對程式語言應用域之廣泛度，擁有多個不同領域之知識才可被認定程式語言能力是熟練，各公司可依本身性質來制定熟悉應用領域數目之限制，因此對於語言的評定得新增一個領域知識經驗的指標：

同樣的，基於以上討論，將「語言能力成熟度指標」整理於下表：

語言能力成熟度指標	
實作經驗	$\text{經驗} = \sum_{i=1}^{N} \text{應用領域之經驗}\ (Ci)$
工作效率	$\text{效率} = \frac{\sum_{i=1}^{N} \text{應用領域之效率}(Pi)}{\text{總應用領域}(N)}$

產物品質	$品質 = \frac{\sum_{i=1}^{N} 應用領域之品質(Di)}{總應用領域(N)}$
領域知識	$Ki = \frac{成熟應用領域數}{總應用領域}$

表格 3-3 語言能力成熟度指標

3.2.2. 能力成熟度等級建立

在專案人才的挑選的時候我們可以採 3-2-1 節所定義 Programmer 能力指標的值來比較兩兩 Programmer 之間的「程式」或是「知識」能力。但使用此方式來進行人員能力的比較則會顯得不夠有效率，且無法看到人員通盤的能力水準。因此需將 Programmer 的能力成熟度指標進行分級評鑑。

Programmer 的能力成熟度區分成下列四種等級：

Programmer 能力成熟度等級定義	
軟體人才等級	***定義***
新進級軟體開發人才(Rookie)	無實作經驗的人員，需人員之協助才得以完成工作。
一般級軟體開發人才(Normal)	已有能力完成組織所指派的一般難度之軟體開發工作。其實作軟體的在效率上及品質上均有一定水準。
菁英級軟體開發人才(Elite)	可以幫助組織完成困難模組的完成。人員於經驗、效率及軟體品質上有突出的表現。

專家級軟體開發人才(Expert)	具有開發新領域及技術之能力，可供其它人員諮詢並提供指導的人員。在多語言及多領域知識部分均為菁英。

表格 3-4Programmer 能力成熟度等級定義

組織可以根據對於各別能力的需求來定義不同能力等級的人員，所需要的標準為何。而能力的定義則可以由等級評鑑表的填寫而來，如表 3-5 及 3-6，我們將要利用表 3-5 的 **JAVA Socket 領域知識等級評鑑標準**來幫助評鑑一位 Programmer 在使用 JAVA 語言，開發 Socket 程式的能力 及利用表 3-6 的 **JAVA 語言等級評鑑標準**，來作為新進級、一般級和菁英級的評鑑範例。並在介紹完此二評鑑等級之後介紹如何評定一位專家級的人員。

(一) 領域知識等級評鑑標準：

[JAVA]][Socket]領域知識等級評鑑表					
標準	***細部項目***	***專家級***	***菁英級***	***一般級***	***新進級***
經驗	經驗(Ci)	**60000**	**16000**	**8000**	**1**
效率	效率(Pi)	**120 行以上**	**100 行以上**	**50 行至 100 行**	**50 行以下**
品質	缺失密度(Di)	**2 以下**	**4 至 2**	**6.5 至 4.0**	**6.5 以上**

表格 3-5Socket 領域知識等級評鑑表(Leaf Domain)

(二) 語言等級評鑑標準：

[JAVA]語言等級評鑑表					
標準	*細部項目*	*專家級*	*菁英級*	*一般級*	*新進級*
經驗	經驗(Ci)	**80000**	**40000**	**20000**	**0**
效率	效率(Pi)	**100**	**100**	**50**	**50**
品質	缺失密度(Di)	**2 以下**	**4 至 2**	**6.5 至 4.0**	**6.5 以上**
領域	總領域知識數(ki)	**菁英:8 領域以上**	**菁英：4 領域以上**	**一般：3 領域以上**	

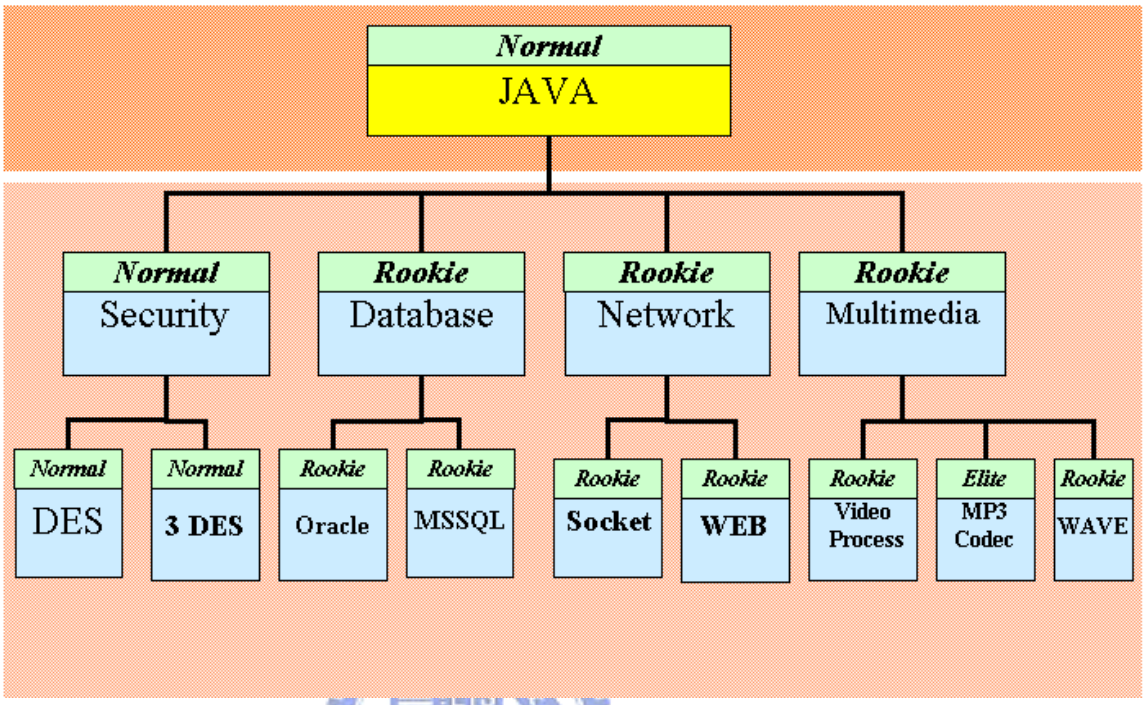
假設此公司 JAVA 實作應用領域共有 10 項

表格 3-6[JAVA]語言等級評鑑表

在提供等級評鑑之後，人員的能力將可以被通盤地被檢視。如圖 3-4 所示。分級法可以幫助我們對於人才能力比較之給予第一階段之篩選，而若是同樣等級的人員則可以利用各項知識評鑑的細部指標，如效率、品質和經驗來排序，如此一來可以斷定每一位 Programmer 之能力的高下。完成了能力成熟度的分級之後，可以得到的優點有：

（一）人員能力的通盤了解：分級法可以通盤地清楚地得知 Programmer 所有能力和專長，因此可以幫助組織了解其能力與競爭力所在。

（二）提供人員能力改進標準：採用訂定好的量化的能力昇級標準，可以幫助人員可了解下一個能力提昇門檻，使人員有一努力、改進的目標。



圖表 3-4　Programmer 之能力等級示意圖

3.3. 能力模型的應用

本研究將把上述之能力模型整合至 Software Workbench 之中，由於 Software Workbench 主要之協助對象為實作階段的 Programmer，對於實作階段的 Programmer 而言，能力模型的加入僅能幫助他們在有困難時可以找尋專家來給予協助。不過除此之外，還可以將將能力模型之概念應用在 Project Manager 和高階主管兩位角色工作協助上面。接下來將會分討論能力模型的應用：

(一)專案管理工具

的工作上，則 Project Manager 可以利用人才評鑑來進行專案管理，包括人員之挑選、時程計劃、資金計劃及專案監控[18]等部分，

(1)**專案人員挑選**：整合程式設計模型一般軟體專案之工作分配均會對於模組之類別、語言及工作的平台進行設定。然而藉由之前所提出之 Programmer 能力模型之概念則可以幫助我們找到適合的人才。如，要挑選 JAVA 程式之 Socket Programming 之程式設計師。則可以先查詢對於 JAVA 熟悉之 Programmer（可以設定能力等級，如需要菁英或是一般），然後再從挑選中之 Programmer 當中找尋對於 Socket Programming 熟悉的人員。而若是有多個人選存在，則可以依據需求來找尋在效率或是品質上較有保證之人員。則如此即可以利用能力模型來挑選適合之人員。

(2) **專案時程之計劃**：利用整合能力管理模型之效率可以幫助時程之計劃。其作法為：第一、採用 Software Workbench 中之專案資料，利用類別、語言、平台、模組複雜度及實作人員的能力來進行模組行數預測。採用 Programmer 之效率和模組預測行數來進行時程之預測。第二，若是資料庫中有相似之模組，則採用該人員之最近一次之時程資料來進行預測。然而在利用完上述兩種預測方法之後。

(二)能力管理工具

除了針對 Project Manager 之管理工具的的開發之外。也將提供給高階主管使用 Programmer 能力管理工具，其功能包括：

(1) **Programmer 能力完整檢視工具:**提供人員完成的專長表,及 Programmer 各設能力的成長驅勢。這些資料可以提供高階主管了解整個團隊或是組織的能力及競爭力所在。

(2) **Programmer 學習管理工具：**派遣人員去進修、及學習的統計等機制。此工具幫助高階主管了解對於人員的培訓成本是否被有效的使用。

(3) **能力等級評鑑訂定工具 ：**針對能力的評鑑標準管理的工具，且可以針對各項能力進行評鑑標準的設定，以幫助能力模型的不斷改善。

第4章　能力精進與缺失知識庫

由第三章之能力模型可知，一位程式設計師之能力是由其熟悉的領域及程式能力之表現來評鑑的，一位新進之程式設計師是經過一連串的學習及工作的磨練，才能到達菁英或是專家級，為軟體開發隊提供更多的貢獻力。本章首先在 4-1 節介紹程式設計師能力精進的方式，由能力精進分析，發現程式偵錯是提昇程式設計師能力的一條快速捷徑，為加速程式偵錯時間而導出缺失資料庫之構想，並在 4.2 介紹錯誤知識庫之設計。

4.1. 程式能力之精進

Programmer 的能力管理評鑑與應用外，更重要的為如何使「Programmer 的能力不斷精進」，能力的精進可分成兩個部分來討論

1. **應用領域的學習**：軟體技術不斷進步，新程式語言，middleware 或 System Software 不斷地推陳出新，程式設計師需不斷學習這些新技術以提昇工作效率及應用領域，因此企業必需有計畫地派遣工程師赴外受訓或自我訓練，以使程式設計師具有更多應用領域程式發展之能力，才能開創新的應用。目前數位學習技術已成熟，各類新課程不斷推出，更方便於學習，公司必須有計畫地依程式設計師之生涯規劃給予訓練，並建立其學習記錄(包括課程、學習時間、學習測驗結果)，作為鑑別學習能力參考之依據。
2. **由錯誤中學習**：程式設計師學了新的語言或是應用領域，需透過實作才能使其能力精進，程式偵錯是提昇其能力最佳學習機會。程式撰寫錯誤代表針對某部分知識了解的不足，偵錯成功可以使程式設計師更了解這部分知識。但程式偵錯是一件費時的工作，我們只知道程式執行結果錯誤，不知道錯誤的原因甚至錯誤的所在，常需透過各種工偵錯技巧才能發覺錯誤之所在，進而

在由應用手冊中找出正確的寫法而修改之。有時甚至費了很多的時間也找不出原因，只好找尋專家級或菁英級之程式設計師協助解決。因此如何提昇程式設計師偵錯效率也是提昇程式設計師工作效率的重要機制。

由實務中發現，一個資淺工程師所發生的錯誤，大部分都是資深工程師在工作過程中曾犯過，換言之，可應用 80/20 理論來解釋此道理，因此若能有一錯誤知識庫記錄程式設計師們所犯過的錯誤與解決方案，對於後進工程師的偵錯時間的節省與學習有甚多助益。且由一工程師所犯錯記錄也可導出 Code Review 的 Checklist，可大幅降低重複犯錯的機會且提昇其工作效率，另一方面對於工程師們常犯的錯誤也可列為教材修正的一部分，使工程師學習新課程後，降低錯誤的發生，如此循環，以先人的錯誤教訓來降低後進者發生錯誤的機率，以達到整體能力大幅提昇之目的，因此有必要在 Software Workbench 上加上錯誤知識庫的機制。

4.2. 錯誤知識庫之設計

錯誤知識庫是幫助程式設計師能力提昇的良方，因此本節介紹錯誤知識庫之設計，包括錯誤知識庫之需求、架構及相關的機制進行討論。

4.2.1. 錯誤知識庫之功能需求

錯誤知識庫之建立建立的目的為記錄程式設計師之錯誤及解答，以前人的錯誤經驗來幫助後進之程式設計師減少錯誤產生之機率。錯誤知識庫功能需求討論如下:

錯誤知識庫記錄「每一錯誤發生時，產生的訊息，錯誤的原因，解決方法及範例」，錯誤知識可以錯誤項目來當主分類，例如 array、Stack 等，再依錯誤訊息當次分類，程式發生錯誤時，可依錯誤訊息或是錯誤項目查詢。除了解決方法

外，若能加上「Check Item」則更能提程式設計師自我檢查，此 Check Item 應自動加入個人之 Checklist 檔中，若一程式設計師已不再犯其 Check item 所提醒之錯誤，可由本人自 Checklist 檔中將此 Check Item 刪除。

此外，當另一程式設計師又犯了同樣之錯誤而至此錯誤知識庫找尋解答時，知識庫應有自動登錯誤累犯次數之功能，當累犯次數高於某一數值，此錯誤可視為常犯錯誤，知識庫管理機制自動通知管理者，將此斜誤列入教材。同樣地，一新誤在同一天有許多人重複犯錯，代表訓練非臻完備，將此錯誤知識通知全體使用該領域之程式設計師及管理者，以防止錯誤繼續發生。

當一新程式錯誤發生，而發生錯誤的程式設計師苦思無解，亦可以經由錯誤知識庫向組織中相關領域之資深者，尋求協助以提昇除錯能力。

綜合以上所述，可歸納知識庫系統之需求如下:

(1) 錯誤項目之登入、修改

(2) 錯誤項目查詢、使用及自動累計錯誤次數。

(3) 個人 Checklist 之 Check Item 的自動加入，及人工刪除。

(4) 當錯誤次數多時自動通知管理者。

(5) 新錯誤於一天發生時，自動通報程式設計師

4.2.2. 錯誤知識庫之系統架構:

錯誤知識庫提供組織內所有程式設計師使用，當程式發生錯誤時，程式設計師可利用此知識庫進行問題原因之判斷、解答之查詢和 Checklist 之閱讀，錯誤知識庫的系統將採 Client-Server 的架構，由於而這些功能均是由程式設計師所主動進行，因此在知識庫部分有資料集中之特性，在 Client 端，可用 Web Page 來表示，因此無論程式設計師可透由 Web Browser 來進行錯誤知識庫之使用。而在 Server 端，則利用 WINDOWS 2003 SERVER 並採 IIS 及 ASP 來作為開發平台。

4.2.3. 錯誤知識庫貯存設計

錯誤資料庫貯存格式包括:錯誤資料貯存格式、解答貯存格式，Checklist 貯存格式，而此三項資訊為組成錯誤知識庫之核心。

錯誤可以分成不同的類別如 Stack、Array 等類型，而每種錯誤類型下又有多個錯誤，每個錯誤均有相對應之解答、Checklist Item、和錯誤訊息。因此一個錯誤資訊之貯存資訊包括: 錯誤類別、錯誤定義、錯誤原因、平台、語言、錯誤類別、IDE 錯誤訊息、解答、Checklist item、範例碼、被引用之次數等資訊。

錯誤格式:

錯誤編號	錯誤類別	錯誤定義	錯誤原因	平台	語言	錯誤訊息	累計次數

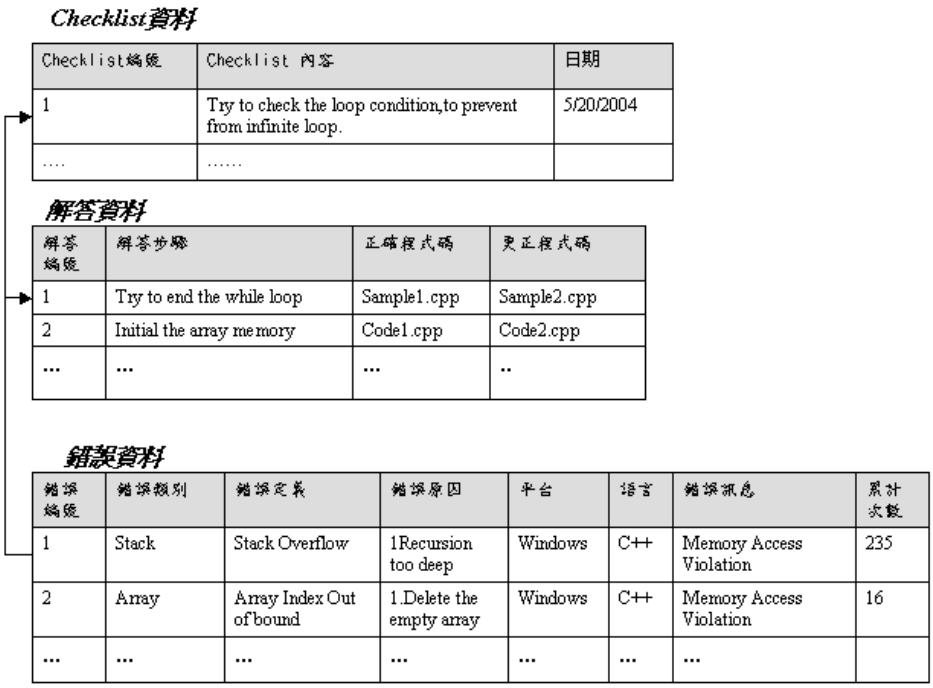
解答格式:

解答編號	解答內容	正確程式碼	更正程式碼

Checklist 格式:

Checklist 編號	Checklist 內容	建立日期

錯誤知識庫貯存格式可以圖 4-1 為例:



Checklist資料

Checklist編號	Checklist 內容	日期
1	Try to check the loop condition,to prevent from infinite loop.	5/20/2004
....	.....	

解答資料

解答編號	解答步驟	正確程式碼	更正程式碼
1	Try to end the while loop	Sample1.cpp	Sample2.cpp
2	Initial the array memory	Code1.cpp	Code2.cpp
...	...	...	..

錯誤資料

錯誤編號	錯誤類別	錯誤定義	錯誤原因	平台	語言	錯誤訊息	累計次數
1	Stack	Stack Overflow	1Recursion too deep	Windows	C++	Memory Access Violation	235
2	Array	Array Index Out of bound	1.Delete the empty array	Windows	C++	Memory Access Violation	16
...	...	...	...	...	...	...	

圖表 4-1 錯誤知識庫儲存範例

4.2.4. 錯誤知識庫使用介面

錯誤知識庫之使用主要為幫助程式設計師找尋在程式寫作之中，錯誤產生之時的解答查詢。對使用者而言主要的功能有「登入權限」、「錯誤查詢」、「Checklist 檢視」、「解答回饋」。而對管理員而言，主要的功能有「人員管理」、「錯誤-解答資料管理」、「解答審查需求發送」、「教材修改需求發送」等功能。

- **系統之主畫面:**使用者在輸入使用者名稱和密碼之後即可進行缺失查詢之功能。而使用者的權限將分成一般使用者及知識庫管理員。

圖表 4-2 錯誤解答查詢主畫面

- **錯誤解答查詢:**一般使用者在登入之後，最主要之功能為查詢自己犯的錯誤的解答，而流程如下:

  1:查詢需填入首先要求人員輸入所使用語言，Visual C++、工作平台，Windows 及錯誤訊息，Overflow。

圖表 4-3 錯誤查詢介面一

2:系統列出可能的錯誤原因，而錯誤依類別和原因排列，Programmer 可以依據類別來判斷可能之錯誤原因。本例錯誤訊息相關之錯誤解答有五項，包括 File、Arithmetic、Recursion、Stack、Floating Point 等六種

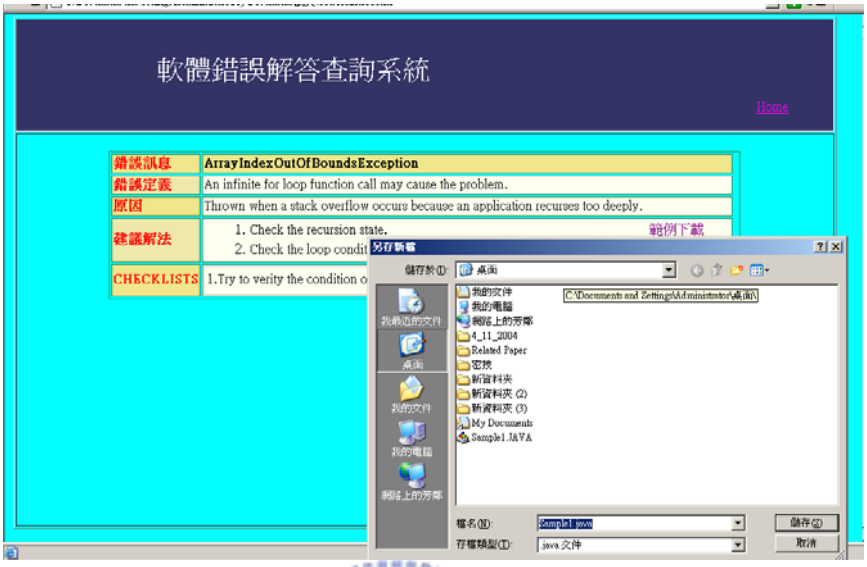
錯誤類別，而在此點選了 Recursion 之 Stack overflow.

圖表 4-4 錯誤查詢介面二

3.在參閱了類別及原因之後，則使用者點選可能之相關原因，而針對錯誤原因則有解答，解答包括錯誤訊息、錯誤定義及原因說明及相關解法與 Checklist.。在解答下方則供了解答是否解解了使用者問題之確認按鈕，藉由使用者點選此按鈕，則可以統計錯誤產生累積次數，可將累積次數分兩部分來進行使用，包括教材修改及解答發送之功能。

圖表 4-5 解答檢視介面

4:在得到了解答之後，則可以依據解法來進行程式之偵錯與修正，若是對於解法之說明還不甚了解，可以利用範例程式碼之下載來幫助對於解答的了解。



圖表 4-6 範例程式碼下載介面

5.若此錯誤對於使用者個人而言是一個常犯之錯誤，因此也可以點選 Checklist 欄位中之「新增」，則此 Checklist Item 將會被加入使用者個人之 Checklist 之中。因此在點選了檢視個人的 Checklist 之後，則可以看到預設語言(JAVA)之 Checklist 內容，而可以挑選其它語言之後，則可以觀看其它語言的 Checklist 提示。而在確定個人不錯再犯類似錯誤即後，即可以將 Checklist Item 移除。

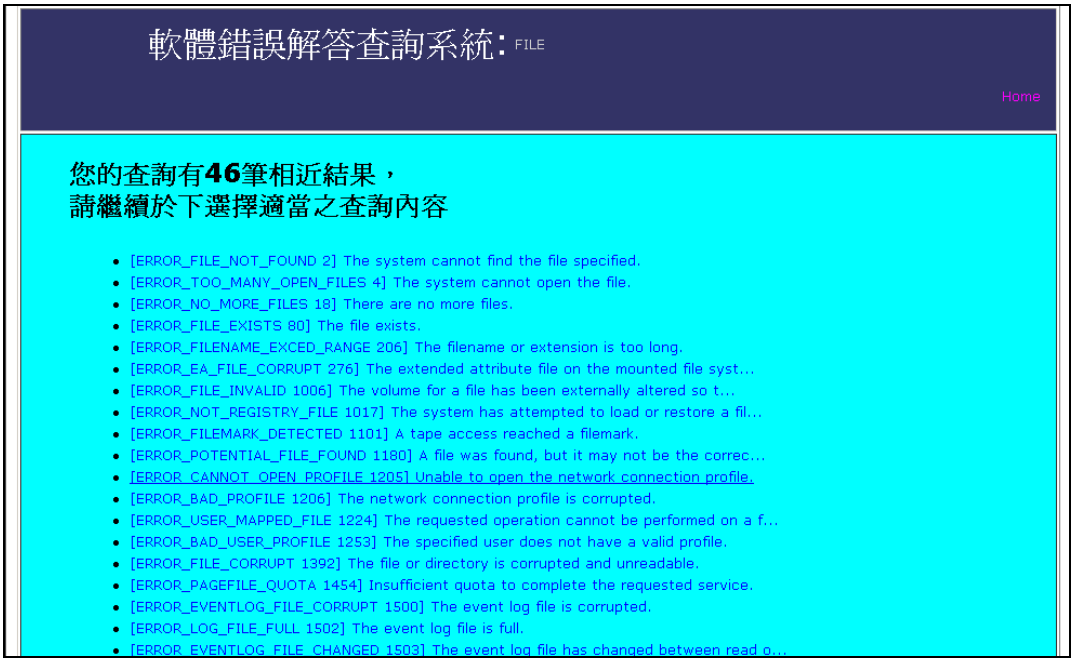
圖表 4-7 個人 Checklist

- 類別查詢使用者介面:

1. 點選了依類別找尋則可出現目前系統中擁有之錯誤類別列表。

圖表 4-8 依類別查詢

2. 點選了 FILE 類別之後，則發現可能之錯誤有 46 筆，因此使用者可以於此資料中，進一步地找尋是否有自己所需要之解答。在點選了可能之解答之後，就與依錯誤訊息來查詢之介面相同。



圖表 4-9 依類別查詢介面二

- **解答回饋介面:**若是使用者發現一知識庫中尚無解答之問題並找到解法，則可以利用解答回饋介面來將知識回饋給錯誤知識庫，以幫助後來程式設計師解決類似之問題。回饋介面如下:

圖表 4-10 解答回饋介面

在完成了解答回饋之後，則需求將會被送至知識庫管理員，待知識庫管

理員，請專家進行解答審核之後，則會回信給提問人員。若是解答審核通過，則系統同時將錯記錄解答回饋之人員之解答提供次數，以作為人員貢獻度之評鑑。

- **錯誤知識管理員介面:**

使用者在解答回饋完了知識之後，則錯誤知識管理員則可以藉由管理介面來處理此解答回饋之功能。

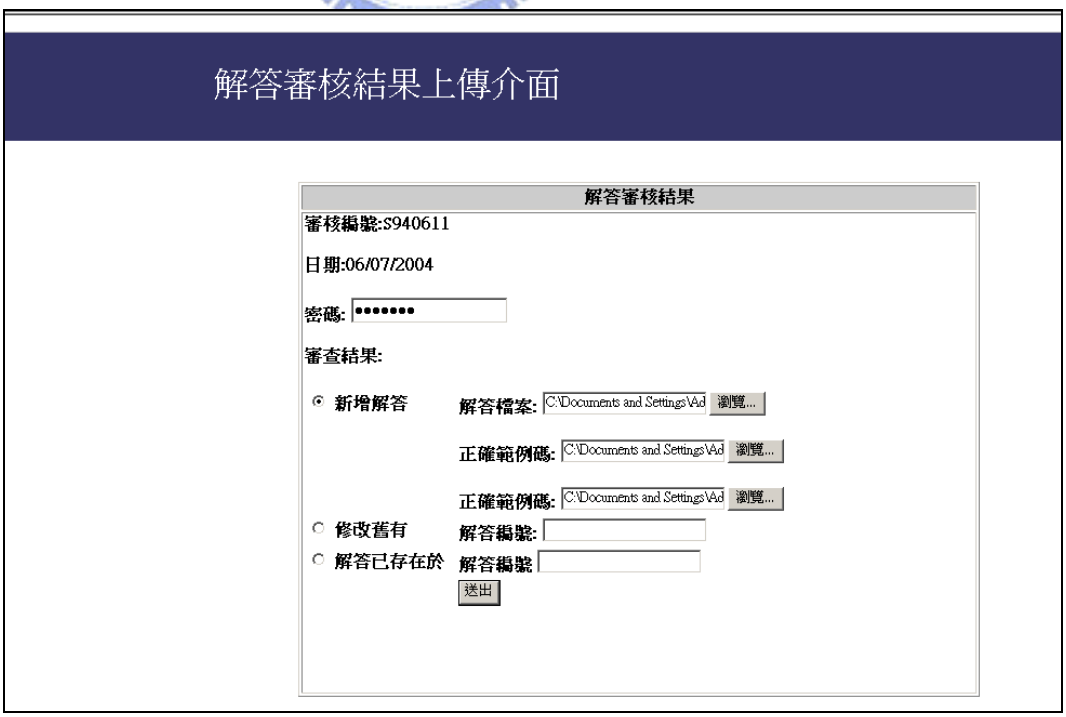
1. 管理員登入系統之後，可以看到待辦工作，包括由系統所發送的和由使用者所發送之需求。因此在使用者回饋了知識之後，則在管理員之待辦事項上則多了一項解答回饋之需求工作，管理員則可以點選該項目，並進行處理。

圖表 4-11 管理員工作介面

2. 點選了解答回饋之後，則為解答寄送介面。知識庫管理員，可以挑選各種語言之專家來進行協助。在語言附件中，提供待審解答之 Doc 檔案，而專家在審查後，應該至審查結果上傳網站登錄最後審查結果。

圖表 4-12 解答寄送介面

3. 專家解答審查結果介面:包括密碼之填寫，和解答審查結果的挑選。而在專家送出之後，則系統管理員則可以在待辦工作中，收到審核結果，並可以開始進行資料庫之修改。而解答審查同時也會寄送給，提供解答之人員。



圖表 4-13 解答審查結果上傳介面

系統還有一系列自動化工作需要進行，即在錯誤累積次數過高時，發送給教材管理人員教材新增之要求，及錯誤一日之內多次產生時，則自動發送解答給使用者。

- 教材修改需求介面:系統若發現某錯誤之發生累計次數過高，則系統自動進行寄送教材修改需求給教材管理員，寄送內容如下圖。

圖表 4-14 教材修改需求結果

第5章　程式設計師能力庫系統之設計

本章將說明「程式設計師能力管理資料庫」系統之設計與實作。5.1 節中介紹程式設計師能力庫之設計與功能的介紹。5.2 節則是介紹能力庫之資料儲存設計，最後於 5.3 節中介紹能力庫之使用者介面範例。

5.1. 程式設計師能力庫之功能需求

程式設計師能力庫之系統將分成兩個部分，第一個部分為與 Software Workbench 整合之部分，於 Software Workbench 上新增之機制包括:

1. 模組資訊收集機制:模組結束時之模組資訊之記錄，包括 LOC、類別、工作時間等資訊之收集。
2. 專案人員挑選機制:本系統將會提供專案經理一專案人員挑選機制，系統可以針對專案之需求，幫助挑選適合之程式設計人才。
3. 時程預估:模組開發時程可以依程式設計人才之效率來進行預測。
4. 成本預估:人員工作效率可以推測某工作所需之時數，而搭配人員之基本薪資，則可計算出模組開發所需之成本。
5. 人才能力監控:高階經理可以針對人員能力的成長曲線來進行人員能力之監控。

除了整合於 Software Workbench 上之機制之外，則還有程式設計師能力資料庫管理之功能，包括:

1. 能力評鑑機制:Software Workbench 收集了所需之模組資料之後，則能力評鑑機制將必須主動地進行人才能力之判定。
2. 能力標準管理:為完成能力評鑑則需要能力評鑑標準之建立，而依據組織之需求或時間的改變，評定標準可以隨時間修改或是出現了新之技術做，則需具有新增評鑑標準之機制。

5.2. 能力庫資料儲存設計

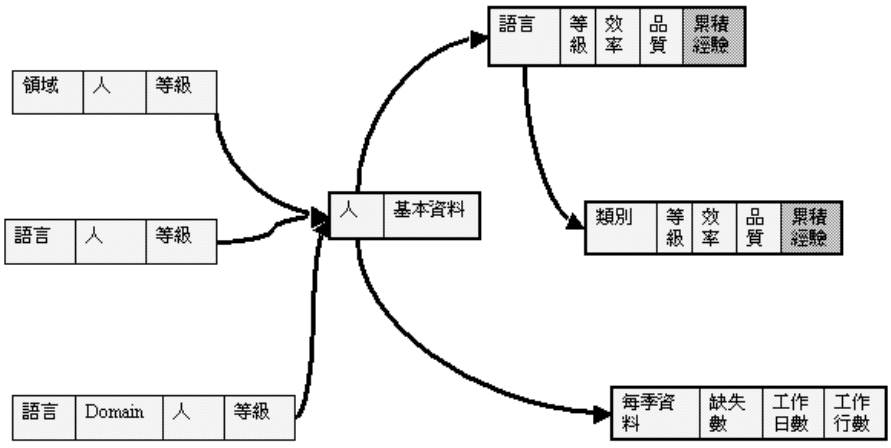
程式設計師能力資料將需要從 Software workbench 中取得，而所需要用來評鑑程式設計師之能力資料，則需從 Software workbench 上得到下列之資料:

Module Data
1. *Language* 2. *Domain* 3. *LOC* 4. *Complexity* 5. *Work days* 6. *Major Defect number* 7. *Medium Defect number* 8. *Minor Defect number*

圖表 5-15-1 回饋模組資料

語言、領域的不同則代表不同能力之資料，LOC、Complexity 則可以用來作為經驗之評鑑，work day 指的是此模組總共之實作天數，若與經驗頁除即可得到效率資料。而 Defect 之等級與錯誤數合起來再除以千行則可以成為評鑑軟體品質的標準。因此如此的專案資料回饋是足夠的。

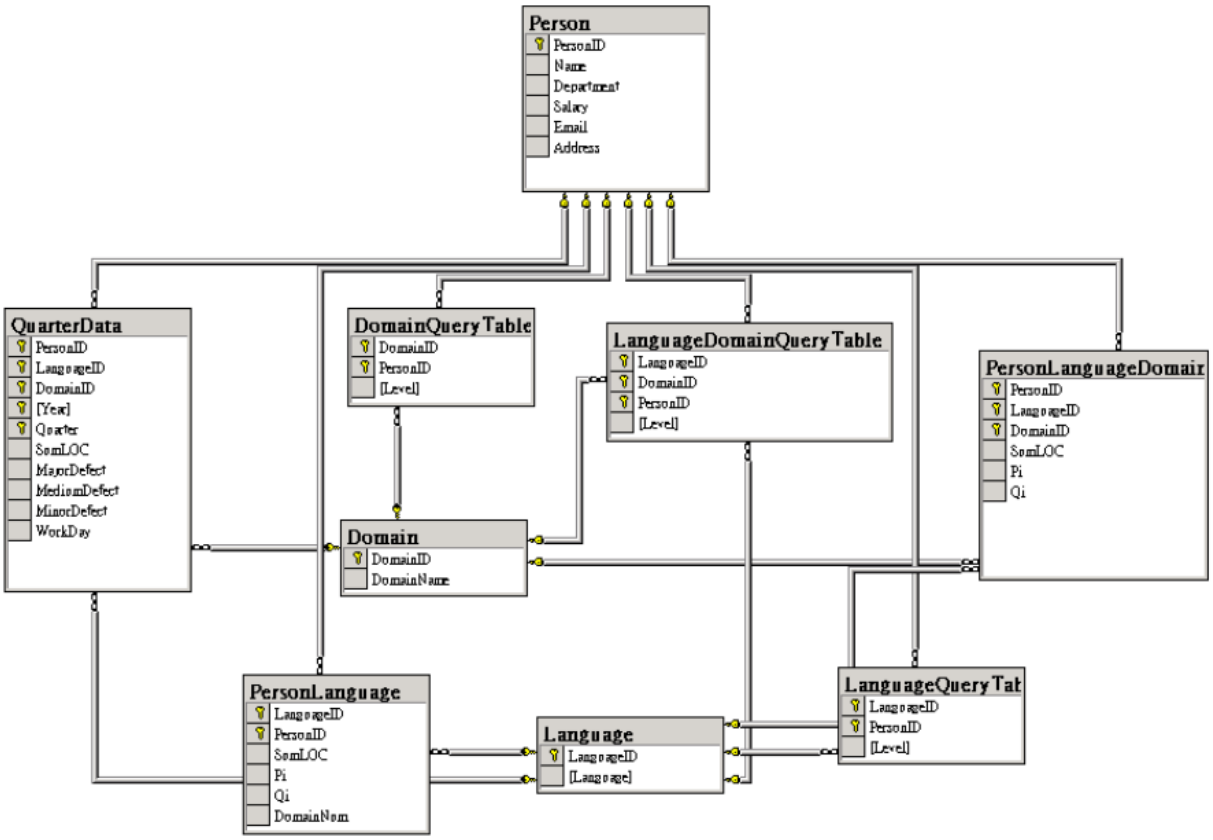
因此若要能快速地完成上述之需求，將資料庫設計概念如下:



圖表 5-2 能力資料庫概念圖

如此之資料庫設計，是從使用面的角度來思考的，整個資料庫之設計以人員之資料為中心。左半邊是作為加是查詢速度所用的，查詢的方式不外乎為從語言、或從語言+領域或是單一領域之人才查詢而來。右半部則分成三個部包括每季資料，每季資料則是儲存了人員每一季之專案累積之資料，而這些資料被依語言、領域分門地被歸類儲存。因為能力模型之設計，語言之下為模組，因此在資料庫上的設計也是如此，當資料被新增的時候，則人員之領域、語言的累積經驗將需要被馬上的更新，但是在效率、品質和等級的評鑑方面則為一季評鑑一次，因此當每季結束之後，則每季所有新增的資料(除了累積經驗之外)全部需往上更新，新一季人員在領域方面完成了能力之評鑑，則利用所有完成的領域的表現來為語言之能力等級進行新的更新。在更新完了左邊三個資料表之後，則為了加速查詢，則同時應將新能力等級評鑑結果同時回饋至左邊三個資料表中，雖然如此之做法會造成資料之重複儲存，但卻可大大地減輕資料庫系統所需之查詢負擔。

因此在每一季結束之後，資料庫將重新計算人員新的能力表現，由於累積經驗是動態直接新增至資料庫中，因此也程式設計師無需擔心因為三個月評判一次對於其經驗之累積空窗期。在討論完概念圖之後，可以畫出程式設計人才能力管理資料庫之資料儲存圖。



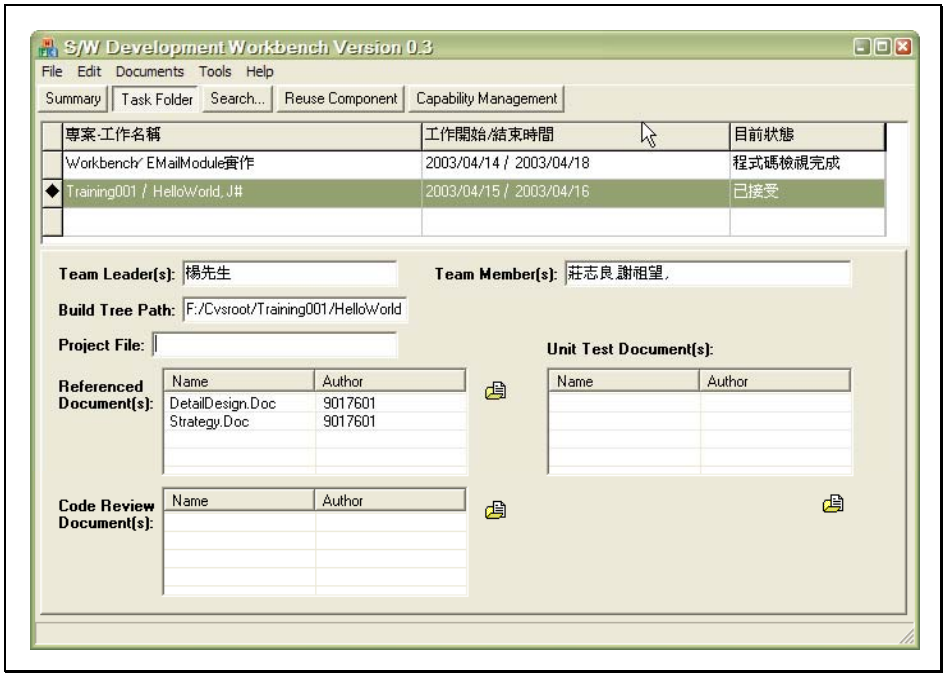
圖表 5-3 能力資料庫儲存

5.3. 程式設計師能力庫使用者介面說明

以下將會針對需求一一地進行系統功能與使用者介紹。

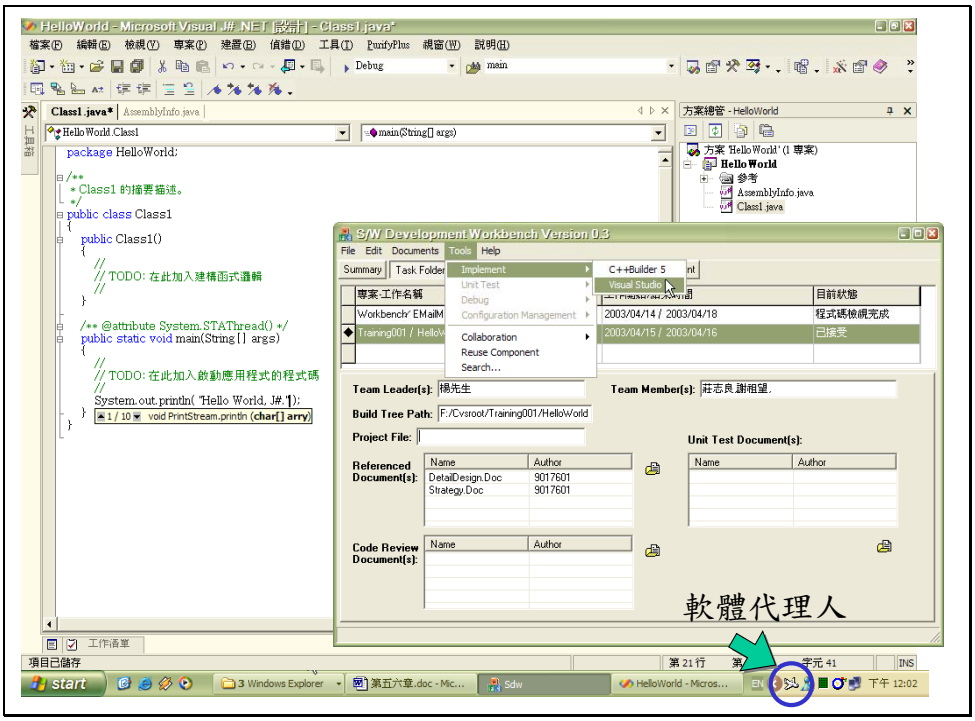
- 模組資訊收集機制:

  程式設計師登入 Software Workbench 之後，並進行程式的開發



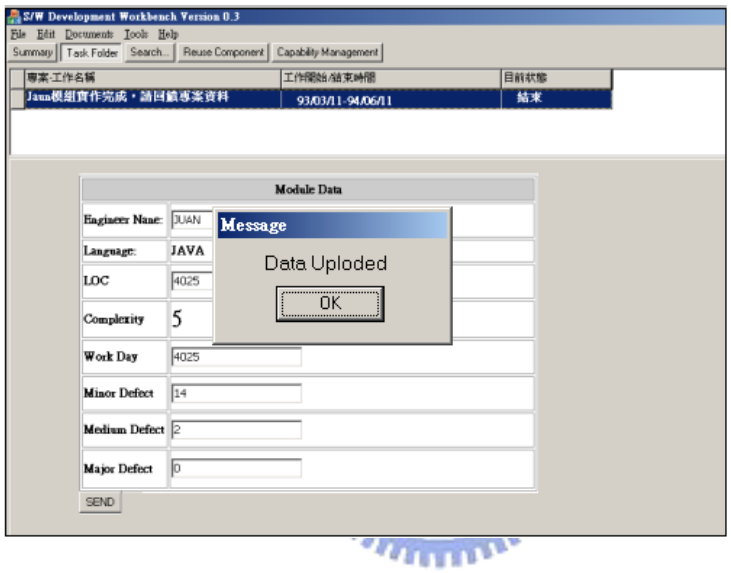
圖表 5-4Software Workbench 工作介面一

使用者可點選軟體發展環境選單：Tools→Implements→Visual Studio，系統即叫出 Visual Studio 開發工具供使用者進行程式碼的撰寫，系統並啟動軟體代理人來執行該工作相關文件的檢視，相關文件檔案會直接在 Visual Studio .NET 環境中開啟並顯示，使用者毋須在開發工具與軟體發展環境之間來回切換便能檢視文件內容。



圖表 5-5Software Workbench 工作介面二

當程式設計師完成了工作之後接著點選 Tools→Configuration Management→WinCvs 介面，開啟該工作的軟體產物上傳視窗，將相關的軟體產物上傳到伺服器端儲存並結束該項工作。而系統將會回應一個 Module Data 之資料給能力資料庫，並在工作完成之時回傳一個 Data upload 之介面。



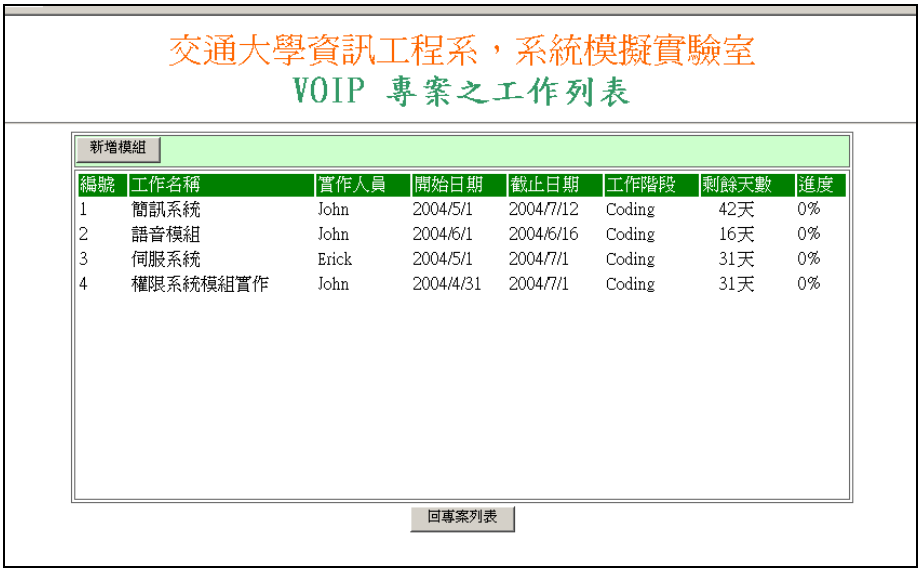
圖表 5-6 專案資料上傳能力庫結束

- 專案人員挑選機制：

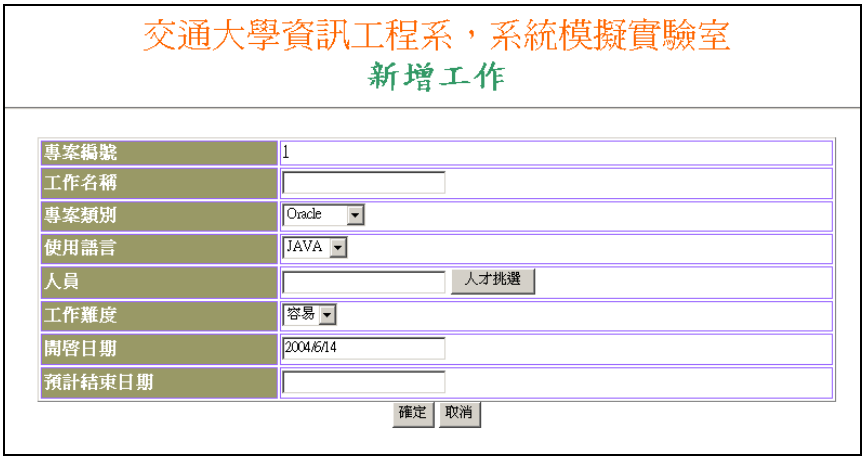
  Manager 在登入 Software Workbench 之後，點選 Tool→ Management->Project_List。而 manager 將針對 VOIP 進行管理。

圖表 5-7 進行中專案列表

Manager 在專案工作列表中，點選新增模組。

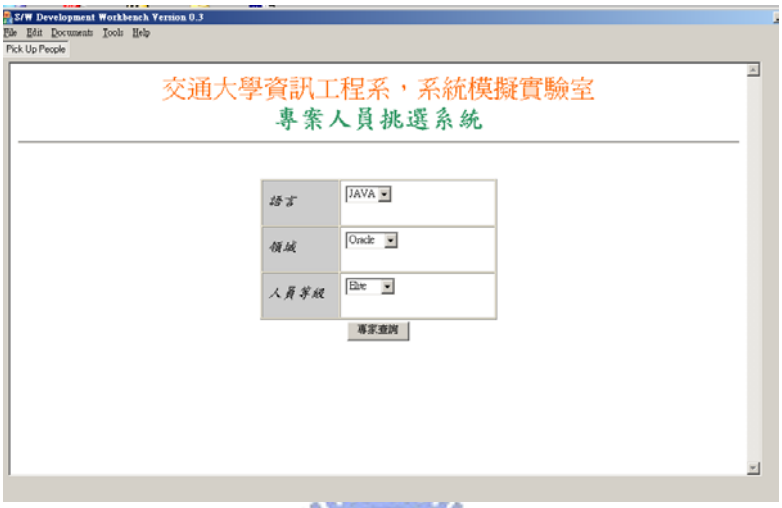


圖表 5-8 檢視工作介面



圖表 5-9 新增工作介面

在 Manager 登入了 Software Workbench 之後，點選人才挑選，人才挑選則會連到能力資料庫之專案人員挑選網頁。Manager 選取其所需之人才條件，如 JAVA 語言，Oracle 之菁英人員。在設定完之後按下專家查詢之按鈕。



圖表 5-10 人才挑選介面

在三項條件均提供之下，則可得目前之符合條件之人員。而若經過複步之檢視還是無法決定人選則可以再點選進階「查詢」。

圖表 5-11 待選人員列表

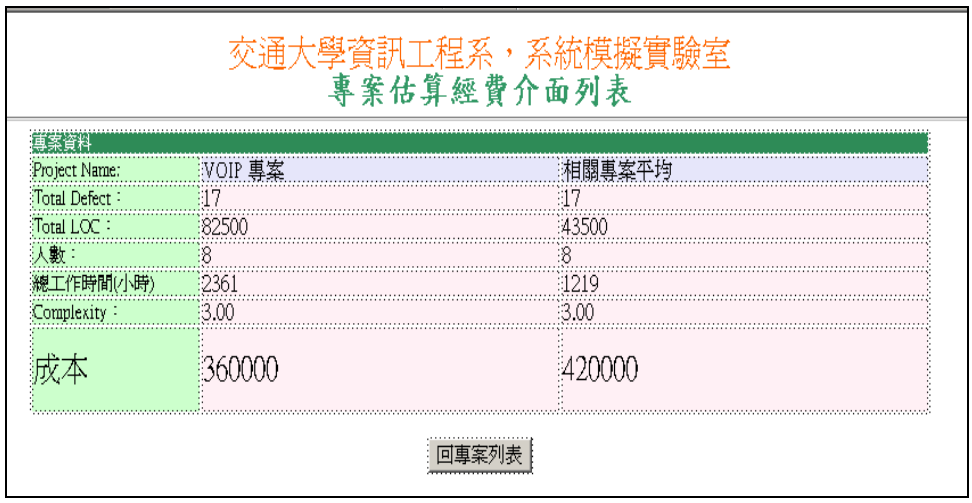
個人之資料可以將分為語言能力及和領域知識（Oracle）能力之資料。此部分之內容將包括語言之成熟度數據，及 Oracle 相關之模組的實作經驗。而 Manager 可以藉由這些資訊，而行更詳盡之專案人才挑選。

圖表 5-12 專案人才之詳細資料檢視

若決定了人才之挑選，，則可在待選人員列表上點選挑選之選項。則可以在 VOIP 專案列表中新增一項內容。利用效率和過往之專案經驗則可以幫助 Manager 進行更合理的時程預測及程式可能行數預測。

圖表 5-13 人才挑選完畢

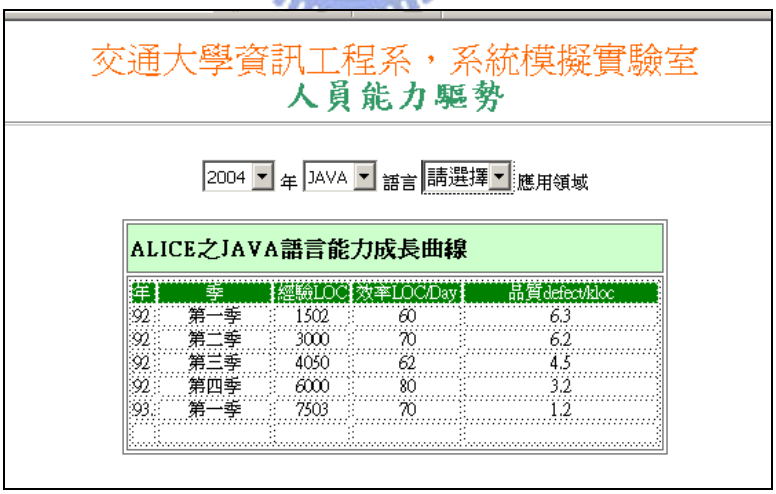
- 成本預估:Manager 在完成了 VOIP 專案之工作之後，可以利用能力庫中人員之薪資，及工作的天數，估算出此專案於開發所需之成本。



圖表 5-14 專案經費估算

- 人才能力監控:

Manager 可以藉由 Software workbench，可以提供 manager 對人員能力之觀察，在登入 Software workbench 之後，點選 Tool->Management->View Engineer Capability。而在人員列表中選擇了 Alice，即可以看到 Alice 之能力成長曲線。而在此介面之中，可以選取不同之語言，不同之應用領域。而此工具的提供，將可以幫助進行對於程式設計師之能力的監控。



圖表 5-15 驅勢檢視介面

第6章 結論

軟體是一「以人才為本」的產業，從 PSP、TSP、CMM 等研究已證實，軟體開發人員能力的提升是加強軟體品質及開發效率的一個極為有效的方法，因此如何做好軟體人才的能力管理是軟體企業的重要工作。軟體人才除了要做好專長及能力之分類外，更重要的是對其能力之評鑑，能力評鑑須從其平常工作表現的資料取得。若有軟體開發平台，軟體開發人員的工作表現，可由其開發成果取得，依據構想，本實驗室廖元誠提出一套「具知識管理與能力管理」之軟體開發平台，然而此平台中對於工作成果資料之取得有完整的設計，但在專長能力管理之分類、評鑑及應用則鮮少交代，本研究延續其研究，在此平台進行專長／能力管理子系統之設計。

由於軟體人才角色之多樣，能力分類過於複雜，因此本研究先從程式設計師專長能力管理著手，藉由對程式設計師工作的分析，將程式設計師之能力依程式語言及應用知識來做分類。並用實作經驗、品質及效率三大指標，來評鑑其能力成熟度，以依成熟度之不同將程式設計師之能力分成新進、一般、菁英及專家等四等級。

分析程式設計師能力的精進發現軟體缺失是造成程式設計師工作效率下降及品質不彰之重點，因此本研究提出了一個錯誤知識庫的概念，應用軟體缺失之 80/20 理論，利用前人錯誤知識的保留與記錄來幫助新進之人員解決程式之錯誤，並利用教材和 Checklist 等機制來幫助降低程式設計師之錯誤的產生。而藉由此方式，程式設計師之能力將被不斷地精進。

本研究將專長能力資料庫及錯誤知識庫建構在本實驗室廖元誠所開發之 Software Workbench，而構成此軟體平台之專長能力管理子系統

本研究之主要貢獻為:

(一) 提供程式設計師之能力評鑑機制：建立起專長導向之評鑑機制，利用模

組實作之資料，提供程式設計人才能力評鑑機制。

(二) 錯誤知識庫建立：錯誤知識庫可幫助程式設計師預防錯誤發生及加速偵測錯誤，提升程式設計師之工作效率及能力。

本研究之未來研究方向有:

(一) 更完整之程式設計師能力模型：針對如程式設計師之溝通能力、合作能力及創新能力等這些非顯性之能力，提出能力模型及評鑑機制。

(二) 更完整的軟體人才角色能力模型：目前僅有程式設計師一個角色，而未來將軟體人才之能力管理擴展到包括系統分析師、系統設計師及測試工程師等人員之能力評鑑。

Reference

[1] P. Naur and B. Randell (eds), *Software Engineering: A Report on a Conference sponsored by NATO Science Committee*. NATO 1969

[2] 劉文謙,施向玨與鍾乾癸, “軟體知識管理,” 第十三屆物件導向技術及應用研討會, 臺中縣 霧峰鄉 臺中健康暨管理學院, 中華民國九十一年九月十三日

[3] Charles W. Krueger, “Software Reuse,” *ACM Computing Surveys*, Vol. 24, No. 2, June 1992, pp. 131-183

[4] R. Davis, P. May, D.R. Wardell, and T. Wooding, "Techniques for Developing Reusable Business Components," *J. of Object-Oriented Programming (ROAD)*, Vol. 9, No. 7, pp. 40-43, Nov.-Dec. 1996

[5] Ruben Prieto-Diaz, "Status Report: Software Reusability," *IEEE Software*, pp. 61-66, May 1993

[6] Even-André Karlsson, *Software Reuse : A Holistic Approach.* NY: John Wiley & Sons, 1995

[7] Michel Ezran, Maurizio Morisio, Colin Tully, “Success and Failure Factors in Software Reuse,” IEEE Transactions on Software Engineering, 2000

[8] 施向玨，軟體開發實作階段的知識管理，碩士論文國立交通大學資訊工程學系, 2002

[9] 廖元誠，結合知識管理與能力管理之軟體發展環境，碩士論文國立交通大學資訊工程學系, 2003

[10] Watts S. Humphrey, “Introduction to the Personal Software Process”, Addison Wesley Longman, Inc., 1997

[11] Watts S. Humphrey, “Introduction to the Team Software Process”, Addison Wesley Longman, Inc., 2000

[12] Bill Curtis, William E. Hefley, Sally Miller, “”,

[13] Frank Maurer, Harald Holz: Integrating "Process Support and Knowledge Management for Virtual Software Development Teams," Annals of Software Engineering, Vol 14, 2002

[14] Britta Hofmann, Volker Wulf, "Building Communities among Software Engineers: The ViSEK Approach to Intra- and Inter-Organizational Learning," 4th Learning Software Organization, Chicago, Illinois, USA, August 6, 2002

[15] A PRACTICAL APPROACH TO COMPETENCE MANAGEMENT THROUGH METRICS IN INNOVATIVE ORGANISATION

[16] Capers Jones, “Software Quality: analysis and guidelines for success”, Thomson, 1997

[17] Roger S. Pressman," Software Engineering a practitioner's approach", 5th Edition, McGraw Hill

[18] B. Hughes, M. Cottrell, "Software Project Management",3rd Edition McGraw Hill

[19] Tomas Hellström, Peter Kemlin, Ulf Malmquist ,"Knowledge and Competence Management at Ericsson: Decentralization and Organizational Fit"

[20] Gray Hamel,and Aime Heen,Competence based competition

[21] http://www.sei.cmu.edu

[22] D.Yakimovic, G.H.Travassos,V.R Basili. "A Classification of Software Components Incompatibilities for COTS Integration"

[23] Jeffery S. Poulin "Experience with a Faceted Classification Scheme in a Large Reuse Software Library (RSL)"

[24] http://www.laatuk.com/tools/testing_tools.html

[25] http://www.acm.org/class/1998/

[26] http://www.msi.ms/MSJ/People-Capability_Maturity_Model.htm