國立中山大學 資訊管理 學系碩士在職專班

碩士論文

以功能點估算軟體規模之研究

研究生：王德榮撰

指導教授：鄭炳強

中華民國 九十二 年 六月

致 謝 詞

首先感謝中山大學的在職專班制度，使我在大學畢業二十五年後，能有一個再度進修的機會。浸潤在依山傍海的學術殿堂裏，與師長們互相切磋著學術理論與管理實務，不知不覺中二年已飛奔而過，雖然歲月使人早生華髮，但伴隨著我的卻是終身學習的喜悅。

本論文之完成，先要感謝指導老師 鄭炳強博士的悉心教導，老師不僅在教學、研究之餘，撥出與家人相聚的寶貴時間，給予啓迪與教導，更讓我自由翱翔在軟體工程領域中，揮灑著資訊管理實務經驗，在此對老師致上無盡的敬意與謝忱。

其次，感謝本論文口試委員 趙善中博士及高雄師範大學葉道明博士的悉心指正論文缺失，尤其是趙老師的再三叮嚀，使得本論文得以更趨完整。

再來，感謝資管所學長，也是大學老同學的蔡振昆，當初他的鼓勵，激起我前來中山資管專班進修的意志。也感謝眾多專班及一般生同學的相互支持，讓我有毅力在夜深人靜時繼續奮鬥。

最後，感謝內人瑞娥及兩個兒子的打氣，使我順利完成學業。

王德榮　謹識于
中山大學資管專班
民國九十二年七月

論文提要

學年度	091
校院名稱	國立中山大學
系所名稱	資訊管理學系
學位類別	碩士
論文名稱（中）	以功能點估算軟體規模之研究
論文名稱（英）	A Study of Software Size Estimation using Function Point
語文別	中文
學號	9042302

研究生姓（中）	王
研究生名（中）	德榮
研究生姓名（英）	Wang, Der-Rong
指導教授（中）	鄭炳強
指導教授（英）	Jeng, Bingchiang

關鍵字（中）	軟體工程；軟體規模估算；功能點
關鍵字（英）	Software Engineering；Software Size Estimation；Function Point

提要（中）	從程式設計到軟體專案管理，軟體規模估算一直都是一件困難度很高的任務。本研究發展出一套方法，以功能點分析來衡量軟體的規模。由於本研究之對象為一成熟企業，具有 ERP 資訊系統之成熟度與人員低流動率之特性，因此，我們收集程式設計與單元測試兩階段的人日耗用資料，利用迴歸分析建立起一套軟體規模估算模式。經過測試資料驗證，其預測的準確性可以高達 90%左右，未來不僅可用於企業內部資訊資源的分配，亦可用於資訊工程師的績效評估。
提要（英）	Software size estimation has been long a challenging task over a software development process. This paper presents an approach that uses the function point analysis to estimate program coding and testing effort in a MIS department, which maintains an ERP system with low employee transfer rate. The method first analyzes the historical data using regression analysis, and then builds a software estimation model with elaborated coefficients for related parameters. The estimation model is tested with the remaining set of historical data to evaluate its predict accuracy. It is shown that the size estimation model can be as accurate as about 90% correctness. Thus it is useful not only in company-wide information resource allocation, but also in performance evaluation of software engineers.

提要開放使用	是
頁數	52

目　錄

圖表目錄

第一章 緒論

第一節 研究背景與動機

近年來，在知識管理、電子商務、供應鏈管理及顧客關係管理等資訊技術的大力推動下，企業資源規劃系統（ERP-Enterprise Resource Planning）不僅持續擴大其規模，同時也變得更複雜，資訊主管要去瞭解它已經很困難，更遑論其他的評估衡量工作。發展 ERP 系統就如同企業追求永續經營一樣，須制定資訊發展策略。企業資訊資源－不論是資金、軟硬體設備、開發人力及時間都有限，而資訊需求－不論是新應用系統開發或者舊有系統的增強卻源源不斷而來。在面對競爭的時代，如何制定一套軟體估算模式，據以運用有限資源，來達成企業最大的資訊效益，已成為資訊主管必須面對的課題。

國內企業資訊部門，過去的軟體規模估算（Software Size Estimation）都依賴專家判斷法，即資深工程師以其類似或已完成專案的經驗為基礎來估計新專案。若低估了軟體規模，會造成資源的不當使用，使工作無法如期完成；而高估了軟體規模，卻會對其他工作造成排擠。總之，不適當的軟體規模估算，對資訊部門的資源運用，會是不小的衝擊。

學術上常被提及的功能點分析法，是一種結構化解決軟體規模估算問題的技術，它將資訊系統功能逐步往下細分至較小的構面（Component），使得系統較容易被瞭解與分析。功能點分析法用於單一系統的軟體規模估算，由文獻上得知是相當被肯定的，惟企業內各資訊系統間之關係錯綜複雜，若需引用功能點分析法，應在構面選擇與構面評價上做修正。另外使用功能點分析法，須對其理論有相當透徹的瞭解，以近年來企業人力精簡的現況，是不容易培訓功能點估算的分析人員。因應企業的現況，基礎於功能點分析法，將其軟體規模估算的範圍與估算模式，做適度的調整，是有其必要。

第二節 研究目的

功能點分析法用於新開發專案的規模估算，一旦系統的輸入、輸出畫面及產出的報表、檔案確定，即可由交易與檔案的複雜度相互對照，查表計算出功能點。現有 ERP 系統的需求，往往只是某一部份功能的替換增強，有些時候會跳過系統分析階段，直接進行程式設計即可滿足需求。若仍循序計算功能點，軟體規

模恐會有高估之虞。

本研究的目的，在提出一個適用於現有 ERP 系統的軟體規模估算的模式，基礎於功能點分析法，延續其構面的選擇、構面的複雜度分類及構面的功能點賦予等架構，並依實務經驗作修正；再將軟體規模估算的範圍，縮小至程式撰寫與單元測試階段；然後收集企業過去的實際資料，利用迴歸分析，求出各構面依功能需求分類的功能點與每一功能點代表的衡量單位如人日、人月等等。

以新程式各構面的功能點與每一功能點代表的人日，所估算出的人日，可用於估算業務單位提出工作的規模。資訊主管不僅能用以掌控資訊資源、制定年度工作計畫，也可使用每月工作完成的功能點，來衡量工程師的生產力。

第三節 研究流程

首先對軟體規模估算的相關文獻做一探討，以瞭解各種估算方法的優缺點，並選定一個值得研究的方法，設計出軟體規模估算的數學模式。然後以個案公司－中鋼公司的開發環境特性，準備適量的訓練資料，以迴歸分析方法修正該數學模式。最後再以其他的歷史資料，驗證數學模式的準確性，並對本研究做出結論以及後續研究做出建議。研究各步驟的關係與流程可用圖示，詳如圖 1-1：

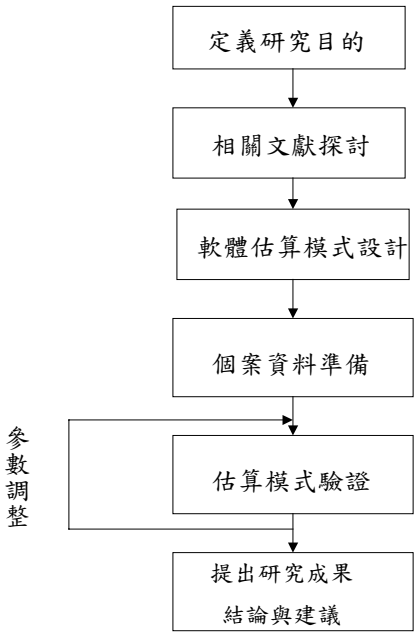

圖 1-1　研究流程圖

第四節　研究範圍與論文架構

一、研究範圍

軟體規模的估算可大至整個軟體專案管理活動的生命週期，包括需求分析、系統分析、系統設計、程式設計與測試、系統測試及系統上線，小則僅涵蓋程式設計與單元測試的階段。

軟體工程（Software Engineering）兼具工程與藝術特性，大的軟體專案活動，由於牽涉面廣泛，所以較偏向工程面考量，而小型的程式設計與測試階段，牽涉到工程師個人學養，差異頗大，藝術性的成分較重。一個開發環境近乎標準化且人員流動率低的資訊組織，以過去的程式設計與單元測試階段的人日歷史數據，經過迴歸分析後，用來預估新程式設計與單元測試的人日，技術上應屬可行。

因此，在我們的研究中，軟體規模估算即以程式設計與單元測試為範圍，期以研究出一套資訊人力資源估算的模式。

二、論文架構

本論文共分六章，各章內容摘要如下：

第一章為緒論。敘述本研究的背景與動機、研究目的、流程、研究範圍及論文架構。

第二章為文獻探討。從資訊專案管理三大支柱，工作時程、成本及交付產品間之平衡，引發資訊資源的運用與分配，為何要做專案組合的選擇，而談到專案（軟體）規模估算的重要性。然後論及軟體規模估算的行數導向衡量、功能導向衡量及機能導向衡量的幾種作法與優缺點，最後並選擇出以功能點分析方法，做為研究軟體規模估算的基礎。

第三章為功能點分析方法。詳細介紹功能點分析的設計架構與其估算構面，包括外部輸入、外部輸出、外部查詢、內部檔案及外部檔案。何謂未調整功能點？各構面未調整功能點的評價條件為何？如何考量應用系統特性去決定調整因子？如何運用功能點去執行資訊專案管理？再論及其優缺點與使用上如何調適。

第四章為軟體規模估算的數學模式。基礎於功能點分析衡量方法的構面選擇與構面分類設計架構，本研究侷限於現有應用系統增強時，以工程師執行程式設計與單元測試兩部分之人日消耗，設計出軟體規模估算的數學模式。

第五章為個案研究：中國鋼鐵公司。以該公司的軟體開發環境，探討第四章的體規模估算數學模式，選擇影響程度較大的構面，包括傳輸介面、相關連系統、更新檔案、使用的業務單位及連結公用模組等。先以該公司九十一年十月至九十二年三月間的程式設計與單元測試訓練資料六十三筆，經過迴歸分析運算，實證軟體規模估算的數學模式，再以另外的五十筆資料去檢測該數學模式的準確性。

第六章為結論與建議。總結研究的成果，提出一個可供其他行業參考的軟體規模估算模式與施行步驟；闡明此估算模式的優點、缺點及其限制，及提出一些未來在企業實務上或學術界上，仍值得去努力探討的空間。

第二章 文獻探討

第一節 軟體規模估算的探討

一、資訊管理基礎

資訊專案管理控制著三角形的三個端點，即在工作時程、成本、和需求功能（產品）中取得平衡，其中時程可說是最重要的因素。有時為了在既定的時程內交付產品，不惜增加成本或遲緩部分需求功能，以趕工的方式完成專案。時間與成本實為一體兩面的關係，為了趕時程，專案管理會採增加人手或者加班的方式因應，不管用那一種，都會排擠到其他專案的資訊資源【鄒正平 99】。時程壓縮是軟體專案管理最常遭遇的問題，造成時程壓縮的原因，不外乎業務單位(User)已訂定底線，或者資訊部門缺乏估算模式，工程師也無過去的經驗可供參考，因而低估了專案的規模。就時程、成本、和需求功能三者關係而言，時程估算不足，必然會使專案成本增加或者犧牲專案的部份品質。

資訊管理是包含估算需求功能的規模大小，去取得適合這個功能規模的資訊資源，制定一個管制資訊資源的工作計畫，然後以監控與管理手段，妥善運用資訊資源，及時在適當的成本下，完成資訊化工作。執行順利的專案會經歷三個基本步驟，來製訂一個軟體工作時程。首先估算專案的規模，再來是估算建立專案所需投入的資訊資源，然後以專案可投入的資源來估算整個工作時程。

軟體專案管理能維持長期進步的關鍵之一，就是持續收集計量資料來估算軟體規模與分析資訊資源的耗用。由估算與分析中，制定出分配與運用的模式，使有限的資訊資源得以發揮其最大的效益。

二、專案組合選擇

專案組合的選擇（Project Portfolio Selection）起因於公司的各種資源有限，為確保資源有效運用，專案必須有所取捨。有許多不同技術可用來評估選擇專案組合，但有的太複雜，要求太多的資料輸入，或提供不合適的風險與不確定因素處理方式，有的相互關聯性規範做的不好，或太難瞭解等，導致這類技術並沒廣泛使用。由於全球化競爭的壓力，在 1990 年起專案組合分析與規劃變得和 1970、1980 年代的企業組合計畫一樣重要，公司希望在有限資源限制下，透

過一些模式，選擇適當的專案加以執行，以提升公司的競爭力。

專案組合選擇是一個重要且循環性的活動，不斷地將新提出的專案納入並考量已進行中的專案，在不超出可用資源或違反其他限制下，選取專案組合，來完成公司最佳目標。專案組合選擇包括下列三個階段【Norm 99】:

1、策略考量階段（Strategic Considerations Phase）

在專案組合挑選之前，策略決定組合重點和全面預算考量，這裡面必須包含企業內外部相關因素等廣泛內容，決策者用來評估公司目前的市場定位與將來競爭優勢，如是否引進供應鏈管理技術，建立知識管理平台等。在專案展開之前，資訊部門必須非常清楚公司的策略方向，儘早刪除不符合策略的專案。

2、專案評估階段（Individual Project Evaluation Phase）

一個專案是否值得去投資進行，須對專案先做評估，常見的評估項目包括：

‧財務的回收－包含Net Present Value(NPV), Internal Rate of Return(IRR), Return on Original Investment(ROI), Return on Average Investment(RAI), Pay Back Period(PBP),及 Expected Value(EV)等，這些技術都依時間性，考量投資與收入的現金流量。

‧獲利／成本比例計算－依相同基礎計算獲利與成本的現值。

‧風險評估－風險是一件無法預期發生的事件與相關事件後果的整合，每個專案均有無法達成目標的風險，要分析專案風險，第一步要先分解專案成細項活動，建造專案的 Work Breakdown Structure(WBS) 【林信惠 02】，再依據軟體生命週期劃分的階段，鑑定出各 WBS 進行的深度與可能的風險。有關的每個活動事件風險確認後，要進行可行性與後果評估。評估風險資料可來自於專家意見、技術資料、或之前類似專案經驗，以風險模型整合每個活動或活動相關事件的風險，然後評估出全面性專案風險，分析風險的模型包括 Monte Corlo Simulation，Decision Theory 等，當考慮組合專案時，風險是非常重要的，必須避免過高的風險，導致危害到組織的未來。

每一種型態專案之評估方式不盡相同，最好是幾種方式一起使用。選擇共同量測基準來分別計算每一個專案，在組合選擇過程，對各專案有一個公平公正的比較。進行中的專案要定期再評估，以確認資源使用情況，是否值得繼續執行？再評估時機如下：

- 某一專案結束或者放棄時；
- 其他新專案提出計劃書時；
- 當公司資訊策略變更時；
- 當資訊資源有所調整時；
- 當經營環境改變時。

3、專案組合選擇階段（Portfolio Selection Phase）

專案組合選擇包括相當多的專業，各專案在特定的尺度上同時進行比較，希望排入值得進行的專案，目前可用的組合選擇技術類別如 Ad hoc approaches, Comparative approaches, Scoring models, Portfolio matrices 及 Optimization models。專案組合選擇應考量資源競爭的交互作用，並把時間依存性質的專案資源消耗列入計算，決策者必須要求各專案提供互動機制，用來控制與調整任何模型所產生的組合選擇，並及時得到調整後的回饋。

三、專案規模估算

對資訊主管而言，正確估算資訊系統的開發成本是一個重要的議題。當成本估算者低估了成本時，開發人員會被委任新的專案，這不僅浪費有限的資源，與不能完成預期貢獻，也破壞了估算者與開發人員的信用。當估算人員高估了成本時，則開發人員就會被禁止去開發其它的專案。所以一個錯誤的資訊系統成本估算，對組織而言是一個很大的衝擊。

事實上有研究指出，三分之二的專案開發其成本是被高估的。Albert L. Lederer and Jayesh Prasad【Albe 92】在其研究中指出，根據 115 位資訊主管與專家指出，成本估算是採一般的估算方式，即採用慣例而忽略了大量的過去的研究。以往的成本估算大多定義在演算的技巧上，大多數的研究會定義一些構面，這些構面是被相信會影響資訊系統的開發，這些構面包括系統的大小、系統的複雜度、個人的能力和經驗、硬體設備、新工具的使用、使用者對資訊技術的認知與需求等等。估算者根據過去系統開發的資料或直覺上的認知做估算，但這易流於偏頗與不切實際。

同一研究並對專案成本估算提出一些準則（Guidelines）如下：

- 將最初的估算工作指定給最後的開發者；

・儘量延遲宣布成本的估算，直到瞭解整個專案評估的結果接近正確為止；

・預估且控制使用者的需求改變之影響因素；

・利用成本估算來監督專案之進度；

・藉由獨立稽核評估計畫的進展；

・依靠足堪證明之事實、標準及簡單數學公式，比猜測、直覺、個人記憶或複雜的公式來的好；

・不要依賴預估軟體 (Estimating Software)去做預估。

另外，該研究也提到，專案被提出時，專案主管必須就下列問題先有瞭解：

・要付出多少的人力才能完成此專案；

・需要多少的日曆天才能完成此專案；

・此專案要花費多少成本。

要解決上述問題，首先要能估算軟體的規模。

第二節 軟體估算的方法

一、軟體估計的分類

軟體估計方式通常依 Boehm（1984）的分類如下【Rich 85】【Step 93】【Ians 01】:

1、演算法：採用成本導向，以目標系統重複出現的某些屬性與環境的構面預測需投入的資訊資源。例如功能點演算法，從程式的功能性來估算程式的規模。

2、專家判斷法：依賴一個或多個專家的經驗來做估計，其判斷的資訊包括經驗法則、可用的資源、過去專案的資料、過去估計的回饋及過去類似專案的詳細功能等。

3、類比法：由類似與已完成專案的實際成本為基礎來估算新專案，估算新系統的每項主要構面就如同舊系統相類似構面規模的百分比,將每個構面的估算規模加總起來就是新系統的總規模。

4、巴金森法：認為工作的多寡是將所有資源耗盡為原則，因此著眼於有多少預算與多少時間就做多少事。工作會展開到用完所有能使用的資源，並以此為基礎導出估計。

5、價格勝算法：以爭取合約為原則，因此以足夠取得合約的價格為基礎所做的

估計。

6、由上而下法：由整體估計專案需投入的工作量，再往下分割到個人負責的細部內容所需的工作量。

7、由下而上法：由個人負責的細部內容做區別估計規模大小，再整合成整個專案的估計方式。

Robert T. Hughes【Robe 99】在其研究中指出，巴金森法被質疑並不是針對投入多少工作量的估計方式，卻是設定專案範圍的方法；價格勝算法是一種訂價格的策略而不算是預估；而專家判斷法，通常會有輕鬆的工作被低估，困難的工作就高估的傾向。Boehm 說明事實上，為綜合各種方法的優缺點，傾向依情況互相搭配。如由上而下的方法與由下而上的方法結合，演算法結合專家判斷法。一個典型的組合技術，是使用不同的方法將結果拿來比較，經反覆估算後，不一致的地方自然會呈現出來。Ian Sommerville【Ians 01】則認為此類估計方式，皆可使用由上而下或者由下而上的應用，來完成軟體估算。由上而下應用，即由系統功能自上而下，細分至子功能收集成本，其優點是可同時收集整合、組態（Configuration）及文件整理成本，缺點是某一子功能有非標準介面需求產生時，成本會增加；由下而上應用，即先建立一個雛型系統，將系統切成組成塊，再逐一由下而上加總算出成本，其缺點在系統整合的成本不易估算。

二、演算法

演算法又稱參數模式或統計模式，它是軟體估算研究中最引人注意的方法。這類模式的基本概念是，軟體估算的兩大變數為軟體規模與調整因子，軟體規模的單位可為原始碼（Source Code）、功能點（Function Points）或其他。

專案的生產力可以軟體產出的功能單位數，除以專案的使用總人月來估算，由於有許多的解法用來處置軟體需求，因此專案生產力的估算也就不具客觀性。生產力的估算是資訊主管的任務，可以用來評估開發過程或者技術性的改善工作是否值得。Roger S. Pressman 等【Roge 01】【Ians 01】指出演算法常用的生產力衡量有三種：

1、行數導向衡量（Size-related Measures）

指軟體活動所產出的行數，最常用來做行數衡量的是交付的原始碼，其他被

選用的尚有執行碼（Object Code）指令數，交付的系統文書頁數等。原始碼衡量（LOC-Lines of Code）是直接衡量的方法，與使用的程式語言相依附，簡單且最被廣泛使用，但也缺乏公信力。其衡量方式，即以專案產出的原始碼數除以專案的全部人月，此人月包括分析與設計、程式撰寫、測試及系統文書的所有人月。LOC 亦可用作其他衡量指標，如錯誤數/KLOC（Thousand LOC）、瑕疵數/ KLOC、費用/LOC 或者文件頁數/KLOC。

2、功能導向衡量（Function-related Measures）

指軟體活動所產出的功能，生產力是以某一限定時間的功能數產出來表示，常用的衡量方式包括功能點與物件點（Object-points）。

（1）、功能點衡量

功能點衡量是間接衡量的方法【Step 93】。最早由 Albrecht（1979）提出，再經 Albrecht 與 Gaffney（1983）修正，它的優點是避開選用不同程式語言撰寫造成的原始碼誤差，功能點考量構面包括：

- 程式的輸出與輸入－包括畫面螢幕欄位、對話框、控制、或訊息等。
- 與使用者的互動－如使用人數、使用者的資訊經驗等。
- 外界的交換介面－主要的邏輯資料群體或進入與離開程式的控制資訊。
- 系統使用的檔案數－檔案可能是由單一的平面文件或一個相關資料庫的單一表格所組成。

由上述構面，資料項數（Data Element）對照檔案數或記錄數（Record Format）之關係，求得各別的功能點，再彙總算出未調整功能點。以未調整功能點，再考慮應用系統的複雜度做進一步調整，考慮項目如是否為分配式處理作業、模組重複使用程度及專案績效等，計算出調整後的功能點。Furey & Kitchenham（1997）指出每一估算者對調整係數的考量不一樣，因此最後的功能點是會有所誤差。雖如此，部分學者 Kemerer（1992）仍認為功能點衡量是一個很實務的演算法。

（2）、Full Function Point 衡量

COSMIC-FFP 是由 Common Software Measurement International Consortium 協會所推廣的軟體功能衡量法【Serg 99】，它是一種不牽涉到語言技術，也不牽涉到開發程序的衡量法。其步驟有二：首先，依應用系統（Application）、作業系統（O.S.）及載台（Drivers）界定使用者的需求，然後去衡量軟體的基本功

能構面（BFC-Basic Function Component）。

BFC 是將系統拆成許多的資料移動的子處理（Sub-processes），子處理共有四種，Entry、Exit、Read 及 Write。Entries 指資料從系統外部搬到內部；Exits 指資料從系統內部搬到外部；Reads and Writes 指從系統做檔案輸出、輸入到儲存體（Storage）。每一個子處理得到一個功能單位（COSMIC-FSU, Function Size Unit），最小可衡量的是 2 個 COSMIC-FSU，因為構成系統起碼有一個輸入（Entry or Read）與一個輸出（Exit or Write）。

（3）、COCOMO 衡量

COCOMO（Constructive Cost Model）模式是個非線性的模式，由 Boehm 所發表【Rich 85】【Step 93】【鄒正平 99】【林信惠 02】，它分為三個不同的詳細程度，稱為基本模式（Basic Model）、中級模式（Intermediate Model）及詳細模式（Detailed Model）。每一個模式都將軟體分為三個複雜程度，分別稱為簡單型、中等型及複雜型。是適合於以第四代（4GL）等語言，使用物件導向開發系統的衡量方式。以簡單型為例，物件導向程式大小的衡量，可以下列因素去估算：

- 顯示的螢幕數，例如簡單的螢幕給予 1 物件點，中等的給予 2 物件點，複雜的則給予 3 物件點。
- 產出的報表，例如簡單的報表給予 1 物件點，中等的給予 5 物件點，複雜的則給予 8 物件點。
- 為供應 4GL 碼而開發的 3GL 模組數，每一 3GL 模組給予 10 物件點。

COCOMO 模式的優點是架構非常完整，其缺點是分類過於詳盡，若沒有龐大的資料庫去支撐，則參數估計的誤差會很大，COCOMO II 是其改善的模式。

3、機能導向衡量（Feature-point Measures）

在 1986，Software Productivity Research（SPR）因工程軟體與一些嵌入式軟體，相對一般商業軟體較為複雜，乃創立機能導向衡量【Rayb 02】。其估算是以演算法則的步驟（Algorithm steps）來分類，例如波音公司的 3D Function Point。機能導向衡量相對功能點衡量言，公信力較不足，這與機能導向衡量尚缺專責學術組織去推動有關。

第三節 估算模式的選擇

軟體規模估算模式無論是演算法、專家判斷法或類比法等，必須配合資料的收集。數據愈明確、數量愈豐富，則估算的準確性就隨之提高。資訊組織過去的軟體規模估算，幾乎全採行專家判斷法或類比法。倘若一個資訊組織已建立良好的制度，並能就以往軟體開發專案的數據加以整理，即可採行演算法估算。演算法是以迴歸分析的方法來估計，近年來人工智慧的發展，也提供了其他的方法論，如類神經網路法與基因演算法等【林信惠 02】。

軟體規模估算模式的選擇，必須配合組織的資訊政策與應用系統開發環境。首先要決定採用行數導向衡量或者功能導向衡量為主的模式，以第三代語言撰寫的程式如 COBOL、FORTRAN 等，可適用於兩種模式；物件導向（Object-oriented）語言或者程式語言含呼叫指令（Calling-command）時，採用功能導向衡量的模式較為恰當。當要求的準確性較高時，往往還要有其他衡量的模式再一起估算、比較。

行數導向衡量法簡單且最被廣泛使用，但有下列缺點【Step 93】:

- LOC 只能衡量程式撰寫與單元測試部份，卻無法有效衡量軟體生命週期的其他部份。
- 軟體專案使用兩種程式語言開發或者使用 4GL 開發，LOC 衡量就失去其意義。
- LOC 定義不易取得共識，是執行碼呢？還是包括資料定義部份的原始碼？或者連程式內的說明碼都可算進去？

估算模式的選擇，須搭配大量歷史資料的搜集、完整制度的建立及經驗的累積，最後要考慮經濟性與方便性。資料的搜集分析要花費大量成本，愈複雜的估計模式，需要投入愈大量的人力物力，不僅成本高，執行也困難。因此，資訊組織採用軟體規模估算模式時，應權衡各種因素，再做決定。

第三章 功能點分析衡量方法

第一節 功能點分析創立與運用

一、功能點分析的創立

在 1970 年代末期，IBM 公司認為有必要開發一套與程式語言無涉，衡量軟體發展成果的方法，它的一位研究人員 Allan Albrecht，於是發展出功能點估算。80 年代早期，IBM 公司 GUIDE 部門，將功能點技術整理並出版了計算功能點的手冊。80 年代末期一個以美國、加拿大為主的國際性學術組織 IFPUG（The International Function Point Users Group）組成了，1994 年 IFPUG 出版了 Counting Practice Manual 4.0 版，同時宣稱它一直繼承了 20 年前 Allan Albrecht 原設計的理念，即「功能點是一種量測，用來衡量電腦應用系統或者軟體專案的大小。這種衡量是從功能或者使用者的觀點來看，與所使用的程式語言工具、系統開發的方法論及專案開發團隊的技術與能力無涉」【Rayb 02】。

二、功能點分析的運用

功能點分析的運用方面，較常提到的如下：

・衡量生產力－許多的高階經理常常忽略資訊團隊的貢獻，由每月完成的功能點，可讓他們知道公司的資訊團隊每個月做了那些貢獻。

・預估開發與維護的人力－既然功能點分析是一種估算技術，用來估算應用系統的成本效益是有其必要，即使不計成本的策略性專案，正確的估算仍有助於人力的取得。

・監督委外工作的進行－公司將部分或全部資訊工作委外（Outsourcing）已成趨勢，使用功能點計算可以衡量外包商，有否依合約承諾完成應交付的成果。一些軟體服務公司，常用完成的功能點來支撐其服務績效。

・決定資訊策略－公司必須就應用系統或專案計畫做組合（Portfolio）選擇，功能點的多寡是一個很好的追蹤指標，無論應用系統的維護（Retaining）、下線（Retiring）與重建（Redesigning）都可以使用功能點，參酌其他數據來做決策。

・正規化（Normalize）其他的衡量－功能點可以當作其他衡量的比較基礎，舉例說，交付 100 個功能點的應用系統，有 100 個缺陷不一定是很好的表現，但交付 5000 個功能點的應用系統卻只發現 500 個缺陷，那就是一件不容易達

成的工作。

第二節 功能點分析的設計架構

一、資料定義的抽象層次

應用系統是由一組處理元件（Elementary Processes）或稱為交易（Transactions）所形成，它們之間並不完全獨立，事實上，彼此交錯影響。功能點分析試著去瞭解交易與資料儲存間之動態關連，換個角度說，功能點分析協助定義資料的兩種抽象層次（ Abstract level），即資料停留（Data at Rest）與資料動向（Data in Motion）【Davi 02】。

1、資料動向

應用系統包含許多的處理元件（交易）會去移動資料，交易將資料從應用系統外面帶進領域裏面者，稱之為外部輸入（External Inputs）；交易將停留資料從應用系統送至領域外面者，稱之為外部輸出（External Outputs）或者外部查詢（External Inquiries）。

2、資料停留

應用系統儲存資料往往是為了後續的處理，停留的資料若是由應用系統本身自行維護者，稱之為內部檔案（Internal Logical Files），若是由其他應用系統維護者，稱之為外部檔案（External Interface Files）。

二、功能點分析的構面

功能點分析共有五項衡量構面，包括外部輸入、外部輸出、外部查詢、內部檔案及外部檔案。

1、外部輸入（EI-External Inputs）

將資料從應用系統的外面帶至裏面，資料可能是從畫面直接輸入或者從其他應用系統傳過來的。這些資料可以用來維護一個或多個內部檔案。資料可以是控制資訊，也可能是業務資訊，若是控制資訊，就與維護內部檔案無關連。

2、外部輸出（EO-External Outputs）

從應用系統裏面衍生資料到領域外面去，此外，這些外部輸出資料亦可能同時用來更新內部檔案。這些外部輸出資料，可以報表形式產生，也可能是一個輸出至其他應用系統的檔案。這些外部輸出資料，是多個內部檔案與外部檔案的資料經過邏輯運算或計算公式所衍生出來的。

3、外部查詢（EQ-External Inquiries）

從內部檔案或外部檔案的資料擷取後，顯示在畫面或報表上面，但它不是一個外部輸出，不能去維護任何的內部檔案或外部檔案，也不產生經過邏輯運算或計算公式所衍生出來的資料。

4、內部檔案（ILF-Internal Logical Files）

是一組使用者可辨認的資料，永遠保留在應用系統內部，其只能由外部輸入來維護，包括新增、更新或刪除。內部檔案至少會被一個外部輸出或者外部查詢使用到。

5、外部檔案（EIF-External Interface Files）

是一組使用者可辨認的資料，僅供參考使用。它存在應用系統外面，並由別的應用系統去維護。簡單說，外部檔案就是另一個應用系統的內部檔案。它至少會被一個外部輸入、外部輸出或者外部查詢使用到。

三、功能點計算的過程

功能點計算過程共有六個步驟，從決定系統的範圍、計算未調整功能點以至計算調整後功能點，步驟間並無絕對的順序：

- 決定功能點計算的型態
- 決定應用系統範圍
- 確認資料動向類處理元件（交易）並評價，以計算出交易未調整功能點
- 確認資料停留類檔案並評價，以計算出檔案未調整功能點
- 決定功能點調整因子 VAF（Value Adjustment Factor）
- 計算出調整後功能點

四、決定計算的型態與範圍

功能點計算包括了新開發專案功能點計算、重整專案功能點計算及現有應用系統增強功能點計算。而影響此三種型態功能點計算的因素有三，交易複雜度、檔案複雜度及應用系統特性 GSC（General System Characteristics）。

經由系統規範文書、使用者需求規範及資料流程圖等文書，並與使用者詳細面談後，確認應用系統範圍。系統範圍應及早確認，若應用系統屬於重整專案，其系統範圍應與原應用系統相類似；若屬新開發者，也應參考相類似應用系統以確定系統範圍。

五、確認交易與檔案複雜度

確認交易複雜度，須先解析應用系統包括幾個交易，每一交易是外部輸入、外部輸出、外部查詢交易型態的那一類。然後計算出每一交易使用了多少資料項（Data Elements），是屬於內部檔案或者外部檔案，以資料項數對照檔案數，歸納為低功能需求、中功能需求或高功能需求，再就不同功能需求賦予不同的未調整功能點（UFP-Unadjusted Function Point），其過程詳圖 3-1。

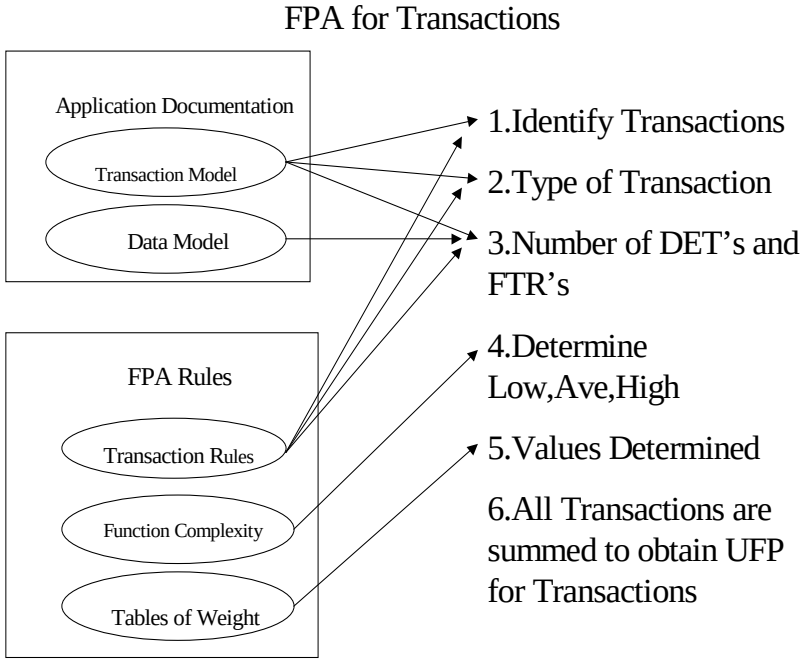

圖3-1 交易功能點分析過程
source from: www.SoftwareMetrics.Com

確認檔案複雜度，須先解析應用系統包括幾個檔案，然後決定是屬於內部檔案或者外部檔案，然後算出所有的交易使用了該檔案的多少資料項，再由資料項數對照記錄數歸納為低功能需求、中功能需求或高功能需求，最後就不同功能需求賦予不同的未調整功能點，其過程詳圖 3-2。

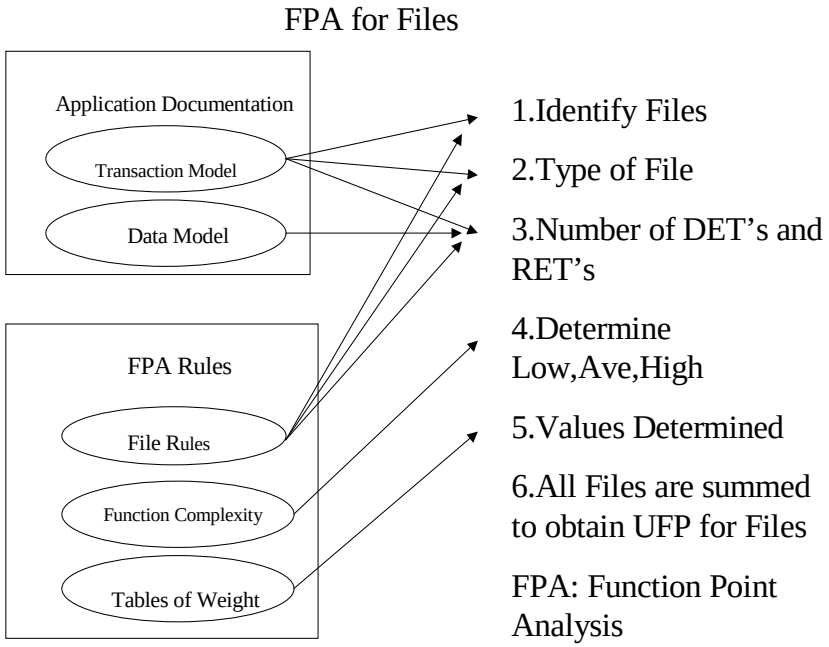

圖3-2 檔案功能點分析過程
source from: www.SoftwareMetrics.Com

第三節 功能點分析的構面評價

一、確認記錄數、檔案數及資料項數

功能點分析歸納為低功能需求、中功能需求或高功能需求時，須對照資料項，記錄或者檔案的個數。

1、記錄數－RET（Record Element Types）指存在內部檔案或者外部檔案中，而使用者能分辨的資料項子集合。

2、檔案數－FTR（File Type Referenced）指交易所用到的所有內部檔案或者外部檔案。

3、資料項數－DET（Data Element Types）指從內部檔案或者外部檔案或者輸出檔案上讀出的動態（Dynamic）資料項，若是重複定義者（Recursive）只計算一次。資料項數尚包括交易所用到的如畫面欄位、報表欄位、錯誤訊息、確認訊息、控制鍵及計算過程中的暫存資料欄位等。

二、構面評價條件

外部輸入、外部輸出、外部查詢、內部檔案及外部檔案構面對照記錄數、檔案數及資料項數之條件如表 3-1，所有構面評價都基礎於資料項數，但只就檔案數與記錄數兩項中挑一項。

表 3-1 構面評價對照

構面	記錄數	檔案數	資料項數
外部輸入（EI）		V	V
外部輸出（EO）		V	V
外部查詢（EQ）		V	V
內部檔案（ILF）	V		V
外部檔案（EIF）	V		V

source from： www.SoftwareMetrics.Com

三、各構面評價

1、外部輸入（External Inputs）

外部輸入交易通常緊跟在一個外部查詢交易後面，而一個外部輸入若同時引發新增、變更及刪除某個內部檔案，將視同是三個外部輸入，因此一個完整功能的資料輸入作業表示有四組功能點。

外部輸入之功能點由檔案個數對照資料項個數取得，並依低、中、高功能需求，分別賦予適當的未調整功能點，如表 3-2 所示，各別給予（3）（4）（6）三個等級的未調整功能點。

表 3-2 外部輸入未調整功能點

檔案個數	資料項數		
	1-4	5-15	大於 15
小於 2	低（3）	低（3）	中（4）
2	低（3）	中（4）	高（6）
大於 2	中（4）	高（6）	高（6）

source from： www.SoftwareMetrics.Com

2、外部輸出（External Outputs）

外部輸出交易是指讀取內部檔案與外部檔案的資料，經過邏輯運算或計算公式衍生出資料，形成檔案或報表送到應用系統外面，其功能點由檔案個數對照資料項個數取得，並依低、中、高功能需求，分別賦予（4）(5)(7）三個等級的未調整功能點。

資料項數考量應包括經過計算的數值（衍生出來的資料)、錯誤訊息、確認訊息及報表上出現原屬於內部檔案或外部檔案的資料等，但重複的欄位與數值只算一次。報表上的表頭資料或者作業系統資料，應忽略不計，但表頭資料是經由檔案上取得仍應計算。每一個外部輸出至少會關聯到一個內部檔案或外部檔案。

3、外部查詢（External Inquiries）

外部查詢交易其功能點由檔案個數對照資料項個數取得，並依低、中、高功能需求，分別賦予（3)(4)(6）三個等級的未調整功能點。外部查詢與外部輸出評價方式相似，唯外部查詢不能有衍生出來的資料，也不能去更新內部檔案，外部查詢只能提供從內部檔案或外部檔案上取得的資料。

資料項數包括查詢之輸入，如搜尋範圍、錯誤訊息、確認訊息及控制鍵；查詢之輸出如從檔案上搜尋到的資料、錯誤訊息及確認訊息等，但重複的欄位與數值只算一次。每一個外部查詢輸出至少會關聯到一個內部檔案或外部檔案。

4、內部檔案（Internal Logical Files）

內部檔案起碼會被一個外部輸出或者外部查詢使用到，同理，內部檔案也關聯到一個外部輸入。內部檔案其功能點由記錄個數對照資料項個數取得，並依低、中、高功能需求，分別賦予（7)(10)(15）三個等級的未調整功能點，如表 3-3 所示。

記錄個數是最難認定者，對一個檔案言，有可能擁有數個資料群（Data Group）格式，這時記錄個數也隨著增加。計算資料項數時，應從內部檔案去計算使用者能辨認的欄位，重複出現的欄位只計算一次，名稱不一樣但資料項定義相同的，也只計算一次。

表 3-3 內部檔案未調整功能點

記錄個數	資料項數		
	1-19	20-50	大於 50
1	低（7）	低（7）	中（10）
2 to 5	低（7）	中（10）	高（15）
大於 5	中（10）	高（15）	高（15）

5、外部檔案（External Interface Files）

外部檔案起碼會被一個外部輸入、外部輸出或者外部查詢使用到。如同內部檔案，外部檔案其功能點也是由記錄個數對照資料項個數取得，並依低、中、高功能需求，分別賦予（5）（7）（10）三個等級的未調整功能點。

記錄個數的認定，如同內部檔案評價，有可能擁有數個資料群（Data Group）格式，這時記錄個數也隨著增加。計算資料項數時，應從外部檔案去計算使用者能辨認的欄位，重複出現的欄位只計算一次，名稱不一樣但資料項定義相同的，也只計算一次。若有數個查詢鍵（Key Fields），只有被交易使用到的才能納為資料項個數計算。

在交易內，會去參考常駐在程式內部的代碼表（Code Table），此類代碼表不被外部輸入所更新，其身分與功能可視為一個外部檔案。

四、考量應用系統特性

完成未調整功能點的計算後，須考量應用系統特性 GSC，決定功能點調整因子 VAF，再以 VAF 乘上未調整功能點，得到調整後功能點。一個機構其系統開發環境已趨穩定，則 GSC 可以設定成一個常數，十四項 GSC 構面與其說明如表 3-4。每一項 GSC 構面，再就其作業方式與程序複雜度分別賦予 0,1,2,3,4 或 5 的量度，舉資料通訊特性為例，說明如表 3-5。【吳琮璠 00】

功能點調整因子 VAF 的公式，IFPUG 建議如下：

VAF=(65+各應用系統特性的量度和)/100

表 3-4 應用系統特性構面說明

應用系統特性		概要說明
1	資料通訊	運作在資料傳輸上的設備與複雜度
2	分配式資料處理	資料或功能是否分配處理？
3	績效衡量	是否滿足使用者特殊的需求,如回應時間在數秒之內？
4	硬體平台負載度	是否將在一部作業擁擠的電腦上執行,因此程式編寫要特別考慮？
5	交易頻率	交易頻率是每日，每週，或者每月等？
6	線上輸入比例	線上資料輸入的比例
7	使用者效率	是否特別顧慮使用者的效率訴求？
8	線上更新	多少個內部檔案是採線上更新？
9	處理的複雜度	系統邏輯與運算是否具有延伸性？
10	多人使用性	是否多人可同時使用？
11	安裝親和度	是否有強調轉換與安裝的便利性？
12	操作親和度	是否考慮啓動、備援及復原程序？
13	多地區/多平台	是否會給好幾個電腦中心使用？
14	可轉移性	是否會要求特殊設計，以便於修改？

＊ 本研究整理

表 3-5 應用系統特性構面量度表（資料通訊）

量度值	應用系統作業方式說明
0	純批次作業或 PC 作業
1	批次作業加上遠端輸入或者遠端列印
2	批次作業加上遠端輸入與遠端列印
3	單一端點線上資料處理
4	多端點單一通信協定資料處理
5	多端點多通信協定資料處理

五、計算調整後功能點

得到未調整功能點與功能點調整因子 VAF 後，則調整後功能點

FP=UFP*VAF

，計算過程如表 3-6。

表 3-6 調整後功能點計算表

構面型態	構面複雜度			
	低	中	高	小計
外部輸入	n1×3＝	n2×4＝	n3×6＝	UFP1
外部輸出	n1×4＝	n2×5＝	n3×7＝	UFP2
外部查詢	n1×3＝	n2×4＝	n3×6＝	UFP3
內部檔案	n1×7＝	n2×10＝	n3×15＝	UFP4
外部檔案	n1×5＝	n2×7＝	n3×10＝	UFP5
FP=（UFP1+ UFP2+ UFP3+ UFP4+ UFP5）*VAF				

＊ 本研究整理

功能點分析方法可分別應用於現有應用系統增強、新開發專案及重整專案之功能點計算。現有應用系統增強功能點計算，其公式如下：

FP=AUFP*VAF

AUFP 表示新增作業的未調整功能點

；新開發專案功能點計算，其公式如下：

FP=（UFP+CFP）*VAF

UFP 表示開發作業的未調整功能點，CFP 表示現有作業的未調整轉換功能點

；重整專案功能點計算，其公式如下：

FP=（AUFP+CUFP- DUFP +CFP）* VAF

AUFP 表示新增作業的未調整功能點，CUFP 表示變更作業的未調整功能點，
DUFP 表示刪除作業的未調整功能點，CFP 表示現有作業的未調整轉換功能點。

第四節 功能點分析的優缺點

一、功能點分析的優點

1、功能點分析被使用的最大原因，是 LOC 估算法不科學化。同一功能，對不同的程式語言，其 LOC 計數是不一樣的。即使是相同語言，一個指令橫跨數行，

或者一行內有數條指令的情形也多所見。使用功能點分析，因只計算功能，就能避開這些缺點。另對資訊部門言，易推動使用高生產力程式語言並增加程式碼重複使用機會。

2、功能點分析有一國際性組織 IFPUG 負責標準訂定與推動事宜，較具客觀公正。

3、功能點分析可以儘早估出應用系統的規模，一旦輸出、入畫面，輸出報表及相關聯檔案決定，就能估算。而且可隨軟體開發週期之進行，愈來愈精確。反之，LOC 法只能等程式撰寫完成後，才知道其規模。

4、功能點分析計算可由使用者執行，只要使用者能分辨畫面、報表欄位及檔案內資料項定義，即有能力核對資訊部門的計算結果。

5、其他優點，如用以衡量資訊部門或工程師生產力、預估開發與維護的人力需求、監督委外工作進行的成果交付及因應資訊資源短缺，做專案組合選擇。

二、功能點分析的缺點

1、 交易的五個構面外部輸入、外部輸出、外部查詢、內部檔案及外部檔案，有時不易分辨，以外部輸入言，若同時引發新增、變更及刪除某個內部檔案，將視同是三個外部輸入，稍一不慎，功能點就少算了。另對內部檔案與外部檔案言，除非有系統文書協助，僅憑印象不容易歸納記錄的個數。

2、 使用功能點分析，須對該方法論有相當瞭解，資訊部門無法大量培訓估算分析人員，更遑論讓使用者做核算。

3、 各構面評價時之功能點賦予，是一個反覆試驗的較佳值，欠缺客觀公正；同樣的，功能點調整因子 VAF 也欠缺客觀公正。

4、 決定 VAF 的十四項應用系統特性，IFPUG 是否跟上開發環境腳步及時做版本更新。

5、 五個衡量構面外部輸入、外部輸出、外部查詢、內部檔案及外部檔案，都是可計數的。但最重要的，應用系統內部的複雜度，卻無法有效衡量。

6、 五個構面算出的功能點數，只能用作資訊資源估算，例如由前一個專案 1000 功能點耗費了 250 人月，對下一個提出的新專案，若其需求具有 1500 功能點，僅知應籌措 375 人月，其他管理資訊就不易得到。

三、功能點分析的進一步探討

功能點分析自 1994 年起研究已漸趨定型，部份學者建議對下列四項做進一步探討【Char 02】。

1、 十四項應用系統特性 GSC 是否要及時修正？

2、 功能點分析法可否用來衡量演算法則（Algorithm）？

3、 外部輸入、外部輸出或外部查詢其使用到的資料項數特別多時，其功能點是否要加倍計算？

4、 外部輸入、外部輸出、外部查詢、內部檔案及外部檔案五個構面的功能點是否都等值？

第四章 軟體規模估算的數學模式

第一節 軟體規模估算的範圍

本研究主要探討企業以大型主機運作，使用第三代語言，如 COBOL 等設計程式之環境，如何運用軟體規模估算技術，歸納出各類資訊需求的功能點，進而預估次一年度工作計畫所需的人力資源，提供管理階層決策的參考。不同於功能點分析方法的可分別應用於現有應用系統增強、新開發專案及重整專案之功能點計算，本研究重點僅侷限於現有應用系統增強之功能點計算。

在軟體生命週期中，相關工作包括了需求分析、系統分析、系統設計、程式設計與單元測試、系統測試、系統建置與上線等系統開發階段工作及系統維護工作，分別由系統分析師與程式設計師來執行。為使本研究較具客觀，系統維護工作部份暫不研究，僅研究系統開發各階段工作部份。

企業若資訊部門人員流動率低與系統開發已實施標準化，則系統開發各階段工作，佔全體開發工作的比例可視為接近一個常數。基於此假設，本研究擬以工程師執行程式設計與單元測試兩部分所消耗人日，來代表整個軟體生命週期活動的人日消耗，即由程式設計與單元測試規模估算，依其比例換算代替軟體規模估算。

第二節 規模估算構面探討

依據系統開發成本影響因子之規模屬性、產品屬性、資訊科技屬性、人員屬性、專案屬性、環境屬性及管理屬性【林信惠 02】，考慮企業發展整體資訊系統之特性，影響資訊資源分配與運用的構面可篩選為五大類，即程式功能需求，系統功能需求，資料庫科技，系統使用者及程式撰寫者，其構面如圖 4-1。

一、程式功能需求

由業務單位提出的業務電腦化需求說明表，以業務流程、處理程序、處理邏輯與規則、作業憑證與輸出表單等資料，可判定程式複雜度與預估出大概的規模。

二、系統功能需求

系統複雜度另需考量此電腦化需求是新的單元系統，或需串接既有的系統。其他如是否用於網際網路上、異動資訊量是否很大、是否特別顧慮使用效率、是否有與使用者對話的功能，並強調安裝的便利性及是否要滿足特殊的需求，如回應時間在數秒之內等。

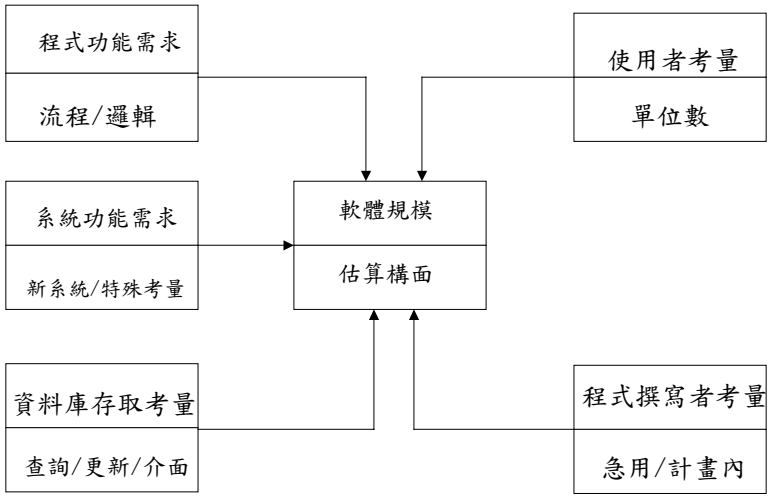

圖4-1、軟體規模估算影響構面

三、使用者考量

探討資訊系統橫跨的業務單位，只有一個單位或者數個單位；若是多單位，是在同一部門下或者分屬多部門；若是多部門，其工作時間為 8 小時或是需 24 小時不停頓工作，愈屬後者，其相對溝通與協調工作愈多。另外探討使用者組成，歸類為管理階層、管理師、工程師、基層作業員或者網際網路使用者。不同的使用者，依其使用慣性須有不一樣的人機介面設計。

四、程式撰寫者考量

探討程式是由誰來撰寫，就需求分析、系統分析、系統設計、程式設計與單元測試、系統測試、系統建置與上線等整個週期來探討，由同一個人來執行，或分由數人共同執行，其人日耗用考量是不一樣。若由數人共同執行，協同執行者是資淺工程師或是資深者，其考量也都不同。

五、資料庫科技

本研究的資料庫系統選擇，僅以系統開發人員的工作負荷來考量，不涉及 DBA 的工作。資料庫系統有三類，DB2--一種關連資料庫系統；DL/I--一種階層式資料庫系統；VSAM--一種索引檔案系統，選擇不同的資料庫系統，因 Data Integrity 考量不一樣，相對其工作負荷也不相等。另外亦得探討資料庫軟體平台與使用資訊科技的成熟度。

第三節、軟體規模估算模式

本研究的數學模式,採用作業研究的線性規劃模型,求取最小誤差平方和【李茂興 00】。

假設 X 為構面的功能點，是自變數，共有 n 個構面；W 為每一功能點的人日值，是一未知常數；Γ 為構面的 GSC 調整，是一常數；C 是因變數，為程式設計與單元測試實際人日，可得到公式（1）：

$$\sum_{i=1}^{n} X_i \cdot W \cdot \Gamma = C \quad \cdots(1)$$

公式（1）中，本研究先不考慮 GSC 的影響，令Γ=1，並假設以 n=6 展開如下：

$$X_1 W + X_2 W + X_3 W + ... + X_6 W = C \quad \cdots(2)$$

公式（2）中，X1,X2,X3..,X6 是自變數，W 雖是常數，但在未得到最佳解前，仍需以自變數看待，因此必須將非線性的公式（2）簡化為線性函數。假設每一筆資料的自變數 X1 都可以一個已知常數 A1 乘上另一自變數 π1 替代，則 X1W 可化為 A1π1W，X2W 可化為 A2π2W,X3W…X6W 類推，分別化為 A3π3W…A6π6W，再將 π1W 轉化為另一自變數 W1，則 A1π1W 轉化為 A1W1，A2π2W 轉化為 A2W2, A3π3W…A6π6W 類推，經過兩次轉換，公式（2）可以公式（3）取代：

$$A_1 W_1 + A_2 W_2 + A_3 W_3 + ... + A_6 W_6 = C \quad \cdots(3)$$

以 m 筆訓練資料帶入公式(3)迴歸後，可得到一組 $\overline{W_1}, \overline{W_2}, \overline{W_3} .... \overline{W_6}$ 使每一筆訓練資料滿足公式（4）：

$$A_1\overline{W_1} + A_2\overline{W_2} + A_3\overline{W_3} + ... + A_6\overline{W_6} = C + e \cdots (4)$$

同時使公式（5）得到最小的誤差平方和 S：

$$s = \sum_{i=1}^{m} e^2 \cdots (5)$$

檢視 $\overline{W_i}$ ，若最大與最小值差大於 5% （本研究設定的條件），則以最小的 $\overline{W_i}$ 去除各個 $A_1\overline{W_1}, A_2\overline{W_2}, A_3\overline{W_3}, ... A_6\overline{W_6}$ ，可得到一組新的 $A_1, A_2, A_3, ... A_6$ ，再帶入公式（3）繼續迴歸，直到 $\overline{W_i}$ 收斂在 5% 準確性之內。

第五章 個案研究：中國鋼鐵公司

第一節 中國鋼鐵公司與資訊系統

中鋼公司創立於民國六十年，是台灣唯一的一貫作業大煉鋼廠，生產鋼板、條鋼、線材、熱冷軋鋼捲及鋼胚，年粗鋼生產量超過一千萬噸。投資龐大、鋼種多、製造程序繁複，是一種資本密集、技術密集及知識密集的產業，年營業額達一千億台幣。

中鋼公司追求永續發展，持續維繫在國內鋼鐵產業的領導地位，同時以穩健的步調來發展新事業，繼續朝多角化、集團化及國際化經營發展，充分扮演「工業材料、腦力資源、優質生活供應者」的角色。公司依職務功能分設有財務、企劃、行政、生產、業務及技術等六個部門，部門下共設有三十多個一級廠、處。

整體資訊系統（Management Information System）以會計為核心，涵蓋營業管理系統（Marketing）、生產管理系統（Manufacturing）、購運儲管理系統（Material）、設備管理系統（Machine）、行政管理系統（Man）及財務管理系統

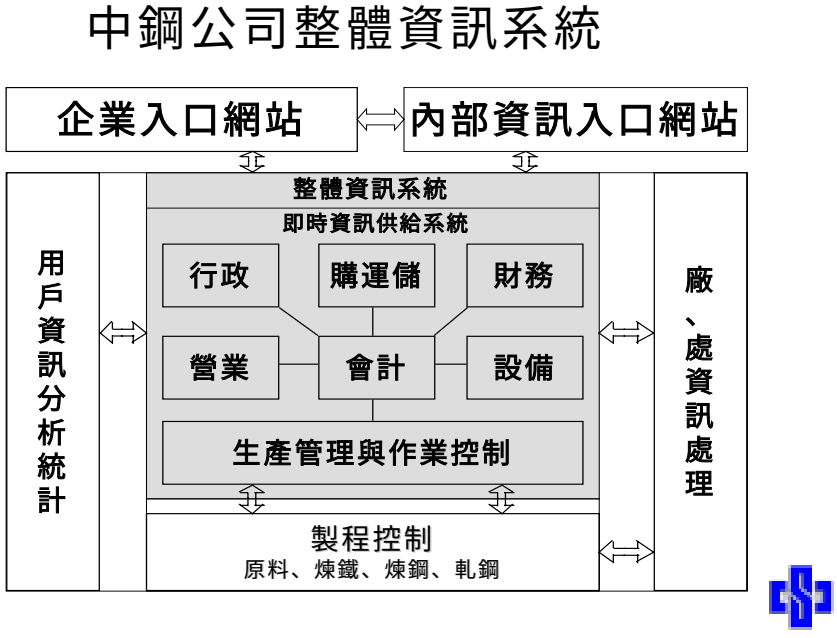

圖 5-1 整體資訊系統架構

（Money）等六 M 體系，如圖 5-1【中國鋼鐵 98】。整體資訊系統為 Thin-client 架構，以大型主機當 Server，其下接有大約 2800 部個人電腦與 100 餘部程序控

制電腦（Process-computer），提供 24 小時不間斷 Intranet 的資訊服務，目前有 121 個系統共 11,000 支程式。

資訊系統處為中鋼公司下的一級單位，共有 35 位制度工程師（Application Programmer），此外協力（Outsourcing）廠商亦提供約 40 位工程師長駐中鋼協助執行資訊系統之開發與維護工作。此 75 位工程師劃分成五個常設專案，分別負責六 M 體系之一部份，各專案之工程師會視歷練需要，不定期予以輪調。

整體資訊系統歷經近 30 年之自行開發，系統涵蓋範圍其深度、廣度已頗具規模。近年來隨著網際網路與供應鏈業務之盛行，日常工作之自動化、合理化開發維護已無法滿足各業務單位，轉而必須提供大幅度圖形化介面（Graphic User Interface）系統增強的服務。

第二節 中鋼資訊工作規劃與管理

資訊系統處於每年十一月，請各廠、處彙集其下一年所有電腦化需求，內容包括業務流程、處理程序、處理邏輯與規則、作業憑證與表單、業務量說明、資訊需求的時效，以及業務單位欲藉此解決的問題或困擾等。經系統負責人確認後，成立工作項目，並視其輕重緩急，納入下年度開發計畫。若未及編入下年度計畫者，則可於隔年六月年度計畫修正時再提出，或以緊急需要即時提出。

資訊系統處編排下年度開發計畫時，通常只考量 60%的可用人月，另保留 40%人力供系統維護使用，與學理上維護人力應保留 67%是有一段差距【Step 93】。依多年來經驗，資訊資源分配的瓶頸在工程師，因整體資訊系統的主要業務邏輯使用 COBOL 程式撰寫，市場上不易找到這類工程師。在工程師短缺的情況下，排入年度開發計畫之工作項目，往往會有高達 30%無法於承諾期內完成，部份工作項目甚至拖延一年以上才能結束。

中鋼的成本制度，資訊系統處之所有系統開發與維護費用，由該處自行編列費用預算吸收，並不分攤至各業務單位；加上資訊人力資源短缺，造成業務單位提電腦化需求時不夠慎重，常有某些電腦化需求之提出，只為佔有資訊人力資源配額。由於資訊系統處缺乏有效的資訊需求估算尺標，使得專案負責人在安排年度開發計畫時，不易掌握重點，只能以專家判斷法或類比法，憑過去的專案經驗

與業務單位協調訂出完成日期。再者，各專案因工作項目超出負荷，提出資訊資源滑落量（Backlog）時，這些不足的資訊資源，到底要委外處理呢？或是將資訊需求延後，對專案以上的主管言，也欠缺資訊以供決策。因此資訊資源如何有效的運用，以滿足大部份業務單位的需要，就值得去深入探討。

業務單位提出業務電腦化需求說明表時，其中之業務流程、處理程序、處理邏輯與規則、作業憑證與表單等資料與未來系統分析設計工作之負荷間，似乎可找出一些關連。舉例說，若業務流程、處理程序、處理邏輯與規則能以 ASIS-TOBE（現況-未來）對應方式提出，工程師將其轉換成系統流程就很容易。其次分析此項需求是以現有系統的資料庫就可因應，或者需另建新的資料庫。若需新建資料庫還需要 DBA（Data Base Administrator）協助建立與測試，人力或時間都需再增加。另外，電腦化需求牽涉到的業務單位數也是考慮因素，單位愈多不僅協調工作增加，系統測試時更會增加困難度。

作業憑證與表單雖可協助工程師，估算出程式數量與程式規模來，但仍得衡量此需求，是新建一個系統？或者在現有系統下增加一個子系統？或僅是修改現有系統即可因應。此三者間，其工作量估算是不一樣的。最後，不得不衡量制度工程師的資訊學養，其技術、經驗、過去的工作記錄、專案內有無相同業務背景的備援工程師等，都會影響整個工作時程。

第三節 個案研究與初步發現

一、個案研究設計

本研究主要探討企業內如何運用軟體規模估算技術，歸納出各類資訊需求的功能點，進而預估次一年度工作計畫所需的人力資源，提供管理階層決策的參考。因此在研究資料的選擇上，是以中鋼公司資訊系統處年度計畫的部份工作項目為研究對象，分析新開發的線上（OLTP）程式中之人力資源耗用。

在資料來源方面，本研究取自資訊系統處自民國九十一年十月至九十二年三月共計六個月間的工作項目。再整理出完成這一項工作，是寫了幾個新程式、每個程式動用了幾個資料庫 table、擁有什麼功能、誰使用這些程式、程式撰寫者是資淺制度工程師或是資深者。

在資訊系統處的程式管理系統中，記錄有每一程式是指定誰撰寫，使用什麼語言開發工具，程式的輸入、輸出及更新檔案為何，程式共寫了幾條指令。另外可用工具自動產出程式的架構圖，如圖 5-2，中鋼的 COBOL 程式為閱讀與偵錯方便，制定有撰寫標準，規定其架構由一至五層組成，每一層分由數個副程式（Sub-routine）組成，其中一至四層的副程式間須有臣屬關係，而第五層為共有副程式，包括公用模組與子程式，一至四之任一層的副程式可以直接呼叫第五層，因此經由架構圖的五個層次架構分析，可以評估出程式的複雜度【中國鋼鐵 90】。

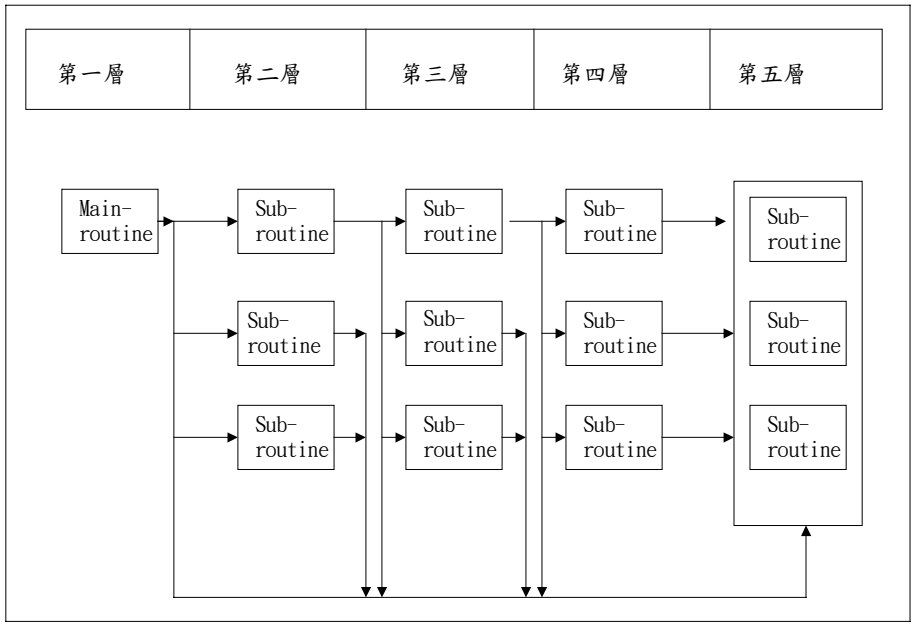

圖5-2副程式架構 ＊本研究整理

由程式管理系統與架構圖整理出的資料，可當作軟體規模估算的部份構面。

二、研究資料整理

參考功能點分析法的五個衡量構面外部輸入、外部輸出、外部查詢、內部檔案及外部檔案，本研究整理出線上作業（OLTP）其程式設計階段的衡量構面，選定下列六個影響程度較大者。

1、程式間傳輸介面個數－OLTP 程式的特色在於程式間可以互相呼叫，也可以呼叫程式本身，呼叫間以傳輸介面傳遞訊息。承襲功能點分析估算方法的外部檔案與內部檔案衡量觀念。計算每一程式與別的程式交換資料時，所使用到的記

錄數，依過去的經驗知，愈多表示愈複雜。

2、相關連系統個數－程式視其業務流程，分別納入營業管理系統、生產管理系統、購運儲管理系統、設備管理系統、行政管理系統或及財務管理系統等121系統中，若一個程式使用到的資訊牽涉愈多系統，就表示須多用一些時間，去瞭解那些領域的資料項，這是影響複雜度的重要構面。

3、更新檔案個數－程式對檔案做新增、更新及刪除處理時，除須作相關聯欄位(Data Field)的檢查外，亦須做更新前後資料之保留，以備回復(Recovery)之需，雖然資料庫軟體已具備回復功能，但設計良好的程式，仍須做此考量。此構面承襲功能點分析估算方法的內部檔案衡量觀念。

4、使用的業務單位個數－OLTP程式的好處，在於可供多人同時使用，為使各個使用者間彼此不互相影響，須有鎖住（Locking）與回復（Recovery）的機制。業務單位是指公司的二級單位，是業務流程的主辦組織。單位個數的多寡，與交易的複雜度有關，這一個構面承襲自功能點分析估算方法的外部輸入、外部輸出及外部查詢衡量觀念。

5、連結公用模組個數－常用的代碼轉換（Code Table）與資料檢查等，中鋼資訊系統處已制定150支公用模組程式，對某一程式言，若呼叫愈多的公用模組，表示需有更多的資料轉換動作發生，此亦影響程式的複雜度。在功能點分析估算方法中，代碼轉換視為外部檔案構面。

6、呼叫子程式個數（Subprogram）－子程式與公用模組一樣，是程式呼叫進來的一段副程式，不同的是，子程式型的副程式，會包括檔案處理指令。同理，若呼叫愈多的子程式，亦影響程式的複雜度，在功能點分析法中並未探討，本研究認為其相當於外部檔案的功用，是一個值得研究的構面。

7、各構面是以個數計算功能點，程式依六大構面各別計算一次功能點，但某一構面個數為0時，則該某構面無功能點。

本研究收集民國九十一年十月至九十二年三月六個月間的部分新完成程式共一百一十三支，先請制度工程師填報，其在程式設計與單元測試兩階段所使用

的人日。再依程式間傳輸介面、相關連系統、更新檔案、使用的業務單位、連結公用模組、呼叫子程式及其他研究需要，整理出本研究的數據。

三、研究構面比較

本研究選定的六個構面與功能點分析法的五個衡量構面做一比較如表 5-1：

表 5-1 軟體估算構面比較

本研究構面	相當於功能點分析構面或系統特性
相關連系統	處理的複雜度、使用者效率、績效衡量
程式間傳輸介面	外部檔案、內部檔案
更新檔案	內部檔案、外部檔案
使用的業務單位	外部輸入、外部輸出、外部查詢
連結公用模組	外部檔案
呼叫子程式	外部檔案

功能點分析法執行時機，是完成系統分析階段工作。五個構面中，外部輸入、外部輸出、外部查詢是以逐一交易各別去計算，而外部檔案與內部檔案是一個系統下的每一個檔案僅計算一次。本研究執行時機是系統設計完成後，程式規範已確定，倘若每一程式使用到的外部檔案與內部檔案全納入計算，會有十個程式使用，就計算十次檔案功能點之不合理現象。

本研究外部檔案與內部檔案都可以直接去更新，不似功能點分析法的嚴謹，在功能點分析法中，是不允許對外部檔案做更新，僅能產生一個外部輸出當介面，再由其他應用系統接續去做內部檔案更新。因此本研究選擇以更新檔案當構面，來代替功能點分析法的外部檔案與內部檔案的一部份功能。程式間傳輸介面之功能，主要是將前一處理的檔案查詢資料傳至下一處理，其執行的正是外部檔案與內部檔案的另一部份功能。

中鋼公司實務上屬於報表的外部輸出，幾乎以畫面的外部查詢取代，而且資料輸入畫面同時設計兼具查詢功能。另外，為了效率考慮，大部份的業務單位會以共用畫面，各別使用欄位的整合方式，要求資訊部門設計程式。因此本研究以使用的業務單位，同時取代功能點分析法的外部輸入、外部輸出、外部查詢三個

構面。

相關連系統當作構面，是將功能點分析法的三項應用系統特性，處理的複雜度、使用者效率及績效衡量列入考量。連結公用模組與呼叫子程式兩個構面，執行類似常駐在程式內部的代碼表（Code Table）工作，其身分與功能可視為一個外部檔案。

四、個案研究初步結果

首先由六十三筆訓練資料中分析（詳附錄一），程式 PTOU53 之生產力為 515 指令/人日，而程式 IHOUAB 之生產力只有 23 指令/人日，很顯然 LOC 估算法並不適用於軟體規模的估算。

先以程式間傳輸介面個數、相關連系統個數、更新檔案個數、使用的業務單位個數、連結公用模組個數及呼叫子程式個數當作功能點，直接帶入公式（3，詳 27 頁）Ai 中執行迴歸分析，得到表 5-2 結果，經探討訓練資料，若以此結果來預估新軟體專案的規模，會有程式 IHOUAB 使用一個傳輸介面耗用 0.7375 人日，而 IKOR44 使用八個傳輸介面耗用 5.9 人日之不合理現象。

表 5-2 構面未分類前功能點的人日

Ai=1	傳輸介面	相關連系統	業務單位	子程式	公用模組	更新檔案
Wi	0.7375	1.0813	0.3953	-0.4192	0.1997	0.3392

第四節 個案研究結果與文獻探討比較

一、功能點分析法的分類

由表 3-2（詳 18 頁）得知，外部輸入功能點由檔案個數對照資料項個數取得，並依低、中、高程度分別賦予適當的功能點，各別給予（3）（4）（6）三個等級的未調整功能點。

本研究先將六十三筆訓練資料，依各個構面的數量分類成低、中、高功能需求，再以此參考功能點分析法，分別賦予 Ai 為（3）（4）（6）如表 5-3。以相關連系統構面為例，個數等於 1，為低功能需求；個數等於 2，為中功能需求；個

數介於 3 至 6 間，為高功能需求，但個數為 0 時，則 Ai 訂為（0）。

表 5-3 構面數量低中高需求分類

項次	構面個數	低功能需求(3)	中功能需求(4)	高功能需求(6)
一	程式間傳輸介面個數	1	2,3	4-8
二	相關連系統個數	1	2	3-6
三	使用的業務單位個數	1	2,3	4-7
四	呼叫子程式個數	1	2	3-5
五	連結公用模組個數	1-4	5-8	9-17
六	更新檔案個數	1	2	3-5

經轉換後的功能點帶入公式（3）Ai 中執行迴歸分析，得到表 5-4 結果，每個構面的一個功能點的人日值是不相同。

表 5-4 構面分類後功能點的人日

Ai=1	傳輸介面	相關連系統	業務單位	子程式	公用模組	更新檔案
Wi	0.4054	0.5734	0.23	0.0298	0.2615	0.1462

二、構面敏感度分析

參考功能點分析法的其他構面功能點賦予，如外部輸出的給予(4)(5)(7)；內部檔案的給予（7)(10)(15）及外部檔案的給予（5)(7)(10)。經調整 Ai 後，以（3)(4)(5)；(3)(4)(6)；(7)(10)(15）三組 Ai 搭配，採用五十筆訓練資料，分別以公式（3）執行迴歸分析後，本研究發現子程式構面一個功能點獲得的人日值，其值遠低於其他構面，詳表 5-5。

因子程式與公用模組，其功能都是提供一組可執行的副程式，除減輕工程師的工作負擔外，並使程式具結構化而更容易維護。本研究決定，將子程式納為公用模組之一部分，以 Ai 為（3)(4)(6）帶入公式（3）執行迴歸分析，得到表 5-6 結果。

表 5-5 構面敏感度分析

一個功能點的 Wi	傳輸介面	相關連系統	業務單位	***子程式***	公用模組	更新檔案
全部功能點 3,4,6 分類	0.2969	0.5298	0.1904	***-0.0074***	0.3860	0.1627
將相關連系統設定由 3,4,6 調為 7,10,15	0.3128	0.5401	0.2054	***-0.0109***	0.1410	0.1660
將公用模組設定由 3,4,6 調為 7,10,15	0.4054	0.5734	0.2030	***0.0298***	0.2615	0.1462
全部功能點 3,4,5 分類	0.2667	0.6260	0.1705	***0.0148***	0.4187	0.1418

表 5-6 子程式與公用模組合併後功能點的人日

Ai=1	傳輸介面	相關連系統	業務單位	公用模組（含子程式）	更新檔案
Wi	0.3863	0.5794	0.1584	0.3235	0.1265

三、功能點探討

功能點分析法以五個衡量構面外部輸入、外部輸出、外部查詢、內部檔案及外部檔案，對照記錄數或檔案數及資料項數之低、中、高功能需求，分別給予(3)(4)(6);(4)(5)(7);(7)(10)(15) 及 (5)(7)(10) 四組適當的功能點。倘若一個專案共有 1000 個功能點，最後完成時共耗用 250 個人月，則一個功能點代表 0.25 個人月。若有新的專案提出時，一個外部輸入交易屬於低功能需求 (3) 是三個功能點，代表此交易將使用 0.75 個人月，一個外部檔案屬於高功能需求 (10) 是十個功能點，代表這個檔案將使用 2.5 個人月。

以表 5-6 之結果探討，每一構面一個功能點的人日值並不相當，無法滿足 Wi 最大與最小值差不逾 5% 之收斂條件。參考功能點分析法的每一構面功能點可有不同賦予之架構，本研究決定以表 5-6 中最小的 Wi 為基數，以 Ai*Wi 除以這個基數，算出所有新的 Ai，繼續帶入公式 (3) 執行迴歸分析，再得到一組新的 Wi，一再重複此步驟，詳如附錄二，最後 Wi 收斂如表 5-7 的結果。

表 5-7 構面的功能點過程

	傳輸介面			相關連系統			業務單位			公用模組（含子程式）			更新檔案		
低中高功能點	9	12	18	14	18	27	4	5	7	8	10	15	3	4	6
Wi	0.1158			0.1266			0.1640			0.1064			0.1633		
最佳低中高功能點	8	11	16	11	15	22	10	12	16	8	11	16	3	4	6
最佳 Wi	0.1205			0.1202			0.1180			0.1176			0.1195		

四、功能點敏感度分析

表 5-7 是構面功能點 Ai 以（3)(4)(6）當起始值，執行迴歸分析模式的結果。現在將功能點 Ai 改以（7)(10)(15）當起始值，以相同程序再執行一次迴歸分析，其比較如表 5-8。

由表 5-8 知，以起始值 3,4,6 或者 7,10,15 執行迴歸分析，得到幾近相同的各構面低中高需求功能點與人日值。

表 5-8 構面功能點敏感度分析

最佳值 / 起始值	傳輸介面			相關連系統			業務單位			公用模組（含子程式）			更新檔案		
3,4,6	8	11	16	11	15	22	10	12	16	8	11	16	3	4	6
Wi	0.1205			0.1202			0.1180			0.1176			0.1195		
7,10,15	9	12	13	12	17	26	6	9	12	7	10	15	3	4	6
Wi	0.1239			0.1227			0.1193			0.1213			0.1217		

五、構面不同分類敏感度分析

本研究另以構面五分類取代表 5-3 的三分類，並分別賦予 Ai 起始值為（4)(5)(7)(10)(15)，以相同程序再執行一次迴歸分析，不同的構面分類間得到比較如表 5-9。

表 5-9 構面不同分類敏感度分析

起始值＼最佳值	傳輸介面			相關連系統			業務單位			公用模組（含子程式）			更新檔案		
7,10,15	9	12	13	12	17	26	6	9	12	7	10	15	3	4	6
Wi	0.1239			0.1227			0.1193			0.1213			0.1217		
4,5,7,10,15	18	22	31	44	66	25	30	43	61	90	4	5	7	10	15	6	7	10	15	21	11	13	18	27	39
Wi	0.0669			0.0670			0.0643			0.0693			0.0669		

六、估算模式檢測

另選定五十筆資料，以表 5-7 之最佳功能點模式去檢測，得到結果如附錄三，並說明如下：

1、誤差最大為 1.36 人日，平均為 0.577 人日，其分佈如表 5-10，52%檢測資料其誤差落在 0-0.5 人日之間。

表 5-10 誤差人日分佈

誤差人日	0-0.5	0.5-1	1-1.4	合計
筆數	26	12	12	50
%	52	24	24	100

2、誤差率平均為 10.756%，其分佈如表 5-11，58%檢測資料其人日誤差落率在 0-10%之間。

表 5-11 人日誤差率分佈

誤差人日%	0-5	5-10	10-15	15-20	21-25
筆數	16	13	4	6	11
%	32	26	8	12	22

第五節 個案研究的其他發現

一、構面低、中、高功能需求選擇

功能點分析法的五個衡量構面外部輸入、外部輸出、外部查詢、內部檔案及外部檔案，記錄數或檔案數對照資料項數分類成低、中、高功能需求，它雖採用九等分法，但最後仍給予（3)(4)(6)；(4)(5)(7)；(7)(10)(15）或（5)(7)(10）的三組功能點。本研究將訓練資料依構面數量分為低、中、高三個功能需求後，再以迴歸分析求出每個構面的三個最佳功能點，當然這些功能點，可能隨訓練資料增加，或者構面數量分類條件改變而有所調整。

二、軟體規模估算運用

使用功能點分析法，可以參考過去相當規模專案的功能點，估算出新專案的需求人月，以安排各項資訊資源；同時也可運用於專案執行過程中的檢討，以未完成的部份還有多少功能點，估算出新的需求人月，作資訊資源的安排。倘若無法取得新資源，則可調整工作時程，或者縮小專案規模因應。本研究的估算模式，亦同樣以過去某段期間之程式生產力，估算出每一支新程式的需用人日，再由程式彙集估算出專案的需求人月，達到相同的專案管理目的。

三、工程師績效評價

以往績效考評時，是以工程師完成幾支程式，工作有無落後做為依據，較不客觀。利用軟體規模估算模式，資訊主管可以模式得到的功能點數為基準，再參考其他評價項目，做工程師績效考評。

第六章 結論與建議

第一節 研究貢獻

使用功能點分析法的最主要缺點，須對該方法論有相當瞭解，一般資訊部門不易引進。本研究利用迴歸分析求出的軟體規模估算模式，提供資訊部門一套比較簡易的程序，可藉由選擇所能掌控的構面，去做各項資訊資源規劃與管理。相對功能點分析法的繁複程序，實施的可行性增高了。軟體規模估算模式施行步驟，如圖 6-1：

一、決定軟體估算範圍（Scope）

在軟體生命週期中，相關工作包括了需求分析、系統分析、系統設計、程式設計與單元測試、系統測試、系統建置與上線等系統開發階段工作及系統維護工作。實施本研究模式前，須先決定軟體估算範圍，是選用軟體生命週期的全部，或者只挑選某些階段。挑選的原則，在於資訊部門可否完整收集到過去各階段的足夠人日（月）資料。

二、決定軟體規模估算構面

就資訊部門開發應用軟體開發程序與程式語言工具中，影響開發人日的構面逐一列出，建議挑選 5 至 7 項最重要構面納入模式中。建議以功能點分析法的外部輸入、外部輸出、外部查詢、內部檔案及外部檔案構面做為基礎，再加上部份應用系統特性構面做考量。

三、收集過去的資料

決定估算構面後，接著進行訓練資料整理，本研究的數學模式，是採用作業研究的線性規劃模型，求取最小誤差平方和。理論言，若以 N 個構面做估算，收集 N 組資料，即可得到一組最佳解。為了避免人為誤差影響過鉅，建議以相當於 5 至 10 倍的資料，執行迴歸分析。

四、估算構面分類與評價

參考功能點分析法對估算構面的評價方式，以低、中、高功能需求先分別給予諸如（3）(4)(6）的評價。低、中、高功能需求分類，可使用三等份法或者其他分類法。

五、執行迴歸分析

每一筆資料依構面數量分類給予評價後，套入 A1W1+A2W2+…+AnWn=C 線性規劃模式中，執行迴歸分析，求得一組功能點 Ai 與人日值 Wi。

六、調整構面評價

以求得的 Wi，檢測其是否收斂，條件可假設為最大的 Wi 與最小的 Wi 值差小於 5% 。若收斂，則得到最佳的功能點與人日值估算；若未收斂，則以每一組 Ai*Wi 除以最小的 Wi，算出新的 Ai，回到步驟五。

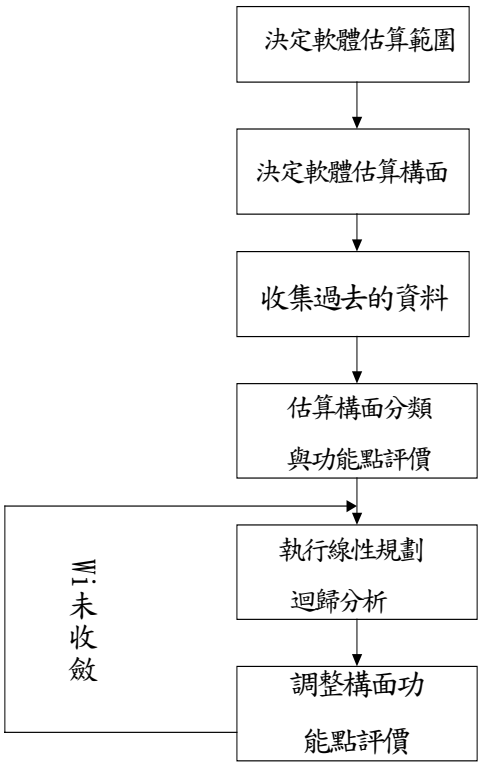

圖 6-1 軟體規模估算模式施行步驟

第二節 研究限制

本研究基礎於功能點分析法的理論架構,並考量個案公司－中鋼公司的開發環境特性，選定程式間傳輸介面、相關連系統、更新檔案、使用的業務單位及連結公用模組（合併呼叫子程式）等影響程度較大的構面，進行迴歸分析，求取每一構面低、中、高需求的功能點。唯受研究資源與時間的限制，下列子題並未進一步去探討：

一、構面低、中、高功能需求分類

功能點分析法以記錄數或檔案數對照資料項數之低、中、高功能需求劃分成九等分後，給予三個不同的功能點，本研究則分析各構面之數量分佈，給予三個不同的功能點。雖同樣是三個功能點決策，但功能點分析法考量資料項數之多寡，較能反映交易之複雜度。

二、構面的選擇

傳輸介面、相關連系統、更新檔案、使用的業務單位及連結公用模組五個構面以外，建議程式撰寫者構面、資料庫科技構面或者其他未考量的構面亦須納入。以程式撰寫者構面，生產力高低可差至十六倍以上【Rich 85】,在個案公司－中鋼公司，訂有程式撰寫標準與制度工程師流動率不到 2%的情形下，是可以忽略不計。至於資料庫科技構面,也因中鋼公司的資料庫使用只侷限於 DB2、DL/I 及 VSAM 三種且使用多年，而忽略不計。

三、功能點選擇

功能點分析法依不同的構面給予(3)(4)(6);(4)(5)(7);(7)(10)(15)或(5)(7)(10),這四組功能點中，低、中、高功能需求之比例是有些微不同。各個構面之功能點決定，應是經過一番嚴謹的檢定。而本研究是在(3)(4)(6)及(7)(10)(15)兩組起始值設定下，得到表 5-8（詳 38 頁）之結果。

四、計算調整後功能點

功能點分析法以未調整功能點，經過十四項應用系統特性如資料通訊、分配

式資料處理、分配式資料處理等估算，得到功能點調整因子 VAF 後，計算出調整後功能點。而本研究只納入使用的業務單位數、相關連系統個數為構面考量，似有所不足。

五、訓練資料收集之限制

本研究功能點換算單位，程式撰寫與單元測試之人日耗費，因其範圍侷限在 1 至 10 人日之間，稍一不慎，1 人日的差異就會產生 10%的誤差。若能改以人時來衡量，誤差即可降至 1.25%，惟實務上，工程師不易做到工作人時之收集。

第三節 後續研究之建議

本研究將功能點分析法的未調整功能點、應用系統特性 GSC 考量、計算調整因子 VAF 及計算調整後功能點之架構，重做整理，選定程式撰寫階段的影響構面，去架構軟體規模估算模式。此模式運用在個案公司－中鋼公司，已可證實，在此，我們提出一些後續的研究方向：

一、調查影響軟體規模的構面，除了傳輸介面、相關連系統、更新檔案、使用的業務單位、連結公用模組、程式撰寫者構面、資料庫科技構面外，還有那些構面尚須納入。

二、構面低、中、高功能需求分類，須考量什麼？從交易複雜角度去分類，或者從檔案複雜角度去分類？考量資料項數是否較為合適？

三、應用系統特性 GSC 當作影響軟體規模的構面？還是搭配在主要影響構面，如傳輸介面、相關連系統、更新檔案中，做為構面低、中、高功能需求分類之考量因素？

第四節 結語

軟體規模估算工作，其範圍大至軟體專案管理活動，小則僅至一支程式的設計，是一件困難度頗高的任務。在國外，有國際性學術組織 IFPUG 推動功能點分析法，而在國內，軟體公司都以過去的承包經驗，來做未來的專案管理，雖不理想，但也有前例可因循。至於一般公司內的資訊管理活動，其軟體規模估算就

乏善可陳。

本研究，以中鋼公司實際的管理數據，推動程式設計與單元測試兩階段的人日耗用分析，建立起一套軟體規模估算模式。不僅可用於公司內部資訊資源的分配，亦可供後續學術研究的參考。

為使管理能更上軌道，未來須不斷回饋更多的數據，使估算模式能契合開發環境的變化；亦希望能有其他更進一步的學術研究投入，使軟體工程的規模估算領域更實務化。

參 考 文 獻

【中文部份】

1、【中國鋼鐵 98】中國鋼鐵股份有限公司，資訊系統處白皮書，1998。

2、【中國鋼鐵 90】中國鋼鐵股份有限公司，資訊作業標準，1990。

3、【林信惠 02】林信惠等，軟體專案管理，台北：智勝文化，2002。

4、【吳琮璠 00】吳琮璠等，資訊管理-理論與實務，台北：智勝文化，2000。

5、【李茂興 00】李茂興譯，Michael E. Hanna 著，作業研究，台北：揚智文化，2000。

6、【鄒正平 99】鄒正平譯，Steve McConnell 著，微軟開發快速秘笈，台北：華彩，1999。

【英文部份】

1、【Albe 92】Albert L. Lederer and Jayesh Prasad，”Nine Management Guidelines for Better Cost Estimating”，Communication of the ACM Vol. 35，No. 2，1992，51-59。

2、【Char 02】Charley Tichenor，”Recommendations for Further Function Points Research”，www.SoftwareMetrics.Com，2002。

3、【Davi 02】David Longstreet，”Function Points Analysis Training Course”，www.SoftwareMetrics.Com，2002。

4、【Davi 02】David Longstreet，”Fundamentals of Function Point Analysis”，www.SoftwareMetrics.Com，2002。

5、【Ians 01】Ian Sommerville，Software Engineering，Harlow：Pearson Education，2001。

6、【Norm 99】NP Archer and F. Ghasemzadeh，”An Integrated Framework for Project Portfolio Selection”，International Journal of Project Management Vol. 17，No. 4， 1999，207-216。

7、【Rayb 02】Ray Boehm，” Frequently Asked Questions”，www.ifpug.org，2002。

8、【Rich 85】Richard E. Fairley，Software Engineering Concepts，New York：McGraw-hill，1985。

9、【Robe 99】Robert T. Hughes，”Expert Judgment as an Estimating Method”，Information and Software Technology 1996，67-75。

10、【Roge 01】Roger S. Pressman，” Software Engineering” ，New York：McGraw-hill，2001。
11、【Serg 99】Serge Oligny, Alain Abran and Denis ST-Pierre，” Improving Software Functional Size Measurement” ，www.lrgl.uqam.ca，1999。
12、【Step 93】Stephen R. Schach，” Software Engineering” ，Illinois：IRWIN，1993。

附　錄

附錄一

九十一年十月至九十二年三月訓練數據

程式名稱	DB2/DLI	介面數	系統數	使用單位數	子程式	GE數	檔案更新	人日	指令數	自行撰寫	資深
PTOU53	DB2	2	1	3	0	3	2	8	4124	N	Y
TQOUSV	DLI	2	4	4	0	3	1	6	912	N	Y
PTOU52	DB2	6	1	1	0	8	1	8	2012	N	Y
SPOUDC	DB2	3	1	3	2	9	5	7	1176	N	
SPOUDD	DB2	1	1	3	1	5	4	6	691	N	
SPOUDF	DB2	1	2	2	0	2	2	7	1097	N	
KVOU41	DB2	2	2	1	0	3	2	7	3282	N	
SPOUPC	DB2	0	1	1	0	6	0	6	534	N	
PTOR5B	DB2	1	1	1	0	5	0	6	509	N	
KVOI42	DB2	2	1	1	0	4	1	4	2343	N	
KVOI44	DB2	4	1	1	0	3	1	4	937	N	
PTOR5A	DB2	3	1	1	0	3	0	6	751	N	
PTOU51	DB2	2	1	1	1	4	1	7	1591	N	
SPOICK		6	1	1	0	9	0	6	702	N	
SPOUD1	DB2	1	4	1	1	6	0	7	736	N	
SPOUD2	DB2	1	4	1	1	6	0	7	816	N	
SPOUD6	DB2	0	3	1	1	7	0	5	512	N	
SPOUD7	DB2	2	3	1	0	1	0	5	943	N	
SPOUDG	DB2	0	2	3	0	5	1	6	522	N	
SPOUDH	DB2	1	1	2	0	7	1	6	576	N	
SPOUDS	DB2	0	3	3	0	7	0	6	521	N	
MPOR4P	DLI	2	1	1	0	2	1	6	419	Y	Y
MPOR8B	DLI	3	1	2	0	4	1	3	368	Y	Y
ICOIHB	DB2	1	1	1	0	4	1	5	193	Y	Y
ICORHC	DB2	1	1	1	0	3	1	5	172	Y	Y
ICOUHA	DB2	2	1	1	1	9	1	10	1022	Y	Y
IHOUAB	DB2	1	1	1	1	4	1	8	177	Y	Y
SCOR55		1	1	6	1	1	0	3	127	Y	Y
SCOR56	DB2	1	2	6	0	2	0	3	308	Y	Y
SCOU52	DB2	1	1	7	2	4	0	5	805	Y	Y
SCOU53	DB2	3	6	6	5	8	2	7	1352	Y	Y
SCOU54	DB2	3	3	1	3	5	2	5	738	Y	Y

程式名稱	DB2/DLI	介面數	系統數	使用單位數	子程式	GE 數	檔案更新	人日	指令數	自行撰寫	資深
SSOU92		1	1	1	1	4	1	3	604	Y	Y
SSOUB0	DLI	1	2	1	1	5	0	3	567	Y	Y
MAOI21	DLI	2	1	1	0	3	1	5	464	Y	Y
MSOU20	DLI	3	1	1	0	3	2	6	1226	Y	Y
SPOU3K		1	5	4	0	3	1	7	296	Y	Y
TOOU12	DB2	2	1	1	0	6	1	1	351	Y	Y
TOOU81	DB2	1	3	1	0	8	1	5	1509	Y	Y
TOOU82	DB2	1	1	1	1	7	1	5	959	Y	Y
MLOU33	DB2	3	2	2	0	7	1	5	753	Y	
KWOIX2	DB2	2	1	1	0	4	0	7	574	Y	
SPOUDK	DB2	1	1	3	0	7	2	7	909	Y	
SPOUDL	DB2	3	1	3	2	7	2	7	948	Y	
SPOUDM	DB2	1	1	3	1	5	2	5	531	Y	
SPOUDP	DB2	3	1	3	2	8	2	7	682	Y	
SPOUDR	DB2	2	1	3	1	8	3	7	892	Y	
SPOUDT	DB2	1	1	2	0	2	11	5	285	Y	
SPOUPL	DB2	1	2	1	0	2	0	3	89	Y	
EBOI52	DB2	4	1	2	0	5	0	6	1020	Y	
EBOU53	DB2	2	1	1	0	2	3	3	1021	Y	
EBOU54	DB2	3	1	1	0	4	2	4	835	Y	
PHOUC1	DB2	2	2	1	0	2	1	10	851	Y	
PHORDD		4	1	1	0	6	1	10	218	Y	
PNOU93	DB2	3	1	1	0	2	1	10	674	Y	
IKOR44	DB2	8	2	1	0	4	0	10	729	Y	
KVOU43	DB2	1	2	1	0	5	1	7	2305	Y	
SBOIQ4	DLI	2	1	1	0	4	0	4	184	Y	
KAOU61	DB2	3	1	1	0	17	2	4	1209	N	Y
KAOU62	DB2	4	1	1	0	11	2	4	779	N	Y
KAOU63	DB2	5	1	1	0	10	2	4	708	N	Y
KAOU64	DB2	5	1	1	0	12	1	4	650	N	Y
JCOUBS	DB2	2	1	1	1	3	1	5	489	N	

附錄二

迴歸分析過程

日期	程式間傳輸介面			相關連系統			業務單位			連結公用模組			更新檔案		
構面影響度	低	中	高	低	中	高	低	中	高	低	中	高	低	中	高
功能點同為3,4,6設定	3	4	6	3	4	6	3	4	6	3	4	6	3	4	6
每一構面之人日/功能點	0.3863			0.5794			0.1584			0.3235			0.1265		
將人日/功能點比例帶入	9	12	18	14	18	27	4	5	7	8	10	15	3	4	6
人日/功能點	0.1158			0.1266			0.1640			0.1064			0.1633		
920424	7	9	13	10	14	21	4	5	7	5	7	10	3	4	6
人日/功能點	0.1634			0.1416			0.2048			0.1939			0.1225		
920425-V1	9	12	17	11	16	24	7	8	11	8	11	16	3	4	6
人日/功能點	0.1169			0.1093			0.1803			0.1125			0.1176		
920425-V2	9	12	17	10	15	22	11	12	17	8	11	15	3	4	6
人日/功能點	0.1117			0.1168			0.1018			0.1379			0.1223		
920428	8	11	16	10	14	21	9	10	14	9	12	17	3	4	6
人日/功能點	0.1183			0.1287			0.1295			0.1140			0.1109		
920429-V1	8.5	11.7	17.1	11.6	16.2	24.4	10.5	11.7	16.3	9.3	12.3	17.5	3	4	6
人日/功能點	0.1103			0.1099			0.1136			0.1105			0.1190		
920429-V2	7.9	10.8	15.8	10.7	15.0	22.5	10.0	11.2	15.6	8.6	11.4	16.3	3	4	6
人日/功能點	0.1205			0.1202			0.1180			0.1176			0.1195		

附錄三

九十一年十月至九十二年三月檢測數據

程式名稱	介面數	系統數	使用單位數	GE數	檔案更新	人日	檢定值	誤差值	誤差%	指令數
KCOUA1	8	22	10	11	3	7	6.48	0.52	8.025	2198
KCOUB1	8	15	10	11	3	5	5.64	0.64	11.35	1353
KCOUC1	8	11	10	11	3	4	5.16	1.16	22.48	1437
KCOUD1	8	11	10	11	3	4	5.16	1.16	22.48	2007
KCOUE1	8	11	10	11	4	5	5.28	0.28	5.303	1994
KCOUF1	8	15	16	11	4	7	6.48	0.52	8.025	1241
KWOR8A	8	11	10	8	0	5	4.44	0.56	12.61	1008
KWOU87	11	15	10	11	3	7	6	1	16.67	1805
OCOUB1	8	11	10	8	4	5	4.92	0.08	1.626	819
OCOUB3	8	11	10	8	3	5	4.8	0.2	4.167	112
OCOUC1	11	11	10	8	4	6	5.28	0.72	13.64	819
OCOUC2	11	11	10	11	6	6	5.88	0.12	2.041	1165
OCOUC3	11	11	10	11	6	6	5.88	0.12	2.041	1592
OCOUC4	11	11	10	11	6	6	5.88	0.12	2.041	1309
OCOUC5	11	11	10	8	6	6	5.52	0.48	8.696	1209
OCOUC6	11	11	10	11	6	6	5.88	0.12	2.041	1406
OCOUC7	11	11	10	11	6	6	5.88	0.12	2.041	1356
OCOUC9	11	11	10	11	4	7	5.64	1.36	24.11	578
OCOUCB	11	11	10	8	6	5	5.52	0.52	9.42	416
OCOUE1	11	11	10	11	4	6	5.64	0.36	6.383	1006
OCOUE2	11	11	10	16	6	6	6.48	0.48	7.407	1334
OCOUE3	11	11	10	8	6	6	5.52	0.48	8.696	1306
OCOUE4	11	11	10	16	6	6	6.48	0.48	7.407	1649
OCOUE5	11	11	10	11	6	7	5.88	1.12	19.05	1614
OCOUE6	11	11	10	16	6	6	6.48	0.48	7.407	1489
OCOUE7	11	11	10	11	6	6	5.88	0.12	2.041	1661
OCOUE9	11	11	10	11	6	7	5.88	1.12	19.05	963
OCOUEB	11	15	10	8	6	6	6	0	0	512
OCOUEE	11	15	10	11	6	6	6.36	0.36	5.66	1003
PWOU71	8	15	16	16	4	7	7.08	0.08	1.13	2990
TMOI9S	0	11	10	11	0	3	3.84	0.84	21.88	393
TMOR9K	0	11	10	11	0	4	3.84	0.16	4.167	1817
TMOR9P	0	11	10	11	0	4	3.84	0.16	4.167	2799

程式名稱	介面數	系統數	使用單位數	GE數	檔案更新	人日	檢定值	誤差值	誤差%	指令數
TMOU9A	8	11	10	11	3	4	5.16	1.16	22.48	716
TMOU9B	8	11	16	11	6	7	6.24	0.76	12.18	1812
TMOU9C	8	11	16	11	3	6	5.88	0.12	2.041	1722
TMOU9D	8	11	10	11	3	6	5.16	0.84	16.28	2716
TMOU9E	8	11	10	11	3	5	5.16	0.16	3.101	287
TMOU9F	8	11	10	11	3	4	5.16	1.16	22.48	1022
TMOU9G	8	11	10	11	3	4	5.16	1.16	22.48	814
TMOU9H	8	11	10	11	3	4	5.16	1.16	22.48	836
TMOU9I	8	11	10	11	3	4	5.16	1.16	22.48	881
TMOU9J	8	11	10	11	3	4	5.16	1.16	22.48	1592
TMOU9L	8	11	10	11	3	6	5.16	0.84	16.28	2615
TMOU9M	8	11	10	11	3	5	5.16	0.16	3.101	2018
TMOU9N	8	11	10	11	3	5	5.16	0.16	3.101	2940
TMOU9O	8	11	10	11	3	4	5.16	1.16	22.48	498
TMOU9Q	8	11	10	11	3	6	5.16	0.84	16.28	839
TMOU9U	8	11	12	11	3	5	5.4	0.4	7.407	1993
TMOU9V	8	11	12	11	3	5	5.4	0.4	7.407	2238