國立交通大學

工業工程與管理學系

碩士論文

建構軟體發展流程之多變量管制系統

Construct the Multivariate Control System for Software Development Process

指導教授：唐麗英　博士

洪瑞雲　博士

研 究 生：陳啟揚

中華民國九十七年六月

建構軟體發展流程之多變量管制系統

Construct the Multivariate Control System
for Software Development Process

研 究 生：陳啟揚　　Student：Chen Chi Yang
指導教授：唐麗英 博士　　Advisor：Lee-Ing Tong
洪瑞雲博士　　Ruey-Yun Horng

國立交通大學

工業工程與管理學系

碩士論文

A Thesis
Submitted to Department of Industrial Engineering and Management

College of Management
National Chiao Tung University
In Partial Fulfillment of the Requirements
For the Degree of Master of Science
In
Industrial Engineering
June 2008
Hsin-Chu, Taiwan
Republic of China

中華民國九十七年六月

建構軟體發展流程之多變量管制系統

研究生：陳啟揚　　指導教授：唐麗英

國立交通大學工業工程與管理學系碩士班

摘要

近年來能力成熟度整合模式(Capability Maturity Model Integration, CMMI-SE/SW/IPPD/SS)已經成為軟體產業品質保證及國際合作的認證標準，其中 CMMI Level 4 所要求的數量化管理必須使用統計製程管制(statistical process control, SPC)來監控軟體發展流程(software development process, SDP)，為了使得軟體發展流程具有穩定性與可預測性，越來越多軟體廠商使用管制圖來監控軟體發展流程。雖然製造業已經成功的運用管制圖來偵測產品或製程變異，但是軟體發展流程與製造流程彼此之間卻有相當大的差異，使得軟體廠商不易成功地將 SPC 應用於軟體發展流程上。由於軟體指標不僅一個，且彼此有關連性，因此本論文之主要目的是發展一套多變量管制系統來同時監控多個軟體指標。本論文最後利用台灣某軟體公司的實際資料來說明本論文之管制系統。

【關鍵詞】CMMI、多變量管制圖、軟體發展流程、軟體指標

Construct the Multivariate Control System
for Software Development Process

Student：Chen Chi Yang　　Advisor：Lee-Ing Tong

Ruey-Yun Horng

Department of Industrial Engineering and Management

National Chiao Tung University

Taiwan

Abstract

In the recently year, the recognizing of Capability Maturity Model Integration (CMMI) have became insurance of quality of a software industry and standard of international cooperation. The level 4 of CMMI called Quantitatively Managed Level requires using statistical process control(SPC) to monitor software development process(SDP). In order to make the SDP more stable and more predictable, more and more software companies start using control chart to monitor SDP. Although manufacturing industry have use control chart to detect variation of products or variation of process successfully for years, the difference between SDP and manufacturing process make it is hard to use control chart on SDP successfully for software companies. Since there are more than one software metrics need to be monitored and one metric correlates with another metric, this study propose a multivariable control system to monitor more than one metrics at the same time. A case study is also presented to verify this system does work.

【Key Words】CMMI、multivariable control chart、software development process、software metrics

目錄

表目錄

圖目錄

第一章 緒論

1.1 研究背景與動機

資訊軟體是目前各個產業最重要的技術之一，這些產業包括：高科技產業、科學、金融業、太空等等。軟體專案的運用及其對各個產業的影響非常重要，然而根據Baker[3]在2001年的研究顯示，大約有30%到40%的軟體專案是失敗的，此外，根據Standish Group在2000 年針對全球財星五百大企業所做的分析顯示，這些大企業的軟體專案成功執行的比例只有26%，其中46%的專案執行不符合原先規劃的需求，而失敗的比例更高達28%。在這些專案中，成本超支平均為189%，專案重新執行的比例佔94%，而時間超支更是高達222%[25]。在軟體產業高度競爭之情況下，軟體品質的穩定性(stability)與可預測性(predictability)成為該產業擁有市場競爭優勢的重要因素。因此監控軟體品質一直是一個重要的議題，一但軟體品質發生了變異，則若能愈早偵測出變異將會節省愈多的成本。

1980年代中期，美國聯邦政府為了對軟體承包商的軟體發展能力進行評估，委託卡內基美隆大學(CMU)軟體工程研究所(SEI)所發展軟體能力成熟度模式(Software Engineering Capability Maturity Model, SW-CMM)，其後又整合了系統工程(Systems Engineering CMM, SE-CMM)、整合產品發展(Integrated Product/Process Development CMM, IPPD-CMM)與採購(Supplier Sourcing CMM, SS-CMM)成為現在的能力成熟度整合模式(Capability Maturity Model Integration, CMMI-SE/SW/IPPD/SS)，與ISO-9000不同的是CMMI特別強調流程改善，目前CMMI已經為軟體的品質保證及國際合作的基本要求，因此政府從2004年開始與民間機構合作CMMI認證制度，提升國家資訊軟體產業的競爭力。CMMI認證共分為五級(Level)，Level 1：初始化(Initial)、Level 2：已管理階段(Managed)、Level 3：已定義階段(Defined)、Level 4：數量化管理(Quantitatively Managed)、Level 5：最佳化(Optimizing)。目前台灣尚未有軟體公司通過CMMI Level 4認證[24]，主要

的困難在於CMMI的條文只是一般性原則，並沒有詳細說明哪些軟體品質與特性需要數量化以及要如何數量化，而且每家軟體公司有興趣的品質特性(quality characteristics)都不同，對每個軟體品質特性衡量的單位或指標(metrics)的定義也都不一樣，這些都使得軟體公司要通過CMMI Level 4非常困難。

管制圖(control chart)是一個功能強大的品質監控工具，製造業常運用管制圖來偵測製程變異與監控製程，現在已有越來越多廠商嘗試在軟體發展流程(software development process, SDP)[1]中使用管制圖，然而成功的經驗卻相當少，主要原因是軟體產業與製造業有以下幾點重要差異：[4][5][8][9][14][15][16]：

(1) 軟體產業重視流程導向(process-centric)

製造業一般都是產品導向(product-centric)，關心的是產品的變異；而軟體則是一種邏輯實體，而不是實際實體，當軟體開發完成後產品是可複製的，而此時軟體的品質也已經確定，因此軟體產業關心的是軟體發展流程的變異，流程的好壞會決定軟體品質的好壞。

(2) 軟體是由人而非機器製造

軟體發展流程是將使用者需求轉換成程式，這過程中包含人的認知活動且需要大量的創造力，因此產品的品質受到人的因素所影響，包括：經驗、知識等。每個專案流程都是獨一無二的，如此會使得軟體發展流程的測量結果不穩定。另外，在製造業中，若機器的產出發生了變異，我們可以輕易地去調整機器的參數，但在軟體產業中，若要調整與人有關的參數卻極易遭到反抗與質疑。

(3) 軟體產業難以獲得大量的同質性(homogeneous)資料

在軟體發展流程中所蒐集的資料量通常都很少，可能一星期甚至一個月才能蒐集到一筆資料，且不同的軟體專案資料特性通常都有相當大的差異，因此

[1]軟體發展流程模型一般是使用瀑布模型(waterfall model)。實務上通常分為需求分析(requirement analysis)、高階設計(high-level design)、低階設計(low-level design)、編碼(coding)、單元測試(unit test)、整合測試(integration testing)、系統測試(system testing)與維護(maintenance)等階段。

難以針對軟體產業及資訊進行合理的分群(subgroup)及建立管制界線(control limit)或基準(baseline)。

(4) 軟體產業包含多個一般原因(multiple common causes)

影響軟體發展流程的因素很多，包括：工具、方法、專案類型、開發環境、語言、專案大小、設計複雜度等。這些因素造成的變異會使得管制界線或基準訂得太大，以致當有其他可歸屬原因(assignable cause)發生時則難以偵測出。

(5) 軟體產業指標(metrics)的表現會隨著不同的發展階段而改變

測量軟體的指標在軟體發展流程的不同階段會有不同的型態(pattern)，如果在每個階段都使用相同的基準將會導致錯誤的結果。

(6) 某些軟體測量是主觀的

由於軟體是看不見的與複雜的邏輯實體，軟體測量可以分為直接測量(direct measures)與間接測量(indirect measures)，直接測量的數據是客觀的，包括：缺失數、成本等，間接測量的數據則是主觀的，例如：功能不能直接測量而是經過功能分數(function point)來量化，計算過程中權重(weight)必須經過人的主觀判斷。即使這些測量都經過良好的定義，也會造成許多測量上的變異。

(7) 軟體產業難以找到有效的測量指標

軟體的品質特性包括：可靠性(reliability)、可修改性(modifiability)、易學習性(understandability)、效能(efficiency)、易使用性(usability)、測試性(testability)、可移植性(portability)、研發成本(cost-to-develop)、研發時間(time-to-develop) 、穩健性(robustness)、顧客滿意度(satisfaction)等。由於上述品質特性都是在最終產品完成後才能確定，在軟體發展流程中監控的指標並非軟體品質，就算軟體發展流程中的指標在管制界線內，也不能代表軟體品質一定是好的。而且這些品質特性之間有些是互相衝突的，例如可移植性高則效能可能差，因此找到一個有效的指標是一個很重要的議題。

為了幫助軟體廠商提升軟體品質及通過CMMI Level 4的認證，本研究希望能針對軟體產業與製造業的差異，建立一套多變量管制系統來有效監控軟體發展流程。

1.2 研究目的

現有中外文獻均是針對每個指標分別繪製單變量管制圖(Univariate Quality Control Chart)進行管制，此表每個指標之間沒有相關性，然而這些指標之間通常有相關性存在，當兩個變數具有相關性時，其二元變數的信賴區間應為橢圓分佈之圖形，如果忽略這些相關性，則所建立之個別管制圖很容易有誤判的情形發生[17]。針對上述問題本研究之主要目的是建立多變量管制圖(Multivariate Quality Control Chart)同時監控兩個以上的軟體品質指標，以準確的偵測軟體品質失控的情況，並快速找出可歸屬原因，然後進行軟體發展流程的改善，以維持軟體品質的穩定。

1.3 研究範圍與假設

本研究之範圍限於軟體專案指標之監控。本研究之假設有以下四點：

1. 軟體指標之間有關聯性且服從多元常態分配。
2. 不同專案之間彼此獨立，即某專案失控的機率不受其他專案影響。
3. 相同類型專案的觀測值來自相同之母體。
4. 不同類型專案的觀測值來自不同之母體分配。

1.4 研究架構

本研究共分五章，第一章為緒論，說明本研究之研究動機與目的；第二章介紹軟體產業界在軟體發展流程中應用管制圖之相關文獻，以及多變量管制圖之介紹；第三章說明如何建構本研究所提之統計製程管制流程；第四章利用台灣某軟體商所提供的實際資料來建構此公司之多變量管制系統，並利用模擬資料來測試此系統確實有效；第五章則是本研究之結論。

第二章　文獻探討

本章首先介紹其他文獻在軟體發展流程中成功使用管制圖所遭遇到的困難與解決的方法，再來則是使用軟體指標應注意的事項，最後簡單介紹多變量管制圖。

2.1 使用 SPC 監控軟體發展流程

在軟體發展流程中使用SPC一直是許多公司所期望的目標，關於SPC是否適用於軟體發展流程一直是一個爭論的議題之一，Lantzy最早開始探討這個議題的[15]，他針對軟體發展流程要成功實施SPC，提出了七個步驟。該研究指出了以下幾個重點：

- 好的指標必須要與最終產品品質具有高度之相關性。
- 必須針對產生實體物件的活動建構指標，例如：程式的撰寫。
- SPC必須用在關鍵的流程。
- 使用SPC的流程必須有能力製造出符合規格的產品。

Kan[13]針對軟體發展過程的指標之理論與應用做了廣泛且深入的研究，他認為繪製在管制圖上的點是在流程進行中(in-process)量測而得，不能代表產品的最終品質，即在專案結束前永遠不會知道產品的品質，因此軟體品質的管制若要做到與製造業一樣的即時(on-time)管制是非常困難。SPC在品質工程模型(quality engineering models)中是一項非常有用的工具，例如：缺陷模式(defect models)與可靠度模式(reliability models)。

在2001年 High Maturity Workshop [19]中SPC是一個重要的議題，共有35個CMM Level 4以上的組織參加此研討會，他們做成以下幾點結論：第一，CMMI Level 3的公司所使用的量測數據常常是無法使用管制圖進行管制的，因為這些量測都還停留在專案階段等級(project phase level)，因此資料粗糙而無法提供Level 4所需要的解析度(granularity)。第二，由於太多共同因素使得變異太大時，可預

測性變差或者無法找到真正的可歸屬因素，使用管制圖的效果會很差，甚至比不使用管制圖更差。

Weller[23]提供了一個成功運用SPC的實例，說明了缺陷(defect)雖然無法全部移除卻可以有效的管理。整個流程可以分為兩階段，第一階段是監控編碼階段後期的檢查流程(Inspection process)[2]，目的是希望流程是穩定的且可預測的，這階段使用個別值移動全距管制圖(*X-Rm* chart)監控檢查速度(inspection rate)與準備速度(preparation rate)、使用單位缺點數管制圖(*U* chart)來監控缺失密度，當流程穩定時便可以估計出此專案總共有多少個缺失數；第二階段則是監控測試階段，測試階段又可分為單元測試(unit test)、整合測試(integration test)與系統測試(system test)等，每個階段都會有期望移除的缺失數目，經由監控實際移除的缺失數目來判斷測試階段是否正常並決定測試階段何時可以結束，這模型對專案經理(project managers)、版本經理(release manager)與軟體發展團隊都有相當大的幫助。Weller以下三提到幾點注意事項：

1. 產品品質與流程品質是不同的，在第一階段檢查速度與準備速度是流程品質的指標，缺陷密度是產品品質的指標，當流程品質穩定時再監控產品的品質才有意義。
2. 變異的來源有可能是來自流程或者是產品，當有失控點時，表示流程失控或者是產品的品質失控。
3. 當有點超出管制界線時，必須找到可歸屬原因(assignable causes)，確定此點屬於特殊情況(special case)才可將此點移除；若其他資料點也受到此因素影響，就算在管制界線內也必須移除。

Florac 等人[7]利用 NASA 專案的資料展示利用管制圖來建立不同專案的流程表現基準(process performance baseline, PPB)的詳細過程，他們針對管制圖首先提出以下六點重要的觀念：

[2]在多種不同的同仁審查方式中，最正式及嚴謹的就是檢查(Inspections)， 檢查的活動依序為規畫、準備、團體審查會議、矯正、驗證。

1. 選擇關鍵的流程。
2. 提供操作型定義。
3. 注意資料的同質性與合理的分群。
4. 使用正確的管制圖。
5. 了解多原因系統(multiple-cause system)與混合原因系統(mix-cause system)。
6. 測試管制界線並且重新計算與更新管制界線。

Jakolte與Saxena兩位[11]針對軟體發展流程的管制圖經濟性設計提出一套成本模型，目前軟體發展流程所使用的管制界限都是沿用製造業的三倍標準差觀念而建，可是軟體產業與製造業不同之處是型二誤差所造成的成本比型一誤差大很多，例如一個缺失如果在編碼階段已經產生卻沒被找到，到了測試階段才被找到的話，修正此缺失的人力成本會比在編碼階段大上很多倍，而且當超出管制界限時並不需要停止整個軟體發展流程來進行修復，型一誤差的成本比製造業小很多，因此這兩位學者建議軟體發展流程所使用的管制界限應小於三倍標準差。

Komuro[14]分享了HSE(Hitachi Software Engineering)在軟體發展流程使用SPC的經驗，他認為在使用SPC於軟體發展流程時應該先監控的流程要視企業的目的而定。若將SPC用於整個軟體發展流程只會增加企業的負擔。經由早期缺失偵測比率(early bug detection rate)與最終缺失率(bug rate)的分析發現，在測試階段之前若能找到越多的缺失就會使產品品質明顯提升，如果想提升最終產品的品質，則必須監控測試階段之前的同仁審視(peer review)流程。Komuro於是進一步分析同仁審視階段與測試階段修正缺失的成本，發現在同仁審視階段若能找到缺失會節省一半以上的成本。Komuro建議使用*X-Rm* chart監控審視速率(review speed)與審視效率(review efficiency)，使用*Z* chart監控缺陷密度，並示範如何找到共同原因並修正管制界限，最後建立出PPB。

Sargut與Demirörs[20]認為SPC對成熟度低的組織也是有幫助的，他們使用某家公司在進入CMMI Level 3前的資料進行分析，雖然此公司已經有定義良好的流

程與指標資料，但是要進行量化分析還是有困難的，在使用SPC前，有許多基礎工作需要完成，包括：診斷公司的現況、規畫改進的方案、選擇指標、定義新的流程、重新組織資料庫並找到初始的基準、使用其他量測來標準化指標資料。在使用SPC之前的資料都是以表格、直方圖或柏拉圖等方式呈現，分析也是憑經驗或直覺，而使用SPC後最大的好處即是可輕易找到離群值並偵測到特殊情況，且在做決策的時候獲得很大的幫助。在管制圖方面還有以下兩個重點：

1. 當管制圖超出管制界限時必須找到原因才能採取對的行動，例如以缺陷密度為例，超過上限時可能是軟體品質差或者檢驗流程效率太好，而超過下限的時候可能是軟體品質佳或者是檢驗效率太差，這時要因分析圖(Cause and Effect Diagram)是一個重要的工具。
2. 由於軟體發展流程的複雜性與許多外部的參數會膨脹資料的變異，如果只分析單一特殊流程，則資料量可能太少，如果同時分析多個一般流程，由於變異的膨脹則可能找不到某些失控點，因此我們常常在此做取捨。

Jacob與Pillai[10]使用管制圖監控編碼階段的同仁審視流程與缺失密度，並且持續改善編碼流程。Manlove與Kan[16]使用管制圖監控設計、編碼、測試與維護階段的不同軟體指標。目前SPC已經成功地使用於同仁審視、編碼與測試階段，不過每個成功的經驗差異都很大，使用的管制圖與軟體指標也都不同，而且同時監控多個管制圖在實際使用上較不方便。

2.2 軟體指標的相關議題

由於CMMI並沒有規定要用哪些指標與這些指標該如何使用，因此不同的組織可能會使用不同的指標，但對指標的定義與使用的方法不同是很正常的，然而這些組織的目標全都是相同的：高品質、高生產力、低成本與短的週期時間(cycle time)。所有軟體指標的使用是與這些目標都有高度相關，因此高成熟度的組織(CMM Level 4 or 5)在指標的選擇與使用的方法上應該會有許多的共同點，而根據Paulk[18]在1999年針對13個高成熟度組織的調查會發現確實有許多相似的地

方。

2.2.1 基礎建設

成熟度低的組織由於基礎建設做得不好，在使用SPC前會遇到許多的困難，且付出許多額外的成本而且效果也差。有好的基礎建設才會有好的資料可進行分析。而通常基礎建設才是最花成本與時間的，但是做好基礎建設除了有利於使用SPC的技術之外，還有許多好處，包括：更有效率的偵測離群值(outliers)、找到改善的方案、更好的流程控制，更重要的是同時改善了整個軟體發展流程。當做好基礎建設之後繪製管制圖與分析的工作變得很容易，因此成熟度高的組織在使用SPC時不需要花太多成本。

Sargut與Demirörs[20]提出在使用SPC之前有三項重要的基礎建設，包括：(1)文件化程序(documented procedures)(2)資料庫(database)的維護(3)流程的量測必須有良好的定義。指標的量(資料的個數)與質(精確與詳細)對於建立可靠的管制界限是非常關鍵的。完成基礎建設後，每個組織對每一個指標都有不同定義、蒐集方式與解釋，因此相同的指標有不同的特性與複雜度，每個指標的標準化方法也不同，所以很難建立一個相同的邏輯來實現SPC。

Jalote[12]針對高成熟度的組織在指標的基礎建設做了詳細的研究，這些組織通常都有專門負責改善軟體發展流程的SEPG(software engineering process group)，此單位在組織中扮演協調者與提供協助的角色來幫助組織進行流程改善，其主要工作有：品質保證、訓練、流程分析與定義、顧問服務等。除了SEPG外還有兩個重要基礎建設就是：流程資料庫與流程能力基準。事實上CMMI規定組織必須有一個流程資料庫以儲存專案的歷史資訊，但是沒指定資料庫要一定維護哪些資料，這些高成熟度的組織大部分會紀錄以下5種資料：專案大小、人力、行程、缺陷與風險。所有的組織會根據過去的資料建立流程能力基準，而這也是CMMI Level 4的主要目的之一，當基準建立之後，組織可以很明確的確定當流程在管制

下的話產出會是怎樣的結果。能力基準是組織在某個時間點的流程能力之現況，經由固定的時間重新計算流程能力基準，便可以找到趨勢並設立流程改進目標。基礎建設可以建構一個如圖2.1之流程。

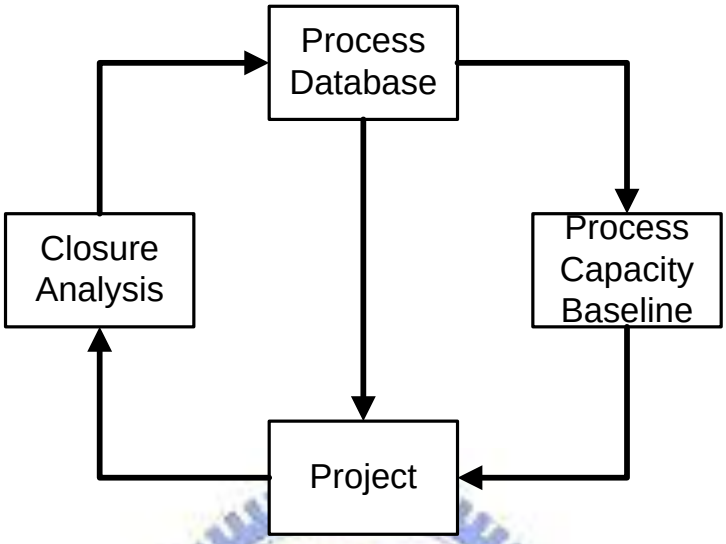

圖2.1 流程資料庫與流程能力基準[12]

2.2.2 軟體指標的選擇

如之前所述，軟體指標並非軟體品質，指標只是在軟體發展流程中所蒐集的量測數據，但我們希望藉由監控指標來保證當我們的流程是穩定的時候則軟體品質也必定是穩定的。在使用軟體指標之前我們必須清楚甚麼是軟體品質，才有辦法根據軟體品質建構合適的量化指標。

Kan[13] 詳細的分析軟體品質，他認為：(1)品質是一個多維度(multi dimension)的概念，品質的維度包括：有興趣的個體(軟體)、對此個體的觀點(使用者的觀點或者工程師的觀點)與此個體的品質特性。(2)抽象化的程度(level of abstraction)不同，以軟體的執行速度為例，當一般使用者可能會憑感覺去認定程式執行的快慢，而工程師會跟根據演算法的時間複雜度[3]是$O(n)$或者$O(n^2)$ [4]來判

[3] 演算法的時間複雜度是指演算法需要消耗的時間資源。一般來說，電腦演算法是問題規模 n 的函數 $f(n)$，演算法的時間複雜度也因此記做 $T(n)=O(f(n))$ 。

[4] Big O notation 是用於描述函數漸進行為的數學符號。更確切地說，它是用另一個（通常更簡單的）函數來描述一個函數數量級的漸近上界。在電腦科學中，它在分析演算法複雜性的方面非常有用。

斷程式的快慢。一般在業界在實務上會把軟體品質定義為兩個層面：(1)產品固有的品質，或稱為small q。(2)顧客滿意度，或稱為big Q。

Lantzy提出一個找到好的軟體指標的方法[15]，首先組織與顧客必須協商與溝通，找出軟體品質的優先順序(priority)與可接受的公差(acceptable tolerances)，然後再根據品質優先順序尋找相關性高的指標進行管制。

Jalote拜訪多家高成熟度的組織並研究這些組織如何有效的使用指標[12]。通常使用指標有三大目的：(1)規畫專案(2)監控與控制專案與(3)整個軟體發展流程管理與改善。其中(1)與(2)是屬於專案層級而(3)則是屬於組織層級，在使用指標前管理者一定要清楚知道使用的目的，才不會用錯方法與用錯指標。

2.2.3 軟體指標的特殊形態

許多軟體指標會呈現上升趨勢、下降趨勢、S曲線(S-curve)或指數行為(exponential behavior)，例如：編碼階段的編碼整合百分比(percentage of code integration)與編碼審查進度(code review progress)、測試階段的剩餘錯誤(remaining errors)與失效強度(failure intensity)。圖2.2是一個典型的指數遞減曲線。圖2.3則是具有指數行為的軟體指標。

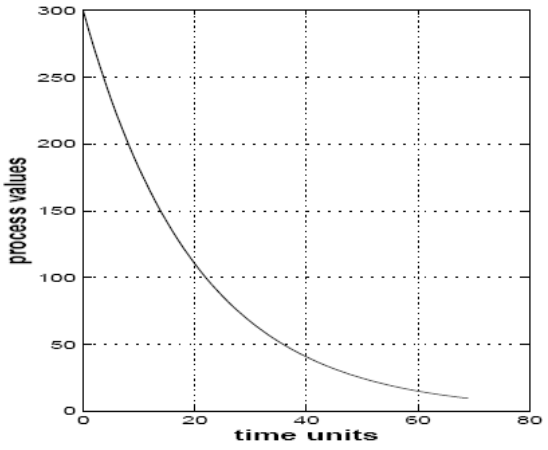

圖2.2 指數遞減曲線[6]

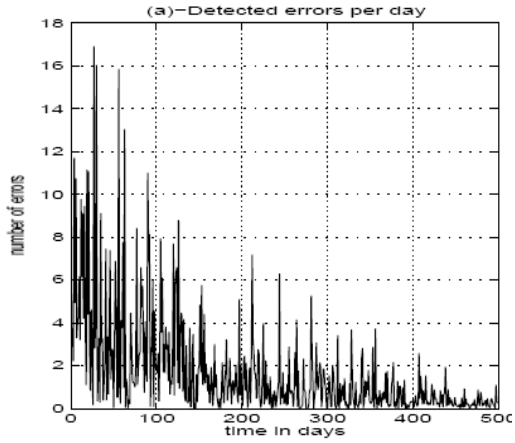

圖2.3 具有指數行為的軟體指標[6]

Cngussu et al. [6]提出一個對數的資料轉換方法來處理指數行為，步驟如下：

Step 1：將每個觀測值v_i取對數，$ld_i = log(v_i), i = 1, \dots, n$。

Step 2：計算ld_i的最小平方回歸線lsf_i。

Step 3：計算ld_i與lsf_i的歐式距離，$x_i = \sqrt{(lsf_i - ld_i)^2}$。

Step 4：使用x_i計算管制界線。

Step 5：繪製管制圖。

而Manlove與Kan兩位學者使用四次迴歸處理S-curve[16]，模型如下：

$$y = a + b \times x + c \times x^2 + d \times x^3 + e \times x^4 + \varepsilon$$

其中y是指標，x是時間，ε是殘差項。

Step 1：使用最小平方法將以前的專案資料適配4次回歸線，估計出迴歸係數。

Step 2：利用MSE估計標準差，即$\hat{\sigma} = MES$。

Step 3：以回歸線為CL，$y + 3\hat{\sigma}$為UCL，$y - 3\hat{\sigma}$為LCL繪製管制圖。

2.2.4 選擇何種管制圖

管制圖的操作與原理是由蕭華特(W.A. Shewhart)在1924年貝爾實驗室內所提出的，典型的管制圖會包含一中心線(central line, CL)，用來表示品質特性之平均值。另外還有管制上限(upper control limit, UCL)與管制下限(lower control limit,

LCL)，用來表示變異的容許範圍。繪製於管制圖上的點是一群統計資料且依其發生時間的先後順序排列，因此是屬於時間序列(time series)。管制圖可分為兩大類：

(1) 計量值管制圖(Control Charts for Variables)

1. 平均值與全距管制圖($\bar{X}-R$ chart)。
2. 平均值與標準差管制圖($\bar{X}-S$ chart)。
3. 中位數與全距管制圖($\tilde{X}-R$ chart)。
4. 個別值與移動全距管制圖(*X-Rm* chart)。

(2) 計數值管制圖(Control Charts for Attributes)

1. 不良率管制圖(*p* chart)。
2. 不良數管制圖(*np* chart)。
3. 缺點數管制圖(*C* chart)。
4. 單位缺點數管制圖(*U* chart)。
5. 標準化之單位缺點數管制圖(*Z* chart)。

我們要先了解要進行管制的軟體指標是何種變數，再決定使用何種管制圖。在軟體發展流程中的觀測值經常是個別值而非一組樣本，因此大部分的學者都建議使用*X-Rm* chart、*p* chart、*C* chart、*U* chart、*Z* chart與趨勢圖(*X* chart)，工業界常用的$\bar{X}-R$ chart幾乎沒在使用。接下來針對文獻上常用的軟體指標之整理如表2.1：

指標	流程	定義	管制圖
缺陷密度	測試、編碼、同仁審視	$\frac{\text{\# of detected defect}}{\text{size of work product}}$	*X-Rm* chart、*U* chart、*Z* chart
檢查效率	編碼階	$\frac{\text{\# of defects removed in inspection}}{\text{\# of all major defects}}$	*XmR* chart

(Inspection Effectiveness)	段的檢查		
編碼檢查速度 (Inspection Preparation Rate)	編碼階段的檢查	$\frac{\text{\# of SLOC for preparation}}{\text{effort spent}}$	*X-Rm* chart
檢查績效 (Inspection Performance)	編碼階段的檢查	$\frac{\text{\# of defects found}}{\text{effort spent}}$	*X-Rm* chart
審視效率 (Review Efficiency)	同仁審視	$\frac{\text{\# of detected defects}}{\text{effort spent}}$	*X-Rm* chart
審視速度 (Review Rate)	同仁審視	$\frac{\text{\# of modified and new SLOC for review}}{\text{effort spent}}$	*X-Rm* chart
Defect Backlog	測試	# of open defects	趨勢圖
剩餘錯誤數 (Remaining Errors)	測試	# of remaining errors	趨勢圖
重工百分比 (Rework Percentage)	專案結束後、專案期間	$\frac{\text{rework effort}}{\text{total effort}}$	*X-Rm* chart
測試進度S曲線 (Test Progress S Curve)	測試	cumulated number of checked test cases	趨勢圖
編碼整合比率	編碼	percentage of integrated SLOC of the	趨勢圖

		entire software project	
Backlog Management Index A (BMI-A)	維護	$\frac{\text{\# of defect closed during a mouth}}{\text{\# of defect arrivals during a mouth}}$	*X-Rm* chart
Backlog Management Index B (BMI-B)	維護	$\frac{\text{\# of open defect at the end of mouth}}{\text{\# of defect arrivals during a mouth}}$	*X-Rm* chart
成本績效指標 (CPI)		$\frac{\text{earned value}}{\text{actual cost}}$	*X-Rm* chart
進度績效指數 (SPI)		$\frac{\text{earned value}}{\text{planned value}}$	*X-Rm* chart

表2.1 常用之軟體發展流程指標

除了上述論點外，以下幾點必須注意：

(1) 以上的定義只是一般性的原則，每家公司的定義都不相同，以缺陷為例，缺失必須依據缺失類型、優先權與嚴重性來分類(IEEE Std 1044-1993, IEEE Std 1044.1-1995)，因此每個缺失應該有不同的權重。而計算每個單元的大小除了SLOC之外功能點數也是常用的單位之一。在使用每個指標之前都必須經過精確的定義。

(2) 以上使用何種管制圖依問題而定，必須視實際資料而定，以缺陷密度為例，大部分的研究都使用*U* chart[8] [23]，但是*U* chart的管制界線與樣本大小(在軟體則是產品大小)的平方根成反比，因此當樣本大小時常變動的時候*X-Rm* chart也很常用[10][20]。

(3) 要注意分配的假設。

2.3 多變量管制圖

在許多情況下同時監控或者控制兩個或兩個以上的軟體指標是必須的，例如：在專案管理的部分要監控SPI(x_1)與CPI(x_2)，假設這兩個變數是獨立的常態分配，則可以使用一般的$\bar{x}$ chart監控這兩個指標，結果如圖2.4所示，如果此兩個圖都沒有超出管制界限則認為流程在管制下。但若將(x_1, x_2)繪製在同一個平面下如圖2.5所示，此圖可以讓我們同時檢視兩個指標，會發現有一個點似乎遠離其他點，儘管所有的點在原本的管制圖下是在管制下。另外，當在管制下時$\bar{x}_1$或$\bar{x}_2$超出個別管制界限的機率都是0.0027，但是同時超出管制界限的機率則為

(0.0027)(0.0027)= 0.00000729因此型一誤差的機率為

0.0027+0.0027-0.00000729=0.00539271，比我們預期的0.0027還要高許多。

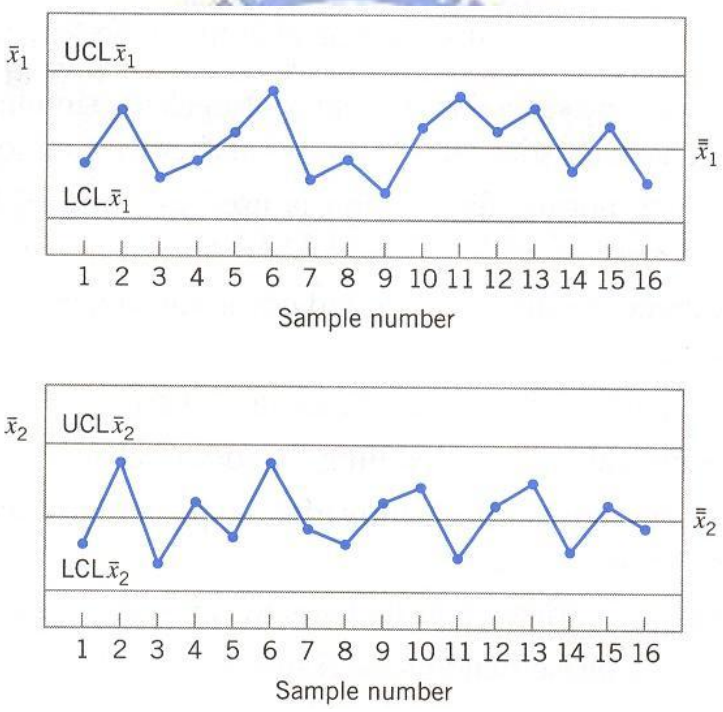

圖2.4 單變量管制圖[17]

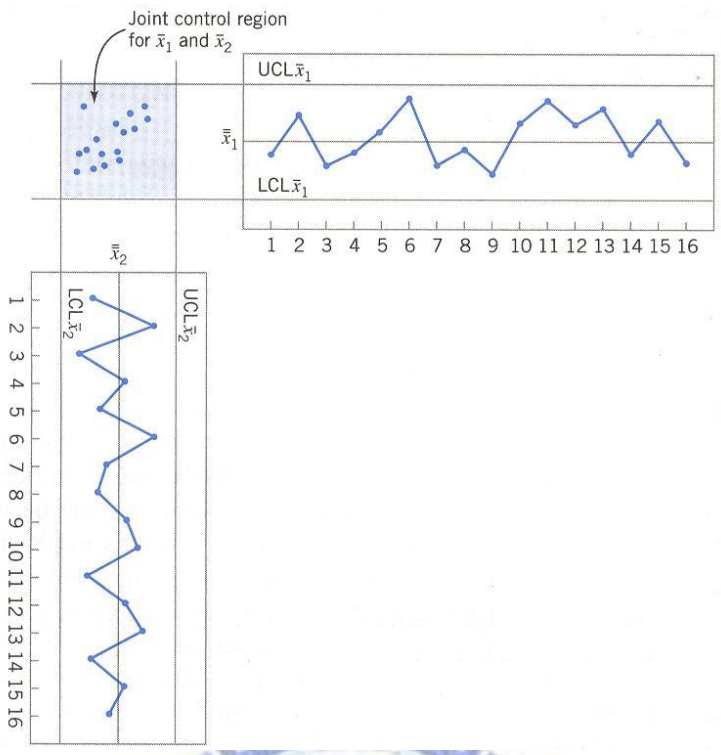

圖2.5 將兩個單變量管制圖之聯合管制區域[17]

2.3.1 多變量常態分配

假設有p個隨機變數$x_1, x_2, \ldots, x_p$，則此多變量常態機率密度函數為

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{S}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

其中粗體的符號為矩陣，$\mathbf{x} = [x_1, x_2, \ldots, x_p]$表示此p個隨機變數所形成之矩陣，$\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_p]$是此p個隨機變數之期望值所形成之矩陣，

$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12}^2 \cdots & \sigma_{1p}^2 \\ \vdots & \ddots & \vdots \\ \sigma_{1p}^2 & \cdots & \sigma_p^2 \end{bmatrix}$是母體的變異數共變數矩陣(variance-covariance matrix)。

假設從此多元常態分配隨機抽出一組樣本$\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_p}$，則此樣本平均數所形成之陣列為

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$$

其中第i個樣本會形成一個p維的陣列$[x_{i1}, x_{i2}, \ldots, x_{ip}]$。而樣本的變異數共變數矩陣為

$$\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x_i} - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathrm{T}}$$

且$\bar{\mathbf{x}}$為**x**之不偏估計量，**S**是**Σ**之不偏估計量。

2.3.2 Hotellint T^2管制圖

Hotellint T^2管制圖是最常見的多變量管制圖，本文先以p=2為例子，x_1與x_2會服從二元常態分配，機率密度函數如圖2.6。當$\sigma_{12} = 0$時$f(x)$=k$(0 < k < 1)$與此函數圖形會相交於一個圓，表示x_1與x_2的聯合信賴區間是一個圓。當$\sigma_{12} \neq 0$時$f(x)$=k與此函數圖形會相交於一個橢圓，表示x_1與x_2的聯合信賴區間是一個橢圓。我們可以在平面繪製一個橢圓代表管制界限，並且把$(\bar{x}_1, \bar{x}_2)$繪製於此平面上如圖2.7與圖2.8所示，由圖可以發現兩個變數的相關會影響管制橢圓(control ellipse)的形狀，然而此圖無法表示每個點的先後關係是其缺點。

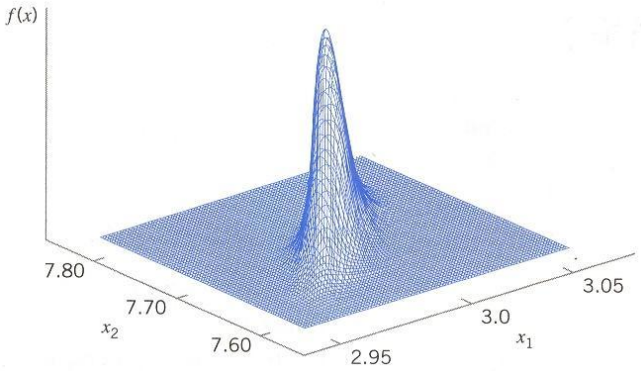

圖2.6 p=2時之多變量常態分配[17]

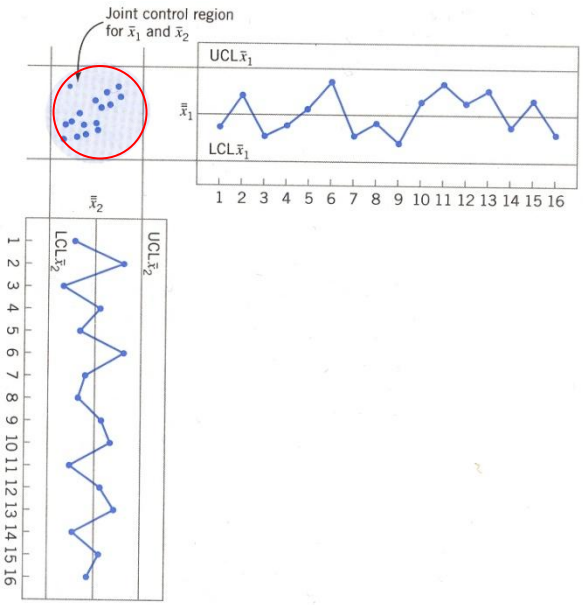

圖2.7 兩變數獨立時之管制橢圓[17]

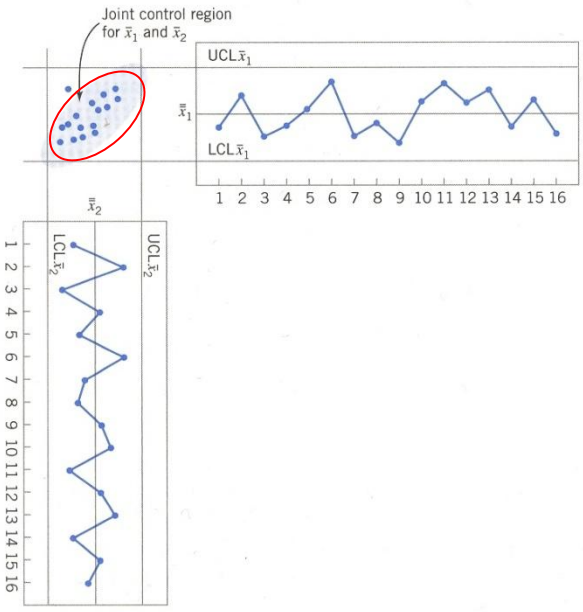

圖2.8 兩變數相關時之管制橢圓[17]

當x_1與x_2服從二元常態分配時則統計量會服從自由度為2的卡方分配：

$$\chi_0^2 = \frac{n}{\sigma_1^2\sigma_2^2 - \sigma_{12}^2}[\sigma_2^2(\bar{x}_1 - \mu_1)^2 + \sigma_1^2(\bar{x}_2 - \mu_2)^2 - 2\sigma_{12}(\bar{x}_1 - \mu_1)(\bar{x}_2 - \mu_2)]$$

如果x_1與x_2的平均數為μ_1與μ_2的話，此式子的物理意義為觀測值$(\bar{x}_1, \bar{x}_2)$與理論值(μ_1, μ_2)之馬氏距離(Mahalanobis distance)，χ_0^2落在UCL=$\chi_{\alpha,0}^2$上方的機率為α，可以卡方繪製管制圖如圖2.9所示，與管制橢圓不同的是繪製在此圖上的資料是時

間序列。

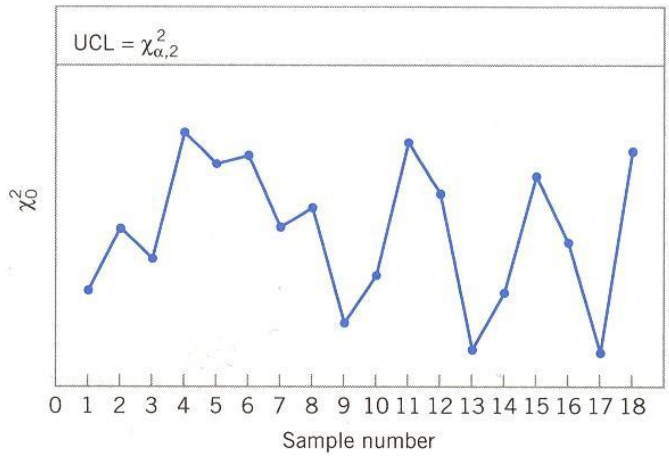

圖2.9 p=2時的卡方管制圖[17]

若推廣到p≧2，則有以下兩種情況：

(1)當**μ**與**Σ**已知時，統計量與管制界限如下：

$$\chi_0^2 = n(\bar{\mathbf{x}} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\bar{\mathbf{x}} - \boldsymbol{\mu})$$

$$\mathrm{UCL} = \chi^2_{\alpha,p}$$

(2)當**μ**與**Σ**未知時，則以$\mathbf{S} = \begin{bmatrix} \bar{s}_1^2 & \bar{s}_{12}^2 & \cdots & \bar{s}_{1p}^2 \\ \vdots & & \ddots & \vdots \\ \bar{s}_{1p}^2 & & \cdots & \bar{s}_p^2 \end{bmatrix}$來估計**Σ**，以$\bar{\bar{\mathbf{x}}} = \begin{bmatrix} \bar{\bar{x}}_1 \\ \vdots \\ \bar{\bar{x}}_p \end{bmatrix}$估計 **μ**，使用以下統計量：

$$T_0^2 = n(\bar{\mathbf{x}} - \bar{\bar{\mathbf{x}}})^{\mathrm{T}} \mathbf{S}^{-1} (\bar{\mathbf{x}} - \bar{\bar{\mathbf{x}}})$$

則解析用管制圖之管制界線如下：

$$\mathrm{UCL} = \frac{p(m-1)(n-1)}{mn-m-p+1} F_{\alpha,p,mn-m-p+1}$$

$$\mathrm{LCL} = 0$$

管制用管制圖之管制界限如下：

$$\mathrm{UCL} = \frac{p(m+1)(n-1)}{mn-m-p+1} F_{\alpha,p,mn-m-p+1}$$

$$\mathrm{LCL} = 0$$

其中p表指標的個數，n表樣本大小，m表樣本的個數。

2.3.3 樣本大小為 1 之 Hotellint T^2管制圖

在軟體過程中樣本大小為1是很常見的情況，因此使用以下公式：

(1)當$\boldsymbol{\mu}$與$\boldsymbol{\Sigma}$已知時統計量與管制界限如下：

$$\chi_0^2 = (\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

$$\mathrm{UCL} = \chi_{\alpha,p}^2$$

(2)當$\boldsymbol{\mu}$與$\boldsymbol{\Sigma}$未知時，估計$\boldsymbol{\Sigma}$有許多的方法，最常見的有兩種：$\mathbf{S}_1$的概念是混合m個觀測值，而$\mathbf{S}_2$的概念是利用相鄰的兩個觀測點相減$\mathbf{v}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$　i=1,2, ⋯,m-1。

$$\mathbf{S}_1 = \frac{1}{m-1}\sum_{i=1}^{m}(\mathbf{v}_i - \bar{\mathbf{x}})\,(\mathbf{v}_i - \bar{\mathbf{x}})^{\mathrm{T}}$$

$$\mathbf{S}_2 = \frac{\mathbf{V}^{\mathrm{T}}\mathbf{V}}{2(m-1)} \quad \text{其中}\mathbf{V} = \begin{bmatrix} \mathbf{v}_1^{\mathrm{T}} \\ \mathbf{v}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{v}_{m-1}^{\mathrm{T}} \end{bmatrix}$$

此時Hotellint T^2統計量為：

$$T^2 = (\mathbf{x}-\bar{\mathbf{x}})^{\mathrm{T}}\mathbf{S}^{-1}(\mathbf{x}-\bar{\mathbf{x}})$$

解析用管制界限為：

$$\mathrm{UCL} = \frac{(m-1)^2}{m}\beta_\alpha\left(\frac{p}{2}, \frac{m-p-1}{2}\right)$$

$$\mathrm{LCL} = 0$$

管制用管制界限為：

$$\mathrm{UCL} = \frac{p(m+1)(m-1)}{m^2 - mp}F_\alpha(p, m-p)$$

$$\mathrm{LCL} = 0$$

2.3.4 尋找失控的變數

當超出管制界限時則必須找出p個變數中是哪幾個失控，一般常用的方法是利用Montgomery等[17]建議的分解T^2統計量(decomposition of T^2)將T^2分解，公式如下：

$$d_i = T^2 - T_{(i)}^2$$

$T_{(i)}^2$是剔除第i個變數並且重新計算T^2值，而d_i為變數i對整體的T^2的影響，當分別計算出所有d_i值後，Montgomery建議以為$\chi_{\alpha,1}^2$臨界值，當d_i超過$\chi_{\alpha,1}^2$的變數很可能就是失控的變數。

第三章　建構軟體指標之管制流程

由於軟體產業許多特性與製造業不同，傳統的管制圖無法直接套用於軟體產業，且考量到軟體法展流程需要同時監控多個具有關聯性的指標，本研究之主要目的是建構一套適用於軟體產業的多變量管制管制流程，以協助軟體廠商監控軟體發展流程中每個指標的變異，並且在失控的時候能夠準確地找到失控的指標。本研究方法假設所有之軟體指標來自相同的母體分配，且不同專案之間是互相獨立的，即某專案失控的機率不受它其他專案所影響。本研究所建構之軟體指標之管制流程共有以下七個步驟：

步驟 1：確定組織目標。

組織的目標可分成以下三類：

(1) 提升產品品質與可靠度。

(2) 縮短週期時間。

(3) 降低成本。

步驟 2：根據組織目標選擇要監控的流程與軟體指標。

如果盲目的將管制圖用在整個軟體生命週期裡將會導致失敗的結果，要找到直接影響組織目標的關鍵流程才是真正有幫助的，而關鍵流程有可能是軟體測試流程、同仁審視流程、過去有出問題的流程、使用新技術的流程、工作負荷量明顯比以前高的流程等，而不同的流程需要監控的指標亦不同。

通常要提升產品品質與可靠度要監控測試階段的缺陷密度或監控測試階段之前的同仁審視流程(peer review)的審視效率(review efficiency)與審視速度(review rate)指標。要縮短周期時間則監控專案的成本績效指標(Cost Performance Index, CPI)。要降低成本則監控時程績效指標(Schedule Performance Index, SPI)。

以下步驟 3 到步驟 6 的主要目的是建構相同類型專案的流程能力基準，此處值得注意的是建構流程能力基準資料的量並非越多越好，資料的“品質”才是最重要的，如此才能能保證此基準用於管制未來的同類型專案是合理的。

步驟 3：蒐集相同專案性質的資料與資料處理。

由於軟體發展流程的觀測資料是要由人為主觀判斷而得因此無法像製造業般客觀，故這些資料之蒐集需有一套完整的計畫，每個指標使用哪些工具來測量、測量的方法、測量的時間點進行測量都必須明確。

由於離群值會誤導管制圖的繪製，繪製所有專案之所有觀測點的所有指標之盒鬚圖以找出可能之離群值，若只有少數的離群值，則可直接刪除，但若有大量的離群值則必須進一步追查造成這些離群值的原因，以決定是否刪除或修正這些離群值。

四分位距法(Forth Spread)常被用來篩選離群值。首先定義Q_1及Q_3為這組資料的第一四分位數(first quartile)及第三四分位數(third quartile)，令$Q_f = Q_3 - Q_1$，則非離群值資料之下限(D_L)及上限(D_U)可以下列兩式求得：

$$D_L = Q_1 - 1.5 * Q_f$$

$$D_U = Q_3 + 1.5 * Q_f$$

若在這組資料中有任何樣本資料點小於D_L或大於D_U則為離群值。

步驟 4：利用所有專案估計管制圖之參數。

一般軟體專案的資料型態如圖 3.1 所示，不同專案的開始時間與結束時間可能相同也可能不同。因此把所有專案的資料重新排列如圖 3.2 所示，會發現軟體專案的資料與一般傳統管制圖最大不同之處就是在每個專案最後一點與下個專案的第一點之間不屬連續的時間序列。

圖 3.1 一般軟體專案的資料型態

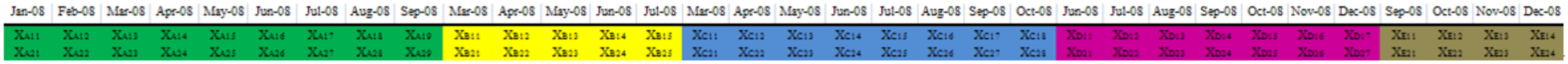

圖 3.2 軟體專案重新排列後的資料型態

利用以下公式可估計變異數與共變數矩陣：

$$\mathbf{S} = \frac{1}{\sum_{l=1}^{q} m_l - 1} \sum_{l=1}^{q} \sum_{j=1}^{m_l} (\mathbf{x}_{lj} - \bar{\mathbf{x}})(\mathbf{x}_{lj} - \bar{\mathbf{x}})^{\mathrm{T}}$$

$$\mathbf{x}_{lj} = \begin{bmatrix} x_{lj1} \\ x_{lj2} \\ \vdots \\ x_{ljp} \end{bmatrix}$$

$$\bar{\mathbf{x}} = \sum_{l=1}^{q} \sum_{j=1}^{m_l} \begin{bmatrix} x_{lj1} \\ x_{lj2} \\ \vdots \\ x_{ljp} \end{bmatrix}$$

其中q表專案個數、m_q表每個專案的樣本數及p表指標個數。

利用以下公式計算管制用管制圖的管制界線：

$$\mathrm{UCL} = \frac{\left(\sum_{l=1}^{q} m_l - 1\right)^2}{\sum_{l=1}^{q} m_l} \beta_{\alpha}\left(\frac{p}{2}, \frac{\sum_{l=1}^{q} m_l - p - 1}{2}\right)$$

$$\mathrm{LCL} = 0$$

關於型一誤差 α 的設定，Jalote[11]認為製造業所使用的 3σ 管制界限之 α 為 0.0027，並不適用於軟體產業，此乃因在軟體產業中型二誤差所造成的損失比型一誤差還嚴重，例如：如果在軟體的設計階段有一個缺陷發生，且在同仁審視流程中並沒有被偵測出，則在測試階段時此缺陷可能已經影響到程式中的多個單元，此時要修復程式所花的成本會比在設計階段高很多。製造業使用的 3σ 的平均串連長度(Average Run Length, ARL)為 370，但是每個軟體專案的觀測值往往不超過 20 個，如此高的ARL_0完全不適用於軟體產業，因此本研究將 α 設定為 0.05 以降低型二誤差。

步驟 5：計算每個專案的每個觀測值的 Hotellint T^2值並繪製於管制界線中。

使用以下公式計算第l個專案的第j個觀測值的Hotellint T^2值：

$$T_{lj}{}^2 = \left(\mathbf{x}_{lj} - \bar{\mathbf{x}}\right)^{\mathrm{T}} \mathbf{S}^{-1} (\mathbf{x}_{lj} - \bar{\mathbf{x}})$$

繪製管制圖如圖3.3所示，注意圖中由於每個專案中是連續的時間序列而專案間則非連續的時間序列，因此此圖可以看做是5個管制圖，每個管制圖內除了看是否超出管制界限之外亦可以用連數測試(run test)來測試是否有上升趨勢或下降趨勢。本研究唯一判斷失控的測試只有任何一點超過管制界線。

步驟 6：針對異常點進行分析。

管制圖繪製完成後如果有點超出管制界限則表示表示可能有可歸屬因素存在，由管制圖可以輕易找出每個專案的失控點，若某個專案只有少數點失控，則直接移除即可；若發現某個專案有大量失控值，則此專案資料可能不適用於建構流程能力基準，計算分解T^2統計量：$d_i = T^2 - T^2_{(i)}$如果超過$\chi^2_{0.05}(1)$則表示該指標可能為失控的指標，跟專案人員溝通並了解失控的原因之後，確認可歸屬因素存在則把此專案資料移除，否則不應該隨便刪除資料。

重複步驟 2 到步驟 6 直到所有失控點移除為止就可進入步驟 7。

步驟 7：計算流程能力基準並持續監控組織所有同類型的專案。

假設此步驟剩下q'個專案，每個專案剩下$m_{q'}$個觀測值，則利用以下公式計算流程能力基準：

$$\mathrm{UCL} = \frac{p\left(\sum_{l=1}^{q'} m_{l'} + 1\right)\left(\sum_{l=1}^{q'} m_{l'} - 1\right)}{\sum_{l=1}^{q'} m_{l'}^{\,2} - \sum_{l=1}^{q'} m_{l'}\, p} F_{\alpha}\left(p, \sum_{l=1}^{q'} m_{l'} - p\right)$$

$$\mathrm{LCL} = 0$$

利用以下公式計算 Hotellint T^2值，並繪入流程能力基準中：

$$T^2 = (\mathbf{x} - \bar{\mathbf{x}})^{\mathrm{T}} \mathbf{S}'^{-1} (\mathbf{x} - \bar{\mathbf{x}})$$

其中$\mathbf{S}' = \frac{1}{\sum_{l=1}^{q'} m_{l'} - 1} \sum_{l=1}^{q'} \sum_{j=1}^{m_{l'}} (\mathbf{x}'_{lj} - \bar{\mathbf{x}}')(\mathbf{x}'_{lj} - \bar{\mathbf{x}}')^{\mathrm{T}}$

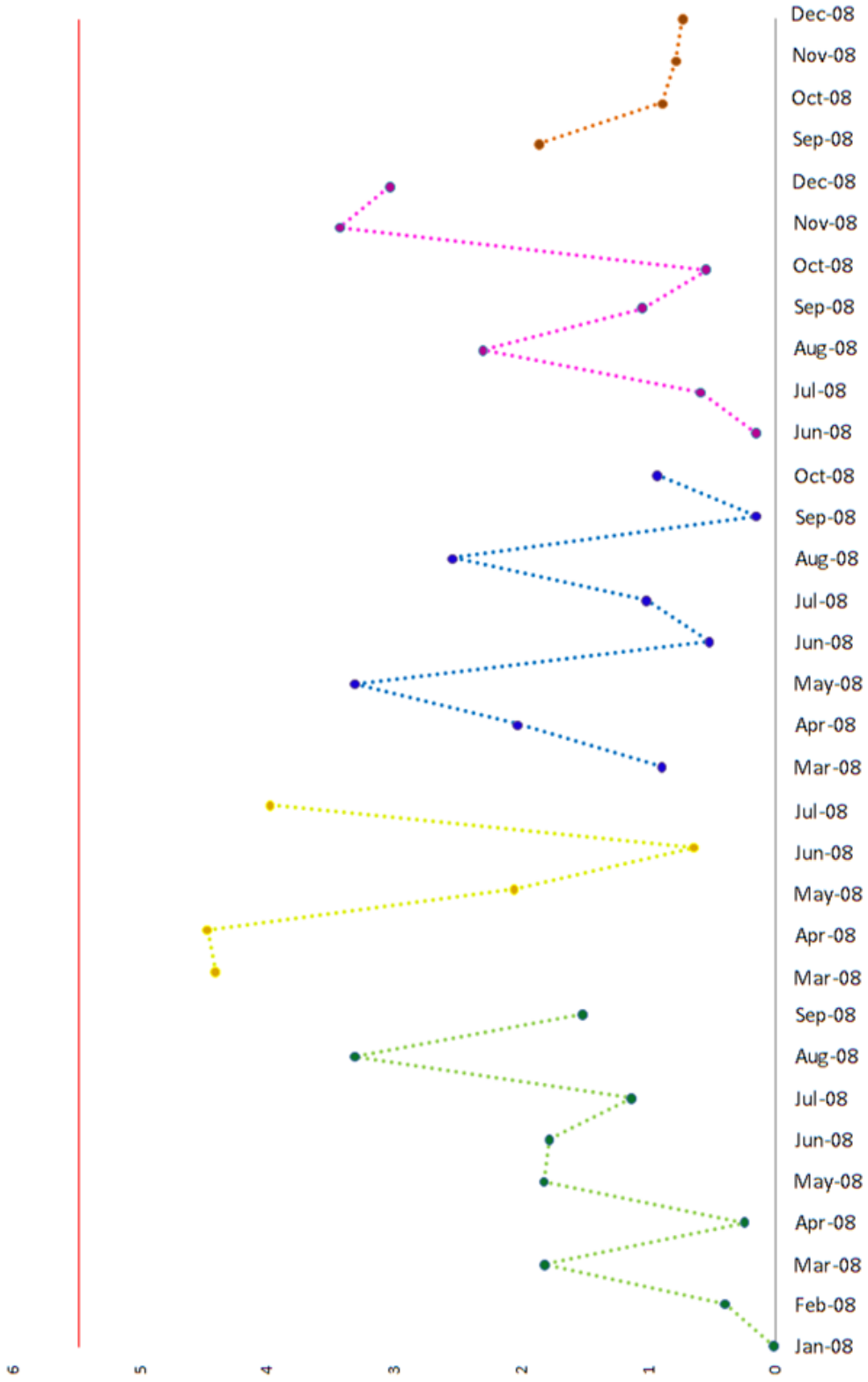

圖 3.3 本研究所提出之管制圖

上述的七個步驟可以用圖3.4之流程圖示之。

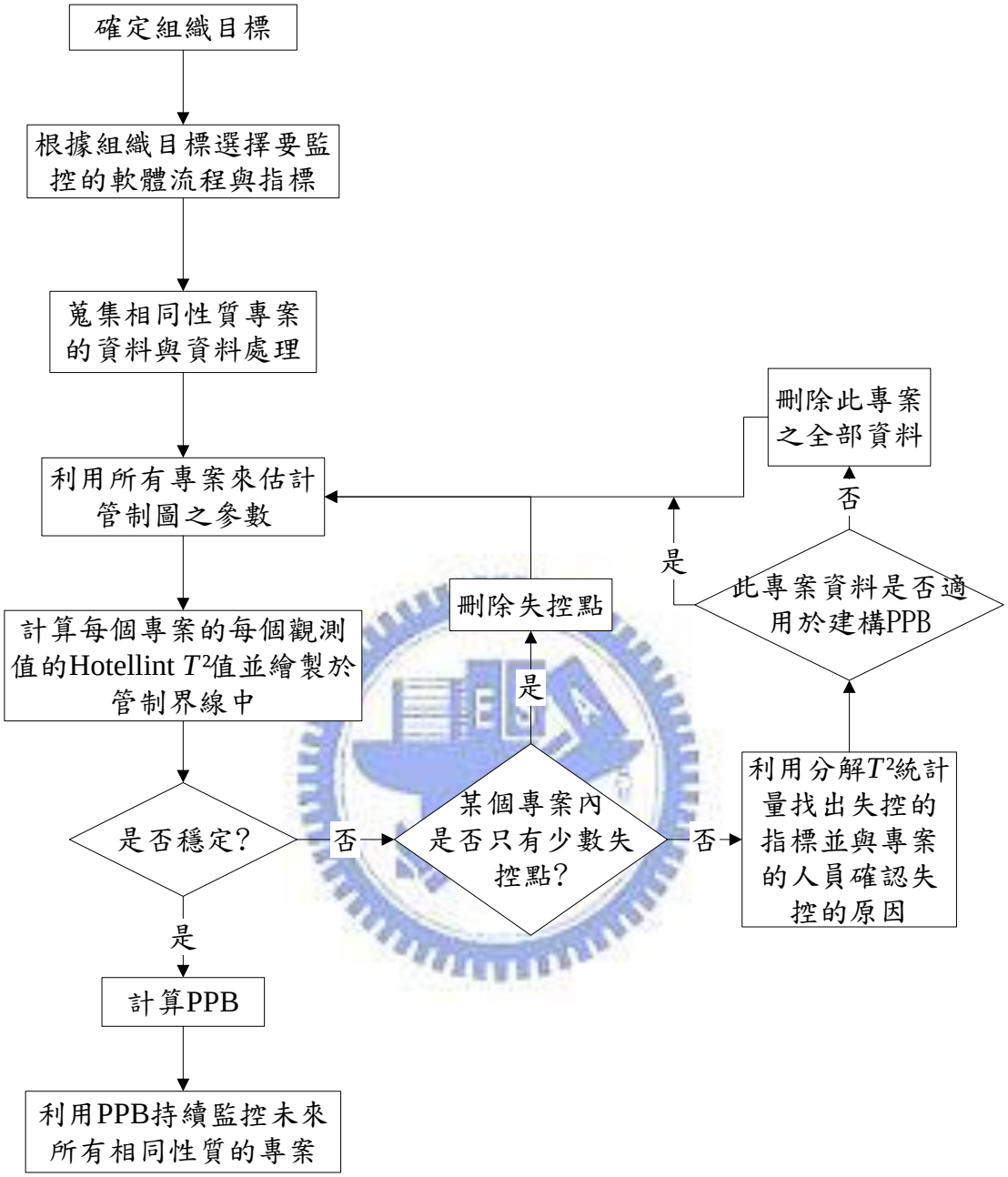

圖 3.4 建構軟體發展流程之多變量管制系統之流程圖

第四章　實例驗證

本研究利用台灣某軟體公司的實際資料來說明本研究方法之有效性。

4.1 建構某軟體公司之多變量管制系統

步驟 1：確定組織目標。

本案力公司欲對所有專案的時程與成本進行控管。

步驟 2：根據組織目標選擇要監控的流程與軟體指標。

監控組織所有專案之 CPI 與 SPI 指標。

步驟 3：蒐集相同專案性質的資料與資料處理。

蒐集該公司 2005 年 8 月到 2007 年 12 月相同類型的 29 個專案總共 362 筆 SPI 與 CPI 資料。首先初步的檢視資料發現有以下許多問題：大量的遺漏值、某些專案的 SPI 指標變異數為 0、某些專案後面的數據完全重複，與公司內的工程師溝通後確認原因主要有二:某些專案因為客戶方面的問題而未能繼續進行該專案，但因為還沒結案所以每個月指標值都沒變動；而某些專案執行時間超過預定成本原本預估的期限,造成SPI值是沒有定義的,因此之後的數值就不會再更新。刪除這些資料後，剩下 21 個專案共 256 筆觀測值。針對全部專案資料繪製盒鬚圖如圖 4.1 所示，SPI 有一個離群值，CPI 有七個離群值，由於只有少數離群值因此直接刪去即可。

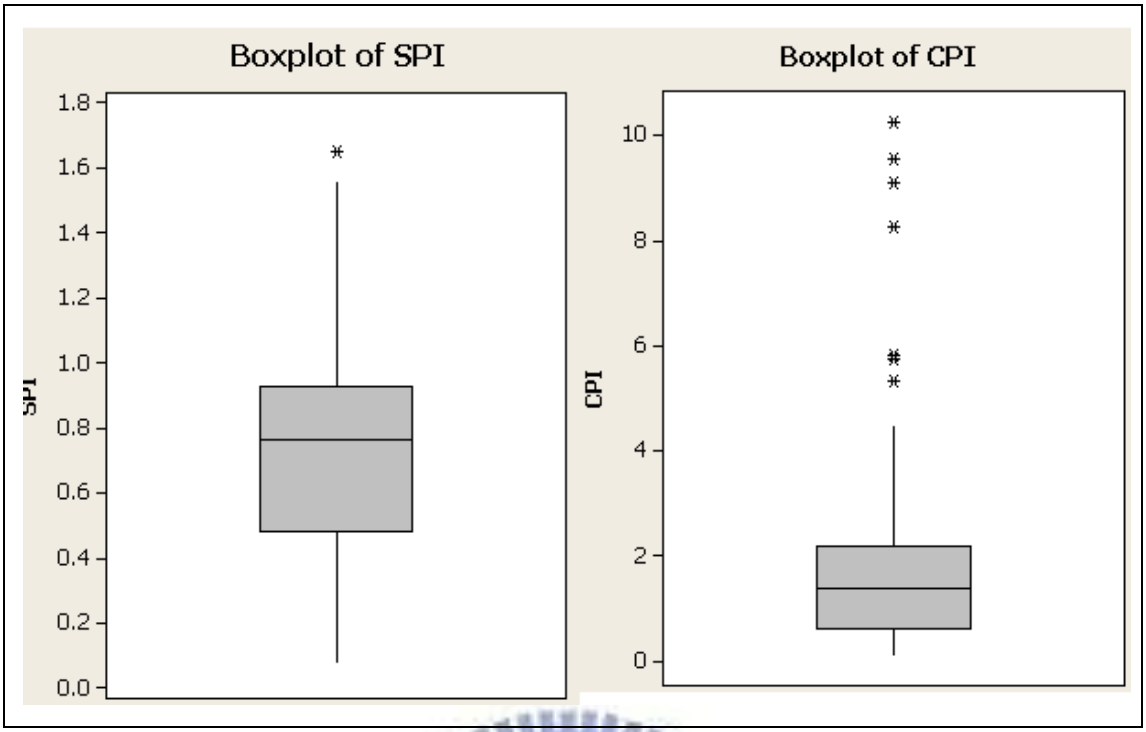

圖 4.1 全部資料的盒鬚圖

步驟 4：利用所有專案資料估計管制圖之參數如下

$$\mathbf{S} = \frac{1}{256-1}\sum_{l=1}^{21}\sum_{j=1}^{m_l}(\mathbf{x}_{lj} - \bar{\mathbf{x}})(\mathbf{x}_{lj} - \bar{\mathbf{x}})^{\mathrm{T}} = \begin{bmatrix} 0.0974 & 0.1262 \\ 0.1262 & 0.9293 \end{bmatrix}$$

$$\mathrm{UCL} = \frac{(256-1)^2}{256}\beta_{0.05}\left(\frac{2}{2}, \frac{256-2-1}{2}\right) = 5.9437$$

$$\mathrm{LCL} = 0$$

步驟 5：計算每個專案的每個觀測值的 Hotellint T^2值並繪製於步驟 5 所建立的管制界線中。

計算全部觀測值的T^2值結果如表4.3所示，並繪製管制圖如圖4.2所示。

專案代號	日期	T^2值	專案代號	日期	T^2值	專案代號	日期	T^2值
A	Aug-05	0.863	F	Mar-07	0.894	M	Nov-06	2.173
A	Sep-05	0.259	F	Apr-07	0.891	M	Dec-06	2.078
A	Oct-05	0.560	F	May-07	1.375	M	Jan-07	1.955
A	Nov-05	0.736	F	Jun-07	1.287	M	Mar-07	1.958
A	Dec-05	0.577	F	Jul-07	1.296	M	Apr-07	1.963
A	Jan-06	1.230	F	Aug-07	1.206	M	May-07	1.782

A	Feb-06	0.980	F	Sep-07	1.283	M	Jun-07	1.954
A	Mar-06	0.870	F	Oct-07	1.328	M	Jul-07	1.923
A	Apr-06	0.666	F	Nov-07	1.395	M	Aug-07	1.876
A	May-06	1.570	F	Dec-07	1.560	M	Sep-07	1.822
A	Jun-06	0.781	G	Jun-06	1.513	M	Oct-07	1.797
A	Jul-06	0.491	G	Jul-06	1.905	M	Nov-07	1.745
A	Aug-06	0.382	G	Aug-06	2.196	M	Dec-07	1.764
A	Sep-06	0.377	G	Sep-06	1.393	N	Jun-07	2.180
B	Aug-05	7.310	G	Oct-06	1.149	N	Jul-07	0.080
B	Sep-05	1.690	G	Nov-06	0.330	N	Aug-07	0.063
B	Oct-05	2.341	G	Dec-06	0.741	N	Sep-07	0.155
B	Nov-05	0.993	G	Jan-07	0.478	N	Oct-07	0.316
B	Dec-05	0.879	G	Mar-07	0.189	N	Nov-07	0.711
B	Jan-06	1.695	G	Apr-07	0.333	N	Dec-07	1.083
B	Feb-06	1.418	G	May-07	0.301	O	May-07	1.453
B	Mar-06	1.797	G	Jun-07	0.181	O	Jun-07	0.938
B	Apr-06	0.861	G	Jul-07	0.481	O	Jul-07	0.783
B	May-06	0.105	G	Aug-07	0.651	O	Aug-07	1.443
B	Jun-06	0.164	H	Nov-07	0.749	O	Sep-07	1.729
B	Jul-06	0.300	H	Dec-07	1.091	O	Oct-07	2.645
B	Aug-06	2.046	I	Jun-07	3.829	O	Nov-07	2.606
B	Sep-06	1.625	I	Jul-07	3.983	O	Dec-07	1.657
B	Oct-06	1.228	I	Aug-07	4.052	P	Aug-05	0.096
B	Nov-06	0.949	I	Sep-07	0.341	P	Sep-05	0.026
B	Dec-06	0.943	I	Oct-07	0.432	P	Oct-05	0.160
B	Jan-07	1.116	I	Nov-07	0.092	P	Nov-05	0.505
B	Mar-07	1.693	I	Dec-07	0.065	P	Dec-05	0.963
B	Apr-07	1.898	J	Apr-07	2.331	Q	Jun-06	16.336
B	May-07	7.847	J	May-07	5.471	Q	Jul-06	6.754
B	Jun-07	7.550	J	Jun-07	1.637	Q	Aug-06	6.593
B	Jul-07	7.042	J	Jul-07	1.597	Q	Sep-06	6.181
B	Aug-07	6.268	J	Aug-07	0.662	Q	Oct-06	3.581
B	Sep-07	5.735	J	Sep-07	0.466	Q	Nov-06	1.774
B	Oct-07	5.191	J	Oct-07	0.611	Q	Dec-06	2.163
B	Nov-07	4.787	J	Nov-07	0.869	Q	Jan-07	0.947
B	Dec-07	4.431	J	Dec-07	1.553	Q	Mar-07	0.217
C	May-07	5.874	K	Aug-05	2.521	Q	Apr-07	0.153

C	Aug-07	12.716	K	Sep-05	2.047	Q	May-07	0.961
C	Sep-07	8.767	K	Oct-05	1.730	Q	Jul-07	1.171
C	Oct-07	7.251	K	Nov-05	1.610	Q	Aug-07	1.637
C	Nov-07	6.320	K	Dec-05	1.221	Q	Sep-07	1.736
C	Dec-07	5.775	L	Dec-05	1.318	Q	Oct-07	1.748
D	Jun-07	15.296	L	Jan-06	4.264	Q	Nov-07	1.795
D	Jul-07	0.601	L	Feb-06	4.543	Q	Dec-07	1.811
D	Aug-07	0.771	L	Mar-06	4.842	R	Mar-07	3.587
D	Sep-07	1.183	L	Apr-06	5.087	R	Apr-07	3.592
D	Oct-07	0.532	L	May-06	7.063	R	May-07	3.177
D	Nov-07	0.627	L	Jun-06	5.105	R	Jun-07	2.323
D	Dec-07	0.613	L	Jul-06	2.858	R	Jul-07	0.714
E	Jan-06	4.796	L	Aug-06	1.338	R	Aug-07	0.346
E	Feb-06	3.911	L	Sep-06	0.661	R	Sep-07	0.417
E	Mar-06	0.371	L	Oct-06	1.640	R	Oct-07	0.354
E	Apr-06	0.182	L	Nov-06	0.731	R	Nov-07	0.338
E	May-06	2.450	L	Dec-06	0.973	R	Dec-07	0.199
E	Jun-06	1.406	L	Jan-07	0.939	S	May-06	1.408
E	Jul-06	0.774	L	Mar-07	0.702	S	Jun-06	2.192
E	Aug-06	0.472	L	Apr-07	0.627	S	Jul-06	2.890
E	Sep-06	0.100	L	May-07	0.688	T	Aug-05	1.270
E	Oct-06	0.878	L	Jun-07	0.562	T	Sep-05	3.364
E	Nov-06	1.187	L	Jul-07	0.588	T	Oct-05	1.405
E	Dec-06	0.939	L	Aug-07	0.513	T	Nov-05	1.847
E	Jan-07	1.130	L	Sep-07	0.796	T	Dec-05	1.947
E	Mar-07	1.756	L	Oct-07	0.718	T	Jan-06	2.040
F	Sep-05	2.253	L	Nov-07	0.477	T	Feb-06	2.226
F	Oct-05	2.295	L	Dec-07	0.378	T	Mar-06	2.968
F	Nov-05	0.739	M	Aug-05	4.498	T	Apr-06	3.201
F	Dec-05	0.782	M	Sep-05	5.223	T	May-06	5.352
F	Jan-06	2.643	M	Oct-05	0.735	T	Jun-06	1.611
F	Feb-06	2.550	M	Nov-05	0.088	T	Jul-06	1.701
F	Mar-06	2.963	M	Dec-05	0.038	T	Aug-06	2.419
F	Apr-06	2.778	M	Jan-06	0.032	U	May-07	3.814
F	May-06	2.656	M	Feb-06	0.025	U	Jun-07	1.873
F	Jun-06	2.785	M	Mar-06	0.019	U	Jul-07	2.707
F	Jul-06	3.200	M	Apr-06	0.465	U	Aug-07	1.839

F	Aug-06	2.017	M	May-06	0.740	U	Sep-07	1.643
F	Sep-06	0.958	M	Jun-06	0.782	U	Oct-07	2.257
F	Oct-06	1.173	M	Jul-06	1.134	U	Nov-07	1.925
F	Nov-06	1.361	M	Aug-06	1.602	U	Dec-07	2.294
F	Dec-06	1.235	M	Sep-06	1.203			
F	Jan-07	0.957	M	Oct-06	1.351			
臨界值		$\mathrm{UCL}=\frac{(\sum_{l=1}^{q} m_l-1)^2}{\sum_{l=1}^{q} m_l}\beta_\alpha(\frac{p}{2},\frac{\sum_{l=1}^{q} m_l-p-1}{2})$=5.9437						

表 4.1 全部專案的 Hotellint T^2值

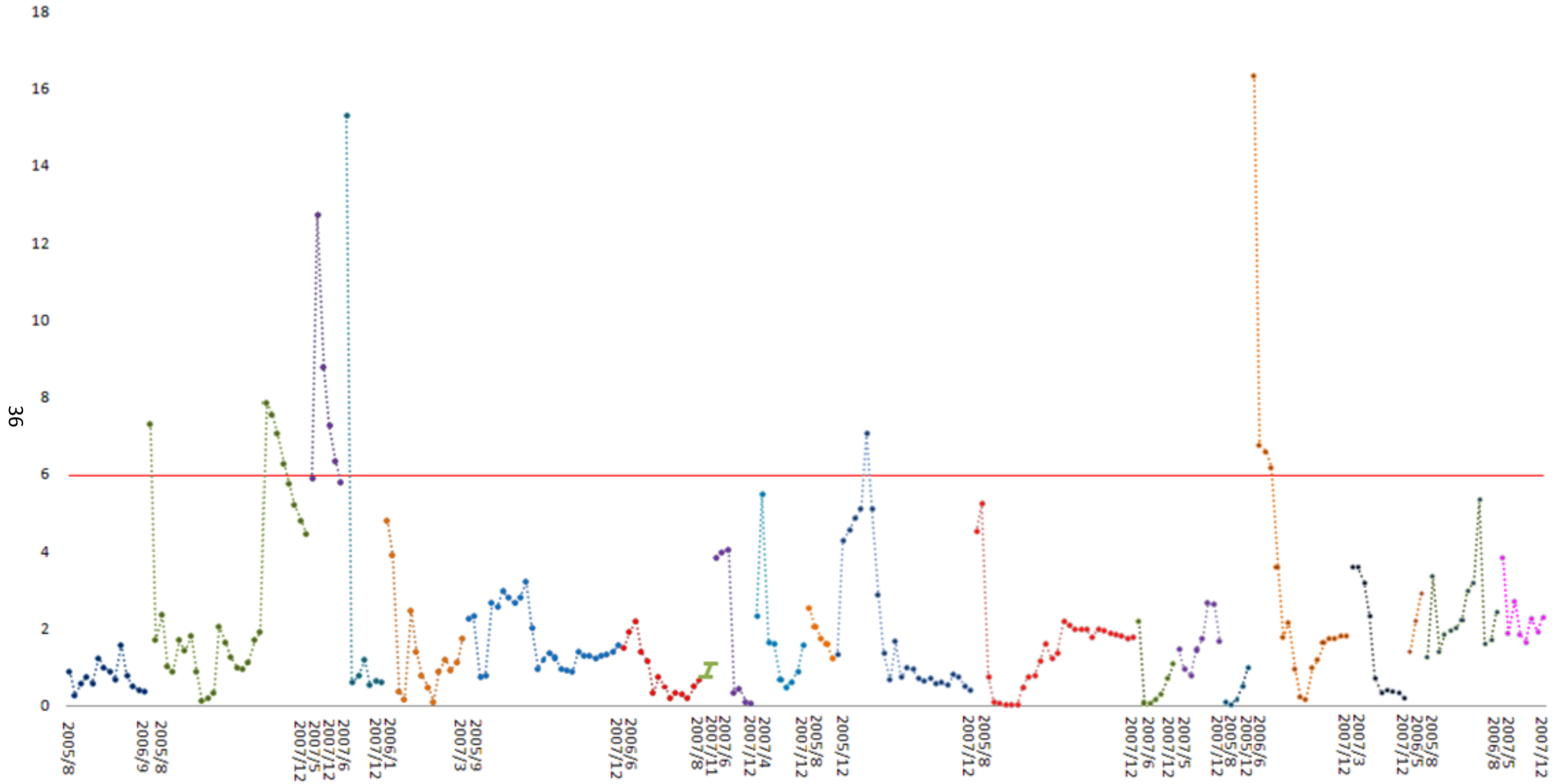

圖 4.2 全部專案資料的管制圖(橫軸只標示出專案開始時間與結束時間)

步驟 6：針對異常點進行分析。

將超過管制上界的觀測值之分解T^2統計量整理於表4.2 所示。

專案代號	日期	T^2	d_{SPI}	d_{CPI}	SPI	CPI
B	Aug-05	7.310	5.377	4.209	1.144	0.295
B	May-07	7.847	-1.795	7.025	0.931	4.132
B	Jun-07	7.550	-1.760	6.710	0.934	4.086
B	Jul-07	7.042	-1.698	6.160	0.939	4.005
B	Aug-07	6.268	-1.560	5.386	0.939	3.870
C	May-07	12.716	0.578	1.103	0.311	2.546
C	Aug-07	8.767	-0.278	1.519	1.555	4.456
C	Sep-07	7.251	-1.330	2.905	1.377	4.049
C	Oct-07	6.320	-1.531	3.404	1.225	3.982
D	Jun-07	15.296	8.158	11.788	0.238	3.762
L	May-06	7.063	3.590	-1.636	1.441	3.072
Q	Jun-06	16.336	4.115	15.387	0.462	4.466
Q	Jul-06	6.754	3.947	4.918	0.367	2.912
Q	Aug-06	6.593	3.917	4.744	0.366	2.878
Q	Sep-06	6.181	4.791	3.263	0.279	2.488
臨界值	$\chi^2_{0.05}(1) = 3.84$					

表4.2 失控值之表

經過與公司內部的工程師溝通之後，確認以下的失控原因：由於B專案是使用.NET平台開發而非以前的Java平台，成本就難以估計，且此專案在開始前的資料蒐集不夠完整，此在估計專案預算時有高估的情形。而專案C與專案Q則是屬於設備案，設備案只需要修改原有的package而不用從無到有開發出整套系統，且因為之前的package需要做多少幅度的修改就難估計，因此高估了成本。

重複步驟3到步驟6直到沒有失控值為止，最後會剩下共下224筆穩定的觀測值。

步驟7：計算流程能力基準並持續監控組織所有同類型的專案。

計算流程能力基準如下：

$$\text{UCL} = \frac{2(224+1)(224-1)}{224^2 - 224p} F_{0.05}(2,224-2) = 6.1276$$

$$\text{LCL} = 0$$

利用以下公式計算未來同類型專案的 Hotellint T^2值，並繪入流程能力基準中。

$$T^2 = \left(\mathbf{x} - \begin{bmatrix}0.6832\\1.2514\end{bmatrix}\right)^{\mathrm{T}} \begin{bmatrix}0.0651 & 0.0923\\0.0923 & 0.4818\end{bmatrix}^{-1} \left(\mathbf{x} - \begin{bmatrix}0.6832\\1.2514\end{bmatrix}\right)$$

4.2 使用模擬資料測試本研究方法所得之流程能力基準

本節利用模擬案例來說明本研究方法所得之流程能力基準可以準確偵測到失控的情形。本研究使用 Minitab 內建的亂數產生器來模擬具有相關性的兩個軟體指標數據。

在模擬情境方面，總共模擬 25 筆觀測值，其中第 16 個到第 20 觀測值的 SPI 指標偏移-1.5 標準差，第 21 個到第 25 觀測值的 CPI 指標偏移+1.5 標準差。產生表 4.4 的資料。

編號	SPI	CPI	編號	SPI	CPI	編號	SPI	CPI
1	0.403	1.275	10	0.862	1.710	19	0.571	1.459
2	0.826	0.655	11	0.907	1.575	20	0.016	1.584
3	0.973	2.083	12	0.697	0.720	21	0.457	2.538
4	0.845	1.656	13	0.991	2.797	22	1.141	3.404
5	0.693	1.978	14	0.534	0.208	23	1.202	3.030
6	0.657	1.098	15	0.869	1.361	24	0.836	3.169
7	0.626	0.667	16	0.308	1.977	25	0.802	2.814
8	0.457	1.113	17	0.311	2.019			
9	0.681	1.425	18	0.164	0.859			

表 4.3 模擬數據

由圖 4.5 之管制圖可以發現，當失控的情況發生時馬上就偵測出偏移，且從分解 T^2統計量也找出失控之原因，證明本論文所找到之流程能力基準是有效的。

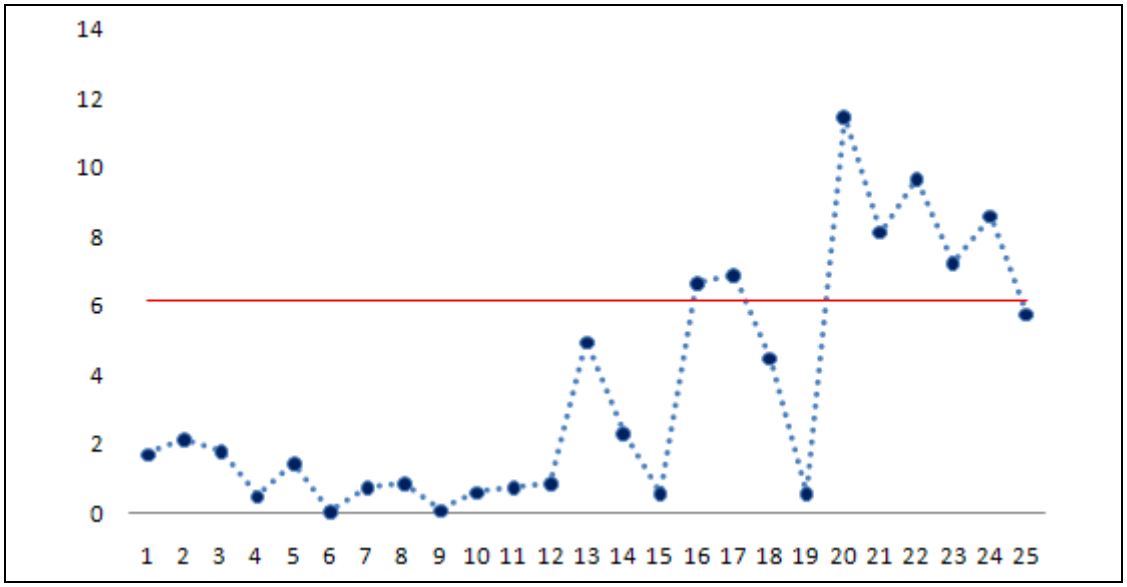

圖 4.3 模擬數據之多變量管制圖

編號	T2	d(SPI)值	d(CPI)值
16	6.669	5.168	3.701
17	6.902	5.224	3.983
20	11.477	11.162	2.105
21	8.139	3.422	7.064
22	9.666	-3.541	5.241
23	7.233	-1.779	1.564
24	8.602	-1.876	8.108

表4.4 模擬資料的失控點

4.3 與單變量管制圖之比較

就所有專案分別繪製SPI與CPI指標的變動樣本大小的$\bar{X}-S$管制圖，且以2倍樣本標準差為管制界線，如圖4.3與圖4.4所示，只有第6、7、8個觀測值(2006年1、2、3月)有偵測到變異，本研究的多變量管制圖並未偵測到此失控值，而本研究所偵測到之失控點單變量管制圖也未偵測到，原因是因為SPI與CPI指標的相關係數為0.465且顯著水準為0.00使用傳統單變量管制圖會有誤判的情形，在檢視過SPI與CPI的原始資料後並且與工程師確認後發現此三個月份事實上並無失控。

日期	SPI	CPI	日期	SPI	CPI	日期	SPI	CPI
Jan-06	0.674	2.375	Feb-06	0.652	2.227	Mar-06	0.668	2.211
Jan-06	0.723	1.652	Feb-06	0.665	1.474	Mar-06	0.674	1.502
Jan-06	0.349	0.650	Feb-06	0.412	0.519	Mar-06	0.339	0.616
Jan-06	0.800	0.449	Feb-06	0.800	0.392	Mar-06	0.896	0.434
Jan-06	0.254	0.519	Feb-06	0.258	0.622	Mar-06	0.223	0.545
Jan-06	0.132	0.267	Feb-06	0.115	0.197	Mar-06	0.095	0.170
Jan-06	0.160	1.493	Feb-06	0.160	0.262	Mar-06	0.533	1.213

表4.5 2006年1、2、3月之SPI與CPI原始資料

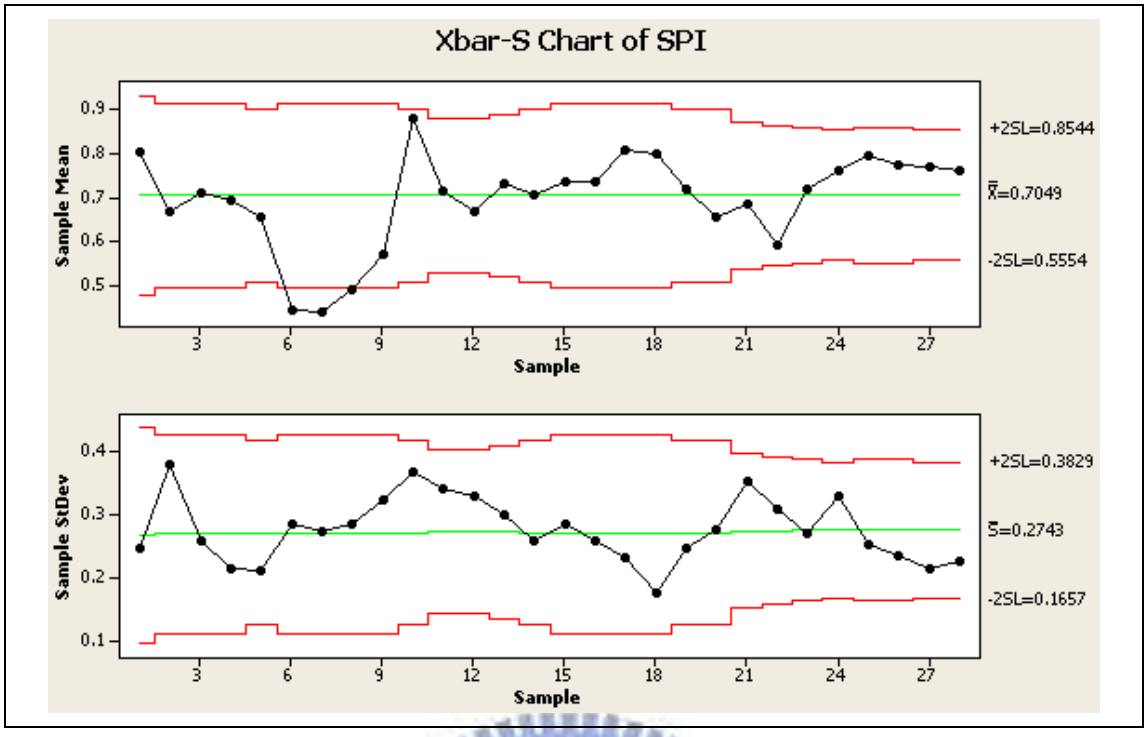

圖4.4 全部專案的SPI指標的變動樣本大小的平均值與標準差管制圖

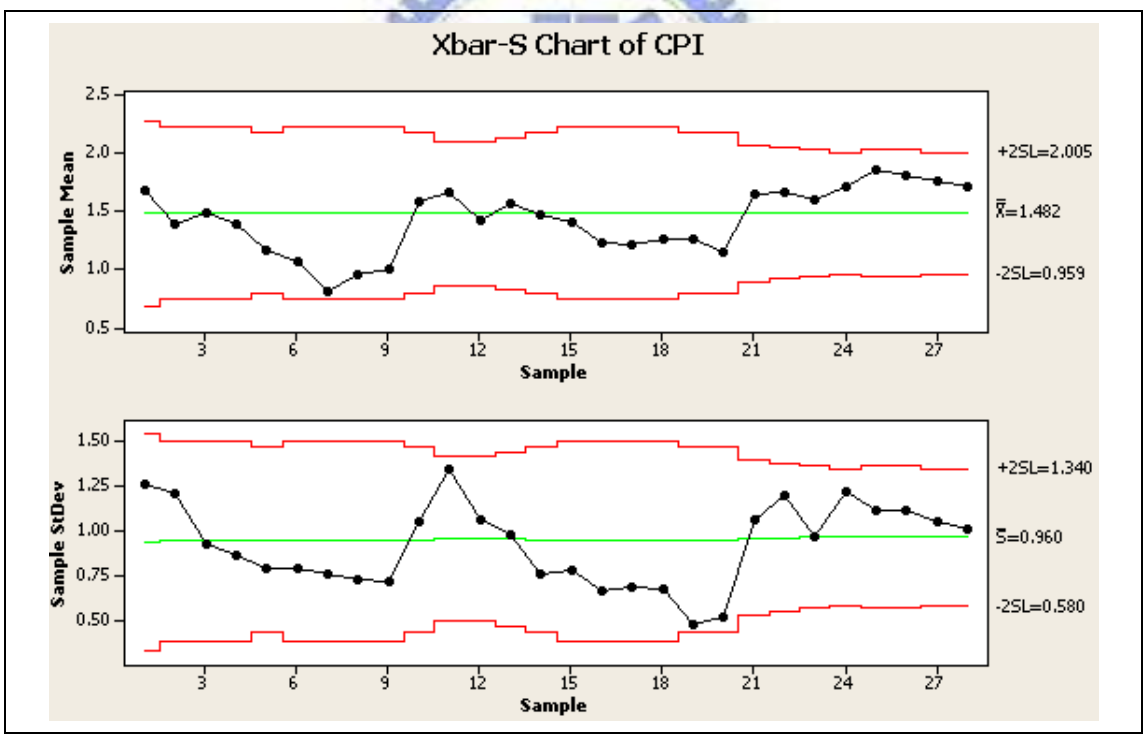

圖4.5 全部專案的SPI指標的變動樣本大小的平均值與標準差管制圖

第五章　結論與建議

以下兩小節分別為本研究之結論與建議。

5.1 結論

本論文的結論有兩部分，首先為實例之結論：

1. 目前所建立之流程能力基準必須每年再做修正。
2. 資料的「品質」比資料的「數量」重要，在資料的初步分析可以發現許多觀測值是非常的不合理的，公司內部應該檢討在資料蒐集的過程中是否有缺失需要改進。

第二部分為本研究的管制系統之結論：

1. 本研究提供一個適用於軟體產業的資料型的管制流程，此流程與一般管制圖不同的地方為利用步驟 4 將每個專案資料重新排列。
2. 步驟 4 的方法亦可以套用在單變量的管制圖，以軟體業最常用的 *X-Rm* 管制圖為例，計算全部每個專案內的 MR，則$\hat{\sigma} = \frac{\overline{\text{MR}}}{d_2} = \frac{\frac{\sum_{l=1}^{q}\sum_{j=2}^{m_l}\left(x_{lj} - x_{l(j-1)}\right)}{\sum_{l=1}^{q} m_l}}{d_2}$ (自由度為$\sum_{l=1}^{q}(m_l - 1)$，$\bar{\text{x}} = \frac{\sum_{l=1}^{q}\sum_{j=1}^{m_l} x_{lj}}{\sum_{l=1}^{q} m_l}$。
3. 由於目前台灣的軟體公司資料蒐集完整，甚至資料蒐集的程序可能是錯誤的使得資料是無意義的，因此花大量時間於步驟 3 的資料處理是必須的。

5.2 建議

本研究未來研究方向有以下兩點：

1. 由於軟體產業的工程師對管制圖並不熟悉，本研究的管制圖撰寫程套裝軟體會使得無統計背景知識的工程人員較能輕易的使用。
2. 利用管制圖的統計經濟性設計來改善本研究的管制界線。

參考文獻

[1] 吳炤華，2000，「積體電路生產線上結合缺陷數與缺陷群聚之 Hotelling T^2多變量管制圖」，國立交通大工業工程與管理學系，碩士論文。

[2] 黃志力，2001，「積體電路生產線上結合缺陷數與群眾指標之 Hotelling T^2多變量管制圖」，國立交通大工業工程與管理學系，碩士論文。

[3] Baker E. R.，"Which Way, SQA"，*IEEE SOFTWARE*，January/February (2001)

[4] Brooks F. P., F.P. 1987. No Silver Bullet: Essence and Accidents of Software Engineering, *IEEE Computer Magazine*.

[5] Caivano D., "Continuous Software Improvement through Statistical Process Control," pp. 288-293, Proc. of the 9th European Conference on Maintenance and Reengineering 2005.

[6] Cngussu J.W., R.A. DeCarlo, A.P. Mathur, "Monitoring the Software Test Process Using Statistical Process Control: A Logarithmic Approach," Proc. of the 9th European software engineering conference, pp. 253-265, Sept.,2003.

[7] Florac W.A., A. D. Carleton, and J. Barnard, "Statistical process control: Analyzing a space shuttle onboard software process," *IEEE Software*, vol. 17, pp. 97–106, July/August 2000

[8] Florac W.A. and A.D. Carleton, *Measuring the Software Process: Statistical Process Control for Software Process Improvement*, Addison-Wesley, 1999.

[9] Investigating Suitability of Software Process and Metrics for Statistical Process Control

[10] Jacob A., S.K. Pillai, (2003), Statistical Process Control to Improve Coding and

Code Review, IEEE Software, May/June, 50-55

[11] Jalote P. and A. Saxena 2002. Optimum Control Limits for Employing Statistical Process Control in Software Process, *IEEE Transactions on Software Engineering* 28(12): 1126–1134.

[12] Jalote P., "Use of Metrics in High Maturity Organizations" *Proc. Software Eng. Process Group Conf. (SEPG '00)*, Mar. 2002.

[13] Kan S.H. 1995. *Metrics and Models in Software Quality Engineering*. Addison-Wesley Publishing Company. ISBN 0-201-63339-6.

[14] Komuro M., "Experiences of applying SPC techniques to software development processes," in *ICSE '06: Proceeding of the 28th international conference on Software engineering*. New York: ACM Press, 2006, pp.577–584.

[15] Lantzy M.A., " Application of statistical process control to the software process," pp. 113 – 123, Proc. of the ninth Washington Ada symposium, 1992.

[16] Manlove D.; Stephen H Kan, "Practical Statistical Process Control for Software Metrics," Software Quality Professional; Sep 2007; 9, 4; ProQuest Computingpg. 15

[17] Montgomery D.C., "Introduction to Statistical Quality Control," John Wiley & Sons Inc. 2005, pp. 486–522.

[18] Paulk M. C., "Practices of high maturity organizations", *1999 SEPG Conference*, Atlanta, Georgia, March 1999.

[19] Paulk M. C. and Chrissis, M.B. 2002. *The 2001 High MaturityWorkshop, (CMU/SEI 2001-SR-014)*, Carnegie Mellon University.

[20] Sargut K. U., O. Demirörs, 2006.“Utilization of statistical process control (SPC) in emergent software organizations: pitfalls and suggestions,”……

[21] Tracy N. D., J. C. Young, and R. L. Mason, 1992.“Multivariate Control Charts for Individual Observations,” *Journal of Quality Technology*, Vol. 24(2), pp. 88-95.

[22] Wang Q., N. Jiang, L. Gou, X. Liu, M. Li, and Y. Wang, “BSR: a statistic-based approach for establishing and refining software process performance baseline,” in *ICSE ’06: Proceeding of the 28th international conference on Software engineering*. New York: ACM Press, 2006, pp.585-594.

[23] Weller E. F. 2000. Practical Applications of Statistical Process Control, *IEEE Software*, pp. 48–55.

[24] 網站資料，http://www.cmmi-taiwan.org.tw/

[25] 網站資料，http://www.microsoft.com/taiwan/msclub/events/vsnet/default.htm