國立交通大學

資訊學院　資訊學程

碩 士 論 文

以累積經驗知識為目的之問題追蹤系統

Issue Tracking System for Knowledge Accumulation

研 究 生：李勝斌

指導教授：黃世昆 教授

中 華 民 國 一百零一 年 六 月

以累積經驗知識為目的之問題追蹤系統

Issue Tracking System for Knowledge Accumulation

研 究 生：李勝斌　　Student：Sheng-Pin Lee

指導教授：黃世昆　　Advisor：Shih-Kun Huang

國 立 交 通 大 學

資訊學院　資訊學程

碩 士 論 文

A Thesis

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

June 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年六月

以累積經驗知識為目的之問題追踪系統

學生：李勝斌　　指導教授：黃世昆

國立交通大學　資訊學院　資訊學程　碩士班

摘　要

生產問題的控管，一直是現代產業的重要課題，在問題發生時所產生的便是損失，不論是有形的金錢或是無形的商譽，因此有效率的問題控管、追踪、處理便能有效地將損失降低。但另一方面，問題的解決也代表一個新的知識的產生，以目前的科技產業而言，同產業的競爭十分激烈，能力較好的人才往往是挖角的對象，如果企業內部對於自身的企業知識沒有良好的記錄，一旦人員流失，很容易造成知識的斷層，因此如何能有效地將經驗及知識記錄、整理、運用也成了十分重要的課題。

本論文研製之目的在於透過整合、修改軟體缺陷追踪系統(Bug Tracking System)、維基系統(Wiki)、行動裝置成為生產問題追踪系統，將生產上的問題有效率地追踪、記錄下來，並且易於將知識整理、運用，讓問題在發生時能夠有適當的參考，且於人員異動時，不會產生過大的知識斷層。另外透過改良統計製程管制 SPC(Statistical Process Control) 中的指標，建立問題處理能力指標，該指標可以找出問題處理時的瓶頸，改善人員處理問題的效率，減少企業因生產問題的損失。

Issue Tracking System for Knowledge Accumulation

Student：Sheng-Pin Lee　　Advisors：Dr. Shih-Kun Huang

Degree Program of Computer Science
National Chiao Tung University

ABSTRACT

Issue tracking is an important process in enterprise applications. It usually causes loss of money and reputations when issue occurs. If issue can be better controlled, solved, and tracked, the loss will be reduced. Nowadays, there is high job-change rate in high-tech industry because of competitions and human resource needs between corporations. Once the employee was off the job, the knowledge and experience won't be preserved. It is therefore an important consideration to store, organize, and reuse knowledge of former employees.

This research integrates the bug tracking system, wiki system, and mobile devices into a issue tracking system. This system is not only tracking and controlling issues efficiently but also easy to organize experiences into knowledge. This system will provide good reference data when the issues are resolved. On the other hand, the knowledge will still be preserved when the employee leaves the job because of the experience has been stored properly. This research also modified the SPC (Statistical Process Control) index to measure the issue solving capability. This index will help figure out the bottleneck of issue resolving process and reduce the cost.

誌　　謝

三年的研究所生活終於到了結束的時候，回想過去三年，發生了不少事，在這段期間第一個要感謝的是我的指導教授黃世昆教授，不論是在論文的瓶頸，還是工作上的不順遂，都給了我不少寶貴的意見，讓我受益良多，非常感謝教授在這段日子的熱心教導及鼓勵。再來要感謝的是口試委員孔崇旭教授及宋定懿教授，在口試時給予的意見與指導，讓我的論文更加完整。

最後要感謝的是我的家人，在這段期間的鼓勵與諒解，讓我可以無後顧之憂，順利地完成研究所的學業。三年的研究所生活，說長不長，在這三年結束後，人生也將邁入另一個階段，最後再次感謝這些我生命中的貴人。

勝斌 2012/6/30

目　　錄

表　目　錄

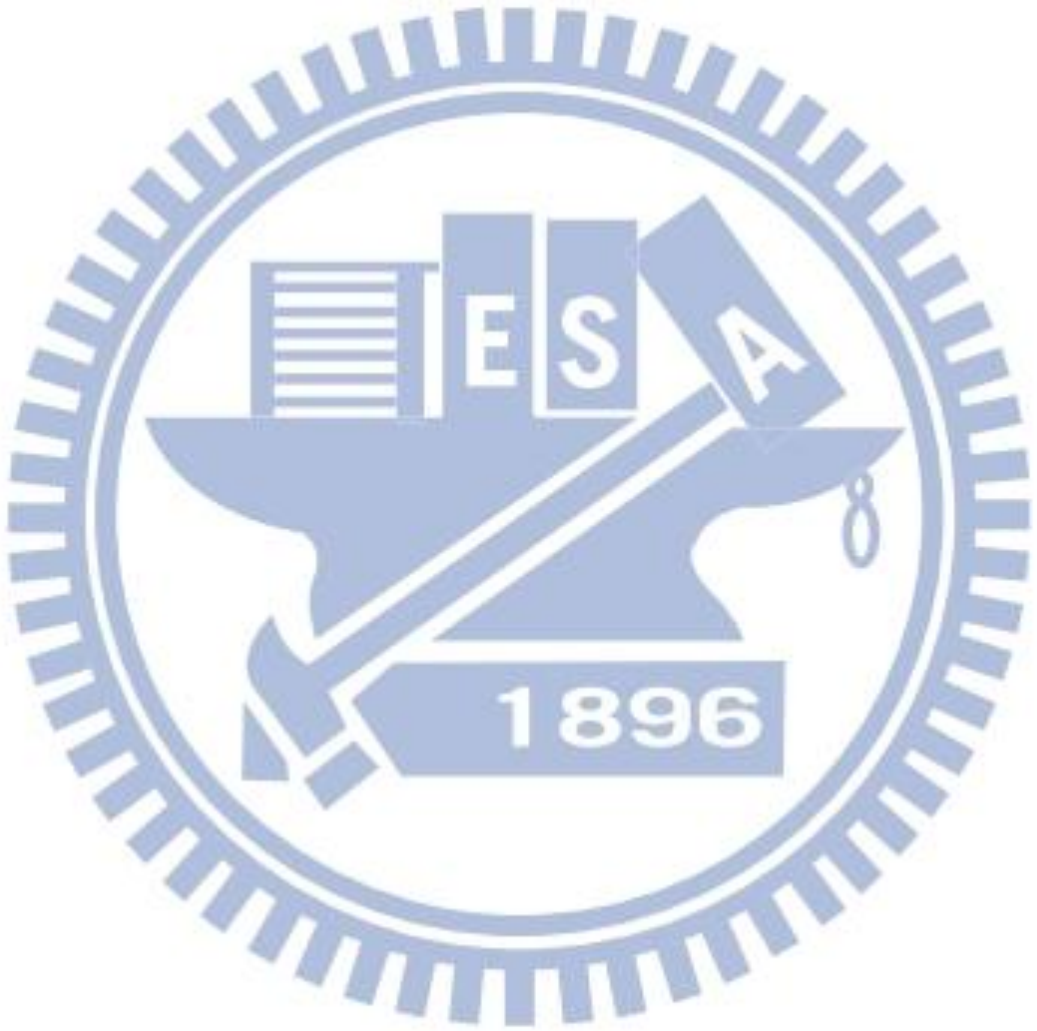


圖　　目　　錄

一、緒論

1.1 研究背景與動機

目前的科技產業，不論是設計或是代工，對於生產的硬體的投資通常十分充足。因為對於生產能力而言，硬體的影響是十分顯著而明顯的，相較於硬體而言，在生產過程仍有許多非硬體的角色，例如：操作人員、相關機台之參數設定、環境因素…等，於生產的過程中相輔相成，缺一不可。

硬體可以可透過修改、更換來提昇生產能力或適用性，只要有一組機組修正，其它的機組亦可以相同的方式而達到生產能力的提昇。而人員則是透過經驗的吸收來提昇價值，但是這樣的能力往往無法很有效的複製、傳承，且對於所謂的經驗沒有一個比較客觀的指標來衡量。

知識與經驗不同的地方在於知識是有系統的，易於了解吸收，但是經驗的吸收就比較不容易，因為經驗本身較無系統，通常是透過長時間累積而成，不容易有較具結構的文件可以參考，使得經驗的傳承相對於知識是困難的，且同一領域的經驗，在不同的企業往往不是全盤皆適用，因為不同的企業，作法不一定相同，這點與對錯無關，很多時候是習慣問題，儘管如此，如果能快速吸收他人經驗，則可以提昇問題處理的效率。

所以一旦遇到人員的異動，往往會造成企業內部知識及經驗上的斷層，這點在科技產業的影響尤其明顯。以現代的生產而言，處理問題的效率還可能影響企業的獲利，例如：停工損失、延遲出貨的賠償…問題等，因此如將經驗轉化為知識並記錄下來，成為企業知識，將可避免這些潛在的損失。

工業生產的過程，一般而言與軟體的生產過程概念上是有相通之處。同樣的投入資源、生產、修正錯誤…等過程，而修正錯誤的過程，正好就是最佳的經驗累積過程及知識的來源。軟體發展可藉由錯誤追踪系統來修正、控制錯誤，若工業生產亦可使用類似系統，將生產過程中所產生的問題、修正過程、解決方案記錄下來，並且轉化成知識，將可減少人員的訓練成本、提昇生產能力，進而產生有價值的企業知識，減少因人員異動所造成的傷害。

另外若能提供一個較為客觀的指標，除了可以作為工作團隊的能力依據，指出團隊能力不足之處，還可作為改善的方向，持續且有系統的提昇團隊的能力。

1.2 研究目的

在科技產業中，生產過程所產生的錯誤或問題，在時程上通常有極高的急迫性，時間的容忍度上比較低，因此需要系統支援，將錯誤的排除過程，能在適當時程內控管，可找出重覆發生的問題並改善。若能善用這些「錯誤排除」過程的知識，將可提昇對於類似問題時的反應能力及處理參考，進而提昇產能，另外提供一個指標可作為工作團隊改善的方向、能力的評量等。

本研究的目的在於以產生知識庫為目的，利用問題管理之概念，並配合軟體開發中常用的錯誤追踪系統(Bug Tracking System)，建立一個生產問題追踪系統，並且在未來對於這些資料，可以方便整理成有結構的知識彈性再利用，將之應用於工業生產的過程，記錄生產過程中所產生的問題及其解決方案，於日後作為知識之傳承及決策參考。

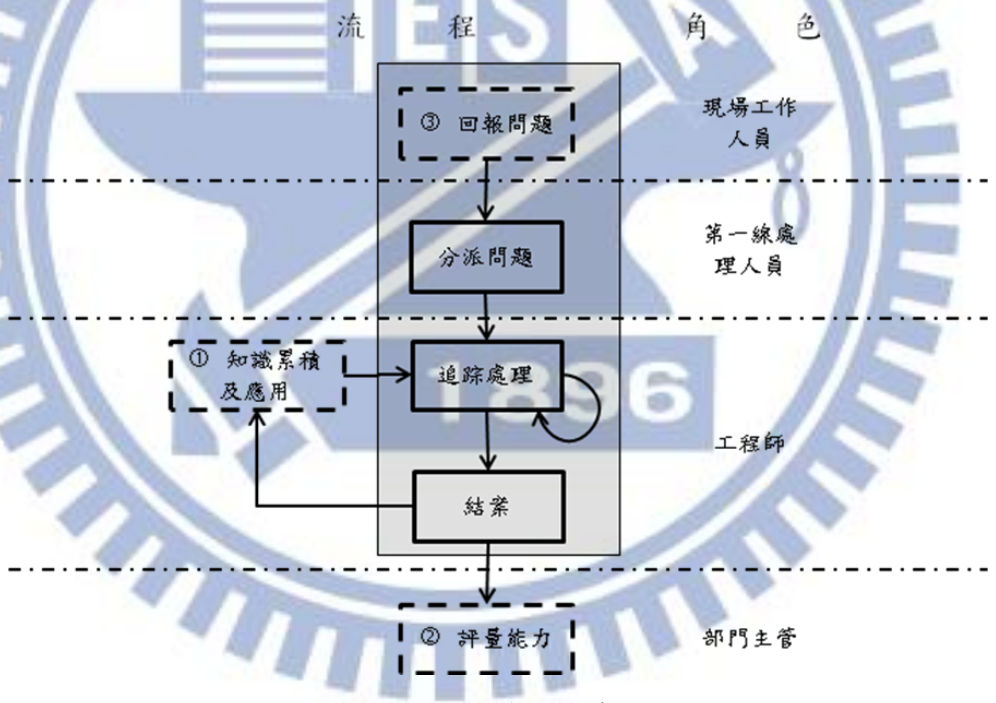


圖 1：問題追踪流程及角色

一般問題追踪系統，主要的功能為圖 1 灰色部份，主要流程為問題回報、分派問題、持續追踪處理及結案，以問題追踪的角度而言，這樣的生命週期已經足夠，雖然這些資料可以累積出不少的知識，但因為這些問題追踪時所留下來的資料，較缺乏結構，且資訊可能正確與不正確的交互參雜，也可能因為時間點的不同，解決的方法也不見得一樣，所以在日後參考時有一定的難度，如果能利用這些問題追踪資訊，透過較好的工具將這些資料整理起來，妥善的整理成為其它系統的資料來

源，將可節省不少的時間及功夫，所以將累積知識的動作移出問題追踪系統，由另一個系統作整理，而形成問題追踪、結案、累積知識與應用這個生命週期，對於相同的問題不斷地補充相關的知識，進而達到累積知識的目地。且工程師大都忙於處理各式各樣的問題，一遇到要撰寫技術文件時，通常沒時間也不知道要寫什麼，若能在處理問題時，將資料順手完成，那麼對於他的工作能力與知識的累積將有很大的幫助，這是第一個目標（圖 1－①）。

在問題追踪的過程中，系統通常會記錄下問題處理的時間、同類別問題發生的次數、問題轉手次數、問題的狀態…等資訊，這些資訊都有助於評估一個工作團隊的能力，除了一些常用的分類統計資料外，若可以利用指標來評量，找出問題處理時的瓶頸，針對這些瓶頸採行有效的管理，將可提昇團隊的能力，這是第二個目標（圖 1－②）。

另一方面，由於場地、工作性質的關係〔1〕，很多時候在現場並無法使用電腦，必須於事後記錄，使得資料失去即時性，另外很多時候使用文字描述問題，不如使用一張圖片、一段錄音或一段錄影來得清楚明白，但是這類的資料通常得在事後利用附件的方式附上，這些不方便的情況，將使得整個系統的可用性上受到很大限制。所以針對這點，將利用行動裝置的多媒體能力來消除這些限制，這是第三個目標（圖 1－③）。

以上三個目標是本系統所希望達成的，因為一般的問題追踪系統基本的功能已相當不錯且十分容易了解，所以本論文不針對問題追踪系統本身加以研究，而直接採用現有的系統，透過加強系統來達成以上三個目標。不過為了避免此工具成為壓迫員工的工具，須注意不應將此系統作為績效評核之標準，因為若作為績效評核之標準，將會使得人員因為績效壓力而避免使用此系統，採取避重就輕的作法，這樣就無法將實際問題反映出來，使得系統失去本意，達不到預期的效果。

1.3 研究範圍

研究範圍，將依據上節所提到的，將研究主題分成三大部份：問題回報的方式、能力的評量、知識的理及應用。

問題的回報上，將比較傳統使用電腦記錄問題方法及行動裝置優缺點，另一方面因為近年來行動裝置的普及，系統種類也很多，因此行動裝置在導入企業內部之前，勢必有許多必須考量的問題，其中包括了系統的選擇、導入企業時所需注意的問題，皆屬此列。

能力的評量方面，一般在問題追踪系統都有一些統計資料，但是這類的資料比較無法看出一些隱藏的問題，所以需要找出一個合適的指標計算方式，在本論文中將使用統計製程管理(SPC)的概念來作為評量能力的指標，並作適當的改寫，用來評估能力，但此指標並非用來精確計算品質，主要目的在於與其它時間作比較找出處理能力的變化，並提供適當的線索。

最後知識的撰寫及應用上，是透過常用的維基系統，整合問題追踪系統，並以問題追踪系統的資料作為撰寫的來源，由於問題追踪統的資訊本身是較無結構的，且追踪的過程中，有效與無效的解決方式相互夾雜，在閱讀上比較不容易，所以透過問題處理人員，於問題處理完成後，可以依其專業的知識，將問題整理成較好閱讀的文件。在知識的應用上，希望能在撰寫知識時能一次到位，撰寫的同時順便作知識應用的前置處理，而不必再多維護其它的系統，主要想法來自 Javadoc 的文件觀念，於維基系統中建立新的標記格式，於未來可以透過抽取的方式，將資料抽取出來，其應用可以是技術文件、教育訓練、成果測驗…等，而不用擔心其它的系統的同步更新問題，因為只要透過重新抽取便可以得到最新的資料。

最後這些資訊可以在日後作為問題發生時的第一線參考資料，用以輔助團隊作為第一時間處理時的參考，或者用以避免更嚴重的後續問題的發生，並且持續的改善團隊的能力。

1.4 論文架構

本論文的架構為：

1. 緒論：說明研究動機、提出所要解決的問題及研究方向。
2. 相關研究：依照研究範圍中的使用場合，研究及比較相關的理論與工具。
3. 設計與方法：依照相關研究所提的方向，設計系統及流程。
4. 系統實作：依設計與方法中所選擇的系統、方法，實作出系統功能。
5. 結果分析：比較與系統優勢。
6. 結論及建議：系統未來加強的方向。

二、相關研究

相關的研究上，會以研究目的中三個重點：問題回報、問題處理能力評量、知識的累積與應用上，分別討論在系統的各階段中所需要解決的問題或者系統的選擇。在這之前先對於所謂的問題(Issue)作一個定義，一般而言，知識是透過解決問題的過程產生，其產生的過程如下圖：

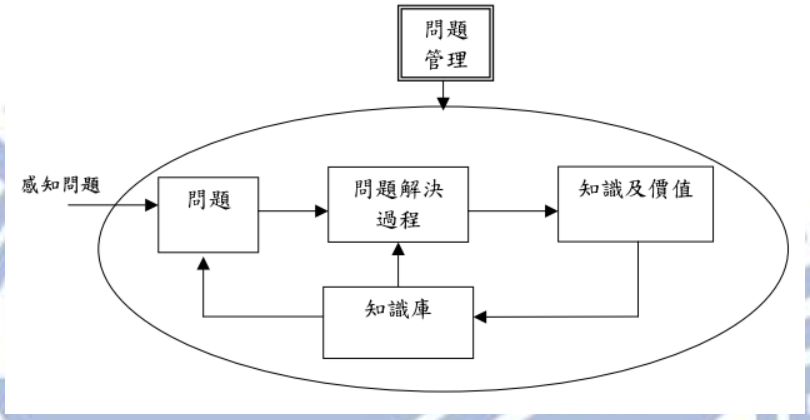


圖 2：問題管理流程

在問題管理中所指的問題，是英文中的“Problem”〔2〕，除了指的是尚未解決的課題之外，亦指「難解之題」、「不可解之事物」，與 Issue 的定義有所差異，問題管理系統，所重視的除了解決問題外，尚注重問題構成的過程、分類等過程及管理問題，重點上與 Issue 比較不同，Issue 屬於已知類型的問題，對於問題的構成、分類較無疑義，不過雖然重點不同，其流程大至上是相同，所以取其流程作為參考。

由上圖 2 可看出，於問題產生時，便進入了問題管理的過程，問題管理的最後產出，便是知識與經驗，其經驗便可於日後相類似的狀況再發生時作為參考，並且再進入新的知識累積過程〔3〕。

一般問題管理系統，在不同的領域之下，其著重的重點也不相同，如客服系統，其著重的是減輕客服人員的負擔，當客戶問題出現的時候，客戶可以透過系統的 FAQ 先找看看是否有答案，若沒有再回報給客服人員，客服人員可以透過系統內的資料庫找出相關的答案再回答給客戶。

2.1 問題回報

在問題追踪系統中，問題回報是整個處理流程的起點，所以一個容易使用的回報方式，是一個重要的環節，如果不能即時將問題回報，整個處理就會延遲，造成其它更大的損失，所以一個好的回報問題的界面是十分重要。

問題追踪系統是用於處理軟問題，所以使用者界面大多是以個人電腦主，大部份情況下使用是沒什麼問題，但是要將這類使用者界面應用到一般工作場合使用，可能就不大合適，例如：無塵室內，電腦大多是作業員日常作業上使用，放置地點有一定的限制，無法隨意移動，當問題發生的地點沒有電腦的情況下，則需要先用紙筆記錄，之後到最近的電腦作回報，在這種情況下可能會有細節漏掉或者因轉述而產生錯誤。

再者問題的描述若只是單純的文字，有時不如使用一張圖片或者影音來得詳細，因為觀察問題的人不同，觀察的角度也不一樣，可能會對問題發生時產生先入為主的想法。一般的作法上，雖然可以透過其它如相機等裝置記錄，於事後再將這些資料附上，但是因為企業內對資訊設備的使用通常有安全性上的考量(如病毒感染)，可能需要到特定的電腦將資料取出，這些的不便，都會降低系統的可用性與使用者的使用意願。

行動裝置在這方面應該是一個不錯方案〔1〕，因為現代行動裝置體積小，可以適應各種空間，可以在有限的空間下操作，且附有多種感測器如：Camera、GPS、G Sensor 等特性，可以提供比一般個人電腦更多的資訊記錄方式，其無線通訊的特性，可以去除有線網路的限制，因此使用行動運算裝置作為另一個問題題回報的界面，以達到一般個人電腦所不易達成的功能。

一般人對於企業內部使用行動裝置的第一印象大概都停留在 BlackBerry(黑莓機)〔4〕，但是由於其系統的專注於端對端(End-to-End)應用，所以想要客製程式來使用的話，會因為其封閉的系統架構而困難重重且成本高昂，不過近年來由於行動裝置的普及，其可選擇的系統也多了，除了 RMI 的系統之外，其中較具代表性的系統大概有二個，iPhone 及 Android，且企對內部近年來開始導入雲端服務，因此未來行動裝置將有應用的可能性，不過在導入企業內部前，還是得要多作評估才行，如：

1. 程式和部署要花多少時間
2. 需要支援的平台有多少

3．是否需要整合多種媒體

4．裝置來源(公司配給還是員工自行購買)

以上因素都是行動裝置部屬於企業時要考慮的，如果以上的問題在評估後沒有問題，再來就是平台選擇的問題了。

2.2 問題處理能力評量

一般的問題追踪系統，本身只有提供一般性的統計資訊，其資訊不外乎依狀態、優先權、嚴重性、類別…等分類，其資訊不容易用來比較前後期的能力變化，所以需要一個指標來評量問題處理的能力。因此在資料的使用上，會應用到問題追踪系統的問題處理時間、問題發生的次數、問題的轉手次數這三組資料來設定管制規格並計算能力，使用的方法會以 SPC 統計製程管制的指數計算方式，作為計算的基礎。

於本系統而言，將使用工業管理上常應用的 Six Sigma〔5〕中用於 SPC 之 C_p、C_{pk}、C_a 製程能力指標〔6〕，用於評量問題處理能力，這三項指標一般用於量測產品的實際產出與規格的偏移程度，藉此找出問題，提高製程能力。Six Sigma 理論為 Motorola 於 1986 年提出的觀念〔7〕， Six Sigma 的流程為 DMAIC，即：

Define：定義問題、專案。

Measurement：測量缺陷、指標。

Analysis：分析問題。

Improve：改進。

Control：控管方式。

運用以上六個步驟，改進品質，在此不討論 Six Sigma 的流程，只取其製程能力指標，一般在工業界是在 SPC 系統利用 C_p、C_{pk}、C_a 三個指數〔8〕，以下作一簡單的介紹。

C_p (Capability of Precision) 製程精密度：

C_p 指數是用來評估產品的分佈情況，當產品的分佈愈集中時，其 C_p 值愈大，製程愈穩定且趨於集中，但 C_p 指數是假設產品的平均值等於規格中心值，所以看不出產品是否有整組偏移的情況，因此，相同的 C_p 值，並不一定代表產品的品質是相同的，其公式如下：

雙邊規格：$\dfrac{(USL - LSL)}{6\sigma}$

只有規格上限時(C_{pu})：$$\frac{USL - Mean}{3\sigma}$$

只有規格下限時(C_{pl})：$$\frac{Mean - LSL}{3\sigma}$$

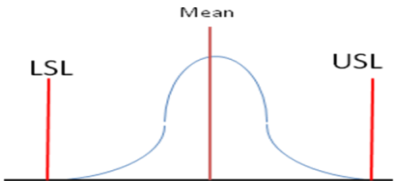


圖 3：平均值=規格中心

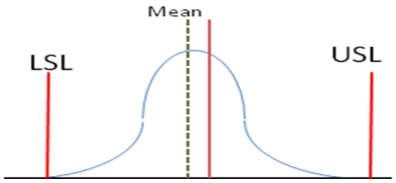


圖 4：平均值<規格中心

由上圖觀察，圖 3 與圖 4 的分佈是相同的，所以 C_p 值計算起來是一樣的，但圖 3 的平均值等於規格中心，而圖 4 的平均值則比規格中心小，所以以製程能力而言，圖 3 是比圖 4 好的，這是在 C_p 值所看不出來的，不過這個問題可以由 Ca 值來補足。

C_a(Capability of Accuracy) 製程準確度：

C_a 值是指產品的平均值與規格中心的差異程度，若產品的平均值與規格中心的差異愈小，則 C_a 值小，代表其產品愈接近期望的規格，反之則是指產品的規格偏離中心值，不過此指數僅能看出產品平均值是否接近規格中心，看不出產品的分佈是否集中，其公式如下：

雙邊規格：$$\frac{|\ (USL + LSL) / 2 - Mean\ |}{(USL - LSL) / 2}$$

單邊規格：無法計算。

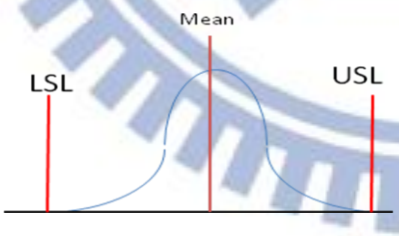


圖 5：分佈較為集中

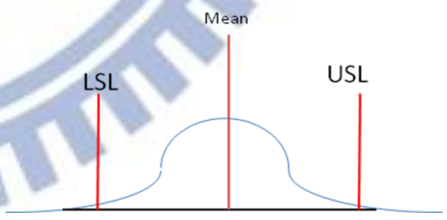


圖 6：分佈較不集中

由圖 5 和圖 6 可看出，雖然二者的平均值皆在規格中心上，C_a 值極小，但是可以明顯看出圖 5 的分佈比圖 6 來得集中得多，因此圖 5 的製程能力是比圖 6 來得好，以上二個指數皆有其盲點存在，而在 C_{pk} 則可提供一個綜合能力指標。

C_{pk}(Capability of Process Index) 製程能力指數：

C_{pk} 是將 C_p 及 C_a 值合併考量，同時評估產品的精密度及分佈情況，其公式如下：

雙邊規格：$|1-C_a| * C_p = \text{Min}[\frac{USL - Mean}{3\sigma} ; \frac{Mean - LSL}{3\sigma}]$

只有規格上限時：$\frac{USL - Mean}{3\sigma}$（同 C_{pu}）。

只有規格下限時：$\frac{Mean - LSL}{3\sigma}$（同 C_{pl}）。

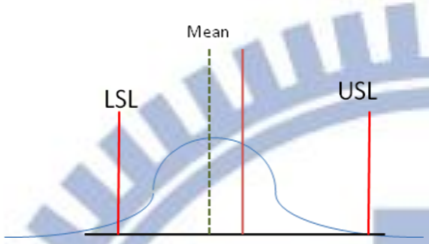


圖 7：Ca 差　Cp 差　Cpk 差

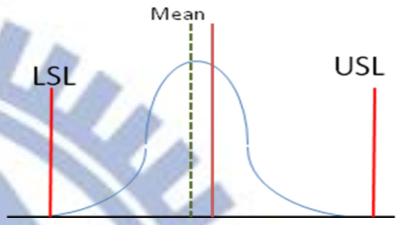


圖 8：Ca 差　Cp 好　Cpk 普通

圖 9：C_a 好　C_p 差　C_{pk} 普通

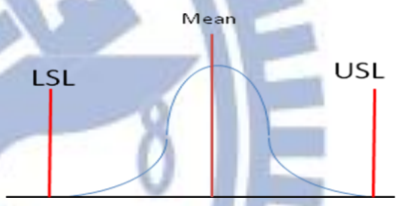


圖 10：C_a 好　C_p 好　C_{pk} 好

由上面四個圖來看，C_{pk} 由 C_a 和 C_p 共同評量，其中之一較差時，會造成製程能力下降，所以若要達到較佳的製程，除了產品要符合規格外，其產品的穩定度也是製程能力的考量之一。

USL：規格上限。

LSL：規格下限。

Mean：平均數。

σ ：標準差。

以上是統計製程管制使用的指標，在系統設計一節，將會針對這個指標作一些修改，以符合問題管理的領域。

2.3 知識的累積及應用

在知識累積的部份，其流程如研究目的一節所提，其知識將以問題追踪系統上

的記錄作為撰寫的來源，將這部份由問題追踪系統分離出來，而形成問題追踪處理、結案、知識的撰寫及應用這個循環，而分離出來的部份，將以維基系統作為撰寫的平台，透過擴展維基系統的語法，來達成知識應用的前置處理。

因為維基系統具有線上多人共同創作、編修歷史記錄…特點，且語法比網頁簡單等特性，所以選擇這個平台作為知識撰寫及應用資料來源使用，透過研究顯示，維基百科對於工作的績效有顯著的影響〔9〕，企業可以使用維基系統來增加員工的能力。

維基百科的影響不只於閱讀的人可以透過有條理的閱讀來增加知識，其撰寫的人也可以透過撰寫的過程，重新將知識組織、匯整成易於理解的資料，這對於撰寫的人也有相當多的益處。

但維基百科也並非沒有缺點，其缺點主要是它的撰寫較為正式，換句話說就是必需如同寫文章般有條理〔10〕，無法類似筆記式的記錄（因為它必需要給不同領域的人閱讀），所以有時撰寫的人會因此而怯步，如果能一開始就有一些資料，撰寫的人能透過「整理」的方式，其撰寫的意願應該會提高很多。

在知識的運用方式上，目標是能在一次的編輯中，就將知識應用的前置處理作好，之後可以透過抽取程式，將不同目的的資料抽取出來，應用於其它的用途或者作為其它系統的資料來源，日後原始資料若有更新，只要重新執行抽取程式，就可以同步更新其它系統，而不必再維護不同的系統。

其主要的想法來自 Javadoc 的註解方式，Javadoc 可以在程式碼的文件中，註解使用 /** … */ 這個標記時，表示這個註解未來可以透過 Javadoc 這個程式，將這個註解抽出來作為說明文件。

其註解的用的工具尚有 doxygen，其工具提供了更多語言的支援及更多的註解風格，doxygen 除了支援 Java 外，還支援了 C/C++、IDL (Corba, Microsoft 及 KDE-DCOP 類型)，可產生的文件有 HTML、XML、LaTex、RTF、Unix Man Page 等格式，並且可以在 Project 的組態檔中，預先設定好要產生的文件規格。其註解的方式除了 Javadoc Style 外，還支援 Qt Style /*! … */及單行註解等，是十分彈性的註解工具。

系統設計上會在維基系統中以擴充語法的方式，將文件作上標記(Markup)〔11〕，於日後透過類似於 Javadoc 的抽取程式，將特定的資訊抽取出來，作為其它系統資料的來源，將知識再利用。目前已設計了二種類型，一為重點標記，這個標記的用途是作為抽取重點資訊，產生技術文件、訓練素材或教育用的簡報等。另一種是將問題的追踪過程，作為一個測驗題目使用，因為追踪的過程處理方式有對

也有錯，很適合作為測驗用的題目，未來可以在訓練完成後，作成果測驗。

三、設計與方法

在系統的設計及方法，著重在以下幾個重點上：

1. 系統的設計是以問題追踪系統為中心，其流程為下圖 11 的問題、處理與追踪、結案，因此對會對於不同的問題追踪系統作一比較，找出符合需求之問題追踪系統。
2. 在問題回報的設計上，需要使用行動裝置，所以對於行動裝置的平台作一比較，選擇較為符合需求之平台。
3. 問題處理能力的評量，其設計目的是用來計算出前後期的能力指數，透過前後期的比較，找出效率的瓶頸發生在何處，進而達成改善效率的目的，其流程為下圖 11 的結案、檢討問題處理能力、處理與追踪這裡將討論使用問題追踪統中哪些資料用來評量處理能力及 SPC 能力指數的改寫。
4. 最後是知識的整理與應用，如何應用問題追踪系統的資料，整理到維基系統並且設計出技術文件、題庫的標記規格，其流程為圖 11 的處理與追踪、知識、枝文件及題庫。

圖 11：作業流程

整個系統的流程如圖 11，透過二個主要的過程：問題處理能力、知識的整理與應用，結合到問題追踪系統中，以達到強化問題處能力及知識的累積等目的。

3.1 問題追踪系統的選擇與比較

一個軟體工作開始進行的時候，所面臨的第一件事就是 Bug 的管理，一開始可能是使用一個簡單的表格來記錄，不過隨著時間過去，很快的 Bug 的數量就多到無法使用一個簡單的表格就可以處理了，所以就需要一個好的系統來記錄、追踪、管理這些問題，這類的系統便是 Bug Tracking System〔12〕〔13〕〔14〕.

Bug Tracking System 一般也稱為 Issue Tracking System，基本上這樣的系統是用在軟體生命週期中，回報、指派、追踪、歸檔問題的系統〔15〕。

這類的系統演化到了後來，功能愈來愈強大，除了一般的追踪記錄外，不同的系統還發展出了各自不同的特色，因此目前有很多的系統可供選擇，目前在設計上有幾個選擇上的重點，系統的在運作上可以有多種平台選擇、多種資料庫支援、使用人數的限制上少（商用軟體大多會以使用人數作為計價標準）、SOAP 支援(在行動平台上會以 SOAP 作為通訊的格式)及易於使用開發語言…特性，作為選擇的標準，以下就列出幾個候選的系統，逐一比較。

表 1：Bug Tracking System 比較

特色 \ 系統	Bugzilla	iTracker	Mantis	IBM Rational ClearQuest	Track Studio
平台獨立性	主要為 Linux，在 windows 系統上使用時需要 Cygwin 程式	Yes，以 J2EE 為基礎	Yes，使用 Appache Http Server + PHP，跨平台性佳。	Yes，Linux、Windows、Solaris 等	Windows、Linux
資料庫獨立性	No, 只能使用 MySQL	Yes	Yes，但主要為 MySQL	Yes, 主流資料庫皆支援 (DB2、SQL Server、Oracle 等)	DB2、SQL Server、Oracle、MySQL、PostgreSQL
自訂功能	低，需自行撰寫	中，可自訂欄位及報表	高，可自訂欄位，並且可透過 Plug-in 系統自行撰寫功能	低，可透 Java、Perl 進行二次開發	中，可自訂欄位、報表
使用人數限制	無	無	無	視授權而定	視授權而定
主要操作界面	Web	Web	Web	Web Interface	Web Interface

	Interface	Interface	Interface		
SOAP 支援	需自行撰寫	自行撰寫	內建 mantisconnect 的 SOAP 界面	需自行撰寫	內建 SOAP 界面
維基系統整合	Yes	No	Yes	No	No
開發語言	Perl CGI	Java	PHP	Java、Perl	Java
價格	無	無	無	視授權而定：USD1349~USD8004	視授權而定：免費~USD4495

由以上的比較來看，以 Mantis 的擴充性、支援 SOAP 界面及可整合維基系統等特性較為符合系統上的需求，且在使用上的限制較少(免費)，且該系統在設計上有 plug-in 的設計，在日後撰寫能力指數時，可以不需更動到系統程式，且軟體更新時不會影響自行撰寫的功能等特性，所以系統的發展上將以 Mantis 為基本系統。

3.2 行動平台選擇

行動裝置的平台，在目前有好幾個平台，其特色都不一樣，在此以使用率、平板支援、續航力、開發平台多樣性、第三方軟體支援…等幾個考量作為平台選定的條件，比較表如下：

表 2：行動裝置比較表

	iOS	Android	Window phone	Note Book
使用率	中	高	低	高
支援平板	有	有	目前無	無
續航力	佳	佳	佳	普通較差
開發平台	OS X	Windows、Linux、Mac	Windows	大部份為 Windows
輸入裝置	觸控面板	觸控面板	觸控面板	鍵盤
廠牌相容性	佳	視生產廠商實作而定	視生產廠商實作而定	佳
第三方軟體支援	少	多，且可使用 Java 原生程式	目前較少	多

透過上表的比較與表考量下，選定以 Android 作為開發問題回報界面的平台，

因為 Android 平台的開發平台可以跨多種作業系統及第三方軟體的支援較多,所以作為開發的選擇。

3.3 能力評量方式

一般問題追踪系統除了記錄下每次處理、追踪的資料外，尚記錄了計多額外的資訊，如：處理的時間、人員、類別…等資料，大部份這些資料都只有用來簡單的統計而已，若能利用一些方法，進一步的處理，將可用來反應一個團隊處理問題的能力。一般而言，一個企業內部的團隊，平日的工作大多是在處理問題，不論是需求也好、問題也好，一個團隊的能力大致反應在下列三件事上：

1. 問題的處理時間：一般而言，時間愈短，代表對於該領域的事務愈熟悉，不論是新的問題或者一般常見的需求，一般都應比一個剛接手該工作的人所需的時間來得少，若對於同一類的問題處理時間上有減少，表示團隊的能力有進步，反之則為退步。
2. 問題的發生頻率：若問題為常態性發生，則此問題的發生頻率若沒有維持在常態的次數，則為不正常，舉例來說，以某太陽能廠的網印製程為例，一星期要更換三次網版為正常，最多可更六次，若超過則為不正常，平均而言大約是 5 次，若對於此問題有較好的 SOP 來作預防時，則可降到接近三次，便可說此團隊對於問題的處理能力較好，因為能主動預防問題的發生。
3. 問題的轉手次數：一個問題或者需求在產生時，一般而言，第一個收到的人未必是團隊中的負責人，以一個 24 小時都在生產的工廠而言，一般工程師主要在日間上班，到了晚間則只有值班人員，通常這些人員只能負責處理簡單的工作，若較為複雜的工作，須要轉交給專門的負責人員處理，因此若對於問題的屬性或應負責的人員要是不熟悉的話，可能一個問題會如同皮球一般踢來踢去，導致問題不能在合理的時間內完成處理，所以問題的轉手次數亦可視作能力指標。

現在要將 SPC 的三項指標(C_p、C_{pk}、C_a)應用於問題處理能力的評量，不過在過使用這三項指標前，需對於這三項指標的基本假設作檢討：

1. 這三項指標的規格通常是具有上下限，其假設為產品量測結果，愈接近規格的中心愈好，因為不管是接近上限或下限，皆代表產品快超出規格，但是若用於問題的處理，其理解上就不太一樣，大多時候，處理問題的時間或發生的次數是愈小愈好，所以若低於中心時，反而應該是較佳才是，並

不一定是接近規格中心才是進步，若依照原 C_a 的算法，反而是退步。

2. 一般產品的規格，通常是平衡規格(例：85 +/- 0.1mm = 84.9mm ~ 85.1mm)，但是對於處理問題的領域上，規格非平衡規格的情況很常見，例如對於某問題的處理時間為最少為 20 分鐘，最多產線可以停 1 小時，但目標是在 30 分鐘內處理完畢，這就不是一個平衡規格，因此 C_a 在計算上反而不容易反映出真實的情況，因為 C_a 的計算方式中是以規格中心減去平均數來計算。

基於以上的問題，所以對於上述的三個參數的算法作了一些修正，以符合問題處理的領域。

在上述的 C_p、C_{pk}、C_a 公式中，規格的設定只有 USL、LSL 二個，在這裡引入一個新的設定 ET (Expect Target)，使得設定成為 USL、LSL、ET，而規格公差則為 SD(Spec Difference)，改寫後 C_a 公式如下：

$$C_a = \frac{| ET - Mean |}{SD} \qquad 1\text{-}1$$

$$ET = (USL + LSL) / 2 \qquad 1\text{-}2$$

$$= Expect\ Target$$

$$SD = (USL - LSL) / 2 \qquad 1\text{-}3$$

$$= USL - ET \qquad 1\text{-}4$$

$$= ET - LSL \qquad 1\text{-}5$$

所以在未設定 ET 值時，ET = 規格中心，若有設定 ET 值時，則以 ET 值為主，加入這個設定的理由是因為多數時候，規格的設定不一定為平衡規格，另外規格公差的計算方式也改為 SD，若規格設定有 USL 及 LSL 時，以 1-3 的計算方式，若只有單邊規格時，以 1-4 或 1-5 的計算方式。

另外考慮到 Mean 與 ET 的關係，例如：一個星期某類問題發生的次數，不可超過 10 次(USL)，希望平均可以維持在 8 次(ET)，若能低於 8 次則更好，所以原 C_a 的計算方式無法看出這樣的關係，因為原計算的方式是以絕對值的方式計算，所以將其改成：

$$C_a = \frac{ED}{SD}$$

$$ED = \begin{cases} -(ET - Mean) & \text{if Mean < Target is Better} \\ ET - Mean & \text{if Mean > Target is Better} \\ | ET - Mean | & \text{if Mean = Target is Better} \end{cases}$$

以上述例子來看，若平均發生 7.6 次，則 Ca = -(8 - 7.6) / (10 - 8)=-0.2，表示 C_a 比期望還少了 20% 的次數。

C_p 則維持不變：

雙邊規格：$\frac{(USL - LSL)}{6\sigma}$

只有規格上限時(C_{pu})：$\frac{USL - Mean}{3\sigma}$

只有規格下限時(C_{pl})：$\frac{Mean - LSL}{3\sigma}$

C_{pk} 公式則成為：

USL、LSL、ET 至少存在二個設定時：$|1-C_a| * C_p$

只有規格上限無 ET 時：$\frac{USL - Mean}{3\sigma}$（同 C_{pu}）。

只有規格下限無 ET 時：$\frac{Mean - LSL}{3\sigma}$（同 C_{pl}）。

其中 $Min[\frac{USL - Mean}{3\sigma}; \frac{Mean - LSL}{3\sigma}]$ 的公式不再使用的原因為此公式的假設是 Target = 規格中心，因為目前已經引入了 ET 值，所以上述公式不再適用。

如以上一個例子而言，若標準差為 3，則 C_p=(10-7.6)/3 = 0.8，而 C_{pk}=(1+0.2)*0.8 = 0.96，以這個例子來看，因為平均值低於 ET 所以 C_p 便有了加乘的效果，但若是以原來 C_a 的計算方式會使得 C_{pk}=(1 - 0.2)*0.8 = 0.64 ，其 C_{pk} 反而是退步的。

當找到了指標之後，可以將不同的部門及不同的類別的問題，作一個分類，將同一部門相類似的問題分成一類，訂下規格後，便可計算出指標，透過前後二個時間的比較，可以看出問題處理能力的變化，如與上期相比 Cp 值變小了，則可推測是什原因造成，如有新進人員還不熟作業過程，所以導致時間時長時短。若 Ca 值與前期相比偏高，則要檢討人員工作負擔是否過大，還是因為人力不足，亦或是士氣不足，而達不到目標。此指標雖然不能直接的指出問題所在，不過可以提示一個方向，透過問題的分析，找出效率問題且作適當的處置，改進整個團隊的工作表現，不斷的透過這樣的循環，以其將團隊的效率最佳化。

3.4 知識的應用方式及設計

當一個問題處理完成後，問題追踪系統會依時間順序，將追踪、處理問題的過程逐次記錄下來，雖然這樣的資料有一定的參考價值，不過這些資訊都是比較片面，在閱讀上比較不容易，所以希望這類的資料透過處理人員，以其專業的判斷將有用的資訊整理出來，以維基系統作記錄下來，而問題追踪系統將保留與維基系統的連結，在日後二個系統可以相互參照。

另外一個問題是如何將知識妥善的運用，在知識的累積上，使用維基系統是十分容易的，因為它可以針對同一條目，透過不斷的更新資訊，將知識不斷地累積，而閱讀者則可以在每次編修後閱讀到最新的資訊，但是在這些原始的資料與其它的系統通常不太容易整合，如測驗系統、技術文件等，雖然這些資料都源自維基系統，但大部份都必需利用人工將資料整理到其它系統，如果原始的維基系統資料有了更動，其它系統也必須手動更新，這種作業方式十分容易因人為疏失而出錯，因此系統設計上，須達到維基系統更新時、只要透過自動化工具便可其它系統更新的方式進行，因此在維基系統中導入標記(Markup)功能，而自動化工具則為其它系統對維基文件的抽取工具，其它週邊系統只要針對標記的規格，就可以抽取出所需的資訊。

其概念來源是 Javadoc 及 doxygen 的運作模式，在維基系統中實作出註解、標記的語法，而其它的外部系統就可以就其所需的資訊，針對特定的標記抽取出來，基本上這些標記必須不妨礙原來維基的文件的閱讀，所以維基文件呈現時須去除記號，改以其它不妨礙閱讀的方式呈現。

其標記的方式會以類似 html 的方式，以前後標籤的方式作註解，如<xxx> … </xxx> 的方式，因為在抽取資料時，可以只使用簡單的正規化語法(Regular Expression)就可以將資料抽取出來，這個功能將以 plug-in 的方式，在維基系統中開發。

整個的應用上，將開發二個主要的標記，一個為 Note 標記，作用是將資訊標記重點，因為維基系統上的資訊較為繁雜，若要作為技術文件、講義則需要將資訊抽出，簡化資料，語法如下：

<note type='title|text'>

內文……

- 維基系統項目語法標記

  ……

</note>

以 note 將資料標記起來，其 type 有二種：

title：

指的是筆記的標題，於擷取的時候可以作為章節的標題。

text：

為筆記的內文，於擷取時作為筆記的內文。

這個標記的目的在於生成技術文件，其 Title 的部可以是該文件的標題，而 text 則為內文，另一個用途可以作為教學用簡報，透過這個標記可以自動生成簡報文件。

另一個為 Exam 的標記，可以在維基系統中即時寫下一些小測驗，可以讓閱讀者在閱讀後作一個自我測驗，另外也下以在日後抽取出來，作為測驗系統的材料或者是 FAQ。其語法如下：

```
<exam type='multi|fill|question'>
    測驗題目…
    <ans value='O|X'>答案 1</ans>
    <ans value='O|X'>答案 2</ans>
</exam>
```

exam type:

測驗題型，有以下三個類型。

1. multi:多選。
2. fill：填充。
3. question：問答。

ans value：

在 type='multi'時有效，當 value 為'O'時，表示此答案為正確，'X'時則表示此答案為假。

這個標記的資料來源，為問題追踪系統中，針對某一問題的追踪過程，同一個問題，可能由不同的人提出不同的意見而出現各種不同的條目、資料，其解法方案

也因人而異，但最終在問題結案時，總會有一個結論，其中的方案可能有效也可能無效，故是十分有用的測驗題材，針對這些方法，可以透過上面的標記，標記出正確與不正確的資料,可以作為日後的測驗素材,也可以針對正確的答案,抽取成 FAQ。

四、系統實作與問題探討

系統設計將會說明整體的系統架構，包括軟、硬體系統設計及軟體實作等部份，並說明使用到的技術等，並探討可應用的場合等。

4.1 系統架構

系統硬體的架構圖如下：

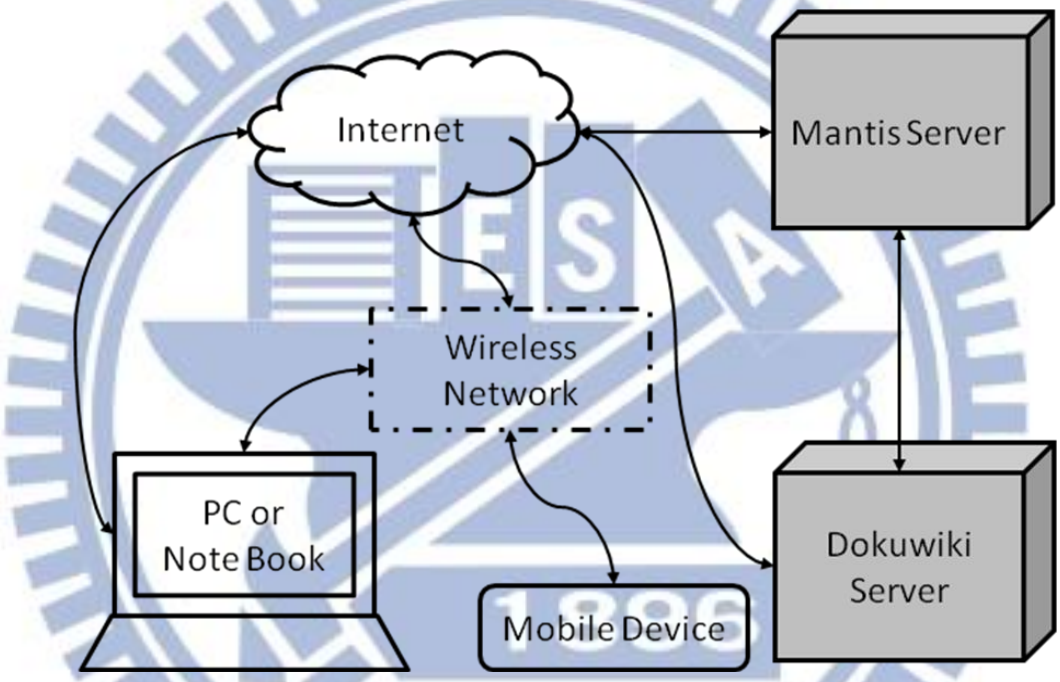


圖 12：硬體架構

系統主要由 Mantis Server 為中心，透過網路或同一台 Server，整合 Dokuwiki Server，行動裝置的部份，則是透過無線網路的方式與 Mantis Server 連線，將問題回報給 Mantis Server 或者查詢過去的案例、維基系統資料，於現場初步排除問題。

而接收到問題的人員，則使用 PC 連線至 Mantis Server 處理、分配問題，並將有價值的案例，整合到維基系統。

主要硬體需求：

1. Mantis Server：Intel、Solaris、Mac 系統皆可，但是由於需要處理上傳檔案資料，所以記憶體至少具備 2G MB。
2. Dokuwiki Server：與 Mantis Server 同。

3. Mobile Device：Android Device，具備 500 MB 記憶體，照相模組，錄影(音)功能，無線網路或行動網路功能。

4. PC、Notebook：具備網頁瀏覽器，有線或無線網路即可。

系統軟體架構如下：

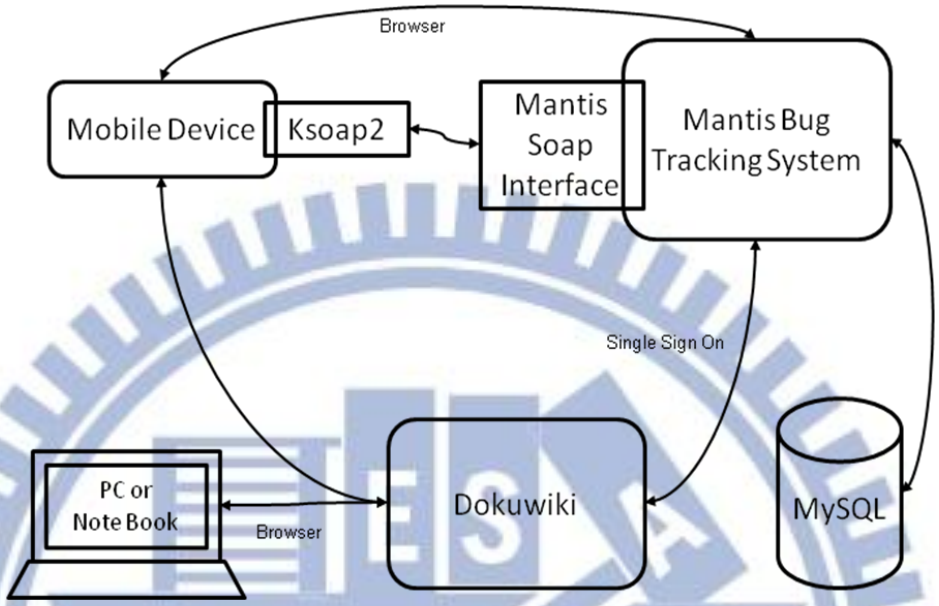


圖 13：系統架構

Mobile Device 需要將問題即時記錄下來，所以不論網路是否連線，都必需記錄，所以功能上不適合使用網頁，因此另外撰寫 Android Application，資料於網路有連線時，再將問題上傳，所以透過 ksoap2〔16〕〔17〕與 Mantis 的 soap 介面整合，以達到 Store and forward 的功能，未來亦可以加上一些其它的特性，例如：GIS 資訊。

Dokuwiki 與 Mantis 透過 plug-in 整合單一登入，減少帳號管理上的麻煩。其它的 Client 皆使用 Web 的方式作業即可，因為其即時性需求較不明顯。

軟體需求如下：

1. Mantis Server：
   a. 作業系統：Windows、MacOS、OS/2、Linux、Solaris、BSD 等，以能夠執行伺服軟體即可。
   b. 伺服軟體：Microsoft IIS 或 Apache Http Server。
   c. PHP：最小需求為 5.1 版，建議 5.2 以上版本。
   d. Database：MySQL (4.1.x 或更高版本)、MS SQL Server、 PostgreSQL 或 IBM DB2。

e. Mantis Bug Tracking System 1.2.X。

2. Dokuwiki Server：
   a. 作業系統：Windows、MacOS、OS/2、Linux、Solaris、BSD 等，以能夠執行伺服軟體即可。
   b. 伺服軟體：Microsoft IIS 或 Apache Http Server。
   c. PHP：最小需求為 5.1.2 版。
   d. Dokuwiki 2012-1-25。
3. Mobile Device：
   a. 具備 Android 2.2 以上功能之行動運算裝置。
   b. ksoap2-android-assembly-2.6.0。
4. PC、Notebook：
   具備有常用瀏覽器之作業系統即可，主要支援 Internet Explorer、Mozilla Firefox、Safari、Chrome 等。

4.2 系統設計及實作

整體系統的程式開發主要集中在 Android Client 及 Mantis Plug-in 上，其餘的部份透過系統整合為主。

4.2.1 Android Client

Android Client 功能主要為回報問題及查詢為主，功能上較為陽春，不過已經可以達到作業上的需求，其功能有四個部份：

1. 設定：設定 Mantis 的位址及使用者的帳號密碼。(圖 15)
2. 新增問題回報：回報問題至 Mantis 系統，並有方便的附檔功能，可以於照像後或直接將檔案上傳至 Mantis 系統。(圖 16)
3. 查詢及其它功能：目前暫時以自動登入 Mantis 網頁為主，其功能由原 Mantis 界面提供，因為其它功能不需要有上失敗暫存功能，故以 Mantis 功能，增加自動登入功能即可，因原系統開發商已有開發手機版網頁功能，故不再重複開發現有套件。(圖 17)
4. 斷線上傳問題功能：原問題若在上傳過程中上斷線情況時，系統會自動將資料記錄在行動裝置之資料庫中，待回復連線後再手動將問題上傳。(圖 18、圖 19)

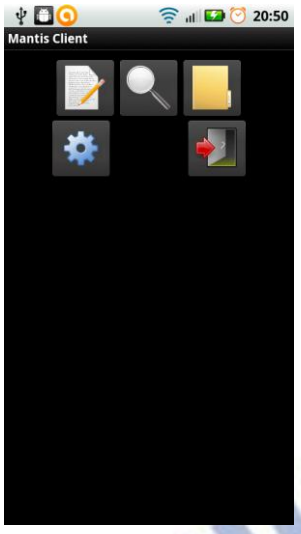


圖 14：主畫面

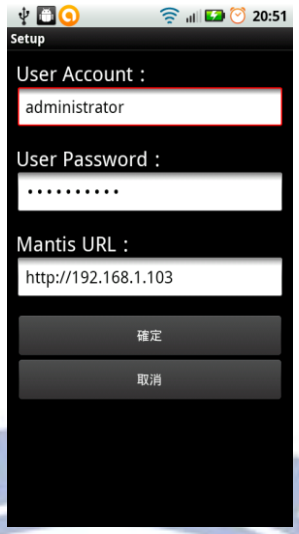


圖 15：設定畫面

圖 16：回報及附件

圖 17：查詢及其它功能

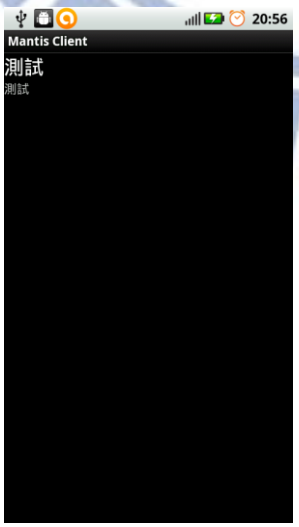


圖 18：暫存問題列表

圖 19：暫存問題處理

流程圖如下：

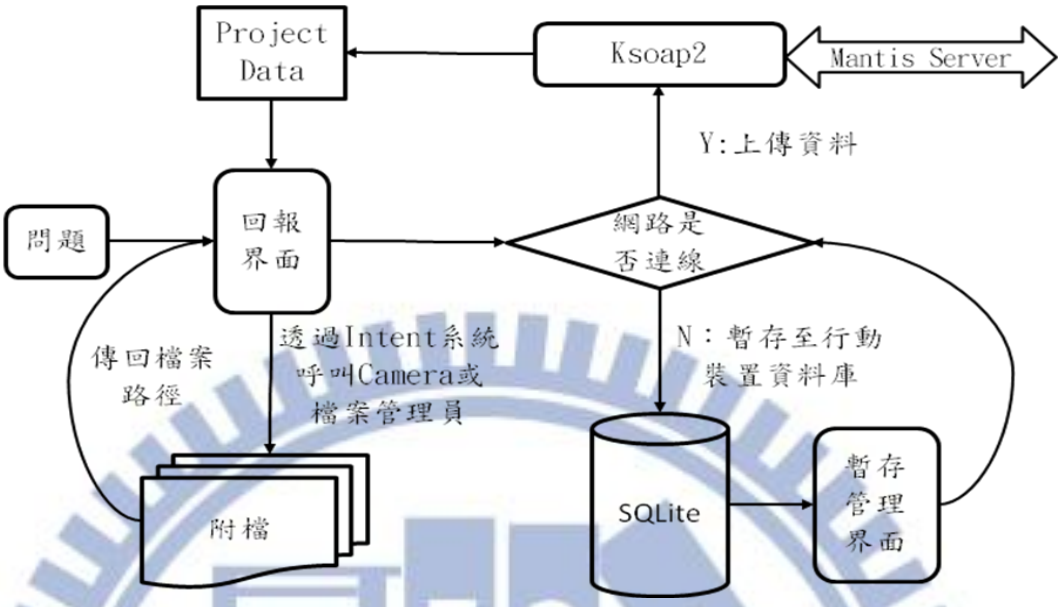


圖 20：Android 程式流程

在問題回報時，透過回報界面(圖 16)，於進入界面時，系統會先透過 Ksoap2 向 Mantis 系統詢問專案、類別、出現頻率、嚴重性、優先權等列表，讓回報人員作選擇，填妥摘要與說明後，可透過界面利用 Android 系統的 Intent 呼叫，使用照相、檔案管理等工具，將檔案夾帶於問題附件。

當回報人員將資料填妥後，系統便會透過 Ksoap2 將資料上傳至伺服器，若網路有問題時，則會暫時將資料存在行動裝置的資料庫中，其資料庫在安裝 Mantis Client 時便會自動產生，並向系統註冊，待網路可用時，則可透過未傳列表(圖 18)選取要上傳的資料，選取上傳(圖 19)。

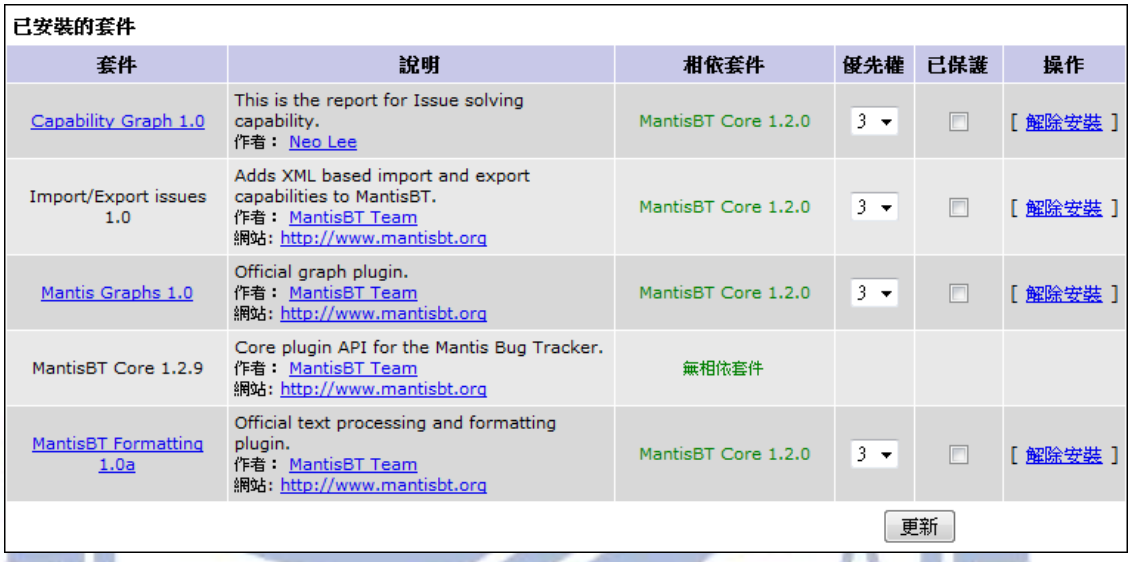
在整個撰寫 Android 裝置的過程中，其實遇上了不少的問題，其中之一就是行動裝置的記憶體有限，雖然 Android 可直接使用原生的 Java 函式庫，不過這類的函式庫大部份是針對桌上型電腦或伺服器，所以應用在行動裝置上時，記憶體的管理上就比較沒那麼輕鬆，有時容易產生記憶過不足的錯誤，例如在上傳檔案時，因為使用 SOAP 界面，所以必產生一份 xml 檔案，若檔案太大則會產生錯誤而無法上傳，取代的方式便是捨棄 Ksoap2 的函式庫，改採 Http 連線的方式，一邊產生 xml 文件，同時上傳 xml 文件。

另外一個問題是由於 Android 的作業系統是仰賴各家廠商自行實作，所以對於其功能未必有完整實作，如：SQLite 的 Data Cursor，於 Motorola 的 Milestone 2 上使用時，其 movenext 函式無法如預期行為將資料指標指向下一筆，須使用其它

的方式替代，如此的不相容會造成開發上的困難。

4.2.2 Mantis Capability Graph

在 Mantis 上所開發的功能，為一個 Plug-in 功能，可以分析其問題記錄，並產生能力指標，其指標的計算方式如前述，其安裝方式為 Mantis Plug-in 的標準安裝方式，其套件名稱為 Capability Graph，如下圖 21：



已安裝的套件

套件	說明	相依套件	優先權	已保護	操作
Capability Graph 1.0	This is the report for Issue solving capability. 作者： Neo Lee	MantisBT Core 1.2.0	3		[解除安裝]
Import/Export issues 1.0	Adds XML based import and export capabilities to MantisBT. 作者： MantisBT Team 網站: http://www.mantisbt.org	MantisBT Core 1.2.0	3		[解除安裝]
Mantis Graphs 1.0	Official graph plugin. 作者： MantisBT Team 網站: http://www.mantisbt.org	MantisBT Core 1.2.0	3		[解除安裝]
MantisBT Core 1.2.9	Core plugin API for the Mantis Bug Tracker. 作者： MantisBT Team 網站: http://www.mantisbt.org	無相依套件			
MantisBT Formatting 1.0a	Official text processing and formatting plugin. 作者： MantisBT Team 網站: http://www.mantisbt.org	MantisBT Core 1.2.0	3		[解除安裝]

更新

圖 21：Mantis Plug-in 管理介面
功能路徑為：管理->套件管理

Capability Graph 的功能在於計算出能力指數，故需先設定規格，因此功能上會有規格編修及報表二部份，如下圖 22：

主頁 | 我的工作 | 檢視問題 | 回報問題 | 版本更動紀錄 | 版本規劃 | 統計資訊 | Wiki | 管理 | 我的帳號 | 登出 問題編號 #

[列印報表] [統計資訊] [進階摘要] [Capability 圖表]
[報表] [設定規格]

設定 新增規格

選擇	ID	上限	下限	目標	方向	類型	註解
	Solve Time	48	12	16	= Target	Solving Time/Issue	
	Assign Times	5		2	< Target	Reassign Times/Issue	
	Happen Frequency	5		3	< Target	Times/Day	

刪除規格

圖 22：報表規格列表
功能路徑為：統計資訊->Capability 報表

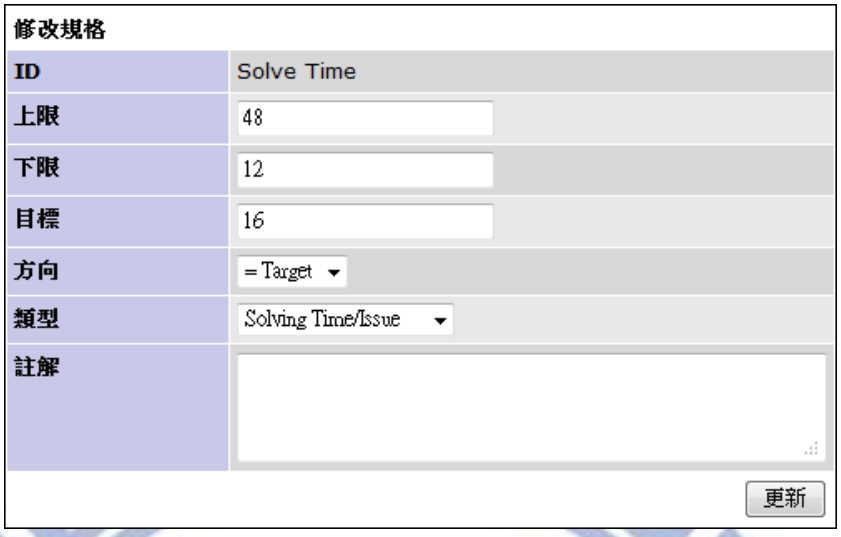


圖 23：設定報表規格

如圖所示，其規格需設定上下限、目標值、方向值及計算類型，上下限值及目標如字面意思，即規格的上限值、下限值及其望的目標值，而方向則是平均值則分為近目標為佳、小於目標值為佳，還是大於目標值為佳，其計算是影響 C_a 值。

類型的部份共有六種：

1. Solving Time / Issue：指的是每個問題從回報到結案或解決的處理時間。
2. Reassign Times / Issue：每個問題被重新指定負責人的次數。
3. Times / Hour：每小時問題發生的次數。
4. Times / Day：每天問題發生的次數。
5. Times / Week：每星期問題發生的次數。
6. Times / Month：每月問題發生的次數。

完成以上的設定後，便可於報表選取規格，計算能力指數。

圖 24：規格設定

計算的方式，是以專案的類別來分類計算，對於不同的類別，其規格應該不相同。如圖 24 先選專案、類別，再選取報表規格，並選取計算的區間，按下查詢之後，便會顯示能力指標及圖表：

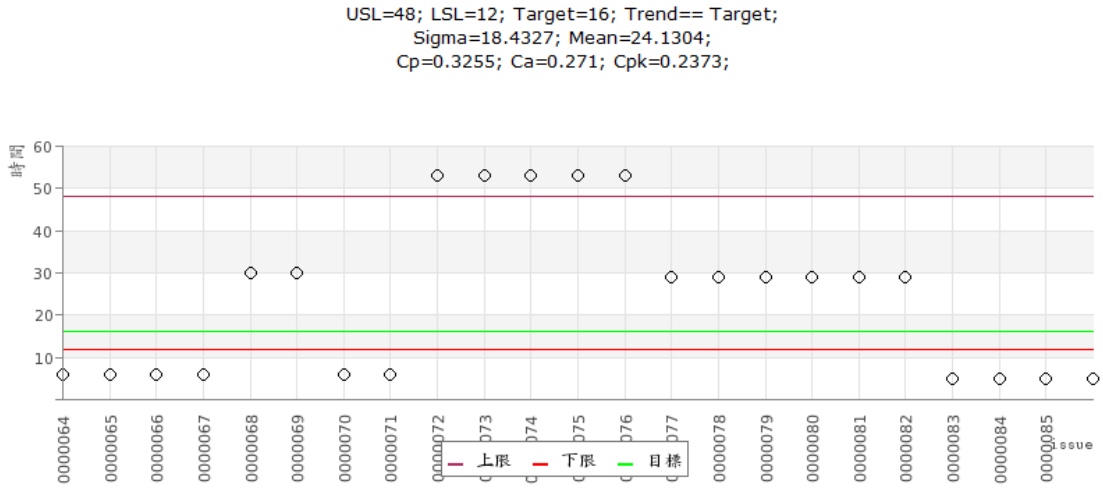


圖 25：報表結果

於上圖可以看出其 C_p、C_a、C_{pk} 值及分佈情況，透過不同的時間可以比較其能力是否有已進步。

Mantis 與維基系統透過整合，可於每個 Issue 連結至維基系統，將記錄與知識作連結，如下圖：

圖 26：維基系統連結

如圖 26，整合 Mantis 後，於 Issue 上方會有 Wiki 連結，可連結至維基系統作編修知識(圖 27)，其知識可以作為日後的參考。

圖 27：Wiki 例子

4.2.3 維基系統標記

在 Dokuwiki 系統中，新增加二個標記功能，分別為 Exam 及 Note，此二個標記功能主要在於未來若需要作訓練或測驗時使用，Note 的功能，可將維基系統中重點的部份標記下來，透過 Parser 程式，可將筆記的部份獨立出來，產生 PDF 或文字檔等，方便未來產生重點筆記或技術文件、投影片等。

而 Exam 的部份，則可以在維基系統中，即時標記題目，在未來需要產生測驗的時候，透過擷取程式將題目擷取出來，產生題庫，應用在紙本或其它測驗系統來應用。

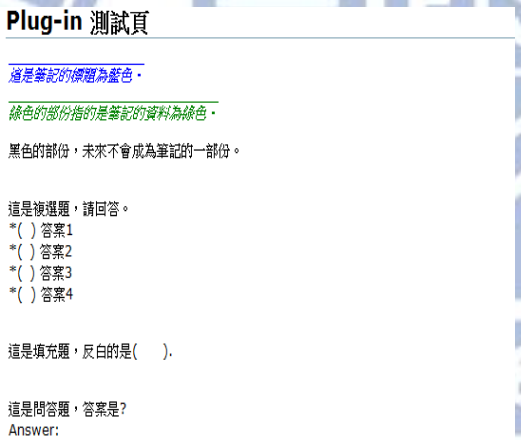
這二個功能在維其系統產生文件的時候，會將呈現方式作一些改變，以方便閱讀，而不會在文件中看到標註碼而影響閱讀，這二個功能之所以採內嵌的方式，主要是使文件的的撰寫人可以在撰寫文件的時候，可以不必再到其它系統撰寫、維護題庫，而可以在同一份文件內就將題庫、筆記、技術文件等一次完成，除了可以減少撰寫的負擔外，對於文件更新時，不必再手動同步其它系統的資料，減少管理上的麻煩。

Exam 功能是透過 plug-in 的系統，產生測驗的題目，在維基系統的文件編輯完成後，維基系統頁面，會將答案隱藏，需透過反白才可將答案顯示出來，規格如系統設計時所規範，其實作完成後例子如下：

```
<exam type='multi'>
這是複選題，請回答。
<ans value='o'>答案1</ans>
<ans value='x'>答案2</ans>
<ans value='x'>答案3</ans>
<ans value='o'>答案4</ans>
</exam>

<exam type='fill'>
  這是填充題，反白的是<ans>答案</ans>.
</exam>

<exam type='question'>
這是問答題，答案是?
<ans>
這是答案，
你答對了嗎?
</ans>

</exam>
```

圖 28：Exam 例子

如上圖(圖 28)例子，exam type 共三種，產生後的文件如下(圖 29)：



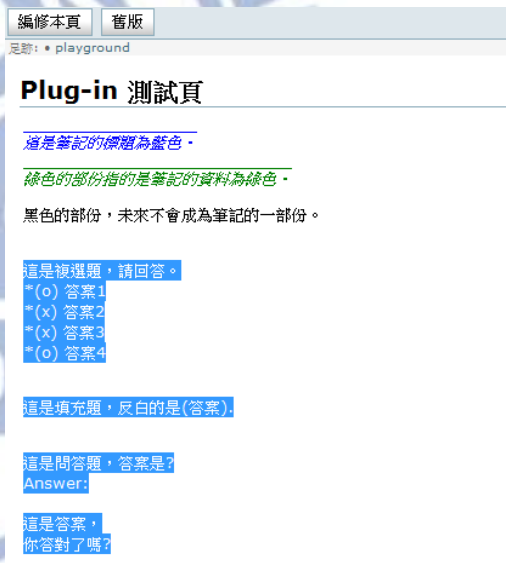


圖 29：Exam 結果

圖 30：Exam 答案

其產生的文件，會將答案隱藏，透過選取反白，方可將答案顯示出來如圖 30。

Note 系統，亦是透過 plug-in 的方式，將資料標記下來，於維基系統產生閱讀文件時，將作文筆記的部份畫上標線，筆記 Title 的部份文字會以藍色，而內文則以綠色標記文字，Note 系統了支援維基系統的標記方式，故在 Note 系統內可以使用維基系統語法，其實作完成後例子如下：

```
====== Plug-in 測試頁 ======

<note type='title'>
這是筆記的標題為藍色。
</note>

<note type='text'>
綠色的部份指的是筆記的資料為綠色。
</note>

黑色的部份，未來不會成為筆記的一部份。


<note type='text'>
  * 項目只會有上標線
  * 2
  - 有序列表項目只會有上標線
  - 2
</note>
```

圖 31：Note 標記

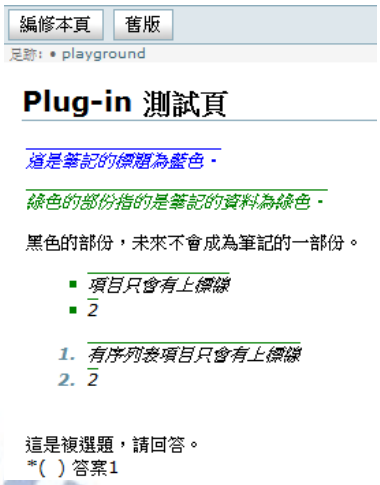


圖 32：Note 結果

上有一個標題，二個內文，其產生後的閱讀文件如圖 32，其 Note 標題的部份會被轉成藍色，且會產生上標線，而內文的部份為綠色，項目的部份，只有上標線為綠色，透過顏色和標線，方便閱讀的人知道哪些項目被標記成了筆記。

其它尚有一些現成的維基系統的外掛，如 wiki slide、wiki book…等，對於這些知識的再運用，都有良好的幫助，不會使得這些知識因為雜亂、無結構而失去運用的可能性。

五、結果與分析

這套系統透過整合，將數個不同領域的工具整合在一起，讓問題處理的工作更加容易，且對於知識的整理、再運用上也更加方便，比起一般的問題追系統，本系統的優點為：

1. 彈性可配合不同的場合應用：

   在一般有電腦可以使用的情況下，可以直接使用 Mantis 系統的介面來操作，但是在無法使用電腦的場合，可以配合行動裝置上的 Android Application 來記錄、查詢資料，達成彈性運用、即時記錄的目的，不會因為沒有電腦系統而造成問題無法記錄、追踪，且可以透過各式的行動裝置的多媒體能力，能更方便、更全面的記錄下問題。

2. 問題處理參考：

   每一個問題，在處理過程的每一個步驟都可以透過 Mantis 的追踪記錄功能，記錄下處理的過程，於日後相類似的問題發生時，可以透過關鍵字句，找出以往的處理方法，作為問題發生時的第一線處理參考，另外有維基系統的資訊，可以更有組織的學習。

3. 處理能力的參考指標：

   系統中所提供的問題處理能力指標，可以提供團隊的能力參考值，於日後找出能力瓶頸並改善團隊的能力，提高其運作效率。

4. 知識的再應用：

   透過新增的標記功能，可以將知識作標記，於將來可以透過類似 Javadoc 的擷取程式，將題目抽出，作為其它測驗系統的題目來源，或者將重點筆記擷取出來，作為教育訓練用的教材或者技術文件資料。亦可透過其它的維基系統外掛功能，將知識作有效的再運用。且當維基系統資料有所改變時，於編輯的過程中所作的更新，於日後抽取資料時，亦會同步更新，免除需維護多個系統的同步問題，減少管理上的負擔。

5. 減少企業損失：

   一般而言，每次問題的處理過程，企業皆需付出一定的成本，不論是實際上金額的損失，如產品損失成本、違約成本，或無形商譽的損失，若能將發生過的問題記錄下來於日後應用、預防，便可將企業的損失減少。

6. 減少因人員異動，產生知識的斷層：

一般而言，人員異動時，於過往的知識常常就隨著人員而消失，導致知識的斷層，而新進的人員又得從頭開始，將過去的錯誤重新再犯一次，重新學習，不但浪費資源，也會造成損失，故若將知識予以保存，則可避免這樣惡性循環。

另外再與其它不同性質的系統作一比較，更有以下的優點，如下表所示：

表 3：系統比較表

項目 \ 系統	Issue Tracking System for Knowledge Accumulation	Bug Tracking System	FDC	客服系統
主要用途	除了可以應用於製程問題處理，及用於找出團隊能力瓶頸與改善。	用於軟體問題追踪與管理。	即時監控機台的狀況，早期偵測與分類故障。〔18〕	建立問題處理知識庫，以利經驗分享，並節省重複處理成本。
問題來源與記錄	可透過行動裝置或者網頁，將問題資料即時記錄下來，亦支援其它媒資料。	只能透過網頁將資料記錄、上傳等。	資料來源為 MES 系統及 CIM 系統自動收集。	由使用人員透過電話或網頁，問題反應於客服人員。
問題分類	需人工分類。	需人工分類。	因資料皆已參數化，固可以透過系統預先設定的規則分類。	透過關鍵字將問題分類。
能力指標	提供問題處理能力指標。	一般統計資料，如未完成項目、完成時間、分類表…等。	無特殊指標。	無特殊指標。
知識應用	透過標記功能，可將知識產	較無結構的處理記錄。	無。	產生客服 FAQ 知識庫。

	生多樣化應用，如：測驗、投影片…等。			
建置成本	因為使用的皆為 Open Source 系統，故系統成本低廉。	其軟體系統本身為 Open Source，故成本低廉。	需有專門人員撰寫機台的連線程式，整合系統，其成本較高。	需有專人配合流程撰寫，其成本較高。
跨領域應用	少許改寫便可跨領域應用，如工地的監工、客服…等領域。	透過修改可符合一般性的問題追踪。	不易用於其它領域。	不易用於其它領域。
說明		以 Mantis 為代表		以經濟部商業司之客戶問題管理服務系統為例(民國 92 年)

以上比較表可看出，除了問題分類上需要人力介入外，本系統不論是在建置成本、應用上、及使用的方便性上，皆較優於其它系統。

其重要貢獻便是可以讓一個團隊了解本身的能力外，還可透過系統的回饋來提昇團隊的價值，並且出工作上的問題予以善，並防止相同的問題一再重覆的發生，減少人力的浪費及成本。

六、結論及建議

6.1 系統可用性及其它應用

本系統適用的對象，不只是工程上問題的追踪處理，只要稍加修改，即使是一般其它的產業亦可使用，例如：

以建築業來說，於施工現場時，一般情況下都不會有資訊設備，所以若是工程遇上問題的時候，通常只能靠都電話，以語言的方式溝通，但是很多情況可能透過照片或者影片更能將問題釐清，所以若行動裝置能支援將資料上傳到系統，處理的人更能了解問題，提出適當的解決方案，且工地現場的人員也可以利用系統，於第一時間找出替代方案處置好問題。

另外，一般公司都會有客服及客訴單位，這個系統可應用於客戶問題回覆品質，透過系統，可以了解客服的回應速度，及對於產品是否有未處理好的問題，例如對於一般性問題必須於 3 小時內回覆，目標 1 小時完成，小於 1 小時更佳的規格，藉此計算其客服能力指數，或者透過客訴的次數來了解產品是否有品質以外的問題，因為一般相信產品於出廠時，其品質應有一定的控管，所以若是客訴依舊很多的時候，可預設其問題應該在產品品質以外。

再者例如旅遊業，於行程中若遇上問題，可以透過統，將其資料回報回去，問題追踪系統的部分則可增加 GIS 的地理資訊，透過地理位置的劃分，來找出糾紛發生次數、處置速度、處置態度較差的行程，予以矯正及預防等。

所以此系統的可應用領域十分廣泛，並不限定於一般的企業內部追踪問題，透過適當的修改，其應用是十分廣泛。

6.2 未來方向

雖然本系統已經有了不少特點，但還是有部份功能是可以再加強的，可加強的部份如下：

1. 本系統對於問題的分類還是依賴人工作業，事實上以目前的技術而言，有不少的演算法可以將問題自動化的分類，直接將問題分配給處理人員，而不需要再透過一個管理人員來分配〔19〕。
2. 人員處理的時候，可以透過關鍵字，將過去的資料作一候選，給處理人員

在處理時可作為參考。

3. 另外，本系統與維基系統整合的部份可以加強，將 Mantis 內的說明資料，可以自動的帶入維基系統，讓人員在撰寫文件時，可以減少輸入的負擔及減少輸入資料的時間。
4. 雖然系統本身已使用了行動裝置來記錄問題，但是免不了還是要輸入文字，所以可以加強開發輸入資料的方式，例如：利用語音辨識系統，使用者只需將問題錄下來，透過辨識系統，可以將語音轉成文字資料，於之後輸入系統。
5. 對於知識的有效性，可以透過投票的機制，找出較為重要的資料，於人員在查詢時，作為排序的依據，減少搜尋資料時的負擔。
6. 在標記的部份，希望可以增加標準處理流程(SOP)的標記，可以用來建立問題處理的除錯系統，幫助人員處理問題。

另外本文也展示了行動裝置應用於企業內部的可能性，因為行動裝置的體積及人機介面，相當適合應用於空間狹小及需要不斷移動的場合，如廠區內部沒有電腦可使用的場合，還有倉儲區可應用於點貨等，所以行動裝置應用於企業內部是未來的趨勢。

本系統除了上述的功能與貢獻外，另一方面也展示了另一種系統開發的可能性，一般在企業的內部系統，大部份是傾向購買現成的系統，加上向系統供應商購買客製服務，或者是由內部人員從無到有，自行開發出整套系統，不論是哪一種，都人分花費人力與財力，而本系統是將其它領域的系統及其它領域的知識，透過適當的改寫，來符合目前的需求。

以本系統為例，對於問題的追踪管理上，借用了軟體品質系統中的 Bug Tracking System，其基本的概念上是相同的，只要稍加修改即可適用，另外透過統計製程管制的概念，產生能力指標，來找出團隊處理問題能力的瓶頸，最後利用 Javadoc 的觀念，將維基系統中的資料作適當的標記，讓其它系統可以更容易的利用維基系統的知識，且更新時更加方便。

以上皆顯示出，未來的系統開發，並不一定需要由零開始，可以藉由其它領域的經驗，應用在現有的問題上，因為在其它的領域上可能已經有了不錯的方法或系統，只需少修改即可，本論文的開發方式，希望可以作為其它開發者的參考。

參考資料

〔1〕 林聖岱、曾惠斌、王明德，「行動式 PDA 工程資訊管理系統與營建知識入口網站整合之研究」，台灣大學，第六屆營建工程與管理研究成果聯合發表會，民國 91 年。

〔2〕 陳義欽，「KT 問題管理模式之研究」，私立中原大學，碩士論文，民國 92 年。

〔3〕 張妤慧，「整合六標準差之通用問題管理歷程之研究」，私立中原大學，碩士論文，民國 91 年。

〔4〕 Mache Creeger, “Mobile Devices in the Enterprise: CTO Roundtable Overview”, ACM Queue, Volume 9, August 29, 2011.

〔5〕 張公緒、孫靜，「六希格瑪工程簡介」，品質月刊，第 37 卷第 12 期，73~78 頁，民國 90 年 12 月。

〔6〕 曹永誠，「先進製程控制技術(APC)導論」，電機月刊，174 期，202~215 頁，民國 94 年 6 月。

〔7〕 李旭華，品質管理，二版，台中，滄海書局，民國九十八年。

〔8〕 王金燦、邱治瑋，「戴明與克勞斯比理論對製程能力指數 CPK 影響之研究—以明鴻企業為例」，中華民國品質學會第三十八屆年會暨第八屆全國品質管理研討會論文，469~480 頁，台北，民國 91 年。

〔9〕 黃教勝，「維基百科對知識分享行為與績效之影響」，義守大學，碩士論文，民國 98 年 6 月。

〔10〕 Jonathan Grudin, “Enterprise Knowledge Management and Emerging Technologies”, System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on, Volume 3, P57a, 04-07 Jan. 2006.

〔11〕 Guerrieri, E., “Software document reuse with XML”, Software Reuse, 1998. Proceedings. Fifth International Conference on , P246-254, 2-5 Jun 1998.

〔12〕 Volker Gruhn , Juri Urbaiinczyk, ”Software process modeling and enactment: an experience report related to problem tracking in an industrial project”, Proceedings of the 20th international conference on Software engineering, P13-P21, Kyoto Japan, 19 Apr 1998-25 Apr 1998.

〔13〕 Source Wikipedia, Bug and Issue Tracking Software: Bugzilla, Comparison of Issue-Tracking Systems, Mantis Bug Tracker, IBM Rational Clearquest, 1st Edition, USA, Books LLC, Wiki Series, August 2011.

〔14〕 Nicolás Serrano , Ismael Ciordia, “Bugzilla, ITracker, and Other Bug Trackers”, IEEE Software Volume 22 Issue 2, P11-P13, IEEE Computer Society Press Los Alamitos , CA , USA, March 2005.

〔15〕 Dane Bertram , Amy Voida , Saul Greenberg , and Robert Walker, “Communication,

Collaboration, and Bugs: The Social Nature of Issue Tracking in Small, Collocated Teams", Proceedings of the 2010 ACM conference on Computer supported cooperative work, P291-P300, New York, USA, 6-10 February 2010.

〔16〕 Johannes Knutsen, "Web Service Clients on Mobile Android Devices", Norwegian University of Science and Technology, Master's thesis, June 2009.

〔17〕 Peddola , Sri Tulasi, "Developing Google android mobile clients for web services", San Diego State University, Master's thesis,25 Jan. 2012.

〔18〕 梁惠姿，「建構半導體製造過程產品異常資料挖礦技術及其雛型系統之研究」，國立清華大學，碩士論文，民國 94 年。

〔19〕 阮淑婷，「運用資料探勘與社會網路分析支援問題管理」，國立交通大學，碩士論文，民國 97 年。

〔20〕 Mantis Bug Tracker Developers Guide, http://www.mantisbt.org/docs/master-1.2.x/en/developers.html

〔21〕 The Development Manual, http://www.dokuwiki.org/development

〔22〕 Android Reference Document, http://developer.android.com/reference/packages.html

〔23〕 KSOAP2 User Document, http://code.google.com/p/ksoap2-android/w/list

〔24〕 Javadoc Tool Home page, http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html

自　　傳

我的名字是李勝斌，在家中我排行老大，底下有一個弟弟，一個妹妹，母親因為生病的因素，在十多年前就過世了，父親目前已退休，因家庭因素，目前居住於苗栗縣竹南。

我的求學過程算是比較長，因長輩的關係，專科念的是淡專國貿，在校成績還算不差，但在第五年時，覺得國貿並非我想從事的工作，所以決定延畢一年，準備二技的資訊組考試，但時間畢竟不夠充裕，只考到了實踐大學資管系。

由於該校在當時第一次辦資訊相關系所，所以各項資源嚴重不足，已影響教學品質，所幸系上教授熱心的教導，參與了國科會醫學影像相關的研究，得到了十分寶貴的經驗，但是由於對該校辦學品質的失望，所以於一年級暑假時，決定插班考上淡江資管，在此時對於資訊方面的能力，才有了較為正規的訓練，大部分的資訊能力都在這時具備。

出社會後的工作是從事資訊相關的工作，大多是與生產、設備相關的系統，於工作後八年左右，因為想要再提昇自己的能力，所以報考了交通大學資訊學院研究所，也很幸運的考上了，在求學的過程中，一方面要工作，一方面要唸書，時間上過得很緊湊也很充實，學了不少的新知識，希望畢業後能再有機會回到校園。