東海大學高階經營管理碩士在職專班(研究所)
碩士學位論文

快速軟體開發模式-以工程計價系統為例

Rapid Software Development Model

– A Case Study of Project Valuation System

指導教授：黃延聰 博士

研 究 生：李其盛 撰

中華民國 105 年 7 月

論文摘要

論文名稱：快速軟體開發模式-以工程計價系統為例

校所名稱：東海大學高階經營管理碩士在職專班 (研究所)

畢業時間：2016 年 7 月

研 究 生：李其盛　　指導教授：黃延聰 博士

論文摘要：

長期以來軟體系統開發就是一件挺折磨人的事情，使用者抱怨工程師不了解領域知識(Domain Konwledge)，工程師抱怨使用者提出的需求不合邏輯或需求講不清楚，雙方在天平的兩端彼此拉鋸，究其緣由在於軟體是一種高度腦力開發的產品，是介面(Interface)、抽象(Abstraction)、封裝(Encapsulation)與相依性(Dependency)分析…等觀念的集合。在產品尚未成型之前，雙方彼此的認知與理解本就有一定的落差，因此軟體系統的開發往往是邊做邊吵，邊吵邊修，順利的可以跌跌撞撞地完成，不順利的則嚴重影響時程或超出預算。

為解決軟體系統開發的問題，本研究採用領域知識系統(Domain Knowledge System，簡稱 DKS) 為軟體開發工具，實作一套工程計價系統，藉此與傳統軟體開發方式進行比較，以驗證 DKS 系統是否可以提高軟體開發效率，增加軟體可維護性並降低開發成本的目的。

【關鍵字】軟體危機、軟體工程、領域知識系統

Abstract

Title of Thesis：Rapid Software Development Model - A Case Study of Project Valuation System

Name of Institute：Tunghai University

Executive Master of Business Administration Program

Graduation Time：（07／2016）

Student Name：Chi-Sheng Lee Advisor Name：Dr. Yen-Tsung Huang

Abstract：

For a long time, software system development is a very grueling thing. Users complained that engineers do not understand the domain knowledge; engineers complained that the demand made by the user are illogical or unclear. Both parties are saw in two side of balance. The reason of the conflict is due to the software is a highly intellectual products and is the assembly of concepts of interface, abstraction, encapsulation and dependency analysis ... and etc. Before the un-molding product, there is a certain on the mutual awareness and understanding between two parties, so the development of software systems is often by doing noisy, repair during noisy; it can be successful completed stumbled when smoothly; it may serious impact to the schedule or over budget when un-smoothly.

To solve the problem of software system development, this study adopts Domain Knowledge System as a software development tool and implements a project valuation system, by using this system to compare with the traditional software development methods, in order to verify whether the DKS system can improve the efficiency of software development, increase the purpose on maintainability and reduce software development costs.

Key words: Software Crisis, Software Engineering, Domain Knowledge System

目　次

表　次

圖　次

第一章　緒論

第一節　研究背景

自 18 世紀工業革命以來，人類的生產力因標準化規格及模組化生產，致使產能大幅提升。然相較之下，軟體產業的生產力卻遠遠落後於硬體產業的生產力，主要是因為軟體系統的開發比過去更加的複雜，在需求面包含電子商務(E-Commerce)、企業資源計畫(ERP)、eMail、簽核作業…等許多功能與服務之整合需求，在技術面有多種程式語言、開發工具、開發標準、作業系統、資料庫…等，因此軟體開發整合速度緩慢，專案開發進度難以掌控，再加上現今軟體之跨系統、跨平台、跨地域及可移動性的需求，以及需配合新的硬體設備、作業系統及軟體技術的更新腳步，致使軟體開發及維護的困難度與複雜度越來越高。

使用者經常以執行效率及彈性來評估軟體系統的優劣，但軟體的可維護性卻經常被忽略。所謂軟體維護(Software Maintenance)是一個軟體工程名詞，是指在軟體產品發佈後，因修正錯誤、提昇效能或其他屬性而進行的軟體修改(ISO/IEC 14764:2006)。一般認為軟體維護只和修正錯誤有關。不過有研究指出 80%的軟體維護工作是用在非糾正性的行動(Pigoski, 1997)。軟體維護是軟體系統交付使用之後，為了修正軟體運行的錯誤、效率的提昇或為了因應用戶新的需求而增加新功能的修改軟體的過程。1960 年以前人們並未重視軟體維護的重要性，直到 1960 年之後，愈來愈多的軟體系統被開發完成並上線使用，人們開始了解軟體系統置換的困難度，因此軟體的維護性才變成軟體生命週期的主要過程。至今為止，使用舊的程式語言，舊的設計方法所開發出來的舊系統數量仍相當的多，這些軟體系統模組間的高偶合度，功能與資料的高相依性提高了該軟體本身維護的複雜度與困難度，最後軟體系統的維護比較於軟體開發所耗費的時間與成本要高出許多，甚至產生軟體危機(Software Crisis)，所謂軟體危機是指在軟體開發及維護的過程中所遇到的一系列嚴重問題，這些問題皆可能導致軟體產品的壽命縮短、甚至夭折。實證研究得知超過 30%的專案未能依照原計畫時程完成，大約 50%的專案超出原

計畫工作量 10%，50%以上的專案因未充份規劃而無法成功(Genuchten，1992)。軟體開發是一項高難度、高風險的活動，由於它的高失敗率，故有所謂軟體危機之說。軟體危機的本源是複雜、期望和改變。這個術語用來描述正急遽增加之電腦的力量帶來的衝擊和可能要處理的問題的複雜性(鄭炳強，2012)。這些嚴重的問題阻礙著軟體生產的規模化、商品化以及生產效率，讓軟體的開發和生產成為制約軟體產業發展的瓶頸。

第二節　研究動機與目的

軟體標準(Software Standard)廣泛運用於軟體開發流程以提昇軟體品質、軟體開發效率、速度及軟體可維護性，例如 UML(Unified Modeling Language)、XML(eXtensible Markup Modeling Language)、設計樣板(Design Pattern)、軟體框架(Framework)及 CASE Tool(Computer-Aided Software Engineering)…等。雖然這些軟體標準提昇了軟體的開發效率，但個別的軟體標準往往只有支援軟體開發的部分過程。只利用軟體標準來支援特定軟體開發階段是不夠的，必須將所有階段的所有軟體模型整合起來以達成軟體模型間的一致性與同步性，才能真正的達成軟體可維護性的需求。要整合所有軟體模型等於是要整合所有的軟體標準，各個標準的軟體模型間需要人為的思考從中穿針引線，其過程容易發生人為的疏失而降低軟體的可維護性。

基於上述原因，本研究採用領域知識系統(Domain Knowledge System，簡稱 DKS) 為軟體開發工具，實作一套工程計價系統，藉此與傳統軟體開發方式進行比較，以證明 DKS 系統是否可提高軟體開發效率，增加軟體可維護性並降低開發成本的目的。DKS 系統拋棄傳統程式導向的開發模式，將知識從程式碼中抽離出來，只要在 DKS 系統中定義描述系統規格，不須撰寫程式碼即可產出可運行的軟體系統，完全做到 Language Independent & Database Independent，所有的需求、分析、設計、測試都在 DKS 系統中統一管理，當需求有變更時，必須從規格源頭進行分

析、設計，所有的元件、功能鍵都有一個獨立的元件料號，當料號有變更，所有引用此料號的規格皆會列出由系統開發人員逐一確認是否須同步調修。

第二章　文獻探討

第一節　軟體危機

一、發展歷史

1960 年以前，電腦剛問世之時，軟體設計一般只是為了某一特定的需求或應用而在特定的電腦上設計和撰寫程式，採用密切依賴於電腦的機器碼(Machine Code)或低階語言，如：組合語言(Assembly Language)，此時軟體規模小，缺乏系統化的開發方法，設計軟體等同於程式設計，基本上是自己設計、自己使用、自己操作、自給自足式的個人化的軟體生產方式。

1960 年中期，大容量之高速電腦出現，擴大了電腦的應用範圍，軟體開發需求量急劇增長。高階語言開始出現；作業系統(Operation System)的發展改變了電腦的應用方式與變化；在大量數據處理的需求下致使資料庫管理系統(Database Management System)的誕生。軟體系統規模越來越大，複雜度越來越高，凸顯了軟體可靠性問題的嚴重性。原本個人化的軟體生產方式不足以滿足外界的需求，迫切需要改變軟體生產方式，提高軟體產能，因此開始爆發軟體危機。

1968 年，北大西洋公約組織的電腦科學家在德國召開國際會議討論軟體危機問題，並正式提出「軟體工程」(Software Engineering)一詞，作為研究和克服軟體危機的方法。但軟體危機不但沒有消失，還有加劇的態勢。主要表現在：

1. 資訊系統總成本中軟體成本在所佔比例居高不下且逐年上升。1985 年，美、日兩國的統計數字顯示，軟體成本大約佔資訊系統總成本的 90%。
2. 軟體生產力提高的速度遠遠落後於電腦應用的需要。

一、外部現象

軟體危機主要表現在：

1. 軟體開發費用和進度失控，為了趕進度或壓低成本無法兼顧軟體品質。
2. 軟體可靠度差,雖然耗費了大量的人力物力,卻越來越難以保證系統的正確性，

錯誤率大幅增加，因軟體錯誤造成的損失十分驚人。

3. 軟體維護困難,很多程式缺乏相關的開發文件,程式中的錯誤難以發現及改正，有時改正了已有的錯誤又引起新的錯誤，引發漣漪效應(Ripple Effect)，漣漪效應是在描述一個事物造成的影響漸漸擴散的情形，類似物體掉到水面上，所產生的漣漪漸漸擴大的情形(Tracy，2008)，在經濟學中有漣漪效應的例子，例如一個人支出的減少會造成其他人收入的減少，連帶也使得他們可支出的金額減少；應用在計算機科學中，則在說明由於一個模組修改，造成其他模組也需隨之修改的情形。
4. 1980 年後，雖然軟體工程研究及技術水準已有長足進步，但是軟體生產水準仍遠遠落後於硬體生產水準的發展速度。
5. 使用者需求變更。使用者需求變更的主要原因：
   (1) 使用者在軟體開發初期無法準確完整的向開發人員表達他們的需求。
   (2) 軟體開發人員對使用者需求沒有正確全面了解的情況下，就急於撰寫程式。
   (3) 外部環境改變，如新技術的產生、法令規章的變更、配合客戶的要求。
   (4) 內部環境改變，如公司組織的調整，公司重大投資，新技術引進。

二、危機原因

電腦軟體是一種邏輯性產品而非物理性產品，軟體樣品就是產品，打樣過程也就是生產過程；軟體不會因使用時間過長而老化或損壞；軟體具備可運行的行為特性,在寫出程式碼並在電腦上試運行之前,軟體開發過程的進度較難以量化，軟體品質也較難以評估，因此軟體開發過程的管理和控制十分困難；軟體品質與每一個組成部分的不同實體的品質緊密相關，因此在運行時所出現的軟體錯誤(Bug)，大多在開發階段就已存在而一直未被發現，修改這類錯誤意味著修改原來的設計，使得軟體維護的困難度遠比硬體維護高。

軟體產品是思維的產物，因此軟體生產力取決於軟體人員的教育、訓練及經

驗積累；另外大型軟體系統往往需要許多人協同開發，更要求軟體人員深入了解應用領域知識(Domain Knowledge)，因此需要在軟體開發人員之間以及用戶與軟體人員之間緊密溝通，過程中難免產生對需求理解上的誤差，導致後續的錯誤設計，而往往需要付出巨大的代價才能消除這些錯誤。由於電腦技術和應用發展迅速，知識更新週期縮短，軟體開發人員處在變化之中，不僅要面對硬體更新的變化，還要研究日趨複雜的應用領域問題；軟體開發人員必須經常調整自己的知識結構來解決新的問題的需求，這也造成了軟體開發與維護的困難，軟體生產具備知識密集和人力密集的特點是軟體危機的根源。造成軟體開發危機的原因歸納如下：

1. 使用者需求不明確

軟體開發過程中，使用者需求不明確問題主要體現在四個方面：

(1) 在軟體開發出來之前，使用者自己也不清楚軟體開發的具體需求。

(2) 使用者對軟體開發需求的描述不精確或有遺漏、甚至有誤。

(3) 軟體開發過程中，使用者還持續提出修改軟體開發功能、介面、環境等要求。

(4) 軟體開發人員對使用者需求的理解與使用者本來的期望有落差。

2. 缺乏有效的理論指導

缺乏一套有力的方法和工具的支持。軟體開發異於其他工業產品開發，其開發過程是複雜的邏輯思維過程，是抽象而難以描述的，軟體產品賴於開發人員高度的智力投入，需要程式設計人員的技巧和創造性，造成軟體開發產品的個性化，這也是造成軟體危機的一個重要因素。

3. 軟體規模日漸龐大

大型軟體開發需要組織各種技能的人力共同完成，彼此間的專長與技能不同，若溝通不及時、不準確、便會產生誤解。軟體開發人員無法獨立自主、有效地處理大型軟體開發的全部關係與分支，因此容易產生錯誤與疏漏。

4. 軟體開發複雜度增加

由於電腦科技的發展快速，使用者對軟體產品的需求與期望也相對提高，導致

軟體開發不只是規模擴大，其複雜度也急速增加，而複雜度相乘的結果，軟體開發先天上的困難。

三、 實際案例

1995 年，Standish Group 研究機構調查美國境內 8,000 個軟體專案，結果顯示有 84%軟體計畫無法於既定時間、經費中完成，超過 30%的計畫於執行中被取消，專案預算平均超出 189%。

1. IBM OS/360

OS/360 作業系統被認為是一個典型的案例。到現在為止，它仍然被使用在 360 系列主機中。這個經歷了數十年，極度複雜的軟體專案甚至產生了一套不包括在原始設計方案之中的工作系統。OS/360 是第一個超大型的軟體專案，它使用了 1,000 人左右的程式設計師。佛瑞德•布魯克斯在他的《人月神話》書中曾經承認，在他管理這個專案的時候，他犯了一個價值數百萬美元的錯誤(Frederick P. Brooks，2011)。

2. 美國銀行信託軟體系統開發案

1982 年美國銀行進入信託商業領域並規劃發展信託軟體系統。計畫原訂預算 2 千萬美元，開發時程 9 個月，預計於 1984 年 12 月 31 日以前完成，後來至 1987 年 3 月都未能完成該系統，期間已投入 6 千萬美元。美國銀行最終因為此系統不穩定而不得不放棄，並將 340 億美元的信託帳戶轉移出去，並失去了 6 億美元的信託生意商機(李允中，2009)。

3. 其他

(1) 1985 年－1987 年，導致病人死於 Therac-25 醫療線性加速器的過量輻射(維基百科，2015)。

(2) 1996 年 6 月 4 日，亞利安 5 號運載火箭首次測試發射，火箭在發射後 37 秒被迫自行引爆，肇因於 64 位元的運算錯誤地變為 16 位元的運算，造成程式崩潰後處理器發生數字溢流，將感測角度的垂直讀值錯誤的代入到水平

值做運算，導致火箭在高速下進行 90 度水平滾轉而崩解，觸發自毀裝置的啟動。總結來說是控制火箭飛行的軟體故障而非無法運算(Simson, Garfinkel，2005)。

(3) 1998 年 8 月 27 日在卡納維爾角空軍基地的首次發射失敗起因於自三角洲二號運載火箭所修改的軟體在第一節飛行時無法導航，控制火箭的液體耗盡後，火箭失去控制亦被銷毀，星系（Galaxy）X 衛星-休茲（Hughes）HS601 HP 型也墜入大西洋中(維基百科，2014)。

第二節　軟體工程

一、歷史由來

1968 年，北大西洋公約組織的科技委員會召集了近 50 名一流的程式人員、計算機科學家和工業界巨頭，討論和制定擺脫「軟體危機」的對策。在這次會議上第一次提出了軟體工程(Software Engineering)這個概念，研究和應用如何以系統性的、規範化的、可定量的程序化方法去開發和維護軟體，以及如何把經過時間考驗而證明正確的管理技術和當前能夠得到的最好的技術方法結合起來的學科。軟體工程主要是希望藉由工程界的通用原則，利用軟體開發工具(Tools)、軟體開發方法論(Methodologies)、處理程序(Process)等來進行軟體系統的開發工作，以建立一個具有品質的軟體系統。它涉及到程式語言、資料庫、軟體開發工具、標準、系統平台、設計模式等方面的領域。軟體工程的目標為在固定預算、時程的前提下，開發出具有可靠性、有效性、可維護性且滿足使用者需求的軟體產品。根據 IEEE (1999)的定義：「軟體工程係以工程的方法來建構軟體，也就是以有系統、有規律、可定量衡量的方式，來評估、定義、開發、維護一個軟體系統」。追求這些目標對軟體產品的品質和開發效率有所幫助，同時也減少了維護的困難。

二、名詞定義

軟體工程有下列幾項定義：

1. 創立與使用健全的工程原則，以便經濟地獲得可靠且高效率的軟體。(Bauer，1968)
2. 實際應用科學知識在設計、建構電腦程式，與相伴而來所產生的檔案，以及後續的操作和維護上。(Boehm，1991)
3. 使用與系統化生產和維護軟體產品有關之技術與管理的知識，使軟體開發與修改可在有限的時間與費用下進行。(Fairley，1985)
4. 應用系統化，遵從原則，可被計量的方法來發展、操作及維護軟體；也就是把工程應用到軟體上。(IEEE，1990)
5. 建造由工程師團隊所開發之大型軟體系統有關的知識學科。(Ghezzi, Jazayeri, & Mandrioli，1991)
6. 系統化地應用工具和技術於開發以電腦為主的應用。(Conger，1994)
7. 與開發、管理及更新軟體產品有關的理論、方法及工具。(Brian Randell，1996)
8. 軟體工程是關於設計和開發優質軟體。(Pfleeger, 2001)
9. 一種知識或學科，目標是生產品質良好、準時交貨、符合預算，並滿足用戶所需的軟體。(Schach，2011)

三、發展歷程

軟體工程的發展過程歷經三個階段，分別如下：

1. 程式設計階段

1946 年～1955 年為程式設計階段。這個階段的特點是：沒有軟體的概念，程式設計主要為特定硬體而開發，規模小、工具簡單、無明確分工、程式設計追求程式技巧和節省空間，沒有設計文件資料，主要用於科學計算。

2. 軟體設計階段

1956 年～1970 年為軟體設計階段。這個階段的特點是：硬體環境相對穩定，「軟體工作室」的開發組織形式因應而生，同時套裝軟體開始廣泛被使用，逐漸建立了軟體的概念。隨著電腦技術的發展和電腦應用的普及，軟體規模也日趨龐大，

產生了各類的高階程式語言，不斷拓寬應用領域，人們對軟體的需求量劇增。但軟體開發技術沒有重大突破，軟體產品的品質也不高，軟體生產力低，導致了軟體危機的發生。

3. 軟體工程階段

1970 年後為軟體工程階段。由於軟體危機的影響，迫使人們研究、改變軟體開發的技術手段和管理方法。這個階段的特點是：硬體朝向巨型化、微型化、網路化和智能化四個方向發展，資料庫技術已成熟並廣泛應用，第三代及第四代程式語言出現；第 1 代軟體技術：結構化程序設計在數值計算領域被廣泛應用；第 2 代軟體技術：軟體生產過程加入軟體測試技術、方法、原理；第 3 代軟體技術：在軟體需求分析和描述方面使用需求定義技術。

第三節　軟體開發模式

軟體開發模式的定義為「軟體開發全部過程、活動和任務的架構框架」，是為了完成特定的目標而設計，由於電腦軟體的需求越來複雜，規模越來越龐大，因此需要軟體系統開發方法來提昇開發品質與效率。

以往軟體開發因為欠缺完善的系統化開發流程，因此常導致開發成本超出預算造成浪費、功能與期望的落差、維護困難、技術疊床架屋或導入系統後效益不彰等軟體危機。資訊人員便將軟體製程以工程化、系統化的模式進行開發，以改善上述軟體危機。軟體開發模式係指各軟體開發流程與軟體開發模型，軟體發展流程可歸納出不同的軟體開發模型，以下章節將簡述軟體開發流程及探討軟體開發模型特性與優缺點比較。

1. 軟體發展流程

專案生命週期是指軟體從計畫、需求開始，經歷分析設計、實現、部署、維護，直到最後逐漸消亡的過程，由專案起始、規劃、執行、結案，監控等五個生

命週期階段，其中以執行階段最容易發生風險。依軟體專案發展流程步驟之順序可分成兩大類，分別為線性發展流程與多循環發展流程，分類說明如下：

1. 線性發展流程

需確認前一階段工作完成後，再進行下一階段。而進入下一階段後，就很難回頭。工作階段的定義明確且嚴謹，系統開發須遵守流程的規範，不允許跳躍或更改其中的步驟。各開發階段，如需求分析、系統設計、實作與單元測試、整合與測試、運行與維護等，跟隨著開發步驟依序進行，概念見圖 2.1 所示。

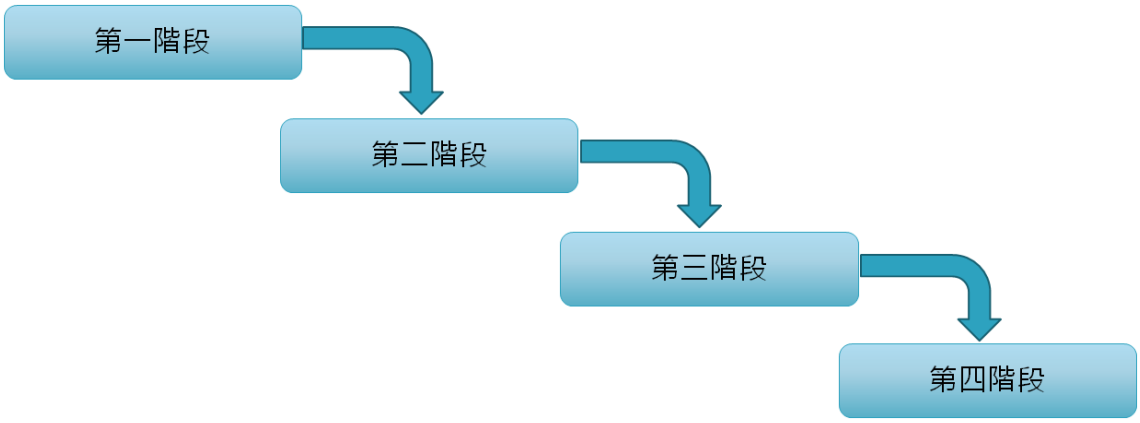

圖 2-1 線性發展流程概念圖

2. 多循環發展流程

不需完成一個階段之後，才能進入下一個階段；所有的階段都可被重新進入。流程的核心思想，是要將有瑕疵的問題在可能導致災害之前，提早揭露出來。多循環發展流程可以被不斷地反覆進行，概念見圖 2.2 所示。

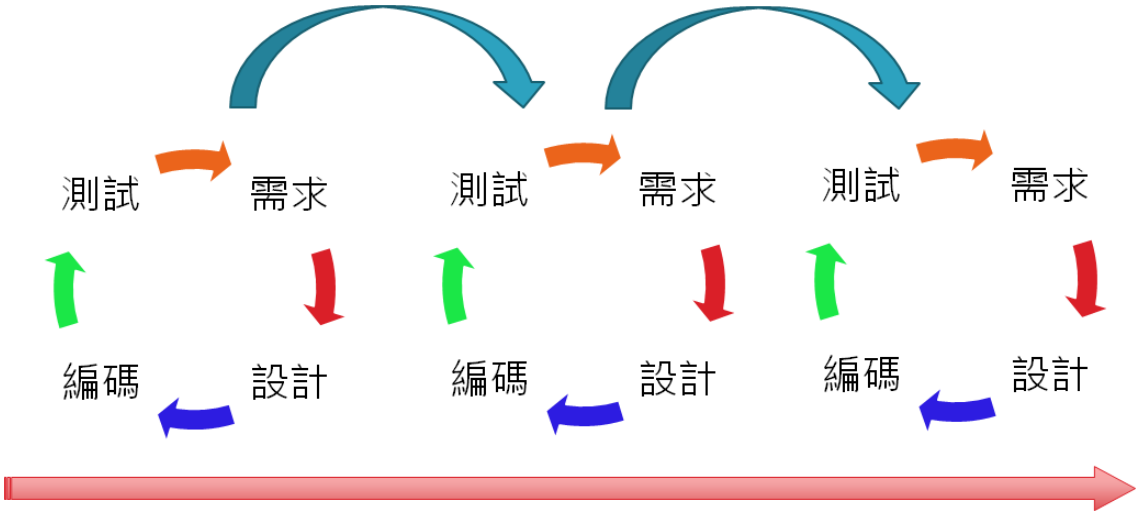

圖 2-2 多循環發展流程概念圖

2. 軟體開發模型

選用適當的軟體開發模型可帶來許多優點，如：提高開發速度、改善軟體品質、降低風險、專案追蹤與控管、改善顧客關係提昇滿意度等。傳統常見的軟體開發模型有瀑布模型、漸進式模型、雛型模型、螺旋模型等模型特性等，分類說明如下：

1. 瀑布模型(Waterfall Model)

瀑布模型於 1970 年由 Royce 所提出，為軟體開發最基礎也最常見的模式，瀑布模型是以活動為主的軟體開發模型，其軟體開發模式包含需求確認(Requirement Confirm)、系統分析(System Analysis)、系統設計(System Design)、系統測試(System Test)、實施運行(System Implement)，是典型的軟體生命週期，瀑布模型屬於線性發展流程，主張以循序方式開發出完整的軟體系統，讓各開發階段依序同向逐步往下一階段進行，每一階段完成時，皆會產出合適的文件，有助於專案的計劃與文件的製作，但模型一經執行如要再回到上一階段，會耗費更多的人力與成本，若開發過程中產物不符合使用者需求，則需回頭從需求甚至分析階段重頭開始，概念見圖 2-3。

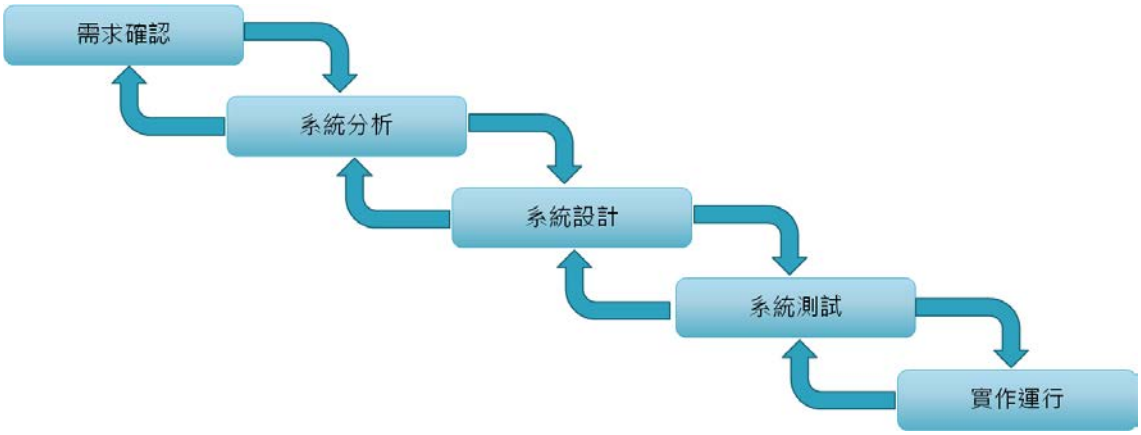

圖 2-3 瀑布模型概念圖

2. 漸進式模型(Incremental Model)

漸進式模型於 1971 年由 Mills 所提出，是將軟體系統分割成數個獨立的子系統，每個子系統再逐一進行開發。也就是將專案開發時程分割成數個階段，在每個階段採以完整的瀑布模型進行開發，再將陸續完成的子系統整合起來，其開發成果較易顯現，有助於依階段性產出結果決定開發是否繼續或改變開發方向。漸進式模型以零錯誤的軟體開發為目標，因此在每個漸進的週期中會不斷的反覆分析與修正需求內容，因此會投入較多的時間成本在分析與調整需求，容易影響專案整體的開發時程，概念見圖 2-4 所示。

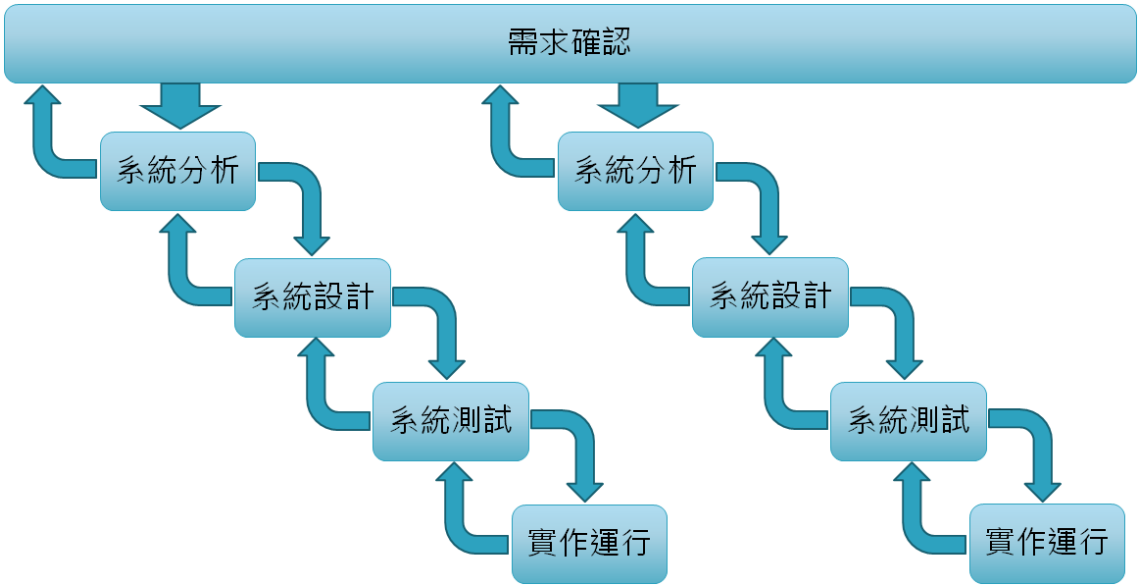

圖 2-4 漸進式模型概念圖

3. 雛型模型(Prototyping Model)

雛型模型於 1977 年由 Bally 等人提出，主要是依使用者需求快速地建置出一個具體化可暫時運轉的軟體系統，讓使用者看得到也可操作的系統，藉以及早確認客戶實際想要的軟體系統開發需求(鄭炳強，2007)，或獲得意見回饋與需求更改之資訊。系統開發初期，開發者依據使用者簡略的需求與規格，未經設計與實作，即能在短時間內於測試環境中建置出具像的軟體系統、功能介面與輪廓，透過靜態畫面顯示結果與使用者進行溝通，預先讓使用者了解未來的系統操作介面、頁面配置與操作流程，使需求與系統功能規格能更具體被提出討論，提高軟體系統開發的正確性。

雛型模型開發軟體系統的優點是(1).可以充分瞭解使用者的需求、(2).允許使用者隨時改變需求、(3).協助使用者發現新的需求、(4).快速的系統發展，降低風險；缺點是(1).使用者需時時參與，(2).缺少嚴謹的分析與設計，所以系統執行效率差，(3).雛型不斷的修改且缺少文件製作的管理，故不易維護，(4).因需求變動時內容須同步更新或甚至回到原點等情況，導致資源管理困難，產出品質不佳。其概念見圖 2-5 所示。

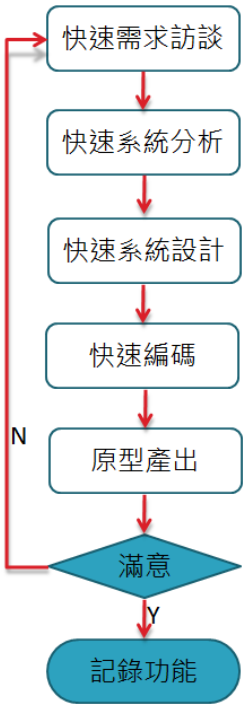

圖 2-5 雛型模型概念圖

4. 螺旋模型(Spiral Model)

螺旋模型於1988年由Boehm提出，專案實行過程依螺旋方式進行開發，持續不斷進行分析、設計、執行與測試等階段，在每個重複循環的過程中以排除專案風險為目標，螺旋模型強調落實風險管理，注重風險分析，並利用每次的循環將風險逐一排除(鄭炳強，2007)，可降低軟體專案開發可能遭到風險的機率或影響程度，因此，螺旋模型適用於需求不確定或目標不清，以及風險較高的軟體專案。其次，螺旋模型也結合雛型模型的快速發展觀念，注重階段性的測試與修正依需求開發之功能，強調在每完成一部份軟體功能就先上線運作，從一個概念產品不斷反覆演進到一個成熟的產品為止。螺旋模型可改善傳統瀑布模型易因需求變動而影響專案目標的缺點，但因需考量的專案因素非常廣泛及複雜，雖然投入的資源愈多，所承擔的風險就愈低，不過也相對提高了投入的成本，因此， 螺旋模型較不適用於小型或定義明確的軟體開發專案，較適用於大型而高風險的軟體開發專案。其概念見圖 2-6。

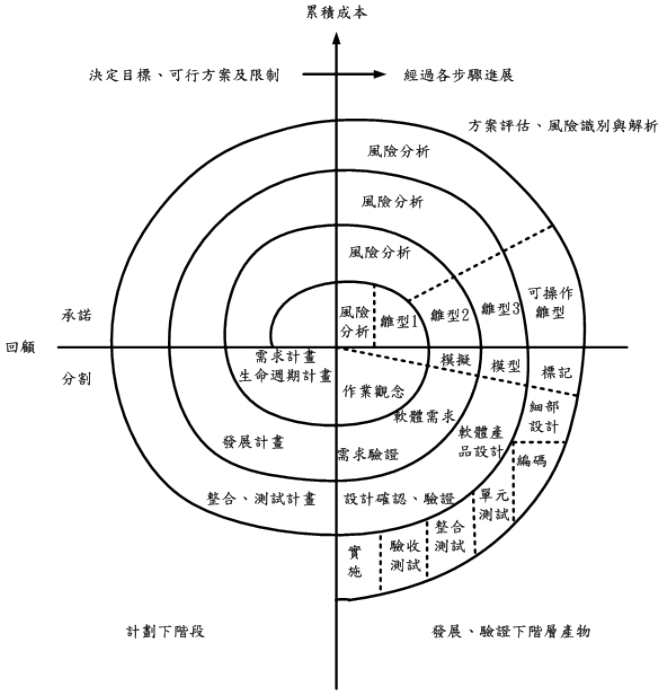

圖 2-6 螺旋模型概念圖

第三章　研究方法

第一節　研究流程

本研究實際採用 DKS 系統進行軟體專案開發，用以驗證使用 DKS 系統與傳統軟體開發方法之優劣比較。為了能正確地完成專案，採改良式雛型模型進行專案開發。本研究流程分別為 1. 確認研究主題與動機，2. 確認研究項目與問題，3. 文獻探討，4.專案確認，5.專案開發，6. 研究結果與分析，7.結論與建議，研究流程見圖 3-1 研究流程圖。

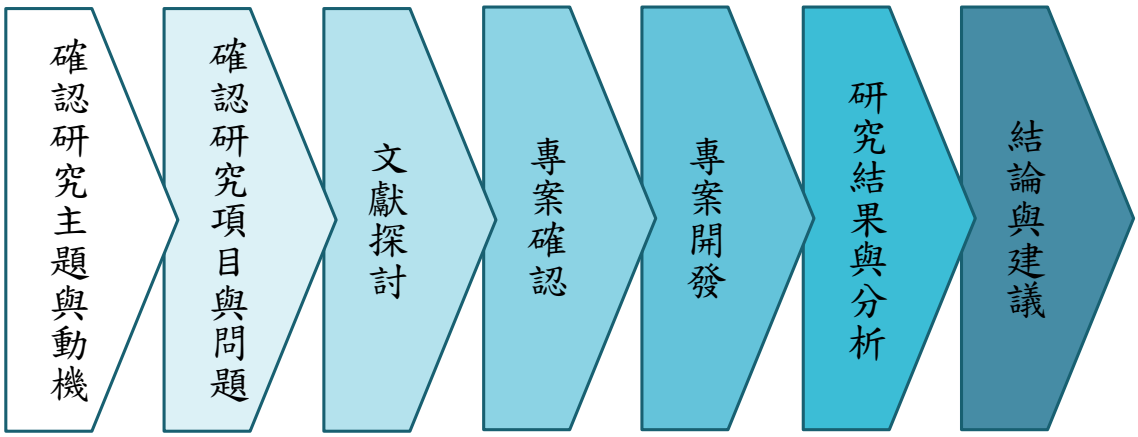

圖 3-1 研究流程圖

第二節　專案描述

本研究配合客戶是一家工程公司，該公司成立於民國 93 年，初設資本額為新台幣仟萬元，民國 104 年年營業額已增為新台幣 5 億元，其服務項目包含空調通風、無塵無菌、恆溫恆濕、節能工程、製程系統、低露除濕、工程設計、施工及維保等。

因客戶公司規模日益增加，原本以人工處理工程計價作業，包含採購發包、應收帳款、應付帳款等作業，因計算複雜，管控不易，經常延誤應收付帳款，甚至資料錯誤，造成客訴，加上人員異動，資料交接斷層，管理不易，因此期望能

將現有人工作業轉成以資料庫系統統一管理。

該客戶先評估市面上已有之套裝軟體，發現功能皆不符合公司需求，因此採客製化專案進行，同時客戶希望整個專案要在 3 個月內完成。

第三節　領域知識平台

領域知識平台是一套提供使用者進行需求設計、規格發行、運行應用之整合性開發環境，以可理解的文字描述規格及功能進行規格參數定義，完成規格參數定義後，經檢錯無誤後即可發行並產生程式碼、規格文件和可運行之應用系統。使用領域知識平台可分為需求設計、規格發型和運行應用，分別說明如下：

一、需求設計

需求設計共分為四個階段，分別流程規劃、表單設計、邏輯設定、打樣測試。

(一) 流程規劃：主要工作項目為確認專案事件及事件內之作業流程及程序步驟，設計介面見圖 3-2 流程規劃介面。

圖 3-2 流程規劃介面

(二) 表單設計：提供 UI 元件供使用者設計表單/報表畫面，並針對畫面元件進行元件屬性設定，UI 元件包含文字標題、文字方塊、多行文字、按鈕選項、核取方境、下拉選項、多筆表格、欄位按鈕、圖片、彈出視窗、清單選項、樹狀清單、頁簽區塊、框線、樞紐元件、連結框線、嵌入物件、功能按鈕、功能選單，設計介面見圖 3-3 表單設計介面。

圖 3-3 表單設計介面

(三) 邏輯設定：邏輯設定的主要工作項目說明如下，設計介面見圖 3-4 邏輯設定介面。

1. 表格定義：依功能分類分別有實體表格(Table)加註及邏輯表格(View)加註。
2. 欄位定義：依欄位功能需求加註，分別有基本設定、顯示設定、預設給值、選項清單、開窗參照、檢控限制、更新給值、編輯能力、樹狀控制、被動更新等。
3. 按鍵定義：依按鍵功能需求加註，分別有基本設定、執行限制、開啟它單、開啟報表、數值影響(過帳)、資料交換、郵件發送、資料載入、資料過濾、外部執行、溝通訊息、特殊處理等。

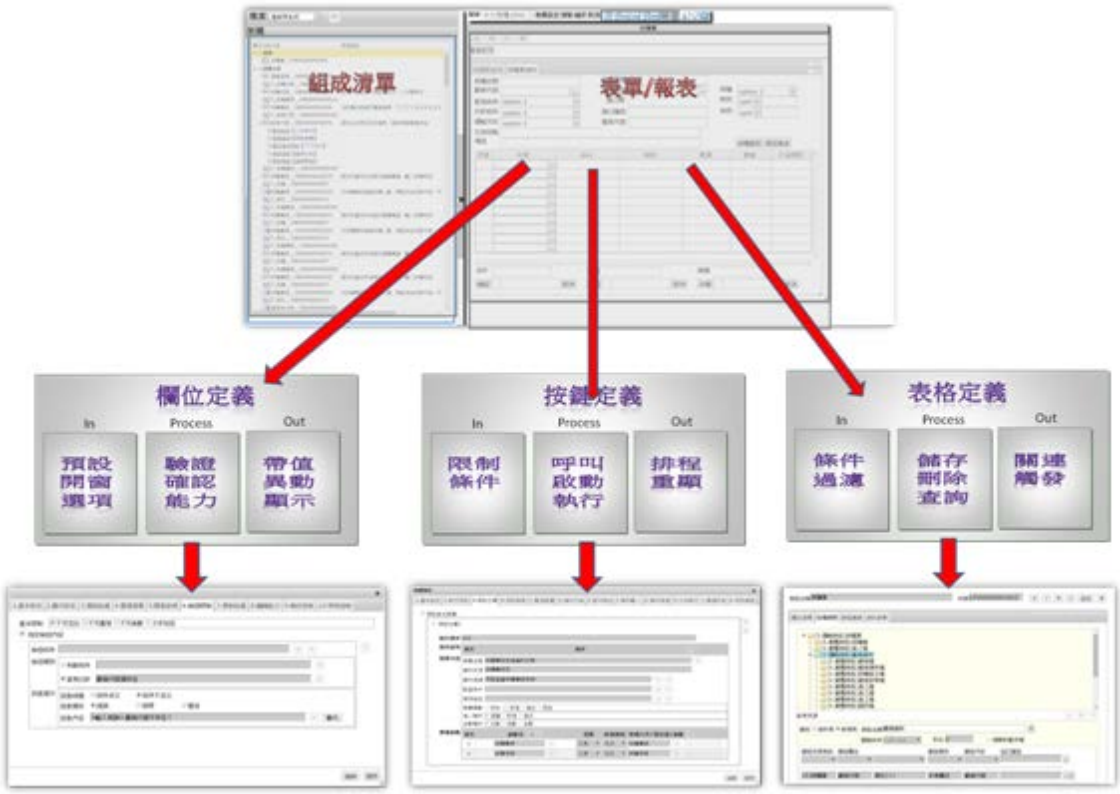

圖 3-4 邏輯設定介面

(四) 打樣測試：邏輯設定完成可先打樣產出可運行表單，實際操作測試規格是否定義正確，並以此樣品與客戶確認是否符合需求，設計介面見圖 3-5 打樣測試介面。

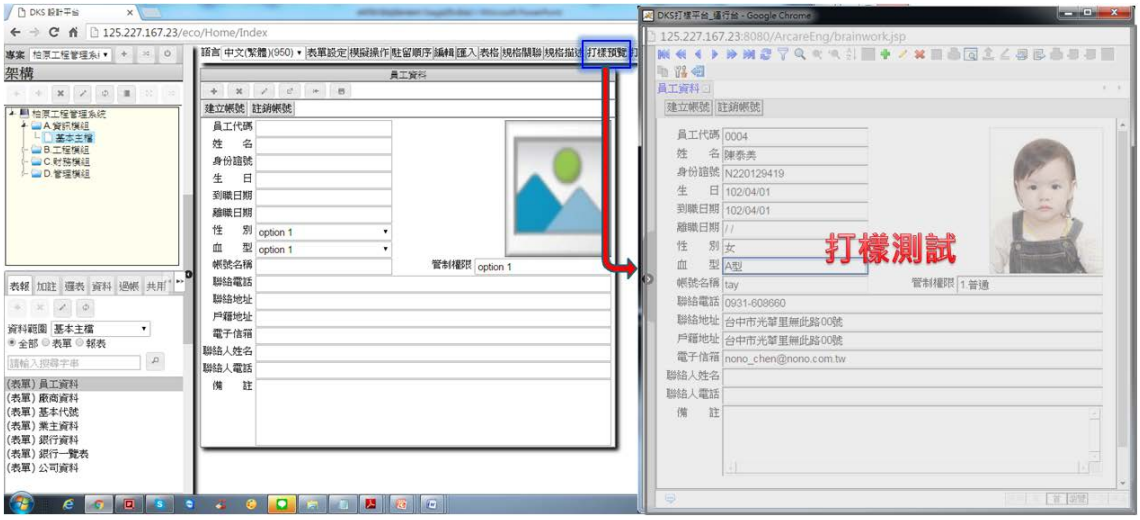

圖 3-5 打樣測試介面

二、規格發行

(一) 版本發行：DKS 系統會記錄每一次發行的規格調修紀錄，也就是工程變更通知(Engineering Change Notice)，其目的是為了在系統開發過程中，有規律地保留規格變更的歷史資訊，讓人能安心地做各種開發測試，並在開發不幸走進死胡同時，將系統恢復到舊有版本的系統，設計介面見圖 3-6 版本發行介面。

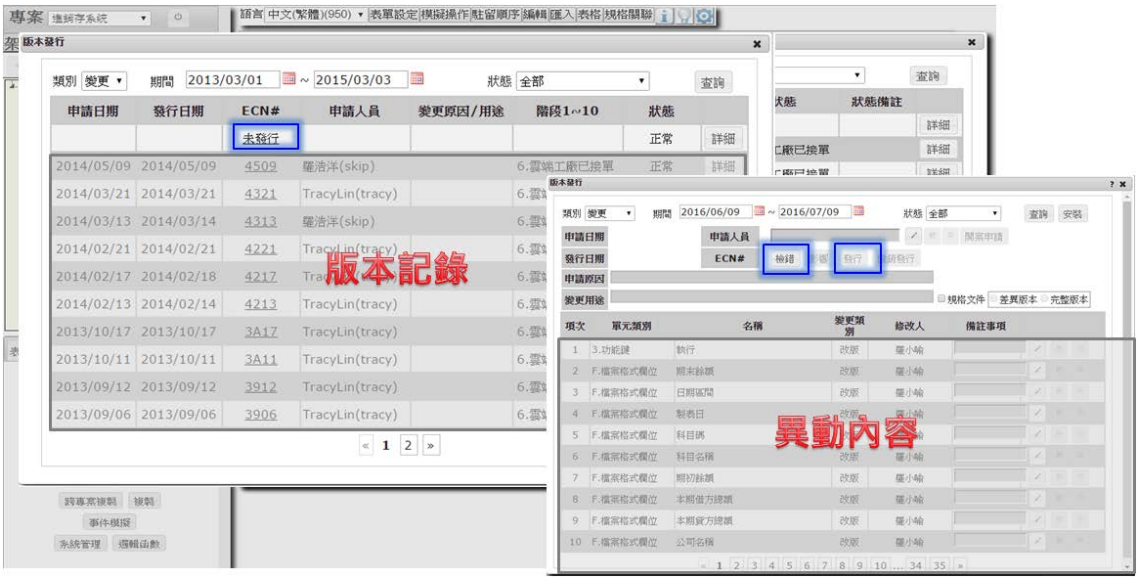

圖 3-6 版本發行介面

(二) 規格產出：DKS 系統可依使用者定義的規格產出系統規格書(System Specification)、系統設計書(System Design Document)，也可產出 KXML 之程式碼，由 DKS 系統產出文件與程式碼，可保持規格與最終交付給客戶的系統的一致性，增加未來系統的可維護性，設計介面見圖 3-7 規格產出介面。

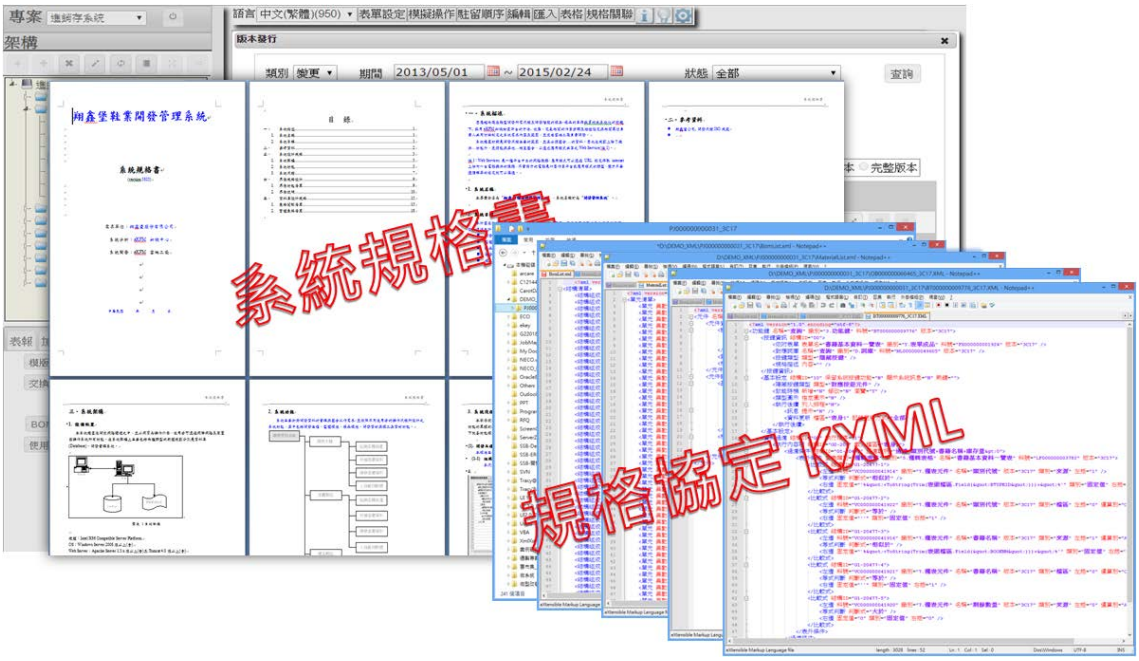

圖 3-7 規格產出介面

三、運行應用

DKS 系統依規格定義產出可運行應用系統，分別有主選單、系統按鍵、自訂按鍵、應用表單等部件，運行介面見圖 3-8 運行應用介面。

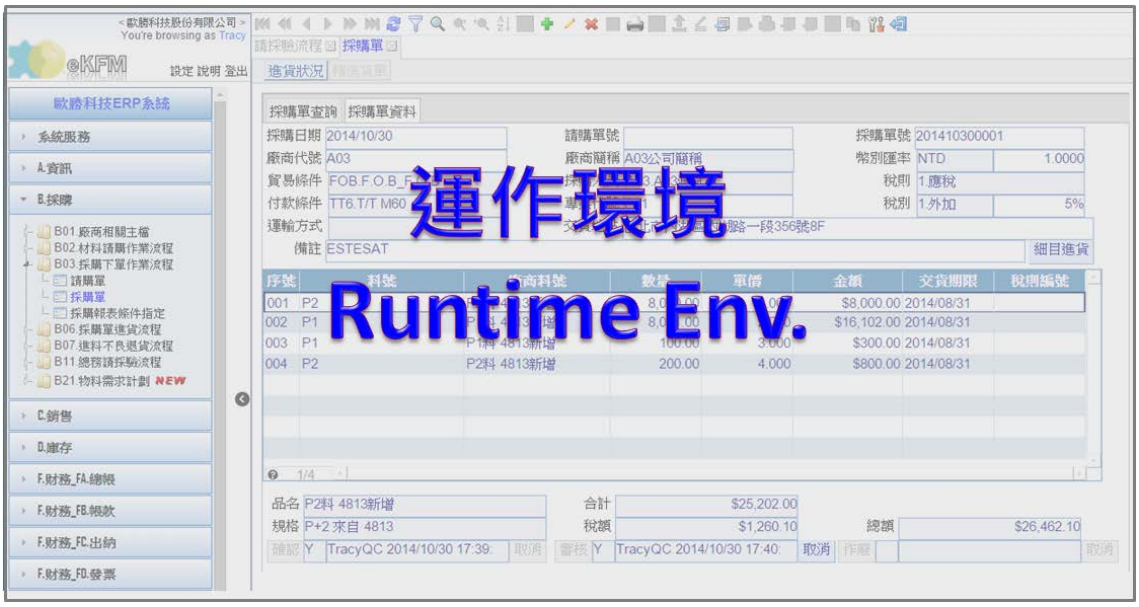

圖 3-8 運行應用介面

第四節　改良式雛型模型

雛型模型可快速產出雛型供使用者確認，使用者在開發初期就能預期最終產品的功能，是最貼近使用者需求的一種開發模型。但因缺乏有效的工具，無法嚴謹的分析與設計，缺少文件製作的管理，不易維護。DKS 系統以知識導向的開發方式，不必撰寫程式，具備模擬操作的功能，可由系統將規劃的規格產出成管理文件的特點，恰好可解決雛型模型的缺點。改良式雛型模型之執行步驟為 1.快速進行需求訪談，2.確認規格後便進行系統分析，設計流程與表單畫面，3.此時可利用模擬操作的功能，與使用者進行需求確認或提早調修，4.確認可行後再進行系統設計，5.設計完畢即可打樣產出樣品供使用者實際進行測試操作，開發過程中減少程式撰寫這個階段，概念見圖 3-9。

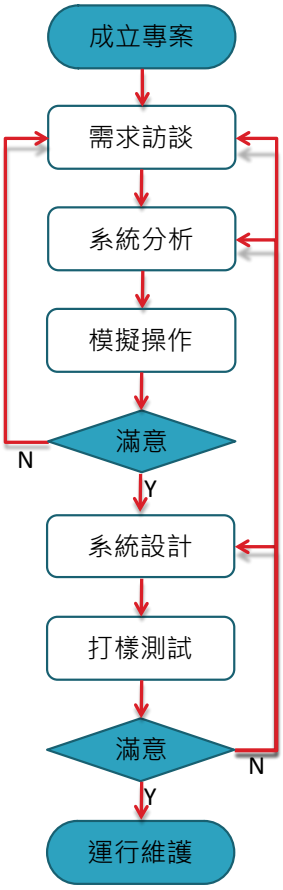

圖 3-9 改良式雛型模型概念圖

第五節　名詞釋義

本章節針對 DKS 系統裡常用之專有名詞進行釋義，說明如下表：

表 3-1 DKS 系統專有名詞釋義

專有名詞	名詞釋義
實體表格(Table)	SQL 資料庫中之資料表(Table)，實際存在資料庫的資料。
邏輯表格(View)	SQL 資料庫中之檢視表(View)，以實體表格或是別的邏輯表格為基礎，用一定邏輯關聯組成之表格。
專案(Project)	相當於系統，如：ERP/HRM/CRM . . . .
事件(Event)	任何想要被管理的事情，都可以稱為一個事件，例如請假作業、零用金管理……等。
事件類別(Class)	事件的上一階，也就是事件的資料夾，內含相關的事件。相當於作業模組，如：庫存管理、生產管理、銷貨管理. . . . .
過帳(DB update)	執行一個功能，以一定邏輯影響(增刪修)資料庫中的實體表格資料的行為。
加註(Spec.)	針對任何組成系統的元件做規格描述。
表單元件(Element)	表單畫面上的欄位及物件(包含隱藏欄位)。
按鍵(Function-key)	又稱功能鍵，包含系統功能鍵(基本的新增、修改、刪除、匯出入、列印)及自訂功能鍵。

資料來源：本研究整理

第四章　研究結果與分析

本專案執行步驟依研究流程進行分別為需求訪談、系統分析、雛型產出、系統設計、打樣測試等階段。

第一節　需求訪談

需求訪談的目的在於透過與使用者訪談互動過程中，詳細了解使用者的作業流程內容與執行範圍，未來在系統分析階段，能將使用者腦海中較為抽象而籠統的使用需求概念，以正規化的規範語言方式，直接將這些需求轉換成系統功能的作用。本階段完成確認 1.系統架構，2.系統作業流程，3.專案組織，4.專案時程。

一、確認系統架構

經訪談後確認系統採 3-Tier 架構，建置雲端主機，供總公司、各分公司及各工地能透過 Internet 即時建立/查詢資料，資料庫採用 MS-SQL，架構圖見圖 4-1 系統架構圖。

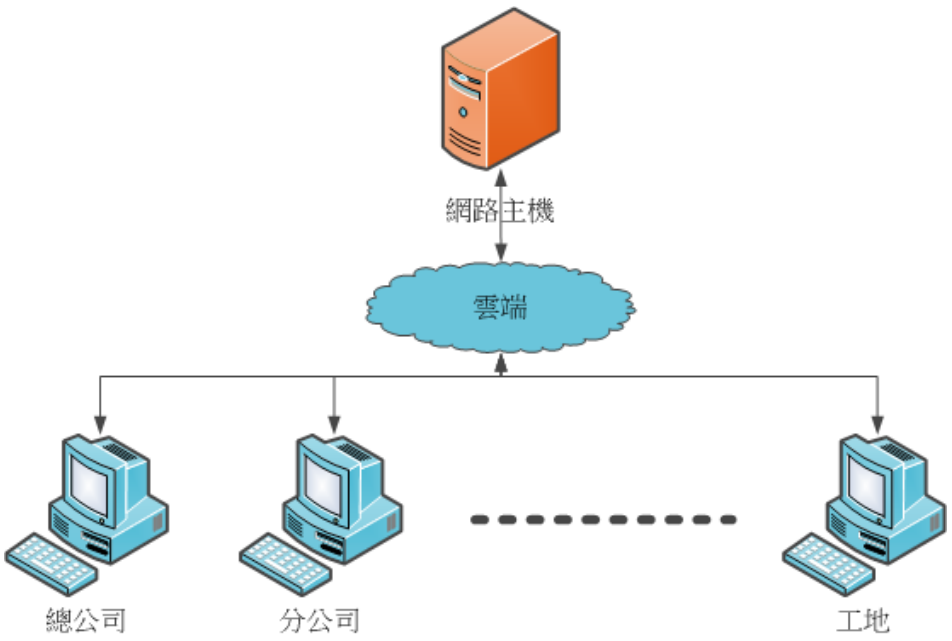

圖 4-1 系統架構圖

二、確認系統作業流程

了解並釐清該公司標準作業流程及作業管控重點，該公司主要以工程專案為管理重點，包含工程合約、工程預算、工程發包、工程計價，應付帳款、工程請款、應收帳款、應收付票據及整合性管理報表，作業流程見圖 4-2 系統作業流程圖。

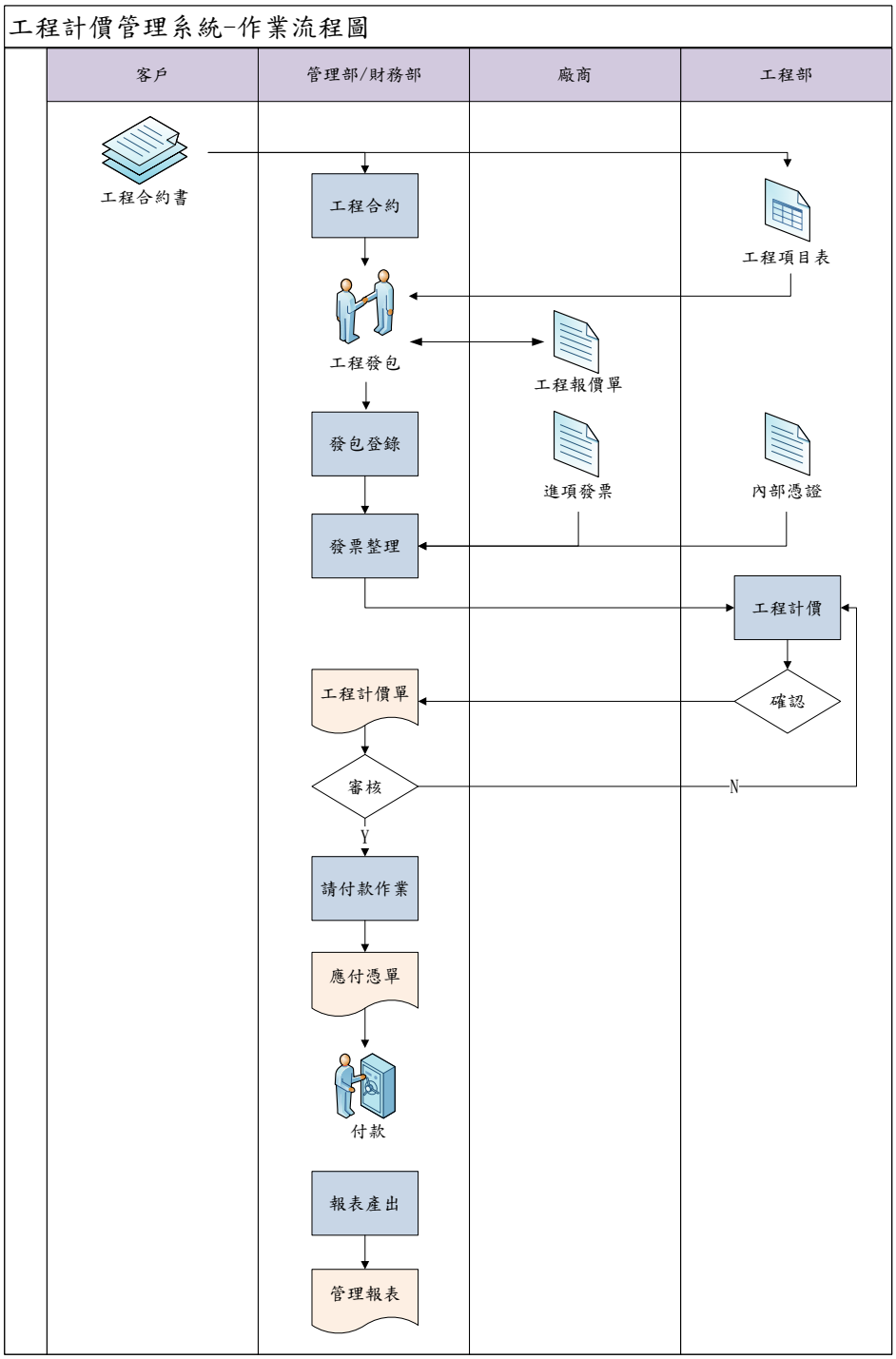

圖 4-2 系統作業流程圖

三、確認專案組織

建立專案組織，並明定每一專案成員的工作職掌，客戶端設專案召集人，主要負責需求確認，下設三名使用者代表分別負責採購發包、應付帳款、應收帳款之需求提出與系統測試；廠商端設廠商代表，主要負責需求接收與系統分析，下設一名知識助理，負責依系統分析結果進行系統設計，組織圖見圖 4-3 專案組織圖。

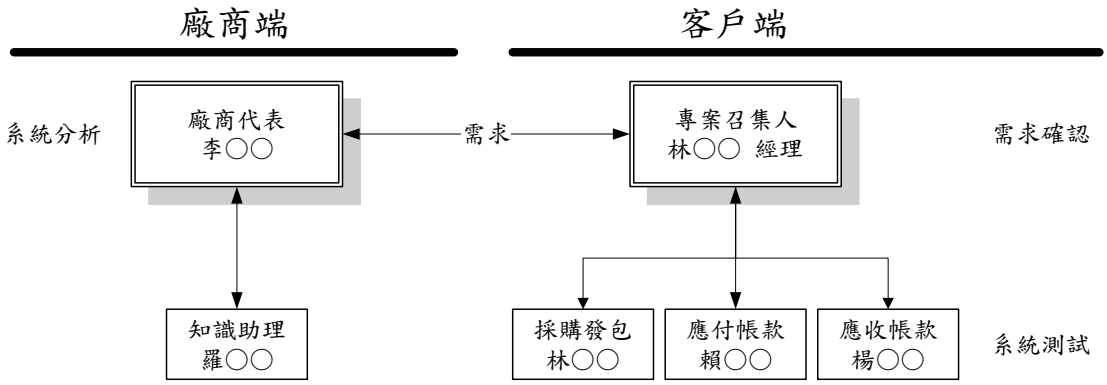

圖 4-3 專案組織圖

四、確認專案時程

與客戶協議 3 個月完成系統開發與建置，細部時程見圖 4-4 系統開發時程表。

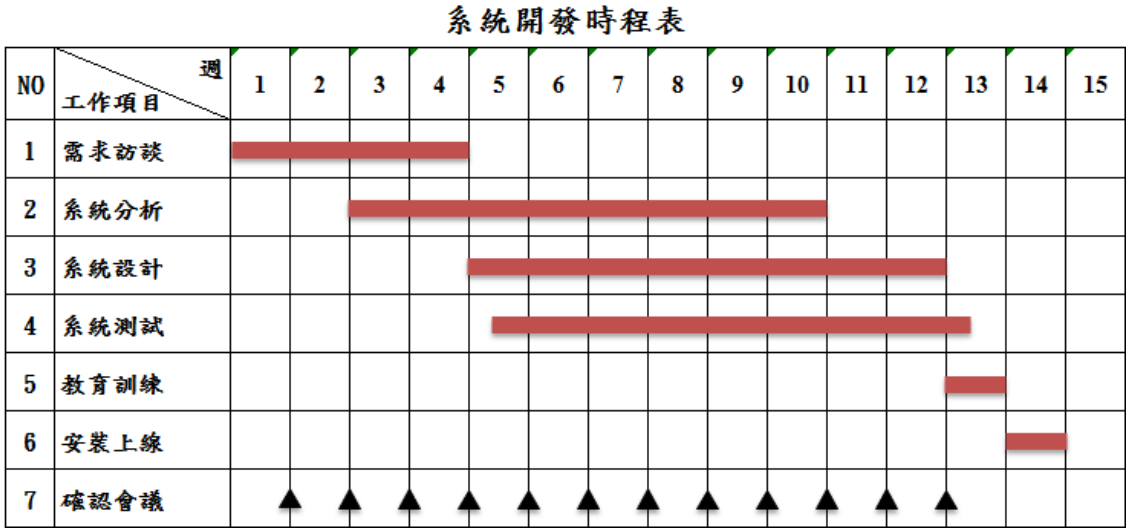

圖 4-4 系統開發時程表

第二節　系統分析

本階段依需求訪談確認之作業流程圖規劃程式流程圖，提出系統規格，包含系統模組、流程架構、畫面初步規劃、功能鍵等。

一、程式流程圖

根據作業流程需求規劃專案模組事件及事件內之作業程序與表單，共分為管理模組、工程模組、財務模組、管理模組，初步規劃見圖 4-5 專案程式流程圖。

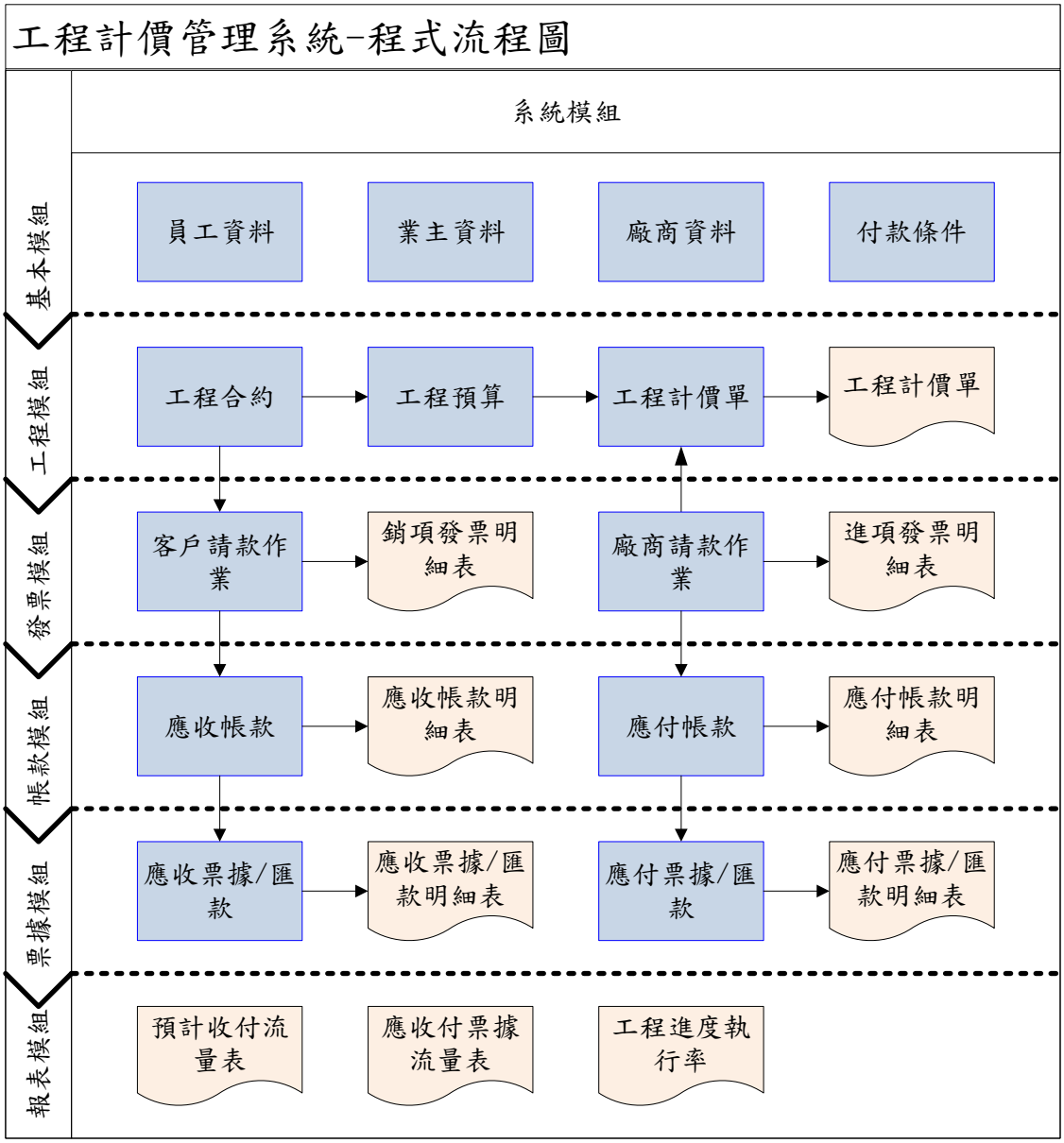

圖 4-5 專案程式流程圖

二、表單畫面設計

採用 DKS 系統進行畫面設計，操作步驟如下：

(一) 建立事件類別及事件(見圖 4-6)

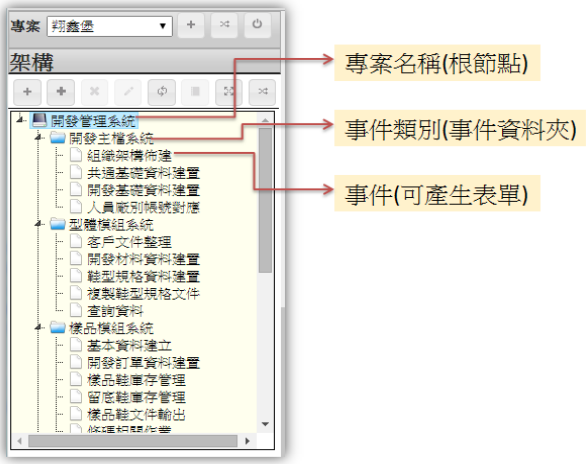

圖 4-6 事件類別及事件

1. 於任何節點按滑鼠右鍵→新增→新增類別，可於節點下新增事件類別。
2. 於任何節點按滑鼠右鍵→新增→新增事件，可於節點下新增事件。
3. 於事件類別或事件節點按滑鼠右鍵→刪除，可刪除該節點及節點以下的資料。

(二) 建立事件流程及表單(見圖 4-7)

圖 4-7 事件流程及表單

介面欄位說明：

1. 使用「平行」，可產生分支線條的起點。
2. 使用「分支」，可產生分支線條。
3. 使用「流程」，可產生紅色方塊，做為流程收斂節點。
4. 使用「單據」，可產生表單。

(三) 繪製表單畫面(見圖 4-8)

進入編輯狀態、開啟元件清單後，可將元件清單內的各種元件拖曳到畫面區。

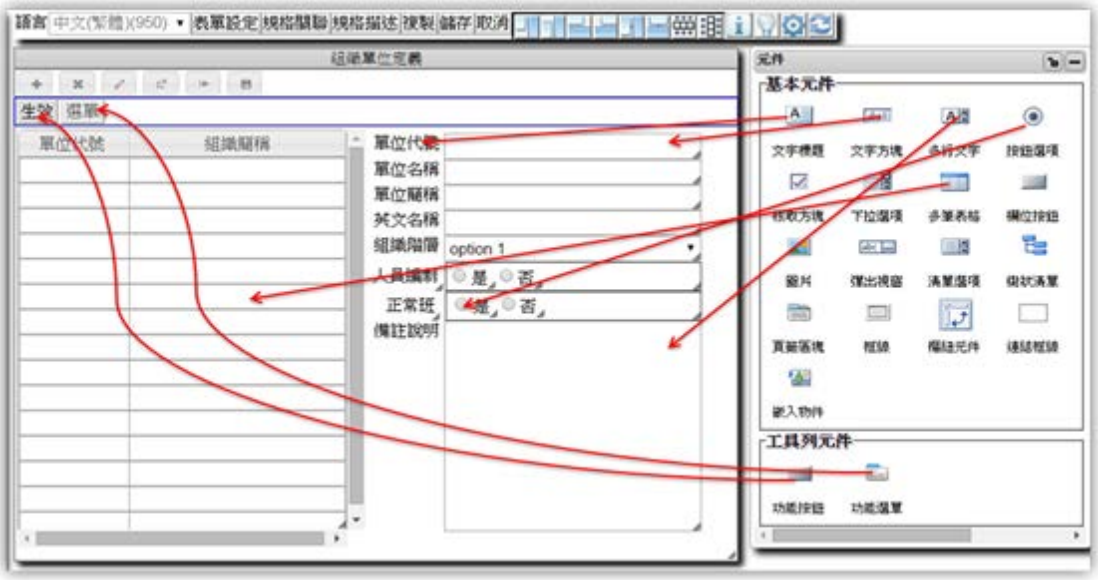

圖 4-8 繪製表單畫面

工具列說明：

1. 表單設定：設定表單的基本參數。
2. 複製：可複製別張表單的畫面。
3. 對齊工具組：用來對齊表單畫面元件。
4. 元件清單按鍵：開啟元件清單。
5. 儲存：存檔用。

(四) 元件設定(見圖 4-9)

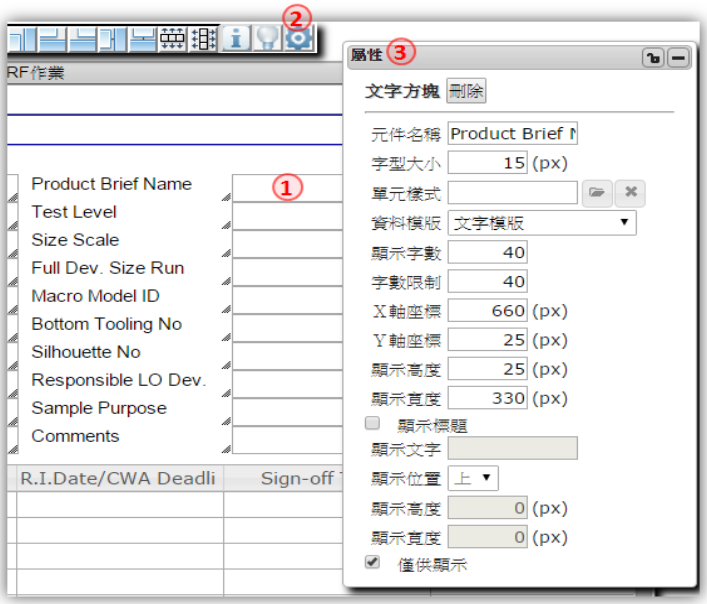

圖 4-9 元件設定

1. 駐留在任何一個文字方塊元件上，畫面右側會顯示屬性視窗。
2. 此按鍵可開啟/關閉屬性視窗。
3. 屬性視窗中可為該元件設定各項參數。

(五) 駐留順序設定(見圖 4-10)

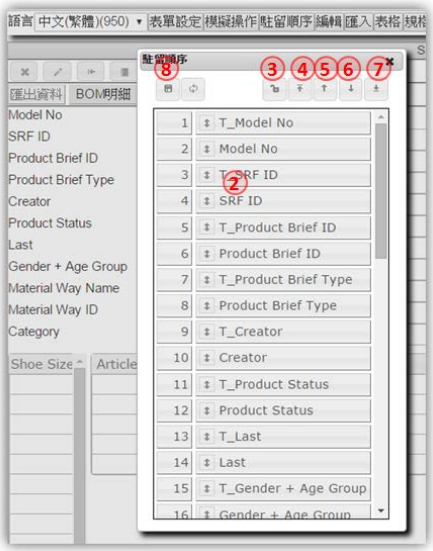

圖 4-10 駐留順序設定

1. 將繪製好的畫面存檔後，按下駐留順序按鍵，可顯示駐留順序的設定視窗。
2. 在此區塊中，可任意拖曳元件到指定的駐序位置。若點選兩個以上元件，元件上會出現鎖定符號，此時可一次拖曳多個元件。
3. 此按鍵可將下方鎖定的元件組解鎖。
4. 將下方駐留的元件移到第一個駐序。
5. 將下方駐留的元件移往上移一個駐序。
6. 將下方駐留的元件移往下移一個駐序。
7. 將下方駐留的元件移到最後一個駐序。
8. 存檔。
9. 不同語言版本的畫面、駐序是分開的，請記得調整所有語言版本的駐序。

三、按鍵設定

採用 DKS 系統進行按鍵設定，操作步驟如下：

(一) 系統按鍵設定(見圖 4-11)

圖 4-11 系統按鍵設定

1. 在表單畫面的編輯狀態下，點選選單列，可顯示系統功能按鍵的屬性視窗。
2. 勾選需要的系統功能，選單列會自動顯示對應的按鍵圖示。

(二) 自訂按鍵設定(見圖 4-12)

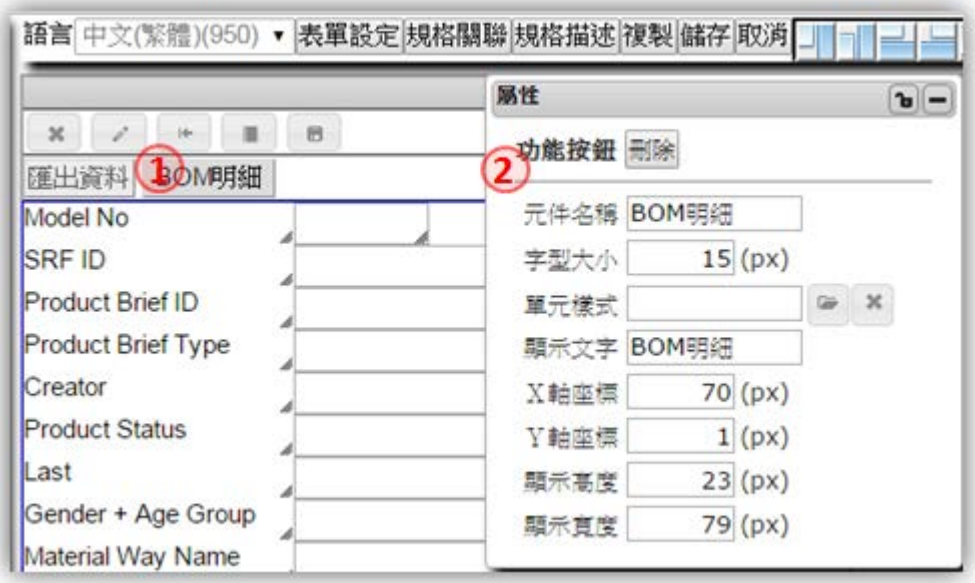

圖 4-12 自訂按鍵設定

1. 在表單畫面的編輯狀態下，點選自訂按鍵，可顯示該按鍵的屬性視窗。

2. 此處可設定按鍵的屬性，包含按鍵名稱、顯示的文字、按鍵的長寬及位置。

(三) 自訂選單按鍵設定(見圖 4-13)

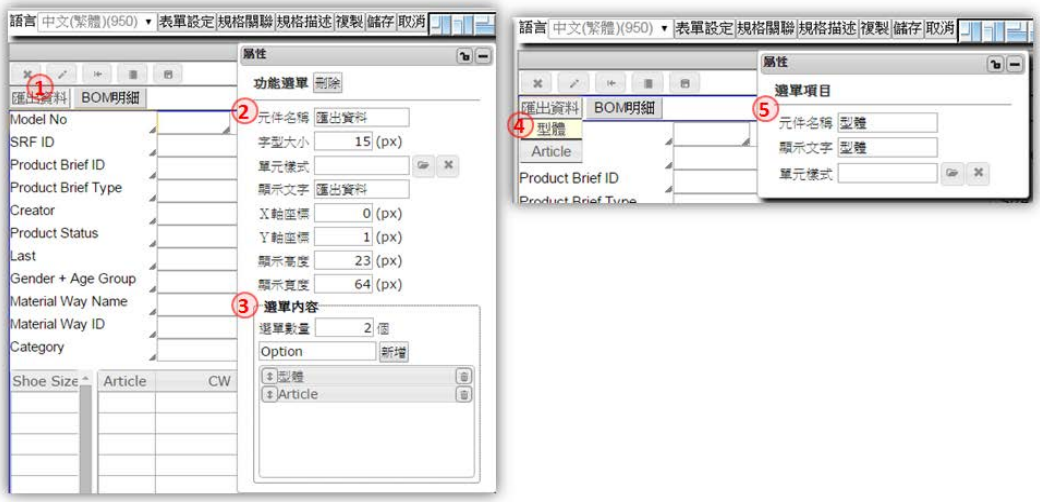

圖 4-13 自訂選單按鍵設定

1. 在表單畫面的編輯狀態下，點選自訂選單，可顯示該選單的屬性視窗。
2. 此處設定該選單的名稱、顯示文字、大小、位置。
3. 此處可設定選單內要有幾個按鍵。
4. 點選選單下的按鍵，會顯示該按鍵的屬性視窗。
5. 針對選單下的單一按鍵設定它的名稱和顯示文字。

四、元件規格描述

採用 DKS 系統進行元件規格描述，操作步驟如下：

(一) 一次描述多個元件規格(見圖 4-14)

執行規格描述按鍵，可開啟下方之規格描述介面。

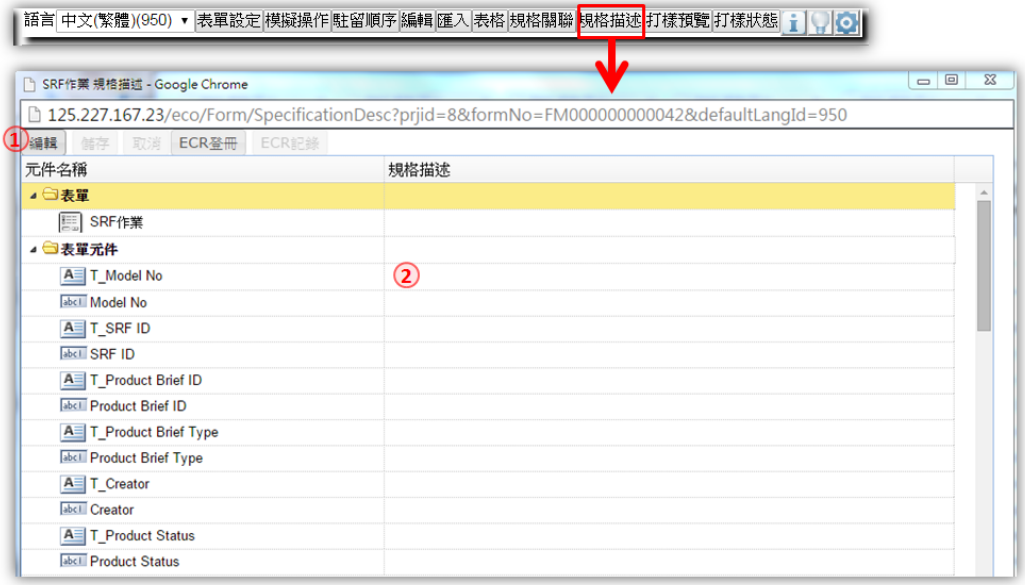

圖 4-14 一次描述多個元件規格

1. 按下編輯按鍵，可進入修改模式。
2. 在每個元件上可白話描述該元件的規格。

(二) 單一元件規格描述(見圖 4-15)

圖 4-15 單一元件規格描述

1. 雙擊元件名稱，會開啟該元件的規格標註視窗。
2. 描述欄位的內容與規格描述欄位內容是一樣的，可自行描述。
3. 此區塊可勾選、挑選、輸入常用的幾種規格描述。
4. 重組按鍵可將上一步驟挑選的規格串起來，放到描述欄位中，每次重組都會覆蓋之前的內容。
5. 說明欄位可做簡單的附註說明，只可輸入 255 個字元。
6. 附件備註中可填寫較複雜的規格，例如案例、上傳附件等等。
7. 設定完畢請存檔。

第三節　雛型產出

畫面設計完成後可經由模擬操作，產出雛型與客戶確認表單功能，經由模擬畫面說明與操作，客戶比較有感，此時可以再提出需求調修或新的需求以降低系統開發後期的返工率，DKS 系統可進行模擬操作，操作步驟見圖 4-16：

圖 4-16 模擬操作

1. 在功能選單中，點選模擬操作按鍵。
2. 開啟駐留表單的模擬操作視窗，此時尚未進行資料庫數計，因此只能進行資料模擬。

第四節　系統設計

本階段依據系統分析所訂定的規格，進行系統功能細部開發。

一、實體表格設計

採用 DKS 系統進行實體表格設計，操作步驟如下：

(一) 新增實體表格(見圖 4-17)

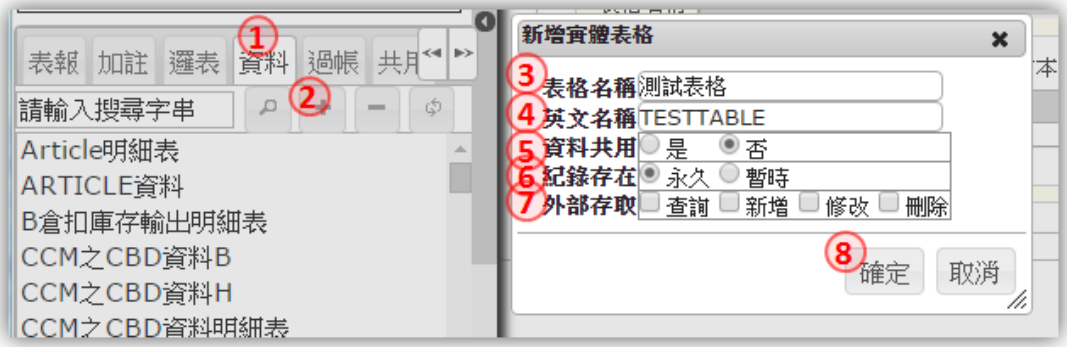

圖 4-17 新增實體表格

1. 點選畫面左下區塊的「資料」頁籤。
2. 點選新增鍵新增實體表格。
3. 設定表格名稱，此名稱在 DKS 系統管理規格用。
4. 設定表格英文名稱，此名稱為 SQL 資料庫中正式建立 TABLE 的名稱。
5. 設定此實體表格是否為跨資料庫共用。
6. 記錄存在若選取永久，則為資料庫中的 TABLE；若選取暫時，則為輸出資料時暫存用的 MDB 檔。
7. 設定外部程式是否可存取此專案資料庫的實體表格。
8. 設定完畢後，按確定，實體表格新增完成。

(二) 新增實體表格欄位(見圖 4-18)

圖 4-18 新增實體表格欄位

1. 按新增鍵新增欄位。
2. 輸入欄位名稱，此名稱為 DKS 系統管理用。
3. 輸入結構命名，此名稱為 SQL 資料表中的資料行名稱。
4. 設定資料型態，如文字、數字、日期等等，其中日期、日期時間在 SQL 中都會開成 date time 型態。
5. 設定欄位長度。
6. 設定初始值，一般可不用設定。
7. 設定是否允許空值(NULL)。
8. 設定此欄位是否為鍵值(Primary key)。
9. 設定後請存檔。

(三) 設定索引(見圖 4-19)

圖 4-19 設定索引

1. 點選此頁籤。
2. 新增索引。
3. 輸入索引名稱。
4. 確定存檔。
5. 挑選要加入索引的欄位。
6. 加入。
7. 指定此欄位的生降冪。
8. 存檔。
9. 挑選要變更順序的欄位。
10. 可上下移動順序。
11. 可改變升降冪。
12. 可刪除駐留的索引。
13. 可修改駐留的索引名稱。

(四) 連續觸發(Trigger)設定(見圖 4-20)

圖 4-20 連續觸發(Trigger)設定

這項功能可設定此實體表格的資料受到某種影響時，連帶影響其他實體表格的資料。設定方式：

1. 點選連續觸發頁籤。
2. 點選新增鍵新增資料。
3. 選擇觸發時機。(本實體表格資料受到何種影響時要觸發。)
4. 選擇目的表格名稱。(連帶要影響的實體表格)
5. 選擇異動方式。(要如何異動被連帶影響的表格)
6. 新增表身記錄。
7. 點選目的表格欄位名。(依據目的表格的哪個欄位異動目的表格的資料)
8. 若要符合條件才異動目的表格的資料，請勾選此欄位。
9. 選擇異動類別。(固定值：目的表格的欄位符合某個固定值就異動。本表欄位：目的表格的指定欄位值與本張實體表格的指定欄位值相符合才異動)
10. 輸入要對應的固定值或是本表欄位。
11. 設定完畢請存檔。

二、邏輯表格設計

邏輯表格相當於 SQL 資料庫中的檢視表(View)，是依據一定的邏輯連結、組合、運算一個或一個以上實體表格/邏輯表格之後，呈現出來的資料。邏輯表格資料不存放在磁碟中，而是存放在記憶體(RAM)中。

本階段採用 DKS 系統進行邏輯表格設計，操作步驟如下：

(一) 建立邏輯表格(見圖 4-21)

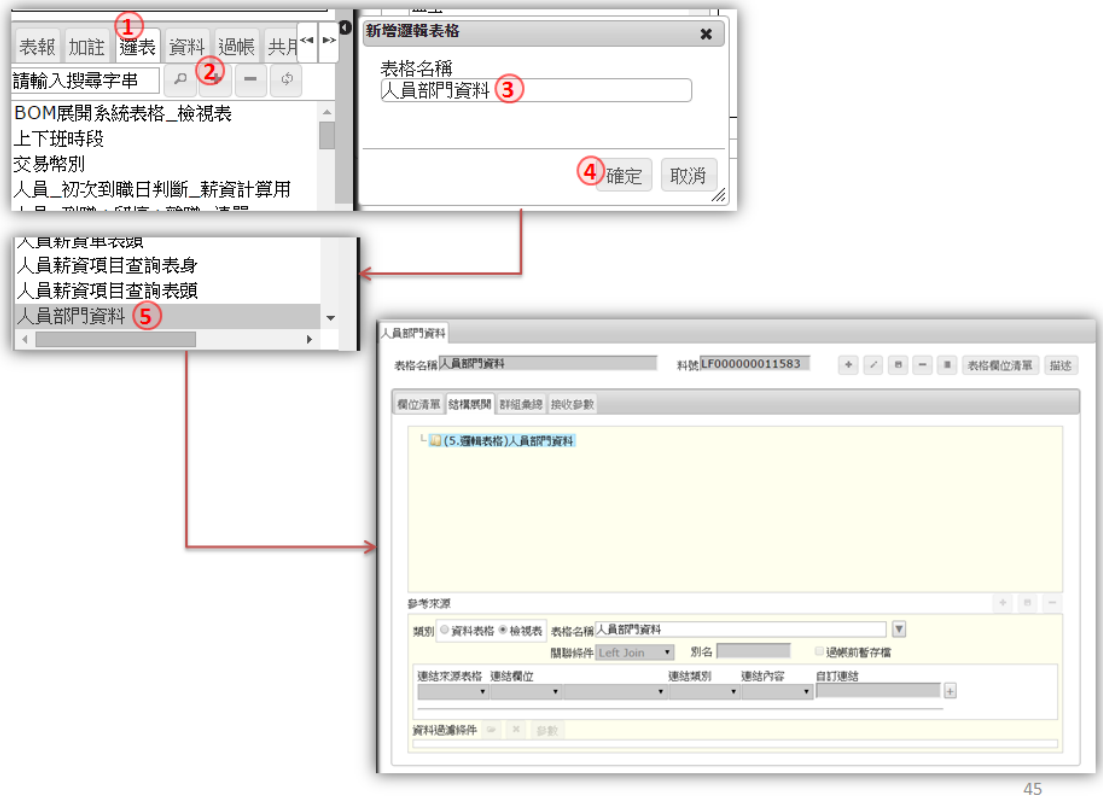

圖 4-21 建立邏輯表格

1. 於系統畫面左下方點選邏表頁籤。
2. 按新增鍵新增資料。
3. 輸入邏輯表格名稱。
4. 確定。
5. 於邏表清單中雙擊邏表名稱，即可開啟設定畫面。

(二) 設定主、副表(見圖 4-22)

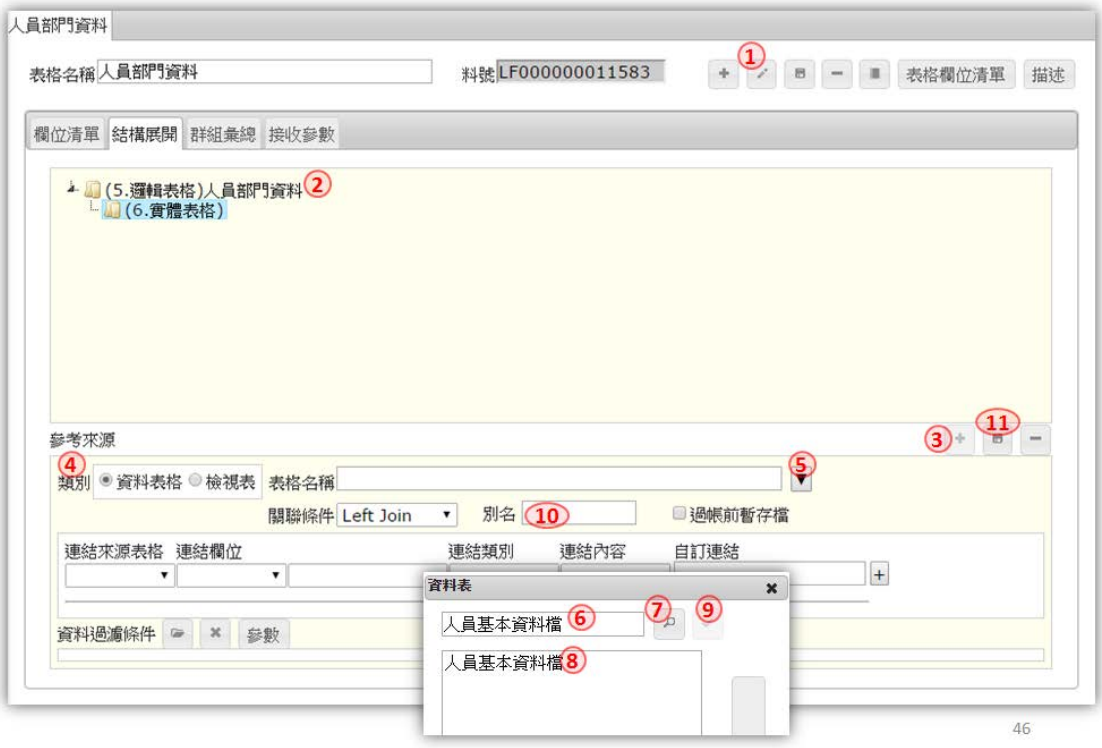

圖 4-22 設定主、副表

1. 按修改鍵進入編輯模式。
2. 駐留根節點。
3. 按新增鍵新增資料。
4. 選擇要加入的主/副表是資料表或檢視表。
5. 按此鍵開啟清單。
6. 輸入關鍵字。
7. 搜尋。
8. 駐留所要帶回的表格。
9. 按回傳鍵帶回資料。
10. 輸入別名，例如 A。
11. 存檔。

(三) 設定副表之關連條件(見圖 4-23)

圖 4-23 設定副表之關連條件

1. 駐留要設定關聯的副表。
2. 選擇要連結的主/副表。
3. 選擇連結來源表格中的連結欄位。
4. 選擇運算子。
5. 選擇連結類別，例如本表欄位、函數等。
6. 選擇連結內容，例如連結類別為本表欄位時，連結內容為本表欄位名稱。
7. 若前面幾個欄位的挑選無法達到規格要求，可自行以 SQL 語法寫在自訂連結中。
8. 設定完畢後按新增鍵即可新增完成。
9. 若連結的欄位不只一個，重複上述步驟即可設定多個欄位的連結。

(四) 設定主、副表之過濾條件(見圖 4-24)

圖 4-24 設定主、副表之過濾條件

1. 駐留欲過濾的主表或副表。
2. 開啟資料過濾條件。
3. 填寫過濾條件式。
4. 存檔。

(五) 設定輸出欄位(見圖 4-25)

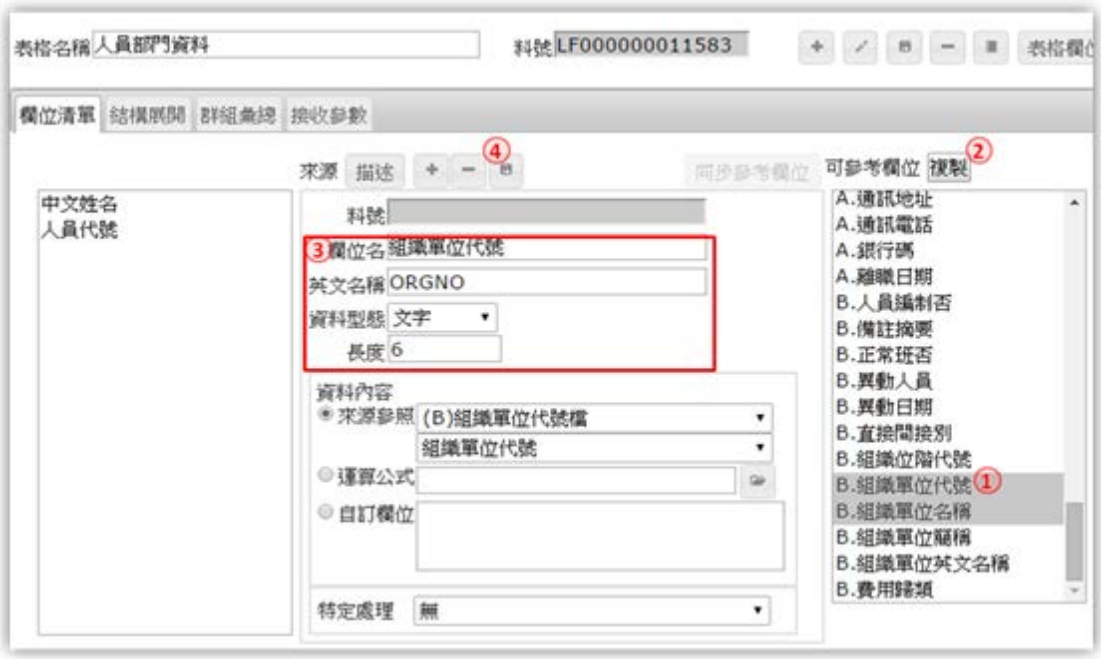

圖 4-25 設定輸出欄位

1. 點選要輸出的欄位，可配合 Ctrl、Shift 鍵選取多個。
2. 執行複製即可。
3. 若要改變輸出欄位的長度、名稱等等，可修改後再按存檔鍵
4. 存檔。

(六) 群組彙總設定(見圖 4-26)

圖 4-26 群組彙總設定

1. 選擇要設定的欄位。
2. 此鍵可將欄位加入群組(group by)。
3. 選擇已在群組內的欄位，再按此鍵可將欄位移出群組。
4. 此鍵可將欄位加入排序(order by)。
5. 選擇已在排序內的欄位，再按此鍵可將欄位移出排序。
6. 此處可限定撈出的資料筆數(Select Top x)。
7. 此選項可去除重複的資料(Select Distinct)。

(七) 邏輯表格接收參數設定(見圖 4-27)

圖 4-27 邏輯表格接收參數設定

邏輯表格可以設定接收參數，接收到的參數可以拿來過濾主、副表的資料，或是拿來參與自定義欄位的輸出。設定方式：

1. 進入邏輯表個的編輯模式。
2. 新增一個或多個參數。
3. 輸入參數名稱。
4. 設定參數的型態，有文字、數字、日期三種。
5. 設定完畢後請存檔。

三、表單加註

採用 DKS 系統進行表單加註，操作步驟如下：

(一) 基本設定(見圖 4-28)

圖 4-28 基本設定

1. 進入編輯狀態。
2. 設定運行環境的首頁選單在此表單開啟時是否要強制關閉或開啟，一般維持空白即可。
3. 設定表單的呈現方式，分別有單筆、多筆、雙檔、雙多筆、條件多筆、單改等。
4. 設定資料的呈現方式，開啟表單時要顯示最首筆或是最末筆資料。
5. 若表單的資料需定時從資料庫更新下載，可設定此選項。
6. 獨立開啟若勾選，代表表單會出現在運行環境的首頁選單中，若不勾選，代表表單的開啟是由其他表單呼叫。
7. 設定在單檔多筆表單中，是否可連續異動資料。

(二) 資料處理(見圖 4-29)

圖 4-29 資料處理

1. 進入編輯模式。
2. 挑選表頭資料的檢視表(邏輯表格)。
3. 若檢視表本身有設定接收參數，請按參數鍵設定參數的傳入。
4. 若表頭的資料在開啟表單時要事先過濾資料，可在此設定。
5. 新增表頭資料排序的欄位。
6. 挑選排序的欄位名稱。
7. 設定欄位的排序方式。
8. 此鍵可刪除排序的欄位。
9. 此鍵可上移/下移排序欄位。
10. 設定完畢請存檔。

(三) 元件對應(見圖 4-30)

圖 4-30 元件對應

1. 進入編輯模式。
2. 此鍵可自動將兩邊名稱相同的欄位對應起來。(左側為表單元件，右側為邏輯表格的欄位)
3. 若要手動對應，先選擇要對應的元件。
4. 選擇要對應的邏表欄位。
5. 此鍵可將兩邊欄位對應起來。
6. 此鍵可取消兩邊欄位的對應關係。
7. 設定完畢後請存檔。

(四) 前單互動(見圖 4-31)

圖 4-31 前單互動

若此表單是被他單開啟的，被開啟時會接收前單傳入的參數，可以在此設定接收參數名稱。若 A 表開啟 B 表，並傳參數給 B 表，則先在 B 表的前單互動設定接收參數，再到 A 表設定傳遞參數，接收參數設定方式：

1. 進入編輯模式。
2. 新增參數項目。
3. 設定參數名稱。
4. 挑選參數型態。
5. 若要設定多個參數請重複步驟 2~4，設定完畢後儲存。

四、表單元件加註

在 DKS 系統裡加註的架構中，元件分為四大類：

- 標題元件：只用來標示表單畫面上的欄位名稱，和表單的來源邏輯表格無關。
- 一般元件：表單畫面中除了標題元件以外的所有元件，能以多種型態呈現，例如一般的文字欄位、下拉選單、開窗欄位…等等。
- 選取元件：以核取方塊的方式呈現。
- 隱藏元件：畫面上看不到、但是與邏輯表格欄位有對應關係的元件，通常用來處理不讓使用者看到、但需要存回到資料庫或從資料庫撈出的資訊。

採用 DKS 系統進行表單元件加註，操作步驟如下：

(一) 元件基本設定(見圖 4-32)

圖 4-32 元件基本設定

1. 進入編輯模式。
2. 欄位名稱，若先前繪製畫面時未設定妥當，此處可再次設定。
3. 挑選欄位的模版。
4. 設定模版的長度，也就是使用者實際可以輸入或是看到的資料長度。
5. 若未在表單加註中設定過元件對應，此處可挑選對應哪個資料檔區。
6. 挑選檔區中的對應邏輯表格欄位。
7. 若此欄位是不可輸入的，請勾選此選項。
8. 若勾選「僅供顯示」，則需決定是否可以駐留在此欄位上(例如選取複製欄位上的資料)。
9. 若顯示欄位允許駐留，需設定允許駐留的時機。
10. 若此欄位要超連結到某一個按鍵，請在此挑選按鍵名稱。

11. 操作提示若有設定，使用者將滑鼠游標移到元件上時，將顯示提示。

12. 設定完畢請存檔。

(二) 顯示設定(見圖 4-33)

圖 4-33 顯示設定

1. 進入編輯狀態。

2. 勾選「指定顯示內容」。

3. 若有需要，設定執行這個顯示設定的條件，條件符合才會照設定的內容顯示。

4. 固定文字：此元件會顯示固定文字。

5. 欄位組合：依據自定的運算式顯示內容。標題題示：可讓元件在角落上有個標記，例如要標示為必輸欄位可以用此選項。

6. 僅改變顏色：單純改變欄位文字的顯示顏色。

7. 依資料顯示多語內容。

8. 顯示列數：顯示多筆表格中的列數。

9. 指定標題提示、僅改變顏色的前景字體顏色。

10. 指定僅改變顏色的背景顏色。

11. 若為顯示元件，可設定固定一段時間重新抓取一次資料顯示，請設定時間間隔的分鐘數和秒數。

12. 存檔。

(三) 預設給值(見圖 4-34)

圖 4-34 預設給值

1. 進入編輯模式後，勾選預設欄位內容值。

2. 若有需要，設定預設給值的條件。

3. 設定給值時機。

4. 設定給值的方式有系統資訊、登入資訊、固定給值、查表給值、流水編碼、單據編碼等。

5. 存檔。

(四) 選項清單-固定下拉選項清單(見圖 4-35)

圖 4-35 選項清單-固定下拉選項清單

1. 進入編輯模式後，勾選選項清單。
2. 若有需要，設定顯示選單的條件。
3. 選擇下拉選項。
4. 選擇固定選項。
5. 新增選項。
6. 設定選項的內容。
7. 設定完畢請存檔。
8. 固定下拉的選項是固定的，若選項名稱有設定，「1.」、「A.」之類的選項代號(與名稱用「.」分開)，則實體表格會存回「1」、「A」，若選項名稱未設定選項代號，則直接存回選項名稱。

(五) 選項清單-變動下拉選項清單(見圖 4-36)

圖 4-36 選項清單-變動下拉選項清單

1. 進入編輯模式後，勾選選項清單。
2. 若有需要，設定顯示選單的條件。
3. 選擇下拉選項。
4. 選擇變動選項，並決定每次下拉時是否重新從資料庫下載選項資料。
5. 設定此元件只能從選項中挑選，或使用者可自行輸入(輸入的內容需符合選項內的任一筆資料)，若可自行輸入，請選擇是否需要錯誤訊息。
6. 設定選項內容的資料來源邏輯表格。
7. 若有需要，請設定資料來源的過濾式。
8. 新增選項資料。
9. 設定選項欄位名稱(來源邏輯表格的欄位)。
10. 設定哪些欄位要帶回，哪些要顯示在下拉的選項中，有帶回的不一定要顯示，有顯示的一定要帶回，設定在項次 1 的欄位一定要顯示+帶回。
11. 設定帶回的對應元件。項次 1 的元件通常就是正在加註的這個元件。
12. 挑選在下拉選單中排序的欄位和排序方式。
13. 存檔。

(六) 開窗參照(見圖 4-37)

圖 4-37 開窗參照

1. 進入編輯模式後，勾選開窗參照。
2. 若有需要，可設定何種條件下才有此開窗功能。
3. 選擇開啟表單。
4. 挑選要開啟的表單，若表單有設定接收參數，請傳參數。
5. 開啟表單顯示的資料若需過濾，請填寫過濾式。
6. 選擇單選或複選。
7. 設定此欄位是否限定勾選，不讓使用者自行輸入。
8. 新增帶回欄位。
9. 設定要勾回到本單的哪個欄位，以及從開窗的哪個欄位帶回。
10. 勾選筆數為複選時，設定各個帶回欄位是否要判斷重複，若有勾選，則有勾選的欄位組合在帶回時會略過已存在多筆表格元件內的資料。
11. 若帶回的欄位有在元件加註中設定 7.更新給值的資料，則可選擇是否要觸發這個更新給值，勾選代表觸發，不選代表不觸發。
12. 設定完畢請存檔。

(七) 檢控限制(見圖 4-38)

圖 4-38 檢控限制

1. 不可空白：限制文字欄位不可空白，數字欄位不可為 0。可空白則相反，但可空白需進一步選擇有值檢錯(欄位不為空白時才檢錯)或是強制檢錯(空白也會啟動檢錯判斷)。
2. 不可重複：表頭欄位會檢查輸入值不可與已存在資料庫中的資料重複，表身欄位僅比對多筆表格元件內的資料不重複。
3. 不可負數：數字欄位不可輸入負數。
4. 文字加密：會將此欄位的資料加密顯示。
5. 若要自訂檢控內容，請勾選此處。
6. 若有需要，可設定何種條件下才執行此檢控。
7. 一般使用查表比對，以條件式設定介面設定檢控的條件式。
8. 可以限制欄位中不可輸入什麼內容。
9. 設定訊息彈出的時機。
10. 設定訊息類別。錯誤：彈訊息並不准輸入。警告：只彈訊息，保留輸入內容。詢問：彈出訊息並詢問使用者是否確定要輸入當時的輸入值。
11. 設定訊息內容。

12. 若訊息內容要隨資料內容變化，需設定替代字元，請見下節。
13. 若在新增存回或(及)修改存回時要重新檢錯一次，可設定要重複檢控的時機點。
14. 設定完畢請存檔。

(八) 更新給值(見圖 4-39)

圖 4-39 更新給值

1. 勾選查表帶值內容。
2. 設定要查詢的邏輯表格(或實體表格)及查表的過濾式。
3. 若查詢過濾出的資料有可能有多筆，可設定來源排序，系統將帶回排序後的第一筆資料。
4. 若勾選此項，查不到資料時會將下方設定之目的欄位清空。
5. 設定要寫入之目的欄位。
6. 挑選查表之來源欄位。
7. 此二按鍵可新增/刪除目的欄位。
8. 設定完畢請存檔。

(九) 編輯能力(見圖 4-40)

圖 4-40 編輯能力

1. 勾選此處以開啟編輯能力控制介面。
2. 如有需要，可設定改變編輯能力的判斷條件，若不設則一律改變編輯能力。
3. 可清除判斷條件。
4. 在控制類別選項選擇「欄位」。
5. 唯讀：欄位只能駐留，不可修改值。
6. 勿駐：欄位不可駐留。
7. 資料隱藏：欄位中的值變為「*」號。
8. 元件隱藏：整個欄位隱藏，畫面上看不到。
9. 設定完畢請存檔。

(十) 樹狀控制(見圖 4-41)

圖 4-41 樹狀控制

1. 此選項表示此樹狀清單為單一檔區的樹狀結構。
2. 用來設定單一檔區的父階欄位、子階欄位、順序欄位、階層欄位，其中父階和子階為必填。
3. 此選項表示此樹狀清單為兩個檔區的樹狀結構。
4. 設定樹狀的上、下層分別屬於哪個檔區。
5. 設定上層檔區的父階欄位。
6. 設定下層檔區的父階、子階欄位。
7. 可設定表單開啟時是否要預設展開。
8. 預設展開的方式，可選擇全展，或是指定展多少階。
9. 若此樹結構可能有多個根節點，請勾選此選項。
10. 可指定樹狀清單各個節點的顯示方式。
11. 設定根節點、父階欄位、子階欄位分別要顯示什麼資料。
12. 可指定節點的圖示。
13. 可挑選圖示。
14. 設定完畢請存檔。

(十一)　被動更新(見圖 4-42)

圖 4-42 被動更新

1. 被動更新：和更新給值相反，此功能用來設定本欄位的值是由哪些欄位更新過來。
2. 如有需要，可設定何種條件下才執行被動更新功能。
3. 單筆：表示來源欄位是單筆的資料。
4. 設定單筆給值的內容。
5. 多筆：表示來源欄位是在多筆表格元件中，累加到本欄位。
6. 選擇多筆表格元件所在的檔區。
7. 若有需要，可設定來源欄位參與累計的條件，不符合條件的資料就跳過不累計。
8. 設定多筆給值的內容。
9. 設定完畢請存檔。

五、按鍵加註

按鍵又稱為功能鍵，在 DKS 系統架構中分為兩大類：

1. 系統按鍵：位於表單選單列，提供基本的系統功能，例如新增、刪除、修改、存回等。
2. 自訂按鍵：位於表單工具列，按鍵的內容完全是由設計者自訂，除了由使用者按下執行，也可設定成系統排程自動執行。而自訂按鍵又包含兩個子類：
   (1) 隱藏按鍵：畫面上看不到的按鍵，通常由其他按鍵或是元件觸發，也可設定成系統排程按鍵，時間到了自動執行。
   (2) 畫面按鍵：放在表單畫面中的按鍵，本身為一個連結器，只能設定成執行其他按鍵。

採用 DKS 系統進行表單元件加註，操作步驟如下：

(一) 按鍵基本加註(見圖 4-43)

圖 4-43 按鍵基本加註

1. 可修改功能鍵的名稱。
2. 設定功能鍵的熱鍵。
3. 如有需要，可設定操作提示。(使用者將滑鼠游標移到按鍵上方時會顯示)
4. 可開啟 SA 規格描述畫面。
5. 設定按鍵何時可致能。不致能時，按鍵會變成灰色，使用者無法執行。
6. 若此按鍵是針對表單畫面上多筆元件內的已勾選資料做執行，則需挑選勾選的檔區。
7. 若勾選的資料需存到暫存實體表格中，讓過帳用的邏輯表格使用，需勾選此項。
8. 此選項可在按鍵執行完畢後，自動清除多筆元件內的勾選標記。
9. 執行後續的設定，可設定系統排程、顯示訊息、更新表單畫面的資料、更新資料的檔區及方式。
10. 設定完畢請存檔。

(二) 執行限制(見圖 4-44)

圖 4-44 執行限制

執行限制的設定原則：檢控一件事情，需獨立一個執行限制。

1. 勾選此項目可開啟設定介面。
2. 設定此執行限制的判斷順序，如果一個按檢有兩個以上的執行限制，順序小的先判斷。
3. 若有需要，可設定判斷時機，符合判斷時機設定的條件時，才會執行這項檢控。
4. 條件說明：設定這項檢控真正要檢查的條件。
5. 在條件說明中無法完整描述的檢控，可使用特殊檢控。
6. 資料檢查：依表單內的資料狀況做檢控。
7. 限定模式：限定要在何種狀態下才允許執行。
8. 特定範圍：檢控某個檔區內的資料，需在資料區挑選要檢控的檔區。

9. 須有記錄存在：若勾選，代表此檔區內需有資料才可執行。若不勾選，則需與上方的 4.條件說明配合使用，代表此檔區內每筆資料都要符合條件。
10. 數值合計判斷，若勾選，需填寫下方的判斷式，「欄位」代表多筆元件中要合計的欄位，然後設定運算子以及要比對的數值或表單元件。
11. 判斷多筆元件內的資料是否有重複，區分群組用來設定群組欄位；檢查欄位可設定多個欄位，需與 8.特定範圍搭配使用。
12. 最大筆判斷：若駐留的資料是多筆元件內最大的資料，則依據 4.條件說明的內容判斷是否符合條件。參與條件：資料符合此條件才判斷是否為最大筆。最大筆判斷欄位：設定以哪個欄位來決定是否為多筆元件內最大資料。
13. 若檢控的結果為不允許執行功能鍵內容時，可指定駐留到哪個元件。
14. 勾選檢查：可檢查多筆元件內的勾選狀態。
15. 已勾選筆數：檢查多筆元件內的已勾選筆數是否符合條件。
16. 設定要檢查的多筆元件檔區。
17. 挑選運算子，例如大於、等於、小於....等。
18. 請輸入數值，例如 1、2、3...。
19. 駐留筆：檢查駐留的資料是否勾選或未選。
20. 設定執行時機，可設定上方設定的各項條件成立時要做什麼動作，或是條件不成立時做什麼動作。只能選擇一方做設定。

    執行：執行按鍵加註中設定的功能。

    訊息：不符合檢控條檢時彈出錯誤訊息。

    除能：按鍵變灰色。

    致能：按鍵可按。
21. 訊息類別：

    錯誤：彈訊息並不准執行。

警告：只彈訊息，訊息按掉後繼續執行。

詢問：彈出訊息並詢問使用者是否確定繼續執行。

22. 按鍵駐留：當訊息類別為詢問時，預設駐留的按鈕。
23. 訊息內容：以多語介面設定訊息內容。
24. 動態訊息：若訊息內容需隨資料變化時，請勾選此選項，並決定訊息的出現方式。
25. 若有動態訊息，需設定替換字。
26. 設定完畢請存檔。

(三) 開啟它單(見圖 4-45)

圖 4-45 開啟它單

1. 表單名稱：設定要開啟的表單名稱。
2. 資料過濾：如有需要，可在開啟表單時過濾被開啟表單資料。
3. 駐留條件：若在開啟表單時要駐留到指定的記錄，請輸入過濾此記錄的條件。
4. 樹根指定：指定樹狀表單 Root(根目錄)的過濾條件。
5. 表單異動：打勾後可限定被開啟表單的異動功能。
6. 承上，設定要給被開啟的表單哪些異動功能。

7. 進入模式：開啟表單時，預設的模式是哪種。
8. 互動模式：假設 A 單開 B 單，用來設定被 A 單和 B 單的互動模式：
9. 主動：A 單和 B 單可自由切換，兩單互不干涉，但必須關閉 B 單後才能關閉 A 單。
10. 被動:A 單開啟 B 單後,只能駐留在 B 單,需關閉 B 單後才能駐留 A 單。
11. 互動：A 單和 B 單可自由切換，且兩單可設定互傳訊息，例如資料記錄的移動。
12. 可用載入功能檢查被開啟的表單是否有設定接收參數，若有，請設定參數的傳遞內容。
13. 設定完畢請存檔。

(四) 開啟報表(見圖 4-46)

圖 4-46 開啟報表

1. 啟動開啟報表的設定介面。
2. 設定執行順序，此執行順序涵蓋按鍵加註的 A~K 頁籤，為文字型態，因此 10 會比 3 先執行，需注意。
3. 若有需要，可新增開啟這張報表的執行條件。

4. 挑選要開啟的報表。
5. 預覽：在運行環境會先用預覽視窗開啟報表，使用者再決定後續的列印動作。
6. 郵件夾檔：可設定為將報表嵌入為電子郵件夾檔，或是直接插入在郵件內容中。
7. 直接下載：直接下載為 PDF 檔案，使用者可儲存到自己電腦中。
8. 存入資料庫：將報表存入資料庫的資料表中。
9. 選擇要存入的實體表格(資料表)。
10. 選擇要存入的實體欄位(資料行)。
11. 選擇存放檔名的實體欄位(資料行)
12. 此過濾條件用來設定要存到實體表格中的哪一筆資料的欄位中。
13. 可用載入鍵檢查報表是否有接收參數，若有，請設定傳入參數。
14. 設定完畢請存檔。

(五) 數值影響(見圖 4-47)

圖 4-47 數值影響

1. 啟動數值影響的設定介面。
2. 設定執行順序，此執行順序涵蓋按鍵加註的 A~K 頁籤，為文字型態，因此 10 會比 3 先執行。
3. 若有需要，可新增執行這個過帳的執行條件。
4. 挑選過帳類型。

   一般：若過帳內容不含迴圈設定，請挑選此項。

   展階層：若過帳內容含有迴圈設定，請挑選此項。
5. 挑選要執行的過帳名稱。
6. 可用載入鍵檢查過帳是否有接收參數，若有，請設定傳入參數。
7. 設定完畢請存檔。

(六) 資料交換

資料交換意指和系統外部的檔案做資料的交換，分為匯出和匯入兩種形式，系統作動的流程如下圖 4-48 資料交換流程：

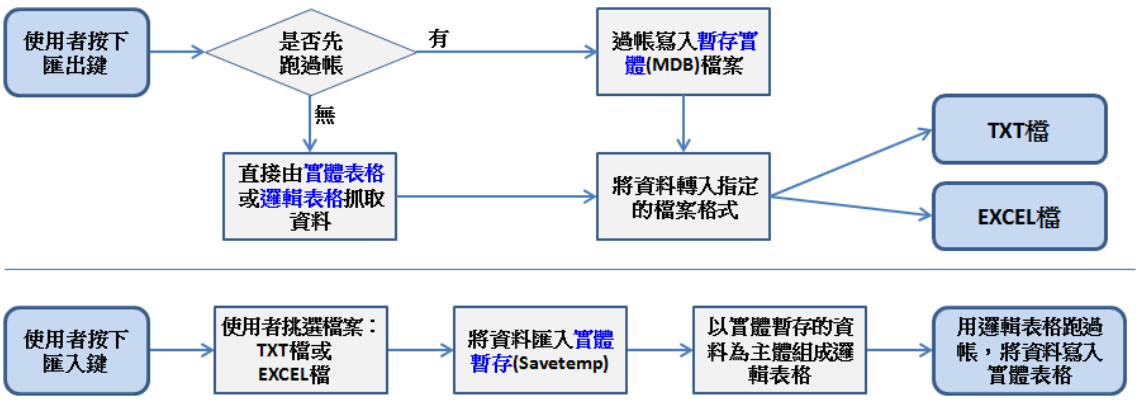

圖 4-48 資料交換流程

(七) 資料載入

資料載入功能需在表單為編輯狀態時執行，載入的資料來源僅限於邏輯表格(檢視表)，載入表身檔區之多筆表格時，可以載入多筆資料，若載入表頭檔區時，只能載入一筆資料，系統作動的流程如下圖 4-49 資料載入流程：。

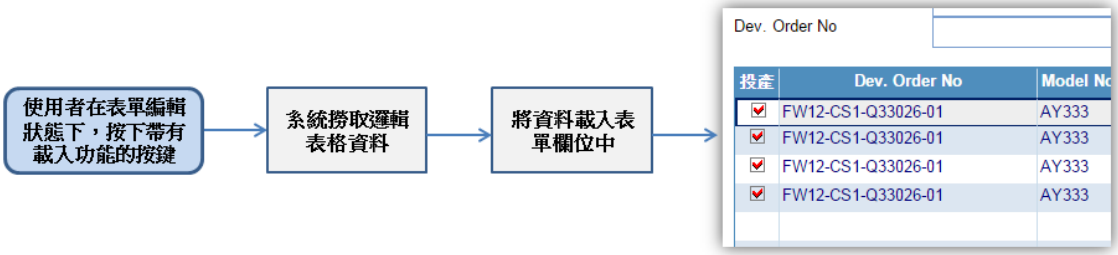

圖 4-49 資料載入流程

第五節　打樣測試

雛型測試的目的是以快速產出雛型(Prototype)實際操作執行來檢查軟體，提早發現軟體錯誤並進而修改軟體，以提昇軟體品質，避免錯誤造成嚴重損失。

一、打樣預覽

DKS 系統提供打樣預覽的功能，由系統打樣產出樣板(Prototype)，方便做初步的除錯或是與使用者確認規格，經由實際體驗來驗證是否符合使用者需求，執行步驟見圖 4-50。

圖 4-50 打樣預覽

1. 可進行打樣預覽。

2. 可查詢打樣狀態。

二、專案版本發行

版本控管是針對開發不同版本進行標識和跟蹤，可保證軟體技術狀態的一致性，目的是方便企業區分、檢索及追蹤系統的不同版本，以識別各版本間的關係。

DKS 系統提供版本管理，執行步驟見圖 4-51。

圖 4-51 專案版本發行

1. 在系統左下方的共用頁籤中，啟動版本發行功能。

2. 輸入所要查詢的區間，預設為最近一個月。

3. 挑選要查詢的狀態。

4. 查詢發行記錄。

5. 可查詢發行的詳細狀態。

6. 可新增發行版本。

第六節　專案產出

本專案依系統開發時程表所規劃時程，採改良式雛型模型進行專案開發，經由雙方專案成員密集會議溝通，以快速產出之雛型進行規格討論與確認，確認無誤後再進行系統設計，最終產出 33 張表單及 37 張報表，計 70 張表報，使用 SA 工時 369 小時，SD 工時 183.5 小時，總開發人力工時為 552.5 小時，產出明細見表 4-1 專案開發產出明細表。

表 4-1 專案開發產出明細表

序	類型	表報名稱	SA 工時	SD 工時	小計
1	表單	員工資料	4	2	6
2	表單	廠商資料	4	2	6
3	表單	基本代號	4	1	5
4	表單	業主資料	4	2	6
5	表單	銀行資料	2	1	3
6	表單	銀行一覽表	1	0.5	1.5
7	表單	公司資料	1	1	2
8	報表	員工明細表	2	1	3
9	報表	廠商明細表	2	1	3
10	報表	業主明細表	2	1	3
11	報表	銀行明細表	2	1	3
12	表單	工程合約	16	8	24
13	表單	發包登錄	24	8	32
14	表單	工程計價建立	30	10	40
15	表單	工程計價查詢	8	3	11
16	表單	廠商扣款單	4	2	6
17	表單	未計價發票一覽表	1	0.5	1.5
18	表單	未計價扣款單一覽表	1	0.5	1.5
19	報表	在建工程明細表	4	2	6
20	報表	訂購單	4	2	6
21	報表	計價單	8	4	12
22	報表	廠商扣款明細表	4	2	6
23	表單	進項發票	12	8	20
24	表單	付款沖銷	12	8	20

25	表單	支票本查詢	2	1	3
26	表單	支票本	3	1	4
27	表單	支票本一覽表	1	0.5	1.5
28	表單	應付票據一覽表	1	0.5	1.5
29	表單	應付票據兌現單	4	2	6
30	表單	未付款憑證一覽表	1	0.5	1.5
31	報表	應付帳款憑證	4	2	6
32	報表	支票簽回單	4	2	6
33	報表	折讓單	4	2	6
34	報表	進項發票明細表	4	2	6
35	表單	銷項發票	16	10	26
36	表單	收款沖銷	12	8	20
37	表單	未收款憑證一覽表	1	0.5	1.5
38	表單	應收票據兌現單	4	2	6
39	表單	應收票據一覽表	1	0.5	1.5
40	表單	電匯收款兌現單	4	2	6
41	表單	電匯收款一覽表	1	0.5	1.5
42	表單	本票資料	2	1	3
43	表單	發票沖銷作業	4	2	6
44	報表	本票明細	2	1	3
45	報表	未收回本票明細表	4	2	6
46	報表	沖銷發票明細表	4	2	6
47	報表	可沖銷發票明細表	4	2	6
48	報表	未收款明細	2	1	3
49	表單	採購查詢	4	2	6
50	表單	帳款兌現查詢	6	4	10
51	報表	工地計價明細表	4	2	6
52	報表	廠商扣款明細表	4	2	6
53	報表	應付保留款表	4	2	6
54	報表	進項發票明細表	2	1	3
55	報表	工程結案執行結果表	8	3	11
56	報表	銀行電匯檔	6	2	8
57	報表	工程收支&損益明細表	8	5	13
58	報表	個案收支明細總表	8	4	12
59	報表	廠商承攬金額明細表	4	2	6
60	報表	工地計價總表	4	2	6
61	報表	應付帳款憑證	6	2	8

62	報表	工程採發成果表	8	3	11
63	報表	年度採購工程比例表	8	3	11
64	報表	應收帳款明細表	6	4	10
65	報表	應收保留保固表	4	2	6
66	報表	應收保固款明細表	4	2	6
67	報表	合約餘額明細表	8	5	13
68	報表	銷項發票明細表	2	1	3
69	報表	工程結案收入明細表	8	5	13
70	報表	零星工程報表	8	6	14
合計			369	183.5	552.5

資料來源：本研究整理

第七節　研究分析與研究結果

一、軟體效益分析

軟體系統開發在各個階段的投入人力比重依專案的複雜度、規模大小與採用開發工具與模型的不同，會有不同的結果，一般而言，系統分析(SA)：系統設計(SD)：程式撰寫(Coding)：系統測試(Testing)的投入人力時間比為 1:2:3:3。本研究實證採用 DKS 系統及改良式雛型模型進行專案開發，各階段投入人力時間比為系統分析：系統設計：程式撰寫：系統測試=2:1:0:0，因此類推若採用傳統開發方式開發本專案，所花費的人力工時預估應為 3,321 小時，是本研究的 6 倍。

二、研究結果

經實證採用 DKS 工具開發工程計價系統，分析結果發現：

1. 本專案開發進度依開發時程規劃如期在 3 個月內完成。
2. 本專案投入人力成本與專案組織規劃人力成本相符。
3. 本專案最後完成之系統架構與原系統架構規劃相符。
4. 本專案完成之作業流程與需求訪談規劃之系統作業流程相符。
5. 本專案產出表單及報表功能與系統分析產出之程式流程圖相符。

6. 本專案產出表單及報表規格與使用者需求相符。
7. 本專案已順利上線，無返工問題，品質通過使用者檢驗。
8. 本專案不須撰寫程式，所有規格皆記錄在 DKS 系統上，使用者有新的需求可立即查看規格，維護度容易。
9. 本專案提供 DKS 系統產出之規格書，以保證產出成品與規格書的一致性。
10. 本專案提供版本管理，可保留系統開發軌跡，避免與使用者產生爭執。
11. 本專案投入系統設計人力不需有程式設計背景，人力取得與養成容易。
12. 本專案提供模擬操作,使用者在軟體開發初期能準確完整的表達他們的需求，同時降低系統開發人員對需求理解上的誤差，改良了雛型模型的缺點。
13. 本專案提供打樣功能，縮短規格系統測試與後續教育訓練時間，同時縮短軟體開發人員對使用者需求的理解與使用者本來的期望的落差。
14. 本專案提供完整且經確認的規格，系統設計人員對規格有清晰全面的瞭解。
15. 本專案採用改良之雛型模型與 DKS 系統可快速將複雜的邏輯思維過程，有系統有步驟地抽離出來產生軟體產品。
16. 本專案採用 DKS 系統受限於系統工具功能限制，無法天馬行空地發揮設計師的創意。
17. 本專案採用 DKS 系統受限於系統工具元件限制，會犧牲某部分的效能。

第五章　結論與建議

本章總結上述內容，並得知以 DKS 系統為軟體開發工具進行工程計價系統開發，藉此開發過程與經驗與傳統軟體開發方式進行比較，以證明 DKS 系統是否可提高軟體開發效率，增加軟體可維護性並降低開發成本的目的。

第一節　結論

據前述研究結果得知以下結論，請參照表 5-1 研究驗證結果：

表 5-1 研究驗證結果表

序	比較項目	DKS 開發	傳統開發
1	可修改性	☺	
2	有效性	☺	
3	可靠性	☺	
4	可理解性	☺	
5	可修改性	☺	
6	可維護性	☺	
7	可重用性	☺	
8	可適應性	☺	
9	可追蹤性	☺	
10	彈性		☺
11	效能		☺

資料來源：本研究整理

本研究實證結果顯示出：

1. 採用 DKS 系統工具開發在可修改性方面優於傳統軟體開發方式。
2. 採用 DKS 系統工具開發在有效性方面優於傳統軟體開發方式。
3. 採用 DKS 系統工具開發在可靠性方面優於傳統軟體開發方式。
4. 採用 DKS 系統工具開發在可理解性方面優於傳統軟體開發方式。

5. 採用 DKS 系統工具開發在可修改性方面優於傳統軟體開發方式。

6. 採用 DKS 系統工具開發在可維護性方面優於傳統軟體開發方式。

7. 採用 DKS 系統工具開發在可重用性方面優於傳統軟體開發方式。

8. 採用 DKS 系統工具開發在可適應性方面優於傳統軟體開發方式。

9. 採用 DKS 系統工具開發在可追蹤性方面優於傳統軟體開發方式。

10. 採用 DKS 系統工具開發在彈性方面劣於傳統軟體開發方式。

11. 採用 DKS 系統工具開發在效能方面劣於傳統軟體開發方式。

一、實務涵義

比較上述目標，實證採用 DKS 系統工具有助於提高軟體產品的品質和開發效率並減少維護的困難。

二、管理涵義

傳統軟體開發是將領域知識(Domain Knowledge)透過程式碼來呈現，其結果是除非原始程式開發人員，否則其他人員很難理解該程式碼所代表的管理目的，因此在軟體購買之前，軟體需求者並無法充分了解其購買的軟體是否符合其需求。DKS 系統將領域知識從程式碼中抽離出來，以可理解的文字描述來定義領域知識，軟體需求者透過可閱讀的文字可充分了解其購買的軟體功能，透過 DKS 系統未來甚至可以將知識封裝成獨立的商品進行知識市集販賣，以實現知識經濟的目的。

第二節　研究限制與建議

本研究主要探討 DKS 系統工具的適用性，因為 DKS 系統工具是一套非常新的科技資訊軟體技術，市面上尚未發現類似的產品，本研究雖已提供了實證上的解釋與結論，但仍有其研究上之限制，而這些限制可以提供未來研究的一些方向。

一、研究結論

本研究結論僅針對工程計價系統進行研究，若要將本研究結果應用到其他產業、其他需求，當注意其適用性。

二、研究對象

本研究對象僅針對單一產業、單一需求進行研究，建議對 DKS 系統工具有興趣的後續研究者可針對不同產業、不同軟體規模進行多方面的研究。

參考文獻

一、中文文獻

1. 李允中（2009），《*軟體工程*》，臺北市：高立圖書有限公司出版
   http://web.ydu.edu.tw/~hjw/course/donet/Chapter01.pdf (檢索時間 2016/05/14)
2. 鄭炳強（2007），《*軟體工程：從實務出發*》，臺北市：智勝文化出版

二、英文文獻

1. Bauer, (1968). "Software Engineering."
2. Boehm, (1988). "A Spiral Model of Software Development and Enhancement.", *IEEE Computer*, Vol.21, No. 5, pp. 61-72.
3. Boehm, (1991). "Software Risk Management: Principles and Practices.", *IEEE Software*, Vol. 8, No. 1, pp. 32-41
4. Brian Randell, (1996). "The 1968/69 NATO Software Engineering Reports.", http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOReports/(檢索時間 2016/06/14)
5. Conger, (1994). "The new software engineering.", *Belmont, Calif.: Wadsworth Pub*.
6. Fairley, (1985). "Software engineering concepts.", *New York: McGraw-Hill*.
7. Frederick P. Brooks, (1975). "The Mythical Man-Month: Essays on Software Engineering."
8. Genuchten, (1992). "Towards a software factory.", *Dordrecht: Kluwer Academic Publishers*.
9. Ghezzi, Jazayeri & Mandrioli, (1991). "Fundamentals of software engineering.", *Englewood Cliffs, NJ: Prentice Hall.*
10. Mills, (1971). "Top-Down Programming in Large Systems, Debugging Techniques in Large Systems.", *Englewood Cliffs, NJ: Prentice Hall*, pp. 41-55.
11. ISO/IEC 14764 (2006). "Software Engineering -- Software Life Cycle Processes -- Maintenance.",

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39064 (檢索時間 2016/06/14)

12. Bally, Brittan, Wagner, (1977). "A Prototype Approach to Information System Design and Development", *Information and Management*, Vol. 1, No. 1, pp. 21-26.
13. Pfleeger, (2001). "Software engineering". *Upper Saddle River, N.J.: Prentice Hall.*
14. Pigoski, (1997). "Practical software maintenance". *New York: Wiley Computer Pub.*
15. Schach, (2011). "Object-oriented and classical software engineering". *New York: McGraw-Hill.*
16. Simson, Garfinkel, (2005). "History's Worst Software Bugs", http://archive.wired.com/software/coolapps/news/2005/11/69355?currentPage=2(檢索時間 2016/06/14)
17. Tracy, (2008). "The Economic Ripple Effect Gone Awry", http://www.cbsnews.com/news/the-economic-ripple-effect-gone-awry/(檢索時間 2016/06/14)
18. Royce, (1970). "Managing the Development of Large Software Systems", *Proceedings of IEEE Wescon*, pp.1-9.