私立東海大學資訊工程學系研究所

碩士論文

指導教授：朱正忠　教授

軟體製品管理系統實作

Implementation of Software Artifact Management System

研究生：廖麒程

中　華　民　國　一　百　年　七　月

摘要

軟體在開發完成並交付給使用者驗收後，隨即進入後續的維護階段，為了延續軟體的生命週期，各種需求變更的維護作業相繼產生，而維護成本也隨之攀升，因此如何有效的提升軟體維護品質同時降低維護成本，已成為值得深入研究的議題。尤其是在大規模的作業環境下，如 IC 製造廠動輒上千台機器的連線程式或是服務業散佈於各服務據點的終端機程式，其軟體佈署更需耗費大量的人力資源才能完成。在進行分工後，再由統一且簡單的程序管理，來減輕工作的複雜度進而提升其維護品質，就成了當務之急。本論文實作一套軟體系統，協助軟體團隊透過本系統做分工與權限的控管，並配合自動化佈署功能，獲取品質與效能兼具的軟體維護方案。本系統使用 MySQL 資料庫，除了儲存相關的管理資訊之外，在釋出檔案前先備份於資料庫中，再做釋出的動作，如此可避免備份檔案遭到更改，而造成與釋出檔案內容不一致的情況，另外也使用 XML 元件作為實體檔案與資料庫之間的轉換媒介。

關鍵詞：軟體維護、軟體佈署、可延伸標記語言

Abstract

While software has been accepted by users, it enters the regular maintenance. In order to prolong the lifespan of software, which result in having multitude of operation maintenance and increasing cost as well to meet different kinds of change. Therefore, how to improve efficiently for the quality of maintenance, and to keep the relatively lower cost in maintenance simultaneously that is a key needed to study. Especially in large operation environments such as link programs in an IC fabrication for thousands of equipments or POS programs among each selling points in service industry, which count on more human resources to accomplish it. After the division of job, that is the most urgent case to manage it by regular and simple process, and to eliminate complexity and improve quality of maintenance. The author developed software system to help software team make the job division and control the privilege, and to get the quality and efficiency of software along with auto-function of deployment. These systems backup all data before releasing except for storage management by MySQL database that can avoid inconsistency between backup and the releasing data. In addition, XLM device is also adopted to be the intermediary between operation files and database.

Keywords: software maintenance, software deployment, XML

致謝

本篇論文之所以能順利完成，首先要感謝我的指導老師 朱正忠教授，在學業上的悉心指導，除了使學生在軟體工程領域有更深的認識，且在研究方法上積極與嚴謹的態度更令學生獲益匪淺，尤其是在系統開發期間遇到問題時的提攜與鼓勵更是讓此論文得以如期完成的最大關鍵。另外感謝筆者所服務的公司提供之工作環境與研究對象，希望此篇研究成果對公司有所助益。

本論文承蒙 盧志偉教授於百忙之中擔任召集人並對論文整體內容提出精闢的建議與鼓勵；並承 楊朝棟教授之於論文結構與內容的建議;承 蔡清叢教授於系統實用性上的建議與 張志宏教授於資料庫設計上的建議均讓此系統更具實務性。感謝 口試委員們對此論文所提供的改善建議與指導，讓本論文的內容更臻完善，僅此向老師們致上學生最高的敬意與謝意。

另感謝資訊技術實驗室的柏宇與系辦的林若瑩小姐等人在許多繁瑣行政事務上的協助，讓我免於往返學校與工作場所。最後感謝我最親愛的妻子，由於妳耐心的照顧兩子，讓我在學業與工作上無後顧之憂，僅將此論文獻給所有關心我的師長、家人、朋友與後進們。

廖麒程 謹上 2011/7

目錄

圖目錄

表目錄

第一章　導論

1.1 前言

綜觀整體軟體生命週期中，軟體系統的維護成本隨著使用期限的不斷展延，更早就超越了當初的開發成本，而軟體維護成本從 1960 年代只佔總成本的 20%不到，到已躍昇到 60%以上[11]，但為何有這樣的差異呢? 或許是因為客製化軟體本身就被賦予修改的彈性與必要性，再加上不同時期的軟體維護人員對同一件事有不同的見解與取捨，故使得軟體維護更具潛在風險，這也使得要做好軟體維護的品質就更加困難重重。雖然本質上的問題要從設計分析階段開始著手規劃，然而由於重新開發的成本也是一筆龐大的支出，故嘗試從維護階段中研究，保留原有的程式架構，找出增加維護品質與效能的方案，以減少軟體改版的風險及其影響程度。

一般軟體維護的目的可分為以下四類:[1][5][6][10][15][16]

1. 修正性維護(Corrective Maintenance): 修改軟體隱含的錯誤。

當軟體的功能性發生錯誤時須經修改軟體以恢復其原有的功能，通常是程式本身沒有寫好或考慮不周所引起的，例如 Y2K 千禧蟲應該可視為考慮不周，沒有預期到軟體的生命長度。

2. 適應性維護(Adaptive Maintenance): 修改軟體以配合新的環境。

因應軟體外部環境產生了變化，軟體為了配合該的環境的改變而

需要隨之修改。例如公司新購入了一個監控系統，故為了該新監控系統的正常運作而須修改已運作很久的資料收集的程式以提供足夠的資料給新加入的監控系統。

3. 完善性維護(Perfective Maintenance): 擴充、更新軟體功能。

一般是為了提升軟體效能或可維護性的修改，例如更改軟體邏輯讓軟體執行更有效率或讓軟體可讀性更佳，該軟體不改也不會影響現行運作的功能，故通常會待與其他維護工作一起做修改。

4. 預防性維護(Preventive Maintenance): 修改軟體以防止故障發生。

有時維護者需要增加一些自我監控的紀錄以避免未來發生重大的問題，例如新增自我檢查記憶體使用情形，若有不足時可即時調解使用量或通知維護人員及早因應。

1.2 研究動機

以筆者所從事的 IC 製造廠之機台自動化連線[18][19][20]軟體維護工作環境中，實際上有 90 份連線軟體(每一個連線軟體須有兩個以上的設定檔，另外所執行的電腦主機須有 20 個共用函式庫)。而每一個連線軟體的成本約三萬美金，故投資約兩百七十萬美金(約八千萬台幣)，然而以目前軟體維護的研究[2]大多是從軟體設計之初的系統架構階段著手做徹底解決維護問題之研究，若要從設計階段來解決本質問題，則須先評估其效益與之前投資的得失，這對已投資高成本之現存系統並不太適用。故本論文以不變動原始軟體架構為原則來提升軟體維護品質與效率。

補充說明：該設備連線軟體在 CIM(Computer-Integrated Manufacturing) 裡的角色是將半導體設備導入 CIM 的整合介面軟體，故開發前除了需先了解許多半導體設備相關之通訊規範[18][19][20]外，還要研究各設備的行為特性，以致於開發成本較高。不過雖然該廠約有 1000 台生產相關設備，但同一機型的設備只需維護一份連線軟體(該軟體可重複執行但須從設定檔中讀取不同的設定，用以連線到不同的實體設備)。

1.3 研究目的

以該廠在每月產出八萬片的 12 吋晶圓時，約需 1000 台設備 24 小時連續的運作，也就有 1000 個網路連線軟體分別執行於 75 台電腦主機上(如第三章之圖 2: 系統架構圖) ，而這 75 台主機坐落在兩個機房內，為了避免單一機房發生災難性意外(如漏水，跳電，網路故障等)而影響生產，故同一實體設備之連線軟體在佈署時須重複佈署到不同機房的主機以作為備援主機。

如此一個軟體改版釋出之最大佈署量約有 540~2040 個檔案(90*3*2 + 75*20)，故每當軟體根據需求修改完成之後，都要以如履薄冰的態度一一的將程式及其相關檔案佈署到正確的位置，所花費的時間約為 30 分鐘。另外以變更需求的量來看，平均每星期約有 5 個變更需求要維護，因為廠內只要有任一個系統需要變更對設備的控制或是在新產品導入時要收集新資訊，該設備連線程式就須配合修改以提供足夠的資料供其使用(此類改版屬於 Adaptive Maintenance)。也就是軟體維護人員每星期約有 150 分鐘花在檔案佈署上，為減少佈署時間同時又提高正確性，故製作一套軟體維護系統來做有效的分工管理與自動化佈署，讓軟體工程師有較多的時間專注於軟體修改，以獲得較好的維護品質與效率。

1.4 章節安排

論文的章節安排分述如下：

第二章　背景知識與相關研究，此章說明與本論文相關的研究，並介紹 XML 的觀念,以及其基本特性及其相關的應用。

第三章　系統設計，此章說明本論文之設計規格與理念。

第四章　案例研究，本章說明利用此系統實際操作範例。

第五章　結論及未來方向，此章將本論文之研究成果做總結，並提出未來研究之方向。

附錄 A 說明本系統之設定檔 SAMS.xml

附錄 B 介紹本系統之操作畫面及說明

附錄 C 相關程式碼範例

第二章　背景知識與相關研究

本章節將簡述軟體生命周期與軟體維護的背景與知識，與簡述XML[3][12][13][17]技術。

XML-based unified model(XUM)[2]的研究主要是涵蓋軟體開發階段到維護階段的實務，而本論文主要只針對維護階段的實務應用，原因是保留原始開發實務以減少變異。

2.1 XML (eXtensible Markup Language)

XML[4][13][17]是由 W3C 簡化了 SGML (Standard Generalized Markup Language)而制訂出來的,格式上類似 HTML 但彌補了 HTML 無法擴充與處理大量資料的缺點。因 XML 具有可擴充性、自我描述性與結構性的特點，故常用來描述文件的結構與內容。XML 包含了 DTD 與 XSL:

DTD(Document Type Definition)是用以定義 XML 文件的規格(Schema)，包括標籤(Tag)、從屬關係及限制條件，XML 文件的驗證則是以 DTD 為驗證條件(Criteria)而得以完成。

XSL(XML Style Language) 是基於 DSSL(Document Style Semantics and Specification Language)與 CSS(Cascading Style Sheet)是用來描述 XML 的顯示風格,將顯示風格與資料處理分開可以讓程式

更簡明更具可讀性。

2.2 XPath (XML Path Language)

XPath 是一種路徑表示法，可用來存取 XML 文件中的資節點資料內容，在 XML 中的元素、屬性與內容都稱作一個節點(Node) 。例如圖 1 以 XML 來描述實體目錄及檔案的結構，程式透過 XPath 查詢到所指定檔案的資料屬性。

```
<?xml version="1.0"?>
- <Dir DirCount="0" FileCount="8" Size="1549" Path="C:\Users\Chaney\THU\Project\DesignTimeSrc\">
    <File Size="784" CRC32="06DE4362" LastWriteTime="2011/2/22 23:53:8" Index="1">01 Project Charter.txt</File>
    <File Size="765" CRC32="99E8D129" LastWriteTime="2011/2/22 23:52:42" Index="2">01 Project Charter.txt.bak</File>
    <File Size="0" CRC32="00000000" LastWriteTime="2009/10/1 1:31:8" Index="3">02 Project Scope.txt</File>
    <File Size="0" CRC32="00000000" LastWriteTime="2009/10/1 1:31:30" Index="4">03 Project Plan.txt</File>
    <File Size="0" CRC32="00000000" LastWriteTime="2009/10/1 1:31:44" Index="5">04 Project Execution.txt</File>
    <File Size="0" CRC32="00000000" LastWriteTime="2009/10/1 1:31:58" Index="6">05 Project Monitor and Control.txt</File>
    <File Size="0" CRC32="00000000" LastWriteTime="2009/10/1 1:32:56" Index="7">06 Project Change Control.txt</File>
    <File Size="0" CRC32="00000000" LastWriteTime="2009/10/1 1:33:10" Index="8">07 Project Close.txt</File>
</Dir>
```

圖 1：以 XML 描述實體檔案的範例圖

XPath 將 XML 文件節點分為以下七種型態:

1. 根節點(Root Node)
2. 元素節點(Element Node): "/"
3. 文字節點(Text Node): "text()"
4. 屬性節點(Attribute Node): "@"
5. 命名空間節點(Namespace Node): "xmlns:"
6. 處理指令節點(Processing Instruction Node)
7. 註解節點(Comment Node): "<!-- -->

2.3 MySQL

MySQL[7]（讀作 my-S-Q-L or My-SEQuel）原開發者為瑞典的 MySQL AB 公司，然該公司於 2008 年被 Sun 收購。旋即於 2009 年，Oracle 收購 Sun 公司，故 MySQL 現為 Oracle 旗下產品。 MySQL 由於效能高、成本低、可靠性好，已經成為最流行的開源資料庫。

隨著自由軟體的拓展，它受到網路使用者的大力歡迎。MySQL 是多使用者、多執行緒的關連式資料庫伺服器。相較於其他大型關連式資料庫，MySQL 之所以受到歡迎，主要有幾個特色：

1. 經濟優勢，由於 MySQL 是開放源碼軟體，因此可以大大降低總體擁有成本約 80%。
2. 高度相容性，適用於跨作業系統平台、程式語言與伺服器，支援 FreeBSD、HP-UX、Linux、Mac OS、Solaris、Windows 等多種操作系統，並使用了多種編譯器進行測試，保證源代碼的可移植。
3. 在許多專業的自由軟體社群支援下，經過優化的 SQL 查詢演算法，有效地提高查詢速度，再搭配許多官方免費的 GUI 工具，亦提升管理資料庫的便利性。
4. 持續改善的發展，使其越來越適用於更多人性。

2.4 CRC(Cyclic Redundancy Check)

CRC (Cyclic Redundancy Check) 循環冗餘檢查[9]，是一種 checksum 它是以二進位的除法做為基礎，其中被除數是需要計算校驗和的信息；除數是一個長度為 n + 1 的預定的二進制數值，主要用來檢測資料儲存或傳送前後可能出現的錯誤，例如現行許多檔案下載伺服器端會將檔案的 checksum 值一併附上，讓下載端確認所下載的檔案與伺服器端一致，避免下載不完全的情形發生。

CRC 有許多變形，從 CRC-1 到 CRC-160 尾數越大代表所產生的 checksum 也就越可靠，然 CRC-128 及 CRC-160 已被 MD5 與 SHA-1 所取代(IEEE-ITU 標準)。

第三章　系統設計

此系統位於下圖 2 中間星形區域內，目的是作為軟體開發環境與執行環境之間的整合管理。圖面左側為開發環境而右側為實際執行環境。本論文是架設於兩者之間的中介整合應用程式，將開發環境的檔案在有條件限制的管理下，佈署到實際的測試或執行環境。

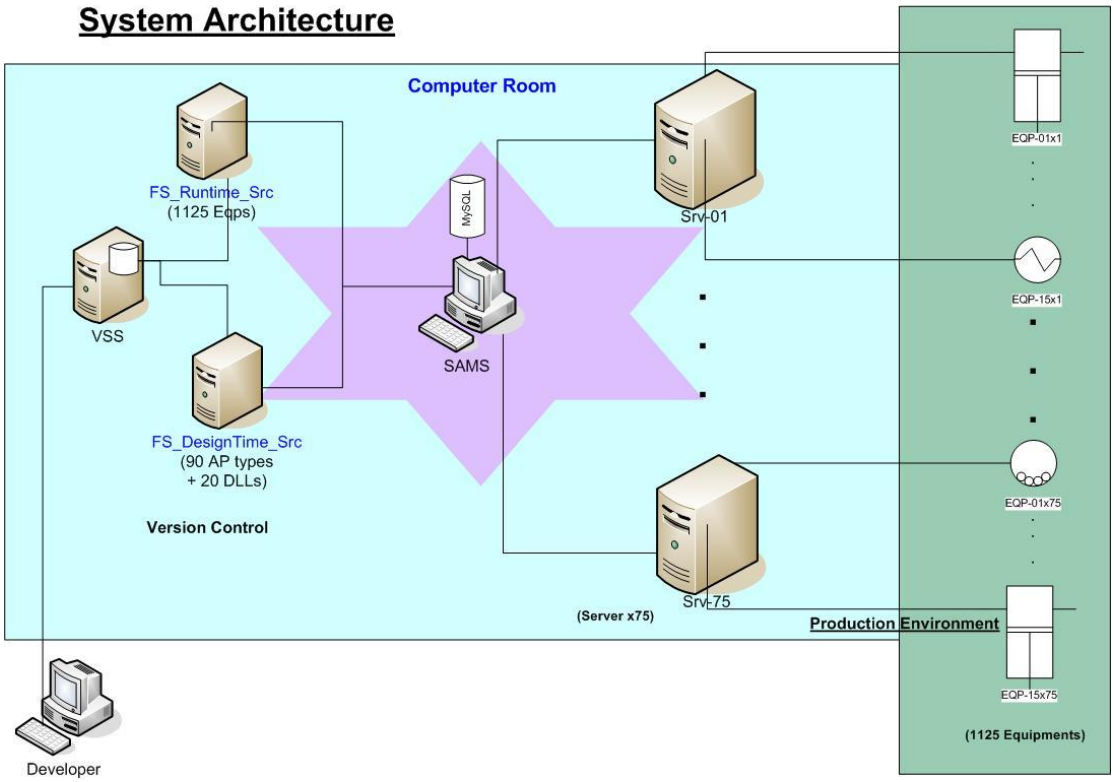

圖 2：系統架構圖

3.1 軟體架構

3.1.1 主程式軟體架構

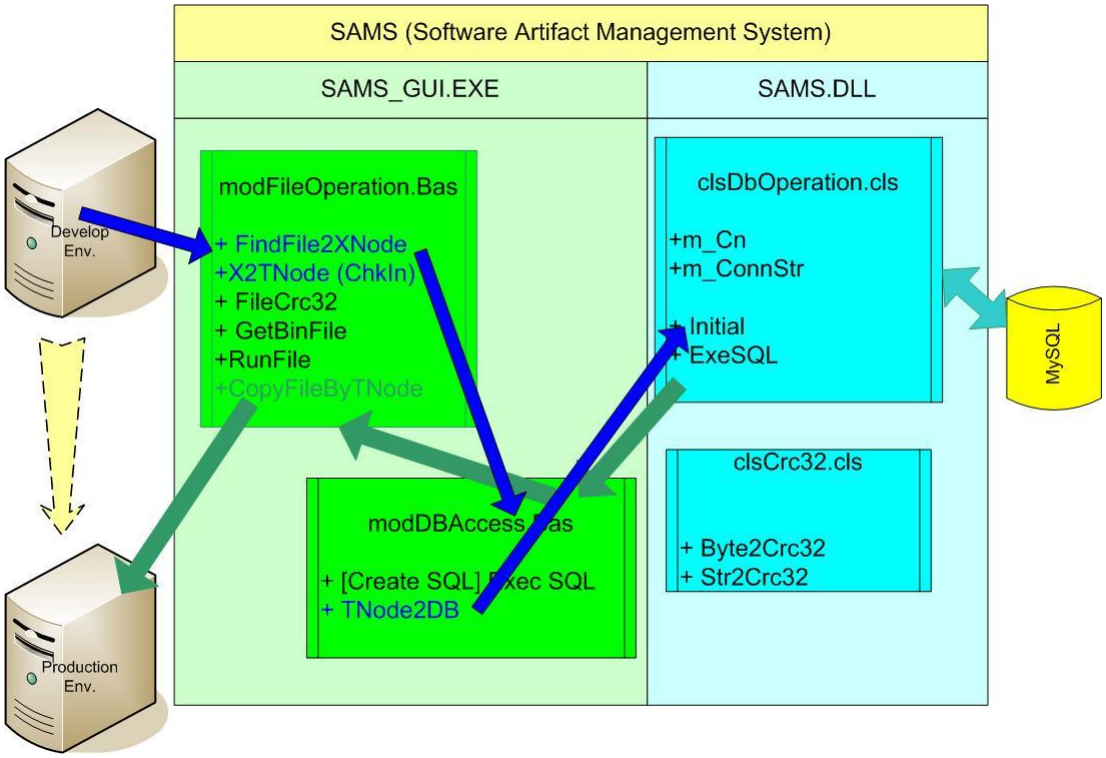

圖 3：主程式軟體架構圖

圖 3 中顯示此系統主要分為 SAMS_GUI.EXE（Client）和 SAMS.DLL（Server）兩部分。其中 XNode 為 XML 物件的節點而 TNode 則為 TreeView 的節點，此兩物件在資料的轉換與介面操作中間占了很重要的角色，也是實現專案管理的主要功臣。

本論文實作 CRC32 的運算，用以比對檔案進出資料庫的一致性，避免檔案因資料庫或人為問題而造成不一致的錯誤。

3.1.2 資料庫設計－MySQL 系統設定

本論文使用 Windows 環境開發故利用 MySQL 所提供的 MyODBC 應用軟體作為 ODBC 之驅動程式，而本論文要將實體檔案儲存於資料庫中所以需要用到 MySQL 的 BLOB 資料型態，以儲存包含非存純文字檔於資料庫中。而要存取 BLOB 的資料則需要使用連線選項設定。以下就 MyODBC 的連線設定(Connect String)做一簡介[8]:

1.user: 登入資料庫所使用的帳號

2.server: 資料庫所在的電腦名稱或 IP

3.database: 所要連線的資料庫名稱

4.port: 使用網路連線接口

5.option: 特性選項 (本論文使用 16427=16384 + 32 + 8 + 2 + 1)

16384: FLAG_NO_BIGINT

32: FLAG_DYNAMIC_CURSOR

8: FLAG_BIG_PACKETS

2: FLAG_FOUND_ROWS

1: FLAG_FIELD_LENGTH

除了以上連線特性選項設定要正確之外還需要了解 MySQL 的四種 BLOB 資料型態，其分別代表不同的儲存容量[14]

1. **Tiny BLOB**: L + 1 bytes, where L < 2^8 (255 bytes)
2. **BLOB**: L + 2 bytes, where L < 2^{16} (65K bytes)
3. **Medium BLOB**: L + 3 bytes, where L < 2^{24} (16M bytes)
4. **Long BLOB**: L + 4 bytes, where L < 2^{32} (4G bytes)

(L 代表實際儲存資料的 byte 數)

然而實際控制 BLOB 欄位的最大容量的是由 MySQL Server 中的 max_allowed_packet 參數，此參數儲存於 my.ini 的[mysqld]中。所以設計者需考慮系統所允許的最大檔案大小來決定使用何種 BLOB 資料形態再與 max_allowed_packet 的設定值一起搭配，方能將檔案正常的儲存於 MySQL 中。

故本論文使用 Long BLOB 資料型態且 max_allowed_packet 設定為 100M，如此資料庫便有儲存檔案大小在 100MB 以下的能力了。

3.1.3 資料庫設計－實體檔案與專案文件

如圖 4，左側兩個資料表 Filesystem 與 Filecontent 為實體檔案儲存於資料庫中的關連圖，其他三個資料表則為專案文件的關連圖，將實體檔案經由使用者權限的管理達成專案成員分工合作的目的。

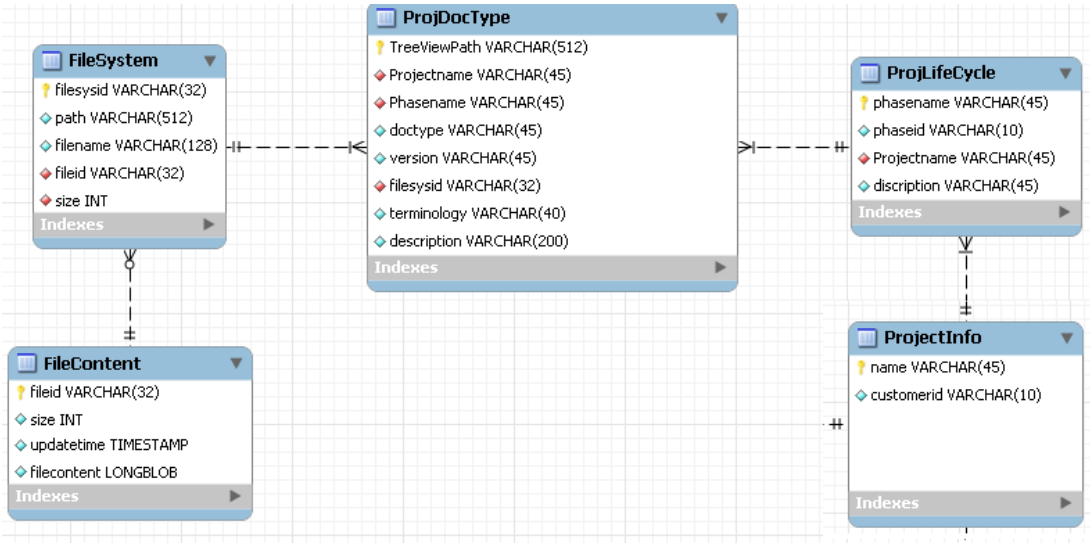

圖 4：資料庫關連圖－實體檔案之於專案文件

資料欄位設計說明如下：

1. FileSystem: 檔案系統資料表

   甲、filesysid: 檔案系統識別碼(Key, Varchar 10)

   乙、path: 實體檔案目錄

   丙、filename: 實體檔案名稱

   丁、fileid: 實體檔案識別碼

   戊、size: 實體檔案大小(byte 數)

2. FileContent: 儲存實體檔案內容資料表

甲、fileid: 實體檔案識別碼(Key, Varchar 32)

乙、size: 實體檔案大小(byte 數)

丙、updatetime: 實體檔案更新時間

丁、filecontent: 實體檔案內容 (BLOB)

以下簡述 BLOB 的儲存及讀取的方法:

儲存時，先利用 ADODB.Stream 的 LoadFromFile 方法將實體檔案讀進系統後，用 ADODB.Recordset 的 AddNew 方法先新增一筆空的資料，再用 ADODB.Stream 的 Read 方法取出檔案內容後設定給 ADODB.Recordset 的 BLOB 資料型態的欄位，最後再用 ADODB.Recordset 的 Update 方法將資料存入資料庫。相關程式碼請參閱附錄 C.1 之”將實體檔案存入 MySQL BLOB 欄位的範例”。

讀取時，先利用 SQL 查詢指令將資料從資料庫讀入 ADODB.Recordset 物件中，再將 BLOB 欄位的資料用 ADODB.Recordset 的 GetChunk 方法以 Byte Array 的型式取出，即可一個個 Byte 處理該檔案的內容。相關程式碼請參閱附錄 C.2 之”將資料從 MySQL 的 BLOB 欄位取出，並存成實

體檔案的範例”。

3.ProjDocType:專案文件形式資料表

甲、TreeViewPath: 專案文件顯示路徑(Key, Varchar 512)

乙、Pojectname: 專案名稱

丙、Phasename: 專案階段名稱

丁、doctype: 專案文件形式名稱

戊、Version: 檔案版本

己、filesysid: 檔案識別碼

庚、terminology: 專用術語

辛、description: 專案文件說明

4. ProjLifeCycle:專案生命週期資料表

甲、phasename: 專案階段名稱(Key, Varchar 45)

乙、phaseid: 專案階段識別碼

丙、Pojectname: 專案名稱

丁、description: 專案階段說明

5. ProjectInfo:專案資訊資料表

甲、name: 專案名稱(Key, Varchar 45)

乙、customerid: 客戶識別碼

每個專案在發展過程中都會有其相對應的專案文件產生，以上三個資料表是將專案文件以 filesysid 對應到實體檔案的設計。

3.1.4 資料庫設計－專案成員與功能管理

下圖 5 為專案成員權限管理的資料庫設計，系統管理者可依不同的專案定訂出每個使用者擔任專案的何種角色（User_Role）與該專案角色擁有哪些系統功能(RoleFunc)，讓檔案的存取更有效率與安全保障，這也讓專案成員能專心於自己相關領域檔案的維護。

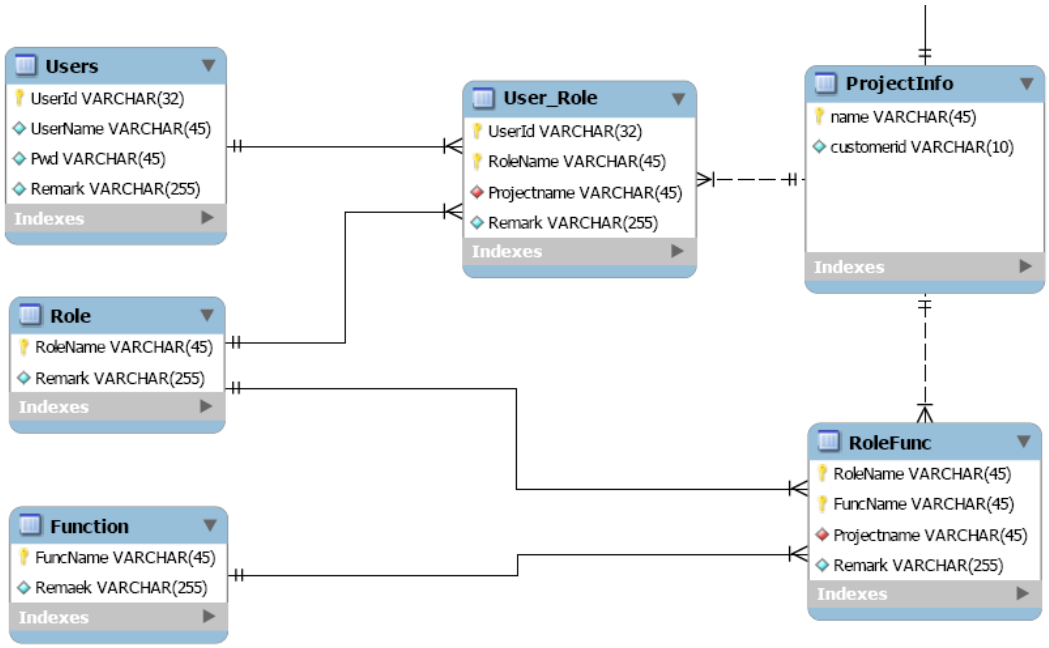

圖 5：資料庫關連圖－專案成員與功能管理

1. Users:使用者資料表

    甲、UserId: 使用者識別碼(Key, Varchar 32)

    乙、UserName: 使用者名稱

    丙、Pwd: 使用者密碼

    丁、Remark: 備註

2. Role:角色資料表

    甲、RoleName: 角色名稱(Key, Varchar 45)

    乙、Remark: 備註

3. Function:功能資料表

    甲、FuncName: 功能名稱(Key, Varchar 45)

    乙、Remark: 備註

4. User_Role: 使用者角色資料表

    甲、UserId: 使用者名稱(Key, Varchar 32)

    乙、RoleName: 角色名稱(Key, Varchar 45)

    丙、Projectname: 專案名稱

    丁、Remark: 備註

5. RoleFunc: 使用者角色資料表

甲、RoleName: 角色名稱(Key, Varchar 45)

乙、FuncName: 功能名稱(Key, Varchar 45)

丙、Projectname: 專案名稱

丁、Remark: 備註

3.1.5 資料庫設計－檔案佈署功能設定

下圖 6 為檔案佈署功能之資料庫關連圖，其中資料表 APType 為一機型連線程式的基本資料，而 ServerInfo 則是定義共有哪些電腦主機來執行所有設備的連線程式，最後 ApInfo 資料表為每一個單一設備連線程式的管理設定。所以自動佈署功能可由 ApInfo 得知哪一設備編號是用哪一機型的程式且可對應到其執行的電腦主機是哪一台。故不論是要佈署單一設備或是單一個機型所有設備的程式都可用程式來達成。

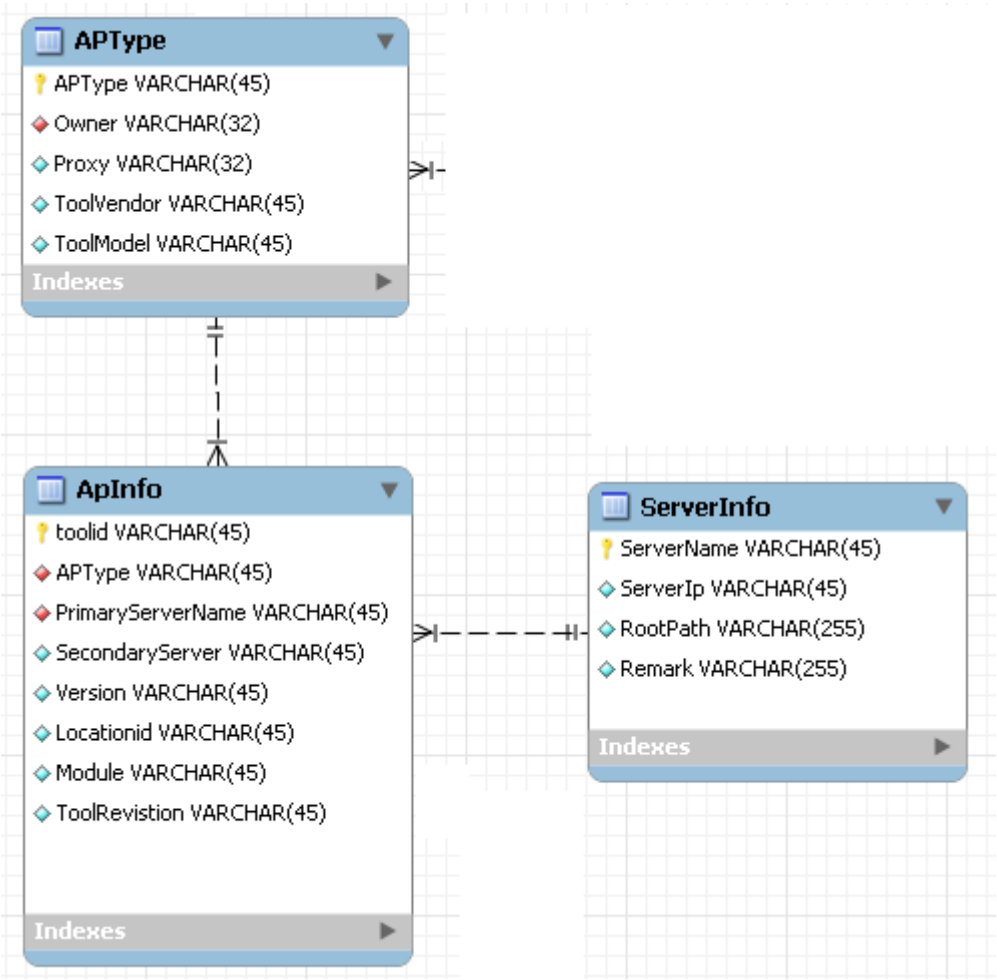

圖 6：資料庫關連圖－檔案佈署功能設定

1 APType: 單一機型之連線程式資料表

1.1 APType: 連線程式識別碼(Key, Varchar 45)

1.2 Owner: 程式維護人員

1.3 Proxy: 程式維護代理人

1.4 ToolVendor: 所連線的設備廠商名稱

1.5 ToolModel: 所連線的設備型號

2 ServerInfo: 電腦主機資料表

2.1 ServerName: 主機名稱(Key, Varchar 45)

2.2 ServerIp: 主機網路 IP

2.3 RootPath: 該主機分享出來的執行環境之根目錄

3 ApInfo: 單一設備之管理資訊資料表

3.1 toolid: 單一設備識別碼(Key, Varchar 45)

3.2 APType: 連線程式識別碼

3.3 PrimaryServerName: 主要執行主機名稱

3.4 SecondaryServer: 次要(備源)執行主機名稱

3.5 Version: 連線程式版本

3.6 Locationid: 設備實體位置

3.7 Module: 設備負責單位

3.8 ToolRevision: 設備版本

其中次要(備源)執行主機通常不執行該設備之連線程式，但會是其他設備的主要執行主機，也就是平時 75 台電腦主機負責 1000 台設備之連線，但當其中一機房有狀況而無法繼續工作時，須將相關的程式切換到另一機房去執行，如此就是用一半的電腦主機

負責全廠 1000 台設備的連線程式，只是此時的效能無法要求太高。另外當任一台主機故障或待維修時相關程式可臨時切換到次要執行主機上執行，故檔案佈署時也要記得佈署到次要執行主機上，以免無法臨時切換而造成產能的疏漏。

3.2 系統功能列表

此系統之功能分為通用功能、管理功能與檔案功能三類，每個功能都賦予一個代號，此功能代號之命名規則為前兩碼是功能分類在加上三碼數字作為序號:

1 通用功能: 是以 GF(General Function) 開頭，再加三碼數字 (表 1)

2 管理功能: 是以 MF(Manage Function) 開頭，再加三碼數字(表 2)

3 檔案功能: 是以 FF(File Function) 開頭，再加三碼數字(表 3)

各項功能可依登入的專案角色而提供相對應的專案功能畫面，故在登入時除了使用者名稱及密碼外還要選擇登入哪一個專案。

通用功能說明: (表 1)

所有使用者共同擁有下表 1 的通用功能，也就是每一個使用者都可以看到登入畫面且登入成功後可執行登出或結束程式。

表 1：系統功能表－通用功能及角色權限設定範例

Function Id	Function name	Function name	Administrator	SA/SD	QA	PM
GF001	Login/Logout	登入/登出系統	V	V	V	V
GF002	Exit	結束程式	V	V	V	V

V 代表該角色擁有此功能

管理功能說明: (表 2)

此類功能是供系統管理者或專案管理者設定基本參數與策略性參數使用，其又分為以下兩類:

1.權限參數設定功能 (MF001~5): 先由系統管理者透過 MF001~3 設定好基本參數，再由專案管理者透過 MF004 指定哪些使用者擔任何種專案角色及 MF005 設定專案角色擁有哪些功能等策略性參數。

2.佈署參數設定功能(MF006~8): 先由專案管理者透過 MF006~7 設定好基本參數，再透過 MF008 指定哪些設備程式屬於哪一程式且該於哪一個主機上執行。

表 2：系統功能表－管理功能及角色權限設定範例

Function Id	Function name	Function name	Administrator	SA/SD	QA	PM
MF001	User management	使用者管理	V			V
MF002	Role management	角色管理	V			V
MF003	Function management	功能管理	V			

Function Id	Function name	Function name	Administrator	SA/SD	QA	PM
MF004	Project User/Role management	專案使用者角色管理	V			V
MF005	Project Role/Function management	專案角色功能管理	V			V
MF006	Server management	電腦主機管理	V			V
MF007	AP Type management	專案程式管理	V			V
MF008	Tool Id management	設備程式管理	V			V

V 代表該角色擁有此功能

檔案功能說明: (表 3)

此類功能是提供系統維護人員先將改版後的檔案匯入系統，再由 QA 測試人員接手並佈署到測試環境進行整合測試使用，當測試完成後再佈署到生產環境。其中限制系統維護人員不可擁有佈署功能主要是避免球員兼裁判之嫌，並藉以增加軟體測試的品質。

表 3：系統功能表－檔案功能及角色權限設定範例

Function Id	Function name	Function name	Administrator	SA/SD	QA	PM
FF001	Import	檔案匯入	V	V		V
FF002	Export	檔案匯出	V		V	V
FF003	Deploy/Trial Run	佈署到測試環境	V		V	V
FF004	Deploy/Release	佈署到生產環境	V		V	V

V 代表該角色擁有此功能

3.3 主畫面功能

1. 下圖 7 之上方為 Menu bar: 大部分的功能由此開始，其功能敘述請參閱 ”附錄 B: 操作畫面及說明” 。
2. 下圖 7 之左方為 TreeView: 用來顯示儲存於資料庫的檔案，在 Project 上按右鍵可顯示 Import/ Export 等專案檔案管理功能。
3. 下圖 7 之右方為 DirListView 與 FileListView: 顯示實體檔案的內容，共定義兩種不同目的實體檔案，一為開發環境的實體檔案，底色為綠色；另外則是執行(測試)環境的實體檔案，底色為紅色。
4. 下圖 7 之下方為 Status bar: 用來顯示目前所登入的專案名稱、登入者名稱與功能執行狀態。

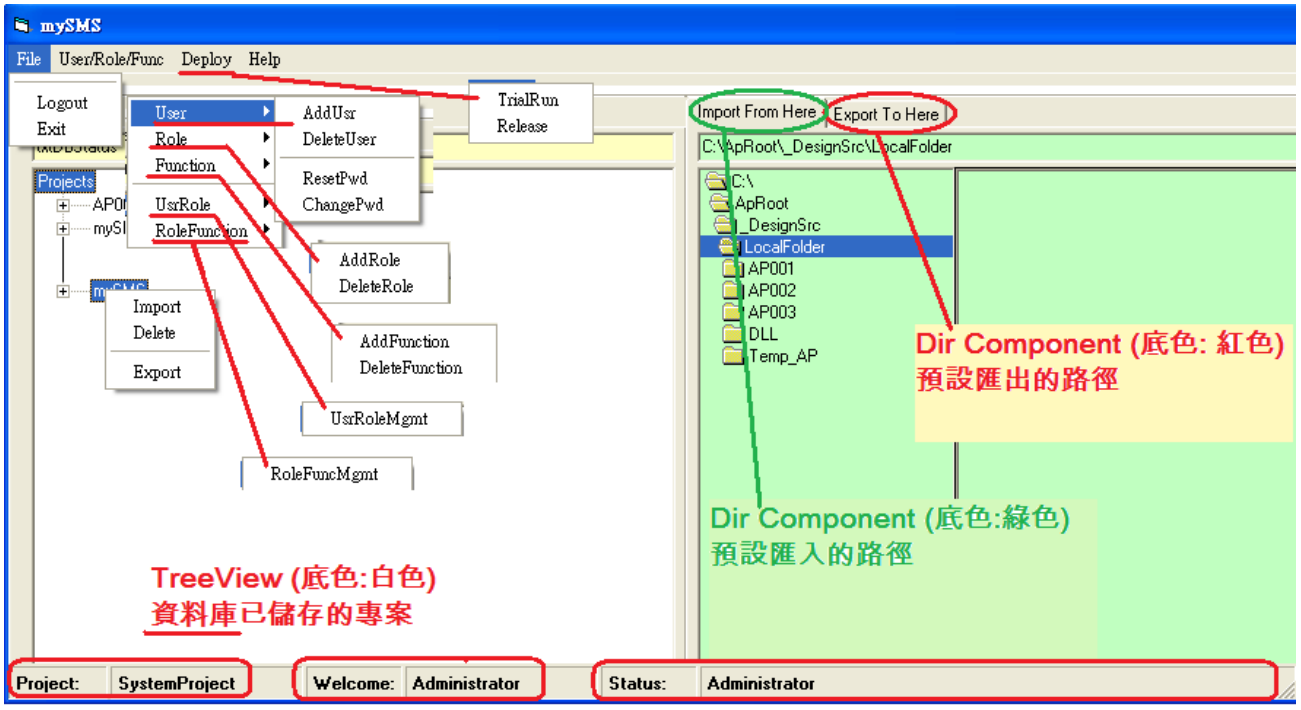

圖 7：主畫面功能

第四章　案例研究

在本章將透過範例說明如何使用此系統，從建立一個軟體專案開始到專案成員的權限控管、專案程式執行環境的規劃、專案檔案匯入、匯出與佈署，其中專案成員的權限控管與專案程式執行環境的規劃通常只在專案建立時做一次，不會因程式改版而修改，而檔案匯入一般是由軟體維護者於每次改版後整合測試後匯入資料庫，但其無法做釋出佈署的動作，必須由品管人員做過線上整合測試之後再作釋出佈署的動作。

4.1 建立新專案

以新購入一批同型設備(設備編號 DEMO_E1~5)，但須新開發一個連線程式為例，系統管理者先用 Import 功能(附錄 B.3.1)從實體檔案建立專案: Proj_A，此時系統會依照實體檔案之結構建立當時的檔案於 ProjDocType 資料表中，供往後的開發人員與測試人員使用。

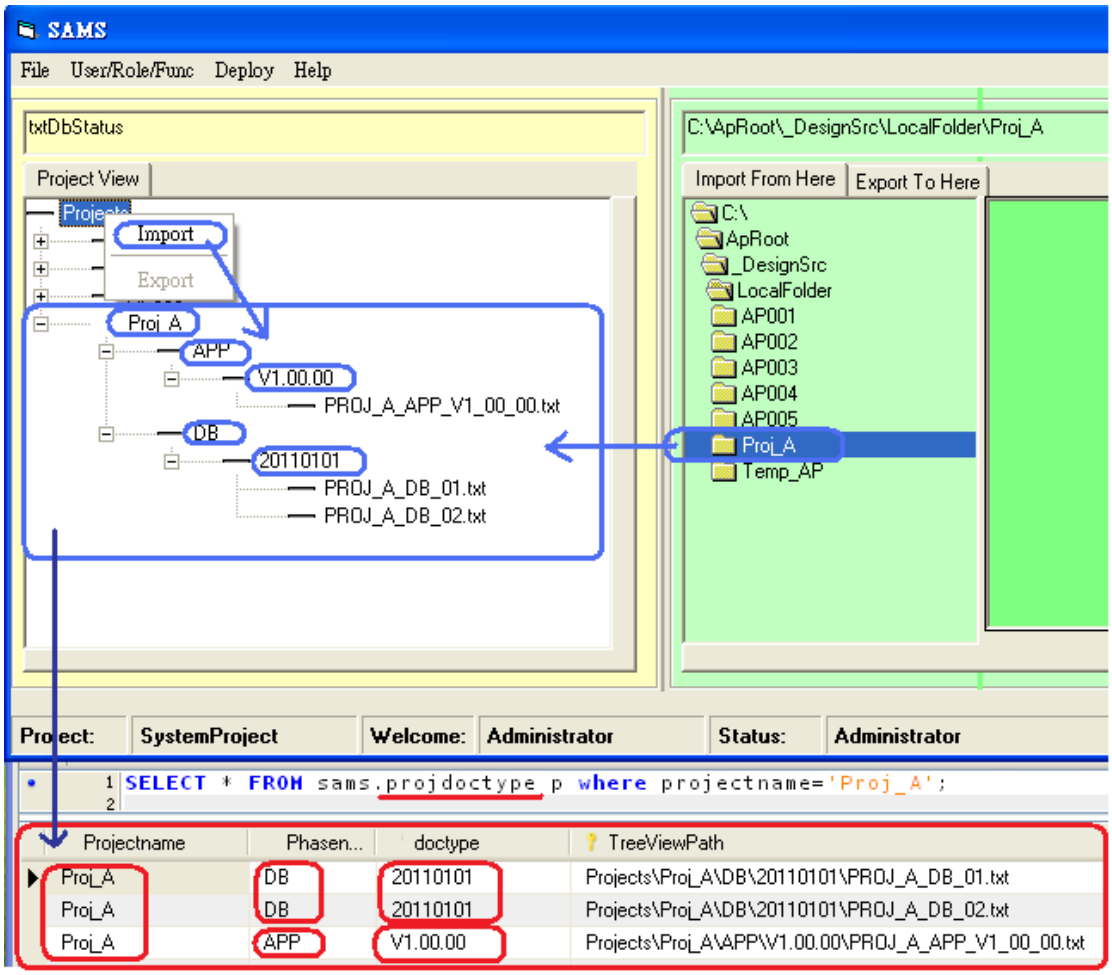

圖 8：從實體檔案建立新專案

4.2 專案成員的權限控管

專案 Proj_A 建立後，系統管理者可使用 ”使用者-角色-功能” 等管理功能（附錄 B.2.4 及 B.2.5）建立專案成員的權限，在此設定使用者 QA_01 在 Proj_A 專案中擔任 QA_ENG 的角色，而 SD_01 擔任 SD 的角色(如圖 9)。 再設定角色 QA_ENG 擁有 Export、TrialRun 和 Release 功能，而 SD 角色只擁有 Import 功能(如圖 10)。

系統將於使用者登入後除了要保留通用功能外(重新登入及個人密碼維護等)，還要根據以上設定取消未設定的功能，保留已設定的功能(如圖 11)。

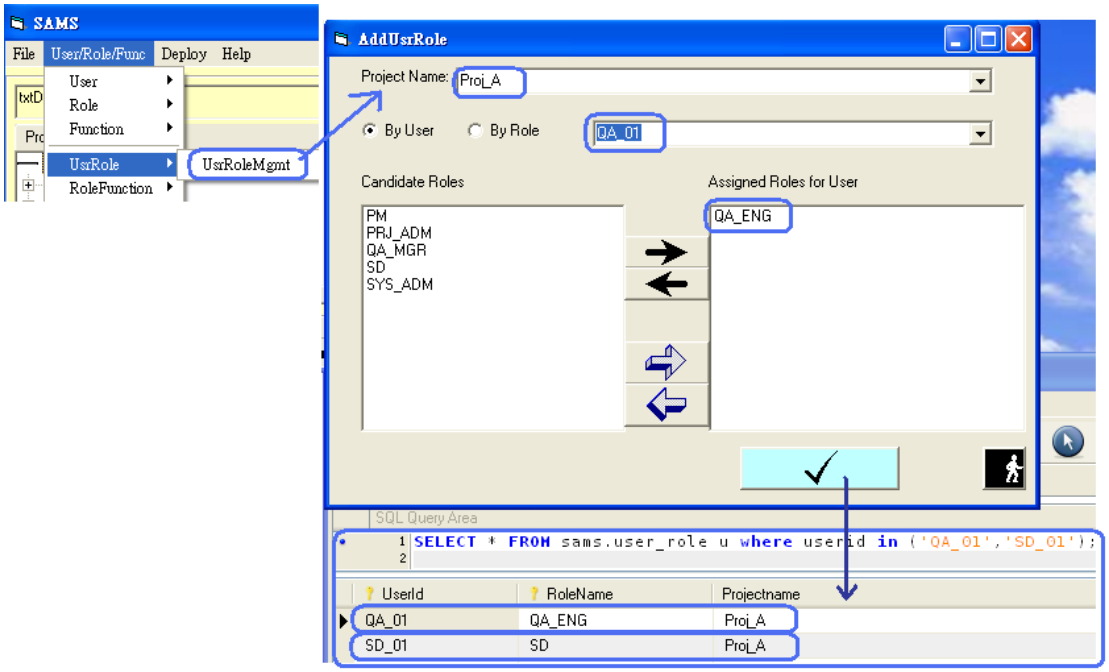

圖 9：設定使用者再專案裡所擔任的角色

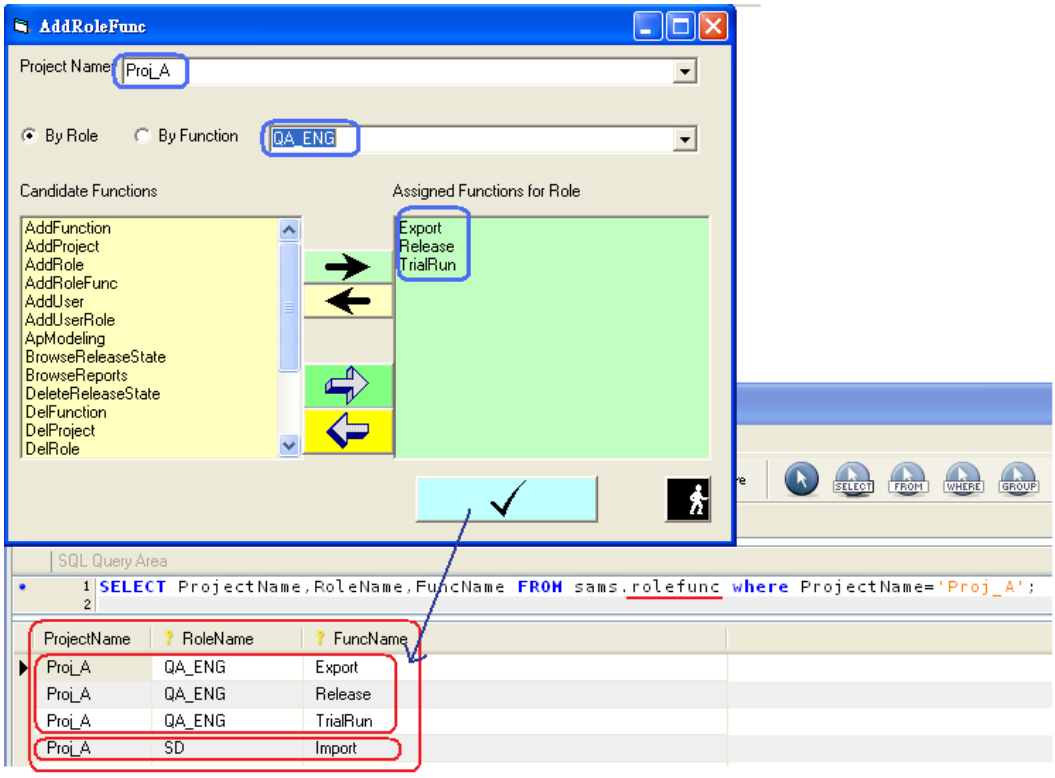

圖 10：設定專案角色擁有的功能

如此即可強制限制系統開發或維護者 SD_01 無法自行釋出程式，程式必須透過 QA_01 先經過測試後再釋出，避免 SD_01 程式開發者自行測試後就釋出的情形發生，以彰顯分工合作成效。

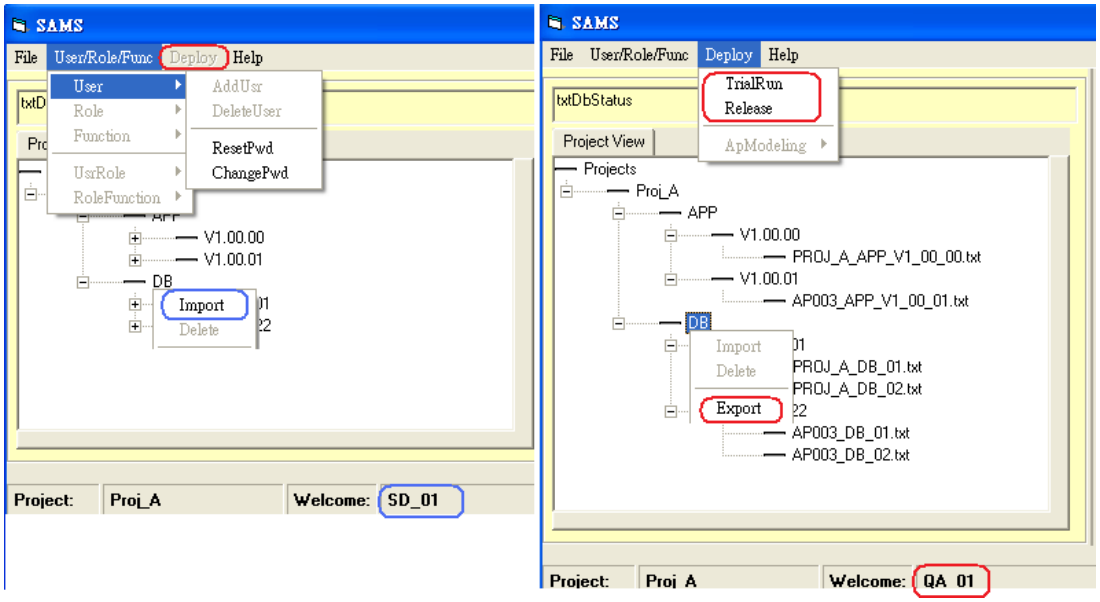

圖 11：SD_01 與 QA_01 之功能差異

4.3 程式執行環境的規劃

接下來做系統程式執行環境的佈署設定，若有新購入電腦主機則須先以電腦主機管理功能(附錄 B.2.6) 將新加入之電腦主機(SRV001~5)輸入系統。再以專案程式管理功能(附錄 B.2.7) 新增 Proj_A 程式並設定 SD_01 為其維護者(如圖 12)。

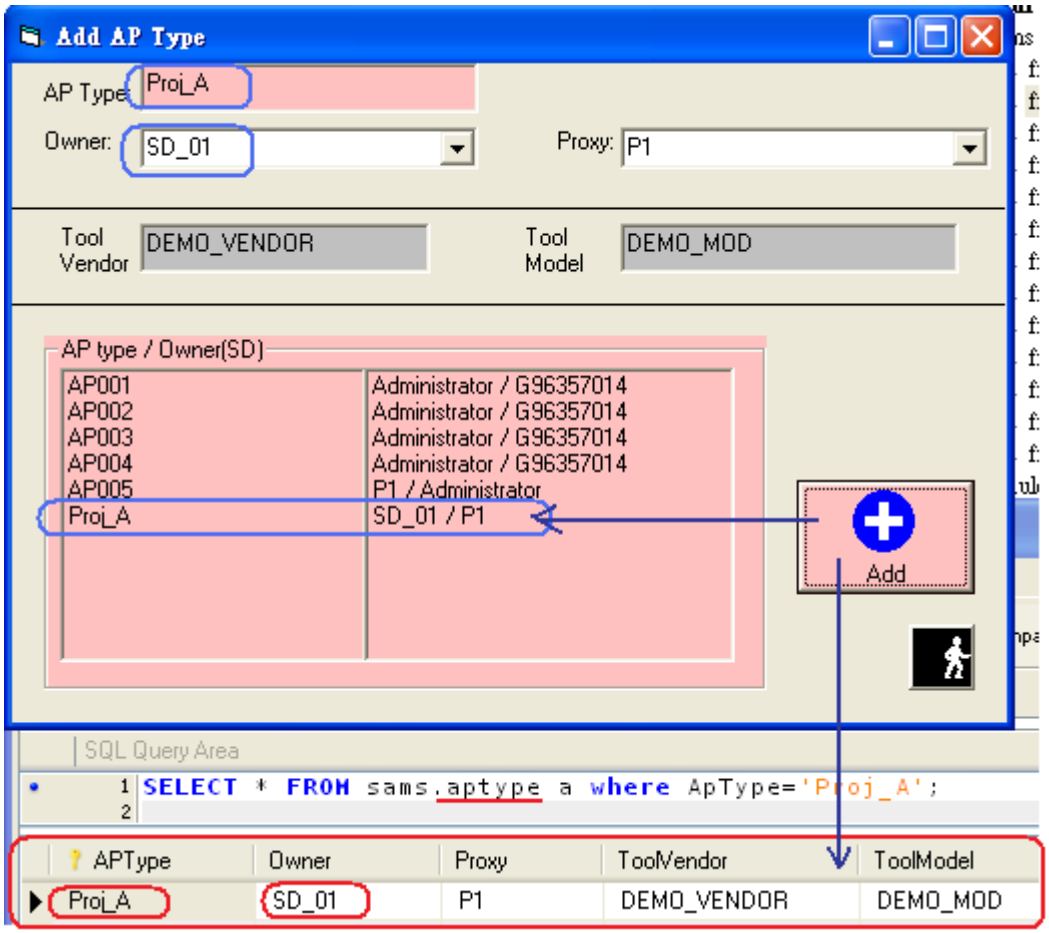

圖 12：設定專案程式的維護者為 SD_01

再設定此專案程式之識別碼(在此使用設備編號)，也就是 Proj_A 共要執行幾次(每次執行連線到不同的設備)，並規劃每個設備編號的程式要在哪一台主機上執行。

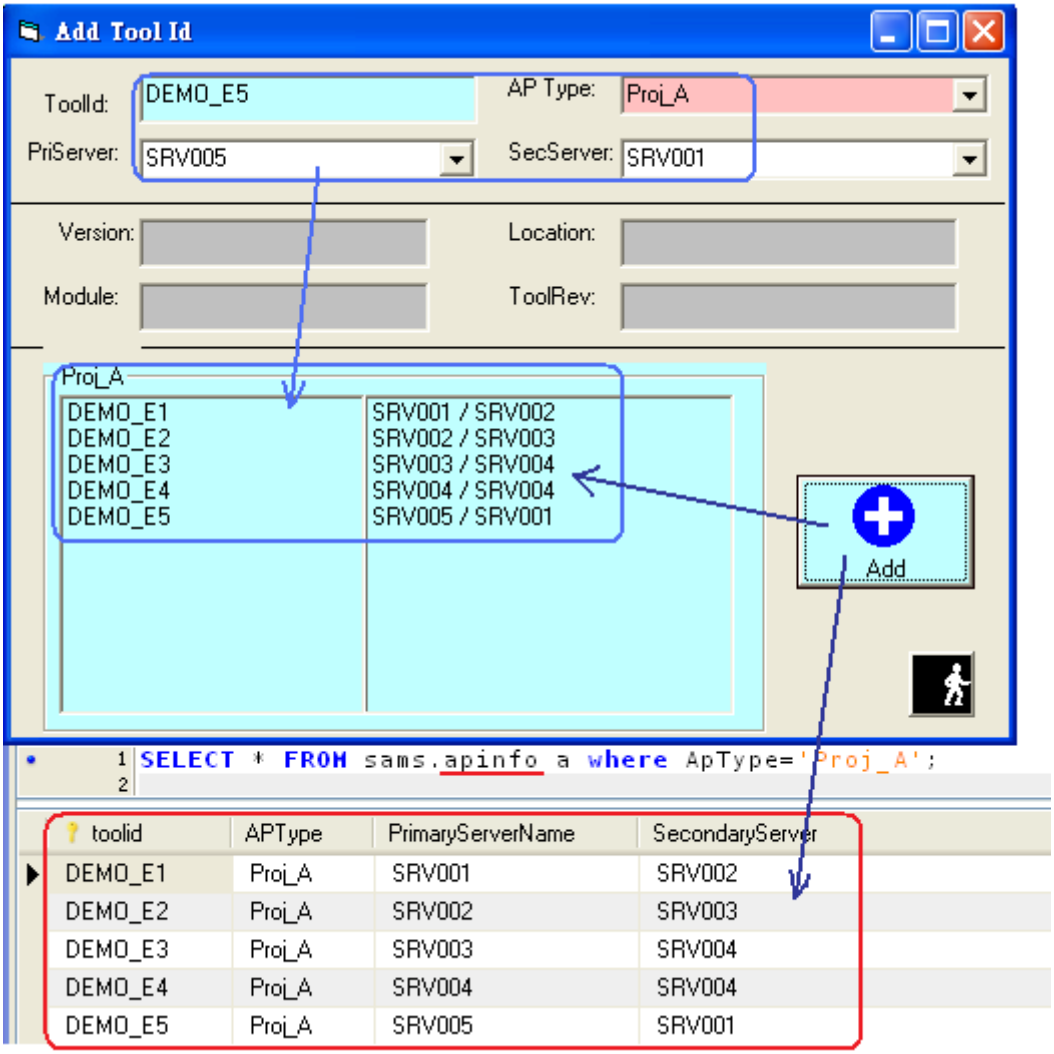

圖 13：設定專案程式的維護者為 SD_01

4.4 檔案匯入及匯出

A. 檔案匯入: Import（附錄 B.3.1）軟體維護者 SD_01 改版並整合測試後於 Proj_A 中匯入 V1.00.01 版的 APP 及兩個置於 20110622 的 DB。

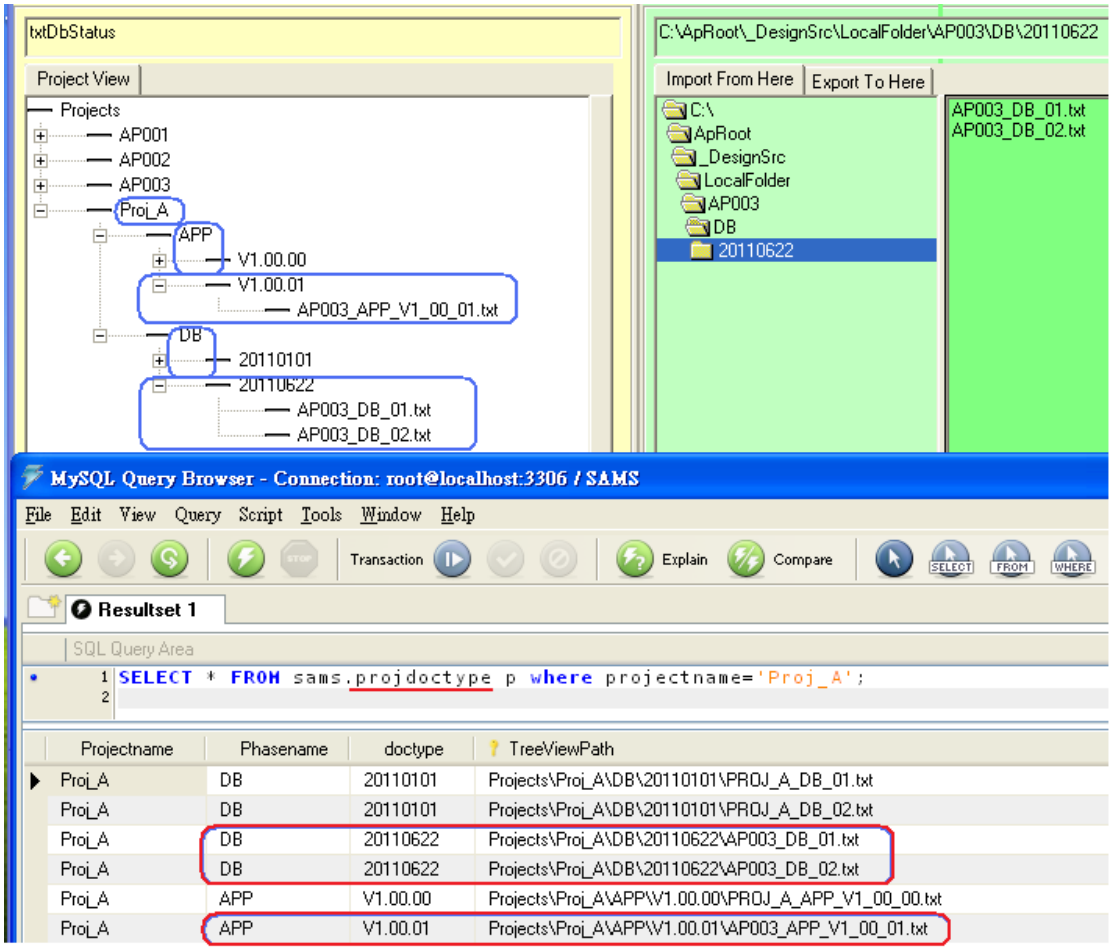

圖 14：SD_01 匯入新版程式與設定

B. 檔案匯出: Export（附錄 B.3.2），QA_01 匯出 SD_01 所匯入的檔案進行測試。

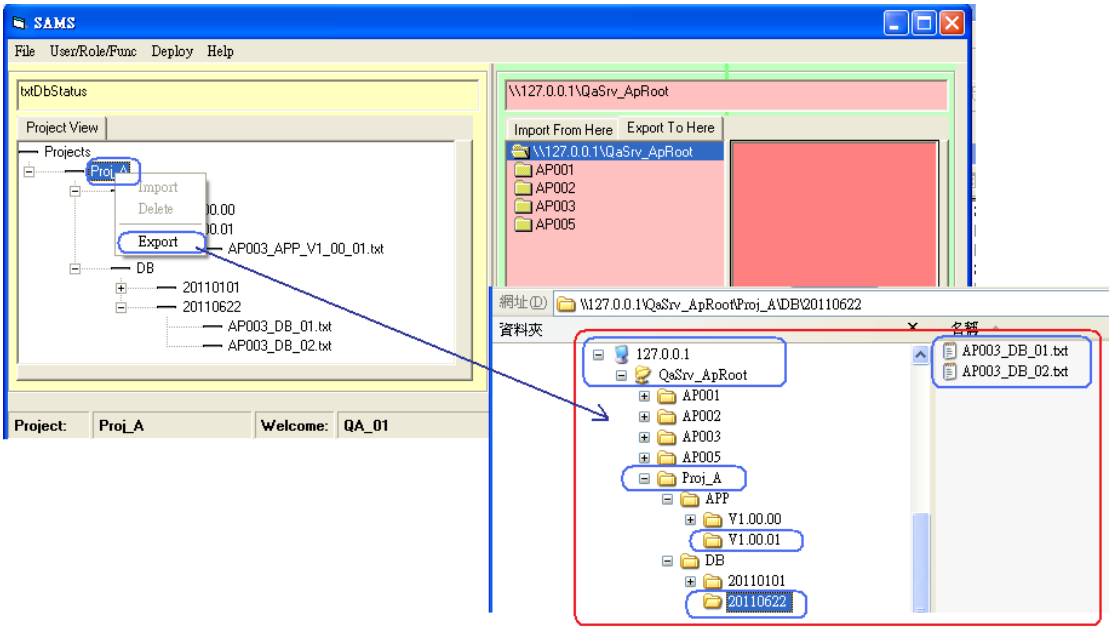

圖 15：QA_01 匯出新版程式與設定

4.5 檔案佈署

A. 程式測試佈署: Trial Run(附錄 B.3.3)，當在測試環境測試完成後，即可選定一設備編號(DEMO_E1)進行實際連線測試，按照之前的設定，按下[Deploy]鍵後 DEMOO_E1 的程式將自動佈署到 SRV001 與 SRV002 上進行線上測試。

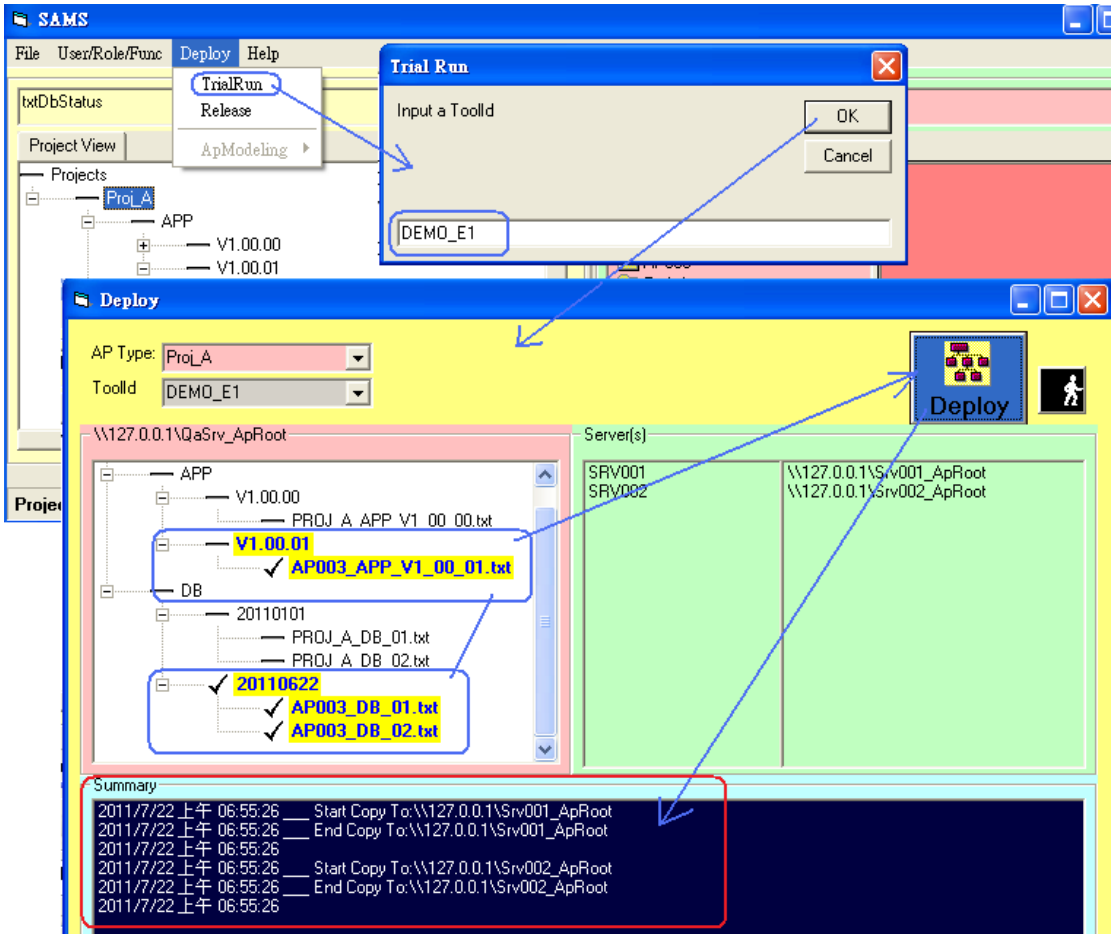

圖 16：QA_01 佈署新版程式與單機設備連線測試

B. 專案釋出佈署: Release(附錄 B.3.4)，當在單機實際連線測試測試完成，且確認沒問題後即可將所有設備套用此版本，在釋出佈署完成之後就等程式重新啟動，則代表新功能完全上線生產。

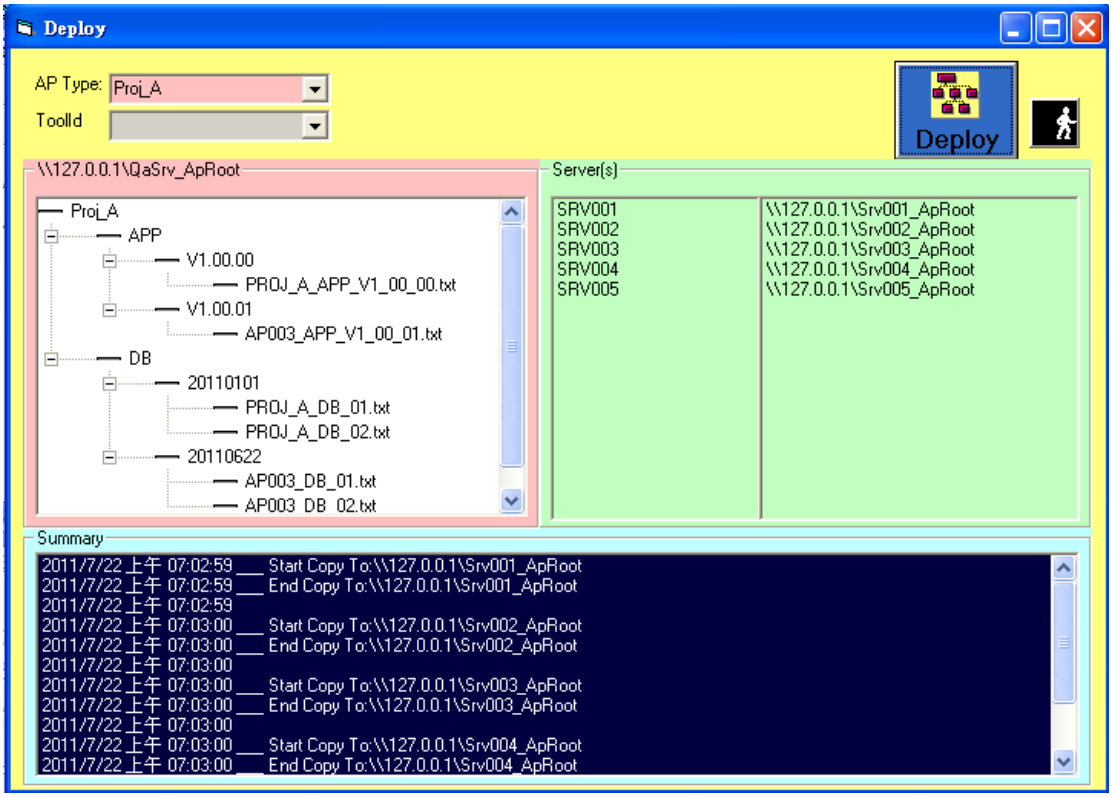

圖 17：QA_01 佈署新版程式到所有相關設備進行連線生產

第五章　結論與未來方向

根據第四章案例研究可以發現比較花時間的規劃及設定工作只須在專案初期做一次就可以免除後續新版程式的繁瑣的佈署問題，可縮減了軟體上線的時程與減少錯誤的發生，預計每次釋出軟體花在檔案佈署的時間將由 30 分鐘降為 5 分鐘以下。

以民國九十九年第二季為例，程式改版共有 56 個版次，外加 11 個共用函式庫，但實際佈署數量卻達 8693 次，雖然隔年(民國一百年第二季)因市場不景氣與系統日趨穩定使得佈署量僅剩三分之一左右，但也還有 3193 次的佈署量。這對工程師來說亦是可觀的數字。

表 4：民國九十九年第二季軟體佈署次數

Deploy Cnt	DB	APP	DLL	
201004 合計	2348	93	235	2676
201005 合計	3269	107	384	3760
201006 合計	2744	132	152	3028
總計	**8361**	**332**	**771**	**8693**

表 5：民國一百年第二季軟體佈署次數

Deploy Cnt	DB	APP	
201104 合計	731	66	797
201105 合計	1315	61	1376
201106 合計	939	81	1020
總計	**2985**	**208**	**3193**

以民國九十九年第二季軟體佈署總檔案數 8693 來看，平均每次佈署檔案以 30 個計，故有 290(= 8693/30)次佈署作業而每次佈署可節省 25 分鐘，則每季可節省 7250 (= 290*25)分鐘，也就是 120 小時，約每週約可節省 10(=120/3/4)小時的時間，以當時四位工程師平均每人每週可多出 2.5 小時的時間可以利用。

本論文是針對筆者目前工作實務上遇到的問題所提出的解決方案，不過如果其他 Windows 系統要使用此系統應無問題才對，因為只要可分享網路目錄作為專案根目錄，使用者即可自行規劃根目錄以下的檔案結構做為佈署目標。

另本論文已將檔案以 byte 為單位讀入系統並計算 CRC32 後就直接將檔案存入資料庫中，未來可加入邏輯將與前一版本做比較，最後存入資料庫的只有差異部分，而當要取出時再利用第一版加上各版之間的差異部分組合出所指定的版本。還有 Checksum 的計算也可嘗試實作 CRC64 以上的演算法，甚至是 MD5 或 SHA-1 的演算法。

基本上，使用自動化系統來讓軟體維護更有效率的進行是需要的，且可確實控管在釋出/交付使用之前的測試，以降低在最後關頭人為的疏失而造成重大的損失，讓軟體維護的過程與結果更為完善。

參考文獻

1 IEEE, *IEEE Standard for Software Maintenance* (IEEE Std 1219-1993), Institute of Electrical and Electronic Engineers, Inc., N *Annals of Software Engineering*, vol. 3, 1997, pp. 319-343NY,pp 39,1993.

2 Chu, W.C, Chang C.H., Lu C.W., Jiau H.C., Chung Y.C., and B. Qiao, "Enhancing software maintainability by unifying standards," To appear in Advances in Software Maintenance Management: Technologies and Solutions, Hershey PA: Idea Group Publishing.

3 Anne, C.L. (1999). XML seen as integral to application integration. *IT Professional, 2(5)*, 12-16

4 Deitel, H., Deitel, P., Nieto, T., Lin, T., & Sadhu, P. (2001). XML *how to program*. Upper Saddle River, NJ : Prentice Hall.

5 Sommerville, I. (2006). *Software Engineering*, 7th *edition*. Reading, MA.: Addison-Wesley, 4-19.

6 IEEE, *IEEE Standard for Software Life Cycle Processes- Maintenance* (IEEE Std 14764-2006)

7 MySQL 官方網站：http://www.mysql.com/

8 MySQL ODBC Connection Parameters http://mysql.biz.net.id/doc/refman/5.1/en/myodbc-configuration-connection-parameters.html

9 CRC32, Wikipedia http://zh.wikipedia.org/wiki/CRC32

10 Swanson. E. B. (1976), "The Dimensions of Maintenance." Proceedings of the Second International Conference on software Engineering, San Francisco, California, USA, pp. 492-97

11 Pressman, R. S. (1987), Software Engineering: A Practitioner's Approach, New York:McGraw-Hill.

12 Anne, C.L. (1999). XML seen as integral to application integration. *IT Professional*, 2(5), 12-16.

13 XML Technology, http://www.w3.org/standards/xml/

14 MySQL 5.1 Reference Manual, 10.5 Data Type Storage Requirements http://mysql.borsen.dk/doc/refman/5.1/en/storage-requirements.html

15 Arthur L.J., "*Software Evolution: The Software Maintenance Challenge*", NY: John Wiley Sons, 1988.

16 軟體工程：從實務出發(初版) 鄭炳強著

17 W3C,"Extensible Markup Language (XML)1.0", 1998.（http://www.w3.org/TR/1998/REC-xml-19980210）

18 SEMI, "SEMI equipment communications standard 2 message content", GlobalInformation Control Committee，SEMI E5-0299，1998.

19 SEMI, "Generic model for communications and control of manufacturing Equipment"，Global Information Control Committee，SEMI E30-0600，2000.

20 SEMI, "High-Speed SECS Message Service (HSMS) Gerneric Service", Global Information Control Committee，SEMI E37-702，2002.

附錄 A: 系統設定檔

本附錄說明系統設定檔之內容。

A.1. SAMS.xml

此設定檔由系統管理者維護，該檔案於程式起始時會被讀入作為系統的預設值，在此設定檔內只看到資料庫連線的資訊，系統管理者可依實際執行環境自行修改之。

```
<?xml version="1.0"?>
<Message xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance Command="MYSMS_CONFIG"
DateTime="">
    <DBInfo>
        <HostName>localhost</HostName>
        <DBType>MySQL</DBType>
        <IP>127.0.0.1</IP>
        <SID>SAMS</SID>
        <UsrName>root</UsrName>
        <Pwd>1111</Pwd>
        <Driver>{MySQL ODBC 5.1 Driver}</Driver>
        <Option>16427</Option>
    </DBInfo>
</Message>
```

附錄 B: 操作畫面及說明

B.1.通用功能

B.1.1. 登入/登出（GF001）：登錄功能

程式執行時首先顯示登入畫面，使用者選擇要登入的專案名稱再輸入用戶名稱及密碼，再按[∨]鍵後即開始驗證帳號與密碼，成功後即可進入主畫面，操作該使用者於此專案的功能。若欲登出則至 File 中執行 Logout 功能即可離開主畫面並切換至登入畫面。

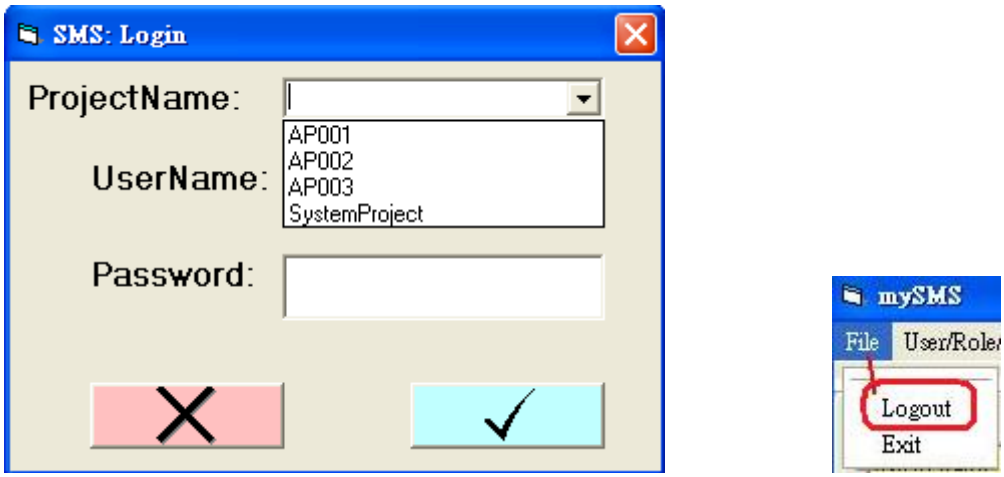

圖 18：登入/登出畫面

B.2.管理功能

B.2.1.使用者管理（MF001）：由 UserRoleFunc/User 進入。

共有四項子功能畫面，其中 ChangePwd 功能則是共用 AddUsr 功能之畫面，不同的是在 ChangePwd 時系統會自動帶入 UserId 及 UserName，使用者只須輸入新密碼並確認。此 AddUsr 畫面會帶出所有 UserId/UserName 供使用者參考。

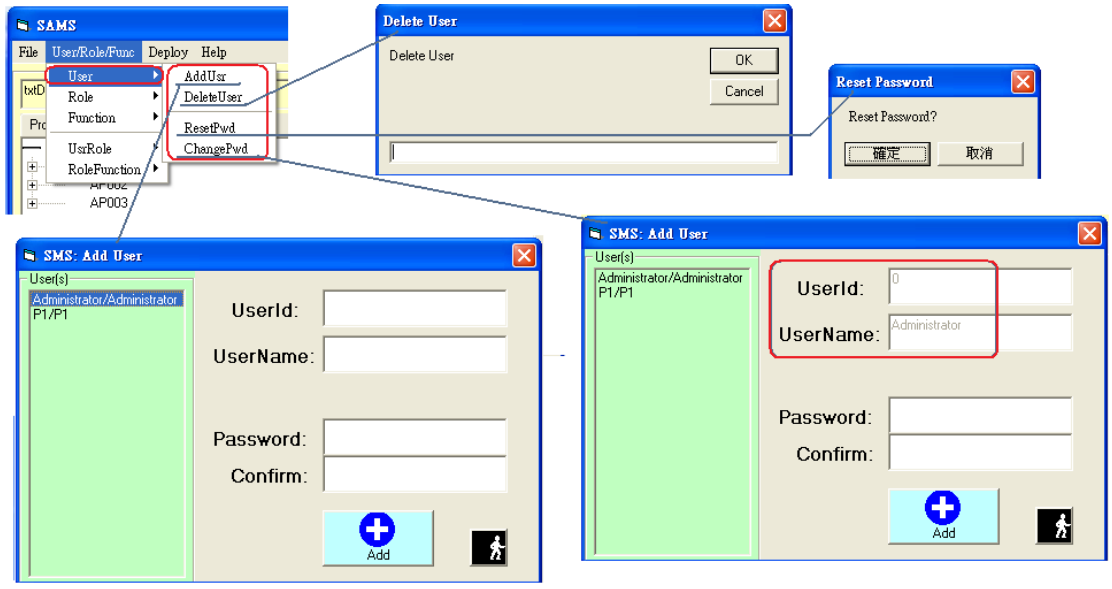

圖 19：使用者管理畫面

子選項說明:

I. AddUser: 新增使用者需輸入識別碼、名稱、密碼及其確認碼後再按[Add]鍵即可新增一使用者。

II. DeleteUser: 僅需輸入使用者名稱按[OK]鍵即可刪除一使用者。

III. ResetPwd: 按[確定]鍵後即可將密碼設成與使用者名稱相同。

IV. ChangePwd: 輸入密碼及其確認碼後按[Add]鍵更改密碼。

B.2.2. 角色管理（MF002）：由 UserRoleFunc/Role 進入。

共有兩項子功能畫面，其中 AddRole 功能會帶出所有 RoleName 供 Add 時參考。

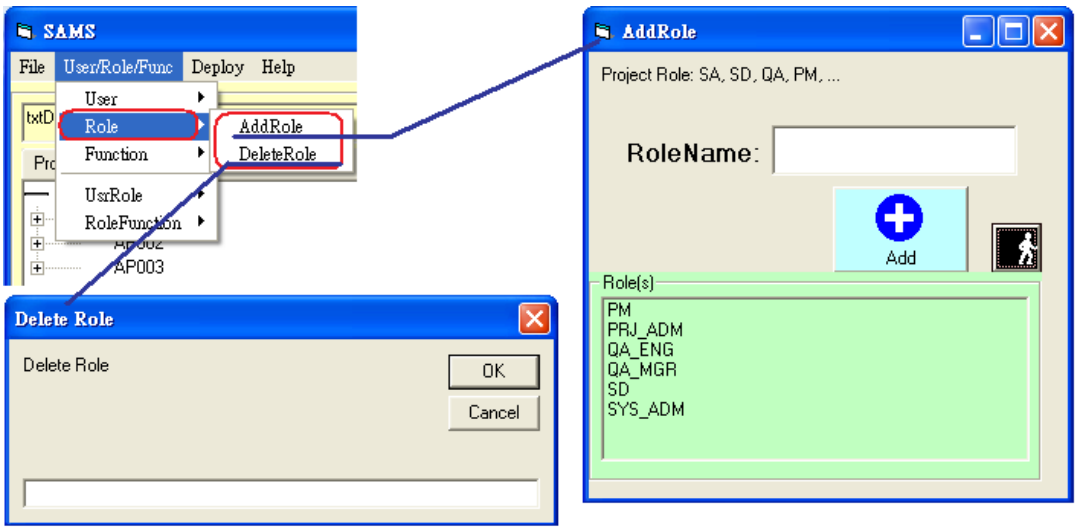

圖 20：角色管理畫面

子選項說明:

I. AddRole: 輸入角色名稱按[Add]鍵即可新增該角色。

II. DeleteRole: 輸入一個角色名稱按[OK]鍵刪除該角色。

B.2.3.功能管理（MF003）：由 UserRoleFunc/Function 進入。

共有兩項子功能畫面，其中 AddFunction 功能會帶出所有

FunctionName 供 Add 時參考。

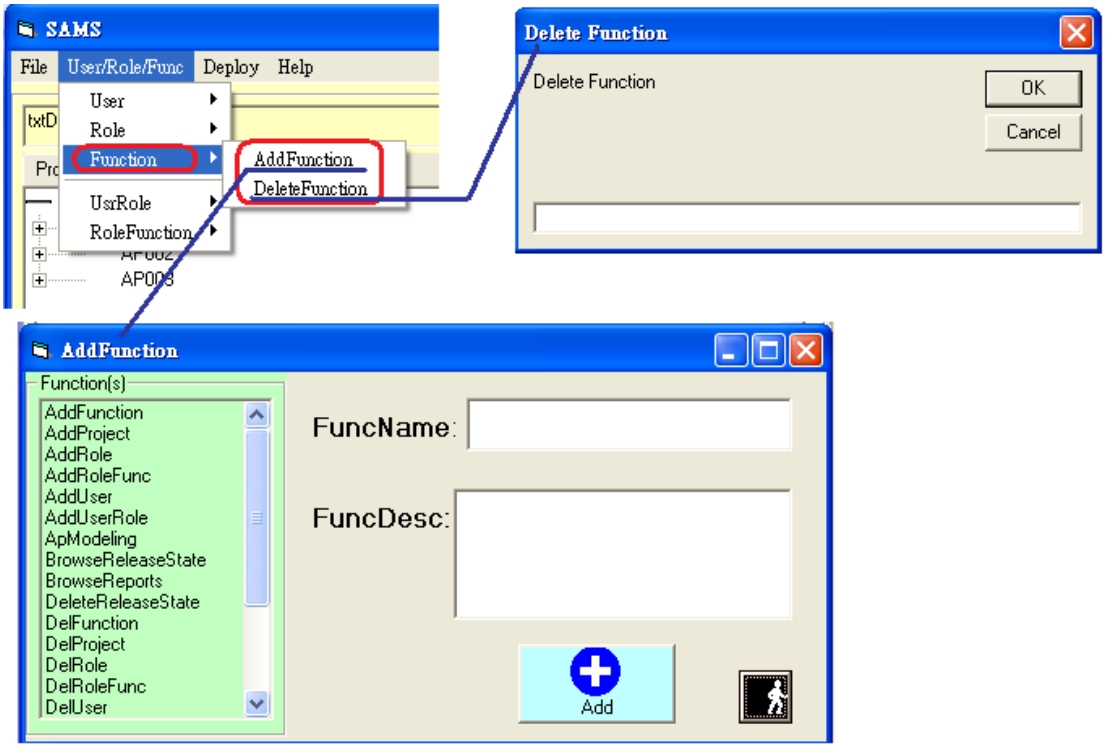

圖 21：功能管理畫面

子選項說明:

I. AddFuncion: 輸入功能名稱按[Add]鍵即可新增該功能。

II. DeleteFunction: 輸入功能名稱按[OK]鍵刪除該名稱。

B.2.4. 專案使用者角色管理（MF004）：由 UserRoleFunc/UserRole 進入。

此功能僅做使用者與角色的對應工作故無新增、刪除等功能，只有 Assigned 或 NotAssigned。

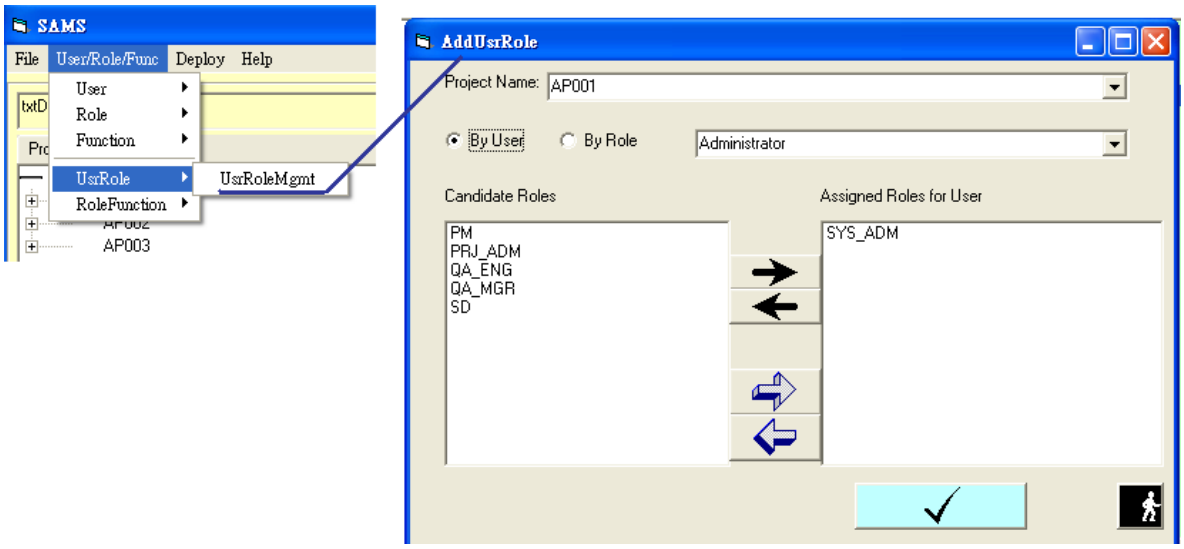

圖 22：專案使用者角色管理畫面: User- Role

操作說明:

I. Project Name: 選定要設定的專案名稱。

II. By User 或 By Role 選項: 決定要以使用者或角色為根本。

II.1 ByUser: 設定單一使用者要加入哪些角色。

II.2 ByRole: 設定單一角色由哪些使用者擔任。

III. 雙擊下方兩個方框內容(Candidate Role 或 Assigned Roles for User)，調整系統所需要的使用者與角色關係。

IV. 按[∨]鍵後即開始執行。

B.2.5. 角色功能管理(MF005):由 UserRoleFunc/RoleFunction 進入。

此功能僅做角色與功能的對應工作故無新增、刪除等功能，只有 Assigned 或 NotAssigned。

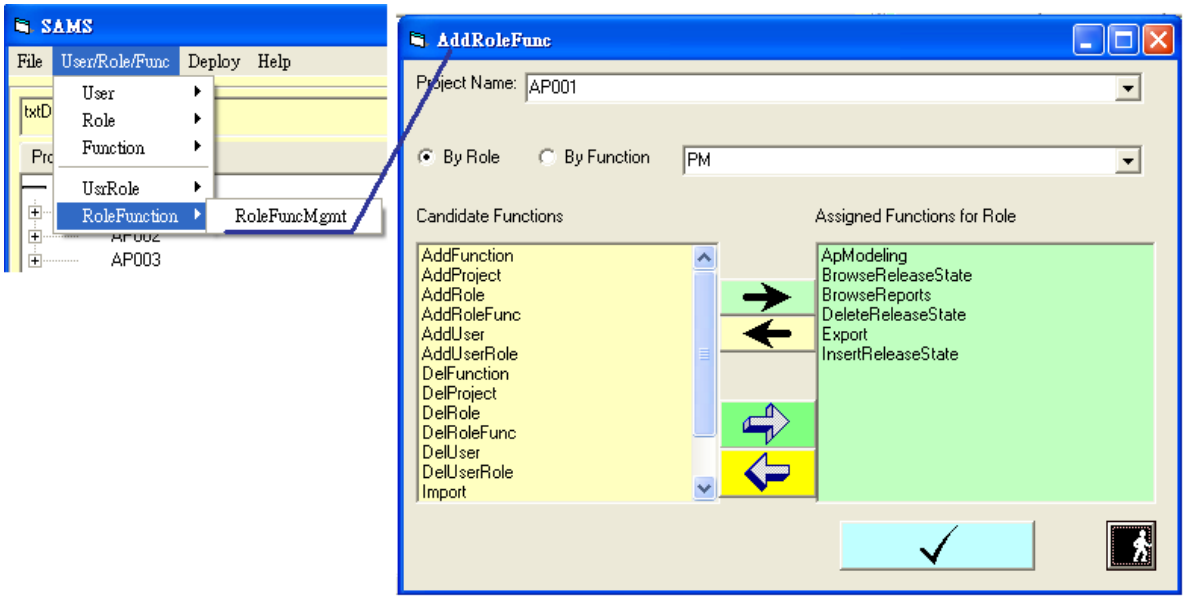

圖 23：角色功能管理: Role - Function

操作說明:

I. Project Name: 選定要設定的專案。

II. By Role 或 By Function: 決定要以角色或功能為根本。

i. ByRole: 設定單一角色擁有哪些功能

ii. ByFunction: 設定單一功能指定給哪些角色擁有

III. 雙擊下方兩個方框內容(Candidate Function 或 Assigned Function for Role)，調整系統所需要的角色與功能關係。

IV. 按[OK]鍵後即開始執行。

B.2.6.電腦主機管理（MF006）：由 Deploy/ApModeling 進入。

此功能會帶出所有 Server 資訊供 Add 時參考。

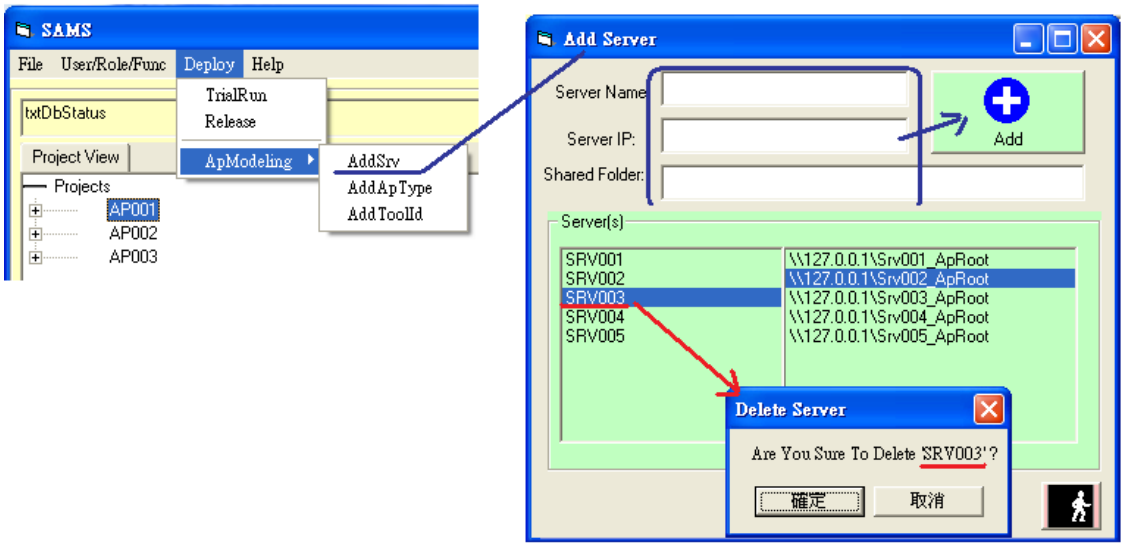

圖 24：電腦主機管理

操作說明:

I. ServerName: 輸入電腦主機名稱

II. ServerIP: 輸入電腦主機之網路 IP

III. SharedFolder: 輸入該電腦主機所分享出來的工作目錄

IV. 再按[Add]鍵即可新增一 Server。

V. 雙擊左下方之 ServerName 即跳出確認是否刪除的畫面，按確定後則可刪除一 Server。

B.2.7. 專案程式管理（MF007）：由 Deploy/ApModeling 進入。

此功能會帶出所有專案程式及其軟體維護人員之資訊供 Add 時參考。

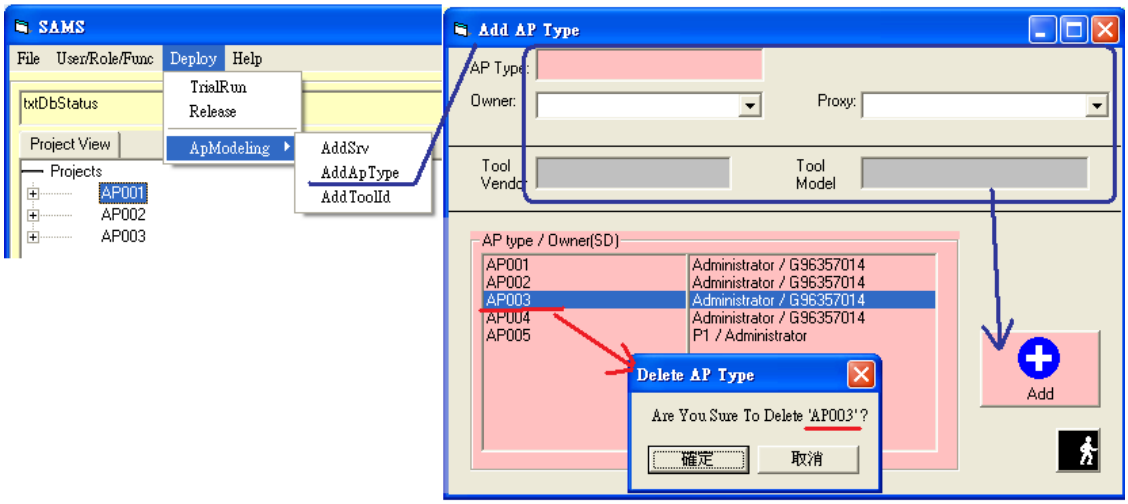

圖 25：專案程式管理

操作說明:

I. ApType: 輸入新專案程式名稱。

II. Owner: 選擇一軟體維護者負責此專案程式。

III. Proxy: 選擇一軟體維護者代理此專案程式。

IV. 最後按[Add]鍵即可新增一專案程式之基本資訊。

V. 雙擊左下方之 ApType 即跳出確認是否刪除的畫面，按確定後則可刪除一專案程式。

B.2.8. 專案程式執行管理（MF008）：由 Deploy/ApModeling 進入。

此功能一開始會帶出所有設備編號(專案程式執行的編號)資訊，包括要在哪一台主機上執行哪一個專案程式。自動佈署功能就是參考這些設定，將所選的檔案佈署到正確的目的地。

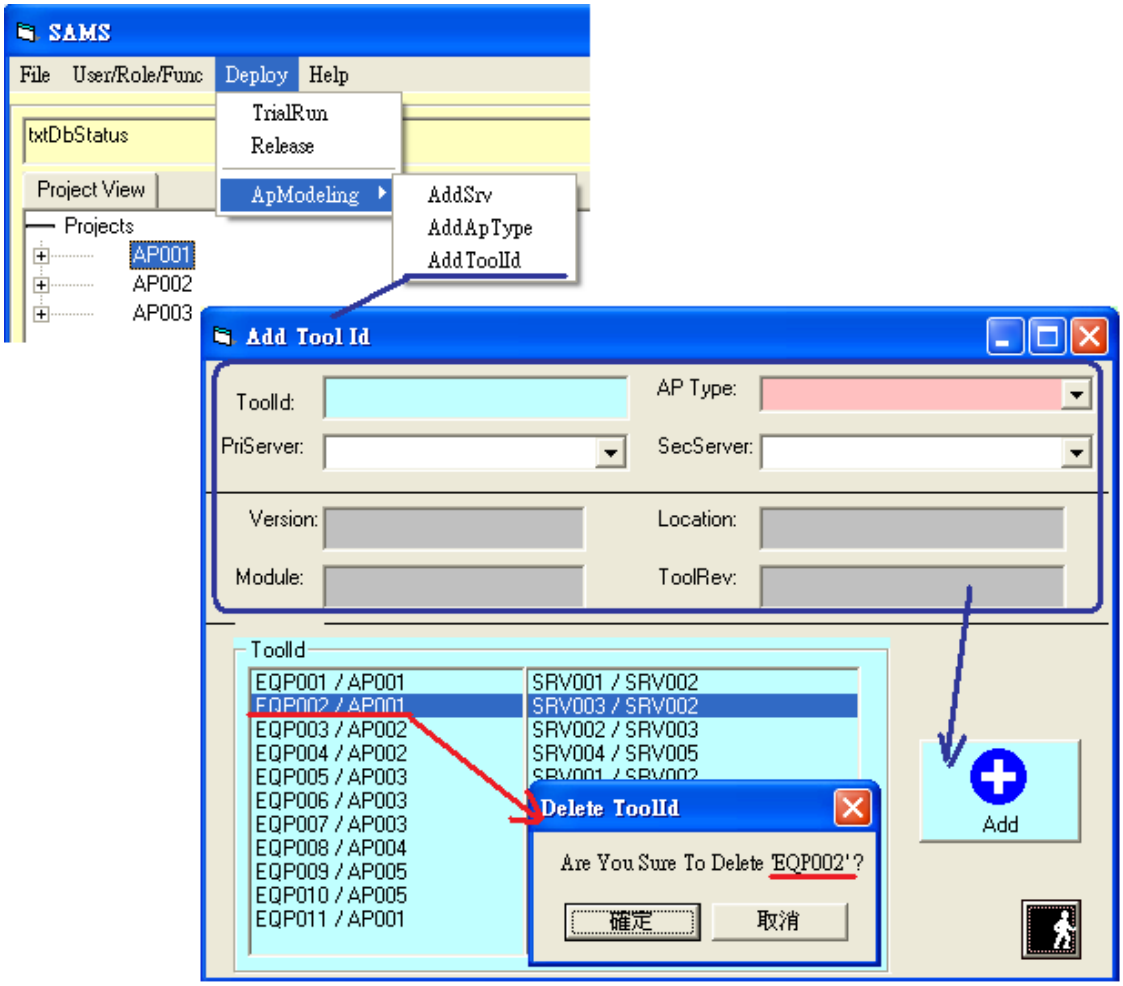

圖 26：專案程式執行管理

操作說明:

I. ToolId: 新增一設備編號(專案程式執行的編號)。

II. AP Type: 選擇一專案程式。

III. PriServer: 選擇一電腦主機作為主要執行主機

IV. SecServer: 選擇一電腦主機作為備源主機

V. 最後按[Add]鍵即可新增一專案程式之基本資訊。

VI. 雙擊左下方之 ToolId 即跳出確認是否刪除的畫面，按確定後則可刪除一設備編號(專案程式執行的編號)。

B.3.檔案功能

B.3.1.檔案匯入（FF001）：將實體檔案匯入資料庫中，並以專案的文件處理。

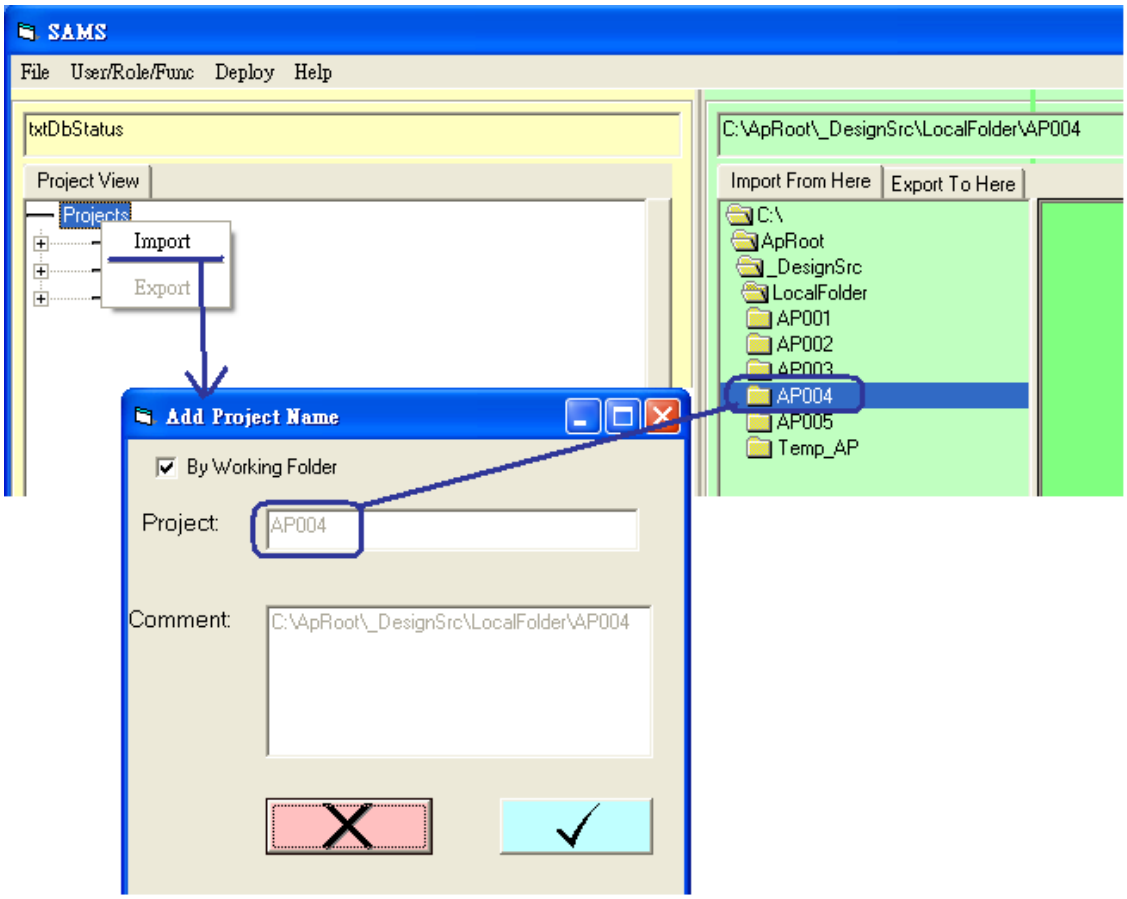

圖 27：檔案匯入管理畫面

操作說明:

I. 先在畫面右側選擇開發環境的專案目錄。

II. 再於畫面左側 Project(Root Node)上按右鍵，並選擇 Import。

III. 隨即跳出 Add Project 視窗，並以開發環境之專案目錄為預設專案名稱。

IV. 若想自訂專案名稱則將 By Working Folder 選項拿掉(不選)即可自行輸入專案名稱。

V. 最後按[∨]鍵即可匯入專案文件。

B.3.2. 檔案匯出（FF002）：將資料庫中的專案的文件匯出到實體檔案以供測試之執行。

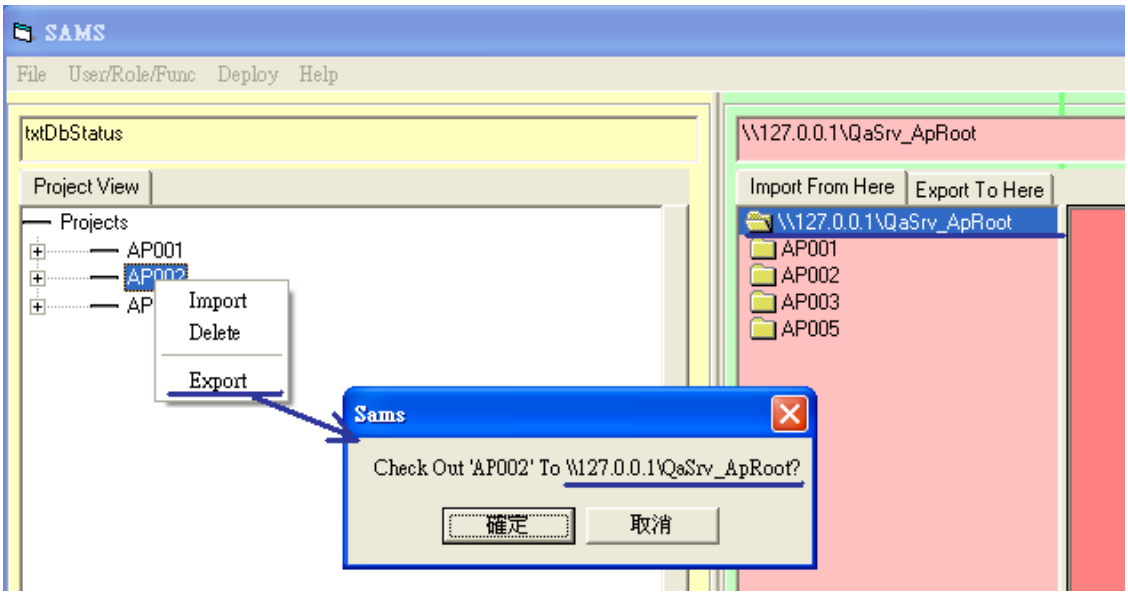

圖 28：檔案匯出管理畫面

操作說明:

I. 先在畫面左側選擇一專案，並按右鍵然後選擇 Export。

II. 此時跳出確認視窗。

III. 按[確定]則系統將資料庫中的檔案匯出到測試環境。

B.3.3. 檔案佈署到測試環境（FF003）：由 Deploy/TrialRun 進入。

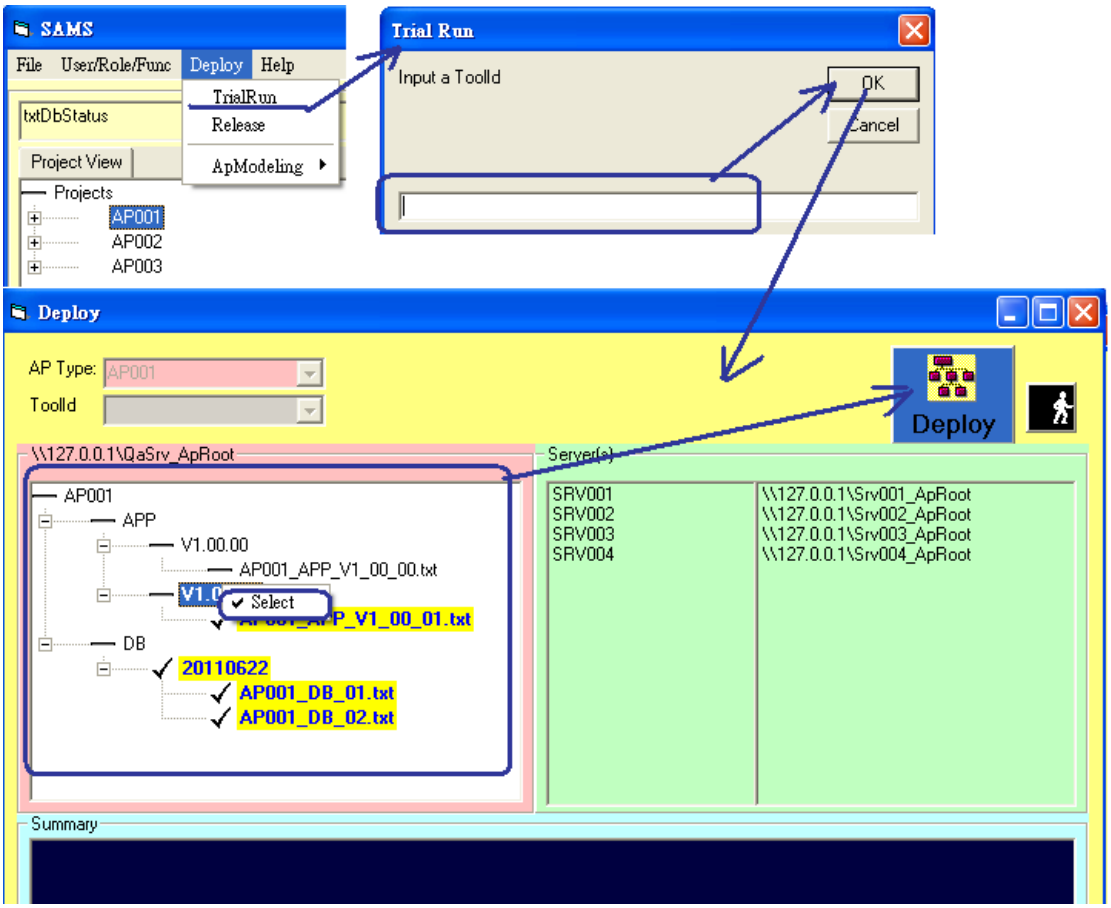

圖 29：檔案佈署到測試環境畫面

操作說明:

I. 輸入要 Trial Run 的 ToolId。

II. 預設將該 ToolId 帶入佈署畫面，Trial Run 只能佈署該 ToolId。

III. 在左側專案文件之目錄或檔案上按右鍵，選 Select，將所要佈署的檔案選取。

IV. 按[Deploy]鍵，系統則將所選擇的檔案佈署到相關的 Server 上。

B.3.4.檔案佈署到生產環境（FF004）：由 Deploy/Release 進入。

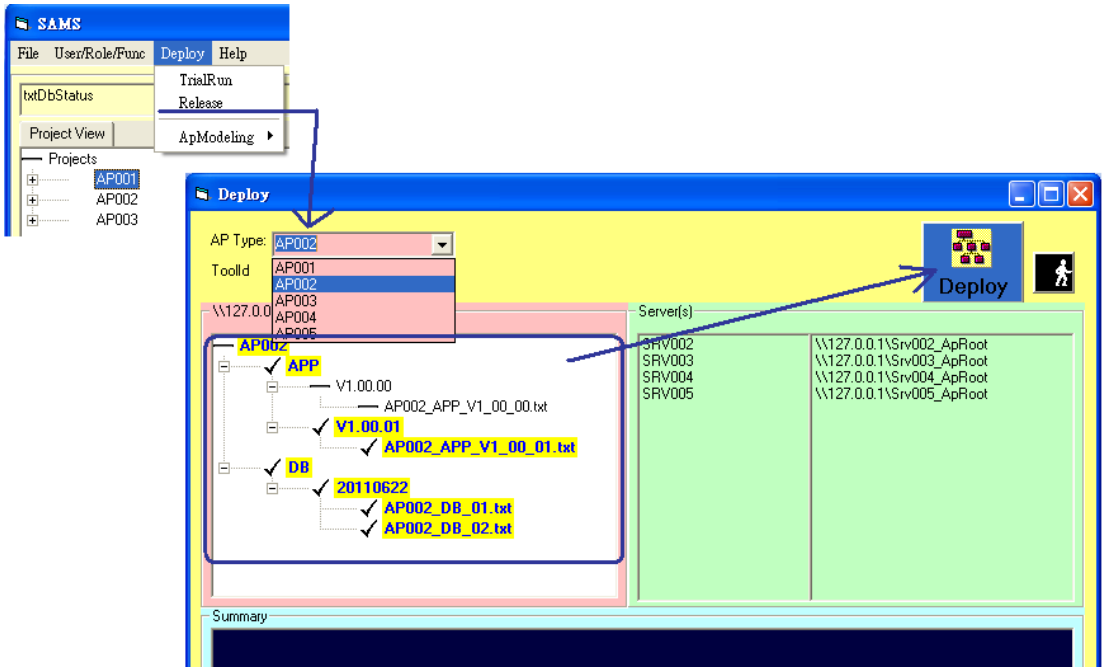

圖 30：檔案佈署到生產環境畫面

操作說明:

I. 選擇要佈署的專案程式。

II. 系統會自動更新左側之測試環境文件，及右側之相關 Server 資訊。

III. 在左側專案文件之目錄或檔案上按右鍵，選 Select，將所要佈署的檔案選取。

IV. 按[Deploy]鍵，系統則將所選擇的檔案佈署到相關的 Server 上。

附錄 C: 相關程式碼範例

C.1. 將實體檔案存入 MySQL BLOB 欄位的範例

```
5         Dim tmpsSql                 As String
6         Dim tmpRs                   As ADODB.Recordset
7         Dim tmplRtn                 As Long
8         Dim tmpoFileStream          As ADODB.Stream
18        If IsNewFileContent(subsFileId) Then
19            'AddNew record of FILECONTENT
20            Set tmpoFileStream = New ADODB.Stream
21            tmpoFileStream.Type = adTypeBinary
22            tmpsSql = "Select * From FILECONTENT Where 1=0"
23            Set tmpRs = go_DbOperatioin.ExeSQL(tmpsSql)
24            'TODO: Open the file and store in a Byte Array (tmpaFileContent)
27            tmpoFileStream.Open
28            If Len(subsFileName) > 0 Then
29              tmpoFileStream.LoadFromFile (subsPathName & subsFileName)
30            Else
31                tmpoFileStream.LoadFromFile (gsEmptyFilePath)
32            End If
33            'If tmpoFileStream.EOS = False Then
34            With tmpRs
35                .AddNew
36                .Fields("FILEID") = subsFileId
37                .Fields("SIZE") = subsSize
38                .Fields("UPDATETIME") = subsTime
39                .Fields("FILECONTENT") = tmpoFileStream.Read
40                .Update
41                .Close
42            End With
44            tmpoFileStream.Close
47        End If
```

C.2. 將資料從 MySQL 的 BLOB 欄位取出，並存成實體檔案的範例

```
17          tmpsSql = "Select * From FILECONTENT Where FILEID='" & subsSrcFileKey & "'" & _
                      " And SIZE=" & sublSrcFileSize
19          Set tmpRs = go_DbOperatioin.ExeSQL(tmpsSql)
20          'TODO: Open the file and store in a Byte Array (tmpaFileContent)
21          SaveFile tmpRs.Fields("FILECONTENT").GetChunk(sublSrcFileSize), subsDestination
```

```
Public Function SaveFile(ByVal subaFileContent As Variant, subsFilePath As String)
As Long
1           Const PROC_NAME As String = MOD_NAME & ":SaveFile"
2           On Error GoTo ErrHandle
3           Dim tmpErrCode              As Long
4           Dim tmpErrMsg               As String
5           Dim tmplRtn                 As Long
6           Dim tmpaFileContent()       As Byte
7           Dim tmpoStream              As ADODB.Stream
9           Set tmpoStream = New ADODB.Stream
10          tmpoStream.Type = adTypeBinary
12          'TODO: Open the file and store in a Byte Array (tmpaFileContent)
13          tmpaFileContent = subaFileContent
14          With tmpoStream
15              .Open
16              .Write tmpaFileContent ' tmpRs.Fields("FILECONTENT")
17              .SaveToFile subsFilePath, adSaveCreateOverWrite
18              .Close
19          End With
```