舊有資訊系統之更新實務探討

林靖真
銘傳大學資訊管理學系
Jingjen.lin@gmail.com

林至中
銘傳大學資訊管理學系
jlin@mail.mcu.edu.tw

摘要

當前的軟體專案開發實務中已有許多案例是與舊有資訊系統有關聯或轉換，但專案團隊往往只專注在與使用者溝通所得來的需求分析結果，而忽略也需求從既有舊系統的本身得到知識。然而目前在教科書上與相關研究中缺乏在新系統開發時融入既有舊系統分析所適用的軟體開發流程。因此我們從現有的軟體維護流程、軟體再生工程與新系統開發流程中各截取其優點與適用之處，發展成一套能夠快速解析出既有舊系統規格並且可以融入新系統開發的『V 字流程模型』。針對既有舊系統做部份必要的軟體再工程分析，把所得到的分析結果拿來協助新系統的規劃與開發，最後也從研究個案中發現 V 字流程模型確實可以避免邊做邊改與提高軟體正確性的結果，同時也能滿足大多數企業主想要節省成本與提升開發效率的要求。

關鍵詞：既有舊系統、軟體再生工程、逆向工程、軟體流程模型

1. 前言

1.1 背景與動機

在這資訊技術突飛猛進的年代，各產業 e 化的腳步也隨之快速發展。許多在早些年所開發出來的系統經過許多的變因導致不適用在今天的組織內，這些舊系統被冠上「既有舊系統」 (Legacy System)的名稱[24]。當前的軟體專案開發實務中已有許多案例是與舊有資訊系統有關聯或轉換，例如企業的合併組織結構需要重整，或是商業流程改變等，而與這些流程產生相對應的系統都會受到影響。也因為很多系統上的流程或元件已經存在原有舊系統中，企業主往往會以已經開發過的理由來要求縮短時間與減少成本，這一來就會造成資訊單位有不少的困擾。導致困難的因素如: 沒有保留完整的相關文件甚至沒有任何文件說明、組織歷經多年的改朝換代當初開發的人員已不在崗位上、在原有舊系統開發的階段是分別由不同的單位來開發甚至是委外開發或是購買套裝軟體所組成，這些種種的原因都構成企業處理既有舊系統的阻礙。

然而目前教科書上與相關研究中甚少提及要在新系統開發時融入既有舊系統分析所適用的軟體開發流程。而從軟體維護(Software Maintenance)與軟體再生工程(Software Re-engineering)的觀點來處理既有舊系統問題也不是很適切，這兩者對系統的變更都有其限制。但在現今的環境中，新技術不斷注入與商業流程持續更新，讓企業中的既有舊系統大多已經是在不合時宜亦或無力再負擔龐大的維護成本的狀況之下才會提出改善的需求。再加上使用者在上一個階段系統的使用過程中，會更清楚自己的需求，進而提出更契合的新需求與流程，這也是在實務上經常遇到的狀況。因此，提供適當的開發流程是有其必要性。

1.2 文獻回顧

1.2.1 軟體演進

無論任何的應用領域、大小或複雜性，電腦軟體都會隨著時代而演進。很多的狀況的發生會牽動著應用程式必須再造或者是改變以保持著隨時為企業提供最大的利益，而這些再造或是變更的過程就是舊系

統的演進。過去有很多的研究已經指出，系統一旦起用，變更的需求是無法避免的。而最常被提到的軟體變更策略有[22][25]:

1.軟體維護(Software Maintenance):軟體為因應需求而變更，但系統的基本結構維持不變。

2. 架構轉換(Architectural Transformation):通常系統從集中式的資料中心架構演變成網路(Client Server)架構。

3. 軟體再生工程(Software Re-engineering):系統實施軟體再生工程時，不增加任何新功能。修改系統好操作且便於往後的變更。

1.2.2 既有舊系統

既有舊系統(Legacy System)是一種社會技術電腦化的系統，包括軟體、硬體、資料和商業流程。既有的系統可能是文件、表單或檔案等人工系統，也有可能已經電腦化或是人工與電腦化參半。無論是何種狀況，大部份既有系統已符合使用者的需求，它勢必經過一段時間的發展去迎合企業需要，且讓使用者熟悉它的操作。也因為這一路的發展下來，系統中的某些部份也不會再符合企業的需要，且無法處理某些事務。會讓這些既有舊系統不適用在今天的組織內的內外在因素有很多如:國內及國際的經濟走勢、市場改變、法律修改、管理策略改變、結構重整都影響著企業不斷再改變。這些改變產生了軟體更新或是修改的需求[24]。

承如以上所言,那些不同型態的經歷使得軟體在使用幾年之後所需的維護或變更成本越來越龐大，承擔的風險也越來越高。而因為具有連續性的經驗已經流失，導致很難獲取系統的原始需求[23]。Rayson 等人所提的連續性經驗指的就是資訊系統在整個生命周期中的所有過程。另外，在學者 Liu 等的研究中也指出，從產業經驗來看，即使原始系統有留下文件，也不應該假設那些文件是有效文件，因為大部份的文件應該已經無法反應出目前系統的現況[19]。企業一旦要開始對既有系統做變更，綜合過去許多學者的研究把會面對到的問題整理出來[3][19][24]，如下:

1. 既有系統很少有完整文件被保留下來，就算有保留部份文件也不可能記錄下既有系統所有曾經變更的詳細過程。而往往最常發生的狀況是，唯一所遺留下的文件就是原始碼。

2. 既有系統的部份子系統或是部份切割過的功能當初是由不同的團隊完成。換句話說，整套系統很可能沒有一致性的規範。

3. 既有系統整套或部份的發展可能是由過時的程式語言。在有限制的條件下，有可能不容易找到懂這些語言的人才，而且還有可能必須向國外廠商採購昂貴系統零件才能做更動的行為。

4. 既有系統經歷過多年的系統維護多少會破壞原有結構，漸漸會變得不容易理解。再加上很多新的程式被加入，並且很可能會另外使用特別的方法和系統其他部份作連結的狀況會產生。

5. 既有系統所處理的資料來源可能被儲存在不同的檔案中，而這些檔案可能會出現不相容的情況。資料本身及其副本可能已經過時、不正確或是不完整。

企業在此時可能就會面臨到進退兩難的局面。若是繼續增加既有舊系統的生命，也是繼續增長維護的成本並承擔系統結構越來越複雜和能承接系統的人才日益減少的困窘。若決定替換新系統，企業也是要另外支付開發成本，而所承擔的風險換成是無法得知新系統是否能帶來實際商業效益的不確定性。

1.2.3 軟體演進之方法論

在 1.2.1 中提到學者 Warren 對軟體的變更提出三種方式，而另外也有學者 Bisbal 對此提出更進一步的分類。學者 Bisbal 把軟體演化匯整成四個運作的活動如圖 1 所示，1.包裝(Wrapping) 2.維護(Maintenance) 3.遷移(Migration) 4.再開發(Redevelopment) [5]。越往右邊對既有舊系統的影響越大所需要更動的數量也越多。由於在這部份的領域中，相關分類名詞的定義上並沒有明

確統一的規範，而有些研究會將包裝法(Wrapping)與遷移(Migration)視為相同[7][17]。

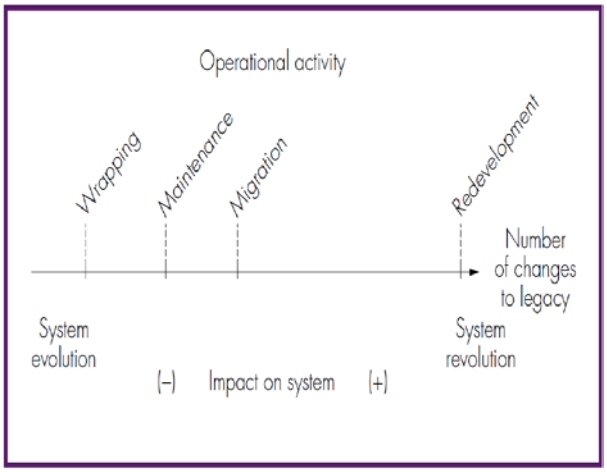

圖1 Bisbal的軟體演化活動[10]

系統維護其主旨為,軟體為因應需求而變更，但系統的基本結構維持不變。而其主要的工作內容就是在既有的系統上所發生的事件，且分成三個主要的類型。1. 改正(Corrective) 2. 調整(Adaptive) 3. 擴充(Perfective)。目前針對軟體維護領域，許多學者都提出適用於各種環境下的流程模型，我們以 Sommerville 所提的模型[24]，Sommerville 把軟體維護流程中所需的活動內容歸納成以下幾個項目:需求變更(Change Requests)、分析衝擊 (Impact Analysis)、計劃釋出 (Release Planning)、實作變更(Change Implementation)與最後的系統版本釋出(System Release)。

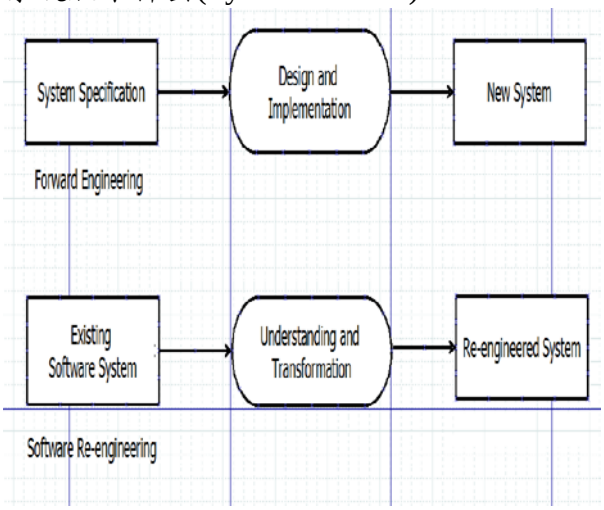

圖2 Chikofsky與Cross的軟體工程流程模型

軟體再工程(Software Re-engineering)是將既有舊系統重新再新製作，它能夠增加既有舊系統的可維護性。再生工程包含系統文件重整、系統重組與結構重建、將系統轉換成核心的程式語言、修正與更新系統資料的結構與價值，軟體功能與系統架構則不變動。學者 Chikofsky 與 Cross 將傳統的開發方式稱為正向工程(Forward Engineering)，以便與軟體再生工程有所區別，如圖 2 所示。軟體再生工程和開發新軟體最重要的差別就在於開發的起始點。正向工程的起始點是系統規格制定，然後是設計並實作系統；系統再生工程的起始點則為現有的系統，而整個流程則以瞭解並轉換原始系統為基礎[8]。

軟體再生工程的流程就如同先前講述的軟體維護流程一樣，在歷經多年的發展許多研究人員都提出適用在不同環境下的流程模型。在此以學者 Sommerville 所展示的軟體再生流程為例。Sommerville 敘述可能的系統再生流程，而軟體再生工程包含了下列幾項活動[24]:

1.轉換原始程式碼: 從舊的程式語言轉換程新版本的相同或不同的程式語言。

2.反向工程: 分析程式與粹取資訊，以便撰寫系統的組織與功能文件。

3.改善程式結構: 分析並修正程式的控制結構，使其易於閱讀且易懂。

4.程式標準化: 將程式相關的部份群組在一起並且適當的移除多餘部份。

5.資料再生: 程式所處理的資料也必須經過改變，以便對應程式的變更。

而綜合過去學者所定義再生工程的內容為: 1.改善人對軟體的瞭解。2. 準備或提昇軟體本身通常可以提高其維護性(Maintainability)、重用性(Reusability)與演進能力(Evolvability)。從技術層面看來，軟體再生似乎是系統演化的解決方式中次等的選擇。由於它沒有更新軟體的架構，因此很可能將集中式系統的功能分散出去。若要徹底改變系統使用的程式語言也是不太可能的，例如將舊系統轉換為物件導向程式語言 Java 或 C++。另外，因為系統功能也沒有改變，所以系統既有限制仍然存在[24]。

逆向工程(Reverse Engineering)的進行對系統瞭解的過程只是再生過程之一。換句話說，逆向工程只是一個分析軟體的流程。其目的是從原始程式碼導出系統設計與規格，進而讓我們可以從原始的程式碼中擷取出原本設計與實現的資訊。

學者在研究中的提出逆向工程的流程，從兩個不同的來源來使用資訊，第一個是由測量模組所組成的常規方法，第二個是重新探討設計的觀點。建議手動的偵查出作業流程與功能的需求，並且要在做切割以前就把這些因素納入考慮。學者 Lu 等人就把逆向工程中需要恢復並記錄的資訊做以下五點的整理[20]:

1.架構(Structure): 高層次的系統架構。

2.功能(Function): 什麼樣的程式負責什麼樣的運作。

3.行為(Behavior): 瞭解系統在執行時期的活動。

4.原理(Rationale): 每個設計步驟之間彼此的牽連，並推理其背後設計的原裡。

5.建構(Construction): 模組化、文件化與測試程序...等等。

逆向工程是由知識摘錄的運作(Knowledge Extraction Operation)所驅動，其背後被後所運用一連串的技術是在對相關的對象做知識的探索。但也有學者指出逆向工程的流程無法完全的自動化，因為電腦善長於資料的分析與過濾，而人類善長於組織與分類，機器在這方面還無法取代人類的工作[9][10]。

1.2.4 軟體流程模型

在軟體工程裡,系統漸進的演化從一開始的計劃到廢止的過程，整個軟體生命週期中所有的工作項目都涵蓋在其中且具有意義[13]。而規範流程模型定義一套明確的活動、行為、里程碑，以要求資訊人員設計高品質軟體的工作產品。這些工作流程模型並不是完美的，但它們對軟體工程的工作提供有用的指標[22]。軟體的流程模型在近半世紀的發展下來，各家學者因應環境的變遷與軟硬體技術革新等因素提出很多適用在個領域之下的軟體流程模型。而一般的流程框架(Generic Process Framework)都會包含溝通(Communication)、計劃(Planning)、塑模(Modeling)、建構(Construction)和部署(Deployment)。

1970 年代由學者 Royce 所提出瀑布模型(Waterfall Model)，它是線性進行的框架活動。循序式的瀑布模型在使用上重視文件密集、速度緩慢且非常昂貴[6]。它與目前實務上的狀況不太一致，例如，連續的改變、演進的系統、緊湊的時間限制。快速應用開發(Rapid Application Development, RAD)是一種涉及類似反復式開發與雛型模型技術的程序設計方法學。採用數種結構化分析與設計技術，特別是使用者驅動(User-Driven)的資訊工程相關技術與原型製作技術來加速軟體系統開發的整合技術[15]。另外，除了早期的流程模線性流程之外，軟體的流程模型還可區分成另一種多循環流程(Iterative Process Models)。學者 Basili 認為系統開發應該以逐步漸進的方式來進行，先提出重要卻也許單純的議題發表最初的版本，再逐步做細化的修正之後衍生發行至最終的版本[4]。現今的軟體開始分成若干版本就是一個很好的應用實例。漸增式開發(Incremental Development)由學者 Basili 所提出，其特點是將軟體規劃成若干個段落，再逐次的進行發展，藉由先前的產出來修正後面的設計[18]。支援漸漸式開發流程模型有:雛型模式(Prototyping Model)、演進式雛型法(Evolutionary Prototyping)、螺旋模型(Spiral Model)等。這些方法雖大多可以克服時間限制問題，也有可以讓錯誤得以即時發現並修正的優點，但由於整個流程採反覆漸進的方式，管理者無法確知需要多少週期才能建構出成品，如此一來著實也加重了專案管理上的複雜度。同時也因為部份模型在發展的過程中並不強調文件的產出，這也造成了日後軟體在演進的後期維護上的困難[1][18][21][22]。

近期的軟體流程模型與過去最大的不同就是以二維的概念發展流程模型，以統一流程(Rational Unified Process, RUP)為

例，縱向由上往下展開的是軟體流程的循環，而橫向水平呈現則是各軟體的階段，用以表示專案的進展。RUP 模式可由動態與靜態兩個層面來闡述系統開發時的執行階段與核心工作[12][16]。Jacobson 等人把 URP 的動態面依序分成四個主要階段: 初始(Inception Phase)、詳述(Elaboration Phase)、建構(Construction Phase)與轉移(Transition Phase)，而靜態結構主要處理依邏輯順序將軟體開發與管理支援工作描述為九個核心工作流程: 商業建模（Business Modeling）、需求分析（Requirements）、分析與設計（Analysis & Design）、 實作（Implementation）、測試（Test）、部署（Deployment）、 組態與變更管理（Configuration & Change Management）、專案管理（Project Management）與環境配置（Environment）[14]。其中前三項是軟體工程工作，後三項是管理支援工作[16]。

1.2.5 軟體開發架構

學者 Gomaa 在其研究中提到，在軟體工程領域之中，以架構與元件導向為方法的開發軟體是未來發展的趨勢。目前的軟體開發方法，都會強調其採用軟體架構為中心的思維，即為 ACSD(Architecture-Center Software Development)。ACSD 的基本思維，認定預先制定的可重複使用的元件和軟體架構，會比直接拿某程式語言或使用類似 J2EE 這樣的技術平台直接來塑模(Modeling)來的更加功能強大，因為架構會重複使用元件 [11] 。而引入可重用的架構、抽象類別和避免程式碼冗餘的元件，並不是一種新的想法。軟體架構是一個較高層次的、概念化的軟體設計方法，以元件的角度來描述整個應用系統的架構及其相互關連性。工作分層的觀念 MVC (Model-View-Controller) 架構源自於 Smalltalk 程式語言。MVC 架構是 Trygve Reenskaug 於 1970 年代所提出，當初是為了「縮小人類精神面 Mental Model 與數位系統 Digital Model 之間的鴻溝」這個議題，企圖透過 MVC 功能性的分層歸類，而達到一個較好的系統設計與互動模型。模型（Model）、檢視（View）和控制器（Controller），分別做資料模型的封裝、使用者介面的處理和邏輯控制的分離工作。而過去研究也指出適當地使用 MVC 樣式可以讓應用系統外觀和邏輯的程式碼分開，讓系統更容易維護。採用 MVC 樣式開發應用系統已成為廣大資訊開發的共識[2]。

1.3 研究目的

為了協助企業處理既有舊系統時所面臨的困難，本研究提出適用在中小型專案下的軟體開發輔助機制。從現有的軟體維護流程、軟體再生工程與新系統開發流程中各截取其優點與適用之處，發展成一套能夠快速解析出既有舊系統規格並且導入新系統發展的 V 字流程模型。我們把此流程模型規劃成七個步驟，分別是 1.舊系統環境確認 2.舊有功能清單匯整 3.舊有功能逆向工程 4.規格輔助文件 5.新系統規格 6.新舊規格的評估 7.整合與測試。針對既有系統做部份必要的軟體再工程分析，產出必要之文件，之後把所得到的分析結果拿來協助新系的規劃，特別是把導入的步驟獨立成三個開發流程分別是 1.全新需求 2.部份延用 3.完全延用三個部份。幫助資訊人員做更精確的分類，同時協助資訊人員達到提高需求評估的正確性並且能夠管控資訊系統的複雜度，進而達成有效管理專案時程的結果。最後本研究將此 V 字流程模型，應用在某大學的校務學籍系統開發專案上，以驗證其有效性。

2.V字流程模型

本研究提出融入既有舊系統開發流程模型，如圖 3。在左邊我們描述符合需要的軟體再工程(Software Re-engineering)，在右邊所描述的是融入舊有規格之後的新系統開發流程。

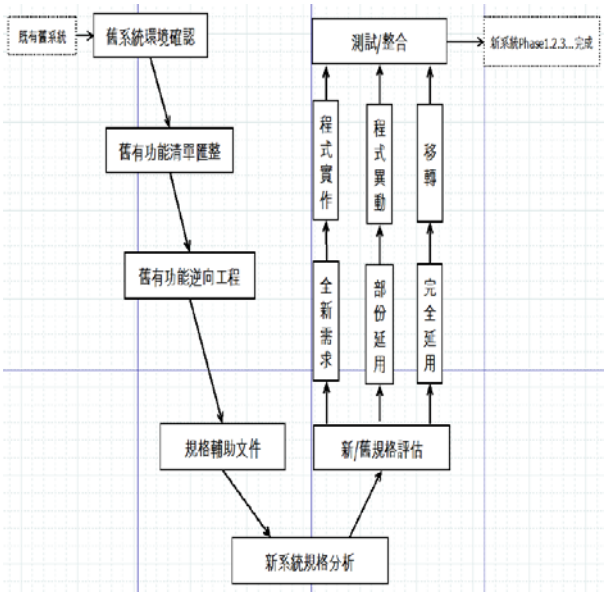

圖3　V字模型-軟體再工程

2.1 V 字模型-軟體再工程

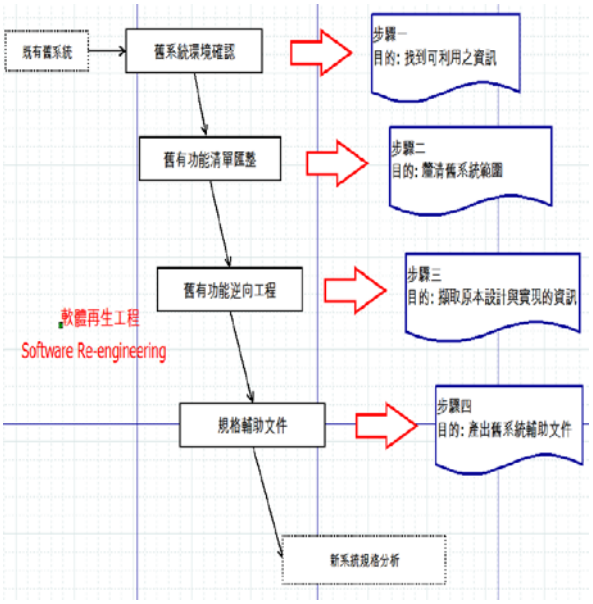

圖4　V字模型-軟體再工程

在進入模型的一開始先把整個流程劃分成兩大區塊。第一部份為軟體再工程，如圖 4。第二部份為新系統開發流程，如圖 7。

(一) 步驟一、舊系統環境確認

搜集環境資訊的目的在於找到還可利用的既有資訊幫助專案管理。應確認的範圍如下:

1. 是否有文件? 這些文件是否完整、一致且最新?

一般而言從系統最初期的規劃開始到分析與設計，接著到系統的開發與測試至最後的系統佈署，這些過程都會產生相對應的文件。

2. 資料儲存的型態為何? 是否有明確的資料模型?

以商業應用系統而言資料的儲存方式通常都會選擇目前市面上個大廠牌所提供的資料庫。其他也有以檔案的方式來儲存資料。也因為可能需要處理的來源會很多，所以是否還保有資料設計文件或相關說明就顯得相當重要。

3. 所使用的程式語言為何?

目前應用系統開發常用語言有 Java、C++與.Net 等等。由於語言很多，先進行了解的目的是若有些較早期所使用的語言，開發團隊中也必須先找到這樣的人才，這也是幫助專案管理的重要項目之一。

4. 是否有使用組態管理系統來管理? 目前所使用的元件是否有詳細且明確的版本描述?

系統在長時間的使用之後所累積的變更也是很可觀，所以此工作的目的就是要找到這些需求變更的歷程記錄。

(二) 步驟二、舊有功能清單匯整

我們以功能需求為單位並利用使用案例(Use Case)來做功能的劃分如圖 5。此步驟主要的目的是要列出既有舊系統中的軟體功能清單。在這份清單裡應列有每個功能或元件的資料，建議的項目如下:

1. 重要性: 確認出主流程。每個系統都有其發展的主軸，例如，校務系統中的學生學籍系統，其主要的流程就是學生在學過程中所有與學籍相關的異動申請和記錄，但要組成這樣一個過程還必須要有其他基本檔的建立。所以我們可以拿這樣概念來定義出主流程、副流程與基本檔等重要性區分。

2. 使用目的: 主要執行的工作內容。以學籍系統為例，學生申請學籍異動的項目

就可能會包括休學、復學與轉系所等等。

3. 使用的對象: 找出每個功能的使用對象。除了管理者的使用功能之外是否還有其他使用者會參與或者是同樣的目的但因為使用者的不同在功能上有些許的差異，這些資訊都可以在這個階段盡早釐清。

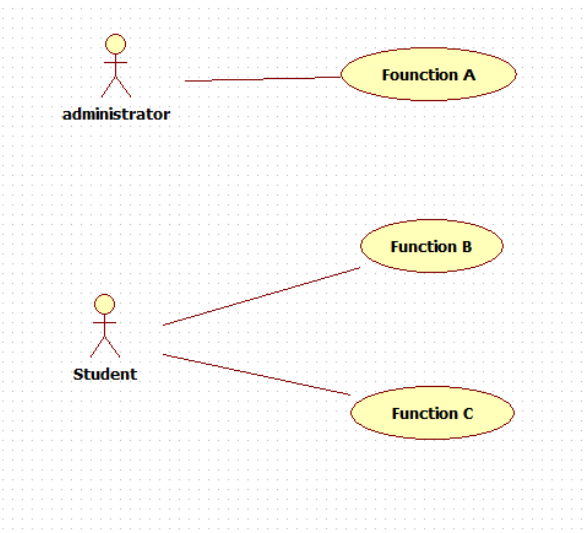

圖5　使用Use-Case匯整功能清單

列出這三個項目的內容還有另一個重要目的，每個系統在進入開發階段之前，專案管理人員都必須要釐清每個功能之間的關聯為何，這也牽涉到是否有需要先找出開發順序的問題。因此除了能瞭解功能在需求面上的來龍去脈之外，此步驟也背負著協助日後專案管理的重大責任。

(三) 步驟三、舊有功能逆向工程

逆向工程的主旨是從原始程式碼導出系統設計與規格。我們可以從原始的程式碼中擷取出原本設計與實現的資訊。以上一個階段中得到的使用案例為基礎，進行逆向工程導出原始的設計。為了可以更精確的描述原始功能的輪廓，我們提出以使用者介面(Interface)、流程(Process)與長久儲存單元(Persistent)來規範功能還原的過程。如圖 6 所示。

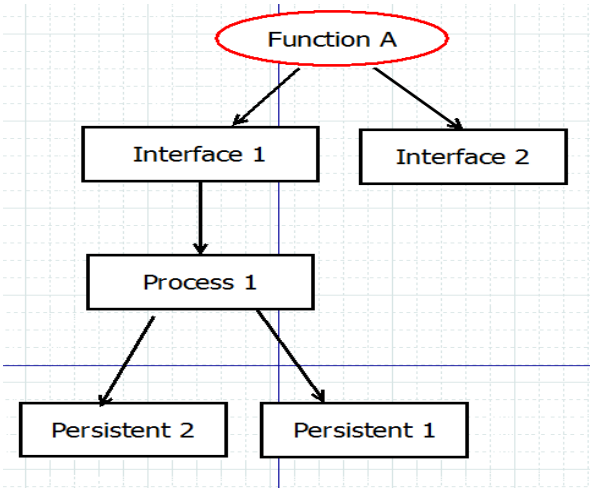

圖6　逆向工程之功能分解

1. 使用者介面(Interface): 系統中屬於該功能的所有的介面。例如學籍系統中學生基本資料修改功能，有一個學生基本資料呈現頁面。

2. 流程(Process): 系統中屬於該功能的所有的處理流程。例如學籍系統中學生基本資料修改功能，就會有一個學生基本資料查詢流程、一個學生基本資料更新流程等等。

3. 長久儲存單元(Persistent): 系統中屬於該功能的所有的資料儲存單元。例如學籍系統中學生基本資料修維護功能，會有一個學生基本資料查詢、學生遞補與更新資料的功能要找出所有資料來源。

(四) 步驟四、規格輔助文件

從前三個步驟得到資料之後，就可以把資料一併匯整成一份完整的既有舊系統規格輔助文件。其內容可分成兩個部份，靜態的資料結構與動態的功能邏輯:

1. 資料結構:從長久儲存單元中得到資料的來源與結構，進而協助我們得到整體的資料模型結構。可以利用 ER-Model 或是類別圖(Class Diagram)來做呈現。

2. 功能邏輯:從步驟二中所得到的使用案例與在步驟三中得到的功能細部流程說明，可整理出一份完整的現有功能說明文件。在此可以使用活動圖(Activity Diagram)或狀態圖(State Machine Diagram)來輔助。

2.2 V 字模型-新系統開發流程

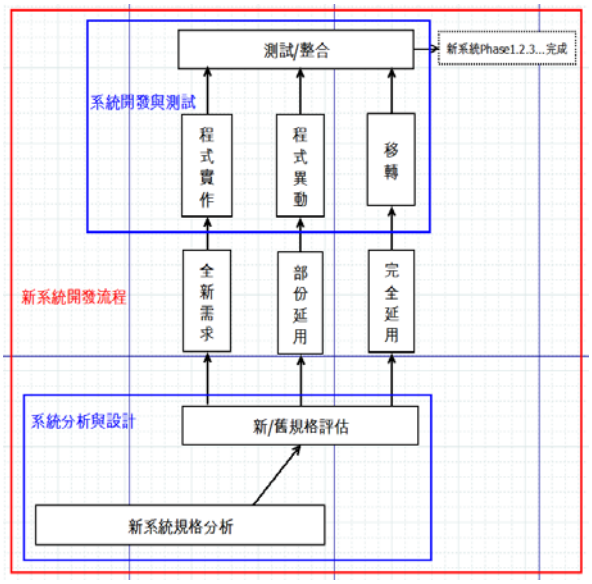

圖7　V字模型-新系統開發流程

(五) 步驟五、新系統規格

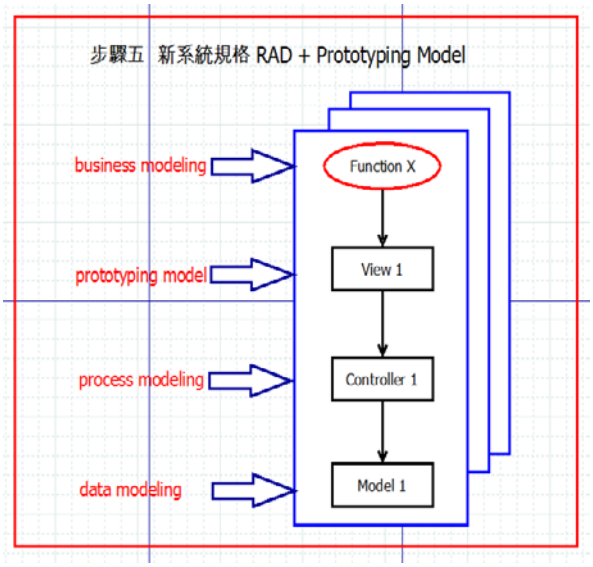

圖 8　V 字模型-新系統規格

針對使用者所提出來的新需求規格，我們也必須要做進一步的釐清。此時我們採用雛型模式(Prototyping Model)結合快速應用開發模式(RAD)來做新系統的開發。快速應用開發模式中的塑模(Modeling)包含三個主要的面向(Phases)-商業塑模(Business Modeling)、資料塑模(Data Modeling)與流程塑模(Process Modeling)[24]。我們以此為基礎，規劃出 Use Case 並利用雛型製作的技術來定義使用者的需求，以模組化(Modeling)的方式來分析系統的結構如圖8。這個階段會產出新系統中初步的系統分析與系統設計文件。而在此初期的規劃中已經會帶入與舊系統有關的分析，但還是著重在專案中的新功能的分析上。

(六) 步驟六、新舊規格的評估

當已經清楚知道既有舊系統的模樣與新系統中所需要的功能以後，接著就是要把兩者之間的相同之處與相異之處做一次整體的評比，找出可再利用的部份。這樣做的好處在於若能找到相同之處或者是可用之處，一來可以避免重複開發，二來也能節省時間提高整個系統開發的效率。而若是釐清功能上的差異之處，則也可以使用在新功能開發之時避免掉不必要的錯誤，同時也能驗證新功能的正確性。

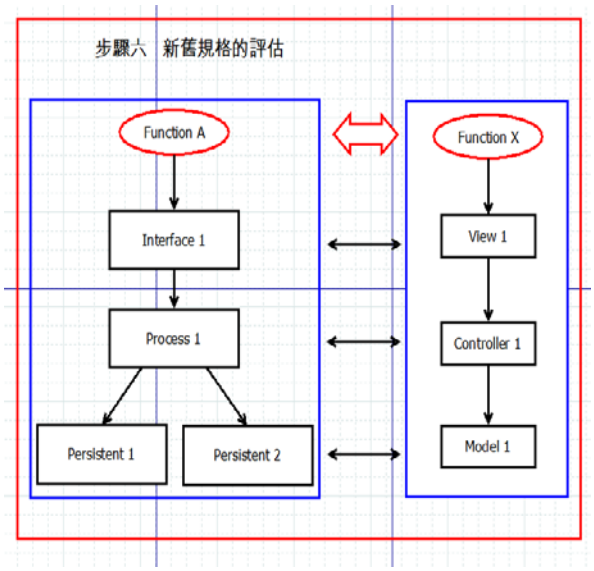

圖9　V字模型-新舊規格的評估

一旦有找出相似的需求功能，例如圖 9 中的舊系統中的 Function A 與 新系統中的 Function X，把規格輔助文件裡對該功能的 Interface、Process 和 Persistent 拿出來與新功能規格中的 View、Controller 和 Model 做比較。藉此得以決定是否可再利用。而做完上述的歸納之後，把系統開發分成三種流程: 1.全新需求 2. 部份延用 3. 完全延用。

這三種流程沒有一定的順序，而也可以是以反覆漸增的方式來發展。一般狀態之

下可把這三種流程視為平行進行的狀態，但專案管理人員可依專案的特性或是團隊人員的規劃等因素自行調整。

(六之一) 步驟六之一、全新需求

從步驟四產出的舊系統規格輔助文件中可以知悉，舊系統更新，可能會有舊系統不足之處，因此會產生全新的開發需求，需要全新的資料結構或/和功能業務邏輯。

(六之二) 步驟六之二、部份延用

由於時空背景的更替，過去可能沒有這樣的需求，在舊系統更新成新系統，使用者會多加一些資料結構或/和功能業務邏輯，因此，直接延用需求，會是較為省工，使用者也會較為容易接受的做法。譬如說，頁面欄位需要多加，或是業務邏輯多加一些判定。

(六之三) 步驟六之三、完全延用

舊系統之所以要轉成新系統，必定有其存在的價值，因此，直接參考與延用使用者需要的舊有系統的資料結構或/和功能業務邏輯，除了可以加快分析效率之外，也可以提升需求品質 。

(七) 步驟七、整合與測試

在軟體工程中的測試有很多種類,如白箱、黑箱測試與壓力測試等等。而這裡需要強調的一種測試,即迴歸測試(Regression Testing)。所謂的迴歸測試，是指確保程式在某些異動與變更發生之後，先前測試過的行為，仍能如預期地正常運作。而新舊系統的關聯與轉換，要從舊系統的使用經驗中，掌握新系統測試的全盤策略，確保舊系統延用的功能也能在新系統上正常運。其他如黑箱測試這種，可以瞭解新系統的功能是否滿足使用者需求的測試，也很重要，可視新系統的特殊性來選擇，在此就不在贅述。

3.個案研究

3.1 個案背景說明

此專案是由台灣某國立大學的資訊與通訊中心所委外開發的校務學籍系統。由於個案使用的系統是建構在舊式的Dbase3Plus 的資料庫之上，且各個單位所需的資訊系統早期又都是由各單位自行找配合的軟體廠商來開發，所以不僅是資料管理上沒有同步亦有很多半自動化的流程與檔案式的儲存內容，讓整個作業程序上顯得零亂又沒有效率。而在系統的部份也因為一開始沒有統一的管理也導致多年累積下來，不只是程式語言有多個版本，使用的平台也不同，這些種種的因素都讓後期接手的資訊單位陷入困境。因此前年底由資訊單位統籌規劃改善整個行政系統，但也因為整個工程浩大，資訊單位也採取降低風險的方式把資訊系統做領域上的切割再逐一的取代與改善。

此專案內容包括有以下幾項學籍資料活動之系統開發: 學生資本資料維護、學生學籍資料維護、學生事件申請與維護、學生資料匯入、教育部制式報表匯出。其中學生基本資料維護的功能已存在既有舊系統。而另外在學生的學籍資料、修業異動事件與學籍異動事件，這些相關功能還是屬於半自動化的作業流程，因此需要把這些項目轉換為資訊系統功能。

3.2 個案開發過程

依據前述的 V 字流程模型來應用於此校務學籍系統，由於受限於文章篇幅的限制，因此我們選用個案中的學生基本資料維護功能，做為整個應用的案例。以下為七個步驟的應用。

(一) 步驟一、舊系統環境確認

在專案開始之初與使用者進行訪談得到以下資訊。

1. 是否有文件? 這些文件是否完整、一致且最新?
   - ■ 沒有文件。
2. 資料儲存的型態為何? 是否有明確的資料模型?
   - ■ 資料是儲存型態有兩種。第一個學生基本資料是儲存在 Dbase3Plus

的資料庫內，第二種是由註冊組記錄與保存學生學籍異動資料在 Excel 內。而在 Dbase3Plus 所儲存的資料欄位有 HSTALL 與 BARN 等十份欄位說明的文件。

3. 所使用的程式語言為何?
   - 在由學生自行維護基本資料的網頁程式是由 ASP 來撰寫，執行環境為 IIS 伺服器。而由管理者所進行的維護則是一般的 SQL 語言。
4. 是否有使用組態管理系統來管理? 目前所使用的元件是否有詳細且明確的版本描述?
   - 組態管理上並沒有做任何管理，所有異動的狀況皆由 DBA 與使用者在溝通後，直接修改資料庫中所儲存的內容。

(二) 步驟二、舊有功能清單匯整

透過訪談得到既有舊系統中有四項功能。1.學生資料匯入 2. 學生基本資料維護 3. 教育部制式報表匯出 4. 學生學籍資料匯入。並且以使用者案例來做圖形化分析。

1. 重要性: 主流程。
2. 使用目的: 提供介面給學生更新基本資料。
3. 使用的對象: 某大學所有在學學生。

(三) 步驟三、舊有功能逆向工程

以學生基本資料維護為例。在既有舊系統中，過去的開發人員把所有的程程式都寫在同一支 ASP 的程式裡如圖 10。所以從 asp 中去找到該功能的程式邏輯與資料儲存的部份如圖 11。

圖10　舊系統中之使用者圖形介面

1. 使用者圖形介面(Interface): student.asp。
2. 流程(Process): 查詢與修改。
3. 長久儲存單元(Persistent):資料來源 HSTALL。

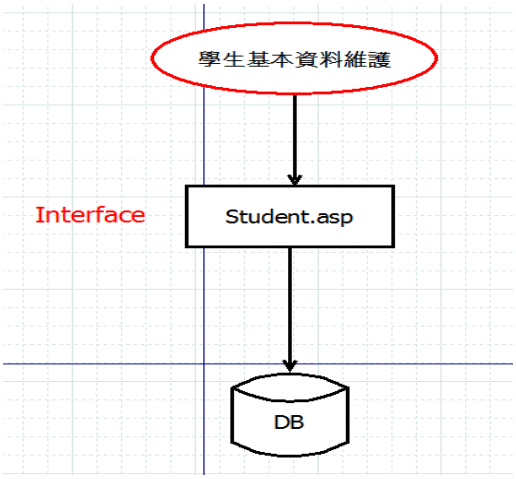

圖11 舊系統學生基本資料維護之功能拆解a

以每個需求為基礎，也就是每一個使用者案例。利用 V 字流程模型在第三步驟的規範再透過圖形化的表示方式，可以畫出圖 12。在這個過程中就能清楚的看到，原有的學生基本資料維護中存在的功能以及可能用到的資料結構為何。

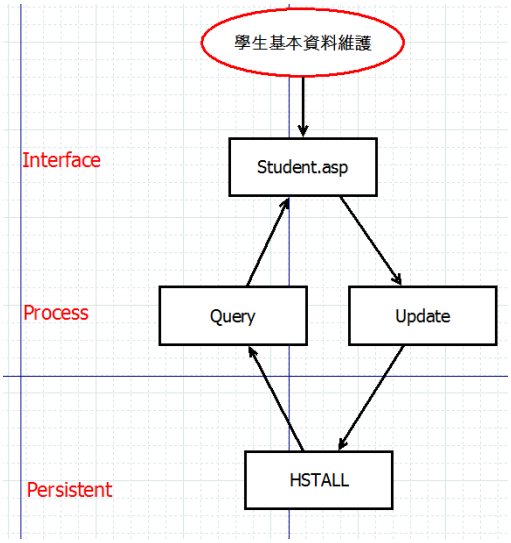

圖12 舊系統學生基本資料維護之功能拆解b

(四) 步驟四、規格輔助文件

主要得到的產出為兩個部份第一份為資料模型，第二份是功能邏輯描述。經過資料搜集與分析之後畫出舊系統的資料儲存方式或 ER-Model 如圖 13。而在搜集步驟

二與三的資料之後把功能邏輯的部份匯整成一份說明文件。

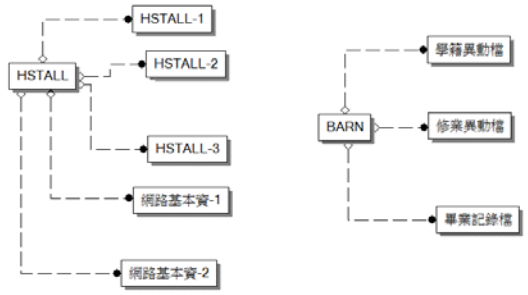

圖13　舊系統中之資料儲存狀態

(五) 步驟五、新系統規格

新的學籍系統規劃是由 Java 來開發，使用 Tomcat6 為前台執行伺服器，後端資料庫為 MS SQL Server2005。

1. 雛型法: 雛型製作的技術來定義使用者的需求
2. RAD-商業塑模(Business Modeling): 以功能為單位劃分出每個功能，並且以繪製出 Use Case 來做圖形化說明。
3. RAD-資料塑模(Data Modeling):把得來的資料匯整，以 ER-Model 的方式來做呈現如圖 14。

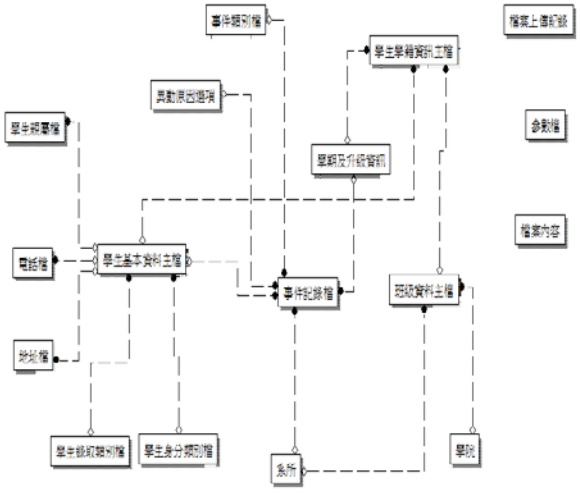

圖14　新系統之ER-Model

4. RAD-流程塑模(Process Modeling):以學生基本資料維護為例。如圖 15 所示。
   4.1 頁面(View): PhoneRecord.jsp、AddressRecord.jsp、StudentBasicInfo.jsp、StudentRelatives.jsp
   4.2 控制器(Controller): MainService.java
   4.3 模型(Model): AddressRecordDao.java、PhoneRecordDao.java、StudentRelativesDao.java、StudentBasicInfoDao.java

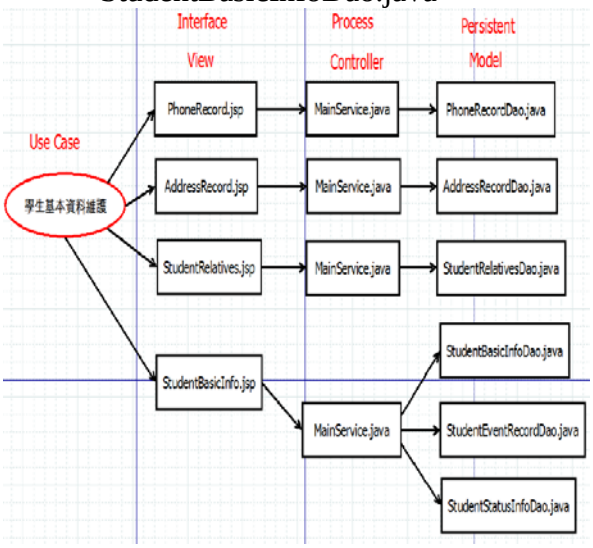

圖15　分析與設計學生基本資料維護功能

(六) 步驟六、新舊規格的評估

個案中因為使用的程式語言不同，所以就程式碼而言沒有可以延用的部份。但從需求面與資料面上經過新舊規格的評估有發現可以直接使用的部份。

(六之一) 步驟六之一、全新需求

首頁查詢、學籍維護、事件申請、事件維護、學籍升級、班級資料維護、身份別資料維護、錄取別資料維護、事件類別資料維護、電話資料維護、地址資料維護、親屬資料維護、事件原因資料維護、學期及升級、資料匯入、首頁 Excel 下、報表。

(六之二) 步驟六之二、部份延用

以學生基本資料維護為例，從圖 10 與圖 16 中可以發現，使用者所需要的顯示與維護的欄位大致相同，所以在頁面(View)上的設計上我們延用了個案上既有的特殊規範。

(六之三) 步驟六之三、完全延用

從分析結果拿到與既有舊系統中可以繼續延用的部份為身份別資料與錄取別資料可以直接使用，所以延用其格式再利用

轉檔的方式來做移轉資料。

(七) 步驟七、整合與測試

以學生基本資料維護為例，如圖 16 所示。採用迴歸測試的基本精神除了學生基本資料被查詢出來之外，另外還有學生遞補流程與學生電話、住址與親屬資料維護流程和到最後所有資料更新資料庫的功能。每個基本檔陸續被整合至主流程中間的每一次測試都要確保不能影響到上一個功能已經完成的行為。其中也包括了從舊系統裡所轉移來的身份別與錄取別，是否符合新系統之需要的測試與驗證。

圖16　新系統之學生基本資料維護頁面

3.3 個案開發結果

個案在經歷四個月開發過程後順利結案。從既有舊系統中所學習到的知識再搭配上與使用者密切溝通的快速應用開發模式(RAD)，讓此專案在過程中避開可能的設計錯誤，同時也滿足了使用單位想要快速開發並且節省成本的需求。

4. 結論

本研究所提出的 V 字流程模型，融合了既有舊系統所需要的軟體再生工程與新系統開發時兩者應該要有的比較分析和開發流程。我們也從個案研究中發現這樣的流程確實可以避免邊做邊改的錯誤方式，因為 V 字流程模型的前四個步驟特別強調既有舊系統的運作狀況及可能的問題，透過訪談與逆向工程相關的技術可以對既有系統的狀況進行瞭解，並且發現可能的問題，之後進入步驟五、六與七來對整個新系統做完整的設計與規劃。如此一來也能滿足大多數企業主想要節省成本與提升開發效率的要求。但這樣的方式也有一些限制，如專案團對中要能做到軟體再工程的分析與規劃，人員能力上的要求就相對的需要有一定的開發資歷。再者，本研究採用快速應用開發模型(RAD)的方式，專案的範圍會局限在以中小型為主的系統開發之上，所以若屬於大型系統專案，此流程步驟可能難以滿足需要。往後的研究還可以朝這些方向去做改善。

參考文獻

[1]吳仁和，*物件導向系統分析與設計：結合 MDA 與 UML*，三版，台北：智勝，2010。

[2]吳信輝、林佑德、廖俊鋒、李蔡彥，“應用 Web 框架工具建立通用資料庫系統 RDAA：以政治大學社會科學研究資料庫之整合為例”，國立政治大學電子計算中心，2005。

[3]趙善中、趙薇、尤柄文，*軟體工程*，台北：儒林圖書公司，2003。

[4]Basili, V.R. and Turner, A.J., “Iterative Enhancement: A Practical Technique for Software Development,” IEEE Transactions on Software Engineering, Vol. 1, pp.390-396, 1975

[5]Bisbal,J., Lawless, D., Wu, B., and Grimson, J. ,“Legacy Information Systems: Issues and Directions” Software, IEEE, Volume 16, Issue 5, pp.103-111. Sept.-Oct. 1999

[6]Boehm, B., “A View of 20th and 21st Century Software Engineering”, ICSE 2006, pp.12-29 ,May 2006.

[7]Canfora, G., Fasolino, A.R., Frattolillo, G. and Tramontana, P. , “Migrating interactive legacy systems to Web services”, Software Maintenance and Reengineering, Proceedings of the 10th European Conference on Volume 00, pp.10-36., March 2006

[8]Chikofsky, E. J. and Cross, J. H., "Reverse

Engineering and Design Recovery: A Taxonomy," IEEE Software, vol. 7, no. 1, pp. 13-17, Jan./Feb. 1990

[9]Durupt, A., Remy, S., Ducellier, G., Guyot, E., “A new reverse engineering process, the combination between the knowledge extraction and the geometrical recognition techniques” Computers & Industrial Engineering, 2009. CIE 2009. , pp. 1367 - 1372., 2009

[10] Fisher, R.B., "Applying knowledge to reverse engineering problems", Computer Aided Design, CAD'04, Vol. 36, pp501-510, 2004.

[11] Gomaa, H. , Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures, Boston: Addi-son-Wesley Professional, 2004.

[12] IBM, Rational Unified Process – Best Practices for Software Development Teams, 2003. Retrieved January, 2010, from http://www.ibm.com/developerworks/

[13] ISO, ISO/IEC 12207, Systems and Software Engineering – Software Life Cycle Processes, 2008.

[14] Jacobson, I., Booch, G. and Rumbaugh, J., The Unified Software Development Process, Addison-Wesley, 1999

[15] Kettemborough, C., “The Prototyping Methodology“., Whitehead College, University of Redlands, 1999

[16] Kruchten, P., The Rational Unified Process: An Introduction, 3rd Edition, Reading, Massachusetts: Addison-Wesley, 2003

[17] Landauer, C. and Bellman, K.L., “Wrappings for software development”, System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on Volume 3, pp.420 -429, 6-9 Jan.1998

[18] Larman, C. and Basili, V. R., "Iterative and Incremental Development: A Brief History". IEEE Computer (IEEE Computer Society) vol.36 no.6,pp. 47–56, June 2003

[19] Liu, K., Alderson, A. and Qureshi, Z., Requirements Recovery from Legacy Systems by Analyzing and Modelling Behavior. ICSM, IEEE Computer Society Washington, DC, USA.. 1999

[20] Lu, C.W., Chu, W. C., Chang, C.H., Chung, Y.C., Liu, X. and Yang, H., Reverse Engineering, Handbook of Software Engineering and Knowledge Engineering Vol. 2, 2002

[21] Nogueira, J.C., Jones, C. and Luqi, "Surfing the Edge of Chaos: Applications to Software Engineering", Proceedings of 2000 Command and Control Research and Technology Symposium, Monterey, CA, pp. 1-13.,26-28 June 2000.

[22] Pressman, R. S., *Software Engineering: A Practitioner’s Approach 6/e*, McGraw-Hill, 2005

[23] Rayson, P., Garside, R., and Sawyer, P., Assisting Requirements Recovery from Legacy Documents., Available from: http://www.comp.lancs.ac.uk/computing/users/paul/publications/rgs00_sebpc.pdf. [Accessed 02/02/2008], 2008

[24] Sommerville, I., *Software Engineering. 6th Edition*, Addison-Wesley, United Kingdom, 2001

[25] Warren, I., The Renaissance of Legacy Systems: Method Support for Software-System Evolution, Springer, London, 1998