運動成績優良學生升學輔導網路作業系統規劃、發展與建置—極致軟體製程之實踐

國立台灣體育學院
郭瑞庭、陳政雄

摘 要

本論文說明『運動成績優良學生升學輔導網路作業系統』建置計畫之背景、功能需求、專案目標、以及所採用之軟體過程模型。其中，系統建置所依循的軟體過程模型則是本論文主要探討的主題。由於專案人力資源有限且時程不容延遲，本專案並未採行嚴謹之軟體工程過程模型，而評估採行極致軟體製程（eXtreme Programming, XP）進行開發，本論文將探討此種敏捷軟體開發方法論之實踐經驗，並彙整本系統建置與運作之情形、專案管理度量、以及所獲得之經驗。

關鍵詞：極致軟體製程、專案管理過程、軟體發展過程模型、敏捷軟體開發、專案管理度量

The Planning, Development, and Implementation of Network Operational System for the Recruiting of Students Who are Superior in Sport Achievement --- The Practices on Extreme Programming

Abstract

This paper describes the background for the implementation plans of the network operational system that is dedicated to the recruiting of students who are superior in sport achievement. Besides that, the functional requirements demand, project goal, and software process model adopted for that system are also explored. Among them, the software process model that is adopted as the implementation approach for the system is the primary issue of this paper. Due to the limitation of project human resource and the prohibition of project schedule postponement, we do not adopt rigorous software engineering process model, and instead, the extreme programming is adopted as the development approach. In this paper, we will discuss the practical experiences acquired from conforming to this agile software development methodology, describe the system operational status, analyze the project management metrics that is collected in the course of the project , and summarize our learning experiences.

Key words: Extreme Programming, Project Management Process, Software Process Models, Agile Software Development, Project Management Metrics

壹、前言

本論文說明『運動成績優良學生升學輔導網路作業系統』建置計畫之背景、功能需求、專案目標、以及所採用之軟體過程模型。其中，系統建置所依循的軟體過程模型則是本文主要探討的主題。本專案並未採用嚴謹之軟體工程過程模型，而評估採行「極致軟體製程」（eXtreme Programming, XP）進行開發。我們先對本系統建置計畫之背景加以說明。其次，描述我們為提升本專案之成功機率所執行之可行性評估暨專案啟動會議。接著，說明本專案之目標，並詳細描述有關本系統開發工程所採行之軟體發展過程模型---「極致軟體製程」以及採用該一敏捷軟體開發模型之原因。最後，彙整本系統建置與運作之情形、專案管理度量、以及所獲得之經驗。

貳、背景說明

教育部依據行政程序法規定以行政協助方式請求國立台灣體育學院辦理「95 學年度中等以上學校運動成績優良學生升學輔導甄審甄試及單獨招生資訊彙整事宜」，詳細之工作內容請參考「94 學年度中等以上學校運動成績優良學生升學輔導甄審甄試試務工作報告」以及教育部 94 年 5 月 4 日召開之「研商由專責學校辦理中等以上學校運動成績優良學生升學輔導甄審甄試及單獨招生資訊彙整等相關事宜會議紀錄」之決議。

『運動成績優良學生升學輔導網路作業系統』建置係該委託案工作說明中之關鍵工作項目。主要包含：「運動成績優良學生甄審甄試作業」以及「大專院校運動成績優良學生單獨招生彙整公告」兩大部分。其中「運動成績優良學生甄審甄試作業」包含「各招生學校甄審甄試名額需求填報」、「運動成績優良學生網路報名」與「運動成績優良學生錄取分發」三大子功能，「大專院校運動成績優良學生單獨招生彙整公告」亦包含「各校單獨招生資訊填報」與「各校錄取放榜後之榜單填報」兩大子功能。

參、可行性評估與專案啟動

『運動成績優良學生升學輔導網路作業系統』建置案經過專案團隊成員進行可行性評估後，認為在技術與成本兩個面向皆屬可行，惟管理面向是一大考驗。主要關鍵在於本案開發人員皆有本職工作，只能運用加班方式進行開發，然而，從管理角度分析，對於為期一個月開發期的小型系統而言，短期加班尚可勉力為之，但對於需要高度腦力的軟體開發工作，開發期長達十個月之久的系統，使得超時工作成為常態時，軟體開發的效率與效能將快速下降，這種狀況不僅違反軟體工程原則，也是軟體發展實務所禁止的。

綜合以上所述，從管理面向而言，本案實應採行委外開發，惟政策指示為達成任務，要求本專案成員依作業時程全力配合。

依據在芝加哥舉行之專案管理年會的調查報告（Gido and Clements, 1999；PMI, 2000）：專案成功的前五項因素分別是使用者直接參與（User Involvement）、高階主管支持（Executive Management Support）、專案需求定義清楚（Clear Statement of Requirements）、規劃適當（Proper Planning）、以及合理的期望（Realistic Expectations）。為了在專案一開始，即將它們安排妥當，本案奉核後立即啟動專案，釐清本專案的定義與範疇（Project Scope）、欲採行之軟體發展過程模型（Software Development Process Model）、工作細分架構（Work Breakdown Structure）、核心成員的角色以及參與期程、對本校現有系統和制度上可能造成的介面影響、專案期程與進度的規劃方式、管控的原則與方法、資源的需求等。其中較關鍵之議題包含專案需求之釐清以及欲採行之軟體發展過程模型。

專案需求之釐清包含欲產生軟體產品之功能，以及用以產生它們的過程；需求工程（Requirements Engineering）過程是一種系統化的需求制定過程，該過程透過適切的方法分析與瞭解問題、並將所獲結果以不同表達方式記錄，並查驗所瞭解事項是否正確，藉以訂出相關的需求。需求工程的重點在於釐清需要設計的事項，而非討論如何將需求事項實現。本案『運動成績優良學生升學輔導網路作業系統』之建置，網頁功能部分事先並無明確的客戶需求，僅有部分輸出表單可供參考，必須採行適當方法在短時間內釐清客戶與使用者需求。

因應不同性質之軟體產品發展，軟體產業界提出了各種不同的軟體發展過程模型。軟體發展過程模型定義一系列的活動以達成產出軟體成品的目的，這些過程成為在軟體專案中的接合劑，連結軟體開發技術實作面及軟體專案管理面的相關議題並提供解決的流程及方向。下節將簡介幾個主要模型，並提出本案所採行之發展方法論。

肆、需求分析與軟體發展過程

典型的軟體發展生命週期（Software Development Life Cycle, SDLC）各階段及其在系統發展生命週期中的角色如圖 4-1 所示（US DOD, 1989）。其中軟體發展生命週期所涵蓋的範圍, 廣義而言, 係從發展規劃開始至系統壓力和情境測試（system stress and scenario testing）結束。若純就軟體發展者角度而言，軟體發展係從軟體需求分析開始至軟體整合測試結束為止。

軟體發展生命週期各階段應執行之工作內容、各階段應達成的目標、以及如何執行評估審查等, 可參考美國軍規 DOD-STD-2167A（US DOD, 1988）、MIL-STD-1521B（US DOD, 1985）、DOD-STD-498 （US DOD, 1994）等, 或是軟體產業界所採用之 IEEE 12207（IEEE/EIA, 1997）等。至於各階段的執行方式則取決於所採用軟體發展生命週期模型或稱軟體發展過程模型。

軟體發展過程模型（Software Process Models）又稱為生命週期模式（Life-Cycle Models）, 它使用概念性的表示描述一連串的軟體開發活動或階段以及各開發階段之間的關係。至於各開發階段的時間先後順序亦表現於過程模型中。不同的過程模型可用於開發不同特性的軟體系統。學者們對軟體發展過程模型所下的定義包括『過程是一系列經過特殊安排的步驟以達到特定的目標，軟體開發過程的目標是完成或改進符合品質要求的軟體產品（Heineman，1994）。』以及『過程是一個系統，它將輸入、活動、工作流程、資訊流程及其他相關的項目轉換成特定的結果與產品（Simon，1998）。』

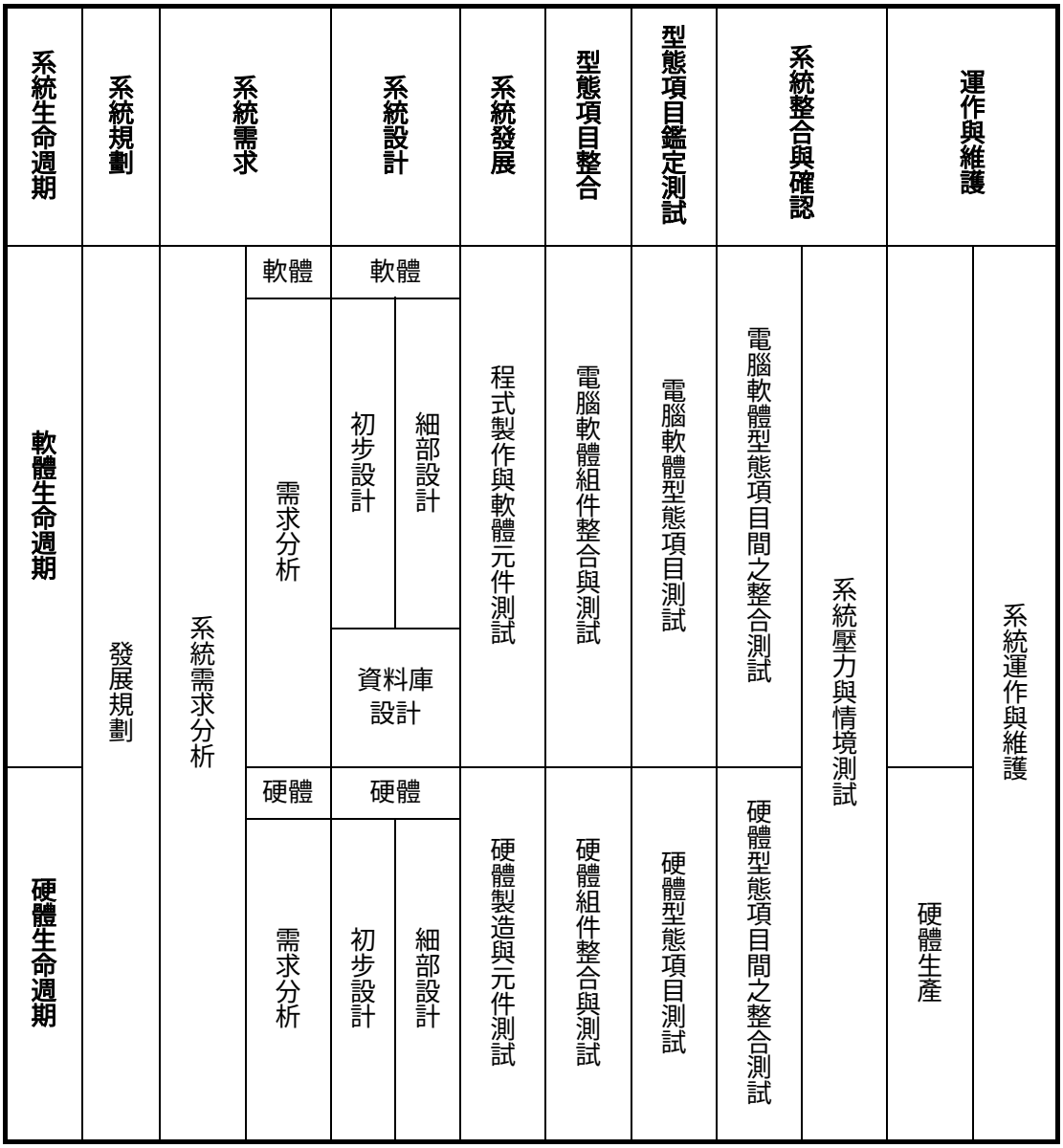

圖 4-1　系統發展生命週期

使用軟體開發過程模型的優點包含：

- 提供標準化的軟體開發過程
- 描繪系統主要的功能及特性
- 統一的概念與術語有助於溝通、規劃及管理。

- 減少軟體開發中的複雜性
- 增加軟體開發的成功率
- 提供評估及查驗的機制
- 提供系統演化的基礎
- 使軟體專案易於管理

藉由掌握各個開發階段的重心及產出，軟體專案管理的難度可以降低，能見度可以提昇。

截至目前為止，已有許多軟體發展過程模型（Hirsch, 1985; Davis, 1992; Davis, 1998; Pressman, 2005）被發展出來，諸如傳統瀑布式模型（Waterfall model）、雛型模型（Prototyping model）、進化式模型（Evolutionary model）、運作式原型模型（Operational Prototyping model）、遞增式模型（Incremental model）、螺旋式模型（Spiral model）等，其中傳統瀑布式模型可說是最早期也是過去最通用的軟體發展過程模型。

Royce 於 1970 年提出瀑布式模型（Royce, 1970）的最初版本，其主要特性如下：

- 每個階段須配合一個驗證和確認（Verification & Validation）行動，其主要目的係儘可能排除該階段產品所可能產生的問題。
- 為了獲得成功的軟體產品，每個階段的目標都依序達成，且相鄰階段的產品須儘可能地反覆驗證。

圖 4-2 係典型之軟體發展過程（US DOD, 1988），其採用瀑布式模型，非常嚴謹的一個階段接著一個階段循序漸進方式進行軟體發展，適合軟體工程能量不高的單位或機構。瀑布式模型的優點是容易管理，它的特點強制分開了設計與實作，可以得到比較穩固且易於修改的軟體系統。瀑布式模型亦有其缺點，譬如，易誤解使用者需求且不易適應使用者針對軟體能量的動態需求。使用瀑布式模型開發軟體時，客戶必須在設計開始之前提出軟體需求；軟體設計師則必須在編碼實作之前提出設計策略。如果開發階段需求有了變更，則軟體需求、設計、以及實作等活動都必須重新來過。

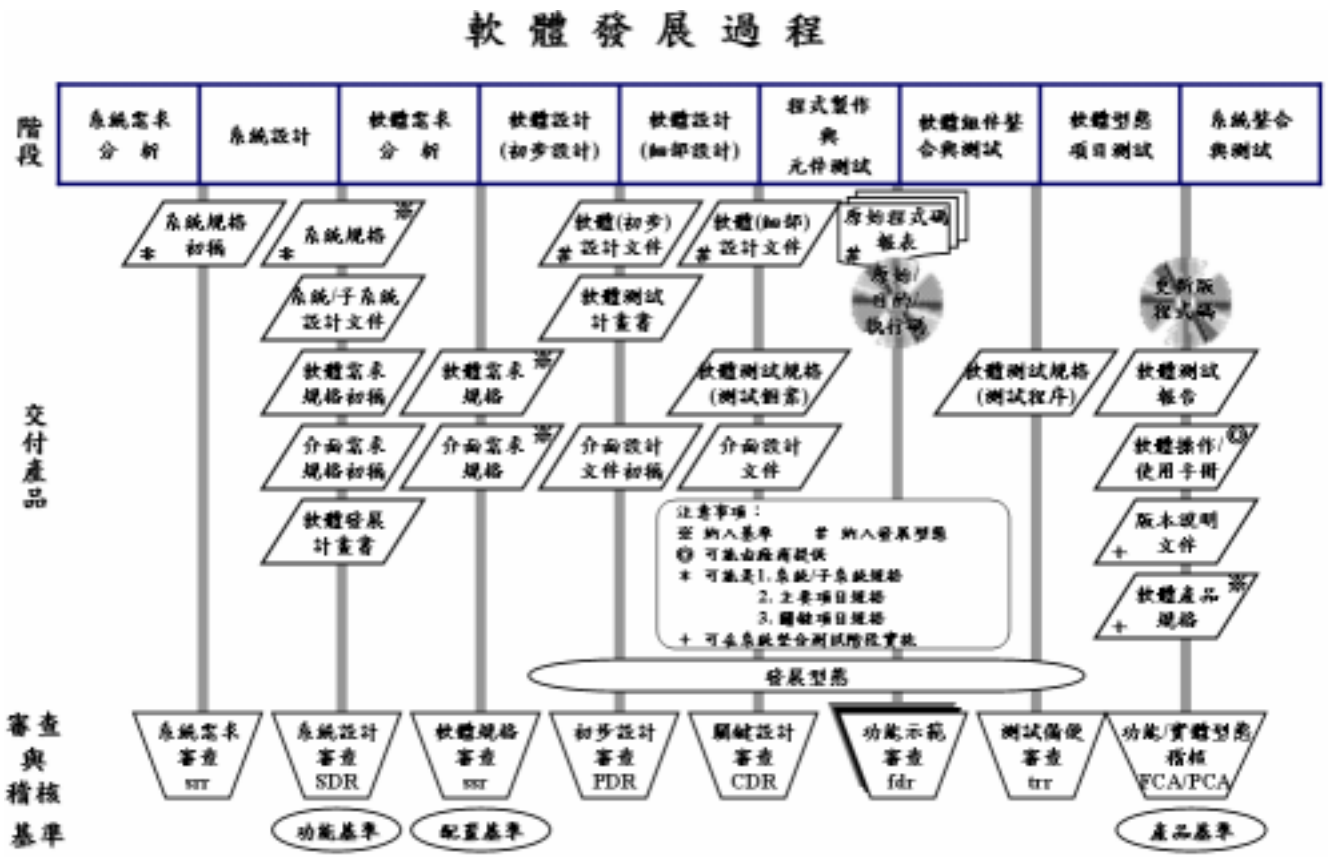
軟體發展過程
階段
系統需求分析
系統設計
軟體需求分析
軟體設計(初步設計)
軟體設計(細部設計)
程式製作與元件測試
軟體組件整合與測試
軟體型態項目測試
系統整合與測試
交付產品
系統規格初稿
系統規格
軟體(初步)設計文件
軟體(細部)設計文件
原始程式碼報表
系統/子系統設計文件
軟體測試計畫書
軟體需求規格初稿
軟體需求規格
軟體測試規格(測試個案)
軟體測試規格(測試程序)
軟體測試報告
介面需求規格初稿
介面需求規格
介面設計文件初稿
介面設計文件
軟體操作/使用手冊
軟體發展計畫書
版本說明文件
軟體產品規格
注意事項：
納入基準
納入發展型態
可能由廠商提供
可能是1.系統/子系統規格
2.主要項目規格
3.關鍵項目規格
可在系統整合測試階段實施
發展型態
審查與稽核基準
系統需求審查 srr
系統設計審查 SDR
軟體規格審查 ssr
初步設計審查 PDR
關鍵設計審查 CDR
功能示範審查 fdr
測試備便審查 trr
功能/實體型態稽核 FCA/PCA
功能基準
配置基準
產品基準

圖 4-2　軟體發展過程

進化式模型的基本概念是，先做出一個初始的版本，然後根據使用者意見再做修改，而且不斷的重複此步驟，直到開發出一個最適當的軟體系統為止。此模型優點是，它的規格可以用遞增的方法發展。但從工程與管理角度來看，此模型的缺點包括程序不是很清楚以致於難以評估進度，且因它允許將需求的制訂和設計決策延後，將導致軟體系統結構不良，難以理解與維護。遞增式模型則是介於上述兩個模型之間的開發方法，它結合了這兩個模型的優點。

遞增式模型於 1985 年由 Hirsch（Hirsch, 1985）提出。採用此一模型的基本前提為發展者已事先瞭解大部分的系統需求。若依據傳統瀑布式模型進行軟體發展，則係在一個軟體發展生命週期內，使軟體產品滿足全部的系統需求；在遞增式模型開發程序中，客戶必須概略的提出系統需提供的功能。客戶必須提出哪些功能是最重要的、哪些功能比較不重要。然後定義出一系列的增量模組，每一個增量模組均提供了某一部份的系統功能。增量模組所配置的功能則根據功能的優先順序來決定，具有最高優先順序的功能最先提供給客戶使用。

軟體系統的增量模組決定之後，就可以先詳細定義出第一個增量模組的功能需求，然後再利用適當的開發程序開發這個增量模組。在開發階段則可以針對下一個增量模組的需求進行分析，但是不允許對目前這個增量模組的需求做變更。若依循遞增式模型，則是用了 N 個軟體發展生命週期，才使軟體產品滿足既定的系統需求，其中每一個軟體發展生命週期只選擇部分系統能量來發展。如圖 4-3 所示，第一個生命週期稱為第一增量（Incremental 1），須完成系統能量一；第二個生命週期稱為第二增量（Incremental 2），不僅須完成系統能量二，亦須將系統能量一和系統能量二整合為一體：以此類推，第 N 個生命週期稱為第 N 增量（Incremental N），不僅須完成第 N 個系統能量，亦須將所有已開發的系統能量整合為一體。

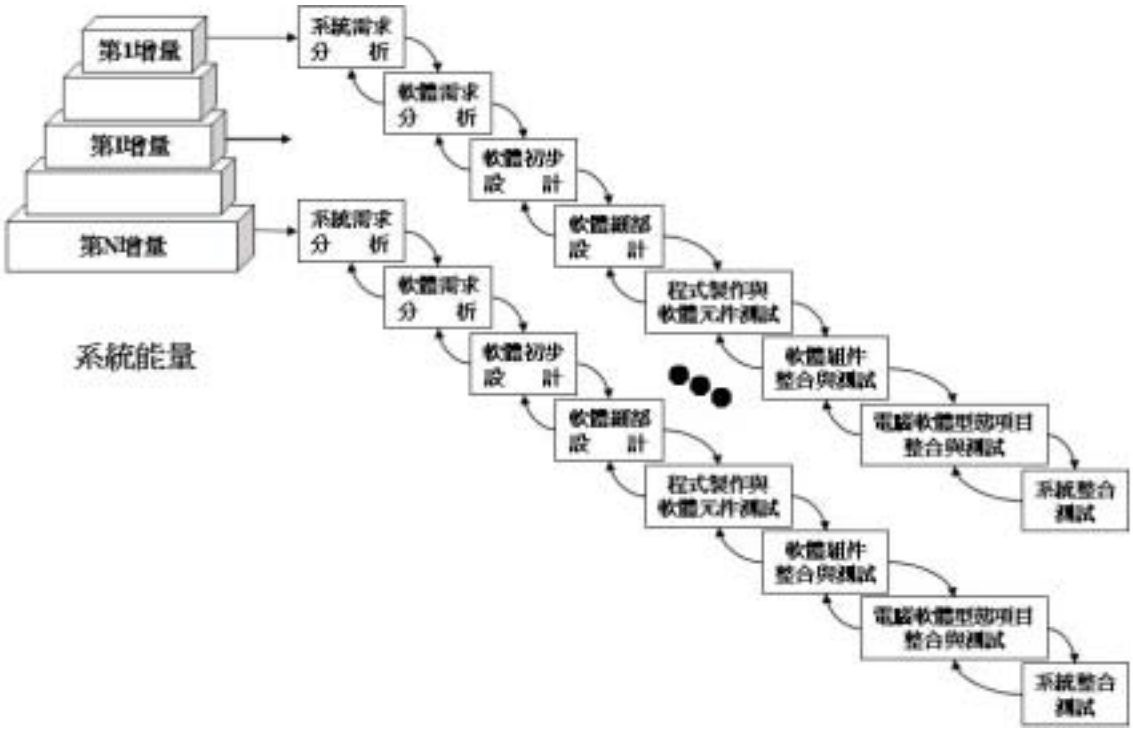

圖 4-3　遞增式軟體過程模型

較之傳統瀑布式模型，遞增式模型具有下列優點：

- 適合大型軟體開發。
- 使用者可提前獲得可運作的軟體系統。
- 軟體存活期長。
- 具軟體性能提昇能量，可降低發展過程中使用者需求變更的可能性。

相對地，遞增式模型亦有其缺點，例如，不適用於小型軟體開發、對文件的一致性造成不良影響、以及增加軟體型態管理（Software Configuration Management）的風險等。

90 年代初期，遞增式模型衍生出一個新的方法論，稱為「極致軟體製程」或「終極軟體製程」（eXtreme Programming, XP）（Beck, 2004）。這個新的方法論由 Kent Back 與 Ward Cunningham 一起合作提出，其成功來自於強調客戶滿意及促進團隊合作。它的特點是開發與提交極小單位的增量功能、客戶參與程序、不斷的改善程式、以及無私的程

式設計方法。XP 的支持者（Wells, 2006）自認 XP 是一種嚴謹且有規律的軟體開發方式，其發展已有 10 年以上的歷史，且已有一些實證個案。XP 授權開發人員自信地回應客戶的需求異動，甚至是在軟體生命週期的後期。XP 同時強調團隊合作，專案管理人員、客戶以及開發人員等利益關係人都是對開發高品質軟體有高度貢獻的團隊成員。

XP 的主要精神是以下列四個面向來促進軟體開發：

- 溝通（communication）：XP 的程式設計師與其客戶及程式設計師同僚溝通。
- 簡化（simplicity）：保持其設計簡單且清晰。
- 回饋（feedback）：一開始便不斷從測試軟體獲得回饋，儘可能提早交付系統給客戶並且立即依據其建議改善軟體。
- 勇氣（courage）：XP 的程式設計師能夠勇於回應持續改變中的需求及技術。

上列四個面向基本上即實踐敏捷（Agile）軟體發展宣言（Martin, 2003）的四項價值觀，如表 4-1 所示。

表 4-1：XP 主要精神與敏捷軟體發展四項價值觀的對照表

XP 的主要精神	敏捷軟體發展的四項價值觀
溝通	個人及相互交流勝過軟體工程和工具
簡化	可運行的軟體勝過完整廣博的文檔資料
回饋	與用戶合作勝過合約談判
勇氣	回應變化勝過服從計畫

『運動成績優良學生升學輔導網路作業系統』之開發具有下列特性：

- 軟體功能需求不明確：客戶僅能提供部分報表格式，無法提供明確的軟體系統之功能需求。
- 潛在之需求異動頻率高：由於軟體功能需求不明確，必須待客戶看到雛型軟體系統時，才會具體提出相關需求，導致軟體生命週期的後期仍有需求異動，提高了軟體功能如期完成的風險。
- 沒有緩衝時間：甄審甄試作業各項時程已公告，不允許任何變動或延遲。
- 明訂的功能增量模組：必須依據甄審甄試作業時程，依序完成五個增量模組。

- 人力資源不足：以 0.5 個程式設計師人力，搭配 0.5 個專案管理與測試支援人力，遠低於正規軟體工程所規範之需求人力。

瀑布式模型、演化式模型、以及遞增式模型等正規軟體工程方法論並不適合解決前述之議題，而 XP 的四個主要精神：積極改善溝通、尋求簡化途徑、獲得回饋合作、以及勇於回應變化等，正是這些議題的較佳解決方案。為了提高生產力，且為了使交付的軟體是客戶真正需要的軟體，因此，本系統採行較不嚴謹的極致軟體製程之開發模型，期望能順利達成任務。

伍、系統概觀與增量模組規劃

95 學年度中等學校以上運動成績優良學生升學輔導網路作業系統主要包含：「運動成績優良學生甄審甄試作業」與「大專院校運動成績優良學生單獨招生彙整公告」兩大部分。其中「運動成績優良學生甄審甄試作業」包含「各招生學校甄審甄試名額需求填報」、「運動成績優良學生網路報名」與「運動成績優良學生錄取分發」三大子功能，「大專院校運動成績優良學生單獨招生彙整公告」亦包含「各校單獨招生資訊填報」與「各校錄取放榜後之榜單填報」兩個子系統，整個升學輔導系統的主要功能如下：

一、運動成績優良學生甄審甄試作業系統

- 各招生學校甄審甄試需求名額填報系統
- 運動成績優良學生線上報名系統
- 運動成績優良學生錄取分發系統

二、大專院校運動成績優良學生單獨招生彙整公告系統

- 各校上網填報單獨招生資訊填報系統
- 各校上網填報單獨招生錄取榜單系統

依據甄審甄試作業時程需求與完成的順序，規劃增量模組如後：

- 第一增量：各招生學校甄審甄試需求名額填報系統
- 第二增量：各校上網填報單獨招生資訊填報系統
- 第三增量：運動成績優良學生線上報名系統
- 第四增量：運動成績優良學生錄取分發系統
- 第五增量：各校上網填報單獨招生錄取榜單系統

陸、最終產出

歷經十個月的開發，各系統依據作業需求與完成的順序，依照各系統所使用程式、開發工具以及相關說明，表列於表 6-1。

表 6-1：運動成績優良學生升學輔導網路作業系統產出與開發工具

一、各招生學校甄審甄試需求名額填報系統			
項次	**程式名稱**	**工具**	**備註**
01	系統登入與密碼檢核	網頁	一般學生
02	忘記密碼處理*	網頁	一般學生
03	填報系統主要管理介面	網頁	一般學生
04	新增申請資料	網頁	一般學生
05	修改申請資料	網頁	一般學生
06	刪除申請資料	網頁	一般學生
07	需求資料上傳與列印	網頁	一般學生
08	申請學校資料維護*	網頁	一般學生
09	各校已申請資料查詢	網頁	一般學生
10	各校學校代碼查詢*	網頁	一般學生
11	各校申請甄審甄試運動分項查詢	網頁	一般學生
12	各校申請甄審甄試所有資料查詢	網頁	一般學生
13	系統登入與密碼檢核	網頁	身心障礙生
14	填報系統主要管理介面	網頁	身心障礙生
15	新增申請資料	網頁	身心障礙生
16	修改申請資料	網頁	身心障礙生
17	刪除申請資料	網頁	身心障礙生
18	申請資料上傳與列印	網頁	身心障礙生
19	各校已申請資料查詢	網頁	身心障礙生

二、各校上網填報單獨招生資訊填報系統			
項次	程式名稱	工具	備註
01	系統登入與密碼檢核	網頁	
02	填報系統主要管理介面	網頁	
03	新增招生資訊	網頁	
04	整批匯入招生資訊	網頁	
05	修改招生資訊	網頁	
06	刪除招生資訊	網頁	
07	申請資料上傳與列印	網頁	
08	各校單獨招生試務資訊查詢（所有資料）	網頁	
09	各校單獨招生試務資訊查詢（依運動項目）	網頁	
三、運動成績優良學生線上報名系統			
項次	程式名稱	工具	備註
01	系統登入與密碼檢核	網頁	
02	報名系統主要管理介面	網頁	
03	新增考生報名資料	網頁	
04	整批匯入考生報名資料	網頁	
05	修改考生報名資料	網頁	
06	刪除考生報名資料	網頁	
07	初審名冊列印	網頁	
08	考生志願填報	網頁	
09	志願選填檢查清單列印	網頁	
10	各考生志願表列印	網頁	
11	申請資料上傳與列印	網頁	
四、運動成績優良學生錄取分發系統			
項次	程式名稱	工具	備註
01	甄審甄試種類維護（匯入、新增、修改、刪除）	VFP 應用程式	甄審、甄試
02	運動種類維護（匯入、新增、修改、刪除）	VFP 應用程式	甄審、甄試
03	參加甄審試學校資料維護（匯入、新增、修改、刪除）	VFP 應用程式	甄審、甄試
04	參加甄審試考生資料維護（匯入、新增、修改、刪除）	VFP 應用程式	甄審、甄試
05	考生准考證列印	VFP 應用程式	甄試
06	考生術科加分等級維護	VFP 應用程式	甄試

07	術科考生名單列印	VFP 應用程式	甄試
08	資料審查名單列印	VFP 應用程式	甄審
09	驗卡資料維護	VFP 應用程式	甄試
10	驗卡作業	VFP 應用程式	甄試
11	筆試科目維護	VFP 應用程式	甄試
12	標準答案建立與維護	VFP 應用程式	甄試
13	試題配分設定與列印	VFP 應用程式	甄試
14	考生答案讀卡	VFP 應用程式	甄試
15	讀卡缺卡列印	VFP 應用程式	甄試
16	讀卡答案比對作業	VFP 應用程式	甄試
17	讀卡缺卡列印	VFP 應用程式	甄試
18	讀卡結果建立	VFP 應用程式	甄試
19	特殊答案給分作業	VFP 應用程式	甄試
20	讀卡結果計算（學科分數計算）	VFP 應用程式	甄試
21	讀卡結果抽測（學科成績）	VFP 應用程式	甄審
22	審查資料維護（條、款、名次）	VFP 應用程式	甄審
23	術科檢定結果輸入	VFP 應用程式	甄試
24	總成績結算	VFP 應用程式	甄審、甄試
25	電腦分發作業	VFP 應用程式	甄審、甄試
26	學科缺考名單列印	VFP 應用程式	甄試
27	術科缺考名單列印	VFP 應用程式	甄試
28	未參加分發名單（放棄、資格不符）	VFP 應用程式	甄審、甄試
29	分發名單列印	VFP 應用程式	甄審、甄試
30	分發失敗名單列印（志願太少）	VFP 應用程式	甄審、甄試
31	高中甄試成績組距列印	VFP 應用程式	甄試
32	國中甄試成績組距列印	VFP 應用程式	甄試
33	考生錄取名單（榜單）列印	VFP 應用程式	甄審、甄試
34	各運動種類人數統計與列印	VFP 應用程式	甄審、甄試
35	考生填報志願表列印	VFP 應用程式	甄審、甄試
36	術科檢定不合格名單列印（缺考、不合格）	VFP 應用程式	甄試
37	各學校錄取分發名單列印	VFP 應用程式	甄審、甄試
38	各畢業學校錄取名單列印	VFP 應用程式	甄審、甄試
39	甄試成績單列印（國中、高中）	VFP 應用程式	甄試
40	甄審上榜名單列印	VFP 應用程式	甄審

五、各校上網填報單獨招生錄取榜單系統			
項次	程式名稱	工具	備註
01	系統登入與密碼檢核	網頁	
02	榜單系統主要管理介面	網頁	
03	新增單獨招生榜單資料	網頁	
04	整批匯入單獨招生榜單資料	網頁	
05	修改單獨招生榜單資料	網頁	
06	刪除單獨招生榜單資料	網頁	
07	單獨招生榜單列印	網頁	
08	單獨招生榜單資料上傳與列印	網頁	

註：'*'代表該程式係各系統共用程式。

柒、XP 的規則與實務作法之實踐

XP 屬於一種輕量級的軟體方法論（Lightweight software methodology）。這種輕量級的方法論，挑戰了軟體開發中許多傳統的信條，例如基準文件（Baseline Documents）的建立、需求異動之管制（Change Control）、以及審查與稽核（Review and Audit）等活動並未在 XP 中有嚴謹的規範。XP 在規劃、設計、編碼及測試四個階段總共提出了 28 條簡單的規則及實務作法（The Rules and Practices of XP）（Jeffries, 2001；Beck, 2004）。這些規則與實務作法也應用了一些軟體工程的原理，例如單元測試（Unit Tests）、程式碼整合（Code Integration）、以及允收測試（Acceptance Test）等，但並未深入說明其實際內容。

本節將針對 XP 在規劃、設計、實現及測試四個發展階段所提出的 28 條簡單規則及實務作法加以描述，並探討這些規則與實務作法在『運動成績優良學生升學輔導網路作業系統』建置案之實踐。

一、規劃階段

- 撰寫使用者情節（User stories are written.）：本案之執行並未依循此一實務作法。
- 藉由發行規劃建立時程（Release planning creates the schedule.）：依據合約規範

之上線日期進行時程規劃。

- 提高小版本的發行頻率（Make frequent small releases）：依據合約規範之上線日期進行發行規劃，但適當時，會另行提高發行頻率，使客戶或使用者能提前使用某些新增功能，即使只比之前的版本多了一點功能。
- 量度專案的速度（The Project Velocity is measured）：專案速度的量度結果請參考第捌節。
- 將專案開發工作切割成許多的反覆（The project is divided into iterations）：本專案開發工作切割成 5 個反覆。
- 以反覆規劃啟動每個反覆（Iteration planning starts each iteration）：本專案以反覆規劃啟動每個反覆。
- 人員輪調（Move people around）：本專案人力不足，並未執行人員輪調。
- 每日的站立會議（A stand-up meeting starts each day）：本專案並未執行每日站立會議。
- 修訂不合用的 XP 過程（Fix XP when it breaks）：本案之執行遵循此一實務作法。

二、設計階段

- 簡化（Simplicity）：本專案之設計考量僅針對系統應有之必要功能加以設計，盡量簡化設計之複雜度。
- 選擇一個系統象徵（Choose a system metaphor）：本專案之設計並未選用特定之系統象徵或隱喻，但在規劃初期會先行評估某些關鍵技術風險的解決方案。
- 設計會議時使用 CRC 卡（Use CRC cards for design sessions）：本專案之設計並未採用 CRC 卡。
- 設計突破方案以降低風險（Create spike solutions to reduce risk）：每一個反覆皆利用雛形法與使用者快速溝通需求。
- 功能不要太早加入（No functionality is added early）：依據反覆規劃之內容納入相關功能。

- 隨時隨地儘可能重整（Refactor whenever and wherever possible）：本案之執行遵循此一實務作法，有任何程式碼完成時，即會持續整合。

三、編碼階段

- 客戶是隨時可得的（The customer is always available）：本案並無客戶或客戶代表可全程駐點參與專案團隊的開發工作，但本系統之客戶或其代表能即時回應系統雛形之運作結果。
- 程式碼必須依循編碼標準（Code must be written to agreed standards）：本系統之程式碼編碼依循 PHP 語言編碼指引。
- 優先編寫單元測試程式（Code the unit test first）：本案之執行並未嚴格遵循此一實務作法。
- 所有產出之程式碼都是雙人組設計（All production code is pair programmed）：本案之執行並未依循此一實務作法。
- 一次只有一對雙人組整合程式碼（Only one pair integrates code at a time）：本案之執行並未依循此一實務作法。
- 隨時整合（Integrate often）：本系統於每一反覆中隨時整合進行整合。
- 採行集體程式碼擁有權（Use collective code ownership）：本案之執行並未依循此一實務作法。
- 直到最後才進行最佳化（Leave optimization till last）：本系統於每一反覆之最後才進行最佳化。
- 決不超時工作（No overtime）：本案之執行並未依循此一實務作法。

四、測試階段

- 所有程式碼必須有單元測試（All code must have unit tests）：本案之執行遵循此一實務作法。
- 所有程式碼在發行前必須通過單元測試（All code must pass all unit tests before it

can be released)：本案之執行遵循此一實務作法。

- 發現臭蟲立即建立測試（When a bug is found tests are created)：本案之執行遵循此一實務作法。
- 隨時實施允收測試並公佈結果（Acceptance tests are run often and the score is published)：本系統於每一反覆中隨時實施允收測試，但客戶或客戶代表並未實際撰寫關於系統情節的測試個案與程序，且測試結果並未公佈。

捌、專案管理度量

在軟體專案的開發過程中或於開發末期度量團隊生產力是一件有趣且必要的工作。軟體產業界對軟體開發生產力之估算或度量主要有兩種方法：原始程式碼行數（Source Lines Of Code, SLOC）與功能點數（Functional Points, FP)。這兩種度量方法各有其優缺點，也各有其支持的對象。本論文將採行 SLOC 的度量方法。雖然 SLOC 本身並未被規範於相關標準文件中，但若探討範圍為原始程式碼行數時，一般都接受 SEI 技術報告 CMU/SEI-92-TR-20 中的計算法（Park，1992)；亦即不計算程式中註解行數以及不可執行的部分。

Reifer 於 2002 年發表了一篇有關軟體產業界生命週期績效資訊的論文（Reifer, 2002)，並於 2004 年更進一步發表軟體產業界之軟體成本、品質與生產力標竿之論文（Reifer，2004)。該論文除了以 12 個應用領域檢視軟體生產力資訊，比較美國地區與其他地區之生產力差異外，並依據最受軟體產業界歡迎的生命週期過程模型，提出人力與時程之階段分配圖。參考該論文調查結果所提供之產業標竿資訊，各軟體公司或軟體開發團隊可用以判定它們的績效表現，並可做為採行或引入新軟體科技之投資報酬的分析基礎。

表 8-1 列出『運動成績優良學生升學輔導網路作業系統』之生產力資訊。其中除了「運動成績優良學生錄取分發系統」之應用領域屬於資料處理外，其他四個子系統之應用領域皆屬於網站應用。表中 SLOC 為各個子系統之原始程式碼行數，SM（Staff Month）

為各個子系統開發所投入之人月，SLOC/SM 則表示每一人月之生產力。網站應用部分涵蓋之範圍從軟體需求概念探究至推出上線止，生產力為 1853，而資料處理部分涵蓋之範圍從軟體需求分析至軟體測試完成止，生產力則為 1627。

表 8-1：『運動成績優良學生升學輔導網路作業系統』之生產力資訊

子系統名稱	應用領域	SLOC	SM	SLOC/SM
各招生學校甄審甄試需求名額填報系統	網站應用	3086	1.35	2286
各校上網填報單獨招生資訊填報系統	網站應用	1428	1.4	1020
運動成績優良學生線上報名系統	網站應用	3126	1.6	1954
各校上網填報單獨招生錄取榜單系統	網站應用	1623	0.65	2497
運動成績優良學生錄取分發系統	資料處理	5775	3.55	1627

表 8-2 列出 Reifer 所研究彙整軟體專案生產力之部分資訊（Reifer，2004），網站應用領域之平均生產力為 330，而資料處理領域之平均生產力則為 275。

表 8-2：相關應用領域之軟體生產力資訊

應用領域	專案數	軟體大小範圍 (KESLOC)	平均生產力 (ESLOC/SM)	平均生產力範圍 (ESLOC/SM)
網站應用	35	20 to 780	330	165 to 500
資料處理	65	10 to 270	275	190 to 985

若以表 8-2 所列之平均生產力為計算基準，運動成績優良學生升學輔導網路作業系統網站應用部分合理之需求人月為 28.1 個人月，資料處理部分合理之需求人月為 21 個人月。因此，本案之合理需求總人月為 49.1 個人月；若以本案之需求時程 11 個月而言，平均每個月應投入 4.46 個人力。

實際上，從我們的專案管理度量分析結果，本案之實際開發總人月為 8.6 個人月，開發期間則為 11 個月。亦即，本案之開發總人月被壓縮為合理需求人力的 17.5%。若採用較嚴謹之生命週期過程模型來開發本系統，以如此高的需求人力壓縮比，管理風險將大幅提高，專案成功的機率也相對的降低。

為了降低前述之風險，本案經評估後採行較不嚴謹的極致軟體製程之開發模型，雖

然順利達成任務，但此種極端違反軟體工程實務之作法，無可避免的衍生出一些品質相關問題；例如，基準文件與發展文件之欠缺將衍生後續系統維護與人員異動的困境、系統上線前頻繁之需求異動將導致系統測試的不完整性、以及審查與稽核活動之欠缺將影響專案的可見度。

玖、學習經驗

整個專案各個子系統陸續上線後，仍有若干問題發生，茲分述如下：

一、 根據系統需求定義，整個網路作業系統使用單位（各相關校院）對本專案僅設定單一窗口，但是實際上線後，由於系統的屬性不同，例如各校需求名額填報系統使用者大多為各校的註冊單位，而運動績優生網路報名系統的使用者並非全是各校註冊單位，有些學校是註冊單位，有些學校為負責體育的單位，這應是「需求分析」階段需求單位未能釐清問題。

二、 運動成績優良學生線上報名系統上線當日，即發現先前報名學校的資料，被後來報名學校所刪除，追究問題的原因為程式編碼時，每個登入網頁的使用者其瀏覽進程（Session）並沒有區隔，導致任何使用者登入後，均寫入到相同的報名資料空間，使得後來報名的使用者刪除先前報名的資料，主要關鍵在於測試階段，測試人員沒有使用足夠的測試資料，測試出程式的缺陷。

三、 運動成績優良學生錄取分發系統接收由各招生學校甄審甄試需求名額填報系統所彙整的各校資料，其中有一所學校體育學系的需求名額較為特殊，共需要招收 12 種不同運動種類的運動績優生，每一運動種類最多不超過 3 人，且整個學系招收人數的上限為 25 人，由於系統操作人員疏忽，將該學系錄取學生設定為 36 人，導致分發錯誤。

針對上述發生的問題，未來改進的方式如下：

（一）經由系統實際運作，規劃系統開發與系統使用單位若為不同，或是在系統分析時，系統相關人員能提供各種不同的參考意見，往後系統完成運作後，較

能涵蓋不同的需求差異，系統也較能考量到不同的需求。

（二）由於人力不足與時程壓力，系統測試階段除了系統開發人員的測試（白箱測試、功能測試、整合測試等）外，僅實施少數的驗收測試個案，且測試資料亦嫌不足，以致於無法盡可能測出系統的缺陷，因而導致系統錯誤。任何系統開發專案應該配置足夠人力，擬定完整的測試計畫，預先做好測試設計，並提供更多完整的測試個案與測試程序，確實執行測試，最後並提出完整的測試報告，才可將系統的缺失減到最低。

（三）針對系統操作容易出錯之處，除了增加「警語」（Warning Messages）外，在系統設計時，亦可增加若干的「檢核點」（Check points）以規範及預防可能會發生的操作錯誤。

拾、結論

本論文先說明『運動成績優良學生升學輔導網路作業系統』建置計畫之背景、功能需求、以及專案目標等。其次，描述我們為提升本專案之成功機率所執行之可行性評估暨專案啟動會議。接著，提出本專案未採行嚴謹之軟體工程過程模型之原因，以及經過評估後所採行之軟體發展過程模型---『極致軟體製程』以及採用該一敏捷軟體開發模型之原因。最後，並彙整本系統建置與運作之情形、專案管理度量、以及所獲得之經驗。本專案之所以能順利完成，我們將其歸結於採行敏捷軟體開發模型之一的極致軟體製程 XP，依循 XP 的四個主要精神：積極改善溝通、尋求簡化途徑、獲得回饋合作、以及勇於回應變化等。

拾壹、參考文獻

Beck, K. (2004). *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 2nd ed.

Beck, K. and Fowler, M (2001).*Planning Extreme Programming*. Addison-Wesley.

Boehm, B.W. （1981）. *Software Engineering Economics*. New Jersey: Prentice-Hall, Inc.

Boehm, B. W., et al.(2000) *Software Cost Estimation with COCOMO II*. New Jersey: Prentice-Hall, Inc.

Davis, A., et al. (1998) *A Strategy for Comparing Alternative Software Development Life Cycle Models*. IEEE Trans. Software Eng, Oct., pp.1453-1461.

Davis, A. (1992) *Operational Prototyping：A New Development Approach*. IEEE Trans. Software Eng, Sep., pp. 70-78.

Gido, J., and Clements, J. P. （1999）. *Successful Project Management*. South-Western, a division of Thomson Learning, Inc.

Heineman, G. T. (1994) *Process Modelling with Cooperative Agents*. Proceedings of the Third European Workshop on Software Process Technology, London: Springer-Verlag.

Hirsch, E. （1985）. *Evolutionary Acquisition of Command and Control Systems*. Program Manager, Nov-Dec. pp.18-22.

Humphrey, W. (1989) *Managing the Software Process*. Addison-Wesley, 1989.

Institute of Electrical and Electronics Engineers/Electronic Industries Association. (1998) *Software Life Cycle Processes (IEEE/EIA 12207)*.

Institute of Electrical and Electronics Engineers. (1997) *IEEE Standard for Software Reviews (IEEE Std 1028-1997)*.

International Organization for Standardization/International Electrotechnical Commission. (1995) *Software Life Cycle Processes (ISO/IEC 12207)*. Geneva: ISO/IEC.

International Organization for Standardization. (1997) *ISO 9000-3, Guidelines for Applying ISO 9001:1994 to Computer Software*. Washington, DC: ISO.

Jeffries, R., Anderson, A., Hendrickson, C., and Jeffries, R. E. (2001) *Extreme Programming Installed*. Addison-Wesley.

Martin, R. C. (2003) *Agile Software Development: Principles, Patterns, and Practices*. Prentice-Hall.

Park R. (1992) *Software Size Measurement: A Framework for Counting Source Statements*. CMU/SEI-92-TR-20, Software Engineering Institute, Pittsburgh, PA.

Project Management Institute (PMI) Standards Committee. (2000) *A Guide to the Project Management Body of Knowledge (PMBOK)*. 2000 edition.

Pressman, R. S. (2005) *Software Engineering: A Practitioner's Approach*. New York: McGraw-Hill., 6th ed.

Reifer, D. J. (2002) *Let the Number Do the Talking*. CrossTalk, The Journal of Defense Software Engineering, March 2002, pp.4-8.

Reifer, D. J. (2004) *Industry Software Cost, Quality and Productivity Benchmarks*. The DoD SoftwareTech News, July 2004, Vol. 7, No. 2, pp.3-8, www.softwaretechnews.com, http://iac.dtic.mil/dacs

Royce, W. W. (1970). *Managing the development of large software systems*. In WESTCON, San Francisco, CA.

Schwalbe, K. (2000) *Information Technology Project Management*. Thomson Learning.

Simon, J. M. (1998) *Assessment Using SPICE: A Case Study* in SPICE: The Theory and Practice of Software Process Improvement and Capability Determination, chapter 18, ed. ElEmam, Drouin, Melo, IEEE Computer Society Press, 1998.

Software Engineering Institute. (2000) *CMMI for Systems Engineering/ Software Engineering, CMMI-SE/SW, V1.02: Staged Representation (CMU/SEI-2000- TR-028)*.

The Standish Group. (1995) *CHAOS*.

US Air Force. (1985) *Technical Reviews and Audits for System, Equipments, and Computer Software, MIL-STD-1521B*. June 4.

US DOD. (1985) *Defenses System Software Development, DOD-STD- 2167*. June 4.

US DOD. (1988) *Defenses System Software Development, DOD-STD- 2167A*. Feb. 29.

US DOD. (1994) *Software Development and Documentation, MIL-STD- 498*. Dec. 5.

US DOD. (1989) *System Engineering Management Guide*.

Website of International Organization for Standardization, http://www.iso.ch/.

Wells, D. (2006) *Extreme Programming: A Gentle Introduction*. Retrieved from http://www.extremeprogramming.org/