小型軟體公司軟體產品發展模式最適化之研究

黃明達
淡江大學資訊管理學系
mdhwang@mail.tku.edu.tw

朱懿中
淡江大學資訊管理學系碩專班
gary0628chu@gmail.com

摘要

小型軟體公司在發展軟體產品的過程中，受限於緊迫的時程與資源，承受了極大的考驗。透過導入大型軟體開發準則雖然可以解決軟體產品管理上的需求，但所需要的費用與資源遠超過小型軟體公司所能負擔的範圍，而導入敏捷式開發模式雖然可以滿足開發管理上的需求，但無法完整涵蓋整個軟體產品的管理範疇。

本研究的目的，係透過個案 S 公司 6 年來在軟體產品發展過程中的實務經驗，與微軟解決方案框架（MSF，Microsoft Solution Framework）進行比較分析，找出能夠提高軟體產品市場符合性、降低開發成本、縮短開發時程的執行方法，並提出小型軟體公司發展軟體產品的實務指引，作為未來執行新軟體產品發展的依據。

本研究所提出的小型軟體公司軟體產品發展指引 SPDG（Small Company Software Product Develop Guideline）遵循著 MSFv4 的基本原則、思維與團隊模型，作為軟體產品管理機制的基礎，並依據個案的實務經驗提出了 15 項的執行方法，達到節省開發時程 16.1%、提昇產品品質、提高工作效率的效益。SPDG 能夠讓小型軟體公司同時兼顧既有任務的執行與開發管理機制的逐步改善，為一套經過驗證且確實可行的實務指引。

關鍵詞：軟體產品管理、軟體開發流程、MSF

1. 緒論

1.1 研究背景與動機

在軟體產業中，將軟體專案轉變為可大量複製且具有共通性的軟體產品，是讓軟體公司能夠提高獲利的重要因素；但是軟體產品發展有別於軟體專案，其所要考量的範圍，除了軟體工程、專案管理、開發技術等項目之外，也需要將管理範圍延伸至市場評估、規劃與需求、設計、維護、下架等階段[4]。另外，Christof Ebert（2007）[7]也提及了軟體產品管理的高複雜度，以及缺少正式的教育和主體知識的窘境，因此，對於小型軟體公司而言，發展自有軟體產品的困難程度不言可喻。

小型軟體公司在發展軟體產品時，要能在緊迫的時程與資源下，保有開發的彈性與管控能力，因此面臨了極大的考驗[16]。國外所引進的大型軟體開發準則，如：能力成熟度整合模式（CMMI，Capability Maturity Model Integration），導入成本所費不貲，遠超過小型軟體公司所能負擔的範圍；而針對中小型開發團隊的敏捷式開發模式（Agile Development），則無法完整覆蓋軟體產品的範疇，因此本研究希望透過小型軟體公司的軟體產品發展個案，尋找出可以減少錯誤嘗試的方法，並提出最適化的實務指引，最適化意即小型軟體公司在有限資源的情況下，能符合高效益的運作狀態。

1.2 研究目的

本論文研究目的為：

1. 探討微軟解決方案框架（MSF，Microsoft Solution Framework）所定義的團隊與治理模型，將其與個案實務經驗進行差異分析，彙整出個案所執行的團隊與治理模型。
2. 透過個案遭遇的問題，進行研究分析，進一步提出軟體產品發展過程中，能夠提高軟體產品市場符合性、降低開發成本、縮短開發時程的執行方法。
3. 提出小型軟體公司發展軟體產品的最適化實務指引，作為未來執行新軟體產品發展的依據。

1.3 論文架構

本論文共分為五個章節，各章節描述如下：

- 第壹章 緒論：敘述研究背景與動機、研究目的、論文架構。
- 第貳章 文獻探討：針對與本研究相關領域的文獻，軟體產品管理範圍、各式的軟體開發方法進行探討。
- 第參章 研究設計：說明研究方法、研究對象與研究流程。
- 第肆章 個案研究與分析：描述個案背景、個案軟體產品發展管理機制、個案問題、分析個案執行方法與效益、個案實務的焦點功能領域，以及建立實務指引。
- 第伍章 結論與建議：提出結論與後續研究建議。

2. 文獻探討

本研究主要目的在提出小型軟體公司發展軟體產品的實務指引，因此本章擬針對軟體產品管理範圍及相關的軟體開發管理方式與框架進行探討。

2.1 軟體產品管理的範圍

Christof Ebert（2009）[6]提及，成功的產品管理是指在正確的時間交付正確的產品到正確的市場中。在軟體產品的生命週期中，從策略（Strategy）階段，依序會經過執行研究專案的概念（Concept）階段，執行開發專案的市場進入（Market Entry）與發展（Development）階段，直至執行服務專案的演化（Evolution）階段。其所定義的階段以及如何透過專案來銜接這些階段，描述了軟體產品管理所要關注的範圍。

2.2 敏捷式軟體開發

敏捷式軟體開發（Agile Software Development）主要的精神在於軟體開發的過程中能夠充分溝通與擁抱變革，敏捷式開發聯盟（Agile Alliance）於2002 的敏捷宣言（Agile Manifesto）中，說明了敏捷式開發的價值觀：”個人與互動重於流程與工具、可用的軟體重於詳盡的文件、與客戶合作重於合約協商、回應變化重於遵循計畫”[3][9]，同時也提出了 12 條原則。

- 極限編程（XP，eXtreme Programming）[13][15]為敏捷式軟體開發方法之一，此方法在強調軟體開發過程中的可適應性而非可計畫性，開發團隊要有能力在任何階段去反應改變所帶來的影響，其主要的核心價值為：溝通、簡單、回饋、勇氣與尊重，並將整個流程歸納為 12 項實務的方法，其中包含了雙人式程式設計（Pair Programming）、小型發佈（Small Release）與測試驅動開發（TDD，Test Driven Development）等重要觀念。
- 測試驅動開發（TDD）為 XP 中重要的實務方法之一[14][20]，主要的方式為先撰寫測試程式再進行程式設計與編碼，並配合著自動測試的工具，以測試來驅動軟體的設計與實作，快速取得回饋並反覆執行測試，即早找出軟體錯誤，降低軟體修改的成本。

2.3 能力成熟度整合模式

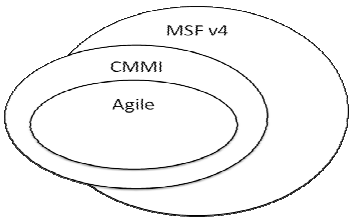
CMMI 是由卡內基美隆大學（Carnegie Mellon University）軟體工程學院（Software Engineering Institute）所提出的方法框架，將流程管理、品質管理應用於軟體發展與維護的過程中，其中定義了 22 個的流程領域（PA，Process Area），並把成熟度分為五個等級，第一級：初始(Initial)、第二級：可重覆(Repeatable)、第三級：已定義(Defined)、第四級：已管理(Managed)、第五級：最佳化(Optimizing)；而 CMMI for Development (CMMI-DEV)則是提供了一個適用於產品與服務的開發和維護之解決方案[5]。

2.4 微軟解決方案框架 v4

微軟解決方案框架[2]是微軟公司依據軟體開發經驗與實務所累積的抽象層框架，提供如何規劃、建立、部署商業導向的技術解決方案，從 1994 年的初版至今日已是第 4 代。

MSF 認為沒有一個管理流程能適用於所有的團隊或是每一個軟體開發專案，因此 MSF 提出了相關的建議與指引，希望每一個團隊或公司發展出屬於他們最合適的軟體發展管理實例，微軟公司的產品 Visual Studio Team System（VSTS）也提供 2 種軟體開發方法的實例：MSF for Agile Software Development 與 MSF for CMMI Process Improvement，此 2 個實例與 MSF 的關係如圖 2-1。

後續小節將陸續介紹 MSF 的基本原則（Foundation Principles）、思維（Mindset）、團隊模型（Team Model）與治理模型（Governance Model）。



資料來源：微軟解決方案框架精要，2007[2]

圖 2-1 MSF 與實作指引關係圖

2.4.1 MSF 基本原則、思維

MSF 的基本原則為：推動開放式的溝通、為共同願景努力、充分授權、明定個人與團隊的職責、漸進式價值交付、保持靈活，預期適應改變、投資品質、從各方經驗學習、與客戶合作等 9 項。每一團隊成員應存在於心中的思維是：培養同儕團隊、專注於商業價值、維持解決方案的大局觀、獲得工作成就感、持續學習、內化服務品質、做個好公民、履行交付承諾。

2.4.2 MSF 的團隊模型

MSF 提出在一個團隊中，區分為 7 種角色，稱之為代言群（Advocacy Groups），包含了：產品管理、計畫管理、架構、程式開發、測試、使用者經驗、發行與作業，每一個代言群可能包含多位成員，並且專注於自己所負責的功能領域，例如：代言群應該要執行測試規劃、測試工程、追蹤與報告等功能領域，透過功能領域中的活動，讓團隊能交付品質良好的解決方案。

雖然 MSF 提及最適合採用 MSF 團隊模型的人數大約是 10-75 人，但也提供小型團隊合併代言群的建議方案，讓 MSF 的適用性更加廣泛。

2.4.3 MSF 的治理模型

MSF 治理模型是將專案治理（Project Government）與流程運行（Process enactment）結合起來，包含 5 個重疊的運行歷程（Enactment Tracks），以及 1 個橫跨各運行歷程的治理歷程。

於 MSF v3 時，以階段（Phase）來表示每一個的時段，但 v4 則將名詞定義改用歷程（Track），主要是因為階段通常為一個個接續的時段，而歷程可以清楚表達出 MSF 每一個時段是重疊的概念。

5 個重疊的運行歷程為：展望（Envision）歷程、規劃（Plan）歷程、建置（Build）歷程、穩定化（Stabilize）歷程與部署（Deploy）歷程，運行歷程與代言群所須負責的焦點功能領域之間的關係，如表 2-1 所示。而治理歷程透過了檢查點與關鍵產出貫穿了每一個運行歷程，其相關項目如表 2-2 所示。

表 2-1　MSF 運行歷程焦點功能領域

歷程	展望歷程	規劃歷程	建置歷程	穩定化歷程	部署歷程
產品管理	・整體目標 ・辨識客戶需求 ・定義遠景/範疇 ・客戶驗收準則	・概念設計 ・商業需求分析 ・溝通計畫 ・排列需求與優先順序	・釐清範疇、需求與利害關係人的期望 ・市場通路與協銷處理	・執行溝通計畫 ・上市規劃 ・決定範疇取捨的優先順序	・蒐集客戶回饋意見、評定 ・每站點完成部署的簽收
計畫管理	・專案結構 ・專案限制因素 ・各項限制因素的取捨	・主專案計畫 ・主專案時程 ・預算	・管理功能規格 ・專案的追蹤 ・計畫的更新	・專案追蹤 ・限制因素和範疇取捨	・部署計畫與時程的更新 ・管理穩定化的相關活動
架構	・設計目標策略 ・概念解決方案 ・可行性分析 ・建置與技術選項	・概念與邏輯設計 ・技術評估 ・初期建置預估 ・功能規格書	・確認架構 ・確認功能規格 ・釐清設計細節	・問題分級	・解決方案/範疇的比較
程式開發	・提出需求與策略	・邏輯與實體設計 ・建置計畫/時程 ・建置預估 ・雛型	・建構解決方案 ・發展基礎建設 ・編寫組態文件	・問題解決 ・解決方案最佳化	・解決問題 ・擴大支援
使用者經驗	・使用者的效能需求與成效 ・使用者驗收準則	・使用情節/案例 ・使用者需求 ・本土化需求 ・可及性需求 ・易用性需求 ・使用手冊 ・教育訓練	・教育訓練 ・易用性測試 ・圖形設計 ・支援工具與使用手冊	・使用者文件穩定化 ・教育訓練教材 ・使用者驗收 ・易用性評估 ・訓練試行使用者 ・作業團隊 ・技術支援團隊	・教育訓練 ・提昇使用者操作能力
測試	・測試策略 ・測試方法 ・測試度量指標	・設計的評估 ・測試需求 ・測試計畫/時程 ・測試情節	・技術/單元測試與少量功能測試 ・辨識問題 ・文件測試 ・更新測試計畫	・功能測試 ・系統測試 ・回歸測試 ・提報問題與狀況 ・組態測試	・問題分級 ・讓部署工作趨於穩定
發行與作業	・部署成效 ・作業管理與支援性 ・服務品質 ・作業驗收準則	・設計的評估 ・作業需求 ・試行與部署計畫/時程 ・定義與建立各種專案環境	・核對檢查表 ・發佈更新與更新試行計畫 ・站點準備檢查表	・試行的設置與支援 ・部署 ・作業與技術支援的準備	・站點部署管理 ・變更的核准

資料來源：微軟解決方案框架精要，2007[2]

表 2-2　MSF 運行歷程的關鍵產出、檢核點

歷程	展望歷程	規劃歷程	建置歷程	穩定化歷程	部署歷程
關鍵產出	・遠景/範疇文件 ・專案結構文件 ・初始的風險評估	・功能規格 ・主專案計畫 ・主專案時程	・行銷文宣、使用者意見交流 ・更新過的主計畫、時程、風險文件 ・已定案的設計、已凍結的功能規格 ・原始程式碼和執行檔、文件、輔助元素 ・使用者參考文件、使用者教育訓練、易用性測試場景 ・測試規格書、測試案例、測試度量指標、測試指令碼、測試資料、測試控管機制 ・部署流程與程序、安裝指令碼與部署組態設定、標準作業程序、服務窗口與支援程序、知識庫、支援小組教育訓練、支援與疑難排解文件	・整合的解決方案元件 ・部署指令碼與安裝說明文件 ・使用者說明文件與教育訓練教材 ・與使用者溝通的機制 ・作業文件 ・測試與問題報告 ・品質度量報告 ・本次發行的相關資訊	・作業與支援的資訊系統 ・修訂的流程與程序 ・使用者與管理員的教育訓練 ・組態文件 ・解決方案的儲存庫
檢核點	・核心團隊已組成 ・遠景/範疇基準已確立 ・遠景範疇已核准	・技術確認完成 ・功能規格基礎已完成 ・主計畫基準已完成 ・主計畫時程基準已完成 ・支援環境已完成 ・專案計畫已核准	・雛型設計完成 ・內部 n 個版本的解決方案發行 ・範疇已確立	・1- n 次功能測試完成 ・問題收斂 ・使用者介面已穩定 ・問題紀錄已清除 ・上線前測試已完成 ・1- n 版發行候選已完成 ・系統測試完成 ・使用者驗收測試完成 ・試行完成 ・發行準備已核准	・解決方案核心元件部署完成 ・站點部署完成 ・部署作業已穩定 ・部署完成

資料來源：微軟解決方案框架精要，2007[2]

2.5　小結

軟體產品管理的範疇包含了許多功能領域，在目前的軟體開發管理方式與框架中，對於小型軟體公司發展軟體產品而言，MSF 是最能滿足需求也是可執行性最高的管理框架，因為 MSF 的主旨即是希望不同型態的公司能夠參考 MSF 而訂定出一套適合於自己公司的軟體發展機制，且涵蓋的範圍包含了產品的評估與規劃、更新與部署，因此本研究選用 MSF 作為主要的理論基礎，建構出小型軟體公司發展軟體產品的實務指引。

3. 研究設計

3.1　研究方法

本研究主要是針對單一公司的軟體產品發展過程進行深度的分析，旨在探究小型軟體公司於產品的發展過程中為何會遇到某些問題，這些問題又是如何被解決，進而歸納出一套實務指引。依據 Robert K. Yin （1994）[17]所整理的多種社會科學研究方法中，本研究使用單一類型的個案研究法作為研究方法。

Yin 同時提出將個案研究所需要的研究證據（Evidence）分為 6 類，進行個案研究時，研究資料應同時涵蓋多各種類，並將證據之間建立關聯。

3.2　研究對象

本研究對象 S 公司，為一間小型的商用軟體公司。本研究透過 S 公司的溝通信件、流程圖、會議紀錄、開發文件、需求與問題管理系統、客戶管理系統、實際產出等項目，取得研究資料，並將相關資料進行三角檢定，研究項目與研究資料之間的關聯如表 3-1 所示。

表 3-1　研究項目與研究資料之關聯

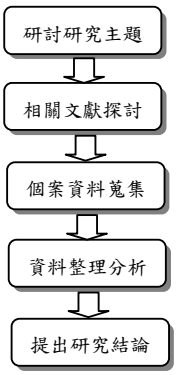
研究項目	研究資料
個案所遭遇的問題	電子郵件、會議紀錄、開發規劃時程與工作項目、需求與問題管理系統、客戶管理系統
個案的執行方法與效益	電子郵件、會議紀錄、測試紀錄、工作紀錄、軟體產品、客戶管理系統
個案的開發工作項目	電子郵件、會議紀錄、開發流程圖、開發規劃時程與工作項目、測試紀錄
個案開發流程	電子郵件、會議紀錄、開發流程圖

資料來源：本研究整理

3.3　研究流程

本研究流程在研究主題確認後，進行了軟體產

品管理、軟體開發模式等相關領域的文獻探討，並開始蒐集研究個案資料及進行資料的整理分析，其中包含了：彙整個案所遭遇的問題、分析個案解決問題的執行方法、比較個案與 MSF 在團隊與治理模型上的差異，最後提出了小型軟體公司的軟體產品發展實務指引作為研究結論，並說明實務指引所能達到的效益。本研究流程如圖 3-1 所示。



資料來源：本研究整理

圖 3-1　研究流程圖

4. 個案研究與分析

4.1　個案描述

4.1.1　個案背景

本研究個案 S 公司，成立於民國 94 年 7 月，是一個專注於軟體產品開發的公司，90%的公司成員為專職技術人員。S 公司成立迄今，透過軟體產品的銷售與維護，均能夠讓營業額維持逐年穩定的成長，並且持續獲利，公司人數也由 2 人擴展至 10 餘人，S 公司產品發展團隊成員人數與所花費的開發人月數如表 4-1。

表 4-1　S 公司產品發展花費人月統計表

年度	團隊成員數	開發人月數
第一年	2 – 3 人	30 人月
第二年	3 人	36 人月
第三年	4 人	48 人月
第四年	5 – 7 人	72 人月
第五年	7 人	84 人月
第六年	8 – 10 人	108 人月
總計		378 人月

資料來源：本研究整理

S 公司所研發 W 產品，主要是針對企業內部 IT 資源管理的商業解決方案，其中包含了資產盤點、端末資訊安全、軟體授權管理、IP 管理、遠端遙控等領域，歷經 v1、v2、v3 等主要版本的發展（W 產品各版本主要功能如表 4-2），目前已累積了 100 家以上的中大型客戶，客戶產業別橫跨了金融業、製造業、流通業以及政府機關。

在 W 產品的發展過程中，累積了許多錯誤與失敗的嘗試，軟體產品發展管理的機制也不斷地修正調整，因此本研究以 S 公司的實務經驗作為主要的研究資料，而本論文的作者為 S 公司之創始成員，負責軟體產品的研發與管理，完整參與了 W 產品從 v1 至 v3 的開發過程。

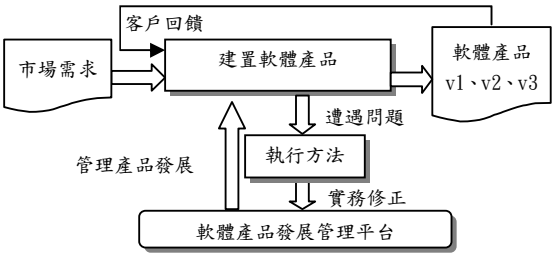
表 4-2　W 產品各版本主要新增功能項目

版本	產品主要新增功能項目
v1 2005.07 ~ 2006.07	・基本功能模組：機台資訊管理、遠端遙控、軟碟機與卸除式磁碟管制、軟體資產管理、排程管理、稽核通知機制、AD 組織/人員匯入精靈、列印匯出機制 ・固定資產生命週期管理模組 ・支援伺服器覆載平衡機制 ・安裝程式與升級程式
v2 2006.08 ~ 2008.09	・基本功能模組：網頁瀏覽管制、軟體使用率統計、IP 網段掃描、支援多國語言 ・固定資產生命週期管理模組：條碼列印與掃描、盤點作業、資產關帳作業 ・周邊安全管理模組 ・軟體安全模組
v3 2008.10 ~	・基本功能模組：Agent 原則管理、遠端檔案搜尋、光碟機管制。 ・周邊安全管理模組：禁用介面、紅外線裝置、藍芽裝置、3.5G 網卡管制 ・IP 管理模組

資料來源：本研究整理

4.1.2　個案軟體產品發展管理機制

S 公司以市場需求和客戶回饋的意見為輸入，經過內部軟體產品發展平台的管理機制後，產出各版本的 W 產品，在產品發展過程中遭遇了許多開發與管理上的問題，S 公司針對這些問題產生了對應的執行方法，並將執行方法的結果回饋至軟體產品發展管理平台中，整體機制運行示意如圖 4-1。



資料來源：本研究整理

圖 4-1 S 公司軟體產品發展管理機制

4.2 個案所遭遇的問題

本節對個案軟體產品發展過程 6 年來所經歷的錯誤嘗試與經驗累積，依循 MSF 框架將其歸納為 3 個主要的問題領域：團隊模型的問題 Team Model Problems（TP）、治理模型的問題 Governance Model Problems（GP）、開發與技術的問題 Develop and Tech Problems（DP），並針對每一個問題的進行編號與說明。

4.2.1 團隊模型的問題

1. TP-1：主管（如：總經理、研發主管）所負責的功能領域主導了軟體產品的走向，造成軟體產品的偏頗，影響產品長期發展。由於個案採用階層式組織架構，團隊成員通常會遵循主管所提出的意見，因而喪失了各功能領域的平衡機制，例如：團隊主管為技術背景時，可能偏重於架構或程式開發領域，而忽略了使用者經驗的重要性。
2. TP-2：未區分產品開發團隊與作業團隊，造成開發進度落後與產品設計偏頗特定客戶的需求。當開發團隊與作業團隊相同時，開發工作經常會被繁瑣的作業問題所干擾，同時開發團隊為提高自己所負責客戶的滿意度，在設計產品功能時，往往欠缺全面性的考量。
3. TP-3：開發人員兼任測試工作，產品問題無法被偵測。由於團隊成員人數受限於有限的資源，部份開發工作與測試工作會由同一人進行，在產品開發時程的壓力下，以及設計者的盲點之故，往往會形成重開發而輕測試的情況，許多產品問題無法在開發與測試時期被發現，造成後續問題修正的成本高昂。

4.2.2 治理模型的問題

1. GP-1：缺乏需求蒐集與分析的機制，客戶端所反應的需求常出現遺漏或是無法滿足的情況。S 公司在有限資源的狀況下，通常在客戶需求非常明確時，才會將此需求轉換為需要開發的功能規格，例如：多位客戶提及同一項功能需求，由於欠缺系統化的管理，許多需求與使用情境在實作時，無法對應當初的需求來源，因此會造成某些需求被忽略，或是產品功能無法實際滿足客戶需求的現象。
2. GP-2：產品規劃時程與實際發佈時程差異過大，無法有效管控資源的使用情形。此問題發生的主因有二項，第一項為產品初期的程式臭蟲數（Bugs）過多，開發團隊忙於修復產品問題，無法確實執行開發計畫，造成了產品時程壓力、忽略測試、修正問題、影響開發計畫的惡性循環；第二項為某些系統底層技術掌握度不足，在實作時會出現技術瓶頸，而技術瓶頸的突破，在時程上通常很難預估。
3. GP-3：缺乏可信賴的產品測試機制，無法確保產品品質。W 產品在 v1、v2 開發過程中，缺乏明確定義的測試流程與測試環境，在沒有測試基準的狀態下，品質自然無法掌控，且受限於測試環境的完整性，產品雖已通過內部測試，但部署至客戶端環境時，仍然可能會發生問題，造成作業團隊沈重的負擔。

4.2.3 開發與技術的問題

1. DP-1：採用了不成熟的新技術，同時使用多種開發語言。個案在 W 產品開發初期，選擇了.Net Framework v1.0 作為開發平台，在 W 產品 v1 尚未完成時，微軟即發佈了.Net Framework v1.1，但由於.Net v1.0 與 v1.1 在遠端呼叫機制上並未完全相容，因此 W 產品被迫修改程式碼，讓產品發佈時間延後了約 3 個月。另外，W 產品在發展初期同時選用了 C#與 J#二種程式語言，造成維護與除錯上的困難，導致 W 產品演進至 v2 時，程式碼的轉換過程至少多花費了 6 個人月以上。
2. DP-2：並未完整驗證第三方元件即採用於產品之中。因為考量時程與當下所掌握的技術能力，W 產品在介面呈現與網路通訊上使用了第三方的元件，但初期採用的介面呈現元件由於功能不足，網路通訊元件也不夠穩定，進而影響了 W 產品的整體品質，因此在 W 產品演進過程中皆進行了更換，因為更換第三方元件所付出的重工時間大於 4 個人月，同時產品除錯的過程估計至少花費了 3 個人月。
3. DP-3：部份技術委外，減緩了核心能力的發展速度。由於資源的限制，W 產品於 v1、v2 的某些關鍵技術採用委外開發模式，但由於委外廠商的品質不容易掌握，程式除錯過程冗長且無法一次解決問題，因而造成客戶對於 W 產品的信任度下降，S 公司要維護與客戶之間的關係，付出了許多額外的服務成本。

4.3 個案執行方法與效益

個案針對軟體產品發展過程中的問題，產生了對應的執行方法，隨著 W 產品的版本演進與經驗累積，讓執行方法的完成度提昇，相關資訊整理如表 4-3，表中將執行方法予以編號為 E-1 到 E-15，用以解決 4.2 小節中所描述的 9 項問題。各版本完成度欄位，是依據研究資料，概略估計該項執行方法在此版本發展中的實施完整程度，並以整數數值

的方式呈現，0%表示此項執行方法尚未實施，100%表示已確實完整執行。而效益欄位，意指消弭 W 產品在 v1 至 v3 完成度的差異，所能產生的效益，如：節省開發人月、提昇產品品質。

表 4-3　執行方法完成度與效益

問題編號	方法編號	執行方法	完成度(%)			執行方法效益
			v1	v2	v3	
TP-1	E-1	主管授權團隊成員可以決定自己所代言的功能領域中，必須要達成的規格標準。在個案中，此方案實施時機是在團隊成員已經累積了足夠的經驗，能夠獨立作業，並清楚明白負責領域的責任。	0	30	70	‧提高產品完整性。
TP-2	E-2	區分開發團隊與作業團隊。作業團隊負責產品的部署與更新，以及客戶使用問題的技術支援，讓開發團隊能專注產品開發。	0	30	70	‧提昇團隊工作效率。
TP-3	E-3	將測試團隊獨立或是以工作時間區分程式開發和測試工作。個案中由於受限於人力，因此採用工作時間區分的方式，定義某一段時間內，此團隊成員只負責執行測試工作，直至完成所有測試工作後才能轉換至程式開發工作。	0	20	60	‧提昇產品品質。 ‧提昇團隊工作效率。
GP-1	E-4	建立資訊系統管理客戶需求與問題。	10	50	100	‧有效管理產品需求與問題。
	E-5	透過多位客戶的深入訪談以及雛型展示的方式，確認客戶端需求。	10	60	80	‧確認開發功能符合市場預期，避免開發出無市場需求的功能項目。個案可節省 8 個人月的程式開發時間。 ‧避免因為需求不明確所造成的重工與程式碼的反覆調整，總計節省程式開發人力約 10 個人月。
	E-6	參考業界標竿產品的功能項目，與客戶需求進行交叉比較。	50	60	70	‧確認開發功能符合市場預期。
GP-2	E-7	在訂定開發規格時，只設定主要的功能項目，並保留獨立的時間修正優先權較高的產品問題，讓開發計畫可確實達成。	0	0	100	‧降低開發風險。 ‧版本開發時程誤差由 40%降低為 20%。
	E-8	需求在納入產品開發規格前，進行技術與可行性評估，降低系統實作時的風險。	0	0	70	
GP-3	E-9	建立標準的測試環境，作為測試基準。其中測試項目包含：不同的作業系統、不同的語言版本、不同的客戶資料庫。	10	50	90	‧提昇產品品質。 ‧減少產品瑕疵數，降低產品更新頻率，由 1-2 月更新一次調整為 2-4 個月更新一次。因為更新次數的降低，可節省安裝與更新程式的開發人力 2 個人月。每次版本發佈所須的測試人力 10 個人月。協助客戶端產品升級的作業人力 12 個人月。
	E-10	定義標準測試流程，產品發佈前，須經過單元測試、功能測試、整合測試、壓力測試、相容性測試等必要的測試流程。	10	50	90	
	E-11	執行產品公開測試，選定少數客戶進行新版本的試行，補強測試環境與實際環境之間的差異。	20	30	70	
	E-12	採用測試驅動的開發模式，開發前先撰寫測試案例並配合自動化測試工具，提昇產品品質。	30	80	80	
DP-1	E-13	產品發展在選擇開發技術時，以技術成熟度作為主要考量，並且統一開發語言，降低維護成本。	50	80	90	‧減少程式碼重新撰寫（統一為同一種開發語言）與除錯所花費的時間，節省程式開發人力 6 個人月。
DP-2	E-14	制定第三方元件使用前的評估流程，確認該元件符合功能需求，並撰寫測試程式實際驗證，確保元件的穩定性。	20	50	80	‧減少重工（變更使用的第三方元件），節省程式開發人力 7 個人月。 ‧提昇產品穩定度與品質，節省因為協助客戶端產品升級或是問題排除所需要的作業人力 6 個人月。
DP-3	E-15	技術委外開發時，需要擁有原始程式碼，並取得技術移轉。	0	30	100	‧提昇核心技術。 ‧提高產品穩定度。

資料來源：本研究整理

4.3.1 團隊模型問題，執行方法 E-2：

產品 v1 時期，由於 S 公司尚處於草創初期，開發、作業、銷售等工作都是交由同一組團隊執行，因此執行方法完成度給予 0%；產品 v2 時期，為了因應產品的市場開拓與需求日漸增加，加入了數位程式開發人員，新進成員專注於產品程式開發的工作，而資深人員仍同時身兼開發與作業的工作，因此執行方法完成度給予 30%；產品 v3 時期，由於已略具市場規模，在人力資源配置上新增了專職的作業人員，負責處理 W 產品的導入與技術支援，但由於 W 產品具有高複雜度的特性，在進行大型客戶導入專案時，會需要資深開發人員的協助，因此給予 70%的執行方法完成度。

4.3.2 治理模型問題，執行方法 E-4：

產品 v1 時期，客戶的需求或問題是以口頭或電子郵件的方式告知整個團隊，沒有指定的負責人員進行系統化的整理，因此執行方法完成度給予 10%；產品 v2 時期，雖然已有產品經理負責將需求與問題記錄於 Excel 檔案中，但有著查詢不易以及無法有效結構化的問題，因此執行方法完成度給予 50%；產品 v3 時期，建立了網頁化的需求追蹤系統，清楚記錄需求或問題的來源，以及處理狀態，並定期召開追蹤會議，因此給予 100%的執行方法完成度。

4.3.3 開發與技術問題，執行方法 E-15

產品 v1 時期，W 產品的某項重要功能是使用委外廠商既有的程式，S 公司無法得知此程式內部的運作邏輯，在新功能開發或問題除錯上受制於委外廠商，因此執行方法完成度給予 0%；產品 v2 時期，更換了新的委外廠商，配合模式更改為由 S 公司定義功能與資料格式，但由於無法掌握核心技術能力，在產品發展上仍然有限制，因此執行方法完成度給予 30%；在產品 v3 時期，S 公司改採委外研究而非委外開發的方式，將開發過程中的某些不確定的技術交付給外部廠商研究，外部廠商需要將研究結果技術移轉給 S 公司，並且提供可執行的實作範例程式碼，至此，S 公司充分掌握核心技術，因此給予 100%的執行方法完成度。

4.4 個案實務的焦點功能領域

本節將對個案在 W 產品 v3 開發時的實務經驗與 MSF 框架中的治理模型：運行歷程、關鍵產出以及檢核點進行差異分析。個案為小型的軟體公司，因此在開發管理的機制上，會著重考量於資源的稀少性、限制條件、功能領域的必要性、功能領域的優先順序等因素，當某些功能領域焦點不是產品開發的必要條件時，會予以調整或是不採用，並且新增實務上需要關注的焦點。以下就每一個歷程中的焦點功能領域、關鍵產出與檢核點進行比較，並彙整個案實務中未採用的功能領域項目，說明可能會造成的風險與影響。

4.4.1 展望歷程

展望歷程中，個案實務與 MSF 建議項目的差異比較如表 4-4，依照不同的功能領域焦點，說明如下：

- 產品管理：由於 MSF 適用於任何的軟體解決方案之發展，而在個案中，特別將焦點聚集於產品而非專案，基於產品發展的基本特性，在辨識客戶需求之前，應該先設定目標市場，才能讓軟體產品符合商業目標與市場需求。
- 計畫管理：以小型軟體公司而言，開發成員與利害關係人多已固定，因此對於專案結構的關注，予以刪除。
- 架構：個案實務與 MSF 一致。
- 程式開發：個案中程式開發所扮演的角色大多都是單純的技術實作者，因此在歷程中，只專注於如何規劃或取得程式開發所需要的開發環境。
- 使用者經驗：以產品而言，所要滿足的使用者就是產品初期所設定的目標客戶，因此使用者驗收條件等同於產品需求功能，此項目予以刪除。另外，在此歷程中，除了以使用者角度提出效能需求外，同時也一併考量其他非功能性的需求，例如：安全性、可用性。
- 測試：除了原先須關注的焦點領域外，個案增加了如何規劃與取得所需要的測試環境。
- 發行與作業：以產品而言，作業團隊較難評量部署成效，而單純是執行新舊客戶的安裝與升級，因此予以刪除。
- 關鍵產出：如同計畫管理中所討論的無須關注於專案結構，因此不需要產出專案結構文件。在確認目標市場與需求後，應該以的高階需求文件，作為後續作業的依據。為降低產品的開發風險，除了初步的風險文件外，最重要的是評估需求在技術面、商業面與市場面的可行性，並將評估結果文件化。
- 檢核點：如同計畫管理的差異所述，小型軟體公司成員固定，因此核心團隊已建立無須是一個檢查點。而產品的目標市場與需求的確認，對後續發展的影響甚鉅，因此將這二項目訂為檢查點，由核心團隊成員共同確認。

表 4-4　展望歷程差異比較表

	MSF 框架建議	個案實務
產品管理	M1.整體目標 M2.辨識客戶需求 M3.定義遠景/範疇 M4.客戶驗收準則	m1.整體產品目標 m2.辨識目標市場與客戶需求 m3.定義產品遠景/範疇 M4.客戶驗收準則
計畫管理	M1.專案結構 M2.專案限制因素 M3.各項限制因素的取捨	~~M1.專案結構~~ M2.專案限制因素 M3.各項限制因素的取捨
架構	M1.設計目標策略 M2.概念解決方案 M3.可行性分析 M4.建置與技術選項	M1.設計目標策略 M2.概念解決方案 M3.可行性分析 M4.建置與技術選項
程式開發	M1.提出需求與策略	M1.提出需求與策略 s2. 規劃開發環境
使用者經驗	M1.使用者的效能需求與成效 M2.使用者驗收準則	m1.提出使用者的非功能性需求 ~~M2.使用者驗收準則~~
測試	M1.測試策略 M2.測試方法 M3.測試度量指標	M1.測試策略 M2.測試方法 M3.測試度量指標 s4. 規劃測試環境
發行與作業	M1.部署成效 M2.作業管理與支援性 M3.服務品質 M4.作業驗收準則	~~M1.部署成效~~ M2.作業管理與支援性 M3.服務品質 M4.作業驗收準則
關鍵產出	M1.遠景/範疇文件 M2.專案結構文件 M3.初始的風險評估	m1.產品遠景/範疇文件 M2.專案結構文件 M3.初始的風險評估 s4. 需求說明文件 s5. 需求評估結果
檢核點	M1.核心團隊已組成 M2.遠景/範疇基準已確立 M3.遠景範疇已核准	~~M1.核心團隊已組成~~ s2. 產品目標已確認 m3.產品遠景/範疇基準已確立 s4. 已初步評估產品需求 M5.遠景範疇已核准

*雙刪除線：未採用、m：修改、s：新增項目

資料來源：本研究整理

4.4.2 規劃歷程

規劃歷程中，個案實務與 MSF 建議項目的差異比較如表 4-5，依照不同的功能領域焦點，說明如下：

- 產品管理：個案實務與 MSF 一致。
- 計畫管理：以小型軟體公司而言，軟體產品發展中最大的成本為人力成本，當專案時程與人力資源配置確認後，個案並未花費額外時間進行預算規劃的活動。
- 架構：技術端可行性評估已在展望歷程執行，因此在規劃歷程中，針對技術細節進行評估，例如：建立出可執行的範例程式。

表 4-5 規劃歷程差異比較表

	MSF 框架建議	個案實務
產品管理	M1.概念設計 M2.商業需求分析 M3.溝通計畫 M4.排列需求與優先順序	M1.概念設計 M2.商業需求分析 M3.溝通計畫 M4.排列需求與優先順序
計畫管理	M1.主專案計畫 M2.主專案時程 M3.預算	M1.主專案計畫 M2.主專案時程 ~~M3.預算~~
架構	M1.概念與邏輯設計 M2.技術評估 M3.初期建置預估 M4.功能規格書	M1.概念與邏輯設計 m2 細部技術評估 M3.初期建置預估 M4.功能規格書
程式開發	M1.邏輯與實體設計 M2.建置計畫/時程 M3.建置預估 M4.雛型	M1.邏輯與實體設計 M2.建置計畫/時程 M3.建置預估 M4.雛型 s5. 開發環境就緒
使用者經驗	M1.使用情節/案例 M2.使用者需求 M3.本土化需求 M4.可及性需求 M5.易用性需求 M6.使用手冊 M7.教育訓練	M1.使用情節/案例 M2.使用者需求 m3.多語言需求 ~~M4.可及性需求~~ M5.易用性需求 ~~M6.使用手冊~~ ~~M7.教育訓練~~
測試	M1.設計的評估 M2.測試需求 M3.測試計畫/時程 M4.測試情節	M1.設計的評估 M2.測試需求 M3.測試計畫/時程 M4.測試情節 s5. 測試環境就緒
發行與作業	M1.設計的評估 M2.作業需求 M3.試行與部署計畫/時程 M4.定義與建立各種專案環境	~~M1.設計的評估~~ M2.作業需求 M3.試行與部署計畫/時程 ~~M4.定義與建立各種專案環境~~
關鍵產出	M1.功能規格 M2.主專案計畫 M3.主專案時程	M1.功能規格 M2.主專案計畫 M3.主專案時程 s4. 系統設計文件與雛型 s5. 主要測試案例
檢核點	M1.技術確認完成 M2.功能規格基礎已完成 M3.主計畫基準已完成 M4.主計畫時程基準已完成 M5.支援環境已完成 M6.專案計畫已核准	M1.技術確認完成 M2.功能規格基礎已完成 M3.主計畫基準已完成 M4.主計畫時程基準已完成 M5.支援環境已完成 M6.專案計畫已核准

*雙刪除線：未採用、m：修改、s：新增項目

資料來源：本研究整理

- 程式開發：在建置歷程之前，須確認所需要的開發環境（硬體、開發工具、開發規範等）已就緒。
- 使用者經驗：使用多語言需求代替本土化需求，因為本土化需求除了文字調整之外，同時須考慮文化背景與操作習慣，就小型軟體公司而言，已經超過可以負擔的範圍。另外在產品的目標市場確立後，便可以排除可及性需求，主要是希望在有限資源的狀態下，發展擁有最大效益的需求項目。使用者手冊與教育訓練的議題，則移至穩定化歷程中一併處理。
- 測試：如同程式開發團隊一樣，在建置歷程之前，必須確認所需要的測試環境（硬體、網路環境、各式作業系統、自動測試工具等）已就緒。
- 發行與作業：個案中的設計評估是由產品管理、架構與程式開發共同執行，非作業團隊所須關注的項目，因此予以刪除。作業團隊此時應該以選定試行對象替代建立各種專案環境，可節省整體的發展成本。
- 關鍵產出：經過架構與程式開發團隊的工作，應產出系統設計文件與雛型，同時測試團隊也應該針對功能規格，進行主要測試情境/案例的撰寫。
- 檢核點：個案實務與 MSF 一致。

4.4.3 建置歷程

建置歷程中，個案實務與 MSF 建議項目的差異比較如表 4-6，依照不同的功能領域焦點，說明如下：

- 產品管理：產品開發的主要利害關係人為公司經營階層、開發團隊、目標市場客戶以及當初需求的提供者，利害關係人的期望應在展望歷程就已經確認了，因此予以刪除。後續進行公開測試（Beta 版本）與部署時，使用者需要得知產品的功能說明與效益，因此在此階段應開始進行產品文件的產出。
- 計畫管理：個案實務與 MSF 一致。
- 架構：個案實務與 MSF 一致。
- 程式開發：因為個案在開發過程中納入了測試驅動開發（TDD）的概念，程式開發人員應對所開發的功能進行單元測試，並先撰寫測試程式。各功能模組間如果有整合項目，會在此階段進行介面測試，確認各功能模組可以順利界接，降低開發風險。
- 使用者經驗：在資源有限的狀況下，使用者經驗代言人由產品經理兼任，因此個案將教育訓練與使用者手冊等焦點領域，移至穩定化歷程再進行，可減少人力資源的使用。
- 測試：除了功能測試外，此階段應專注於高風險項目整合測試。
- 發行與作業：專注於產品更新與更新試行計畫。
- 關鍵產出：如使用者經驗功能領域的調整項目所述，為節省人力資源，教育訓練等項目於穩定化歷程再開始進行即可。在此歷程中，開發的功能應採用逐步交付的方式，內部人員將協助產品的功能測試與確認，因此會產出多個內部測試的 Alpha 版本。
- 檢核點： 此階段主要檢核點應以功能規格的角度，完成所有產品新增、修改項目的功能測試。

表 4-6 建置歷程差異比較表

	MSF 框架建議	個案實務
產品管理	M1. 釐清範疇、需求、與利害關係人的期望 M2. 市場通路與協銷處理	~~M1. 釐清範疇、需求、與利害關係人的期望~~ M2. 市場通路與協銷處理 s3. 建立產品文件
計畫管理	M1. 管理功能規格 M2. 專案的追蹤 M3. 計畫的更新	M1. 管理功能規格 M2. 專案的追蹤 M3. 計畫的更新
架構	M1. 確認架構 M2. 確認功能規格 M3. 釐清設計細節	M1. 確認架構 M2. 確認功能規格 M3. 釐清設計細節
程式開發	M1. 建構解決方案 M2. 發展基礎建設 M3. 編寫組態文件	M1. 建構解決方案 M2. 發展基礎建設 M3. 編寫組態文件 s4. 單元測試 s5. 系統介面測試
使用者經驗	M1. 教育訓練 M2. 易用性測試 M3. 圖形設計 M4. 支援工具與使用手冊	~~M1. 教育訓練~~ M2. 易用性測試 M3. 圖形設計 ~~M4. 支援工具與使用手冊~~
測試	M1. 技術/單元測試與少量功能測試 M2. 辨識問題 M3. 文件測試 M4. 更新測試計畫	m1. 功能測試 M2. 辨識問題 ~~M3. 文件測試~~ M4. 更新測試計畫 s5. 部份整合測試
發行與作業	M1. 核對檢查表 M2. 發布更新與更新試行計畫 M3. 站點準備檢查表	~~M1. 核對檢查表~~ M2. 發布更新與更新試行計畫 ~~M3. 站點準備檢查表~~
關鍵產出	M1. 行銷文宣、使用者意見交流 M2. 更新的主計畫、時程、風險文件 M3. 已定案的設計、已凍結的功能規格 M4. 原始程式碼和執行檔、文件、輔助元素 M5. 使用者參考文	M1. 行銷文宣、使用者意見交流 M2. 更新的主計畫、時程、風險文件 M3. 已定案的設計、已凍結的功能規格 M4. 原始程式碼和執行檔、文件、輔助元素 ~~M5. 使用者參考文~~

	件、使用者教育訓練、易用性測試場景 M6.測試規格書、測試案例、測試度量指標、測試指令碼、測試資料、測試控管機制 M7.部署流程與程序、安裝指令碼與部署組態設定、標準作業程序、服務窗口與支援程序、知識庫、支援小組的教育訓練、支援與疑難排解文件	~~件、使用者教育訓練、易用性測試場景~~ M6.測試規格書、測試案例、測試度量指標、測試指令碼、測試資料、測試控管機制 ~~M7.部署流程與程序、安裝指令碼與部署組態設定、標準作業程序、服務窗口與支援程序、知識庫、支援小組的教育訓練、支援與疑難排解文件~~ s8. Alpha 版本 1-n
檢核點	M1.雛型設計完成 M2.內部 n 解決方案發行 M3.範疇已確立	M1.雛型設計完成 M2.內部 n 解決方案發行 M3.範疇已確立 s4.通過所有功能測試

*雙刪除線：未採用、m：修改、s：新增項目

資料來源：本研究整理

4.4.4 穩定化歷程

穩定化歷程中，個案實務與 MSF 建議項目的差異比較如表 4-7，依照不同的功能領域焦點，說明如下：

- 產品管理：個案實務與 MSF 一致。
- 計畫管理：個案實務與 MSF 一致。
- 架構：個案實務與 MSF 一致。
- 程式開發：程式開發人員應提供完整的版本安裝升級程式，版本升級的過程與相容性也會再列入主要的測試項目中；而 Beta 版本將有外部使用者進行測試，因此有別於 Alpha 版本，產品安裝與升級都應該依照標準的流程。
- 使用者經驗：此階段專注於教育訓練的教材與執行，以及使用者手冊的撰寫。
- 測試：由於產品新增、修改項目的功能測試在建置歷程便已經完成，因此測試人員應以使用者的角度進行產品功能的操作與系統整合測試，確認系統可以符合產品的需求規格。
- 發行與作業： 產品的部署包含新產品安裝與舊有版本升級，因此將此項目作調整。
- 關鍵產出：以各項測試結果作為品質衡量依據，因此不會另行產出品質度量報告。穩定化期間應會產出多個 Beta 版本。
- 檢核點：個案實務進行上，通常會以後期的 Beta 版本，將除錯資訊移除後，直接轉換為發行候選版本，可節省重複建置與版本管理的負擔。個案實務中的上線測試與使用者驗收測試，都在部署歷程進行，因此予以刪除。

表 4-7 穩定化歷程差異比較表

	MSF 框架建議	個案實務
產品管理	M1.執行溝通計畫 M2.上市規劃 M3.決定範疇取捨的優先順序	M1.執行溝通計畫 M2.上市規劃 M3.決定範疇取捨的優先順序
計畫管理	M1.專案追蹤 M2.限制因素和範疇取捨	M1.專案追蹤 M2.限制因素和範疇取捨
架構	M1.問題分級	M1.問題分級
程式開發	M1.問題解決 M2.解決方案最佳化	M1.問題解決 M2.解決方案最佳化 s3. 建構安裝升級程式
使用者經驗	M1.使用者文件穩定化 M2.教育訓練教材 M3.使用者驗收 M4.易用性評估 M5.訓練試行使用者 M6.作業團隊 M7.技術支援團隊	M1.使用者文件穩定化 M2.教育訓練教材 ~~M3.使用者驗收~~ ~~M4.易用性評估~~ M5.訓練試行使用者 ~~M6.作業團隊~~ ~~M7.技術支援團隊~~
測試	M1.功能測試 M2.系統測試 M3.回歸測試 M4.提報問題與狀況 M5.組態測試	m1.以使用者角度執行產品功能測試 m2.系統整合測試 M3.回歸測試 M4.提報問題與狀況 M5.組態測試
發行與作業	M1.試行的設置與支援 M2.部署 M3.作業與技術支援的準備	M1.試行的設置與支援 m2.產品安裝與升級 M3.作業與技術支援的準備
關鍵產出	M1.整合的解決方案元件 M2.部署指令碼與安裝說明文件 M3.使用者說明文件與教育訓練教材 M4.與使用者溝通機制 M5.作業文件 M6.測試與問題報告 M7.品質度量報告 M8.本次發行相關資訊	M1.整合的解決方案元件 M2.部署指令碼與安裝說明文件 M3.使用者說明文件與教育訓練教材 M4.與使用者溝通機制 M5.作業文件 M6.測試與問題報告 ~~M7.品質度量報告~~ M8.本次發行相關資訊 s9. Beta 版本 1-n
檢核點	M1.1- n 次功能測試完成 M2.問題收斂 M3.使用者介面已穩定 M4.問題紀錄已清除 M5.上線前測試已完成 M6.1- n 版發行候選已完成 M7.系統測試完成 M8.使用者驗收測試完成 M9.試行完成 M10. 發行準備已核准	m1.1- n 次回歸測試完成 M2.問題收斂 M3.使用者介面已穩定 M4.問題紀錄已清除 ~~M5.上線前測試已完成~~ ~~M6.1- n 版發行候選已完成~~ m7.整合測試完成 ~~M8.使用者驗收測試完成~~ M9.試行完成 M10.發行準備已核准

*雙刪除線：未採用、m：修改、s：新增項目

資料來源：本研究整理

4.4.5 部署歷程

部署歷程中，個案實務與 MSF 建議項目的差異比較表 4-8，依照不同的功能領域焦點，說明如下：

表 4-8　部署歷程差異比較表

	MSF 框架建議	個案實務
產品管理	M1.蒐集客戶回饋意見、評定 M2.每站點完成部署的簽收	M1.蒐集客戶回饋意見、評定 ~~M2.每站點完成部署的簽收~~
計畫管理	M1.部署計畫與時程的更新 M2.管理穩定化的相關活動	~~M1.部署計畫與時程的更新~~ M2.管理穩定化的相關活動
架構	M1.解決方案/範疇的比較	M1.解決方案/範疇的比較
程式開發	M1.解決問題 M2.擴大支援	M1.解決問題 m2.第二線支援
使用者經驗	M1.教育訓練 M2.提昇使用者操作能力	M1.教育訓練 M2.提昇使用者操作能力
測試	M1.問題分級 M2.讓部署工作趨於穩定	~~M1.問題分級~~ ~~M2.讓部署工作趨於穩定~~
發行與作業	M1.站點部署管理 M2.變更的核准	m1.客戶端的部署管理 M2.變更的核准
關鍵產出	M1.作業與支援的資訊系統 M2.修訂的流程與程序 M3.使用者與管理員的教育訓練 M4.組態文件 M5.解決方案儲存庫	~~M1.作業與支援的資訊系統~~ M2.修訂的流程與程序 M3.使用者與管理員的教育訓練 M4.組態文件 M5.解決方案儲存庫 s6. 驗收紀錄與工作紀錄
檢核點	M1.解決方案核心元件部署完成 M2.站點部署完成 M3.部署作業已穩定 M4.部署完成	~~M1.解決方案核心元件部署完成~~ ~~M2.站點部署完成~~ M3.部署作業已穩定 m4.主要客戶部署完成 s5. 完成既有客戶的更新通知或部署

*雙刪除線：未採用 、m：修改、s：新增項目

資料來源：本研究整理

- 產品管理：產品管理專注於產品意見的蒐集，部署簽收工作改由作業團隊進行。
- 計畫管理：軟體產品的安裝與更新已有既定之流程，因此無須訂立特別的部署計畫。
- 架構：個案實務與 MSF 一致。
- 程式開發：個案中的軟體產品程式開發人員在支援工作上為第二線，讓產品問題處理盡量標準化，作業團隊需要有能力處理產品的相關問題。
- 使用者經驗：個案實務與 MSF 一致。
- 測試：測試人員在穩定化歷程已經完成的全部工作，如部署階段出現問題，作業團隊應直接反應給產品管理人員，納入產品的需求或問題管理中。
- 發行與作業：定義為客戶端的部署管理，包含了部署驗收、撰寫工作紀錄、問題蒐集與排除。
- 關鍵產出：如作業支援工作並未太過複雜，則不需要資訊系統的輔助。每一個客戶安裝了此版本的軟體產品，應該要留下驗收紀錄或是版本升級的工作紀錄。
- 檢核點：檢核的重點應放在客戶是否順利完成產品的安裝或更新，並且是否有主動通知既有客戶進行產品的升級。

4.4.6 未採用項目的可能風險

個案 S 公司囿於有限的資源，並且專注於軟體產品的發展，因此並未全面採用 MSF 所建議的所有功能領域項目，本小節將針對個案在 5 個運行歷程，7 個代言功能領域中未採用的項目逐一進行說明，並分析不執行此項目可能帶來的風險與影響（如表 4-9）。

表 4-9　未採用功能領域項目的風險與影響列表

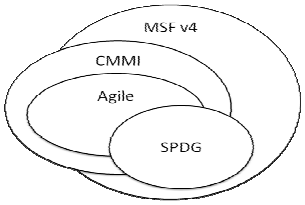
歷程/功能領域	未採用項目	可能風險與影響
展望歷程/計畫管理	M1.專案結構	當組織結構調整時或團隊成員變動時，較無法反應實際資源狀態，可能會造成資源的配置不當。
展望歷程/使用者經驗	M2.使用者驗收準則	無。使用者驗收準則納入產品的功能與非功能需求之中。
展望歷程/發行與作業	M1.部署成效	無。軟體產品與軟體專案的特性不同，因而無須特別評量佈署成效。

規劃歷程/ 計畫管理	M3.預算	未進行預算編列，將無法規劃資金使用時機，也很難評估發展產品的成本，應注意預算超支以及現金流量不足等風險。
規劃歷程/ 使用者經驗	M4.可及性需求 M6.使用手冊 M7.教育訓練	將手冊與教育訓練移至穩定化歷程再進行，雖可節省整體成本，但由於執行的時間被壓縮，所以須特別注意產出物的品質。
規劃歷程/ 發行與作業	M1.設計的評估 M4.定義與建立各種專案環境	發行與作業團隊不參與設計評估，須注意產品部署升級時的適用性，以及不同客戶環境的相容性。 無。在穩定化歷程中，以多個客戶的試行測試來替代建立各種專案環境。
建置歷程/ 產品管理	M1.釐清範疇、需求、與利害關係人的期望	無。軟體產品需求與範疇在展望歷程即已確認。
建置歷程/ 使用者經驗	M1.教育訓練 M4.支援工具與使用手冊	將手冊、支援工具、教育訓練搬移至穩定化歷程再進行，雖可節省整體成本，但由於執行的時間被壓縮，所以須特別注意產出物的品質。
建置歷程/ 測試	M3.文件測試	由於大部分文件製作被移至穩定化歷程，因此建置歷程將不執行文件測試，可能的風險為文件品質不佳，無法符合使用者的需要。
建置歷程/ 發行與作業	M1.核對檢查表 M3.站點準備檢查表	無。軟體產品與軟體專案的進行方式有所不同，因此個案未採用此二項目。
穩定化歷程/ 使用者經驗	M3.使用者驗收 M4.易用性評估 M6.作業團隊 M7.技術支援團隊	雖然在規劃歷程中，會考量到使用者的操作需求及產品的易用性，但由於沒有執行此階段的驗證，因此有可能會造成使用者在使用產品功能時的效果不佳，或衍生出許多的操作問題。
部署歷程/ 產品管理	M2.每站點完成部署的簽收	無。簽收工作改由作業團隊執行。
部署歷程/ 計畫管理	M1.部署計畫與時程的更新	無。軟體產品的安裝與更新已經有既定的流程，因此無須訂立特別的部署計畫。
部署歷程/ 測試	M1.問題分級 M2.讓部署工作趨於穩定	此二項目均轉交給作業團隊來執行，可清楚劃分工作範圍，但可能造成測試團隊無法確切瞭解問題發生的原因、問題重製的步驟，以及不能夠將此經驗反饋至產品的測試案例中，造成同樣問題在後續產品中仍然可能會出現。

資料來源：本研究整理

4.5 建構實務指引

本研究參考了(1) MSF v4 的基本原則、思維、團隊模型(2)個案提出的 15 項執行方法(3)個案治理模型的功能領域項目，最後彙整後提出小型軟體公司軟體產品發展指引（SPDG，Small Company Software Product Develop Guideline），以期能符合國內的軟體開發文化與環境，降低軟體產品發展管理的困難性。SPDG 是參考 MSF 框架所建立的實例，同時也納入許多敏捷式開發的概念，SPDG、MSF 和其他發展實例的關係如圖 4-2。以下就 SPDG 的內容、建議執行步驟以及達成效益作分項說明。



資料來源：本研究整理

圖 4-2 SPDG 與其他實例關係圖

4.5.1 SPDG 內容

1. 目的：讓小型軟體公司在發展軟體產品時，能夠擁有一個確實可行的實務指引，節省因為錯誤嘗試所耗費的時間與金錢，並且以 SPDG 為基礎，透過自身公司的實踐經驗，調整出一套最適合於自己公司的軟體產品發展管理機制。
2. 基本原則：遵循 MSF 所定義的基本原則，在 SPDG 中特別重視下列項目。

- 為共同願景努力：在小型軟體公司內，由於團隊成員數量精簡，每一個人對於產品發展是否能成功都影響甚鉅，因此取得團隊共識，成員擁有共同的願景並一起努力，可以幫助團隊克服在軟體發展過程中所遭遇的各項困難。
- 明訂個人與團隊的職責：因為團隊模型將採用代言人團隊模型，每一位成員都應該要對其所代言的功能領域負起責任，當同一位成員代表多個功能領域時，可參考執行方法 E-3，以工作時間區分的方式，來避免多個代言領域之間立場衝突的問題。
- 投資品質：軟體產品如果希望能夠長期發展，提高品質為最重要的工作之一，因為高品質的軟體產品不只是讓產品競爭力提昇，同時也降

低後續客戶服務與作業團隊的負擔。執行方法 E-5，以規劃的角度確保需求的品質被確認，達到符合市場需要的功能，而執行方法 E-9 至 E-12 則是以檢測的角度，確認產品功能正確運作。

3. 思維：遵循 MSF 所定義的團隊成員中心思維。特別需要注意的是，如何讓每一位成員都專注於商業價值，成員之間必須拋棄本位主義，以提昇客戶對產品認同的價值作為第一要務，產品開發過程中也要避免沒有明顯商業價值的功能開發，以及不必要的新技術採用。
4. 團隊模型：參考 MSF 團隊模型，建立容易調整與可擴充的功能代言群團隊，並針對小型團隊成員人數的多寡予以修正。

- 團隊人數 3 人：MSF 建議第一人負責使用者經驗、產品管理與測試，第二人負責計畫管理與發行管理，第三人負責架構與開發。在 SPDG 則建議第一人負責產品管理、計畫管理、使用者經驗、發行與作業等 4 項功能領域，第二、三人則同時負責架構、程式開發、測試等 3 項功能領域，並使用執行方法 E-3，以工作時間區分來完成產品開發中的各項任務，這樣的團隊規劃，好處在於測試時可以交互測試，開發時則可以實施 XP 中的雙人式程式設計，提昇開發效能與程式碼品質。
- 團隊人數 4 人以上：SPDG 建議將成員分為三個群組，第一個群組負責產品管理與使用者經驗，第二群組負責測試、發行與作業，第三群組人負責計畫管理、架構、程式開發。團隊成員在增加的過程中，優先增加第三群組，其次是第二群組。

5. 治理模型：依據軟體產品發展歷程，執行小型軟體公司在各運行歷程中的焦點功能領域、關鍵產出與檢核點，如表 4-10、4-11，各歷程的重點項目描述如下：

表 4-10　SPDG 運行歷程的焦點功能領域

歷程	展望歷程	規劃歷程	建置歷程	穩定化歷程	部署歷程
產品管理	・整體產品目標 ・辨識目標市場與客戶需求 ・定義產品遠景/範疇 ・客戶驗收準則	・概念設計 ・商業需求分析 ・溝通計畫 ・排列需求與優先順序	・市場通路與協銷處理 ・建立產品文件	・執行溝通計畫 ・上市規劃 ・決定範疇取捨的優先順序	・蒐集客戶回饋意見、評定
計畫管理	・專案限制因素 ・各項限制因素的取捨	・主專案計畫 ・主專案時程	・管理功能規格 ・專案的追蹤 ・計畫的更新	・專案追蹤 ・限制因素和範疇取捨	・管理穩定化的相關活動
架構	・設計目標策略 ・概念解決方案 ・可行性分析 ・建置與技術選項	・概念與邏輯設計 ・細部技術評估 ・初期建置預估 ・功能規格書	・確認架構 ・確認功能規格 ・釐清設計細節	・問題分級	・解決方案/範疇的比較
程式開發	・提出需求與策略 ・規劃開發環境	・邏輯與實體設計 ・建置計畫/時程 ・建置預估 ・雛型 ・開發環境就緒	・建構解決方案 ・發展基礎建設 ・編寫組態文件 ・單元測試 ・系統介面測試	・問題解決 ・解決方案最佳化 ・建構安裝升級程式	・解決問題 ・第二線支援
使用者經驗	・提出使用者的非功能性需求 ・使用者驗收準則	・使用情節/案例 ・使用者需求 ・多語言需求 ・易用性需求	・易用性測試 ・圖形設計	・使用者文件穩定化 ・教育訓練教材 ・訓練試行使用者	・教育訓練 ・提昇使用者操作能力
測試	・測試策略 ・測試方法 ・測試度量指標 ・規劃測試環境	・設計的評估 ・測試需求 ・測試計畫/時程 ・測試情節 ・測試環境就緒	・功能測試 ・辨識問題 ・更新測試計畫 ・部份整合測試	・以使用者角度執行功能測試 ・系統整合測試 ・回歸測試 ・提報問題與狀況 ・組態測試	NA
發行與作業	・作業管理與支援性 ・服務品質 ・作業驗收準則	・作業需求 ・試行與部署計畫/時程	・發布更新與更新試行計畫	・試行的設置與支援 ・產品安裝與升級 ・作業與技術支援的準備	・客戶端的部署管理 ・變更的核准

資料來源：本研究整理

- 展望歷程：產品目標、願景與範疇，需要團隊成員共同確認，凝聚團隊共識。需求也應該在此歷程中進行初步的評估。
- 規劃歷程：將使用手冊、教育訓練等有關使用者經驗的功能項目移至穩定化歷程中，以節省開發時程，並且納入的測試驅動開發（TDD）的概念，主要測試案例以及細部技術評估都應該被執行。
- 建置歷程：以程式開發團隊為主的建置歷程，除了程式碼的開發外，應著重於開發團隊內部的測試工作，以及高風險性的整合項目測試，這些工作可以降低產品發展風險，提昇軟體的品質。
- 穩定化歷程：最重要的工作是產出可公開測試的 Beta 版本，並且採用客戶端試行模式，透過團隊內部與客戶端的雙重測試檢驗，確保產品品質。
- 部署歷程：著重在新版本發佈後，如何通知並完成客戶端的安裝升級，並且透過作業團隊，蒐集客戶所回饋的意見，作為後續版本演進的參考資料。

6. 執行方法：條列在表 4-3 中的 15 項執行方法，為經過驗證且確實可行的方法，除含括了團隊模型與治理模型，也提供了在開發技術上的相關建議事項。

表 4-11　SPDG 運行歷程的關鍵產出、檢核點

歷程	展望歷程	規劃歷程	建置歷程	穩定化歷程	部署歷程
關鍵產出	・產品遠景/範疇文件 ・專案結構文件 ・初始的風險評估 ・需求說明文件 ・需求評估結果	・功能規格 ・主專案計畫 ・主專案時程 ・系統設計文件與雛型 ・主要測試案例	・行銷文宣、使用者意見交流 ・更新過的主計畫、時程、風險文件 ・已定案的設計、已凍結的功能規格 ・原始程式碼和執行檔、文件、輔助元素 ・測試規格書、測試案例、測試度量指標、測試指令碼、測試資料、測試控管機制 ・Alpha 版本 1- n	・整合的解決方案元件 ・部署指令碼與安裝說明文件 ・使用者說明文件與教育訓練教材 ・與使用者溝通的機制 ・作業文件 ・測試與問題報告 ・本次發行的相關資訊 ・Beta 版本 1- n	・修訂的流程與程序 ・使用者與管理員的教育訓練 ・組態文件 ・解決方案的儲存庫 ・驗收紀錄與工作紀錄
檢核點	・產品目標已確認 ・產品遠景/範疇基準已確立 ・已初步評估產品需求 ・遠景範疇已核准	・技術確認完成 ・功能規格基礎已完成 ・主計畫基準已完成 ・主計畫時程基準已完成 ・支援環境已完成 ・專案計畫已核准	・雛型設計完成 ・內部 n 個版本的解決方案發行 ・範疇已確立 ・通過所有功能測試	・1- n 次回歸測試完成 ・問題收斂 ・使用者介面已穩定 ・問題紀錄已清除 ・整合測試完成 ・試行完成 ・發行準備已核准	・部署作業已穩定 ・主要客戶部署完成 ・完成既有客戶的更新通知或部署

資料來源：本研究整理

4.5.2　SPDG 的執行流程

為了同時適用於新成立與既有的軟體產品公司或團隊，本節將 SPDG 的執行流程區分為新軟體產品發展與既有軟體產品發展機制調校二項作說明。

1. 新軟體產品發展

・適用情境：新成立小型軟體產品公司或是在公司內部要執行新的軟體產品發展時。

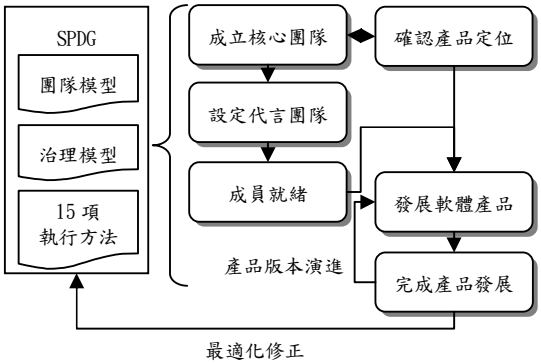
・流程說明：執行流程如圖 4-3，各步驟說明如下。

– 步驟一、確認產品定位：依照市場研究與調查的結果，決定出所要發展的軟體產品與產品的市場定位，此步驟是完成部份在展望歷程產品管理功能領域所要執行的工作。在軟體公司成立時，所要發展的軟體產品定位，通常會影響到核心團隊的組成。

– 步驟二、成立核心團隊：依據產品的定位組成核心團隊。在新軟體公司成立的情境下，核心團隊成員的背景通常會影響產品的定位與方向，因此本步驟與上一個步驟是交互影響的。

– 步驟三、設立代言人團隊：依照 SPDG 的團隊模型，進行代言功能領域的指派與分工。

– 步驟四、團隊成員就緒：各功能領域代言人招募所須的團隊成員，並且讓團隊成員技能就緒。
– 步驟五、發展軟體產品：依照 SPDG 治理模型，進行軟體產品的開發。
– 步驟六、完成軟體開發。
– 步驟七、持續修正與改善：軟體產品可能因為瑕疵或是市場需求而進行後續版本的演進；而產品發展過程中的實務經驗，也將會修正 SPDG 中所定義的相關作業項目。



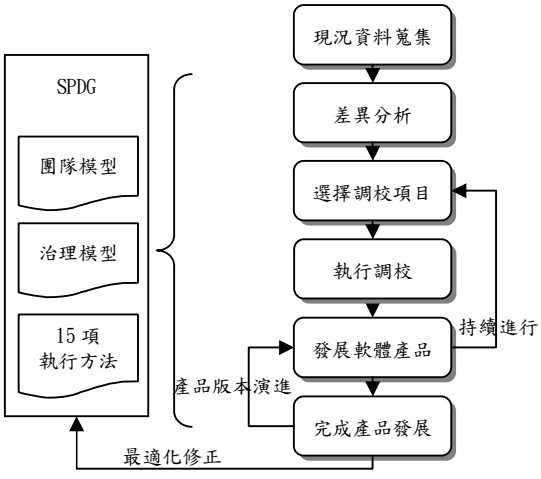
資料來源：本研究整理

圖 4-3 新軟體產品發展執行流程圖

2. 既有軟體產品發展機制調校

・適用情境：對小型軟體產品公司目前既有的開發管理機制進行調校。

・流程說明：執行流程如圖 4-4，各步驟說明如下。



資料來源：本研究整理

圖 4-4 既有軟體產品發展機制調校執行流程圖

– 步驟一、現況資料蒐集。蒐集、整理公司內部軟體產品發展所定義的流程、規章與歷史紀錄。
– 步驟二、差異分析。將既有的管理機制與 SPDG 作比較，找出二者之間的差異項目。
– 步驟三、選擇調校項目。依照優先權或目前所遭遇問題的嚴重性，從 SPDG 中選擇出要執行調校的項目。
– 步驟四、執行調校。執行調校項目來改善既有的管理機制。
– 步驟五、發展軟體產品。
– 步驟六、完成軟體開發。與新產品發展步驟相同。
– 步驟七、持續修正與改善。除了新產品發展步驟七的執行項目外，同時也會重複執行選擇調校項目，進行逐步改善。

4.5.3 SPDG 達成效益

本節所討論的 SPDG 效益，意指如果個案依照 SPDG 重新發展 W 產品 v1 至 v3，即可避免過程中的錯誤嘗試，並解決初期經驗不足的問題，進而達到節省開發成本、加速開發時程等效益，以下就可量化與非量化效益進行說明。

1. 可量化效益

個案所花費的產品發展時間達 378 個人月，如果 S 公司在 W 產品發展初期即執行 SPDG，可以達成的可量化效益，如表 4-2 所示，分項說明如下：

・在治理模型的問題領域：執行方法 E-5，避免了因為產品功能無法滿足市場需求，功能需要重新開發的問題，節省了 18 個人月的開發時間；執行方法 E-9 至 E-12，解決了因為產品品質不穩定，而需要經常更新版本來修正產品瑕疵，可節省程式開發人力、測試人力、作業人力達 24 個人月。

・在開發與技術的問題領域：執行方法 E-13，修正了開發語言不一致性所帶來的除錯、維護等問題，可節省開發人力 6 個人月；執行方法 E-14，避免了第三方元件選擇上的錯誤，節省程式開發人力與作業人力 13 個人月。

在 6 年的產品發展期間中，總計 SPDG 可量化的效益達 61 個人月（如表 4-12），佔總開發人月數（378 個人月）的 16.1%，如以薪資計算，每一個人月所須費用為新台幣 16.5 萬元（99 年度資訊委外服務人員計價參考要點[1]，以第三類程式開發設計人員薪資計算），共節省了新台幣 1,006.5 萬元，對小型軟體公司可謂成效顯著。

表 4-12　SPDG 可量化效益 – 節省人月數

執行方法	描述	(類型)/人月數
E-5	・確認開發功能符合市場預期，避免開發出無市場需求的功能項目。個案可節省 8 個人月的開發時間。 ・避免因為需求不明確所造成的重工與程式碼的反覆調整，總計節省程式開發人力約 10 個人月。	(程式) 18 個人月
E-9 ~ E-12	・減少產品瑕疵數，降低產品更新頻率，由 1-2 月更新一次調整為 2-4 個月更新一次。因此節省安裝與更新程式的開發人力 2 個人月、每次版本發佈所需的測試人力 10 個人月、協助客戶端產品升級的作業人力 12 個人月。	(程式) 2 個人月 (測試) 10 個人月 (作業) 12 個人月
E-13	・減少程式碼重新撰寫（統一為同一種開發語言）與除錯所花費的時間，節省程式開發人力 6 個人月。	(程式) 6 個人月
E-14	・減少重工（變更使用的第三方元件），節省程式開發人力 7 個人月。	(程式) 7 個人月
	・提昇產品穩定度與品質，節省因為協助客戶端產品升級或是問題排除所需要的作業人力 6 個人月。	(作業) 6 個人月
		總計 61 個人月

資料來源：本研究整理

2. 非量化效益

SPDG 的執行效益，除可量化的項目外，也包含了數項非量化效益：

・提高產品完整性：透過團隊模型與治理模型，在不同的歷程中，各焦點功能領域都會被定義與關注，這讓產品在發展的過程中，不會因為成員背景的單一性，造成產品功能設計上的偏頗。

・提昇團隊工作效率：團隊模型配合執行方法 E-2、E-3，讓多樣性的功能領域得以執行，避免一人同時負責多個代言群的衝突問題，提昇團隊的整體效率。

・提昇產品品質：治理模型與多項的執行方法，均能夠提昇產品品質，例如執行方法 E-12，採用測試驅動模式，使測試在程式碼開發前就開始進行，避免程式設計時的盲點；自動化單元測試機制，除縮短測試時間外，也讓測試的頻率增加，避免人為操作的疏失。

・累積核心技術：執行方法 E-15，使得在有限人力資源的環境中，能同時兼顧資源運用與核心技術的累積，有助於小型軟體公司的長期發展。

這些無法明確量化的效益，可以從個案 S 公司各部門工作流程的改善，以及 W 產品於市場上接受度的提昇等跡象中得到驗證。

4.5.4　建議優先實施的執行方法

在業界實務的環境中，大多數的軟體公司都已經存在既有的軟體產品發展管理機制，上一小節說明了既有產品發展機制調校的執行流程（如圖 4-4），其中步驟三為選擇調校項目，以小型軟體公司來說，資源較小且承受風險能力較低，因此本研究特別提出 SPDG 中的四項執行方法，作為建議優先執行的項目。

・執行方法 E-3：將測試團隊獨立或是以工作時間區分程式開發和測試工作。在小型軟體公司開發人數較少的狀況下，很難配置一位專職的測試人員，因此大多數公司採用開發者交互測試的方式作為權變，E-3 以明確規範的時間，區分程式開發與測試工作，避免團隊成員重開發而輕測試，造成產品品質的低落。

・執行方法 E-5：透過多位客戶的深入訪談與雛型展示，確保產品功能符合期望。小型軟體產品公司所發展的軟體多以利基市場（Niche Market）為主要市場策略，因此需求的確認十分重要，透過與客戶的深度訪談以及雛型展示，可以確認需求項目不會偏離，並且隨著客戶數的增長，在開發新產品功能時，應該將多位客戶的需求進行交叉驗證，讓產品功能效益最大化。

・執行方法 E-6：標竿產品功能分析，與客戶需求進行交叉比對。在產品開發初期，由於客戶數量較少，需求的完整性難以確認，因此透過市場上既存的標竿產品研究，可以讓產品功能有明確的發展方向；隨著客戶數的增長，標竿產品功能的分析則轉換為驗證客戶端需求的輔助方式，如此才能建立出既符合市場期待且擁有差異化的軟體產品。

・執行方法 E-11：執行產品公開測試，選定少數客戶進行新版本的試行，補強測試環境與實際環境之間的差異。小型軟體公司所能建置的測試環境受限於經費與人力，規模一定不足以涵蓋客戶的使用環境，因此選取數家有代表性的客戶作為新版本試行的對象，可以大幅降低測試成本，並及早發現產品問題，當然與這些客戶之間必須採取互利的模式，例如：優先滿足需求、免費使用新功能模組等方式。

上述 4 項執行方法，不會對既有的組織結構有所影響，組織內部推動的阻力較小，且執行後，對於軟體產品的長期發展有著明顯的正面助益，因此本研究提出此 4 項執行方法作為優先執行的項目。

5. 結論與建議

5.1 結論

本論文透過個案 S 公司 6 年來的實務經驗，進行資料的蒐集、整理與分析後，作出結論如下：

1. SPDG 的治理模型比 MSF 的治理模型更精簡且更適合於小型軟體公司：個案實務與 MSF 的治理模型之間約有 30%的差異(如表 4-4 至 4-8 所示)，主要原因在於個案屬於小型軟體公司，且只專注於軟體產品的開發，因此本研究所建立的 SPDG 治理模型（如表 4-10、4-11 所示）必然會比 MSF 治理模型更符合小型軟體公司的需求。
2. 使用 SPDG 的執行方法，可節省 16.1%的產品發展時間：小型軟體公司囿於時程、經費、人力與環境等因素，在開發過程中會遭遇許多的問題，本研究所歸納出的 15 項執行方法，可以提昇軟體產品發展的成功機率，並減少共計 61 人月的開發時間(如表 4-12 所示)，換算為開發費用共節省了 1,006.5 萬元。
3. SPDG 為小型軟體公司最適合的軟體產品發展指引，同時兼顧既有任務的執行與開發管理機制的改善：本研究提供了團隊模型、治理模型、15 項執行方法與實施的流程建議，使小型軟體公司得以依據 SPDG 進行新軟體產品的開發或是既有軟體產品機制的調整（如圖 4-3、4-4 所示)，並且能夠讓小型軟體公司在資源不足的狀況下，依照公司需求，排定優先順序逐步實施 SPDG 的各項項目，無須承擔全面性的高風險變更，也能達成改善軟體產品發展機制的目的。

5.2 建議

1. 本研究對象為單一個案 S 公司，建議後續可研究其他小型軟體產品公司，持續 SPDG 的演進。
2. 當小型軟體公司逐漸成長後，在軟體產品管理上，將會面臨許多的考驗，建議後續可研究如何讓小型軟體產品公司配合著公司規模的擴增，順利升級既有的軟體產品發展機制。
3. 本研究並未對小型軟體產品公司所開發的商用軟體進行分類，後續研究可依據不同的軟體產品類型進行細部的探討，發展出各式軟體產品的最佳實務指引。

參考文獻

[1] 中華民國資訊軟體協會，*99 年度資訊委外服務人員計價參考要點*。http://www.cisanet.org.tw/Download/Download/ed0fcc8b-12ec-4e0e-abaa-75967666107e，Accessed 2011/1/15。

[2] 蔡煥麟，*微軟解決方案框架精要*，博碩文化股份有限公司，2007。

[3] Agile Alliance, *Manifesto For Agile Software Development*, http://www.agilemanifesto.org/, Accessed 2010/12/15.

[4] Barry W. Boehm, “Software Engineering Economics,” *IEEE Transactions On Software Engineering*, Vol. SE-10, No. 1, 1984, January.

[5] Carnegie Mellon University, *CMMI For Development*, Version 1.2, CMU/SEI-2006-TR-008, ESC-TR-2006-008, 2006.

[6] Christof Ebert, “Software Product Management,” *CrossTalk：The Journal of Defense Software Engineering*, 2009, January, pp. 15-19.

[7] Christof Ebert, “The Impact Of Software Product Management,” *The Journal of Systems and Software*, Vol.80, Issue 6, 2007, June, pp. 850-861.

[8] Christof Ebert, “Understanding The Product Life Cycle: Four Key Requirements Engineering Techniques,” *IEEE Software*, Vol. 23, Issue 3, 2006, May-June, pp. 19-25.

[9] Craig Larman, *Agile & Iterative Development：A Manager’s Guide*, Addison Wesley, 2003.

[10] David J. Anderson, “Stretching Agile To Fit CMMI Level 3 - The Story Of Creating MSF For CMMI Process Improvement At Microsoft Corporation,” *Proceedings of the Agile Development Conference (ADC’05)*, 2005.

[11] Michael S.V. Turner, *Microsoft Solutions Framework Essentials: Building Successful Technology Solutions*, Microsoft Press, 2006.

[12] Inge van de Weerd, Sjaak Brinkkemper, Richard Nieuwenhuis, Johan Versendaal, and Lex Bijlsma, “Towards A Reference Framework For Software Product Management,” *Proceedings of the 14th IEEE International Requirements Engineering Conference*, 2006, September 11-15, pp.319-322.

[13] Kent Beck, *Extreme Programming Explained：Embrace Change*, Addison-Wesley, 2000.

[14] Kent Beck, *Test-Driven Development By Example*, Addison Wesley, 2003.

[15] Kent Beck, *Extreme Programming*,

http://www.extremeprogramming.org/, Accessed 2010/12/02.

[16] Kristian Rautiainen, Casper Lassenius, and Reijo Sulonen, "4CC: A Framework For Managing Software Product Development," *Engineering Management Journal*, Vol. 14, No. 2, 2002, June.

[17] Robert K. Yin, *Case Study Research: Design And Methods (2nd ed.)*, Beverly Hills, CA: Sage Publishing, 1994.

[18] Sybren Deelstra, Marco Sinnema, and Jan Bosch, "Product Derivation In Software Product Families: A Case Study," *The Journal of Systems and Software*, Vol. 74, 2005, pp. 173-194.

[19] Sybren Deelstra, Marco Sinnema, and Jan Bosch, "Experiences In Software Product Families: Problems And Issues During Product Derivation," *Lecture Notes in Computer Science*, Vol. 3154, 2004, pp. 120-122.

[20] Wiki, *Test-Driven Development*, http://en.wikipedia.org/wiki/Test_driven, Accessed 2010/12/02.