以高品質安全軟體開發製程改善軟體安全品質之研究

A Study of High Quality Secure Software Development Process for Improving Software Security

賴森堂
實踐大學資訊管科技與理學系
大直街 70 號
台北 104 中山區
stlai@mail.usc.edu.tw

Sen-Tarng Lai
Department of Information Technology and Management, Shih Chien University
No.70, Ta-Chih Street,
Taipei 104,Taiwan
stlai@mail.usc.edu.tw

摘要

以資訊爲重心的年代，資訊安全的問題對於人類大環境的影響愈來愈嚴重，網際網路的入侵、病毒攻擊與系統本身的安全漏洞持續危害正常運作的軟體系統，使得資訊系統的安全性受到嚴重的考驗。軟體開發製程經過不斷的演進與改善，已成爲一套嚴謹且成熟的軟體開發程序，不過，軟體製程卻極少深入描繪安全品質，使得軟體安全品質無法有效融入產品中，造成上線使用的軟體系統存在高度的安全危機，成爲安全軟體建置過程中值得深入探究的課題。爲了避免安全缺失與漏洞造成軟體系統難以預期的後果與損失，本文以現有的軟體開發製程爲基礎，加強制度、管理、技術等三個層面的安全措施，進而規劃出一套安全軟體開發製程(Secure Software Development Process ; ***SSDP***)，於軟體開發初期就能標示出階段性的安全缺失與漏洞，有效提昇軟體系統的安全性，且提出一套安全開發製程品質量測(***SSDPQM***)模式，有效監控與不斷改善安全開發製程的問題與缺失，確保安全軟體開發製程能夠持續強化軟體系統的安全性。

關鍵詞: 軟體安全性、安全漏洞、品質量測模式、安全軟體開發製程、安全管制作業。

Abstract

In the information age, information security issues are getting serious to the impact of the human living environment. Network intrusions, virus attacks and system vulnerabilities continue to endanger the normal operation of the software system and severe test the security of software systems. Software process with continuous improvement and evolution has become a rigorous and mature software development model. However, most of software processes very little depth describe the software security, so the security can not be effective injected into the software products. For reducing the software system security risk, secure software development process becomes a worth further exploration topic. According to the

related reports, software security vulnerabilities often cause unpredictable consequences and losses. For this, in this paper, based on the current software processes, combine with the system, management and technology three security strategies, propose a Secure Software Development Process (SSDP). Applying the *SSDP*, the software developer can identify and revise the early stages of security defects and vulnerabilities, to enhance software system security. In addition, in this paper, in order to assure usability of *SSDP*, proposes a *SSDP* Quality Measurement (***SSDPQM***) model. With ***SSDPQM*** model, the *SSDP* operation problems and defects can be effective monitoring and continuous improvement and ensure the SSDP can strengthen the security of the software system.

Keywords: Software security, security vulnerability, quality measurement model, *SSDP*, security control

一、緒論

軟體系統的安全性對於使用單位的影響已經超越功能與效能的需求[3]，因爲安全漏洞所造成的危害與損失，輕者可能迫使一家公司的倒閉，重者甚至危害整個社會的正常運作。造成軟體安全漏洞的因素很多，依據安全漏洞的形成原因，可以歸納成兩大類：(1)軟體開發過程的疏失所形成軟體安全漏洞[13]；(2)運作環境的規劃缺失導致維運環境的安全漏洞[4][17]。不過，受到軟體開發人力不足與時程緊迫的壓力，軟體安全品質經常不受到高層主管的重視甚至被忽略，使得軟體安全漏洞問題越加嚴重，造成的危機與損失更是難以預期。爲了減輕軟體安全漏洞造成的危害，許多安全軟體開發程序與安全軟體的研究成果陸續被提出[8][17][19]，其目的就是希望在軟體開發過程中，就將安全特性視爲重要考量與查核的項目，每個開發階段更能依製程的要求將安全品質融入產品中，以具體提昇軟體的安全性。雖是如此，許多被廣泛引用的軟體開發製程，卻極少將軟體的安全性列入開發作業的考量項目，使得軟體安全特性無法有效融入產品中，造成運作中的軟體系統存在高度的安全危機，成爲安全軟體開發過程中值得深入探究的課題。

軟體系統的安全漏洞造成使用單位的損失與危機是難以預期與估算的，爲了避免安全漏洞造成的嚴重後果，軟體開發過程就應該正視安全品質的問題，軟體測試階段固然是找出安全漏洞的關鍵步驟，但是測試作業卻著重於功能性、整合性與執行效能是否符合規格[10]，而不是以安全漏洞與安全缺失爲確認重心。此外安全漏洞與缺失若未能獲得及時修補與改善，勢必將隨著軟體系統上線後移轉給使用單位，使得安全品質漏洞成爲軟體運作過程中的一顆不定時炸彈，隨時都可能造成難以預期的危害。軟體測試階段是企圖找出不符合需求規格的作業，在軟體開發階段，使用單位若未能針對安全品質定義具體的需求項目，將會造成安全品質無法成爲測試的項目，使得開發作業所植入產品中的安全漏洞，不能有效被標示且隔離出，成爲軟體系統安全品質的一大隱憂。如何在軟體開發作業初期，就考量安全項目且具體融入安全品質，將持續遞延與擴大的安全漏洞與缺失降到最低，不僅可以減輕安全測試作業的龐大負擔，更可以避免軟體安全漏洞遞延至上線的系統中，是安全軟體開發作業必須達成的任務。

軟體系統安全性一般都被列爲事後考慮的情況，當功能規格都能夠滿足需求，且軟

體專案的時程與預算都在規劃的範圍內，軟體系統安全性才會被列入考量，使得軟體安全品質的問題愈加嚴重。造成軟體安全漏洞的原因很多，卻幾乎都與開發作業的疏忽及運作環境的缺失等問題密切相關，其中又以開發作業疏忽衍生遞延與擴大的影響最大且最久[18]。爲了降低安全漏洞的風險，必須強化軟體製程的安全機制，於開發初期就有效標示出安全缺失漏洞，且及時提出修補與改善作業[14][15]。本文以現行的軟體製程爲基礎，加強制度、管理、技術等層面的安全管制作業，進而提出一套安全軟體開發製程(Secure Software Development Process；***SSDP***)，於軟體開發初期即能找出階段性的安全漏洞與缺失，且提出具體的修訂與改善措施，有效提昇軟體系統的安全性。此外爲了確認 ***SSDP*** 的有效性，本研究以制度、管理與技術等三個層面爲基礎，提出一套 *SSDP* 品質量測(*SSDP Quality Measurement*; ***SSDPQM***)模式，用以監控與持續改善 ***SSDP*** 執行上的問題與缺失。本文第二節將探討軟體開發階段所注入的安全漏洞以及如何透過安全控管機制協助找出安全漏洞。第三節將探討缺乏安全管制作業的軟體開發過程中，安全漏洞與缺失被發現的時段及其改善的花費。第四節將結合強化制度、管理、技術等層面的安全管制作業，提出一套安全軟體開發製程(***SSDP***)，透過嚴謹的安全控管機制，找出軟體開發過程的安全漏洞與缺失，以便達到及時改善的效果。爲了確認 ***SSDP*** 的有效性，第五節將提出一套 *SSDP* 品質量測(***SSDPQM***)模式，用以監控與持續改善 ***SSDP*** 執行上的問題與缺失。第六節以電信批價子系統說明本研究提出的 *SSDP* 運作流程與改善作業。第七節將說明安全軟體開發製程的優勢與重要性，且針對本文主題作個結論。

二、軟體系統的安全缺失

安全漏洞的依形成原因，可以歸納成兩大類：(1)軟體開發過程的疏失所形成軟體安全漏洞；(2)運作環境的規劃缺失導致維運環境的安全漏洞。

1. 軟體系統的安全漏洞與弱點

軟體工程早期著重於生產力與品質的提昇，而忽略軟體安全的重要性，在 ISO 的軟體品質特性架構中[2]，安全性並不受到重視，不僅被列爲低層級的次要特性，而且歸屬於功能性(functionality)之下[18]，這完全不能符合網際網路環境下的資訊系統，因爲安全性的問題已經超越功能與效能的需求。依據 CERT/CC 最新的統計數據顯示[22]，從 2003 年至 2008 第三季的軟體系統弱點分析報告總共有 34,910 件 (如表一所示)，彙整最近三年的平均，每年更高達七千多件。資訊系統弱點的數量愈多，就代表安全漏洞愈嚴重，系統被入侵或形成安全危機的比率也會因而增加。在 E 世代的各項活動中，軟體系統與網路結合的環境成爲人們相互溝通與交易的必要裝置，因此當軟體系統無法有效排除安全弱點與漏洞時，將爲個人與社會帶來難以預期的後患，而成爲人們活動的一大隱憂。

軟體系統的安全問題受到許多國際性團體與組織(SANS(安全訓練、認證與研究機構)，OWASP (開放 Web 應用軟體安全計畫，Open Web Application Security Project)的重視，陸續公佈軟體的關鍵安全漏洞與弱點：SANS Top-20 Security Risks[23]與 OWASP Top 10 [24]以協助降低軟體系統的安全風險。從事多年軟體安全研究的學者 McGraw 發表的相關著作與文章中[12][13][21]，也曾對軟體系統開發過程的各項安全風險進行探討，且提出一套風險管理架構(Risk Management Framework)，從軟體系統的企業目標對映出企

業風險，再由企業風險推論出各項安全技術風險。這些統計的資料與數據以及學者的研究成果不僅顯示安全軟體的重要性，也提供改善軟體系統安全性的重要依據。

表一: 2003~2008 第三季的軟體系統弱點數量統計表

年份	弱點數量
2003	3,784
2004	3,780
2005	5,990
2006	8,064
2007	7,234
Q1-Q3, 2008	6,058
加總	34,910

資料來源：CERT/CC（http://www.cert.org/stats/cert_stats.html）

2. 軟體開發過程的安全漏洞

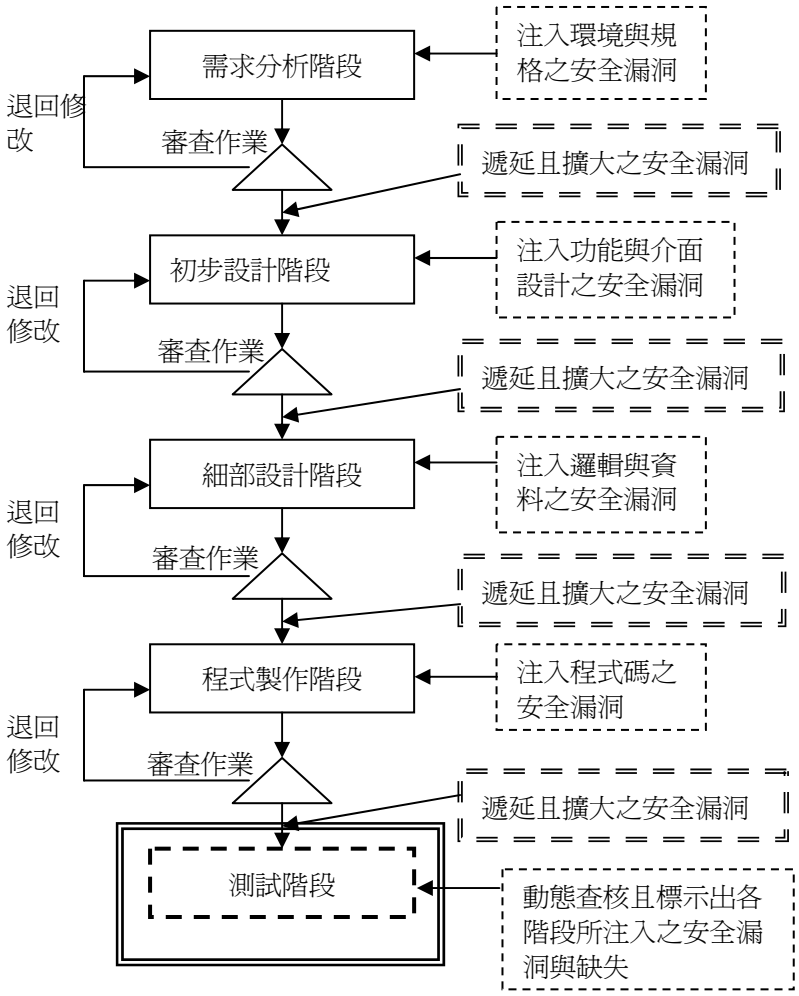
軟體開發作業可以劃分成需求分析、初步設計、細部設計及程式製作等階段[14]，每個階段於開發過程中，都有高度的可能性，因管理、制度或技術層面的疏失而將安全漏洞注入開發的產品及文件中，階段性審查是找出且彌補安全漏洞的關鍵性作業，安全漏洞若不能在審查作業中被標示且修補，勢必隨著開發程序流入後續階段，造成安全漏洞持續的漫延與擴大，使得軟體安全的問題成爲難以預期的危機。爲了有效提昇軟體安全性，許多學者專家都建議，編制一個具有特定工作任務的軟體安全確認(Software Security Assurance; SSA)小組[3][19]，透過第三者的思維可以明確的標示出潛在的安全缺失，用以協助軟體開發團隊建置一套高安全性的軟體系統，不過受限於人力不足與成本的壓力，一般的軟體開發作業中，極少編制一個獨立的 SSA 小組，使得階段性的審查作業無法具體的標示出隱含的安全漏洞。

軟體開發過程，各開發階段有其特定的工作目標與任務，所產生的安全漏洞與缺失必然會有不同的影響，以下針對各開發階段可能形成的安全漏洞與缺失進行說明：

- 需求分析階段：此階段的任務著重於使用者需求規格的制定，因此可能出現的安全漏洞與安全規格制定的不完整、不明確及不一致等缺失密切相關，這些規格上的安全缺失，若不能及時發現且進行補救措施，對於日後的系統運作將形成深遠的影響。
- 初步設計階段：此階段的任務著重於系統功能架構的規劃與外部介面的定義，除了階段本身可能出現的安全漏洞外，來至需求分析階段的安全缺失也是形成漏洞的原因。此階段的安全漏洞一般被隱含在功能及介面的規格中[1]。
- 細部設計階段：此階段的任務著重於模組單元的邏輯結構與資料結構設計，除了階段本身可能出現的安全漏洞外，初步設計階段的安全缺失也將遞延至本階段。此階段的安全漏洞將被隱含在模組單元的邏輯與資料結構的設計中。
- 程式製作階段：此階段的任務著重於程式碼的撰寫與單元測試，除了階段本身可能出現的安全漏洞外，前面階段所注入的安全缺失也將遞延至本階段。程式中缺乏安全缺失的查核與處置步驟是此階段最常出現的安全漏洞[9]，這類型的程式安全漏

洞經常被忽略，卻可能形成很嚴重的後果。

圖一描繪了軟體安全漏洞隨著開發作業的缺失被注入軟體產品中，這些安全漏洞若不能在審查作業中被標示出且加以修訂，安全漏洞將成爲進入測試階段的一大負擔，即使測試階段能夠順利標示出安全漏洞與缺失，卻必須投入龐大的人力與資源進行安全漏洞與缺失的改善作業。



圖一：軟體開發階段的安全漏洞形成圖

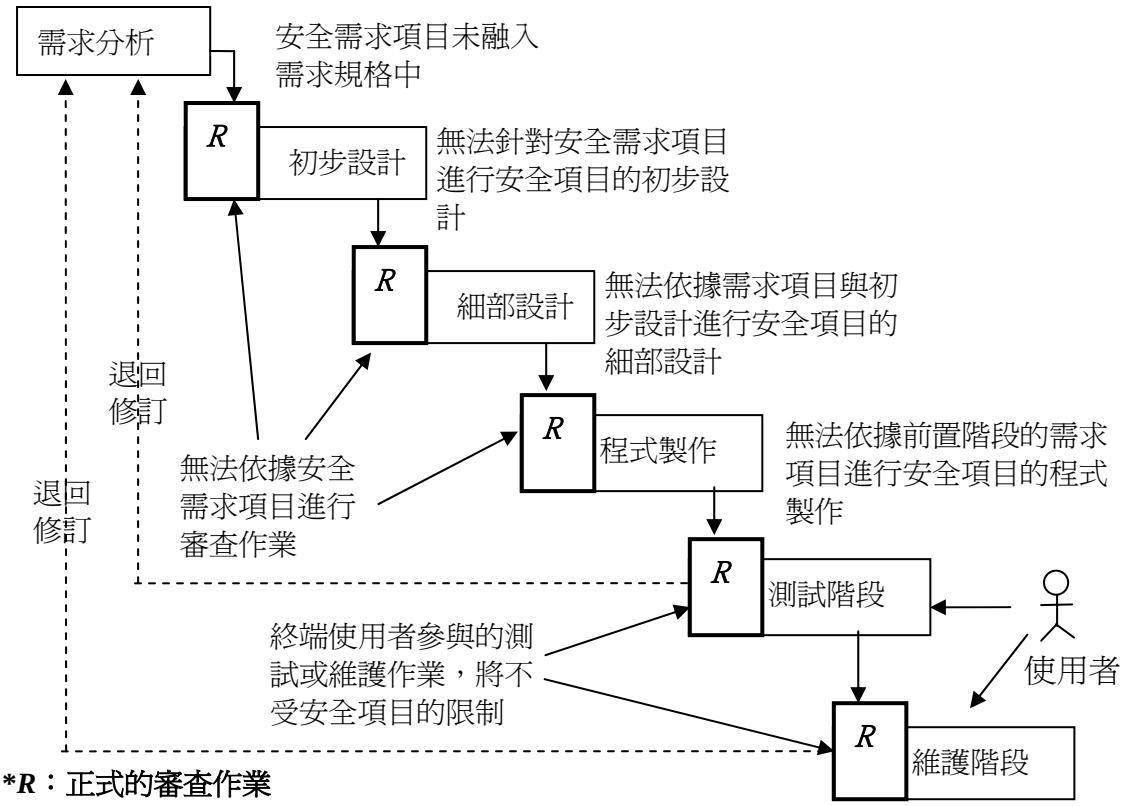
三、未考量安全品質的軟體開發作業

軟體開發初期，就結合制度、管理與技術層面的安全管制作業，將安全品質融入產品中，可以減少維護階段花費龐大的時間、人力與成本來改善軟體的安全品質。

1. 安全漏洞與缺失被發現的時段

使用單位缺乏軟體開發的專業知識，一般無法針對軟體安全品質提出具體的需求，導致軟體需求規格書中將不會明確定義安全規格項目，規格中少了安全需求的規範，軟體開發過程中就不會以安全品質爲考量依據，因此軟體開發初期的系統分析與設計階段，一般都會忽略安全品質的管控措施，不但無法發現安全品質上的問題，更不會針對安全漏洞與缺失提出任何矯正或改善的作業。安全漏洞與缺失將隨著開發作業，持續擴

大且遞延至後續階段(參閱圖二所示)，直到進入測試階段的後面步驟，開始進行系統測試與安裝測試，終端使用者參與進行測試的步驟，安全品質上的問題才會陸續浮現，有些安全漏洞與缺失甚至可能進入維護運作階段才會被發現。安全漏洞與缺失無論是在測試階段或是在維護運作階段被發現，對於整個軟體系統開發作業勢必形成嚴重的傷害，整個開發作業必須退回到系統分析階段，進行大幅的修改與調整，不僅整個開發時程必須往後延長，預算與人力還要再追加，開發團隊的聲譽與可信度更是受到嚴重的打擊。



圖二：未考量安全品質的軟體開發作業流程圖

2. 改善安全漏洞與缺失的花費

隱含在各階段產品文件中的安全漏洞與缺失，若未能及時被標示出，且立即進行修訂與改善作業，安全漏洞與缺失將隨著開發作業持續擴大其影響力且往後續階段遞延，直到測試階段或維護階段，使用單位開始介入後，安全漏洞與缺失的問題將會陸續浮現(參閱圖二所示)，此時才針對發現的漏洞與缺失提出矯正措施及進行改善作業，不僅爲時已晚，更必須投入龐大的人力與成本，才能獲得局部的成效，以下從測試與維護兩階段分別探討安全漏洞與缺失改善作業及花費：

- 測試階段改善花費：軟體開發作業進入系統、現場或安裝等測試步驟，使用單位將派人員參與執行，不受需求規格約束的使用者，較容易找出測試人員沒有發現的安全漏洞與缺失。不過，此時發現的安全問題屬於零亂或突發的事件，更沒有對映的安全需求項目，因此必須先花一些時間進行剖析與匯集，才能找出安全漏洞與缺失的原因，接著再提出修訂與改善措施，調整需要投入的額外人力與時間，且回到需求分析階段逐步進行改善作業，成本與時間上的額外花費，少則增加 20%

~30%，多則可能增加 50% ~60%，甚至更多。

- 維護階段改善花費：進行系統、現場或安裝等測試過程中，使用單位若未派人員參與執行，或是參與的人員經驗不足，未能有效找出測試人員沒有發現的安全漏洞與缺失，在確認軟體系統滿足使用單位的需求規格後，系統將交付移轉給使用單位，軟體系統隨即進入維護階段。運作中的軟體系統一旦出現安全缺失或安全漏洞，將會造成難以預期的後果與損失，企業與組織的商業機密或是個人資料可能因此而外洩，此外運作中的軟體系統找出安全缺失與漏洞的原因更加困難，必須先花更多時間匯集問題，才能具體找出安全缺失與漏洞的原因，接著再針對運作中的軟體系統提出修補作業與改善措施，規劃需要投入的維護人力與時間，在系統持續運作的狀態下，進行安全修補的改善作業，所投入的人力與預算勢必比起測試階段的改善花費多好幾倍。

四、安全軟體開發製程

本節以調整制度層面、搭配管理層面與提昇技術層面的安全管制作業爲依據，提出一套改善軟體安全性的安全軟體開發製程。

1. 結合安全管制作業的軟體製程

爲了解決軟體危機的問題，軟體工程的理念於 1967 年被提出，將近三十年的軟體工程演進過程中，許多軟體開發的方法與技術陸續被發表且提出[18][20]，有效的改善軟體的品質且提昇其生產力，對於解決軟體危機的問題帶來不少的助益。不過，對於軟體安全危害卻極少被提及，使得軟體安全問題成爲軟體開發作業即將面臨的另一項重大危機。爲了提昇軟體系統的安全品質，本文從制度、管理與技術等三個層面來強化軟體開的安全管制作業，以下針對制度層面的調整、管理層面的配合與技術層面的提昇說明之：

(1) 制度層面的調整：

高安全性的軟體系統是開發團隊與使用單位相互合作下的成果，使用單位提出軟體功能、效能與可靠度等各項需求的過程，也必須將開發過程與運作環境可能存在的安全問題與缺失一併考慮，以具體且明確的方式一一標示出安全需求項目，配合開發團隊的軟體安全防範經驗與知識，將軟體安全項目融入軟體需求規格中。不過，受到軟體開發成本、開發人員不足、開發時程與經費等因素的影響，無法將所有的安全項目一次全部考量，此時可以考慮將軟體安全項目依其關鍵性與影響力，劃分成不同的優先順序，配合開發的週期與各方面的影響因素，將安全項目陸續融入軟體產品中。配合安全軟體的開發，制度層面必須具備下面四項的調整：

- 透過制度的規範，需求分析必須將使用單位的安全需求列爲必要的需求項目。使用單位提出的安全需求項目將著重全盤性的考量如終止駭客入侵、杜絕病毒攻擊、確保儲存資料的安全等。
- 透過制度的規範，需求分析必須將安全運作環境需求列爲必要的需求項目。安全運作環境是指系統運作的安全性如資料庫、儲存媒體或網路傳輸等安全。
- 透過制度的規範，需求分析必須將安全介面設計需求列爲必要的需求項目。安全介面設計是指整合作業的安全性如系統、功能模組及元件等介面之間的安全。

- 透過制度的規範，需求分析必須將安全程式撰寫需求列爲必要的需求項目。安全程式撰寫是指程式碼的安全性如排除超出陣列定義範圍、除以零的運算式、檔案處理異常狀態等安全漏洞與缺失。

爲了掌控開發作業都能依制度層面進行調整，必須針對各開發階段所實施的安全項目，且以正確性、完整性與一致性等基本特性確認與判定安全項目的執行品質。

(2) 管理層面的配合：

爲了透過管理層面降低安全風險的威脅，應該將 SANS Top-20 Security Risks[23]與 OWASP Top 10[24]等安全弱點納入安全品質檢視的重要項目，安全品質檢視作業可以比照軟體品質確認(Software Quality Assurance; SQA)活動的檢視作業，由三至五人所組成的正式審查機制[20]。每一個軟體開發階段所完成的文件產品，都必通過嚴格的安全品質檢視作業，才能依程序要求交付建構管理(Configuration Management)進行管制，檢視作業若發現安全漏洞或缺失，必須立即找出原因且進行修訂與改善作業，最後還要配合後續的跟摧活動(Follow up)來確認文件產品已完成缺失矯正與漏洞修訂，否則應持續進行修訂與改善作業，直到確認完成缺失矯正與漏洞修訂改善後，才能依程序將文件交付建構管理進行管制，成爲後續開發階段引用的文件。下面以三項管理層面的執行品質來確認與判定管理層面的配合成效：

- 安全文件完成後的核准與繳交等作業程序品質，安全文件漏洞或缺失修訂前後的確認、提領與繳交等作業程序品質。
- 安全文件漏洞或缺失修訂前後的版本架構規劃程序品質、修訂前後內容差異存放程序品質。
- 安全文件漏洞或缺失修訂的日期、修訂的內容、修訂的人員及歸屬的安全項目等作業程序品質。

(3) 技術層面的提昇：

軟體安全品質所強調的特性與 ISO 早期規劃的軟體品質特性有許多落差，爲此本研究提出安全程式碼撰寫、例外處理(Exception Handling)機制、程式碼分析器、安全查核表及安全審核技巧等五個成熟度進行安全軟體開發的技術層面提昇：

- 以安全程式碼撰寫規則、程式模組之間的介面整合確認、程式模組之前置、後置條件確認等技術，提昇安全程式碼撰寫成熟度[6]。
- 在程式碼中適當融入例外處理(Exception Handling)機制，可以避免發生超出陣列定義範圍、除以零的運算式、檔案處理異常狀態等安全漏洞與缺失，有效提昇程式碼的安全品質。
- 有效結合程式碼分析器協助找出的程式碼中隱含的安全漏洞與缺失，可以有效標示出運算式、輸出/輸入介面與模組元件整合介面的安全缺失。
- 安全文件與產品查核表是內部或非正式審查採用的方式，將來至各方面的安全軟體相關資訊彙整、剖析、分類且設定權重後，可以產生具高度修改彈性安全文件與產品查核表，可以協助於開發早期找出潛在的安全漏洞與缺失。
- 萃取安全軟體專家與學者的知識與經驗、蒐集安全漏洞修補的記錄以及記取安全漏洞造成的慘痛經驗等是提昇安全文件與產品審核技巧的關鍵。

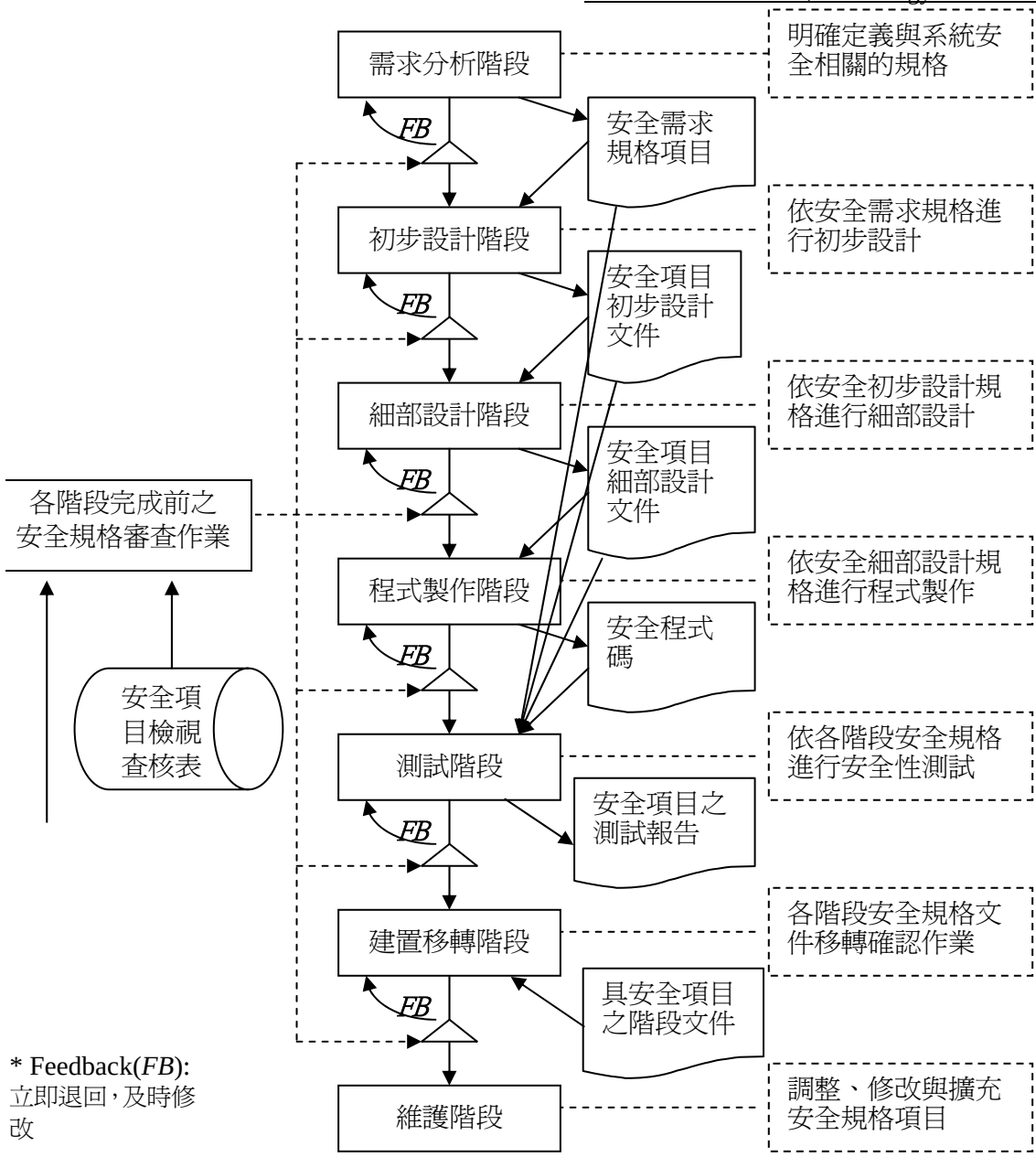
這五項安全技術層面的提昇，可以用較少的人力與成本，減少安全漏洞與缺失注入

產品中，且降低漏洞與缺失遞延至後續階段，有效控制軟體安全問題與缺失。以軟體開發製程爲基礎，結合安全制度層面、安全管理層面與安全技術層面成爲一套安全軟體開發製程(Secure Software Development Process; *SSDP*)，*SSDP* 的運作流程如圖三所示，透過階段性的安全審查作業，可以在開發初期就將安全品質融入軟體產品中，具體提昇軟體系統的安全性。

2. 安全軟體開發製程的關鍵優勢

爲了凸顯對軟體安全品質的重視，安全管制作業應該採取獨立運作，不過，受到人力不足與時程的壓力，也可以將安全管制作業融入軟體開發製程中，整合運作。無論是獨立運作或整合運作，每階段的任務除了依程序的要求完成階段性產品外，還必須依制度要求達成下面的工作內容：

- 需求分析階段：完成的需求規格書中必須涵蓋安全需求規格項目，完成的安全需求規格項目必須通過正式的安全需求規格審查作業，才能發行、交付建構管理進行管制且成爲後續開發作業的遵循依據。其中，安全需求規格項目就是使用單位依系統安全的考量，將安全項目特別融入需求規格書中，成爲後續開發與驗收的依據。
- 初步設計階段：依據通過審查的安全需求規格項目進行架構與介面設計，所完成的安全項目初步設計文件必須通過正式的安全設計審查，才能發行、交付建構管理進行管制且成爲後續開發作業的遵循依據。其中，安全項目初步設計文件就是依據安全需求規格項目的要求，於初步設計過程將安全項目融入初步設計產品文件中，成爲後續開發的依據。
- 細部設計階段：依據通過審查的初步設計文件進行細部設計，所完成的安全項目細部設計文件必須通過正式的安全細部設計審查，才能發行、交付建構管理進行管制且成爲後續開發作業的遵循依據。其中，安全項目細部設計文件就是依據安全初步設計文件，於細部設計過程將安全項目融入細部設計產品文件中，成爲後續開發的依據。
- 程式製作階段：依據通過審查的細部設計文件進行程式撰寫，所完成的安全程式碼必須通過正式的安全程式審查，才能發行、交付建構管理進行管制且成爲測試作業的遵循依據。其中，安全程式碼就是依據安全細部設計文件，於程式設計過程將安全項目融入程式中，成爲後續測試的依據。
- 測試階段：依據通過審查的安全需求規格及各階段文章進行各項細部測試作業，所完成的安全項目測試結果報告必須通過正式的安全測試作業審查，交付建構管理進行管制且成爲維護運作階段引用的文件。其中，安全測試報告是依據各階段安全項目文件進行各項安全測試步驟後，所撰寫完成的一份報告，安全測試報告爲更正安全缺失與修補漏洞的依據。
- 建置移轉階段：依據需求規格書與雙方認定的合約書內容，進行各階段文件的繳交作業，完成建置移轉作業。其中繳交的各階段文件必須涵蓋安全需求規格項目、安全項目細部設計、安全項目細部設計、安全程式碼及安全項目測試報告等內容。



圖三：安全軟體開發製程的運作流程

每階段所完成的文件必須透過安全項目檢視查核表且配合法則式安全知識庫進行嚴格的安全審查作業，一旦發現安全缺失或漏洞，應立即退回前置階段，且隨即進行改善，直到通過安全品質審查活動的要求才能進入下個開發階段。

結合安全制度、安全管理與安全技術的安全開發製程，在安全軟體的開發過程中，可以具體達成下面六項關鍵的優勢(參閱表二所示)：

- 標示軟體安全缺失的時段：需求分析階段就已經考量各類型的安全需求項目，因此後續階段的開發作業，若不能依據安全需求項目完成階段性的工作，將可透階段結束前的審查活動及時標示安全漏洞與缺失，
- 改善軟體安全缺失的作業時段：當審查活動標示出的安全漏洞與缺失，必須立即提出矯正與改善措施，安全漏洞與缺失未來獲得矯正或改善之前，將不允許繳交完成的產品文件，更不能進入下個開發階段。

- 各階段軟體開發安全管控的調整彈性：資訊運作環境與技術持續的成長，使得軟體系統遭受安全漏洞與缺失的威脅也不斷的更新，本文提出的結合安全規格知識庫的法則式改善作業，可以彈性加入新的安全規格知識且有效的調整安全管控法則，具體提昇安全管控的彈性。
- 軟體安全改善作業的人力投入：安全與缺失及時被標示出，可以大幅縮減改善作業的人力投入。
- 軟體安全改善作業的時間花費：安全漏洞與缺失立即提出修訂與改善措施，可以避免漏洞與缺失擴大或遞延至後續階段，可以減少改善作業的時間花費。
- 軟體安全缺失的影響範圍：安全漏洞與缺失及時被標示出，且立即提出矯正與改善措施，因此漏洞與缺失擴大或遞延至後續階段的狀況大幅減少，軟體安全缺失的影響範圍可以有效掌控。

表二：安全軟體開發製程與一般軟體開發製程的特性比較表

軟體開發製程 / 重要特性	安全軟體開發製程	一般軟體開發製程
安全缺失的標示時段	開發初期	測試或維護階段
安全改善作業時段	開發初期	測試或維護階段
安全管控的調整彈性	高	低或無
改善作業之人力投入	少	多
改善作業之時間花費	低	高
安全缺失的影響範圍	小	大

當軟體系統開發計畫依 *SSDP* 完成安全軟體系統開發作業後，接著便是找出 *SSDP* 存在的問題與缺失，且針對這些問題與缺失進行剖析，再提出調整與修訂作業，持續不斷的改善 *SSDP*，才能確認 *SSDP* 的有效性且維持 *SSDP* 的實用狀態。

五、安全軟體開發製程的品質量測模式與改善方式

一套完善的開發製程必須配合環境與需求不斷擴充與調整，以品質量測模式監控與持續改善 ***SSDP*** 執行上的問題與缺失，才能確認 ***SSDP*** 的有效性。

1. 安全軟體開發製程的品質量測模式

個別的量度或量測值只能評量作業品質的某些特質，爲了監控及評量軟體製程品質，必須將個別的量度或量測值做適當的結合。量度結合的方式可以分爲線性結合(Linear Combination)[5][11][16] 與非線性結合(Nonlinear Combination)[7][16]，考量實用性、修改彈性、擴充性與簡單性，本文以線性結合方式建立品質量測模式。影響 *SSDP* 作業品質的三個關鍵項目分別爲制度層面的調整、管理層面的配合及技術層面的提昇，每個關鍵項目的影響指標則是由一些低階的品質因子所組成，透過線性結合公式，可以將高度相關性的基層因子結合成特定量測值，這些特定項目量測值可以進一步結合成高階項目量測值，最後再將高階項目量測值加以結合，而成爲 ***SSDP*** 作業品質量測指標，

以下分四個步驟說明。

(1) 制度層面品質量測值(System Level Quality Measurement; SLQM)：進行制度層面品質量測之前，必須分別先以線性結合方式影響安全需求項目於各階段實施的正確性、完整性及一致性等影響因子加以結合，產生使用單位安全需求、安全運作環境需求、安全設計介紹需求及安全程式撰寫等品質量度，再透過公式(1)，將四項品質量度結合成制度層面品質量測值，結合公式如下所示：

SLQM: System Level Quality Measurement

SRQU: Security Requirements Quality of User	W_{qu}*: Weight of SRQU*
SRQE: Security Requirements Quality of Environment	W_{qe}*: Weight of SRQE*
SRQI: Security Requirements Quality of Interface Design	W_{qi}*: Weight of SRQI*
SRQC: Security Requirements Quality of Coding	W_{qc}*: Weight of SRQC*

$$SLQM = W_{qu}*SRQU + W_{qe}* SRQE + W_{qi}* SRQI + W_{qc}*SRQC$$

$$W_{qn} + W_{qe} + W_{qi} + W_{qc} = 1 \qquad (1)$$

(2) 管理層面品質量測值(Management Level Quality Measurement; MLQM)：進行管理層面品質量測之前，必須先分別以線性結合方式將安全文件的存取監控、版本管制及修訂記錄等三項影響因子，結合成安全文件的存取監控、版本管制及修訂記錄等三項品質量度值，再透過公式(2)，將三項品質量度值結合成管理層面品質量測值，結合公式如下所示：

MLQM: Management Level Quality Measurement

SDACQ: Secure Documents Access Control Quality	W_{ac}*: Weight of SDACQ*
SDVCQ: Secure Documents Version Control Quality	W_{vc}*: Weight of SDVCQ*
SDRRQ: Secure Documents Revision Record Quality	W_{rr}*: Weight of SDRRQ*

$$MLQM = W_{ac}* SDACQ + W_{vc}* SDVCQ + W_{rr}* SDRRQ \qquad W_{ac}+W_{vc}+W_{rr}=1 \qquad (2)$$

(3) 技術層面品質量測值(Technology Level Quality Measurement; TLQM)：進行技術層面品質量測之前，必須先分別以線性結合方式將安全程式撰寫、例外處理機制、弱點掃瞄工具、安全漏洞查核表與安全漏洞檢視等五項成熟度之影響因子加以結合，且對映產生全程式撰寫、例外處理機制、弱點掃瞄工具、安全漏洞查核表與安全漏洞檢視等五項成熟度品質度值，再透過公式(3)，將五項品質量度值結合成技術層面品質量測值，結合公式如下所示：

TLQM: Technology Level Quality Measurement

SCM: Secure Coding Maturity	W_1*: Weight of SCM*
EHM: Exception Handling Maturity	W_2*: Weight of EHM*
VSM: Vulnerability Scanner Maturity	W_3*: Weight of VSM*
SVCLM: Security Vulnerability Check Lists Maturity	W_4*: Weight of SVCLM*
SVIM: Security Vulnerability Inspection Maturity	W_5*: Weight of SVIM*

$$TLQM = W_1*SCM + W_2* EHM + W_3* VSM + W_4* SVCLM + W_5* SVIM$$

$$W_1 + W_2 + W_3 + W_4 + W_5 = 1 \qquad (3)$$

(4) 最後透過公式(4)，將系統層面、管理層面及技術層面等三項高階品質量測值結合成 SSDP 品質量測指標(SSDP Quality Measurement Indicator; SSDPQMI)，結合公式如下所示：

SSDPQMI: SSDP Quality Measurement Indicator

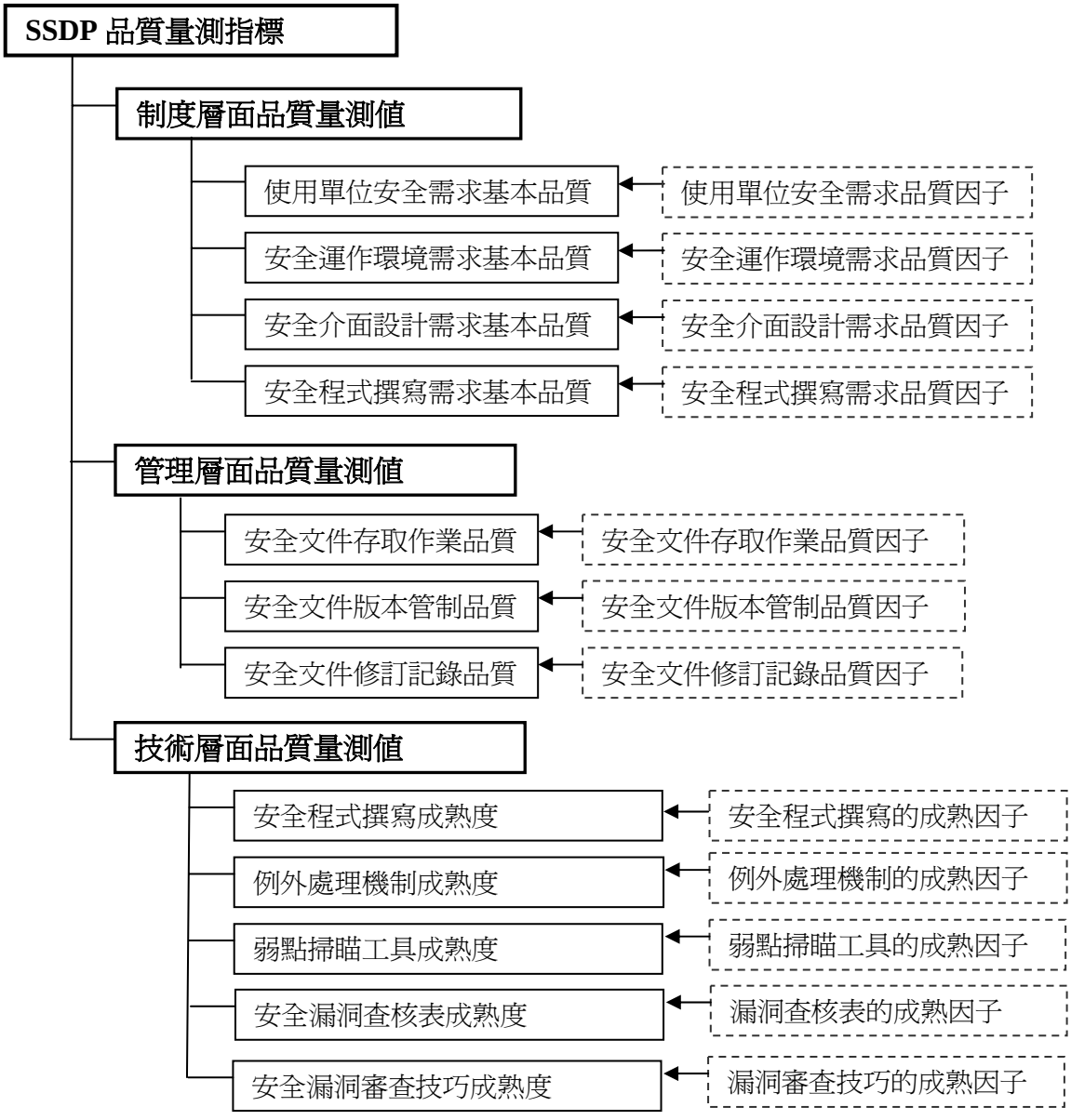
SLQM: System Level Quality Measurement W_{sl}*: Weight of SLQM*

MLQM: Management Level Quality Measurement W_{ml}*: Weight of MLQM*

TLQM: Technology Level Quality Measurement W_{tl}*: Weight of TLQM*

$$SSDPQMI = W_{sl}*SLQM + W_{ml}*MLQM + W_{tl}*TLQM \qquad W_{sl}+W_{ml}+W_{tl}=1 \qquad (4)$$

本品質量測模式由開發製程的子工作項匯集了 12 組品質因子，結合成 12 項基層品質量度值及 3 項高階品質量測值，經過三個層次的量度結合作業，產生 ***SSDP*** 品質量測指標，本研究稱此量測作業模式爲 *SSDP* 品質量測 (***SSDPQM***) 模式，其架構如圖四所示。



圖四：*SSDP* 品質量測模式架構圖

二、安全軟體開發製程改善方式

品質量測模式所估算出的 *SSDP* 品質量測指標，是找出 *SSDP* 潛在問題與缺失的依據，整合各個層面的品質因子、基層品質量度及高階品質量測值等三個層次，結合而成的品質量測指標是找出 *SSDP* 潛在問題與缺失的關鍵。因此，當品質量測指標落在「過低」的範圍時，便可以從品質量測模式的結合公式進行推導，判斷出相關的品質因子，再從品質因子對映剖析出 *SSDP* 的子工作項，進而找出 *SSDP* 實行過程中潛在的問題與缺失，依據問題與缺失可以提出具體的調整措施與監控作業。以下即針對 *SSDP*「潛在問題與缺失」所提出的改善法則：

<Rule 1> IF「*SSDP* 品質量測指標」未能通過品質門檻
THEN 進一步分析「制度層面品質」、「管理層面品質」及「技術層面品質」等量測值是否落在過低狀態，且透過結合公式(4)，對照找出屬於過低狀態的品質特性量測值。

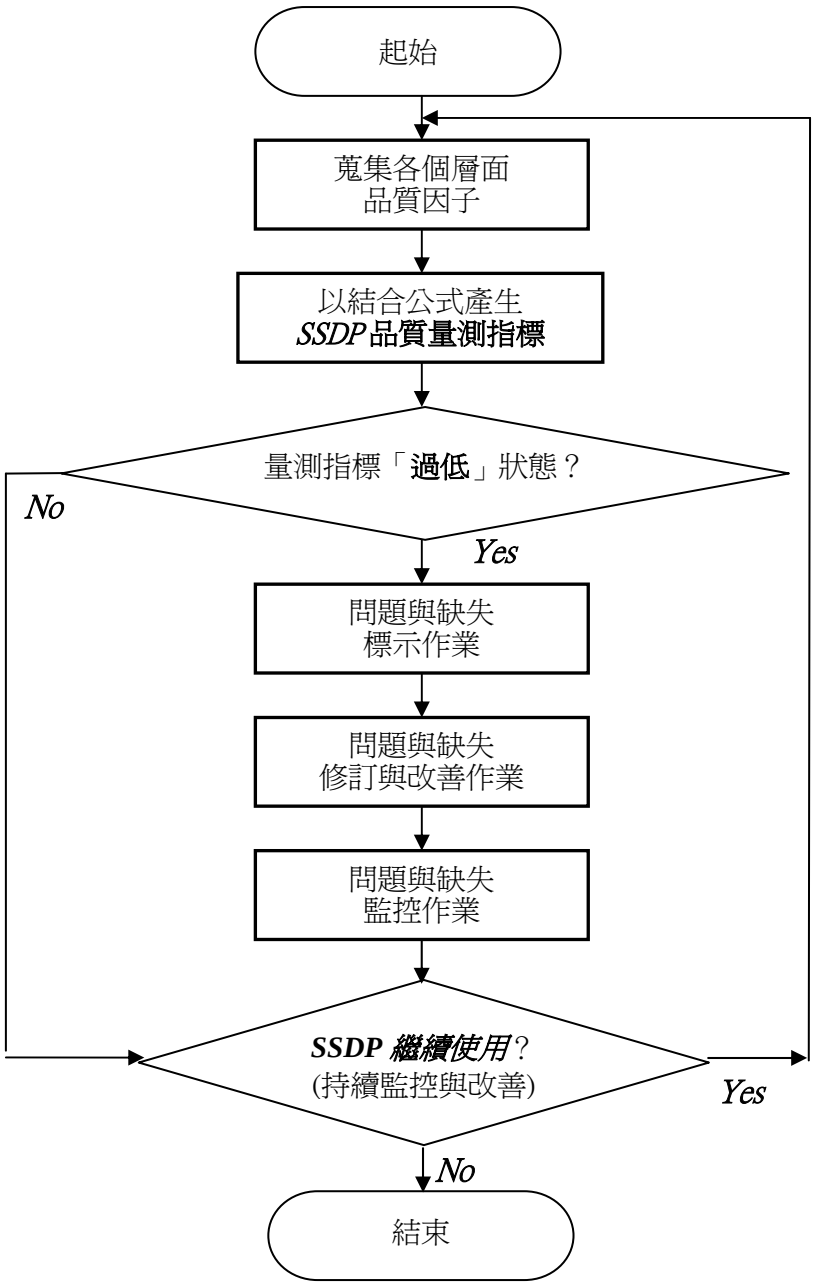
<Rule 2> IF「制度層面品質」量測值屬於「過低」狀態
THEN 可以進一步分析出那幾項基層品質不良造成「制度層面品質」量測值屬於「過低」狀態中，且透過結合公式(1)，對照找出使用單位安全需求、安全運作環境需求、安全設計介紹需求及安全程式撰寫等品質量度值所對應的正確性、完整性及一致性等品質因子，再由品質因子配合找出制度層面相關子工作項潛在的問題與缺失，進行修正、改善與監控等措施。

<Rule 3> IF「管理層面品質」量測值屬於「過低」狀態
THEN 可以進一步分析出那幾項基層品質不良造成「管理層面品質」量測值屬於「過低」狀態中，且透過結合公式(2)，對照找出品質量度值所對應的安全文件存取監控、版本管制及修訂記錄等品質因子，再由品質因子配合找出安全文件存取監控、版本管制及修訂記錄等子工作項潛在的問題與缺失，進行修正、改善與監控等措施。

<Rule 4> IF「技術層面品質」量測值屬於「過低」狀態
THEN 可以進一步分析出那幾項品質不良造成「技術層面品質」量測值屬於「過低」狀態中，且透過結合公式(3)，對照找出安全程式撰寫、例外處理機制、弱點掃瞄工具、漏洞查核表及漏洞檢視技巧等品質量度值所對映的成熟度之品質因子，再由品質因子配合找出安全程式撰寫、例外處理機制、弱點掃瞄工具、漏洞查核表及漏洞檢視技巧等子工作項成熟度潛在的問題與缺失，進行修正、改善與監控等措施。

以品質量測爲基礎的持續改善作業流程(如圖五所示)，可以標示出 *SSDP* 實行過程中潛在問題與缺失，再配合及時的修訂、改善與監控作業，可以具體提昇 *SSDP* 運作品質，有效監控與不斷改善安全開發製程的問題與缺失，確保 *SSDP* 的有效性與最佳狀態，

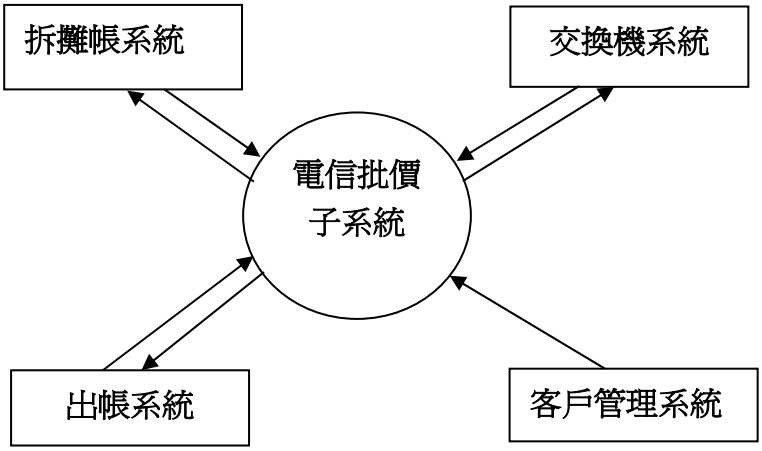
以持續強化軟體系統的安全性。



圖五：以品質量測爲基礎的 ***SSDP*** 持續改善作業流程

六、以軟體系統開發案例說明 SSDP 的運作與改善流程

本節以電信批價子系統說明本研究提出的 *SSDP* 運作流程與改善作業，電信批價子系統是電信帳務系統的核心作業，其主要任務就是進行電信通話費用的計價工作，從系統環境圖(Context Diagram)中涵蓋的交換機系統、客戶管理系統、拆攤帳系統及出帳系統等四個外部實體(參閱圖六所示)，可以瞭解其重要性與複雜度。電信批價子系統的四個主要工作項目分別爲：通話記錄匯入、異常記錄剔除、通話費用計價及通話費用匯出等作業。以下針對電信批價子系統採取 SSDP 的運作步驟進行說明：



圖六：電信批價子系統之環境圖

(1) 採取 *SSDP* 制度的規範：電信批價子系統需求分析階段必須將使用單位的安全需求列爲安全需求項目，全盤性的安全考量應包括客戶基本資料的安全性、客戶通話記錄的安全性、異常記錄的判定規則的安全性、話費資料的安全性與作業時程的安全性等，至於四個主要作業的安全需求項目說明如下：
   - 通話記錄匯入作業：依預定時程將交換機累積一段時間的通話記錄透過網路傳輸方式匯入，且完整記載匯入的機房、時間、記錄時段等相關資料。此項作業應提出資料庫、儲存媒體及網路傳輸等安全需求項目。
   - 異常記錄剔除作業：依異常通話記錄的判定規則剖析匯入的通話記錄，挑出且記載所有異常通話記錄，從通話記錄中剔除且進行異常話費計價、記錄時段等相關資料。此項作業應擬定資料庫、檔案輸出/入及檔案儲存等安全需求項目。
   - 通話記錄計價作業：依客戶承租的服務項目與費率，詳細計算段用戶每通的通話費用，且依通話時間、日期與種類等相關屬性暫時存放。此項作業應著重程式碼的安全性如排除超出陣列定義範圍、除以零的運算式、檔案處理異常狀態等安全漏洞與缺失，也必須考量系統、功能模組及元件等介面整合的安全性。
   - 通話記錄匯出作業：依排定的時程將計價完成的通話記錄透過網路傳輸方式，分別匯出至出帳系統與拆攤帳系統，且完整記載匯出的時間、記錄時段等相關資料。此項作業應擬定資料庫、儲存媒體及網路傳輸等安全需求項目。

(2) 配合 SSDP 的管理層面：電信批價子系統需求分析階段所擬定的安全需求項目，必須在後續的開發階段一一落實，每完成一項開發文件或產品，都必通過嚴格的安全品質檢視作業，才能依程序要求交付建構管理進行管制，檢視作業若發現安全漏洞或缺失，必須立即找出原因且進行修訂與改善作業，直到確認完成缺失矯正與漏洞修訂後，才能依程序將文件交付建構管理進行管制，成爲後續開發階段引用的文件。

(3) 導入 *SSDP* 的技術層面：電信批價子系統開發作業進入設計與程式製作階段時，應適時將安全程式碼撰寫、例外處理機制、程式碼分析器、安全查核表及安全審核技巧等安全設計與安全程式撰寫技術注入相關的階段產品與文件中，以具備階段性產品與文件的安全品質。

電信批價子系統採用 SSDP 的實作過程中，可以透過問卷、訪談、統計與分析等方

式，蒐集制度層面、管理層面與技術層面等作業的品質相關因子，量化後的品質因子依 SSDP 品質量測模式所定義之公式，先產生低階的品質量度，再產生高階的品質量測值，最後估算出 SSDP 的品質量測指標。量化的品質量測指標配合本研究提出的改善法則，可以找出 SSDP 實作過程所隱含的問題與缺失，再透過 SSDP 持續改善作業流程進行問題與缺失的修訂、調整與監控，可以持續強化 SSDP 的運作品質，進而具體提昇以 SSDP 開發之軟體系統的安全性。

七、結論

受到軟體開發人力與時程的影響，軟體安全品質一般都被列爲事後考慮甚至被忽略的項目，使得安全缺失與漏洞隨著開發作業被注入產品中，大幅擴增軟體系統的安全危機。造成軟體安全漏洞的原因很多，卻幾乎都與軟體開發作業的疏忽及運作環境的缺失等問題密切相關，其中又以開發作業疏忽導致的安全缺失，影響最大且最久。爲了減低安全漏洞與缺失的衝擊，必須強化軟體製程的安全機制，於開發初期就有效標示出安全缺失與漏洞，且及時提出修訂與改善作業，勢必可以有效的降低安全問題的危害，提昇軟體系統的安全品質。本文針對未考量安全品質的軟體開發作業進行探討，明確的描述延後發現安全缺失與漏洞的原因，且深入剖析安全缺失與漏洞無法及時發現，造成安全問題持續擴大且往後續遞延的危機，安全軟體管制作業是及時標示安全缺失且提出改善措施的關鍵，更可以有效降低安全缺失造成的衝擊與損失。本文從制度、管理與技術三個層面爲基礎，結合安全檢視查核表與法則式安全知識庫，規劃出一套安全軟體開發製程(***SSDP***)，從管理與技術層面對軟體開發制度進行調整，在軟體開發初期就能有效的標示出軟體安全缺失與漏洞，隨即以較低的人力、時間與成本進行安全缺失的改善作業，將安全品質融入產品中。此外爲了確認 ***SSDP*** 的有效性，本文更提出一套 ***SSDP*** 品質量測(***SSDPQM***)模式，用以監控與持續改善 ***SSDP*** 執行上的問題與缺失，確保 ***SSDP*** 能夠持續強化軟體系統的安全性。***SSDP*** 品質量測模式以制度、管理與技術三個層面爲基礎，採取線性的量度結合模式簡化了複雜的公式，具有高度的修改彈性與擴充能力，可以隨著 ***SSDP*** 的變動進行快速調整，持續確認 ***SSDP*** 的有效性與最佳狀態。

[謝啓]

本論文接受實踐大學 99 學年度「專題研究(蓄積管院研發能量)計畫」(計畫編號：USC-99-05-04004)之補助。

參考文獻

[1] 賴森堂，“以介面設計爲基礎的軟體安全品質量測模式”，*中華民國品質學會第四十三屆年會暨第十三屆全國品質管理研討會論文集*，中華民國品質學會，2007。

[2] 賴森堂，“安全軟體建制基礎—軟體安全特性架構之研究”，*2007 數位科技與創新管理研討會論文集*，華梵大學，2007。

[3] Apvrille, A. and Pourzandi, Makan, “Secure Software Development by Example,” *IEEE Security & Privacy*, vol. 3, (4), 2005: pp. 10-17.

[4] Bishop, M., *Computer Security: Art and Science*, Addison-Wesley, 2003.

[5] Boehm, B.W., *Software Engineering Economics*, Prentice-Hall, New Jersey, 1981.

[6] Bush, W.R., Pincus, J.D. and Siela, D. J., “A Static Analyzer for Finding Dynamic Programming Errors,” *Software Practice and Experience*, 30, (7), June 2000: pp. 775-802.

[7] Conte, S.D., Dunsmore, H.E. and Shen, V.Y., *Software Engineering Metrics and Models*, *Benjamin/Cummings*, Menlo Park, 1986.
[8] Cowan, C., "Software Security for Open-Source Systems," *IEEE Security & Privacy*, vol. 1, no. 1, 2003: pp. 38-45.
[9] Evans, D. and Larochelle D., "Improving Security Using Extensible Lightweight Static Analysis," *IEEE Software*, January/February 2002: pp.42-51.
[10] Fairley, R., *Software Engineering Concepts*, McGraw-Hill, Inc., 1985.
[11] Fenton, N.E., *Software Metrics - A Rigorous Approach*, Chapman & Hall, 1991.
[12] McGraw, G., *Software Security – Building Security In*, Addison-Wesley 2006.
[13] McGraw, G., "Software Security", *IEEE Security and Privacy*, March 2004: pp.80-83.
[14] Hall, A. and Roderick Chapman, "Correctness by Construction: Developing a Commercial Secure System," *IEEE Software*, January/February 2002: pp.18-25.
[15] Howard, M. and D. Le Blanc, *Writing Secure Code*, Microsoft Press, 2002.
[16] Lai, S. T., "A Quality Measurement Model for Software Maintenance", *Proceeding of the Second World Congress on Software Quality (2WCSQ)*, Japan, September 2000.
[17] Leveson, N. G., *Safeware: System Safety and Computers*, Addison-Wesley, 1995.
[18] Pressman, R. S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill, New York, 2010.
[19] Redwine, S.T. and N. Davis, eds., *Processes to Produce Secure Software*, National Cyber Security Summit, 2004.
[20] Schach, S. R., *Object-Oriented Software Engineering*, McGraw-Hill Companies, 2008.
[21] Viega, J. and G. McGraw, *Building Secure Software*, Addison-Wesley, 2002.
[22] CERT/CC (http://www.cert.org/stats/cert_stats.html)(2010/6).
[23] SANS Top-20 2007 Security Risks (http://www.sans.org/top20/) (2010/06).
[24] OWASP Top 10 (http://www.owasp.org.tw/blog/) (2010/06).