CH13 建立資訊系統

1. 建立新系統為何產生組織變革？

描述因 IS 開發造成的 4 種組織變革 – 電腦化、合理化、BPR、典範轉移
定義 BPM 並描述其步驟 – 持續地分析修改優化流程並最佳化；確認、分析、設計、執行、持續評量

2. 系統開發過程中的核心活動是什麼？

區分系統分析和系統設計 – 定義問題、解決方案、滿足需求、財務技術組織之可行性分析；總計畫或模型、詳細規範
定義資訊需求並解釋為什麼難以正確地決定 – 新系統或修改後系統之目標；流程定義不明，工作者意見分歧
解釋為什麼系統開發的測試階段如此重要並描述 3 個階段 – 確定系統是否正確運作；單元測試、系統測試、驗收測試
描述 programming, conversion, production, and maintenance 在系統開發中的作用 - coding、新舊系統之轉換、確定系統程度、維持其正確運作

3. 系統建模和設計的主要方法是什麼？

比較 OOD 和傳統結構化(traditional structured)方法 – 物件導向(UML)；流程導向，資料與流程分開(DFD)
描述 computer-aided software engineering 並解釋其目的 – 自動化開發減少重複工作的工作

4. 建立資訊系統的替代方法是什麼？

SDLC(traditional systems life cycle)與其優缺 – 適合大型專案、按順序進行、嚴謹、結構化；緩慢、不靈活、昂貴、適合小型系統
Prototype 描述步驟與其優缺限制 – 適合需求不明確無法預先確定不清楚的方案、成本相對 SDLC 低、較彈性；可能忽略某些步驟、無法容納大量資料人數
應用套裝軟體與其優缺 – 通用流程、減少測試時間、技術問題、內部成本、提供文件；轉換成本、可能無法滿足特殊需求
End-User Development 與其優缺並列出管理政策和程序 – 快速完成、高用戶與度滿意度；不適合大量交易、缺乏管理合理文件測試
Outsource 優缺 – 營運開發維護移交給外部服務商；減少組織成本、發展核心能力；評估、轉換、監督成本

5. 在數位公司時代，有什麼新的系統建構方法？

說明快速應用程序開發（RAD）、敏捷開發(agile development)和 DevOps –

Interactive、Prototype；Interactive、漸進式開發與交付、分解成小專案、面對面；基於敏捷開發，強調開發人員與營運人員之合作

漸增模式

螺旋模型

雲端運算 –

解釋基於元件的開發和 Web 服務如何幫助公司構建和加強資訊系統 – 通用功能之物件組合；通用標準

解釋行動應用程序開發和響應式 Web 設計的功能

描述因 IS 造成的 4 種組織變革

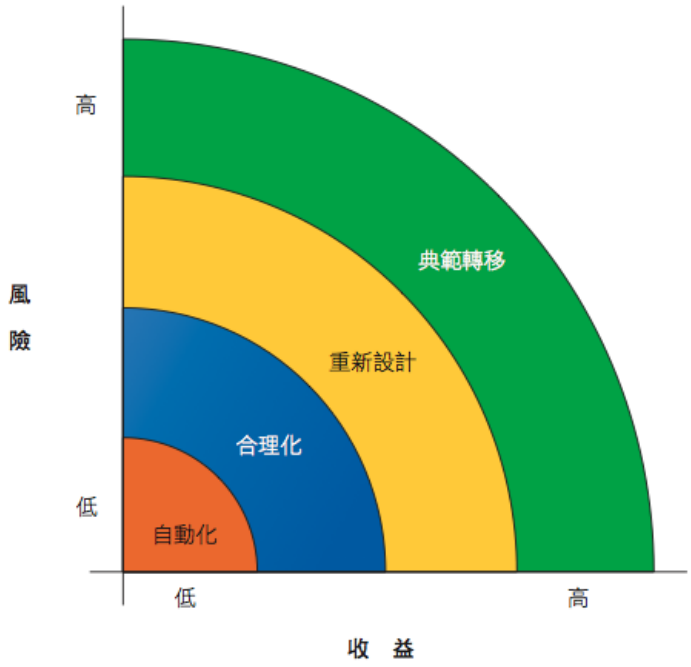
(1) **自動化(Automation)**：低風險，低回報，更高效地執行任務 → POS、TQM、Six Sigma

(2) **合理化(Rationalization)**：中等風險，中等回報，涉及簡化標準操作程序，使作業更平順更有效率→ BPM

(3) **企業流程再造(BRP)**：更高的風險，更高的回報，組織革命性地重新組織了工作流程，以改善速度、服務和品質 → 從順序流程改成平行處理

(4) **典範轉移(Paradigm shift)**：最高風險，最高回報，改變組織如何進行商業活動甚至商業性質，被突破性的新模式所完全取代的一種現象，非連續性、跳躍式的轉換 → 社群化、O2O

圖 13-1 組織變革帶來的風險和報酬



組織變革最常見的是自動化與合理化，較慢速的變革策略代表適度的收益與較少的風險。快速和較全面的變革——像重新設計和典範轉移——往往帶來較高報酬，但失敗機率也相對較高。

[補充]
Total quality management (TQM)：需進行一系列持續地改善
Six sigma：利用各種品管、統計與管理科學的方法論，來有效的辨識與移除流程中潛在的錯誤與瑕疵點(Defects and Errors)，並將產品製造與管理流程的變異(Variability)降至最小，追求產品品質的穩定與不斷的改善稱之
(1) 確認 (Define)
(2) 衡量 (Measure)
(3) 分析 (Analyze)
(4) 改進 (Improve)
(5) 控制 (Control)

定義 BPM，並描述其步驟

企業流程管理：提供工具與方法分析，修改、優化內部眾多企業流程，是持續地改進並將流程作為建構 IS 時的模塊

(1) **確定變更流程(Identify processes for change)**：確定要修改流程的是對的，不然等於做白工
(2) **分析現有流程(Analyze existing processes)**：組織需理解和衡量現有流程的績效並將其作為基準，包括投入、產出、資源和活動順序
(3) **設計新流程(Design the new process)**：設計之新流程可以進行記錄和建模並與舊流程進行比較，需顯示減少了多少時間和成本、增加多少價值或提高客戶服務和價值來證明其合理性
(4) **實施新流程(Implement the new process)**：新流程轉換為一組新的程序和工作規則，開始使用後問題可能被員工發現並解決或是員工提出建議
(5) **持續評估(Continuous measurements)**：員工可能退回使用舊方法或企業經歷其他事情而惡化

區分系統分析和系統設計並描述每個活動

系統分析(System Analysis)：是組織打算利用資訊系統解決問題時所進行的分析，包括定義問題、確定問題的原因、提出解決方案以及確定解決方案必須滿足使用者之資訊需求，需從財務、技術和組織角度確定解決方案是否可行

系統設計(System Design)：展示系統將如何實現系統分析的目標，是系統的總體計劃或模型，包含所有該系統的詳細規範，其概述了系統解決方案的管理、組織和技術元件

定義資訊需求並解釋為什麼難以正確地決定

資訊需求 ： 包括確定誰需要什麼資訊、在哪裡，何時何地以及如何取得，定義了新系統或修改後系統的目標，並詳細描述了新系統必須執行的功能

原因 ： 由於流程定義不明確以及工作者之間的意見分歧，確定資訊需求可能很困難

錯誤的分析 → 導致系統故障和高昂系統開發成本的主要原因

解釋為什麼系統開發的測試階段如此重要並描述三個階段

測試 ： 確定系統是否會正確運作的方法

(1) **Unit testing**： 指分別測試或檢查單個程式
(2) **System testing**： 對整個系統進行測試，以確定程式模組是否依計劃相互作用
(3) **Acceptance testing(接受度測試)**： 系統經過最終用戶的最終認證，以確保它滿足既定要求並已準備好進行安裝

描述 programming, conversion, production, and maintenance 在系統開發中的作用

(1) **Programming**： 將設計階段的規範實作為軟體程式碼
(2) **Conversion :** 舊系統轉換為新系統的過程

	定義	適用	優點	缺點
直接切換（direct cutover）	將原有系統整個廢除，啟用新系統	規模小、風險低，重要性不高的系統	最省錢	風險過高
平行策略（parallel strategy）	將資料同時交由新、系統進行處理	重要性高、較為複雜的系統	較為可靠、風險最低	維護成本高
階段性轉換(phased approach）	系統分成多個子系統，轉換時針對 module 逐一抽換	當系統規模大，而且新系統與舊系統架構相同時	系統建置、轉換成本較低	系統不一定容易分割成邏輯模組
前導轉換(Pilot conversion)	先由一部門試用（白老鼠），若使用狀況良好再全面	風險高，規模大的系統	轉換風險，較直接轉換低	系統試用時程難以掌控

	推廣到各部門			

(3) **Production(上線)：** 運作安裝與轉換完成後的系統，用戶和技術專家都將在生產過程中對系統進行審查，以確定其達到原目標要求之程度，並確定是否需要任何修訂或修改

(4) **Maintenance**：對生產系統的硬體、軟體、檔案或程序進行修改，以糾正錯誤，滿足新需求並提高處理效率

表 13-2 系統開發

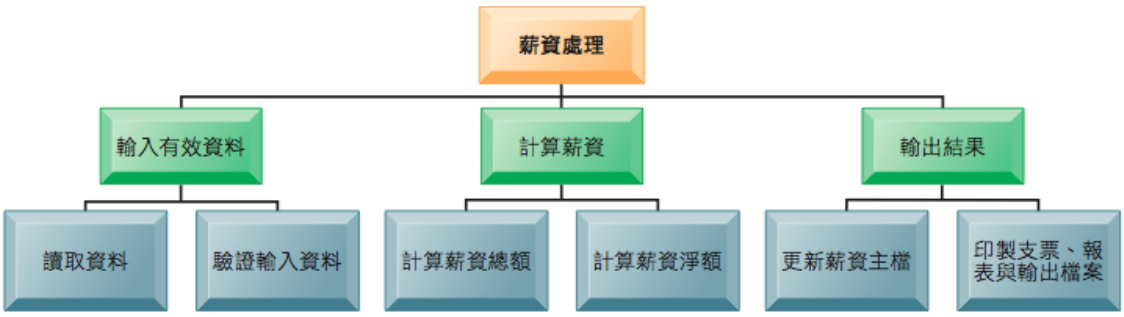
主要活動	說　明
系統分析	確認問題 提出解決方案 建立資訊要求
系統設計	產生設計規格
程式設計	將設計規格轉為程式碼
測　試	執行單元測試 執行系統測試 執行驗收測試
轉　換	計畫轉換 準備文件 訓練使用者及技術人員
上線使用及維護	計畫轉換 準備文件 訓練使用者及技術人員

比較 OOD 和傳統結構化(traditional structured)方法

結構化方法：流程導向(process-oriented) 主要著重於在資料流經系統時對其抓取、存儲、操作和傳遞資料的過程或行為進行建模，建模過程中將流程與資料分開

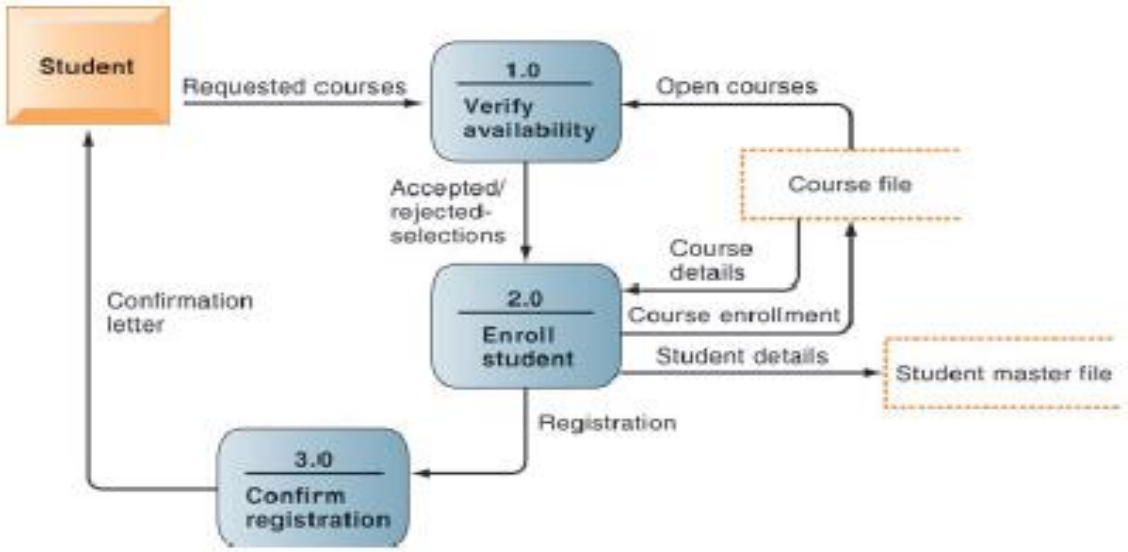
→Data flow diagram (DFD)、資料字典(Data dictionary)、具體流程(Process specifications)、結構圖(Structure chart)

圖 13-7　薪資系統的高階結構圖



這個結構圖表示薪資系統設計最高階層或是最抽象部分，可以綜覽系統全貌。

FIGURE 13.6　DATA FLOW DIAGRAM FOR MAIL-IN UNIVERSITY REGISTRATION SYSTEM



The system has three processes: Verify availability (1.0), Enroll student (2.0), and Confirm registration (3.0). The name and content of each of the data flows appear adjacent to each arrow. There is one external entity in this system: the student. There are two data stores: the student master file and the course file.

OOD：為物件與物件間的集合

(1) **基本單元 – 物件**：將流程和資料組合到一個物件中

(2) **封裝**：在物件中的資料只能藉由與該物件相關的操作或方法來存取和修改

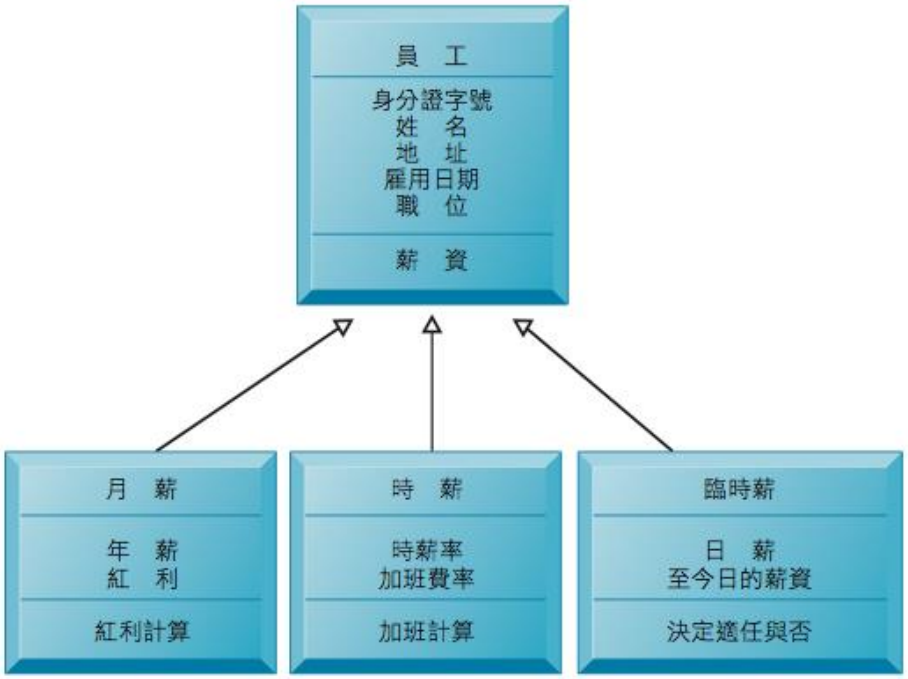
(3) 比傳統的結構化開發更反覆(iterative)與漸進(incremental)

(4) 分析階段要求系統建構者記錄系統的方法(Fuction)需求並具體說明其最重要的屬性

(5) **模組化**：可重複使用

→ UML(Unified Modeling Language；統一塑模語言)

圖 13-8 類別與繼承



本圖說明了類別如何繼承父類別的共同特性。

描述 computer-aided software engineering 並解釋其目的

電腦輔助軟體工程(CASE)：自動化開發並減少重複工作的軟體工具，包括用於產生圖表的圖形工具、報告產生器、報告工具、分析和檢查工具、資料字典、程式碼和文件產生器，藉由自動地調整和更改並提供原型製作工具來支援迭代設計，需要組織紀律確保其有效使用

分類：統一模型化語言 UML、程式碼產生工具資料、模組化工具、軟體重構工、具包含版本控制的組態管理工具

SDLC(traditional systems life cycle)與其優缺

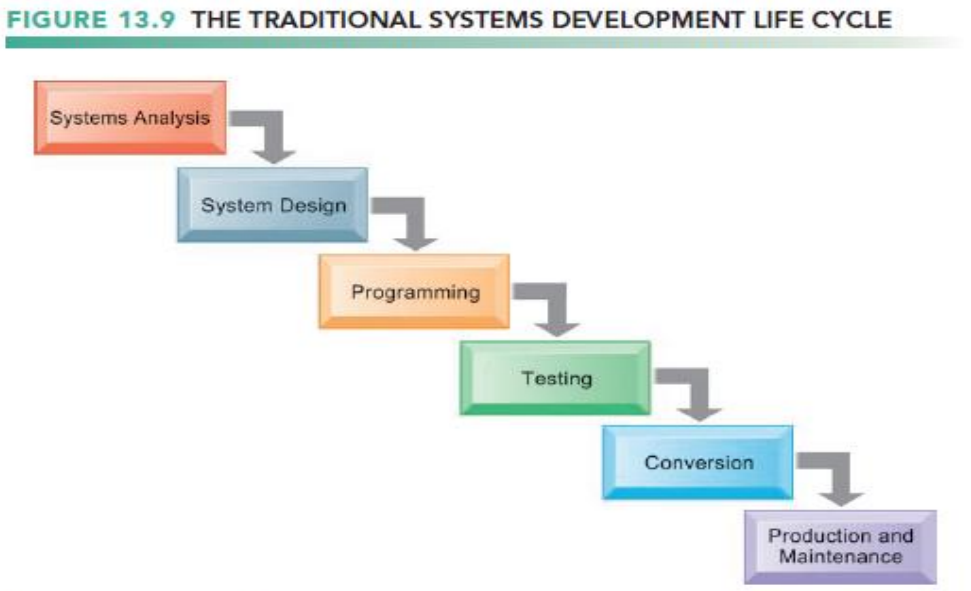
系統發展生命週期 (System Development Life Cycle; SDLC)：是用於管理系統開發的正式方法，仍然是大型專案的主要方法，每個階段基本上須按順序執行，最終用戶和資料處理專家之間要有正式的“簽署”協議，以驗證每個階段是否已完成，該方法緩慢、昂貴、不靈活，不適用於許多小型系統

階段：systems analysis → systems design → programming → testing → conversion → production and maintenance

(1) systems analysis：組織定義需要解決的問題；技術專家辨認問題，收集資訊需求開發可選擇的方案並制定專案管理計畫；企業用戶提供資訊需求，制定財務或操作限制並選擇方案 → 系統需求書、會議紀錄
(2) systems design：技術專家會對設計規範建模和記錄並為解決方案選擇硬軟體技術；用戶則批准該規格 → 系統規格書，會議紀錄
(3) programming：將設計規格轉成軟體程式碼 → 程式
(4) testing：單元測試、系統測試、驗收測試；用戶提供測試方案與情境驗證測試結果 → 測試計畫及測試報告、會議紀錄
(5) conversion：技術專家會準備轉換計劃並監督轉換；用戶評估新系統並決定何時可以將新系統投入生產 → 轉換計畫、評估報告
(6) production and maintenance：技術專家會評估技術性能並執行維護；用戶使用該系統並評估其功能性能 → 安裝說明,、使用者手冊

優：高度結構化、嚴格和正式的需求與規格並在建構過程進行嚴格的控制、適用於建構大型交易處理和管理資訊系統及建構複雜的技術系統

缺：非常昂貴且耗時、缺乏彈性、不適用於非結構化與缺乏明確需求定義的應用程式

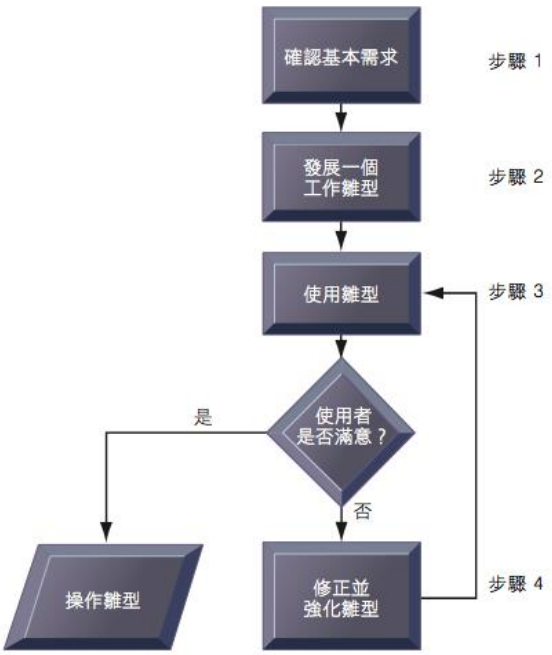


prototype 描述步驟與其優缺限制

雛型設計：是一種互動式系統設計方法，建立系統的實驗模型作為確定資訊需求的一種手段；雛型製作可以快速，廉價地建立實驗系統，以進行展示和評估，從而使用戶可以更好確定資訊需求，並對雛型續持續修改直到完全符合用

戶需求為止，動態地確定資訊需求與設計

圖 13-9 雛型法的流程



開發一個雛型的過程，可分為四個步驟。由於雛型可以快速且低成本的開發出來，開發者可以反覆數次同樣的步驟，步驟 3 與 4 可以重複地來修正及加強雛型的功能，以達到可操作的最終系統。

優：適合用於需求不確定或無法預先完全確定或不清楚合適的解決方案，適合小型簡單的資料操作專案，常用於終端用戶介面設計，相對於 SDLC 成本較低，較靈活具彈性

缺：可能忽略必要步驟(文件與測試)、可能無法容納大量資料或用戶，可能未經過全面測試或記錄

應用程式套裝軟體與其優缺

應用程式套裝軟體：對所有企業組織都是通用的，並且基於具有標準流程的通用功能，這些流程不會隨著時間的流逝而發生很大變化，供應商許滿足用戶所需之功能需求 Request for Proposal (RFP)

優：供應商以建立大多數的設計、程式經過預測試，減少測試時間和技術問題、供應商通常會安裝軟體或協助安裝、定期更新或增強、通常有支援人員減少組織內部成本、提供文件

缺：複雜且已經自動化的系統轉換成本很高、如果無法滿足獨特的需求，則可能需要進行廣泛的自定義或重新撰寫、系統可能無法僅在一個套裝軟體中良好地執行許多功能

[補充]
雲端 Software Services

end-user development 與其優缺並列出管理使用者自建系統的政策和程序

end-user development(使用者自建)：允許最終用戶開發簡單的資訊系統，只需很少或不需技術專家的從旁協助，減少完成應用程序所需的時間和步驟

優：沒有使用者溝通與抗拒問題、降低 MIS 部門負擔、提升系統設計創意、沒有系統開發等待時間、提高生產力、更快完成專案、高用戶參與度和滿意度

缺：缺乏嚴謹的規劃、需設立專門支援技術、須注意基本資料庫的安全、需有良好的教育訓練、MIS 部門的本位主義、執行評估需求問題、無法輕鬆地處理大量交易和具有大量程邏輯和具有廣泛程序邏輯和更新要求的應用程序、缺乏管理和控制、測試、文件

適用範圍：
(1) 不適用複雜、大規模、要求嚴謹的控制、安全及效率的系統
(2) 適用 EUC 的應用系統

管理策略和過程：
(1) 組織須為最終用戶計算建立足夠的支援設施
(2) 培訓和支援受訓人員滿足其特殊需求
(3) 應合併到組織的戰略計劃

控制：
(1) 成本合理性
(2) 軟硬體標準
(3) 公司內的微型計算機，文字處理軟體、資料庫管理系統、繪圖軟體以及查詢和報告工具標準
(4) 品質審查
(5) 最終用戶開發的應用程序控制，包括測試、文件、輸入和更新、備份、恢復和監督的準確性和完整性
(6) 註記向其他重要系統提供資料的關鍵應用程序並遵循更嚴格的標準

[補充]
Domestic outsourcing

outsource 優缺

外包：將組織的計算機中心運營、電信網路或應用程式開發移交給提供這些服務的外部供應商的過程，被視為控製成本或開發應用程序的一種方式

優：
(1) 專注於公司之核心能力
(2) 提升 IS 品質
(3) 降低投資風險
(4) 解決 IT/IS 資源系統不足
(5) 減少長期投資成本

缺：
(1) 確定和評估、過渡到新供應商之成本
(2) 監督供應商以確保它們滿足合約要求
(3) 文化差異
(4) 與供應商需求溝通問題
(5) 易受委外廠商控制
(6) 打擊資訊部門員工士氣
(7) 機密安全保護問題

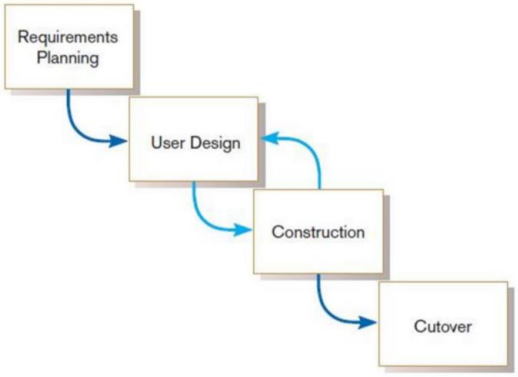
關鍵成功因素：
(1) 雙方的合作態度與溝通程度
(2) 委外廠商的支援能力及服務品質
(3) 委外廠商的配合度
(4) 合約的合理規範與執行
(5) 委外廠商的知名度
(6) 要委外系統的特性
(7) 雙方合作的合夥關係
(8) 專案管理的程度
(9) 專案組織的成立與功能

[補充]
服務層級協議(Service Level Agreement；SLA)：服務供給者與客戶之間，就服務品質、水準以及性能等方面達成協議或訂定契約

定義快速應用程序開發（RAD）、敏捷開發(agile development)和

DevOps 並說明如何加快系統構建

Rapid Application Development (RAD)：是一種使用雛型、疊代式的開發技術，藉由第四代生產工具以及用戶和系統專家之間的緊密團隊合作，在很短的時間內開發系統的過程，通常可以從預先建立好的模組中組裝，程式碼自動產生

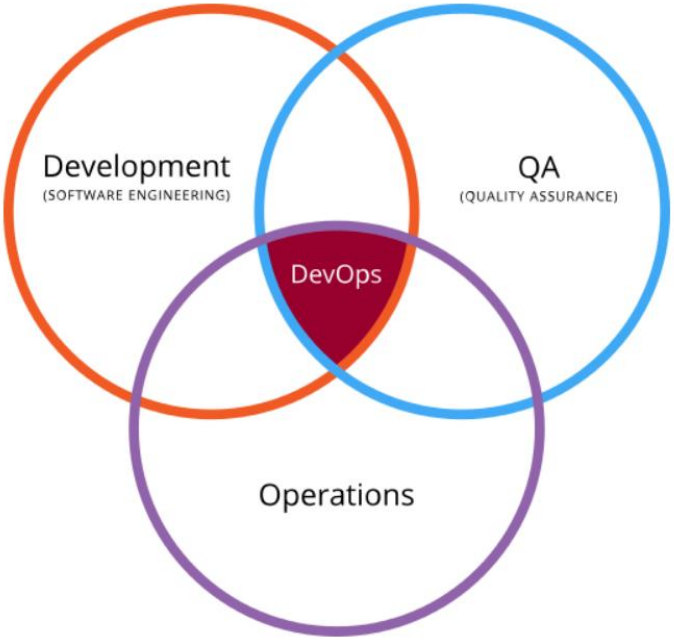


敏捷開發(agile development)：主要的精神在於較短的開發循環（建立在反覆式開發方式上）以及漸進式開發與交付，通過將大型專案分解為較小的子專案來快速建立可運行的軟體，可以更快地適應變化，強調面對面的交流

(1) 個體與互動勝於流程與工具
(2) 可運作的軟體勝於全面性的文件
(3) 與客戶的協同合作勝於契約談判
(4) 因應變化勝於遵循計畫

→ SCRUM

DevOps：建立在敏捷開發原則上，代表“開發和運營”，強調軟體開發人員和 IT 運營人員之間的緊密合作，試圖藉由促進系統開發和運營小組之間更好和更頻繁的溝通與協作來改變這種關係，透過自動化「軟體交付」和「架構變更」的流程，來使得構建、測試、發布軟體能夠更加地快捷、頻繁和可靠。



漸增模式

把「需求」分成幾個部分，然後將「部分需求」之定為一個開發週期，每個週期可依序或平行開發

適用 ：

(1) 組織的目標與需求可完全且清楚地描述

(2) 預算須分期編列，將系統做整體規劃，往後再分期執行

(3) 當組織需要時間來熟悉與接受新科技時，應用漸增模式有充裕的時間來學習與轉移技術

螺旋模型

(1) 定義系統目標與限制，以發展可行方案

(2) 依目標與限制評估相關方案

(3) 由相關的風險決定下一階段進行的方式

(4) 此週期反覆進行，直到系統開發完成為止

解釋基於元件的開發和 Web 服務如何幫助公司構建和加強資訊系統

Component-based development： 藉由一組有提供通用功能的物件組合，並將其整合以建立更大型的企業系統來加快應用程式開發，來自雲端的軟體元件逐漸增加

Web 服務 ： 使公司能夠獲取通過 Internet 交付的軟體應用程序元件，以用於建構新系統或整合現有系統；供了一套通用的標準，使組織可以通過標準的即插即用體系結構連接其他系統，無需考慮其技術平台為何；節省大量成本，並為與其他公司的合作開闢了新的機會。

解釋行動應用程序開發和響應式 Web 設計的功能

mobile website： 常規網站的版本，其內容和導航按比例縮小，以便在較小的行動螢幕上輕鬆訪問和搜尋

mobile web app： 具有行動設備特定功能的 Internet 應用程序

native app： 應用程式在「本地」(locally) 執行，也就是說它們會安裝到行動設備上，獨立的應用程式，在連接網路時可以上傳下載資料，即使未連接網路也可以對這些資料進行操作

Responsive Web design： 使網站可以根據訪問者的螢幕像素自動更改設計，無論是在桌上型電腦、平板電腦還是智慧手機上，跨各種設備和瀏覽器的用戶將可以訪問單一內容來源，其設計易於閱讀和導航，使調整大小、平移和滾動操作最少。