第三章
軟體開發模式與程序

執行單位：國立臺灣科技大學
軟體工程學程中心

大綱

- 導論
- 瀑布模式
- 演化模式
- 轉換模式
- 重複使用導向模式
- 程序重複模型
- 終極制程(eXtreme Programming,XP)
- 總結

學習目標

本章提供您了解以下項目的知識

- 軟體開發程序模式與軟體專案管理的關係
- 軟體開發程序模式的價值所在
- 各種主要的軟體開發程序模式之優缺點
- 採用何種軟體開發程序模式的選擇策略

導論

- 軟體開發程序模式與軟體專案管理的關聯性
- 為何需要軟體開發程序模式？
- 軟體開發程序模式的定義
- 使用軟體開發程序模式的優點

軟體開發程序模式與軟體專案管理的關聯性

- 軟體開發程序模式定義一系列的活動以達成產出軟體成品的目的，這些程序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爭議並提供解決的流程及方向。

軟體開發程序模式的影響

- 提供軟體工程師對於專案中有關技術與實作部份的一系列活動的流程指導。
- 提供一個管理軟體開發及維護的架構；使專案管理者能據此估計所需資源、訂定里程碑及監控專案進度。

為何需要軟體開發程序模式？(1/2)

- 降低軟體生產成本
- 縮短軟體產品「或產出」上市所需的開發時間
- 增進軟體產品「或產出」的品質
- 了解軟體產品「或產出」的整個製造流程及順序
- 可估計出完成一個軟體產品「或產出」所需的時間

為何需要軟體開發程序模式？(2/2)

- 軟體開發程序所面臨的特殊難題
  - 產品需求是否正確無誤？
  - 軟體工程師可能缺乏相關的專業知識 (Domain Knowledge) 而無法釐清使用者的確實需求。
  - 在軟體開發過程中，需求很可能會隨時間而改變。

軟體開發程序模式的定義(1/2)

- 軟體開發程序模式 (Software Process Models)
  - 又稱為生命週期模式 (Life-Cycle Models)
  - 軟體開發程序模式描述一連串的軟體開發活動或階段。
  - 程序模式使用概念性的表示法描述各開發階段之間的關係。
  - 各開發階段的時間先後順序亦表示於程序模式中。
  - 不同的程序模式可用於開發不同特性的系統。

軟體開發程序模式的定義(2/2)

- 學者們對軟體開發程序模式所下的定義
  - 程序是一個系統，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)
  - 程序是一系列經過特殊安排的步驟以達到特定的目標，軟體開發程序的目標是完成或改進符合品質要求的軟體產品。(Heineman，1994)

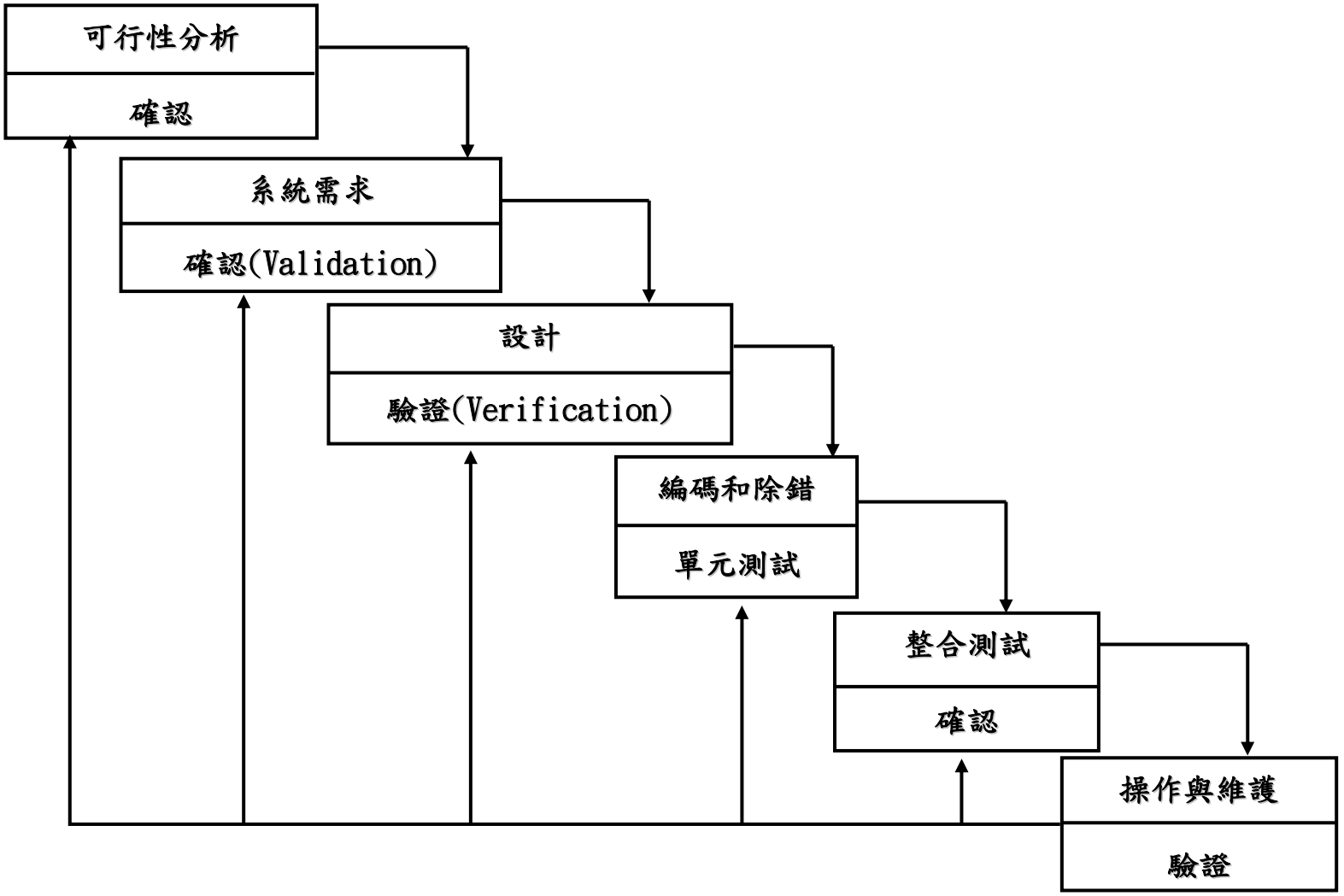
使用軟體開發程序模式的優點

- 提供標準化的軟體開發程序
  - 描繪系統主要的功能及特性
  - 統一的概念有助於溝通、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
  - 經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。

瀑布模式

- Royce於西元1970年提出瀑布模式
- 瀑布模式的設計構想
  - 軟體開發程序是依照各個階段，從分析至實作測試，直線性地進行。
  - 一個階段的產出必須經過驗證或確認才能進入下一個階段。因此任何更改、錯誤或爭議都必須回溯到前面相關的階段加以修正。
  - 軟體開發程序是一個具有整體性的模式，目標就是要在預定日期完成整個軟體系統。
- 瀑布模式是一個理想化的模式，它最大的貢獻是提出了軟體開發程序應有組織、有計畫、需要管理且開發的目標要明確。

瀑布模式


可行性分析
確認
系統需求
確認(Validation)
設計
驗證(Verification)
編碼和除錯
單元測試
整合測試
確認
操作與維護
驗證

瀑布模式的優點

- 每一階段都要求特定的文件產出
- 每一階段的結束都可成為管理控制的一個里程碑 (Milestone)
- 每一階段的產出都必須經過確認、驗證或測試的機制
  - 確認 (Validation) 是檢驗產出是否符合顧客的需求
  - 驗證 (Verification) 是檢驗系統是否依規格正確地執行

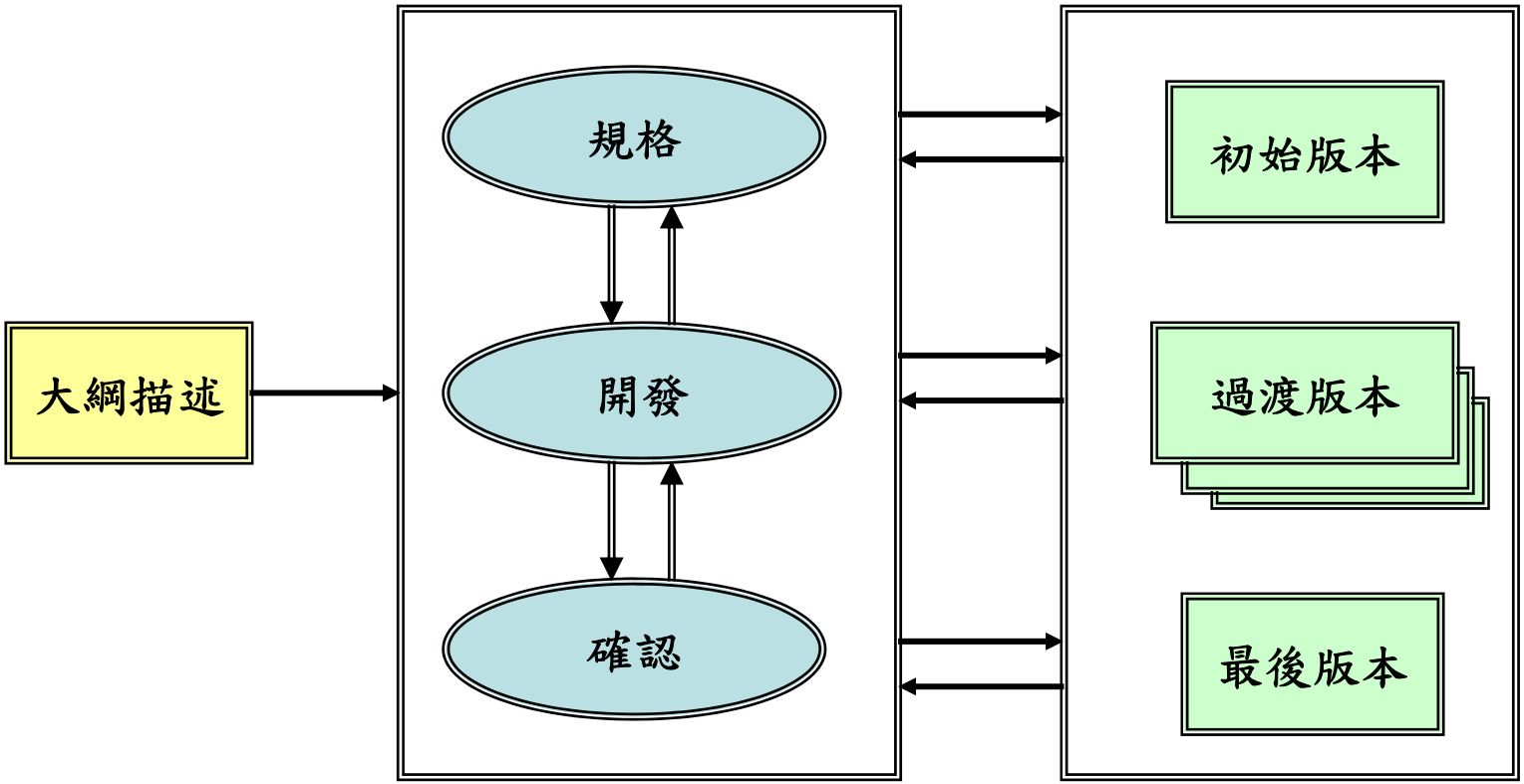
瀑布模式的缺點

- 成本及資源需求估計不準確
  - 不完整的需求分析經常造成此種結果
- 風險高
  - 必須到了最後階段才能看到可執行的軟體系統
- 不能迅速反應需求的改變
- 不能儘早得到使用者對產出系統的回應與建議
- 開發時程很長
  - 若需求不明確且經常改變，則各個階段之間必須反覆地重做以反映新的需求。

演化模式(1/6)

- 演化模式 (Evolutionary Model) 的構想是產生一個初步的系統實作給使用者試驗及操作，以蒐集使用者的意見與需求並按蒐集的意見修改系統；反覆操作此一程序直到系統被使用者接受為止。

演化模式(2/6)



【IAN01】

演化模式(3/6)

- 演化模式的基本分類
  - 演化雛型法(Evolutionary Prototyping)
    - 系統的開發是由已充分了解的需求開始，建立雛型，逐步加入新功能，直到雛型演化成為完整的系統。
  - 丟棄雛型法(Throw-away Prototyping)
    - 先對最不了解的系統需求開始，建立雛型，以增進對系統需求的掌握，確定系統需求後，則將所建立的雛型丟棄。

演化模式(4/6)

- 演化模式的優點
  - 比瀑布模式有效率
  - 對中小型的軟體系統而言，可以於很短的時間內完成開發工作。
  - 經由雛型實作，能夠儘早得到使用者對系統的意見。

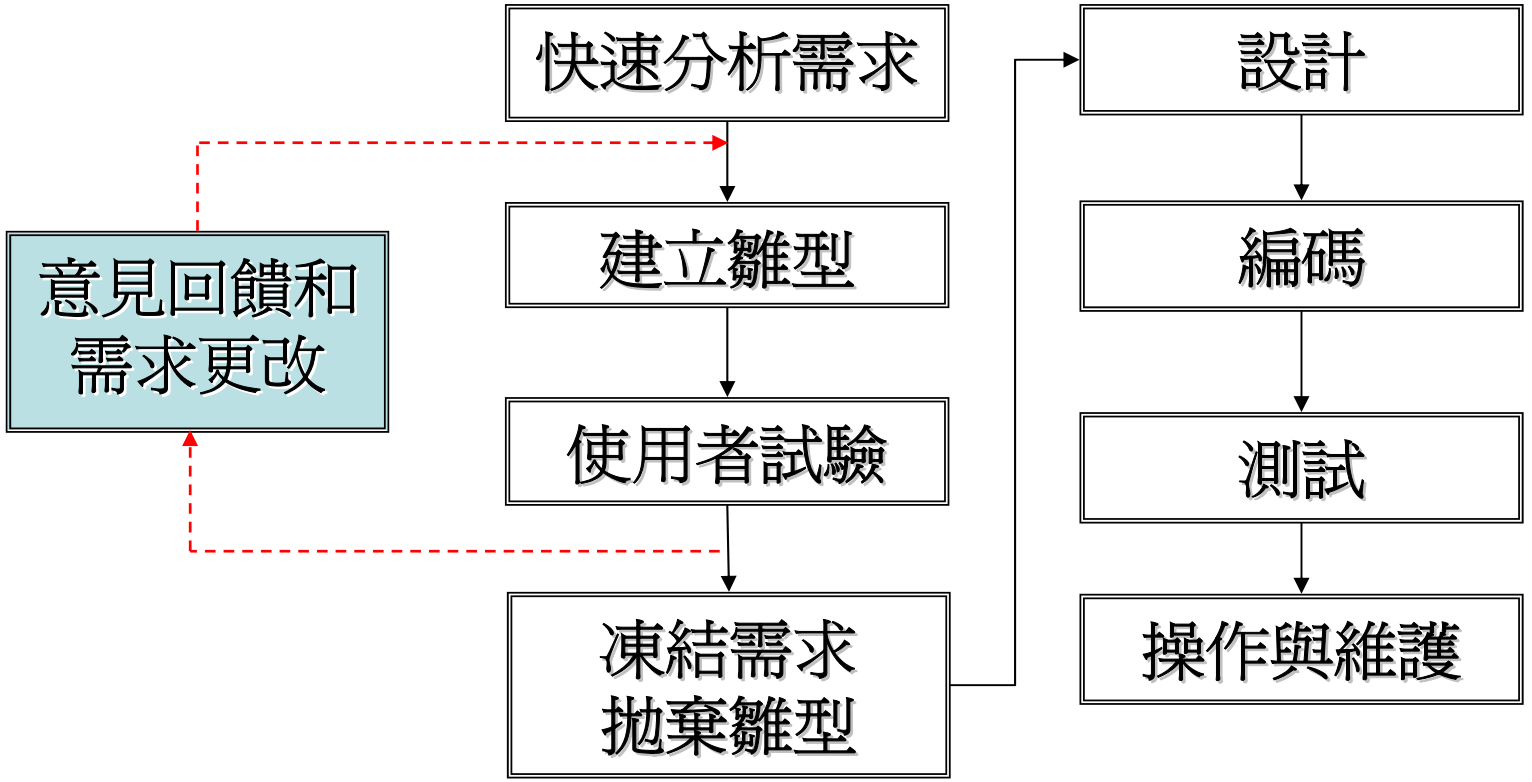
演化模式(5/6)

- 演化模式的缺點
  - 對大型的軟體系統而言，演化模式對於系統架構較不嚴謹，最好與瀑布模式混和使用，效果較佳。
  - 因為系統雛型演化速度甚快，對每一版雛型產生文件甚無經濟效益；但也因此造成專案管理者較難掌握系統開發進度。
  - 系統架構通常較為鬆散，改動困難。
  - 可能需要特殊開發工具及技術，以因應快速開發系統雛型的需求。

演化模式(6/6)

- 幾種常見的雛型法 (Prototyping)
  - 需求分析雛型
  - 分析與設計雛型
  - 子系統雛型
  - 整體系統雛型

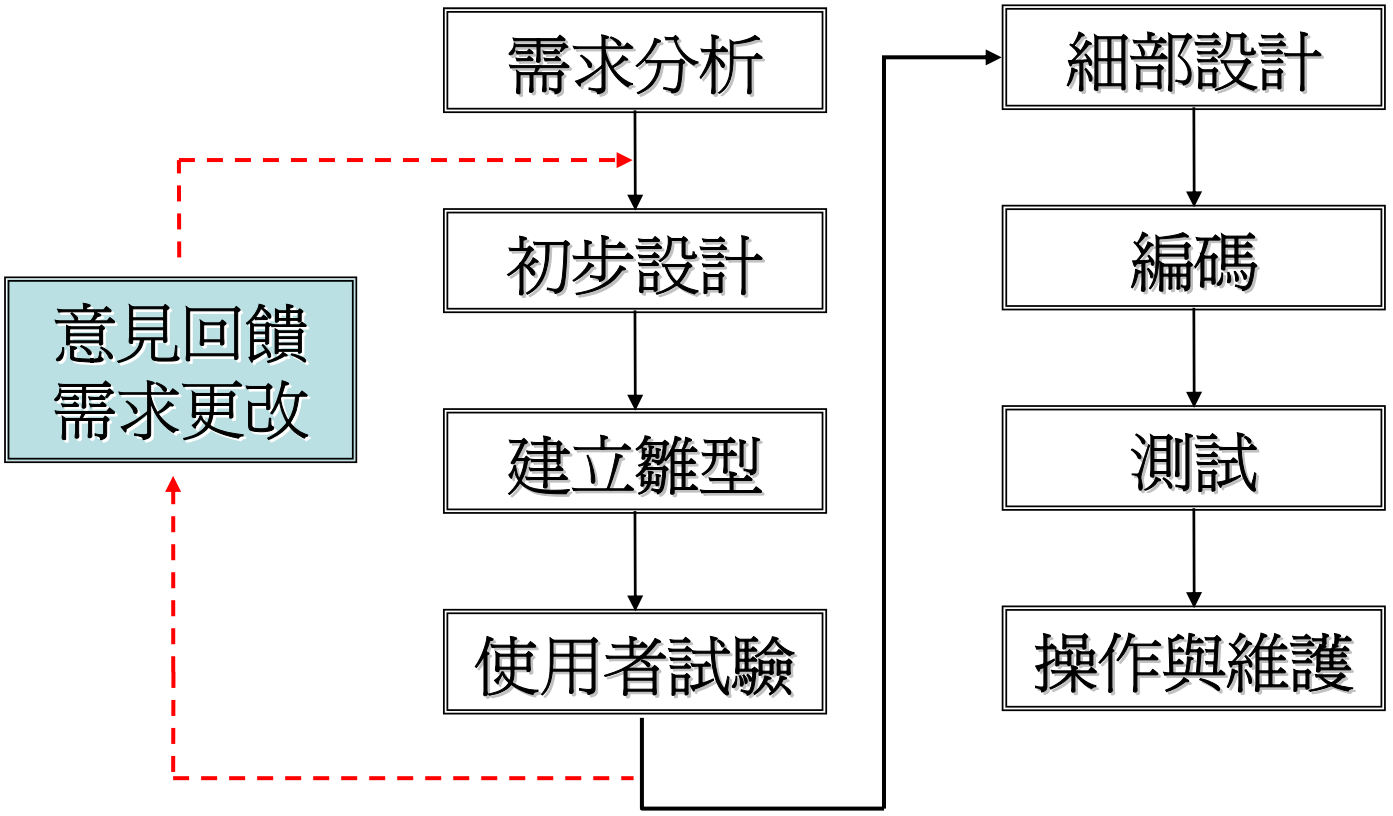
需求分析雛型(1/2)



需求分析雛型(2/2)

- 快速地分析使用需求，並建立雛型。此雛型必須讓使用者可以操作並試驗。
- 蒐集使用者的意見及更改需求，並依此修改雛型。此循環重複數次直到使用者可接受的程度為止。
- 將需求凍結並捨去雛型，接下來的各個階段與瀑布模式的作法一樣。

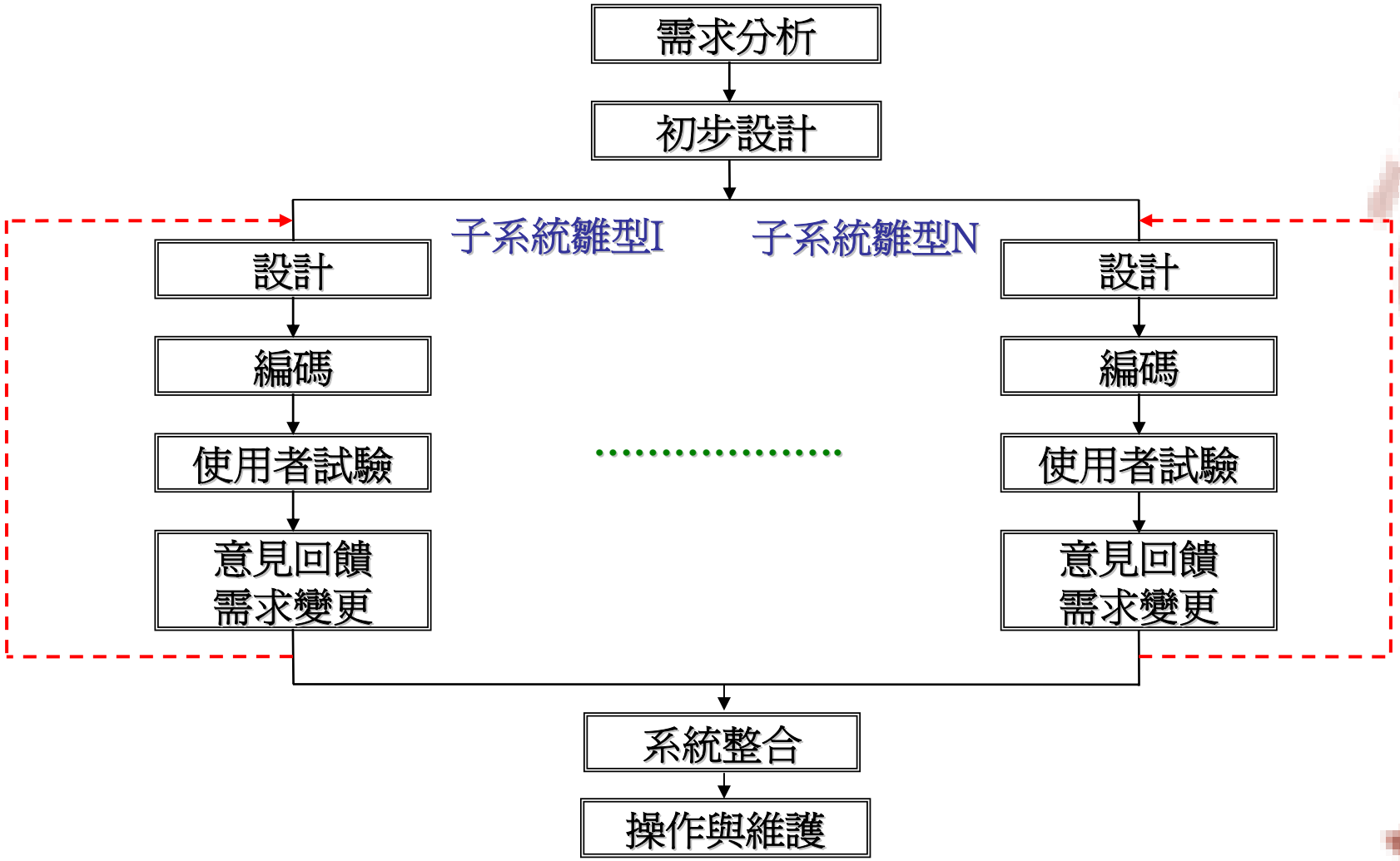
分析與設計雛型(1/2)



分析與設計雛型(2/2)

- 在建造雛型之前先做詳細的需求分析與初步設計，此法可降低重大需求變更和雛型修改的次數。
- 雛型經使用者接受後，接下來的各個階段與瀑布模式的作法一樣。
- 雛型不予拋棄，經演化後成為最終的產品。

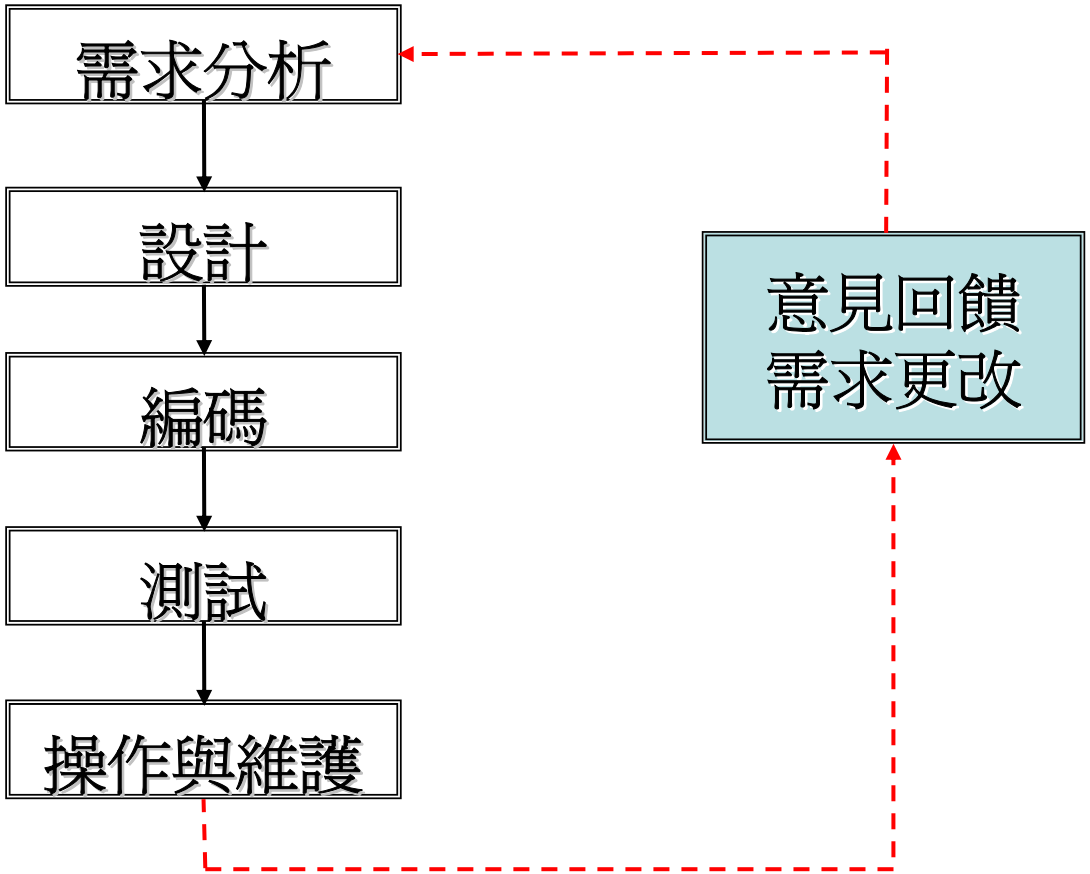
子系統雛型(1/2)


需求分析
初步設計
子系統雛型I
子系統雛型N
設計
編碼
使用者試驗
意見回饋
需求變更
設計
編碼
使用者試驗
意見回饋
需求變更
系統整合
操作與維護

子系統雛型(2/2)

- 需求分析和初步設計依照瀑布模式進行
- 建立雛型時，將系統分為數個子系統雛型。
- 將所有完成的子系統雛型進行整合
- 各子系統雛型不予拋棄，而是演化為最終的產品。

整體系統雛型(1/2)



整體系統雛型(2/2)

- 目標是以最快速度建造一個可運作的雛型系統
  - 可能犧牲或是缺乏考慮其他重要的非功能因子，如執行效率。
  - 必須經過使用者的操作與試驗，以發掘更完整的需求，使系統在演化後能漸趨穩定。

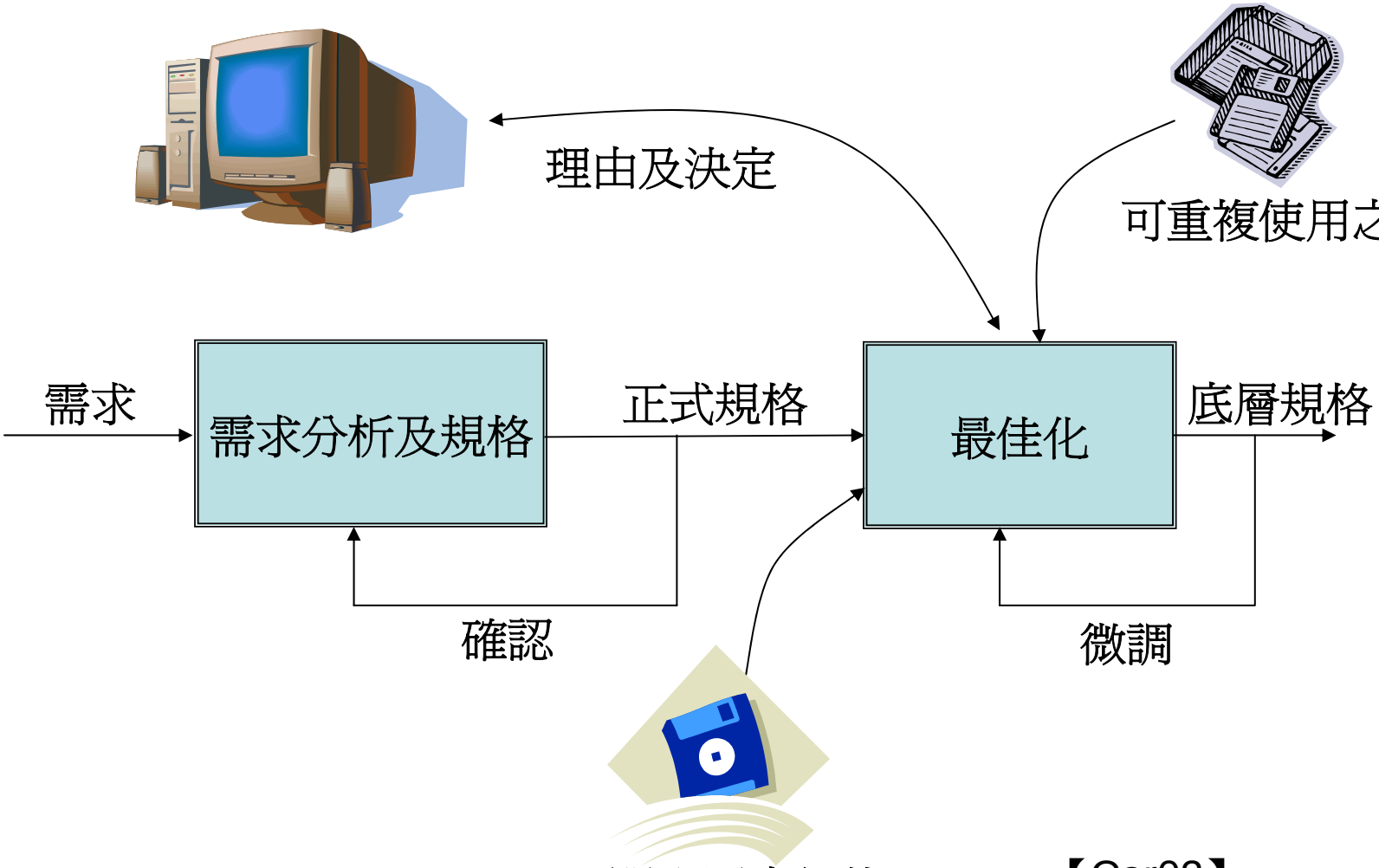
轉換模式(1/3)

- 轉換模式 (Transformation Model) 植基於正式規格說明法(Formal Specifications)，其構想是將軟體開發視為一連串的需求規格轉換成實作產品的轉換步驟。
  - 轉換模式將系統需求分解成更詳細的正式規格，通常是以數學符號來表示。
  - 然後，將這些正式規格經由轉換開發程序(Transformational Development Process)轉換成可執行並供驗證的程式

轉換模式(2/3)

- 轉換開發程序可能的實作方法
  - 軟體工程師作轉換的工作
  - 使用軟體輔助系統作轉換的工作
    - 例如使用市面上現成的可再使用的軟體元件來執行轉換的工作
- 轉換模式目前仍處於研究階段，只有少數實驗環境有使用它。

轉換模式 (3/3)

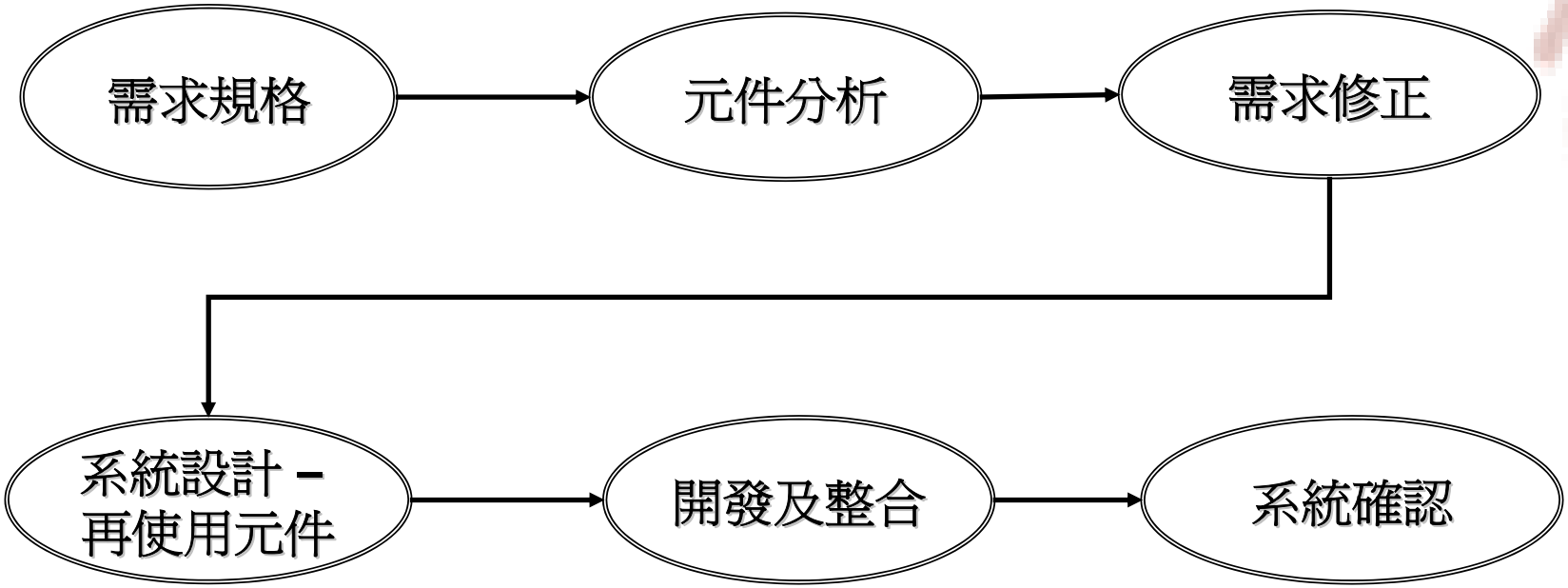


【Car02】

再使用導向模式

- 非正式的再使用模式
  - 可出現於任何軟體開發模式之下
  - 基本上，是將以往的軟體專案中，功用相似的程式碼加以改寫其中的一部分，以符合新專案的要求。
- 再使用導向模式
  - 近幾年來，所興起的新技術。
  - 倚靠一個整合性的架構來達到支援可再使用的軟體元件；此類軟體元件可能是自行開發的，或是向軟體元件廠商直接購買現成的。

再使用導向模式流程



再使用導向模式流程解說(1/2)

- 「需求規格」及「系統確認」部分與其他模式相同
- 元件分析
  - 找尋合適的軟體元件來實作需求，通常很難找到完全合適的軟體元件。
- 需求修正
  - 嘗試修正需求，以使找到的軟體元件與需求相符。若不能如此，則再回至「元件分析」步驟重新找尋合適的軟體元件。

再使用導向模式流程解說(2/2)

- 系統設計－再使用元件
  - 系統架構依據選定的可再使用元件來設計，若某些功能需求，無元件可用，則須自行設計實作。
- 開發及整合
  - 自行設計實作的部分系統必須與選定的可再使用元件進行整合工作。在此模式中，這部分的整合工作是可以與實作自行設計的部分系統之工作同時進行的。

再使用導向模式的優缺點

- 再使用導向模式的優點
  - 減少需要開發的軟體「元件」數量
    - 降低成本及開發風險
  - 能更快速地完成軟體「產品」
- 再使用導向模式的缺點
  - 若使用的軟體元件並非開發團隊所實作出來的
    - 無法確實掌握新版的軟體元件的功能，是否與舊版相容。
    - 若軟體系統需求改變，使用的軟體元件可能無法符合系統期望。

程序反覆

- 程序反覆的需求
- 混合式開發模式
  - 螺旋式開發程序
  - 漸進式開發程序

程序反覆的需求

- 系統需求的改變或演進經常使得系統的設計與實作必須重新來過，因此「程序反覆」是有其必要性的。
- 對大型的軟體系統而言，不同的子系統可能適用不同的開發程序模式；因此混合式開發模式是較適當的選擇。

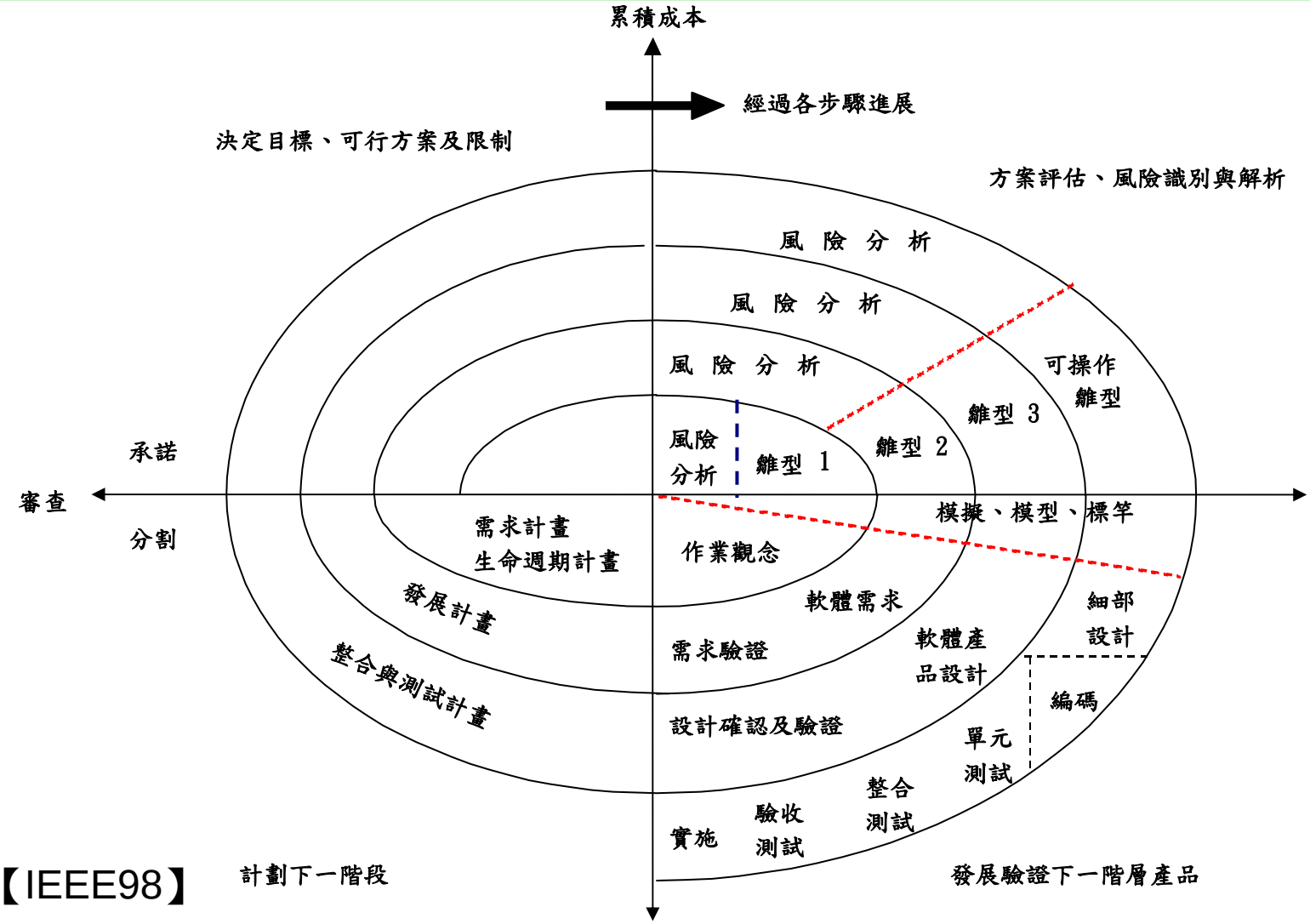
混合式開發模式

- 螺旋式開發程序
  - 將初始的系統大綱依螺旋式的開發流程，步步演進，考慮各種風險，直至系統完成。
- 漸進式開發程序
  - 將系統需求、設計及實作皆分成一系列的子系統，然後輪流完成各個子系統；後續的子系統陸續與已完成的部分整合在一起。

螺旋式開發程序(1/2)

- Boehm 於西元1988年提出螺旋式開發程序 (Spiral Development Process)
- 此開發程序中，開發流程由最內圈螺旋轉出並向外延伸，每一個迴圈代表一個開發階段；有別於一般開發模式的表示方式。
- 螺旋式開發程序對每個迴圈「開發階段」以四個不同的象限「面向」來考量此階段應完成及注意的事項
  - 目標的決定
  - 風險的預估與降低
  - 發展及驗證產品
  - 計劃下一階段

螺旋式開發程序(2/2)



【IEEE98】

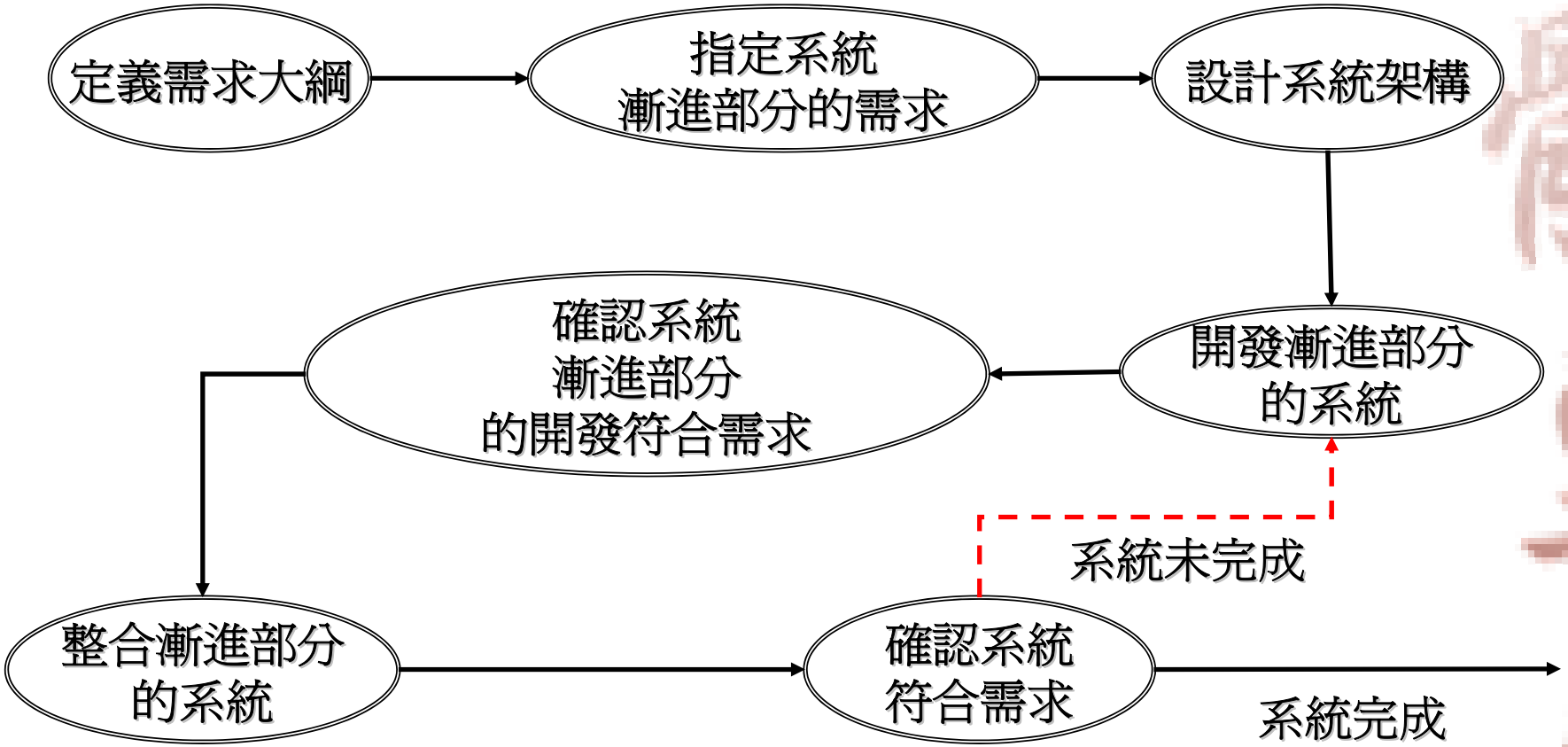
螺旋式開發程序的特性

- 將系統開發風險納入開發程序中評估
- 其他開發模式可直接套用螺旋式開發程序，以得到風險分析的結果。
- 適合大型、複雜性高、風險高的系統開發專案使用。

漸進式開發程序(1/2)

- Mills 等學者於西元1980年提出漸進式開發程序(Incremental Development Process)。
- 漸進式開發程序以結合瀑布模式與演化模式的優點為其設計的目的
  - 目標是設計一個彈性、開放的系統架構，使新功能能夠持續不斷地加入系統，而不需修改系統的架構。

漸進式開發程序(2/2)


定義需求大綱
指定系統
漸進部分的需求
設計系統架構
開發漸進部分
的系統
確認系統
漸進部分
的開發符合需求
系統未完成
整合漸進部分
的系統
確認系統
符合需求
系統完成

漸進式開發程序的優點

- 因為漸進式開發程序會先完成系統中最重要的部分，使用者可以提早使用到部分系統功能，而不需等待整個系統完成。
- 使用者對已完成的部分系統功能的使用經驗，可轉變成對後續漸進開發的系統的新需求。
- 整個系統的失敗風險會降低
- 系統中最重要的部分會在每次的新功能加入的系統整合中，得到測試的機會；因此系統最重要的功能將不易發生實作失敗。

漸進式架構適用的情況

- 預算分期編列的限制
  - 使用漸進式開發程序，系統可在一開始先做整體規劃，往後再分期執行。
- 組織需要時間來熟悉和接受新科技
  - 使用漸進式開發程序來紓解時間壓力
- 組織不願意一次大量投資
  - 使用漸進式開發程序可降低財務負擔及風險

終極制程(1/3)

- 90年代初期，由Kent Black與Ward Cunningham一起合作所提出，其成功自於強調客戶滿意及促進團隊合作。
- 主要以下列四個方向來促進軟體開發：
  - 溝通(communication)
  - 簡化(simplicity)
  - 回饋(feedback)
  - 勇氣(courage)

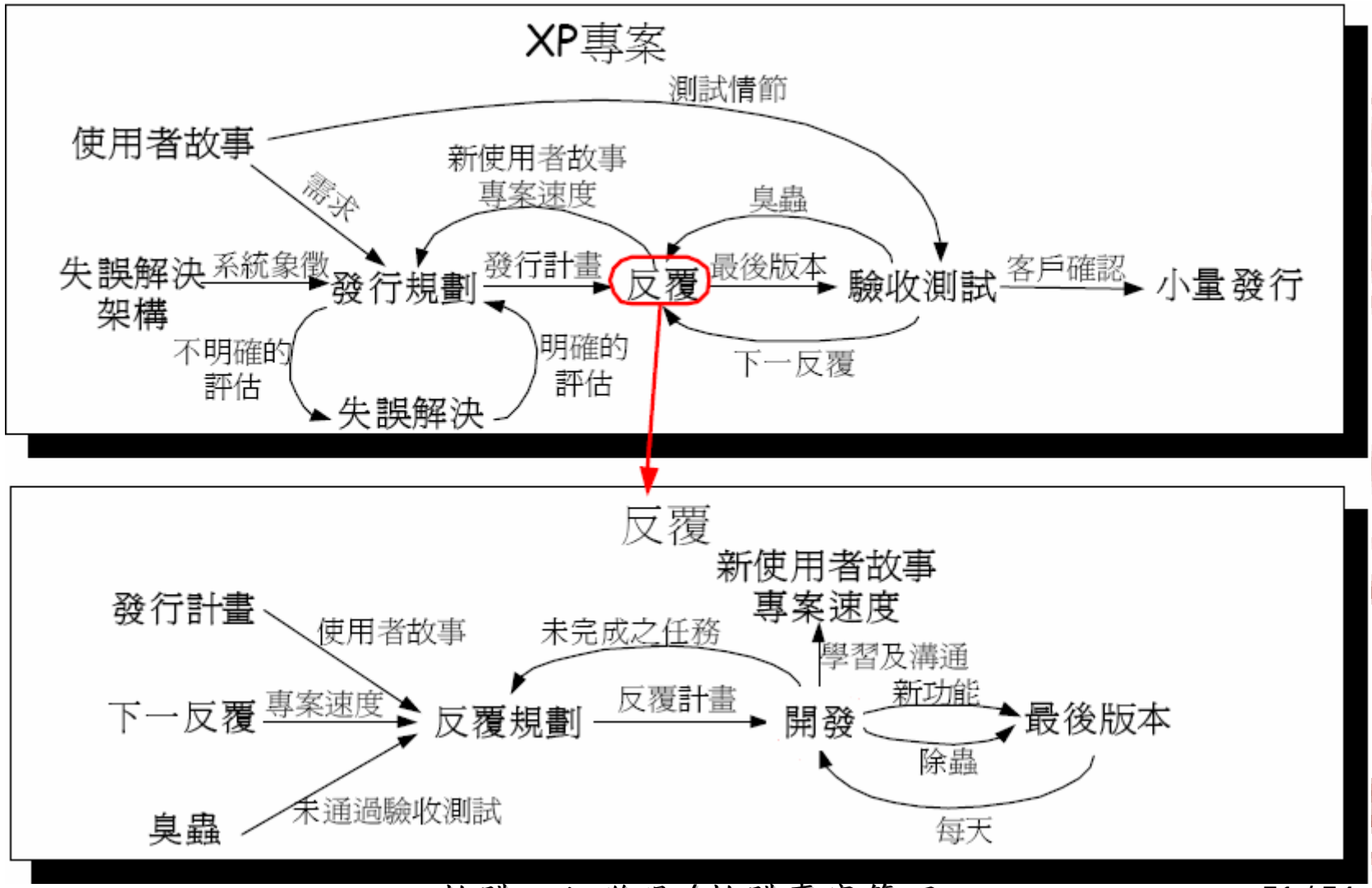
終極制程(2/3)

- XP事實上是一種嚴謹且有規律的軟體開發方式，其發展已有5 年的歷史，它已在一些對成本重視的公司中被實證。
- XP授權開發人員自信的回應客戶需求的改變，既使是在軟體生命週期的後期。
- XP同時強調團隊合作，管理人員、客戶及開發人員都是貢獻於開發高品質軟體團隊中的成員。

終極制程(3/3)

- 溝通
  - XP 的程式設計師與其客戶及程式設計師同僚溝通
- 簡化
  - 保持其設計簡單且清晰
- 回饋
  - 一開始便不斷從測試軟體獲的回饋，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
  - 基於這個基礎XP 的程式設計師能夠勇於回應持續改變中的需求及技術。

XP專案流程圖


XP專案
測試情節
使用者故事
需求
新使用者故事
專案速度
臭蟲
失誤解決
架構
系統象徵
發行規劃
發行計畫
反覆
最後版本
驗收測試
客戶確認
小量發行
不明確的
評估
明確的
評估
失誤解決
下一反覆
反覆
新使用者故事
專案速度
發行計畫
使用者故事
未完成之任務
學習及溝通
下一反覆
專案速度
反覆規劃
反覆計畫
開發
新功能
最後版本
除蟲
臭蟲
未通過驗收測試
每天

結論

- 軟體開發程序模式能幫助軟體專案控制資源及進度，並以完成軟體產出 (產品) 為最終目標。
- 已知的程序模式中，沒有一個能完全適用於任意特性或大小的軟體系統開發。
- 針對待開發的軟體系統之特性，選擇適合的軟體開發程序模式，是軟體專案管理者必須具備的能力。

自我評量

- 軟體開發程序所面臨的特殊難題為何?
- 軟體開發程序模式共有幾種?各為何?
- 試述瀑布模式的優缺點。
- 常見的雛型法有哪些?
- 試描繪再使用導向模式的流程。

附錄

- 【林信惠02】林信惠, 黃明祥, 王文良, “軟體專案管理”, 初版, 第三章, 頁61-104, 智勝文化, 2002
- Walker Royce, “Software Project Management - A Unified Framework,” Addison Wesley, Chapter 1, pp.5-20, 1998
- Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli, “Fundamentals of Software Engineering 2th Edition,” Prentice Hall, Chapter 7, pp.385-456 2002
- Ian Sommerviile, “Software Engineering 6th Edition,” Addison Wesley, 2001
- “PMBOK”, 2000
- “PMBOK中文版”, 2001