軟體專案開發經驗談

專案管理與軟體開發實務

我的簡歷

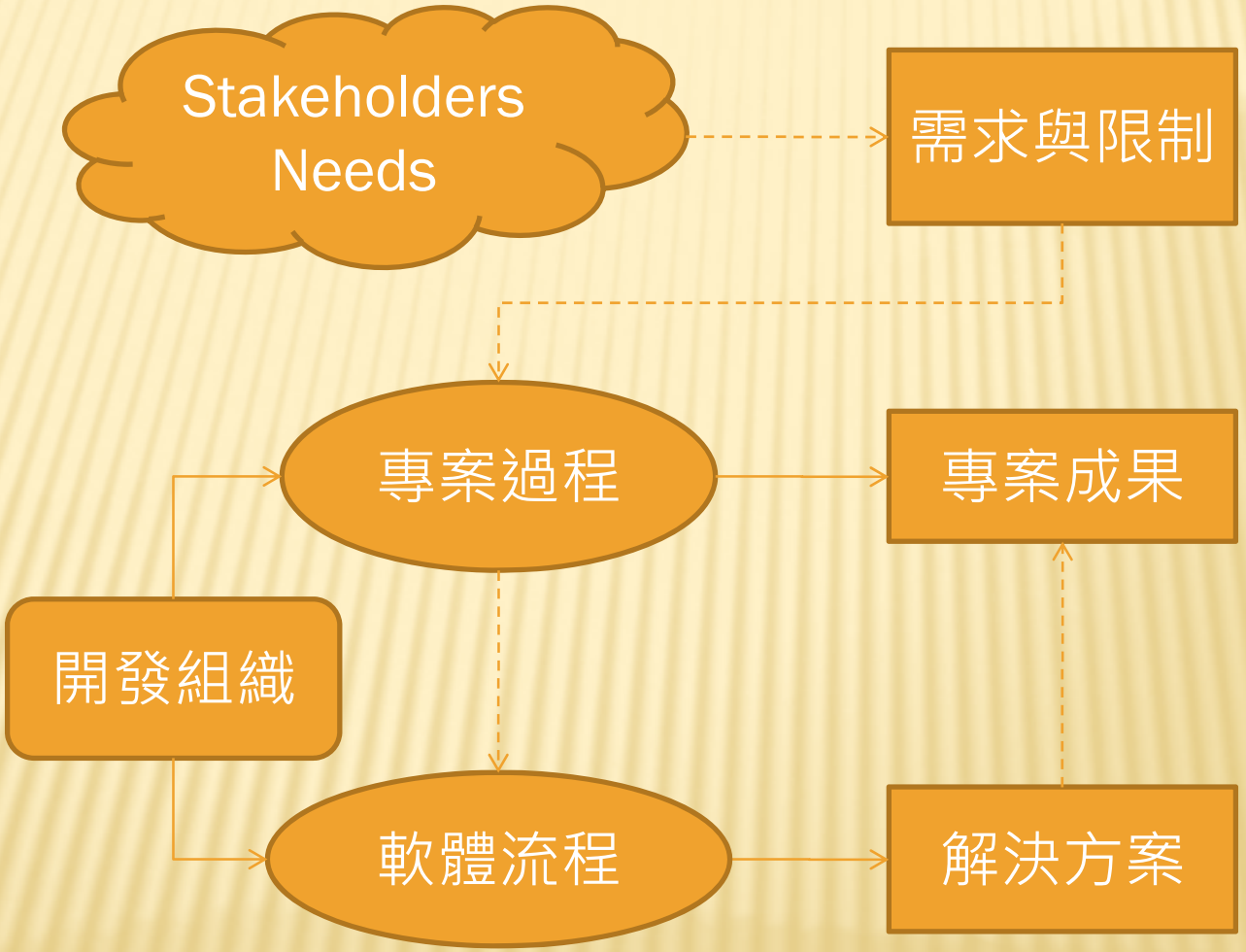
- 從事軟體專案與系統設計開發 20 餘年，歷任神通、全聯社、汎宇、台網、榮電、高格等公司專案經理及系統分析師
- ZDNet Taiwan 名家專欄《軟體開發見聞錄》作者
- 學歷
  - 台灣科技大學資管系EMBA
  - 明新工專機械動力組畢業
- 專案管理專業PMP
- 產業及知識領域：會計、進銷存、POS系統、金融電子商務
- 專長：軟體工程、軟體開發與設計
- 生涯願景：專業顧問

同人的生活派對（http://www.lifeparty.idv.tw/blog）
mailto: inpines@gamil.com

軟體專案開發的困難

- 軟體開發的複雜本質
  - 軟體的抽象與多變
  - Brooks：軟體的複雜是天生的，而不是意外。
  - 軟體開發是工程還是工藝？
- 專案的複雜性
  - 專案具獨特性的特質，牽涉某種程度的不確定性
  - 軟體專案多半需要跨領域的整合
  - 環境變化的衝擊

軟體專案開發相關的過程



相關知識領域

客戶觀點

專案管理

專案管理流程領域

軟體開發流程

品質系統

軟體工程

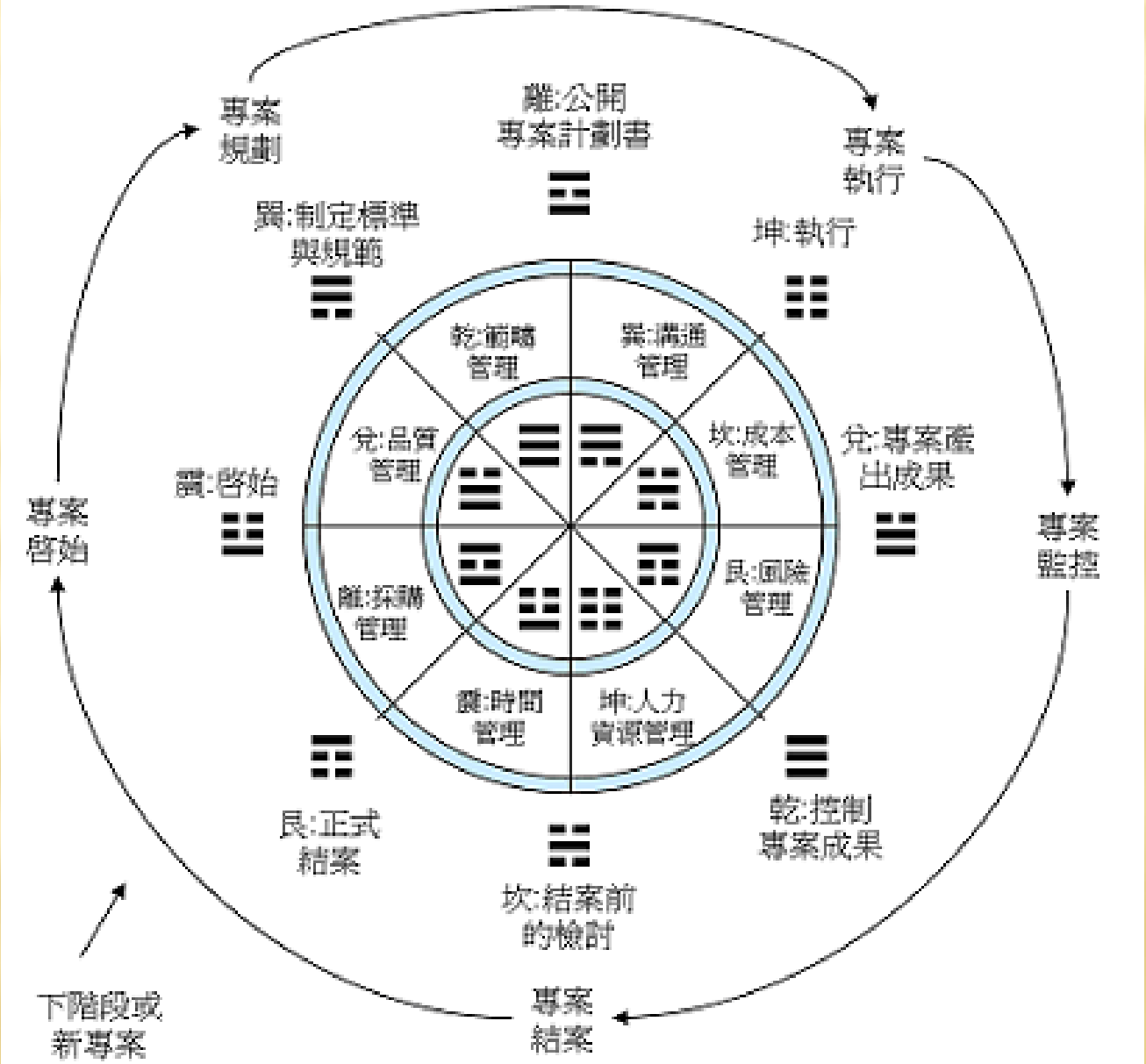
支援流程領域

工程流程領域

組織觀點

技術觀點

從易經看專案管理


專案規劃
離:公開專案計劃書
專案執行
巽:制定標準與規範
坤:執行
乾:範疇管理
巽:溝通管理
兌:品質管理
坎:成本管理
兌:專案產出成果
震:啓始
艮:風險管理
離:採購管理
震:時間管理
坤:人力資源管理
艮:正式結案
乾:控制專案成果
坎:結案前的檢討
專案啓始
專案監控
下階段或新專案
專案結案

專案管理的知識領域

- 事的問題（hard skill）
  - 該做什麼
  - 何時完成
  - 花費預算
  - 其它如品質、風險、採購
- 人的問題（soft skill）
  - 溝通
  - 人力資源（團隊、領導與衝突管理）

專案範疇與產品範疇的不同

- 專案範疇依據專案計劃書為基準，軟體專案規模內容則包含軟體的開發及建置，這些活動是產品範疇的範圍。
- 產品範疇依據產品需求規格為基準，軟體開發團隊依據軟體需求規格書從事架構設計、系統設計、系統實作、系統測試及系統建置等軟體開發流程。
- 軟體開發流程須運用到軟體工程實務。

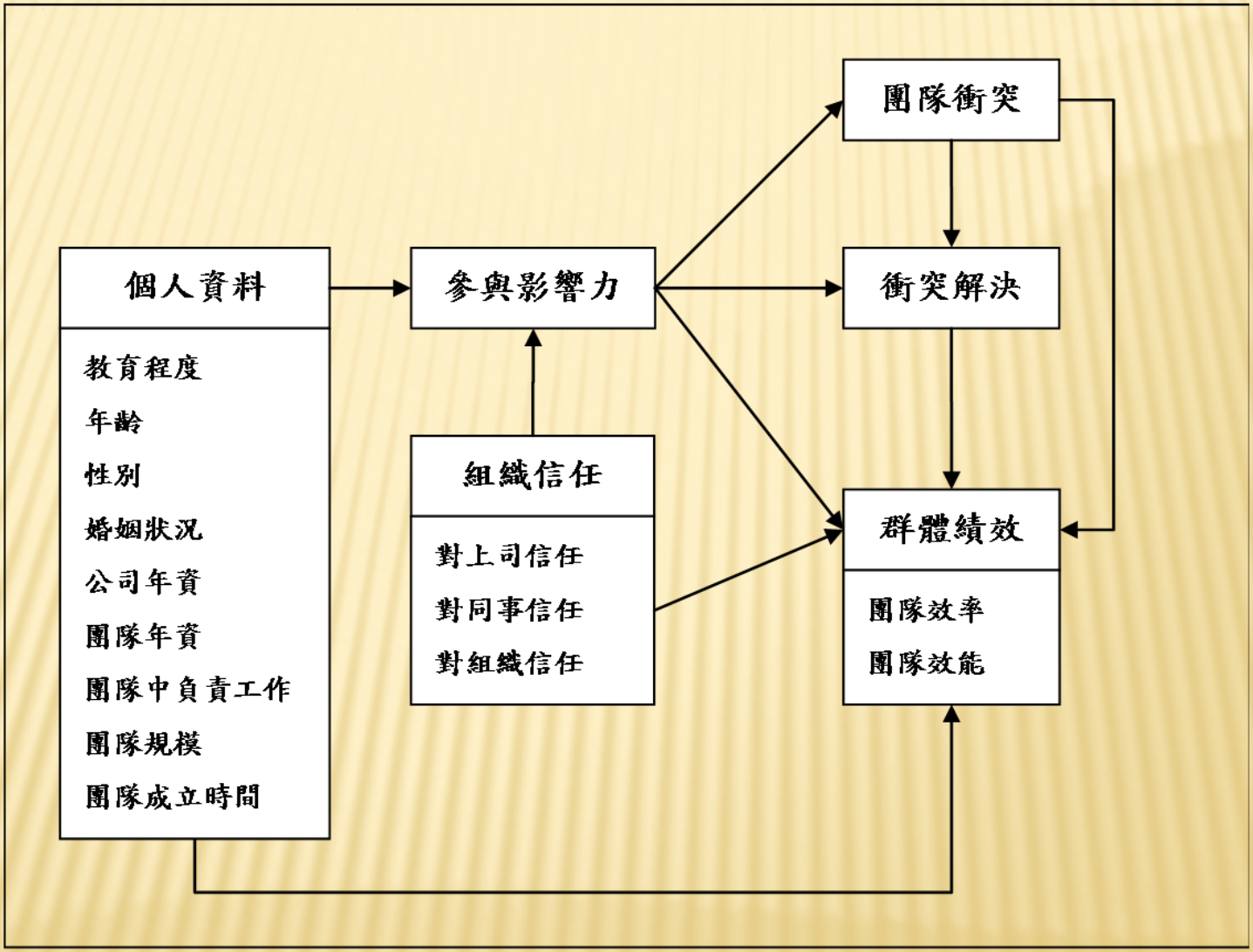
軟體專案的四變數

- 相互影響的四種變數
  - 範疇
  - 時間
  - 成本
  - 品質
- Q = f(S, T, C)
- 任何一項變數改變將會衝擊其它變數，產生專案風險；增加專案不確定性與衝擊專案成果。

需求與風險

- 在時間與成本無法變動的情況下，開發者只能犧牲品質以完成範疇
- 誰來決定開發的優先順序？
  - 客戶知道什麼功能最重要
  - 開發者明白開發最大的風險
- 重點在產生企業價值的承諾與控制風險
  - 制衡敏捷與紀律
  - 關鍵不在核心而是邊緣

軟體專案團隊參與、信任、衝突與績效



資料來源：葉木金（2007），《軟體專案之組織信任、參與影響力、衝突管理與群體績效之關連性》

解決軟體專案有關人的問題

- 軟體開發團隊的有效整合是軟體專案能否成功的關鍵，而團隊衝突往往是團隊整合最大的障礙。
- 鼓勵成員積極參與以增進衝突解決滿意度
  - 成員積極參與有助於凝聚團隊的共識，即使衝突無法避免，也可透過衝突解決的滿意度而增加團隊效能。
  - 良好的團隊運作制度（參與影響力與團隊成立時間正相關，與成員團隊年資負相關）
  - 此外，適度地授權，培養下屬的責任感，讓成員不斷地接受新事物，也將有助於使成員的積極參與。

解決軟體專案有關人的問題（續）

- 重視團隊中之組織信任
  - 團隊領導者應建立起團隊的組織信任，讓成員們相信領導者有能力及意願，讓團隊共同目標可以實現。
  - 並建立良好的團隊運作機制，讓成員相信在這個運作機制下，大家可以有效地溝通與協調，增進彼此熟悉程度與化解歧異，進而增加團隊成員間的相互信任。

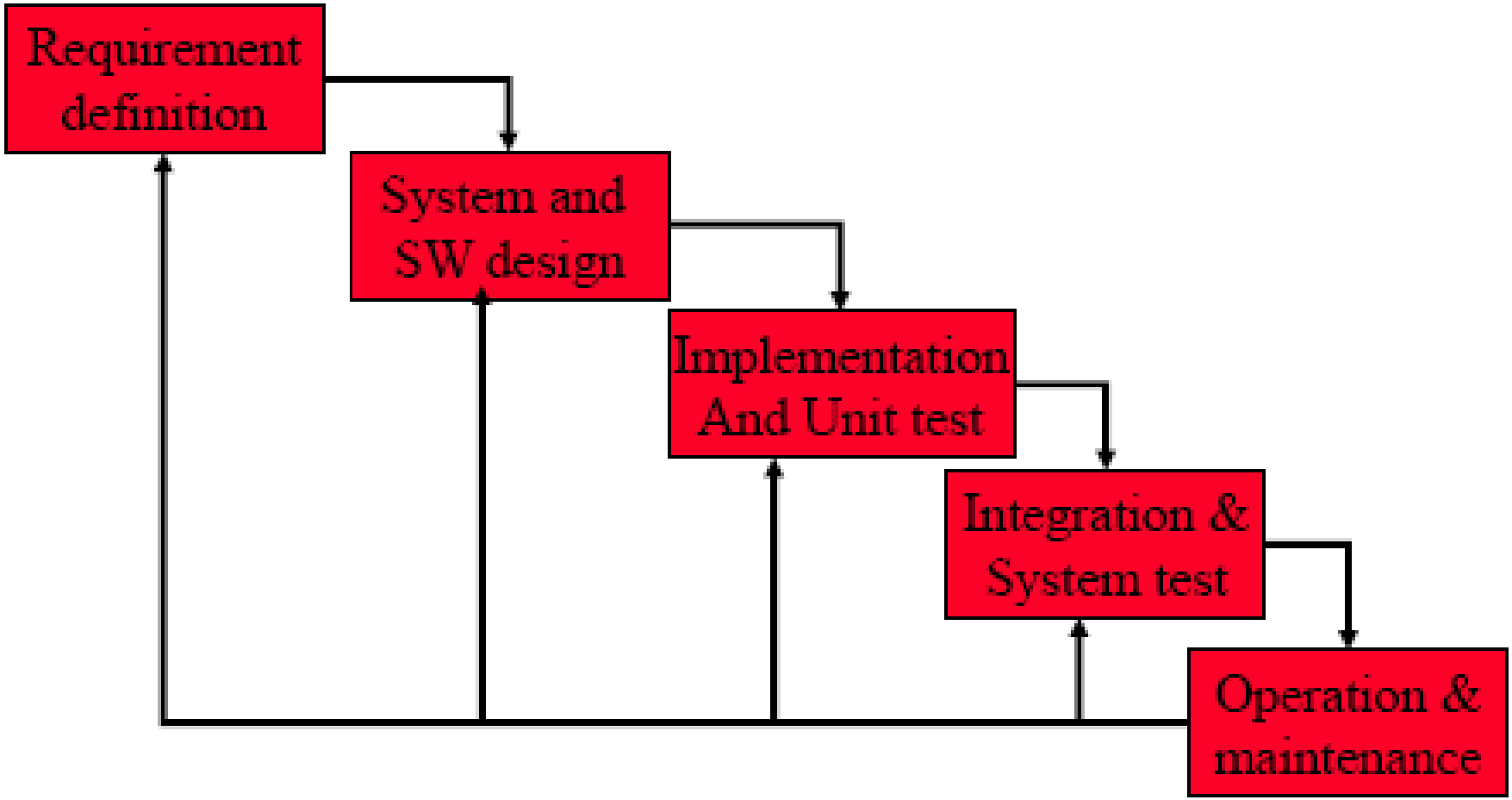
軟體工程為何重要？

- 軟體工程是實務中的理論。
- 軟體工程將工程手法應用到軟體上，使得軟體開發過程是可掌握的，增加軟體專案成功的機會。
- 使用者需求愈來愈複雜，使得軟體開發有多變的本質。
- 管理改變而非迴避改變。

軟體開發流程

- 軟體流程是因應開發軟體系統所需，有結構的作業集合～
  - 需求規格（軟體的功能與限制）
  - 設計與實作（滿足規格）
  - 驗證（確定軟體符合使用者需要）
  - 演進（滿足使用者需要的改變）
- 軟體流程模式是流程的抽象化表述。它以一些特定的觀點來描述流程。
- 在大型系統中，於早期重覆執行的流程反覆，一直都是流程的一部分。

Software Life-cycle & Waterfall model



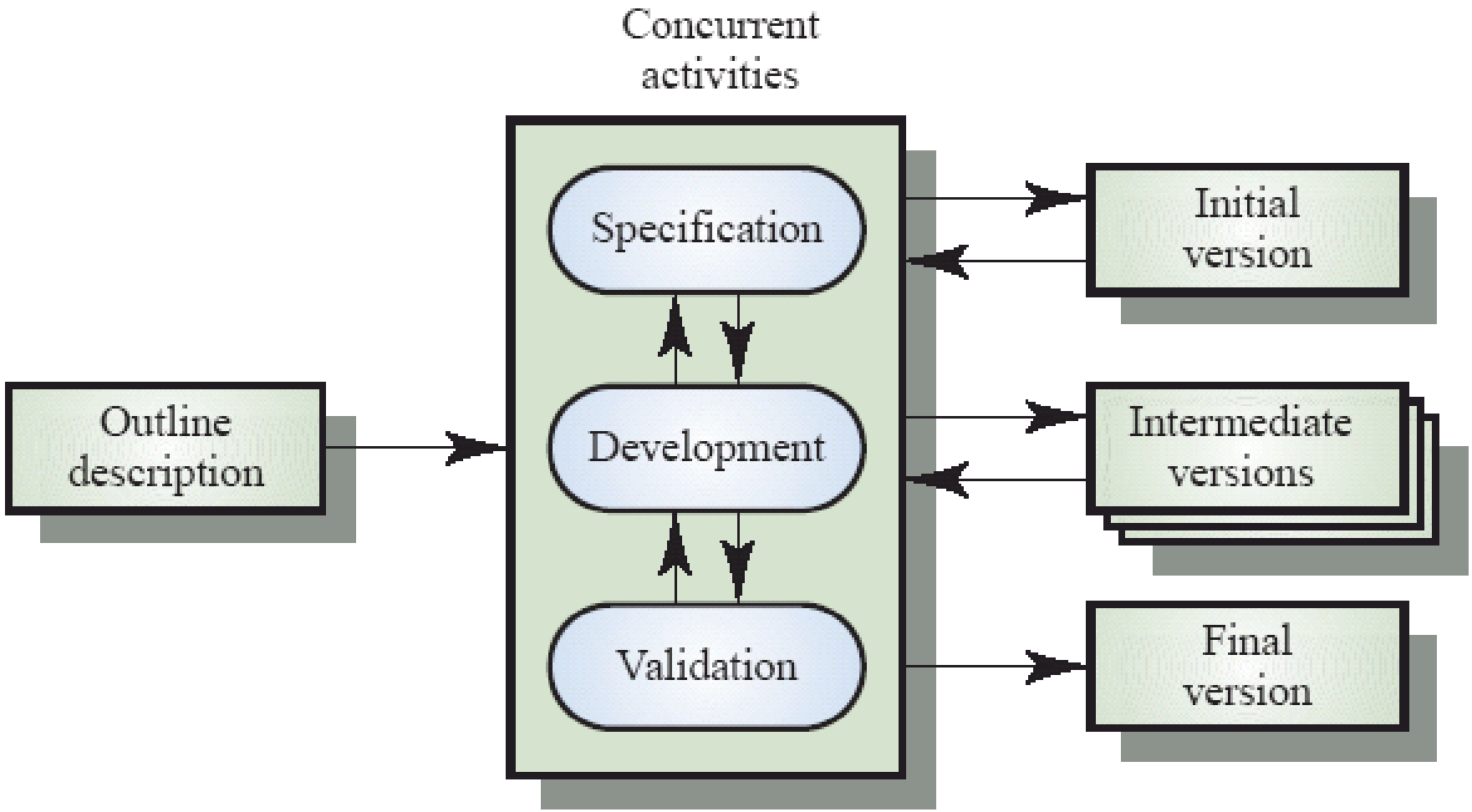
瀑布模式(WATERFALL MODEL)

- 直到前一階段結束，下一階段才能開始。
- 開發完成後，才能看到軟體。
- 問題與限制
  - 早期的錯誤與遺漏，卻到了最後才被發現。
  - 缺乏專案分割成清楚階段的彈性。
  - 難以反應客戶需求的變更。
  - 在進行中的流程很難考慮到改變。
  - 適用於明確需求的軟體開發。

如何改善瀑布模式的問題

- 錯誤與遺漏最後才發現
  - 分析設計須包括具體的概念驗證(POC, Proof of Concept)。
  - 實施風險規劃及監控流程。
- 難以反應客戶需求變更、進行中流程很難改變
  - 執行驗證與確認。
  - 實施變更管理，並加強溝通回饋。
- 缺乏專案清楚分割的階段的彈性
  - 清楚區分需求的重要性及優先順序。

Evolutionary development



演進模式(EVOLUTIONARY MODEL)

- 初步實作雛型，顯露使用者意見並逐版修改，直到適當系統被開發出來為止，有兩種開發方式：
  - 探究式發展*(Exploratory development)*，目標是與客戶共同從初步輪廓需求逐步形成最終系統，應始於需求是易於理解的。
  - 拋棄式雛型*(Throw-away prototyping)*，目標是了解系統需求，應始於對需求了解程度的缺乏。
- 規格能以漸增方式發展出來。

演進模式的問題

- 流程可見度的缺乏
  - 快速開發但缺少文件。
- 開發的系統常缺乏結構
  - 因為需求不斷改變。
- 可能需要特別技能
  - 工具或技術不相容。
  - 擁有此項技能的人很少。

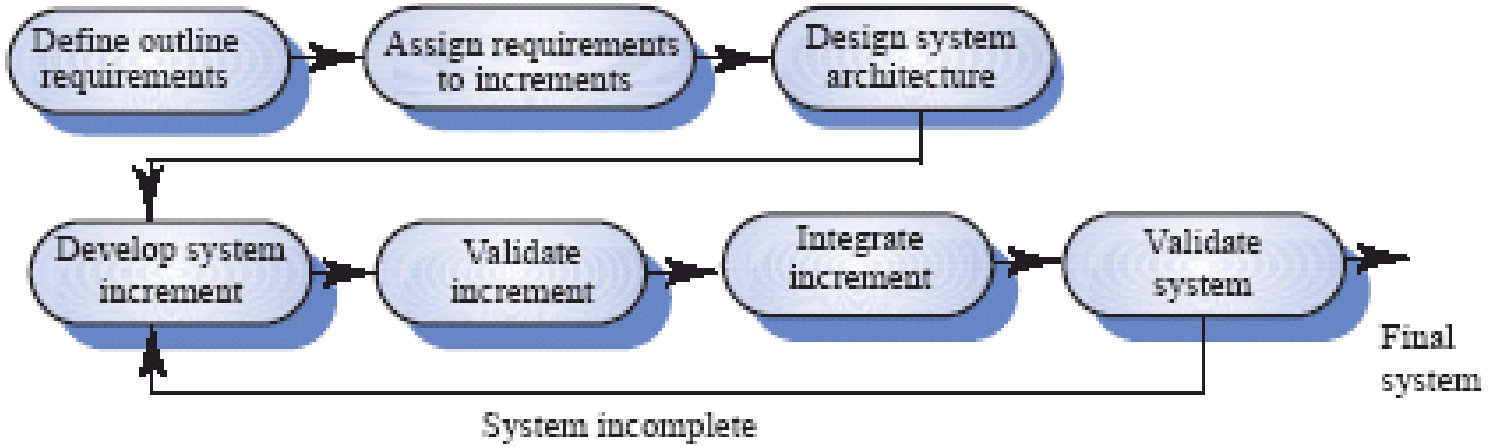
如何改善演進模式的問題

- 流程可見度的缺乏
  - 定期召開專案會議，相互討論並分享經驗及教訓 *(Lesson Learn)*。
  - 執行專案收案動作。
- 開發的系統常缺乏結構
  - 定期整理程式，必要時須重構(Refactoring)。
- 需要特殊技能
  - 鼓勵相互分享經驗，或採用搭檔合作的作業方式，例如導入 Pair Programming。

流程反覆(*PROCESS ITERATION*)

- 系統需求在專案進程上總是逐漸成形。
- 反覆能適用在任何一般流程模式，兩種相關的方法支援流程反覆~
  - 漸進式開發，每個階段分解成一系列增量並依次開發。
  - 螺旋式開發，螺旋向外系統開發從初步輪廓到最終開發系統。

Incremental development



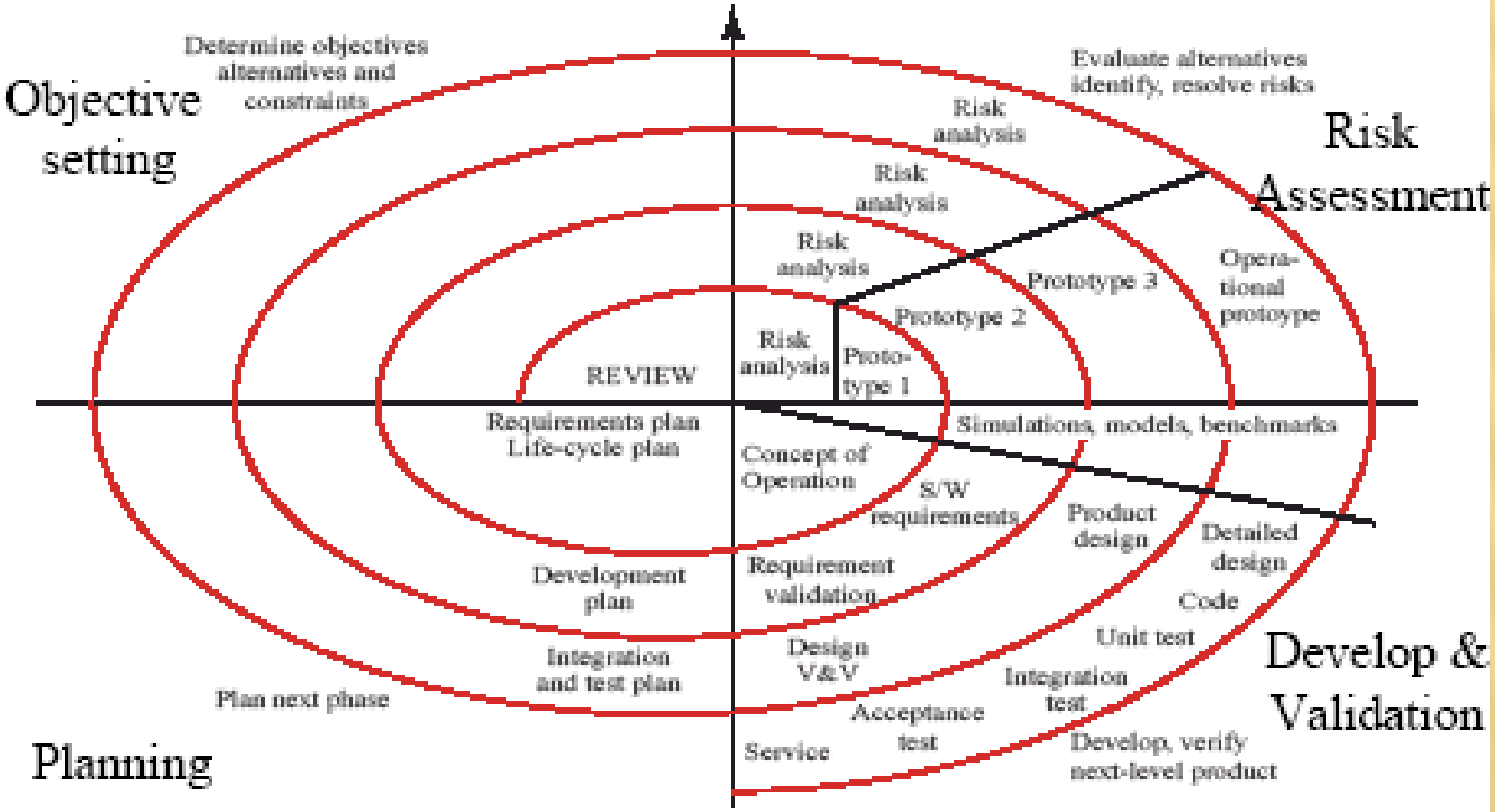
漸進式開發(*INCREMENTAL DEVELOPMENT*)

- 開發並交付必要功能所分解的各部分增量。
- 高優先權的需求包含在早期的增量。
- 優點
  - 客戶有機會延遲決定，可以等到確認需求再討論。
  - 不須等全部系統交付，早期增量滿足關鍵性需求，已交付系統可以馬上使用。
  - 減少開發的重工（不用一直修改規格）。
  - 較低專案失敗的風險。

漸近式開發 - 問題與限制

- 不能接受改變正在開發中的增量需求。
- 增量應該相對少，而且每個增量都應交付一些系統功能。
- 很難應對客戶需求到正確增量的大小。
- 需求直到某個增量出現才能定義得詳細，也就是說，很難找到所有增量都需要的共用模組。

Spiral model of the software process



螺旋式開發(*SPIRAL DEVELOPMENT*)

- 流程以螺旋狀表示，而不是表述成有退回路線的直線順序活動。
- 螺旋的每一圈代表軟體流程的一個階段。
- 沒有固定的規格或設計階段，螺旋每一圈的選擇是取決於需求是什麼。
- 整個流程中，明確地評估與解決風險。

螺旋模式四區段

- 設定目標
  - 識別特定階段目標（識別流程限制）。
- 評估與減少風險
  - 評估風險並提出以雛型減少主風險的活動。
- 開發及確認
  - 選擇軟體開發模式。
- 規劃
  - 審查專案並決定是否繼續下一階段。

反覆與演進

- 如何計劃反覆 ~
  - 風險趨動
  - 客戶趨動
- Timeboxing
  - 固定結束日期，且不允許修改。
  - 進行中的反覆，不得變更其需求。
- Adaptive Planning
  - 取代預測性計劃，短期不會建立詳細時間表。
  - 細節與承諾的水準與資訊品質等量。

品質的觀念

- 品質是
  - 品質是滿足明確或隱含需求能力的特性總和。
  - 品質是滿足客戶需求並適應他們需要
  - 品質是創造客戶的價值
- 技術水準低，代表功能少，不見得有問題；但品質水準低代表錯誤多，絕對有問題。

確認與驗證（V&V）

- 造成專案失敗的三種主要原因：
  - 用正確的答案回答錯誤的題目
  - 用錯誤的答案回答正確的題目
  - 答案與問題皆錯誤
- 避免發生上述錯誤，應進行確認與驗證過程
  - 確認是做對的事 - do right thing，強調命題；即Care what、Know what（重視概念）。
  - 驗證是把事情做對 - do the thing right，強調解決方案；即Know how、Know why（重視操作）。

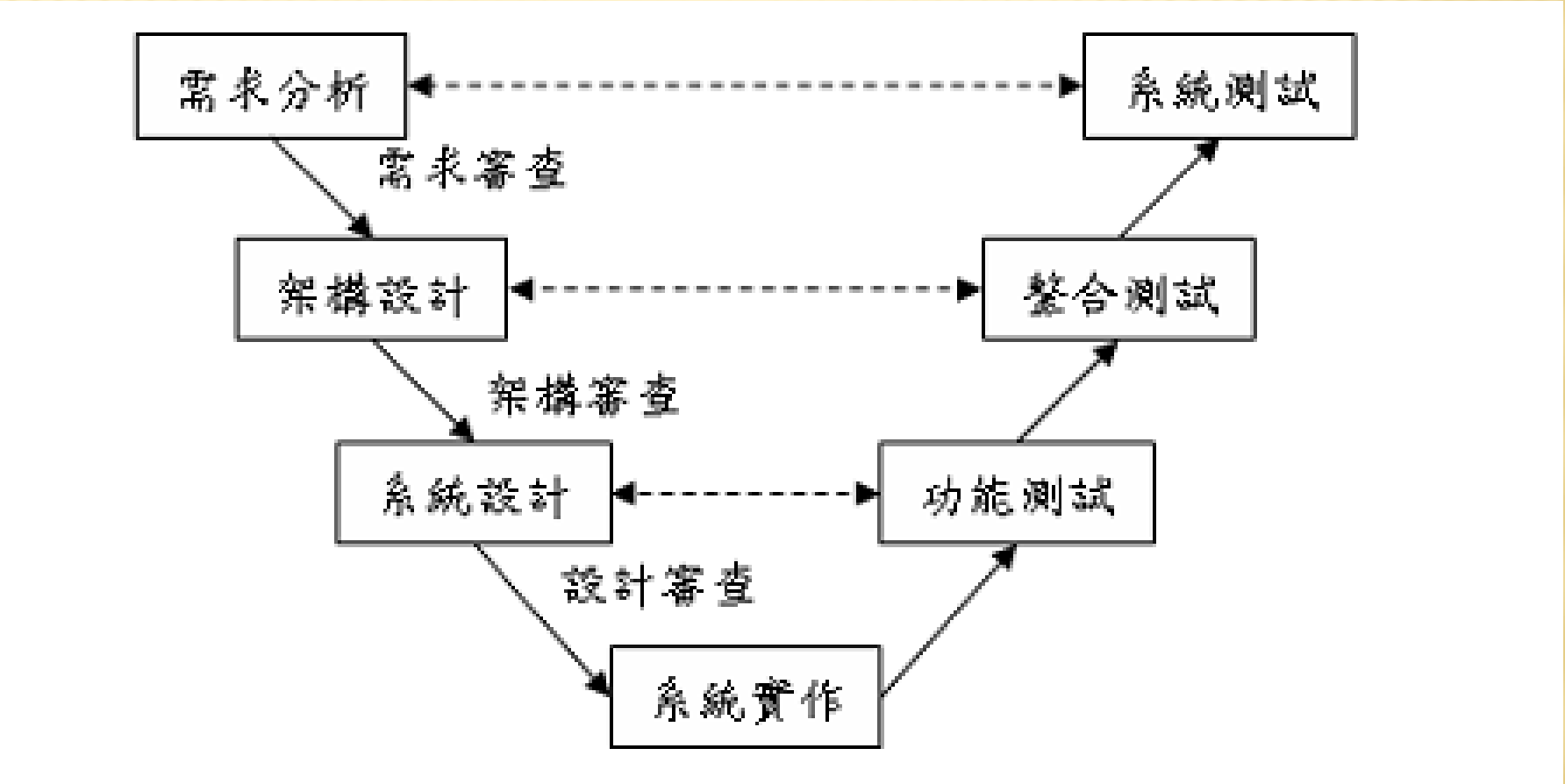
QC與QA

- V&V相當於QC，是用來控制產品品質的過程
  - 各種層面的測試
  - Technical review
  - Inspection
  - Work through
- QA是專案品質的執行過程，是確認開發流程的品質
  - 稽核
  - 內部訪談及訓練

QC與QA（續）

- 品質相關流程
  - 品質規劃：決定品質適用的標準，計劃專案如何滿足這些標準。
  - 品質保證：使專案滿足標準樹立信心所採取所有規劃與活動。
  - 品質管制：監控專案，找出消除不滿足標準的方法。
- 品管為根據需求規格驗證工作產品是否符合要求及沒有程式錯誤(Bug)產生。
- 品保則是根據品質計劃確認是否按照制定的流程進行開發。

V-MODEL



V-MODEL（續）

- 台灣一般大型軟體專案最常用的開發流程，為自瀑布模式改良，並加入 V&V 的基本觀念。
- 展現了不同抽象層次觀點的開發作業與相關的測試作業之間的關係。
- 一般將軟體專案開發分為三個開發層次
  - 高階的抽象層次為處理使用者的需求分析；
  - 中階的抽象層次則將重點置於將所了解的需求轉化為軟體架構；
  - 低階的抽象層次主要為系統功能的細部設計與系統實作。

V-MODEL的問題

- 採用 V-Model 的好處：
  - 配置了良好的結構方法，讓每個開發作業都可以根據前一個開發作業的詳盡產出來進行作業。
  - 而測試計劃可以很好地提早在編程前就開始進行規劃，如此將為專案省下大量的時間。
- 但無法避免在專案後期需求與設計的經常變動，而影響程式碼品質的問題。
  - 愈高階層次的開發產出，其瑕疵是愈晚才會被驗證出來。
  - 相較於低階層次的開發產出，其衝擊影響力又是相當嚴重的，因為這代表瑕疵將會在較低層次的開發產出中擴散與蔓延

解決 V-MODEL 的問題

- 增加測試的效率及涵蓋面
  - 受限於專案的時程與成本
- 運用整合與同步，來提昇分析與設計的品質
- 整合
  - 在分析設計時提出驗證的具體方法
  - 例如概念驗證（POC）
- 同步
  - 交付分析設計前，讓團隊進行實質的技術審查
  - 例如 software inspection