物件導向系統分析與設計

Object Oriented Analysis and Design

系統開發方法發展與沿革

劉儒斌 Paladin R. Liu
paladin@ntub.edu.tw

AGENDA

- 系統開發方法簡介
- 軟體生命週期
- 系統開發方法介紹
- UML 介紹

系統開發方法簡介

- **資訊系統開發活動的一系列步驟及執行程序**
- **以系統化、邏輯化的步驟進行，期望能達到…**
  - 標準、規範的建立
  - 易於管理，開發活動更有效率
  - 確保開發結果的品質

軟體生命週期

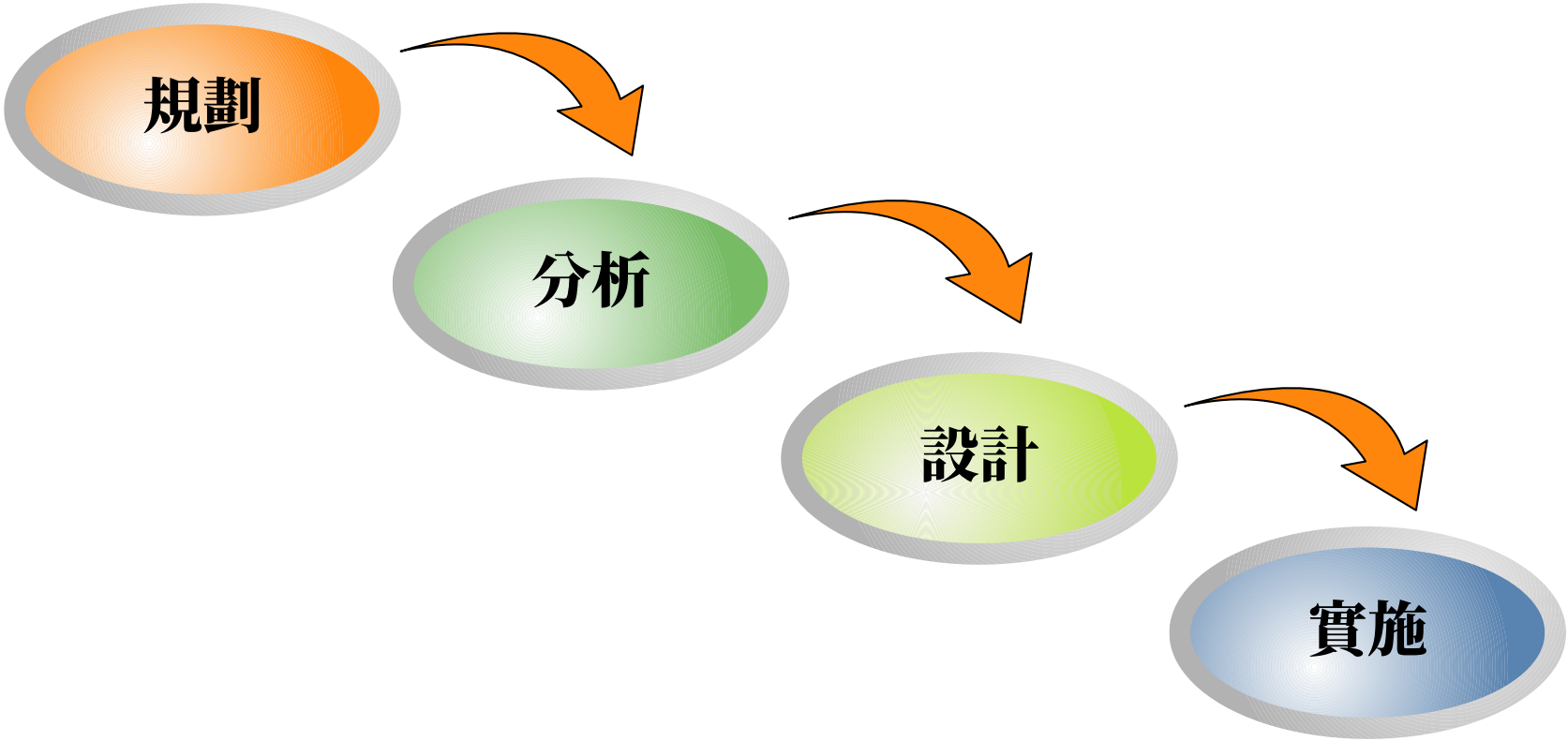
軟體發展生命週期

- **一個系統從無到有的過程**
  - 了解系統如何能支援企業 / 使用者的需求
  - 有了明確的需求後，開始進行系統的分析與設計
  - 將設計予以實作，通過測試後，產出最終的工作成果
  - 系統交付上線、運作

想像一下蓋房子…

- 一開始的構想…
- 繪製外觀、形狀…
- 繪製工程藍圖
- 討論 修改 再討論，直到滿意爲止
- 開工整地、挖地基…

軟體發展生命週期



軟體發展生命週期

- **計劃**
  - 計劃開發活動將如何進行
  - 準備開發活動所需的資源
  - 確認開發活動可能遭遇的限制
- **分析**
  - 整理需求內容
  - 了解產品想要做什麼

軟體發展生命週期

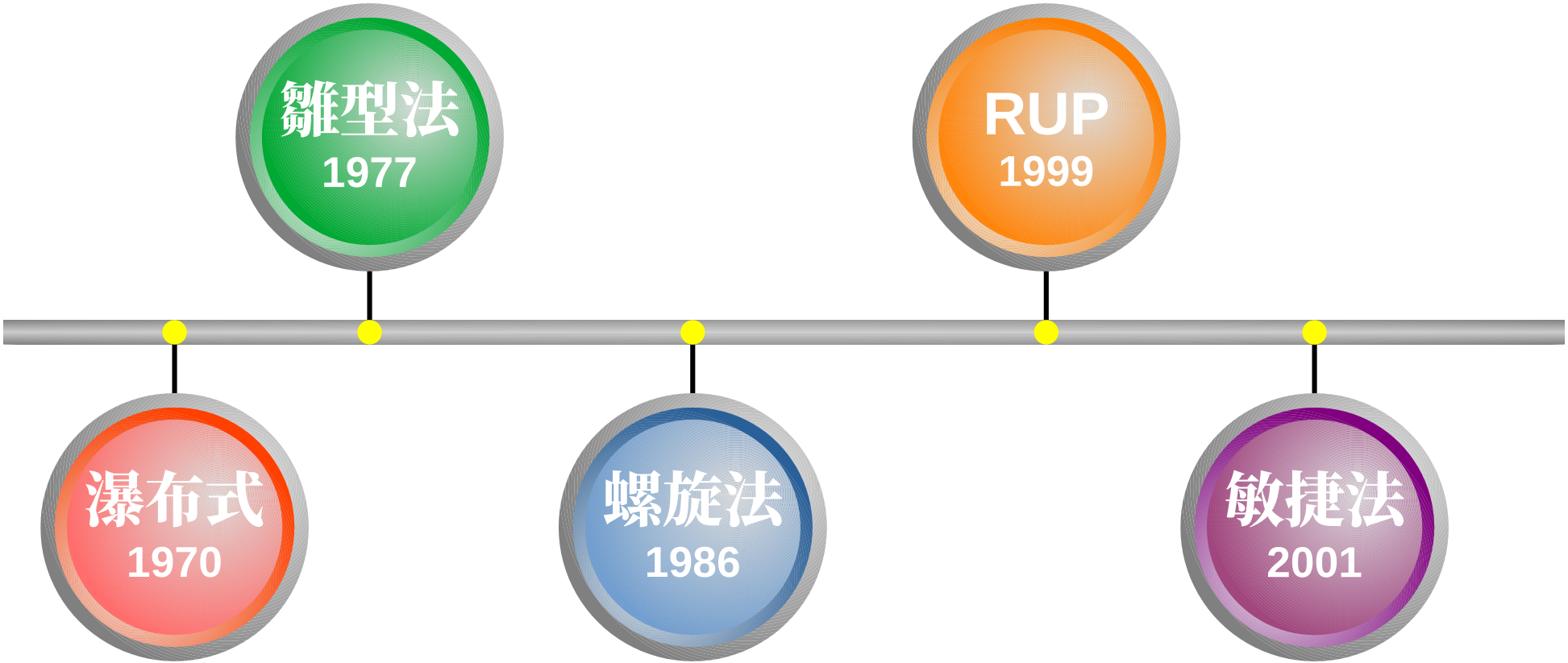
- **設計**
  - 描述產品要如何做成
  - 拆解分類、結構 / 模組設計
- **實施**
  - 實際進行開發、測試
  - 逐步完成產品

軟體發展生命週期

- **七階段法：把實作階段再細分**
  - 編碼：編寫代碼
  - 測試：單元測試、整合測試、使用者測試
  - 操作：系統移交，上線操作
  - 維護
    - 依使用者回饋評估並修改系統、修正系統錯誤，直至系統不再被使用

系統開發方法介紹

開發方法的演進



編碼與修正

- **1950 年以前**
- **早期的開發模式，沒有模式可言**
- **邊寫邊改**
- **主要問題**
  - 沒有規劃與設計，經過幾次修改後，系統變的難以理解與維護
  - 程式架構好不好，完全靠工程師個人經驗，沒有先例可依循
  - 工程導向，開發的過程中，使用者只是被動的接受開發的結果；但結果可能是不可用的

瀑布式開發

- 1970 年由 Royce 提出
- 定義了系統由開發產生，至系統汰換的一系列活動
- 經歷史的驗證，至今依然有用的方法

瀑布式開發

- **特色**
  - 整個生命週期有明確的階段，以定義每個階段
    - 應執行的工作
    - 應有的工作產出（文件）
  - 階段之間是循序性的，下一個階段必須等到上一階段完成後，才會開始進行

瀑布式開發

- **分析**
  - 可行性分析、需求分析、系統分析
- **設計**
  - 概念性設計
  - 細部架構設計
- **實施**
  - 代碼開發、功能測試
  - 系統安裝移轉、教育訓練
  - 系統維護

瀑布式開發

- **缺點**
  - 專案開始時，需求必須明確
  - 因為階段的轉移是循性的，階段完成前看不到工作成果
  - 初期的問題若無法早期發現，可能造成較大的修改成本
  - 某一階段延遲，會影響後續階段的進行
  - 缺乏彈性，開發時間一長，很可能無法應對環境 / 需求的變化

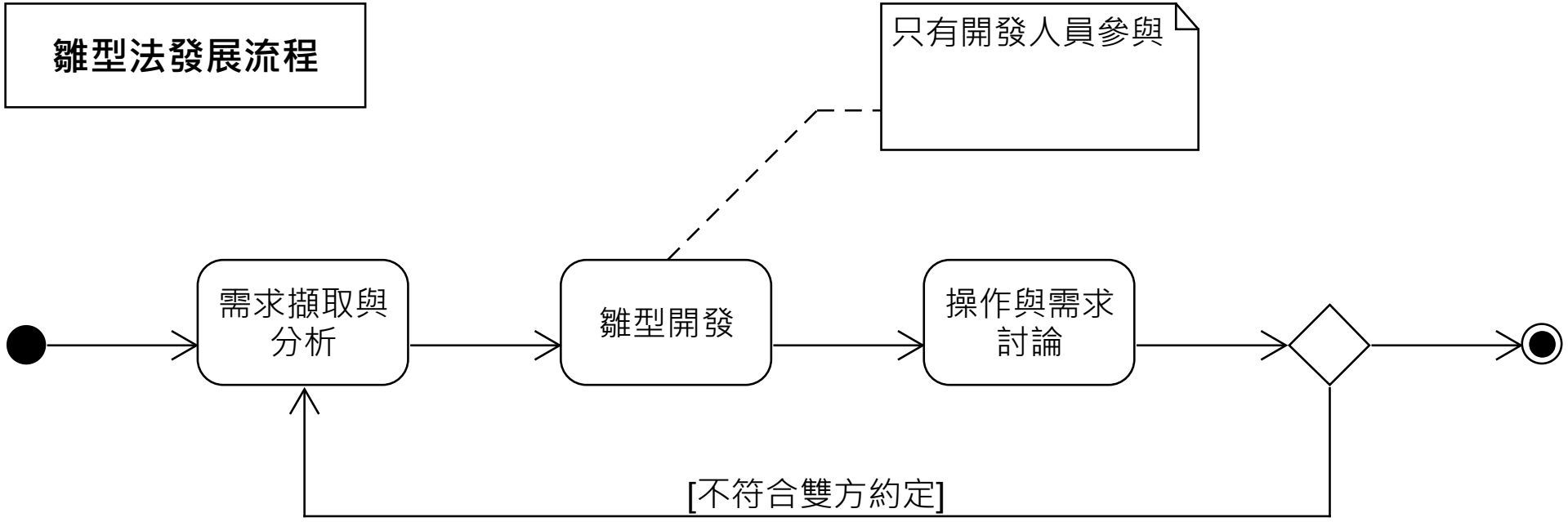
雛型式開發

- **1977 年提出**
- **透過雛型的開發，協助使用者…**
  - 儘快看到系統的重要功能
  - 引導使用者提出真正的需求
  - 不斷的評估修正後，得到最終的產品

雛型式開發

- **特色**
  - 適用於需求經常異動，或不清楚的情況
  - 使用者可以高度參與開發的進行
  - 透過雛型的展示，分析人員可以更好的與使用者溝通確認需求，減少認知上的差異
  - 使用者意見被尊重且反應於系統功能中，使用者的滿意度較高

雛型式開發



雛型式開發

- **缺點**
  - 只有雛型，難以看到系統的全貌
  - 缺少文件記錄，後續的維護困難
  - 雛型到下一個階段就被廢棄，所投入的資源造成浪費
  - 使用者高度參與，但人數多時反而造成意見分歧

螺旋式開發

- **1986 年 Boehm 提出**
- **取瀑布式與雛型式的優點**
  - 增加風險分析、評估及決策機制
- **開發工作如同螺旋般不斷演進推動**
  - 不斷的反覆進行，直至系統開發完成為止

螺旋式開發步驟

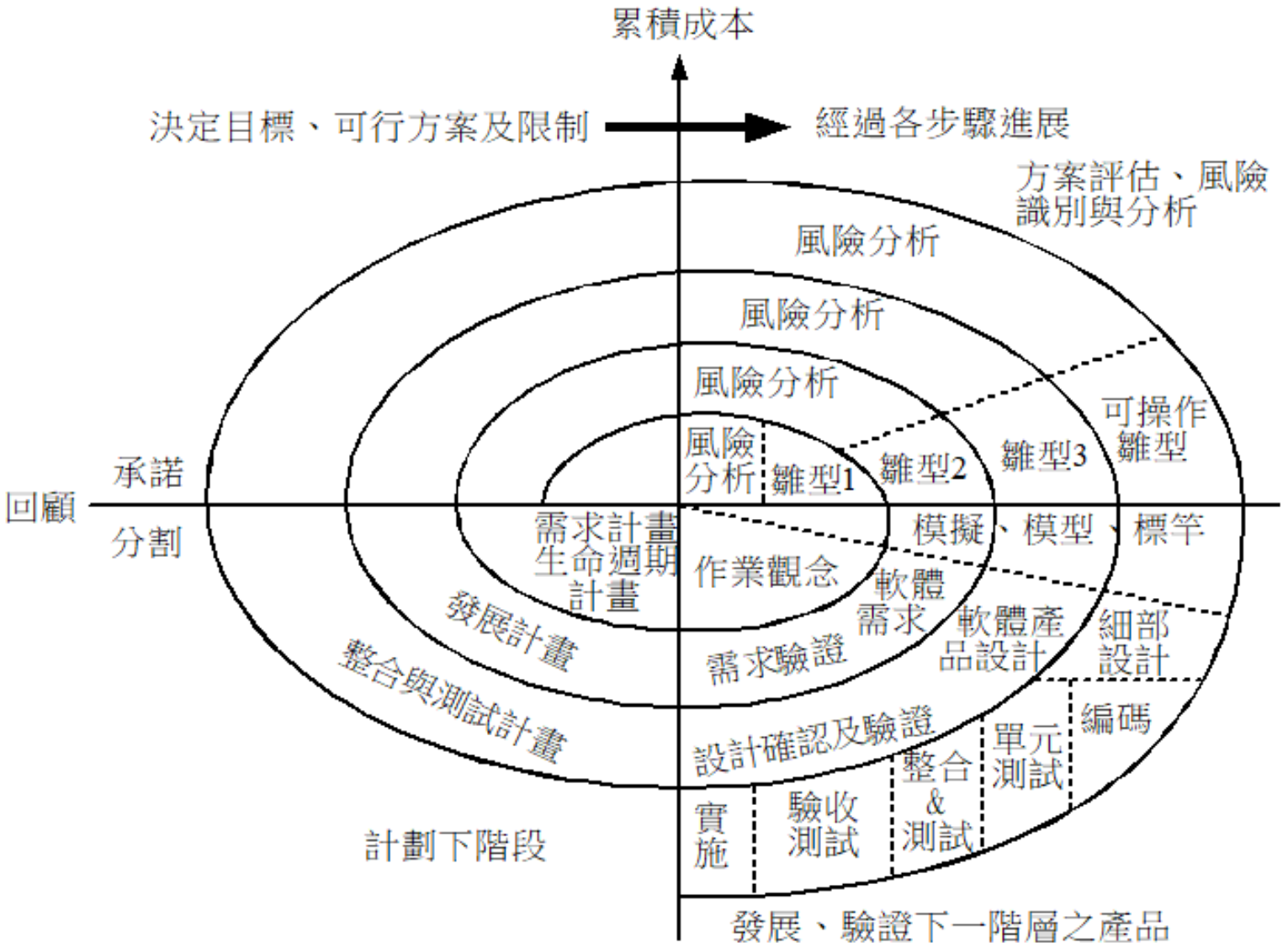
- **決定目標**
  - 找出系統的目標
  - 找出系統的實施方案
  - 確認實施方案的限制

螺旋式開發步驟

- **風險評估**
  - 找出不確定的因素
  - 對可能的風險進行調查與分析
    - 雛型、模擬、標竿、問卷等
  - 提出可能的風險解決方案

螺旋式開發步驟

- **開發、測試**
  - 依風險評估結果，選擇實施方式並執行
- **下一階段規劃**
  - 依此階段產出結果與使用者回饋，進行下階段開發內容進行規劃


累積成本
決定目標、可行方案及限制
經過各步驟進展
方案評估、風險識別與分析
風險分析
風險分析
風險分析
風險分析
雛型1
雛型2
雛型3
可操作雛型
承諾
回顧
分割
需求計畫
生命週期
計畫
作業觀念
模擬、模型、標竿
軟體需求
軟體產品設計
細部設計
發展計畫
需求驗證
整合與測試計畫
設計確認及驗證
編碼
單元測試
整合&測試
驗收測試
實施
計劃下階段
發展、驗證下一階層之產品

螺旋式開發

- **特色**
  - 開發雛型系統，可以了解是否與需求一致，降低不確定風險
  - 初期就對高風險的部份進行評估，可以早期發現錯誤，減少開發成本的浪費
  - 在實際開發代碼時，已有雛型可供參考或展示

螺旋式開發

- **缺點**
  - 風險的評估需要有經驗的人員進行
  - 若前一階段發生錯誤，可能會造成系統後期發展的困難

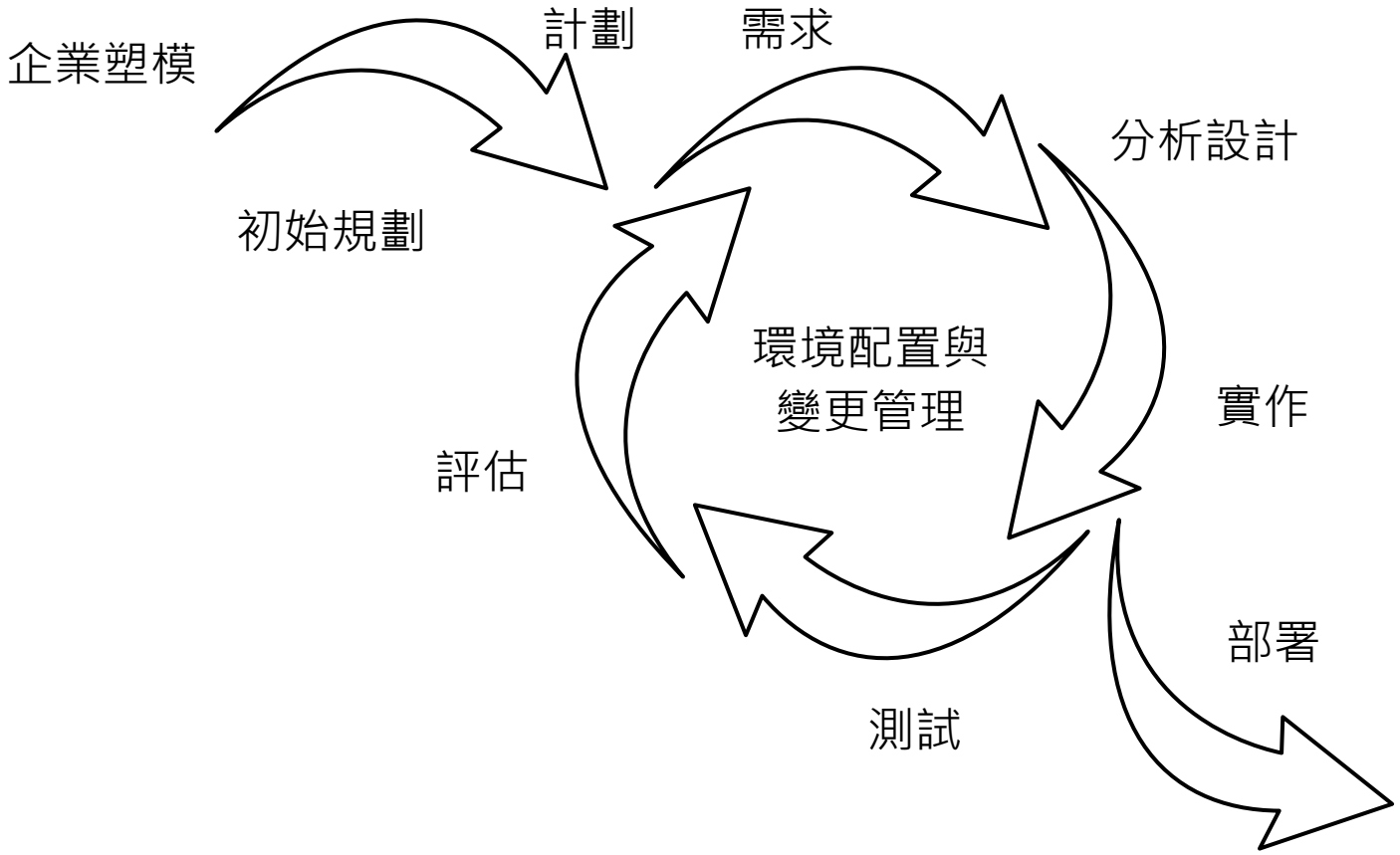
統一流程模式

- **Rational Unified Process; RUP**
- **1999 年，由 UML 三巨頭提出**
  - Grady Booch
  - James Rumbaugh
  - Ivar Jacobson
- **結合螺旋式的概念，以反覆漸增的方式進行軟體發展**
- **在每個反覆中進行風險評估，以盡早發現問題**

統一流程模式

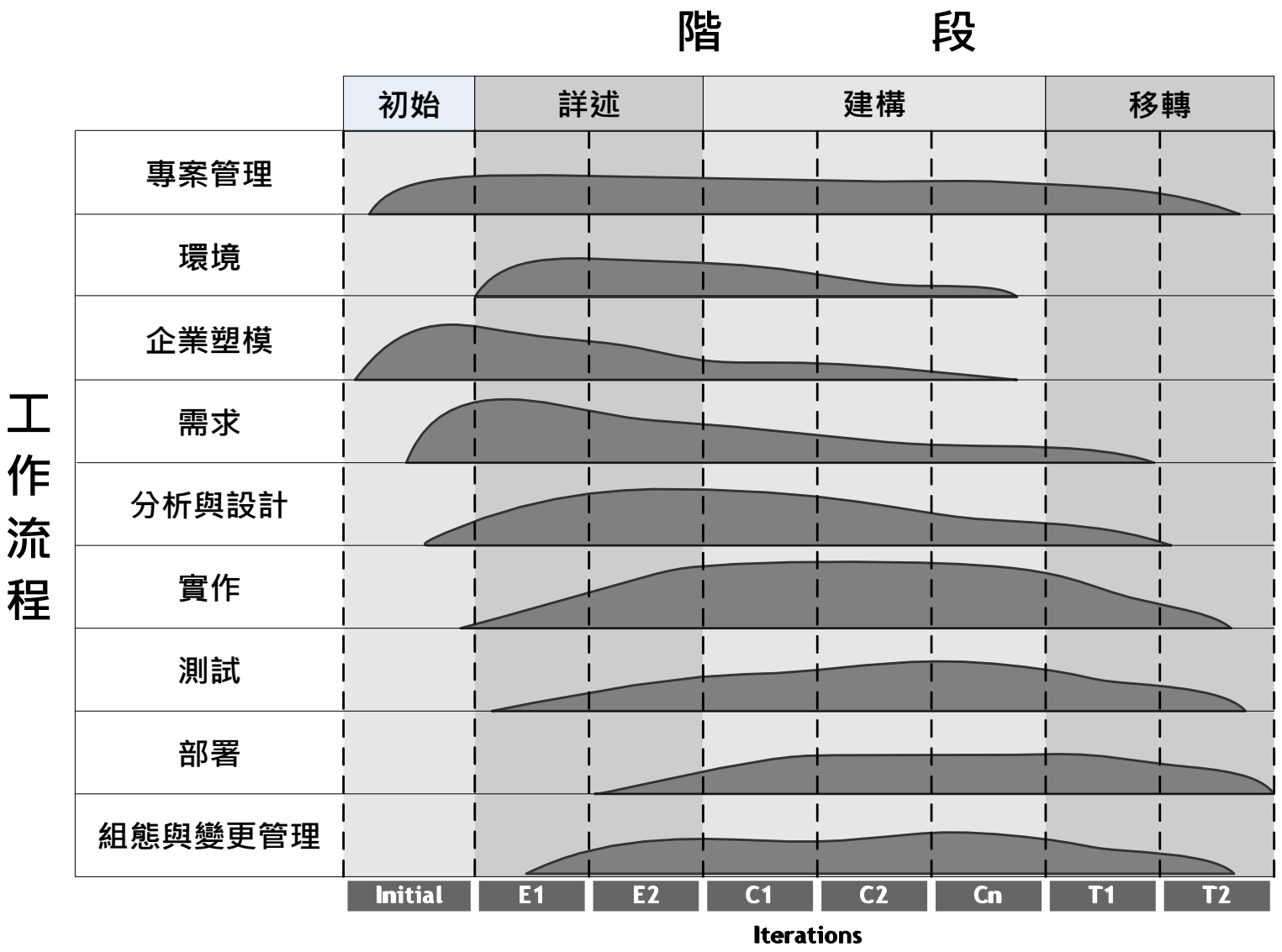
- **RUP 是什麼？**
  - 是一種軟體開發方法
  - 是一種軟體工程的處理流程
  - 是一種流程產品 (Process Product)
- **三個方向**
  - 由使用案例驅動
  - 以架構為中心
    - 4+1 Views
  - 反覆且漸進的方式進行
- **四個階段、九個流程**

統一流程開發程序



統一流程發展階段

- **初始階段**
  - 界定系統範圍，規劃成本與預算
- **詳述階段**
  - 降低主要的技術風險
  - 分析系統需求、設計系統架構
- **建構階段**
  - 元件開發、功能測試
  - 建構可運作的系統版本
- **移轉階段**
  - 安裝部署、教育訓練
  - 完成最終系統，並移交給客戶 / 使用者操作使用


階段
初始
詳述
建構
移轉
專案管理
環境
企業塑模
需求
分析與設計
實作
測試
部署
組態與變更管理
工作流程
Initial
E1
E2
C1
C2
Cn
T1
T2
Iterations

UML 介紹

UML 簡介

- Unified Modeling Language; UML
- 是一種標準化的塑模語言
- 是一種圖像化的語言
- 是一種開發用的溝通工具

UML 歷史

- 1980年開始，物件導向方法就已經被提出；但百家爭鳴的情況，反而讓使用者無所適從
- OMG試圖對這些方法論進行標準化（卻收到抗議信）
- 1995年，Rational公司找來Grady Booch及James Rumbaugh，以Booch的方法爲主，整合兩家的方法，意圖一統物件導向方法論（UML Ver. 0.8）
- 同年，Ivar Jacobson也加入了Rational的陣營
- 1997年，OMG釋出UML Ver. 1.0，後續又修正並釋出Ver. 1.1, 1.2, 1.3

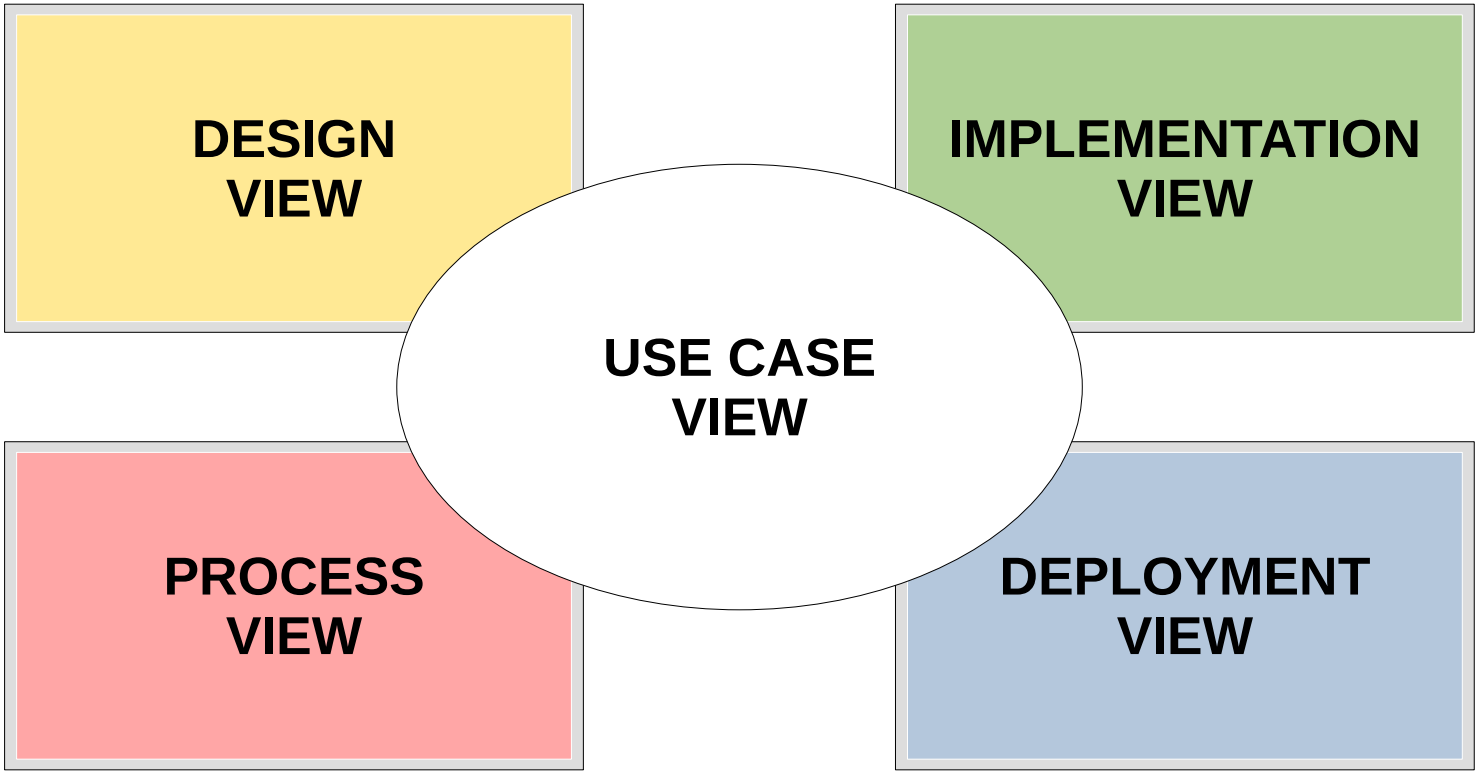
爲什麼要塑模？

- **管理複雜性**
  - 不再見樹不見林
  - 可以更專注於截取、記錄和傳達系統設計的重要資訊
- **模型是眞實事物的抽象**
  - 適度的簡化問題
  - 更快的了解問題

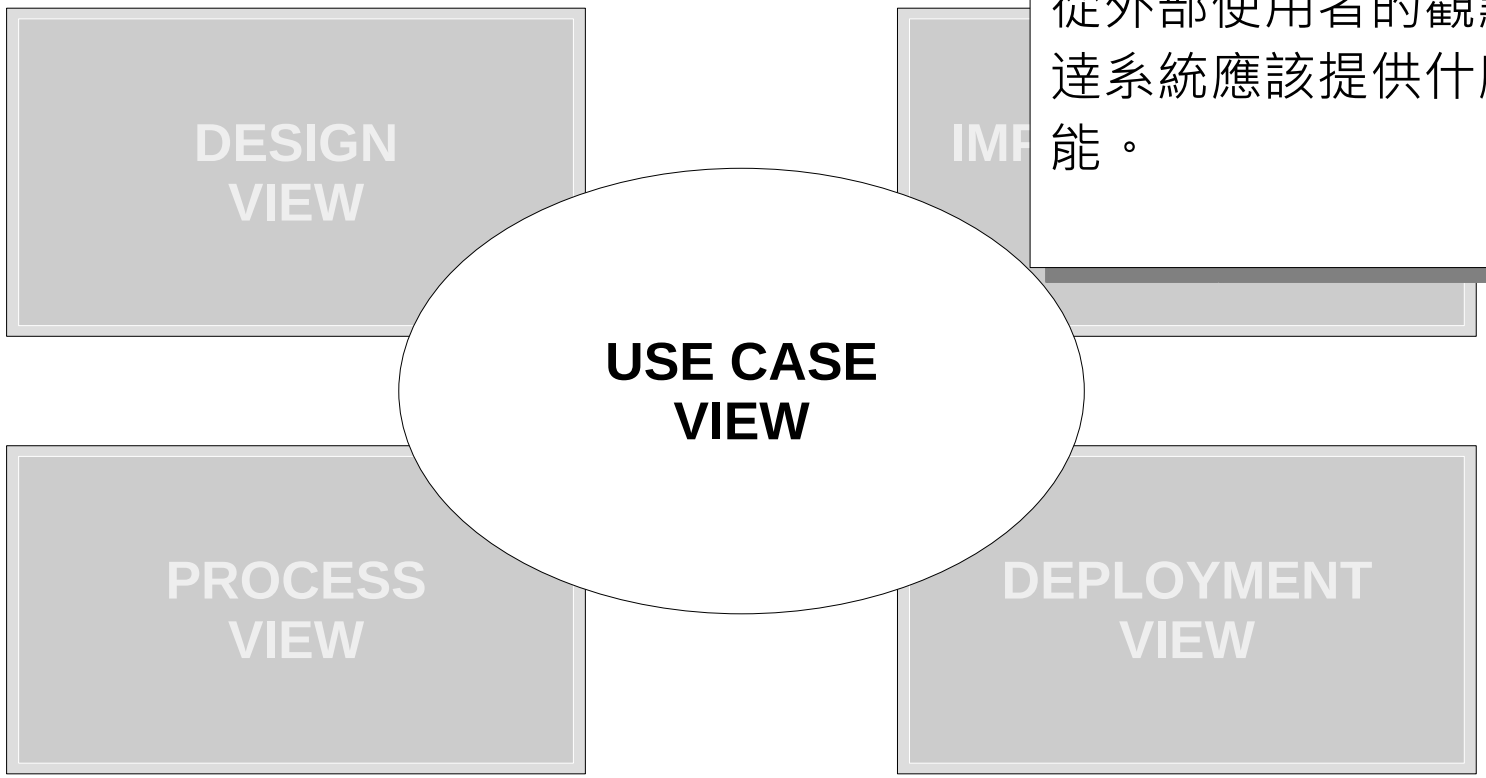
爲什麼要塑模？

- 建構模型比建構實物要來的容易且便宜
- 可以用來模擬，就算發生錯誤，也不會造成太大損失
- 可以幫助我們學習
- 可以是有效的溝通方式
- 可以用來表達不同層次的細節

UML 的 4+1 觀點

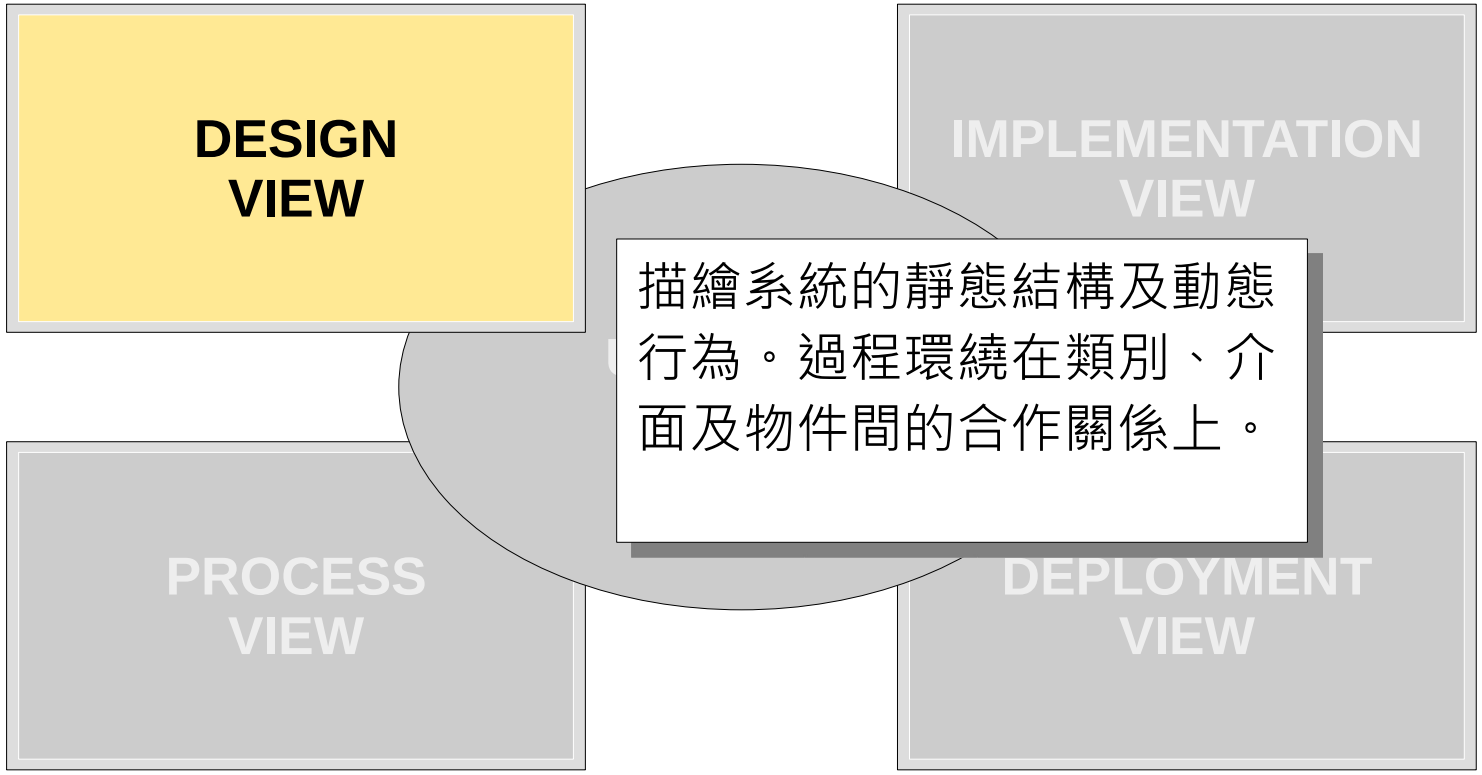

DESIGN VIEW
IMPLEMENTATION VIEW
USE CASE VIEW
PROCESS VIEW
DEPLOYMENT VIEW

UML 的 4+1 觀點

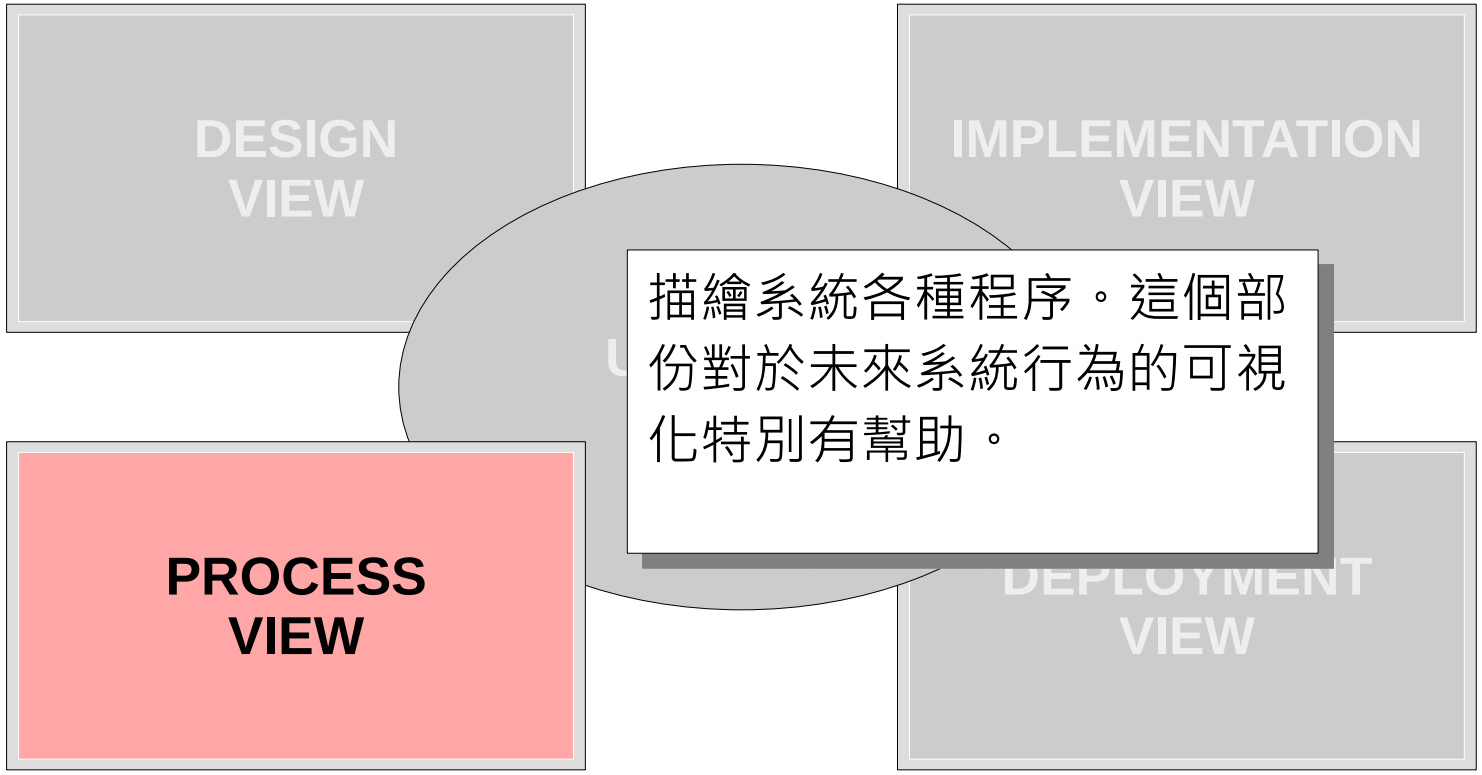


從外部使用者的觀點，來表達系統應該提供什麼樣的功能。

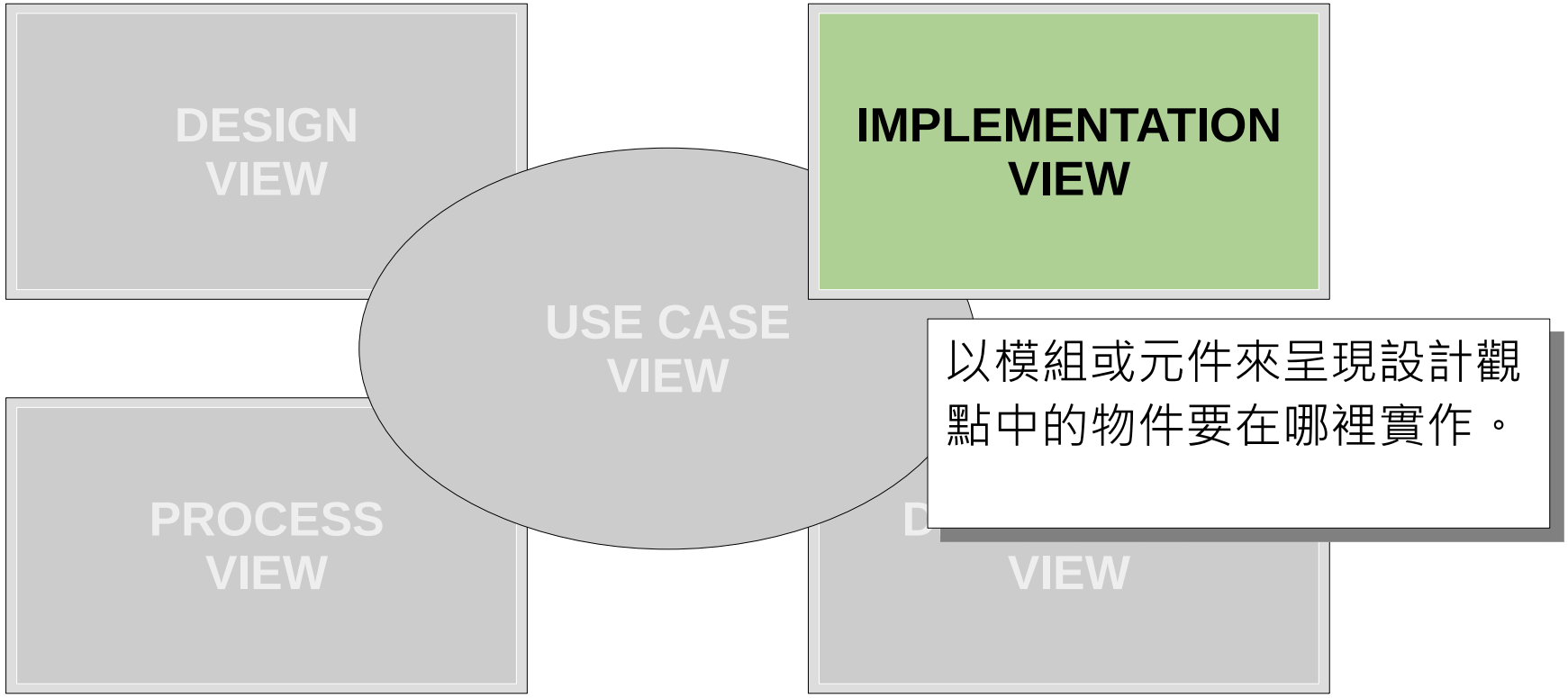
UML 的 4+1 觀點


DESIGN VIEW
IMPLEMENTATION VIEW
描繪系統的靜態結構及動態行為。過程環繞在類別、介面及物件間的合作關係上。
PROCESS VIEW
DEPLOYMENT VIEW

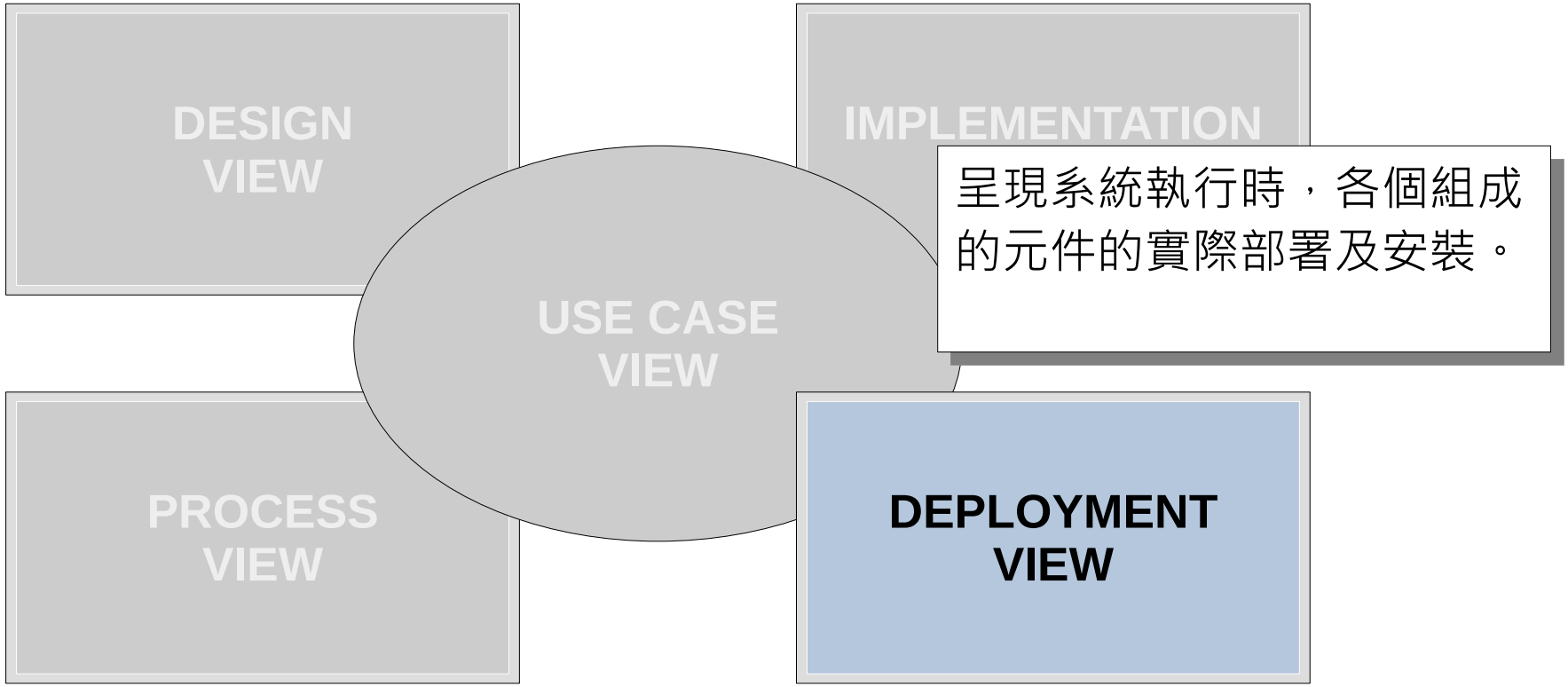
UML 的 4+1 觀點


DESIGN VIEW
IMPLEMENTATION VIEW
描繪系統各種程序。這個部份對於未來系統行為的可視化特別有幫助。
PROCESS VIEW
DEPLOYMENT VIEW

UML 的 4+1 觀點


DESIGN VIEW
IMPLEMENTATION VIEW
USE CASE VIEW
以模組或元件來呈現設計觀點中的物件要在哪裡實作。
PROCESS VIEW
VIEW

UML 的 4+1 觀點


DESIGN VIEW
IMPLEMENTATION
USE CASE VIEW
PROCESS VIEW
DEPLOYMENT VIEW
呈現系統執行時，各個組成的元件的實際部署及安裝。

UML 的 4+1 觀點

	靜態模型	動態模型
使用案例觀點	使用案例圖	互動圖、狀態圖、活動圖
設計觀點	類別圖、物件圖	互動圖、狀態圖、活動圖
流程觀點	類別圖、物件圖	互動圖、狀態圖、活動圖
實作觀點	元件圖	互動圖、狀態圖、活動圖
部署觀點	部署圖	互動圖、狀態圖、活動圖

UML 使用法

- **草圖**
  - 用來捕捉、表達及溝通系統中的關鍵點
  - 最常用的模式
- **藍圖**
  - 使用於正向或反向工程
  - 帶有系統設計的細節
  - PG 只依樣造輪
- **程式語言**
  - 透過 UML 詳述所有的系統細節
  - 需要 CASE 工具的搭配產生原始碼
  - 需要完全了解 UML 的所有細節

只有 UML 夠嗎？

- **UML 提供大量的圖**
  - 你可能不會全都用到
  - 或許可以使用不同的圖來搭配使用
- **如果沒有合適的圖或工具呢？**
  - 不為 UML 而 UML ，以完成分析工作為第一優先
  - 不考慮，直接使用其他不屬於 UML 的工具來取代