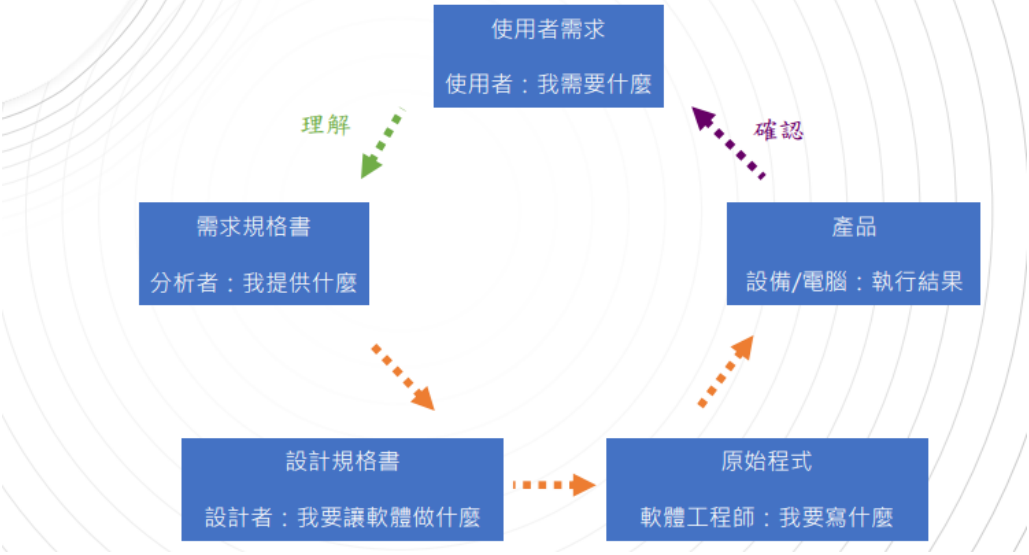
軟體開發生命週期



使用案例圖

■使用案例圖(use case diagram, ucd)：此圖可表達使用者對系統功能的期待，每個use case代表使用者認定系統應提供的某項功能。與該系統互動的人(使用者、維護者)或其他系統，在ucd中稱為角色(actor)。

狀態圖(State diagram)

■狀態圖是用來描述系統行為的技術
■狀態圖主要描述某個物件所有可能狀態，以及物件隨事件發生而改變的狀態
■狀態圖主要為一個類別而畫，用來描述一個物件在存活時的行為
■狀態圖要素
- ✧起始點(start point)：一個連結到物件的初始轉換
- ✧轉換(transition)：當狀態的活動完成後馬上發生
- ✧狀態轉換：需要有唯一的成立條件，即同一時間只能有一個事件條件成立

活動圖(Activity diagram)

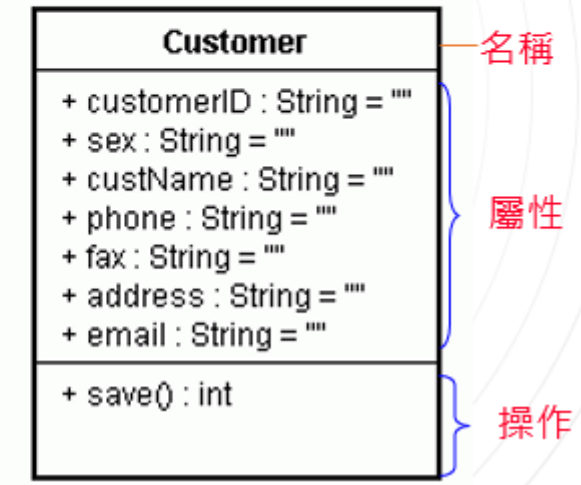
- ■表示一個做事的狀態，可能是一個程序或是一段副程式
- ■活動圖為狀態圖的衍生，所以表示符號與名詞大致一樣
- ■活動圖與狀態圖的差異
  - ✧活動圖：允許平行處理
  - ✧狀態圖：被限制只能做順序處理
- ■活動圖描述活動的發生順序，支援”條件式”和”平行”兩種行為
  - ✧條件式：
    - 分支(branches)：以成立條件來決定轉換分支路徑
    - 合併(merges)：結束分支，數個輸入合併成一個輸出
  - ✧平行：
    - 分叉(fork)：一個進入轉換和數個離開轉換，所有轉換同時平行發生
    - 會合(join)：所有進入轉換的狀態必須完成所有活動後，才會繼續進行後面動作

循序圖

- ■循序圖(sequence diagram)：依時間順序描述系統內部各成員之間的互動，通常可分成兩大類：
  - ✧描述使用情境：主要用於系統分析階段，著重在角色(actor)與系統之間的互動，將系統當成一個黑盒子，通稱為系統順序圖。
  - ✧描述方法邏輯：主要用於系統設計階段，著重在物件之間的訊息傳遞。

類別圖結構

■類別封裝了屬性及操作



- 聚合(Aggregation)是一種特殊的關聯關係，它是用以表達「整體和部份」的關係，它也隱含著所謂包含(Include)的關係。
- 因此，聚合的關係可以用英文的“is-part-of”，“has-a”或是“has-parts”語意上的關係來表示。
- 換言之，聚合的關係不只是指出物件相互了解的關係，更是它們被組合起來以形成一個新的更複雜的物件。
- 聚合關係以一個空的菱形來表示代表整體的一方。
- 一個球隊有球員，球隊如果沒了，球員還可以到別的球隊打球(聚合)。

組合(Composition)關係是一種比聚合關係更強的包含關係。

在一個聚合的關係中，如果整體的消失會造成部分(Parts)的消失，那麼這個聚合是一種組合關係。

組合關係以一個實心的菱形來表示代表整體的一方。

一本書包含有很多章節，如果書沒了，章節也就沒了(組合)

Principles that Action Practice

Principle #1. ***Divide and conquer.*** **(化繁為簡)**
Principle #2. ***Understand the use of abstraction.*** **(提綱挈領)**
Principle #3. Strive for consistency. (一致性)
Principle #4. ***Focus on the transfer of information.*** **(界接介面)**
Principle #5. ***Build software that exhibits effective modularity.*** **(模組架構)**
Principle #6. ***Look for patterns.*** **(模式樣板)**
Principle #7. ***When possible, represent the problem and its solution from a number of different perspectives.*** **(換位思考)**
Principle #8. ***Remember that someone will maintain the software.*** **(維護彈性)**

Communication Principles

Principle #1. *Listen.* **(聆聽)**
Principle # 2. *Prepare before you communicate.* **(事情準備)**
Principle # 3. *Someone should facilitate the activity.* **(決策者)**
Principle #4. *Face-to-face communication is best.* **(當面溝通)**
Principle # 5. *Take notes and document decisions.* **(記錄)**
Principle # 6. *Strive for collaboration.* **(團隊合作)**
Principle # 7. *Stay focused, modularize your discussion.* **(專注聚焦)**
Principle # 8. *If something is unclear, draw a picture.* **(圖優於文)**
Principle # 9. *(a) Once you agree to something, move on; (b) If you can't agree to something, move on; (c) If a feature or function is unclear and cannot be clarified at the moment, move on.* **(持續往前)**
Principle # 10. *Negotiation is not a contest or a game. It works best when both parties win.* **(妥協創造雙贏)**

Requirements Engineering

#1 Inception (建立)
#2 Elicitation (啟發)
#3 Elaboration (制定)
#4 Negotiation (談判)
5 Specification (規格)
#6 Validation (驗證)
#7 Requirements management

需求分析

- 功能性需求
  - 功能的需求主要是在描述系統該做什麼。也就是系統要提供給使用者的服務項目。
  - 對於系統所提供之功能的描述可以包括什麼樣的輸入是這個功能所必須的、這個功能的處理流程與步驟、以及經過資料處理後，這個功能的輸出為何等等。

- 非功能性需求
  - 非功能的需求是指跟系統的執行效率，效能之需求，且可以量度的(measurable)的項目。

需求分析

- 舉例說明
  - 一個線上購物系統，最明顯的事件就是”顧客下訂單”。我們可以分析出：事件名稱就是顧客下訂單。訂單是顧客下的，所以事件的來源來自於顧客。這個事件被觸發是因為訂單資料的到來，這是來自來源的一種要求。對於這個事件，系統應該要產生一筆新的訂單記錄，以記錄這個活動。所以，” 產生一筆訂單”是活動的名稱。而” 訂單編號”是這個活動的回應，此回應的目的地是使用者。

訂單產生之後要將訂單資料傳送到出貨部門

事件名稱	觸發器	來源	活動	回應	目的地
顧客下訂單	訂單	顧客	產生一筆新訂單	訂單編號訂單	使用者 出貨部門