* **軟體**
  1. 既是產品也是工具
  2. 是工程而非製造
  3. 不會耗損，但會漸漸不合乎使用
  4. 工業正朝以零件為基礎的方向前進，但大多數軟體仍是訂製
* **Software Engineering**
  1. 為了以經濟的方式獲得可靠和高效的軟體，建立完善的工程原理
  2. 一個以系統的且可量化的方式來開發、運行、維護的軟體應用程序
  3. 包括過程、管理技術、技術性的方法、工具的使用
* **Software Process**
  1. 高品質的軟體，需要定義活動、任務的架構
  2. 定義軟體設計時的方法
  3. 被那些非常有創意且非常具有知識的軟體工程師所設計出來的軟體，是能滿足市場需求的
* **Umbrella Activities護傘活動**
  1. 軟體專案追蹤和控制
  2. 危機管理
  3. 軟體品質保證
  4. 正規技術評論
  5. 量測
  6. 軟體規劃管理
  7. 重複使用性管理
  8. 工作產品準備和製造
* **Process Flow操作流程**
  1. 描述五個框架的活動及任務
     1. 溝通
     2. 計畫
     3. 建模
     4. 建造
     5. 部屬
  2. 線性流程是用循序的方式來執行，從溝通開始部屬結束
  3. 疊代流程是用循環的方式來執行，藉由每個循環去建立更完整的版本
  4. 平行流程是在一個項目中同時進行很多的活動
* **Prescriptive Process Models規定的流程模型**
  1. 一開始就先提出軟體開發的混亂
  2. 帶來了軟體工程工作，並且提供軟體團隊合理的引導
  3. 無法在一個不斷變化的計算環境下，對軟體發展的問題有明確的答案
* **Software Process Models軟體流程模型**
  1. 瀑布模型(Waterfall Model)
  2. 增量模型(Incremental Models)
  3. 演化模型(Evolutionary Models)
  4. 原型模型(Prototyping Model)
  5. 螺旋模型(Spiral Model)
  6. 並行開發模型(Concurrent Development Model)
* **Agility**
  1. 有效的回應需求改變
  2. 有效地和利益相關者溝通
  3. 顧客和團隊億體
  4. 組織團隊控制工作表現
* **An Agile Process**
  1. 由使用者需求所驅動
  2. 擬定短期計畫
  3. 疊代開發軟體，側重於精隨
  4. 產出多個軟體單位
  5. 接受改變，盡力檢討
* **Human Factors**
  1. 為了符合人們和團隊的需求
  2. 關鍵在機敏團隊
     1. 競爭
     2. 共識
     3. 合作
     4. 決策能力
     5. 解決問題能力
     6. 互信互重
* **Extreme Programming**
  1. 使用最廣泛的機敏過程
  2. XP planning
     1. 由使用者案例著手
     2. 評估每個訴求再分配費用
  3. XP design
     1. 越簡單越好
     2. 鼓勵使用CRC卡
  4. XP coding
     1. 先確立測試方法
  5. XP testing
     1. 每日自行部分測試
     2. 使用者驗收
* **Principles that Guide Practice**
  1. 機敏
  2. 注重質量
  3. 準備適應
  4. 建立一個有效的團隊
  5. 建立一個溝通協調機制
  6. 管理變革
  7. 評估風險
  8. 創造可為他人提供價值的商品
* **Communication Principles**
  1. 聆聽
  2. 事前準備
  3. 會議要有一leader，以利會議朝正確方向進行
  4. 紀錄會議要點
  5. 面對面溝通最好
  6. 爭取合作
  7. 談判不是比賽or遊戲，最好是雙贏
* **Requirements Modeling Principles**
  1. 問題須被了解
  2. 軟體功能需被定義
  3. 要表示軟體行為
  4. 描述資訊、功能、行為時須以分層方式進行
  5. 分析任務須從基本訊息and實作細節開始
* **Testing Principles**
  1. 測試應源於客戶的需求
  2. 開始之前應計畫良久
  3. 測試應由小到大
  4. 詳盡的測試是不可能的
  5. Pareto原則適用於軟體測試
* **Deployment Principles**
  1. 軟體須加以管理
  2. 軟體交付前須建立支持的制度
  3. 有bug的軟體應先修復再交付
  4. 適當的教學材料應交給最終用戶
* **Class 與object的關係**
  1. 延伸 ADT 以設計軟體
     1. 產品是根據抽象資料型態所設計的
     2. 變數(物件) 是抽象資料型態的例子-- objects有各自 data area ,但共用 procedures codes
  2. 定義 class 及延伸 code sharing
     1. Class: 支持繼承的抽象資料型態
     2. Objects 是 classes的例子
* OOP V.S. SP
  1. OOP
     1. 使用多型和繼承的觀念
     2. 將資料抽象化：data、operator
     3. 封裝的特性：object、class
  2. SP
     1. 將重複的程式片段抽出成為函式
     2. 在main裡透過呼叫函式來執行
* **組織運作與 object-oriented program**
  1. 一個組織中,
     1. 每個人都有一職稱，且有規定的工作項目，工作內容可以不同，但工作方式需按職務工作程序的規定執行

🡪object 及 class 的對應關係

* + 1. 一個工作人員可叫另一工作人員執行工作，且必需告知相關資訊 🡪message passing
    2. 一個工作人員叫另一工作人員執行工作時，可等待其完成或繼續執行其他工作(multi-thread)
    3. 當組織執行其特定任務時，將有一群人依規定順序執行其任務
    4. 🡪OO program執行某一功能
    5. 有一 C程式需撰寫,，任一有空的程式設計師均可交付其完成 🡪polymorphism
    6. 組織中有單位，各單位有其特定功能，各單位間有一定的協調對象及方式
    7. 🡪subsystem的觀念
  1. 多揣摩組織與OO program的對應關係
  2. 繼承：一個重要的物件導向語言特性
* **Requirements Engineering**
  1. 初始：問一系列關於建立的問題
  2. 啟發：徵求所有利益相關者的問題
  3. 制定：創建一個可以確定數據、功能、行為需求的分析模型
  4. 協商：同意一個為了開發者和消費者的交付系統
  5. 規格：可以是一個或多個
     1. 書面文件
     2. 一組模型
     3. 正式的數學
  6. 驗證：一個審查機制
  7. 需求管理
* **Eliciting Requirements**
  1. 會議由軟體工程師和客戶雙方出席且舉辦
  2. 建立籌備和參加原則
  3. 建議的議程
  4. 目標
     1. 找出問題
     2. 提出解決方案的要素
     3. 以不同的方法談判
* **Elicitation Work Products**
  1. 必要性和可行性說明
  2. 有範圍的開發，限於系統or產品
  3. 要有一消費者、使用者、利益相關者的名單
  4. 該系統技術環境的描述
  5. 要有一需求and限制的清單
* **analysis model**
  1. 基於場景的元素
     1. 處理軟體的功能敘述
     2. 描述相互作用的執行者與系統
  2. 基於類別的元素
     1. 隱含在場景內
  3. 行為元素
     1. 狀態圖
  4. 流程導向元素
     1. 數據流程圖
* **Requirements Analysis**
  1. 指定軟體的操作特性
  2. 說明與其他系統元件的軟體介面
  3. 軟體須符合建立的限制條件
  4. 讓軟體工程師
     1. 詳細說明早期工程的基本要求
     2. 建構模型以描述使用者案例
* **Domain Analysis**
  1. 定義一個領域進行調查
  2. 收集領域應用中的代表性產品
  3. 分析樣品中的每個應用程序
  4. 開發一個分析模型中的對象
* **analysis class**
  1. 檢查使用場景，建立一部分的需求模型，並執行語法分析
     1. class由每個名詞或名詞短語來強調，並用簡單的表來確定
     2. 同義字應被注意
  2. 然而一但名詞都被隔離，我們該如何尋找
* **CRC Models**
  1. 代表class的標準索引card的集合
  2. card分為三部分
     1. 沿著card的頂端寫class的名字
     2. 在body的左邊列出class的責任
     3. 在body的右邊列出class的合作者
* **Class Types**
  1. 實體類別
  2. 邊界類別
  3. 控制類別管理