課程介紹

課程規劃：

上課時間		講授內容	作業繳交時間與範圍
第2次	9/29	第1～3講	
第4次	10/20	第4～６講	第1次(三週刊第1～6講作業題目)
第6次	11/17	第７～12講	第2次(三週刊第7～12講作業題目)
第8次	12/8	第13～14講	第3次(三週刊第13～14講作業題目)

成績評量：(1)期中隨堂考成績30%
(2)期末隨堂考成績40%
(3)平時成績30%(作業成績、上課表現)

隨堂考範圍：三週刊作業及上課講義；期中：1-６講，期末：７-14講

期中隋堂考時間：１０／２０　下午２：３０－３：２０

期末隨堂考時間：１２／８　下午２：３０－３：２０

未參加隨堂考，請自行洽課務組確認補考時間

作業繳交：(1)作業請以電子檔繳交，上機操作題請檢附操作過程。
(2)作業電子檔命名範例：10714001王曉華SA第1次作業
(2)作業共須繳交3次，請學藝股長於以上規定時間前收齊電子檔於課堂上交給老師。作業遲交扣分；未交以0分計。

什麼是系統分析與設計

- 系統分析與設計是一門關於開發資訊系統的知識領域
- 系統分析主要以研究目前的系統，了解其運作模式以及如何滿足使用者的需求為重點
- 系統設計是以系統分析的結果為基礎，改善現有的系統或開發新的系統來滿足使用端的需求

系統開發

- 各系統之特性、大小、開發方法或技術可能不同，但系統開發之過程可歸納出一些基本而共同的步驟或階段
- 系統開發三階段

資訊系統開發

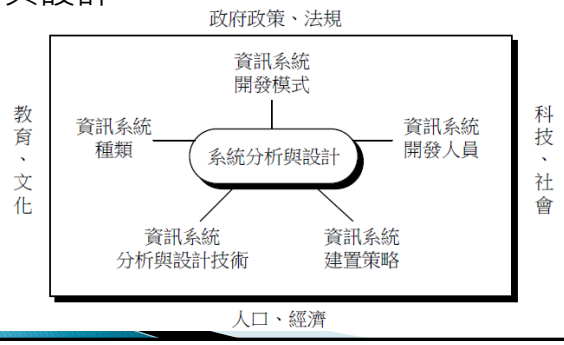
- 資訊系統開發其實就是要運用資訊科技及資訊系統開發方法來建構實體的或邏輯的系統，以協助人們解決資訊處理的需求。
- 成功的資訊系統開發必須把握以下幾個重要原則：系統的目標應該明確定義。
  - 系統的目標要實際而有用。
  - 系統開發要充份運用科技。
  - 系統開發要依循一定的方法。
  - 系統開發必須獲得足夠的資源及支持。
  - 系統開發必須符合限制條件。
  - 系統開發必須考慮環境因素。

資訊系統開發面臨之挑戰

- 資訊科技的快速進步，使得淘汰率驟增、系統的可用壽命變短、技術支援無法持久。
- 經營環境之快速變遷，使得需求亦經常變動。
- 資訊系統開發經常面臨時程延誤、成本超支、品質不良等問題。
- 人們重視看得見的硬體，而常忽略看不見的軟體。

資訊系統開發環境

- 資訊系統開發環境所涉及的層面很廣，包括資訊系統的開發模式、種類、建置策略、開發人員、系統分析與設計技術等。
- 外在大環境的科技、社會、教育、文化、人口、經濟、政府政策與法規等因素，也直接或間接影響系統分析與設計。



資訊系統建置策略

- 資訊系統建置策略乃指資訊系統之建立、修改、擴充或更新等所採取之方式。以系統建置過程所涉及到的主要設計者來區分, 資訊系統之建置策略可分成三種：
  (1)由公司內部獨立完成。還可以再區分為：
  a.使用者自建(End User Computing)。
  b.由公司資訊部門自行開發。
  c.由相關部門人員組成任務編組開發。
  (2)由公司外部取得, 我們可以採取的管道則有：
  a.委外開發(Outsourcing)。
  b.購買現成之套裝軟體(Application Package)。
  c.引進同業之系統。
  (3)其他方式: 資訊系統之建置當然也可以經由綜合上述各種策略, 或由部份同業聯合共同找資訊公司開發等。

資訊系統開發之相關人員

- 要產生出成功的軟體產品須要經由一群人相互的執行、討論、測試、整合而成。人在系統開發過程中扮演了舉足輕重的角色。
- 資訊系統開發之相關人員包括：
  - 系統分析師(**System Analyst**)
  - 程式設計師(**Programmer**)
  - 終端使用者(**End User**)
  - 企業經理(Manager)
  - 資訊系統經理(**IS Manager**)
  - 資料庫管理者(**Database Administrator**)
  - 其他技師(**Technician**)等。

資訊系統開發之相關人員

- 系統分析師
  - 擷取使用者需求，並能清楚且完整地瞭解及表達需求。
  - 進一步將需求轉換成資訊技術、企業處理與知識等元件，並將之有組織地結合起來。
- 程式設計師
  - 主要是依分析與設計之藍圖設計出程式、建立資料庫、測試與安裝系統等。
  - 系統分析與設計的過程中，必須和各種人員保持良好的溝通、互動與互信，以確保系統開發之成功。

資訊系統開發之相關人員

- 終端使用者
  - 是問題領域之專家，但可能並非資訊科技方面之專家，其於系統分析與設計過程中，主要扮演提供使用者需求與企業知識的角色。
  - 因此，終端使用者應能主動、積極地參與系統開發，並盡可能完整與清楚地表達需求和知識，而非被動地等待系統分析師之詢問。
  - 系統分析師要與終端使用者一起工作，並將其需求和企業上的知識轉換成可支援終端使用者工作的資訊系統。

資訊系統開發之相關人員

- 企業經理
  - 是終端使用者之高層主管，這些人沒有很多的時間參與系統的開發，但他們在系統開發的過程中扮演了非常重要的角色，例如承諾對系統開發所需之財力、人力、時間等資源的投入及提出組織之政策與限制等。
  - 由於他們的決策權與對企業經營的專業知識，部門的領導者與執行長也能對資訊系統專案發展設定一般性的需求與限制。
  - 系統分析師必須要這些人保持良好的溝通，瞭解他們的期望和優先順序，並從他們口中瞭解組織內的限制和資源。

常見的資訊系統

- 交易處理系統
- 管理資訊系統
- 決策支援系統
- 高階主管資訊系統
- 專家系統
- 企業資源規劃系統
- APP
- 遊戲開發

交易處理系統

- 商業交易(transactions)是企業流程中最基本的重要關鍵事件。當企業與他的供應商、客戶、夥伴、員工,以及政府互動時,他們是主要的方式。交易幫助企業掌握或產生相關的重要資料。主要的交易例子包括採購、訂單、銷售、預約、註冊、提貨單、裝運、發票,以及付款等等。
- 交易處理系統亦稱資料處理系統(DataProcessingSystem,DPS),該系統主要之目的是將大量的交易處理自動化。此系統的兩種主要功能為交易記錄之保存與交易表單之產生。
- 舉例來說,POS(PointofSale)系統之前檯系統、加油站之加油作業與收銀系統,金融機構之櫃檯系統等屬於交易處理系統。

管理資訊系統

- 管理資訊系統通常架設在交易處理系統之上,用來提供規劃、監督與控制企業運作所需要的管理報表。這些報表通常按照既定的行事曆來產生,並以預先設定的格式呈現。
- 管理資訊系統主要之目的是提供不同層級的管理者有關組織營運狀況不同摘述程度之報表。該系統的兩種主要功能是交易資料之記錄保存與摘述性報表之產生。
- POS系統之後檯系統是屬於管理資訊系統。

決策支援系統

- 主要目的為支援決策者，提升其決策效率**(Efficiency)**與效能**(Effectiveness)**。
- 決策支援系統主要支援半結構化或非結構化之決策活動，其特徵有：
  -能以即興**(Ad Hoc)**、自訂或標準化的方式分析資料與產生報表。
  -能直接與決策者互動。
- **POS**之後檯系統，除了固定式地資料查詢、處理、分析與報表產生外，若還能與使用者互動，並依其需求擷取、分析與展示資訊，則該系統可稱決策支援系統。

高階主管資訊系統

- 高階主管資訊系統是專門設計來提供給高階管理人員使用的決策支援系統。
- 高階主管資訊系統是針對高階主管之資訊需求而設計,其目的是希望高階主管能直接從電腦中及時得到其所需之關鍵資訊,而不需透過中介使用者。
- **EIS**之特徵包括可過濾、摘述關鍵資訊，亦可將資訊以多種方式作圖形化的展示（如儀表板、長條圖），也得以由上而下之方式擷取資訊並進行分解。

專家系統

- 初期發展目的是用以模仿人類專家解決特定問題之能力，並希望專家系統所提供之答案或建議可達人類專家水準。
- 專家系統包含三個主要元件：
  - –使用者介面：為專家系統與使用者交談之媒介。
  - –推理引擎：為專家系統依使用者要求，由知識庫推論出結果或建議之機制。
  - –知識庫：為系統儲存專家知識之處。
- 現今，專家系統不再強調取代專家，而是支援專家，因此專家系統也漸成為另一種形式之決策支援系統。

企業資源規劃系統

- 為一種將多項企業功能整合在一起的模組化與架構化套裝資訊系統，藉由及時整合與掌握企業分散於各地的資源，提供最佳流程典範**(Best Practice)**，以降低企業營運成本並提升客戶服務滿意度。
- 狹義的觀點視**ERP**系統為整合企業內部資源規劃之系統。
- 廣義的觀點視**ERP**系統為整合企業內外部資源規劃之系統，例如企業上下游之供應鏈管理等。

資訊系統種類及特性

資訊系統種類	資訊系統特性
交易處理系統	➢將大量的交易處理自動化 ➢處理程序與資訊需求非常結構化
管理資訊系統	➢提供不同層級之管理者摘述程度不同之報表 ➢資料的處理與報表的產生亦多為結構化
決策支援系統	➢支援決策者半結構化或非結構化之決策 ➢提升決策之效率與效能
高階主管資訊系統	➢支援高階主管即時瞭解所需之關鍵資訊 ➢可過濾、摘述關鍵資訊 ➢可將資訊以多種方式作圖形化的展示
專家系統	➢針對特定應用範圍或領域，集合不同專家知識而成的系統 ➢希望專家系統提供之答案或建議可達人類專家之水準
企業資源規劃系統	➢將多項企業功能整合在一起 ➢可即時整合與規劃企業分散於各地之資源 ➢提供最佳的流程典範

系統分析與設計的技術

- 系統分析與設計是一系列有組織之處理程序，目的是將使用者或企業需求轉換成有組織的資訊科技、企業流程與知識等元件。
- 傳統典型之系統開發技術一般而言即是自1960 年代所流行之結構化軟體發展。這些技術包括有結構化分析與設計、結構化程式設計、以及由上而下發展模式。
- 在軟硬體環境逐漸複雜的情況下, 物件導向程式設計觀念在1960年在一些語言的發展中即可發現。物件導向程式設計在透過強調可重複性解決一定程度的軟體維護問題。

結構化分析與設計

- 結構化技術主要用於系統開發過程之分析與設計階段，幫助系統分析師進行資訊系統之描述與驗證。
- 過程中將企業流程與資料分開處理。
- 以結構化塑模工具幫助系統分析師進行資訊系統之描述與驗證。
- 應用結構化技術時，常用之流程與資料塑模(Modeling)工具有：
  - 事件(Event)
  - 環境圖(Context Diagram)
  - 資料流程圖(Data Flow Diagram, DFD)
  - 資料字典(Data Dictionary, DD)
  - 處理規格描述(Process Specification)
  - 實體關係圖(Entity-Relationship Diagram, ERD)

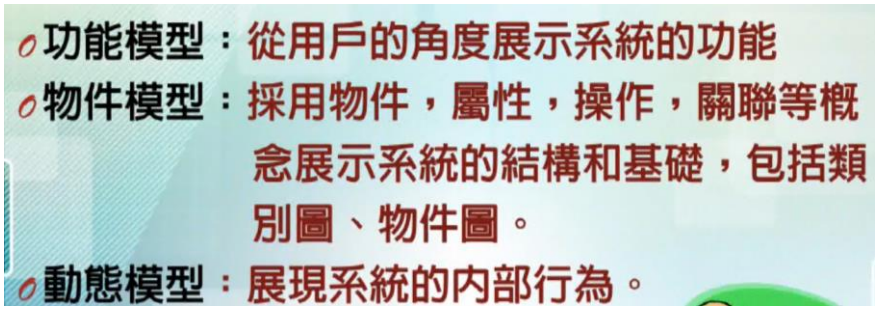
物件導向分析與設計

- 在1960年代時,在軟硬體環境逐漸複雜的情況下,程式設計領域開始面臨著軟體維護危機。物件導向程式設計觀念,透過強調可重複性在一定程度上解決軟體維護這一問題。
- 物件導向程式設計在1980年代開始慢慢成為一種程式設計主流思潮,主要應歸功於C語言的擴充版,C++,廣受系統設計人員採用的緣故。在圖形使用者介面(GUI)日漸崛起的情況下,物件導向程式設計剛好迎合使用者需要的視窗介面應用潮流。
- 物件導向技術是將企業流程與資料封裝成物件，而不像結構化技術是將企業流程與資料分開處理。
- 常用的塑模工具為統一塑模語言。

統一塑模語言(UML)

- 統一塑模語言(UnifiedModelingLanguage,UML)是非專利的第三代塑模和規約語言。UML是一種開放的方法,用於描述、視覺化、構建和編寫一個正在開發的、物件導向的、軟體密集系統的專案的方法。對這三個縮寫字母,我們可以簡單的說明如下：
- Unified：UML是一種標準語言,廣泛運用於全世界。
- Modeling：UML用途在於塑模(Modeling),也就是畫軟體藍圖。
- Language：UML是一種塑模語言,而非程式語言或標示語言。
- UML是軟體系統發展人員用以建造模型,而這些模型使得工作團隊能夠：將系統具象化(Visualization)、將系統結構及行為規格化(Specification)、建構(Construction)系統、以及記錄(Documentation)發展系統過程中之各項決策。
- UML常用來作為形塑物件導向分析和設計(OOAD)過程的產物的圖形化語言。它為物件導向設計中的需求、行為、體系結構和實作提供了一套綜合的表示法。UML是物件導向建模語言的標準,適用於以物件導向技術來描敘任何類型的系統。

統一塑模語言(UML)



統一塑模語言(UML)

- 在資訊系統開始設計之初,常常不夠明確具體；完整物件導向設計模型,可以較順暢地適應原始程式碼的製作；架構可以先建立起來,再慢慢提供細部物件之間的操作,訊息傳遞等等。
- 實際的系統必須適應系統實際實作的環境。為了達到這個目的,分析模型必須轉換成設計模型,且將不同的因素考慮在內。反映出真實世界需要的資訊模型恰好是物件導向設計的核心基本概念。
- 經正確分析、類比實際環境後,我們也將針對在分析規劃階段所得的物件分析模型,做出正式初步分析文件。進行較細部建立系統分析時,我們可以對使前階段的初步模型做出必要的修正與改進,使系統反應實際的規劃、實作、與文件保存作法,可以提高系統的可行性與可維護性。

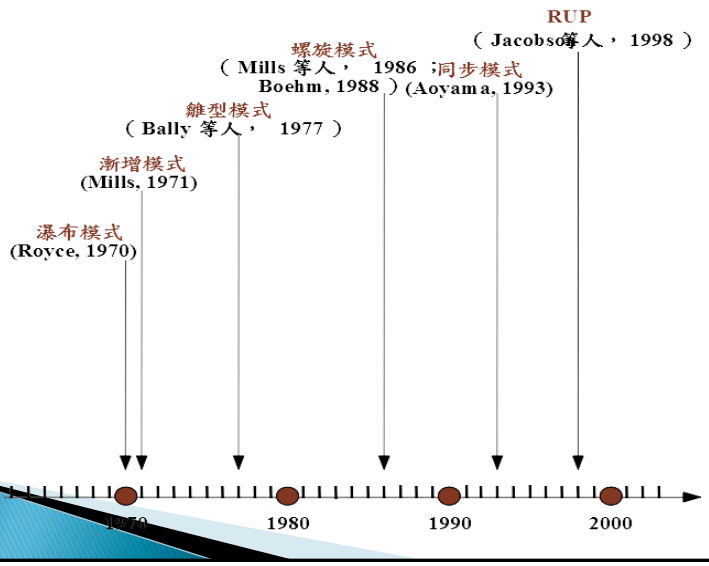
資訊系統開發模式之演進

- 今日發展環境已經有了很大的改變,不單是因為組織需要的不同,也因為電腦技術一日千里的變化。系統發展的基本指導原則還是在許多組織的資訊化過程依舊適用。大部分的企業組織,在發展新的系統專案時,都會利用一套系統發展方法論(systemdevelopmentmethodology)的標準步驟。
- 資訊系統通常是依據其生命週期來發展。因此,系統發展生命週期(systemdevelopmentlifecycle,SDLC)就是在許多組織在系統發展上最常見的方法論,特徵在於分成數個階段,標示不同的系統分析發展成效的進程。
- 資訊系統發展模式在過去五十多年中,學者所提出的資系統發展生命週期模型並不見得一致,從四個階段到二十個階段都有；我們希望藉由較適合的發展模式,提供全面考量分析與適當的設計技術,希望能在更短的時間與更少的資源投入的情況下,完成系統的建置。

資訊系統開發模式

- 資訊系統開發模式是資訊系統開發活動一系列的步驟及其執行程序 ，包含每個步驟之工作、產出及相關評估等
- 系統開發依循系統化、邏輯化的步驟進行時，有利於標準、規範與政策之推行和建立，開發的過程將更有效率、更能確保品質，也更容易管理 。
- 不同的資訊系統開發模式，適用於不同情況的系統開發

系統開發模式之演進



系統開發模式

- 常用的資訊系統開發模式如下：
  - 瀑布模式(Waterfall)
  - 雛型模式(Prototyping)
  - 漸增模式(Incremental)
  - 螺旋模式(Spiral)
  - 同步模式(Concurrent)
  - 統一軟體開發過程(Rational Unified Process)
  - 敏捷開發(Agile development)

瀑布模式

- 將系統開發的過程分成「幾」個階段，且每個階段清楚定義該做哪些工作及交付哪些文件，各階段循序執行且僅循環一次。
- 當問題較小或較單純，劃分的階段可能少至三個，例 如分析、設計、實施等階段；若面對較 大或較複雜之問題時，其階段可再被細分成更多個階段，例如可能擴充至十個階段。

瀑布模式

瀑布模式

- 制定計畫：目的是進行初步調查，評估ＩＴ相關的商業機會或問題。初步調查的一個關鍵部分是可行性研究，可行性研究審查預期成本和效益，在操作、技術、經濟和時程等因素的基礎上，提出建議的行動方案。
- 需求分析：著重需求定義，以符合業務內容及使用者需求為目的。使用各種技術，如調查、訪談、觀察、文件審查或抽樣等，最後結果建立需求模型，將需求文件化。
- 軟體設計：根據需求分析結果，進行包含系統任務目標、功能關聯、邊界範圍、各階層使用者的角色等內外部使用的規劃。

瀑布模式

- 程式設計：落實既有之規劃，符合委託者或使用者的需要，將操作介面、資料處理、功能運作等完整的實現
- 軟體測試：進行運作模擬，檢驗該系統的完成度，確保各項功能皆可符合既定的需求。進行系統部署，確保過程中系統運作無誤，並安排教育訓練，使人員能正確操作系統之功能。
- 運行維護：系統服務之運作維持和更新，確保穩定的服務品質。包括錯誤的修正及適應環境的變化及功能提升。

瀑布模式

- 瀑布模式 除了在階段劃分上較有彈性外，該模式至少 另提供二項主要的加強：
  - 若在各階段發現錯誤可允許階段間之回饋 ，使能盡 早修正以減少系統修改或重做之成本。
  - 各階段明確定義應做之工作及交付之文件 ，使系統 開發之工作更明確且更容易掌握。

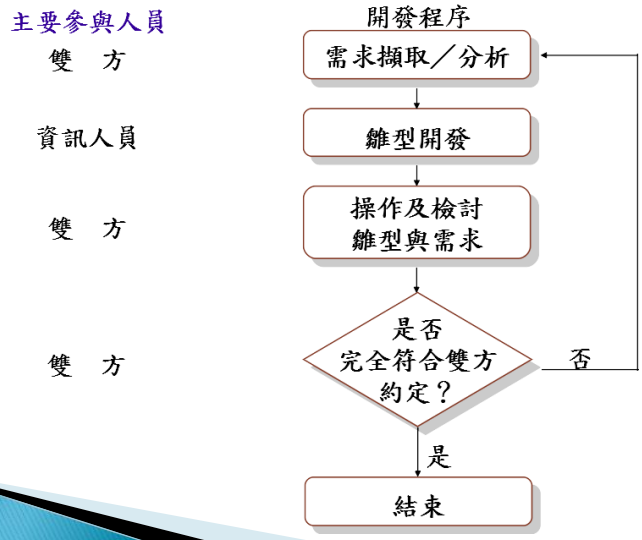
瀑布模式

- 瀑布模式的一些問題
  - 假設在專案開始時，需求可完全且清楚描述。
  - 所有需求在各階段均需同時考量，且系統開發在一 個週期內完成。
  - 在程式編輯前過於強調完整的分析與設計文件，故 一旦需求變更，文件便需大幅修改。
  - 系統開發週期較長且過程中使用者參與不足。
  - 程式編輯於系統開發週期之後段才開始，故風險較 高，且失敗之成本亦較高。

雛型模式

- 雛型模式是先針對使用者需求較清楚的部分或資訊人員較能掌握之部分，依分析、設計與實施等步驟快速進行雛型開發。開發過程中，強調盡早以雛型作為使用者與資訊人員需求溝通與學習之工具，雙方透過雛型之操作與回饋，以釐清、修改及擴充需求，並藉以修改與擴充雛型。上述步驟反覆進行，直到系統符合雙方約定為止。
- 雛型模式先經需求分析、設計、編碼、測試等階段快速建造一個可運作但並不完美雛型系統。經使用者操作與試驗過雛型系統後,開發人員得以發掘更完整的需求系統而能漸趨完整與成熟。

雛型模式



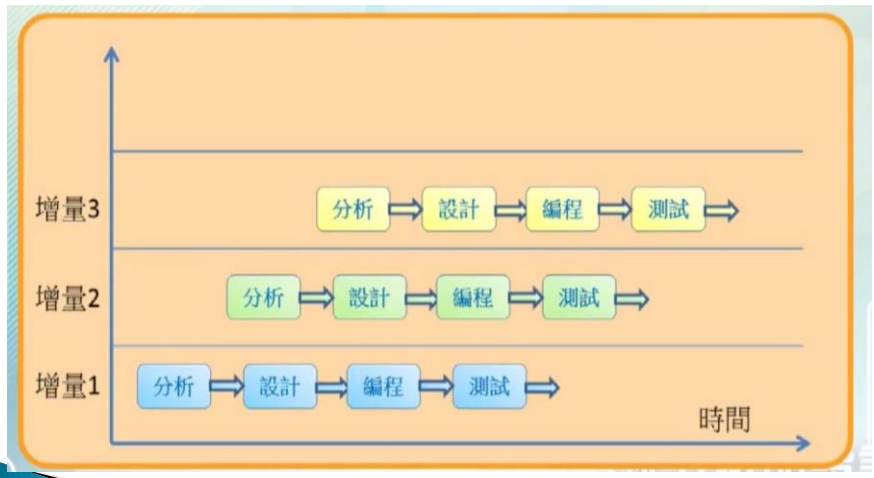
雛型模式

- 雛型模式之主要特性與原則
  - 強調雛型之盡早開發及使用者高度的參與。
  - 強調以雛型作為使用者及系統開發者之需求溝通與 學習機制。
  - 從需求最清楚的部分著手開發雛型，並透過使用者 對雛型之操作與回饋，反覆修改與擴充。每次反覆之週期要盡可能縮短。
- 雛型模式的潛在問題
  - 系統文件較不完備，程式亦較難維護。短期可能較 能滿足使用者需求，但長期而言，系統較易失敗。
  - 因缺乏整體之規劃、分析與設計，故較不適用於大型及多人參與之系統開發專案。

漸增模式

- 漸增模式是把需求分成「幾」個部分，然後依漸增開發計畫將每個「部分需求」之開發訂為一個開發週期，每個週期可依序或平行開發。每個週期之階段清楚定義要做哪些工作及交付哪些文件，每個階段循序進行且僅循環一次。
- 漸增模式是瀑布模式的一種再進化,希望讓瀑布模式更趨於完善；這個模式改善瀑布模式各階段同時考量的需求,在發展漸增模式中,再加上一些新的元素,新功能,把問題再分成各各的小問題,解決小問題後再整合一起開發。

漸增模式



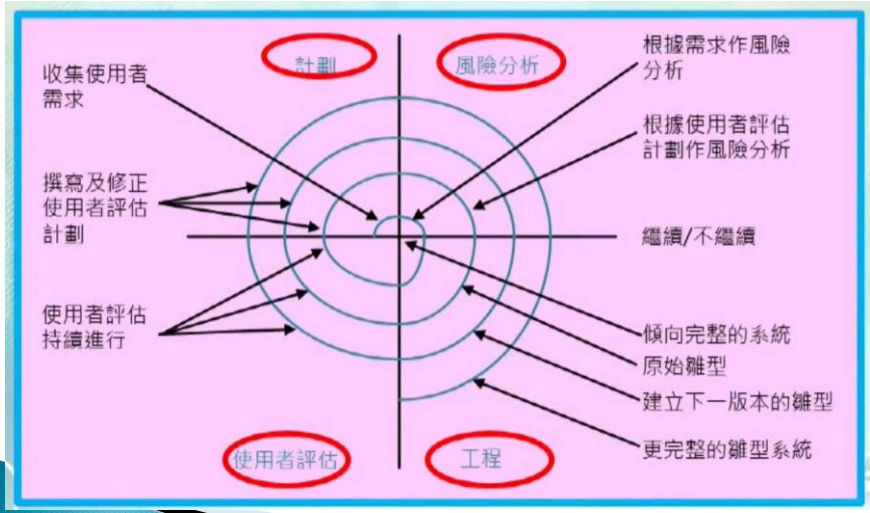
漸增模式

- 漸增模式與瀑布模式大部分相同，但是仍有一些地方 不同，例如：
  - 系統被分成幾個子系統或功能，各子系統可獨立依 序開發；而瀑布模式則是各個子系統須同時開發。
  - 系統開發可由多個週期完成，每個週期表示不同版 本之系統，因此在每個週期均有程式編輯及上線實 施，使用者每個週期均參與，故相較於瀑布模式， 漸增模式之風險較低。
- 漸增模式適用之情況
  - 組織的目標與需求可完全且清楚描述。
  - 預算須分期編列。
  - 組織需要時間來熟悉與接受新科技。

螺旋模式

- 螺旋模式是基於下列的構想而產生：在生命週期的每一階段,應該運用模擬、雛型、模式建立等方法來降低風險。
- 每一階段都強調各開發週期的規劃與風險評估是否會影響到開發的實用性。
- 所以當一部程式系統在開發時,先確定目標後再進行討論,規劃內容後再評估風險,當遭遇風險的考量時,能運用各種的方法來解決問題。

螺旋模式



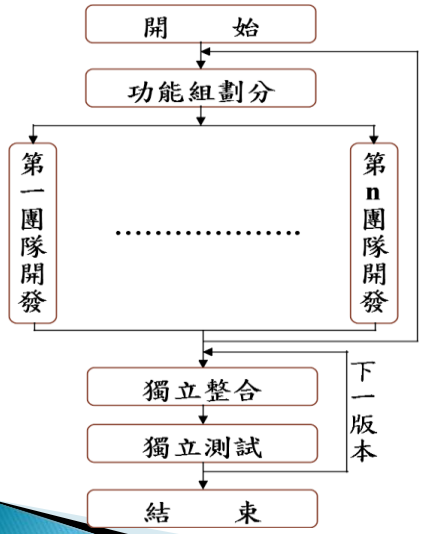
螺旋模式

- 螺旋模式之特色與應用原則
  - 在高風險部分之設計尚未穩定前，規格之發展不需要一致、詳盡或正式，以避免不必要之設計修改。
  - 在開發之任一階段，螺旋模式可選擇整合雛型模式以降低風險。
  - 當更吸引人之方案被找出或新風險需被解決時，螺旋模式整合重做或回到前面之階段。

同步模式

- 同步模式是基於下列構想以縮短開發時間、提高市場競爭力：
  - 多個團隊同時開發,稱為活動同步。同步行為可以是一個階段內的工作、多個階段的工作、或多個專案的工作或軟硬體的工作平行進行。
  - 在處理上要求需求明確與完整的描述,並且有足夠的人力參與。在人力協調上,要求團隊間有良好的溝通。它將開發工作分割並同時進行,整合系統測試完整獨立進行,且各功能系統組都要能確實執行。
  - 由於在處理上是處於同步的型態,在時間的掌控上尤為重視。相較於前面所列出的開發模式,此模式比較能夠因應現代的需要。

同步模式

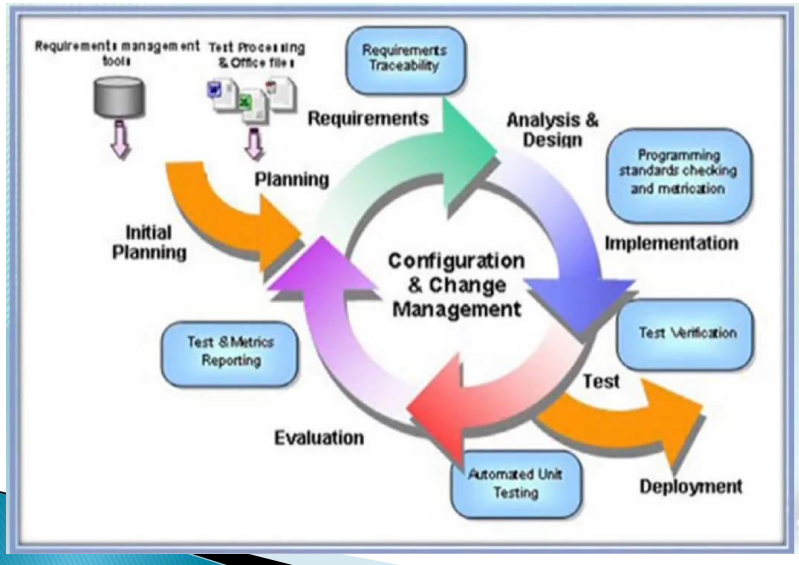


- 同步模式的發展主要是為了因應商業套裝軟體的市場競爭，其優點是開發時間的縮短可提高產品的競爭力，其缺點則是緊湊的步驟及頻繁的資訊溝通，使得專案管理的複雜度大幅提高，人力成本也相對提高，若沒有輔以良好的工具及管理方法，則不易達成目標。

統一軟體開發過程

- RUP模式(Rational Unified Model, Rational統一流程)於1998年由Jacobson 等人提出。
- 該模式結合螺旋模式的概念，以反覆與漸增的軟體發展原理進行軟體開發，且每一次的反覆需產出一個可運作的系統版本，並在每一個反覆週期評估風險，以盡早發現問題。

統一軟體開發過程(RUP)



統一軟體開發過程(RUP)

1. 商業塑模(Business Modeling) :

商業建模工作流描述了如何為新的目標組織開發一個構想，並基於這個構想在商業範例模型和商業對象模型中，定義組織的過程，角色和責任。

2. 需求(Requirements) :

目標是描述系統應該做什麼，並使開發人員和使用者就這一描述達成共識。為了達到該目標，要對需要的功能和約束進行提取、組織、文檔化；最重要的是理解系統所解決問題的定義和範圍。

3. 分析和設計(Analysis & Design) :

將需求轉化成未來系統的設計。分析設計的結果是一個設計模型和一個可選的分析模型。設計模型是源代碼的抽像，由設計圖和一些描述組成。

設計活動以結構設計為中心，由數個結構視圖來表達，結構視圖是整個設計的抽像和簡化，該視圖中省略了一些細節，使重要的特點更加清晰。

統一軟體開發過程(RUP)

4. 實現(Implementation) :

目的包括以層次化的子系統形式定義代碼的組織結構；以組件的形式(原始文件、二進制文件、可執行文件)實現整合；將開發出的組件進行測試以及整合由單個開發者（或小組）所產生的結果，使其成為可執行的系統。

5. 測試(Test) :

要驗證對象間的交互作用，驗證軟體中所有組件的正確集合，檢驗所有的需求已被正確的實現, 識別並確認缺陷在軟體部署之前被提出並處理。RUP提出了疊代的方法，意味著在整個項目中進行測試，從而盡可能早先發現缺陷，從根本上降低了修改缺陷的成本。測試類似於三維模型，分別從可靠性、功能性和系統性能來進行。

6. 部署(Deployment) :

部署工作流的目的是成功的生成版本並將軟體分發給最終用戶。部署工作流描述了那些與確保軟體產品對最終用戶具有可用性相關的活動，包括：軟體打包、生成軟體本身以外的產品、安裝軟體、為用戶提供幫助。在有些情況下，還可能包括計劃和進行beta測試版、移植現有的軟體和數據以及正式驗收。

統一軟體開發過程(RUP)

7. 配置和變更管理(Configuration & Change Management) ：
描繪了如何在多個成員組成的項目中控制大量的產物。配置和變更管理工作流程提供了準則，來管理演化系統中的多個變體，追蹤軟體創建過程中的版本。工作流程描述了如何管理並行開發、分散式開發、如何自動化創建工程。同時也闡述了對產品修改原因、時間、人員保持審查記錄。

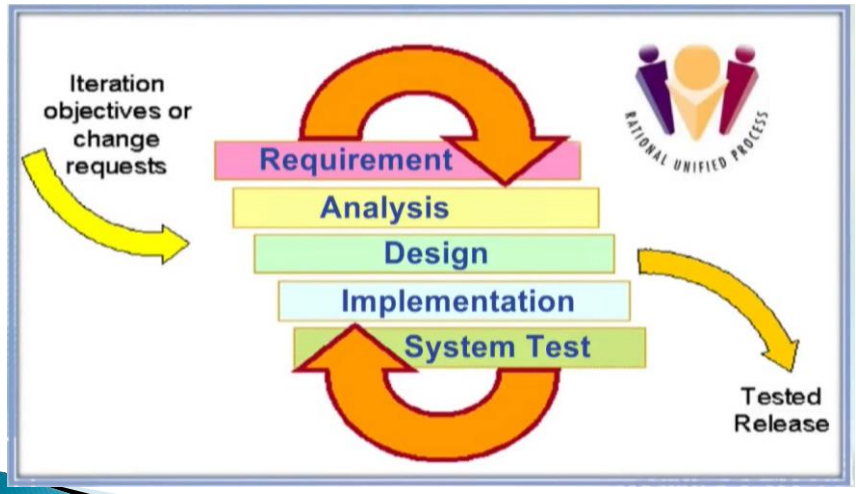
8. 專案管理(Project Management) ：
管理平衡各種可能產生衝突的目標，管理風險，克服各種約束並成功交付使用戶滿意的產品。其目標包括：為項目的管理提供框架，為計劃、人員配備、執行和監控項目提供實用的準則，為管理風險提供框架等。

9. 環境(Environment) :
目的是向軟件開發組織提供軟體開發環境，包括過程和工具。環境工作流程集中於配置項目過程中所需要的活動，同樣也支持開發項目規範的活動，提供了逐步的指導手冊並介紹了如何在組織中實現過程。

統一軟體開發過程(RUP)

- 該程序方法論結合了眾多成功的方法論，提供完整的軟體開發流程。
- 結合採用疊代式（Iterative）開發流程，可降低開發的風險。
- UML作為其視覺化軟體分析與設計語言，使軟體開發人員之間的溝通與資訊的交換無礙。
- 以UML中的使用者案例（Use-Case）作為整個Rational Unified Process的核心。
- 受到工具的支援：程序經過良好的支援，且受到好的工具支援。例如， Rational Rose、Rational Unified Process。
- RUP打破了“需求-設計-編碼-測試”這樣傳統的瀑布模組模式，而是讓這些工作一直進行，只是在不同的時間裡有不同的比例。這個思想與敏捷開發非常相近，也非常符合實際的狀況。

疊代式開發流程



敏捷開發

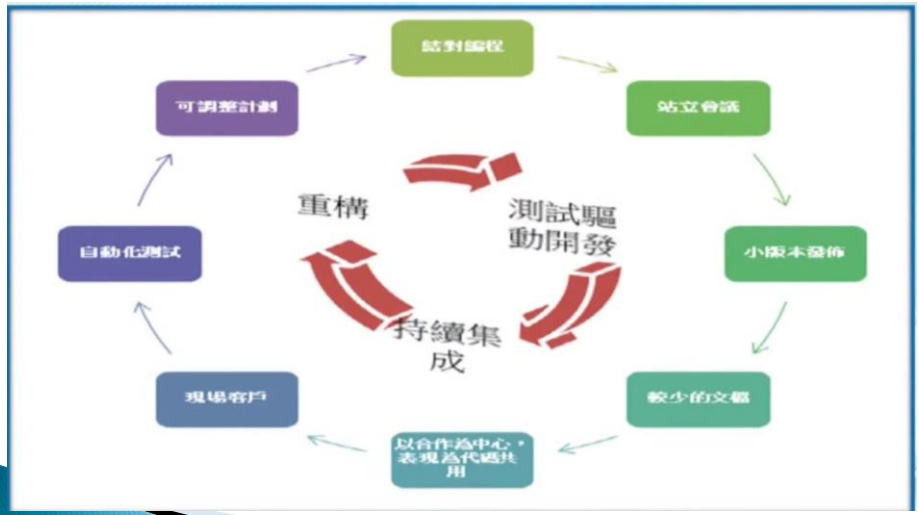
- 敏捷開發是一種從1990年代開始逐漸已起廣泛關注的一種新型軟體開發方法
- 是一種對應快速變化需求的一種軟體開發模式。
- 強調設計師團隊與業務專家之間的緊密協作、面對面溝通、頻繁交付新的軟體版本、緊湊而自我組織型的團隊、能夠很好地適應需求變化的程式碼編寫和團隊組織方法，也更注重軟體開發過程中人的作用
- 是一種以人為核心、疊代、循序漸進的開發方法

敏捷開發

- 軟體開發的構件被切分成多個子項目，各個子項目的成果都經過測試，具備整合及可執行的特徵
- 換言之，就是把一個大項目分為多個相互聯繫，但也可獨立執行的 小任務，並分別完成，在此過程中軟體一直處於可使用狀態。
- 敏捷開發為在短階段工作內快速且輕鬆完成任務分工的能力
- 敏捷開發與APP 開發有某些特性相似：

1．設計應簡單。
2．敏捷開發的目的是比任何其它方法更早地提供成品。
3．嚴格測試及團隊合作是必需的。
4．隨時準備作出修正。
5．歡迎改變並使用它作為一個優勢。

敏捷開發



敏捷開發的特點

- 敏捷開發方法是適應性而非預設性
  - 軟體設計難處在於軟體需求的不穩定，從而導致軟體過程的不可預測。敏捷方法歡迎隨時變化，目的就是成為適應變化的過程，甚至允許改變自身來適應變化，所以稱為適應性方法。
- 敏捷開發方法是以人為本而非過程導向
  - 敏捷開發試圖使軟體開發工作能夠利用人的特點，充分發揮人的創造能力。
  - 敏捷開發的目的是建立起一個任務團隊全員參與到軟體開發中
  - 敏捷開發特別重視訊息交流，
  - 敏捷過程是承認每個人都有特定的能力(以及缺點）並加以利用，而不是把所有的人當成一樣來看待。
  - 更重要的是 ，在這樣的理念下，幾個任務作下來，每個人的能力都能從中得以提高
- 敏捷開發主張適度的計畫、進而開發、提前交付予持續改進，並鼓勵快速與靈活的面對開發與變更

不同方法之比較

模型名稱	技術特點	適用範圍
瀑布模型	容易簡單，分階段，階段間存在因果關係，各個階段完成後都有評審，允許反饋，一般不允許用戶參與，要求預先確定需求	需要比較有經驗的團隊應用，需求易於完善定義且不易變更的軟體系統
雛型模型	不要求需求預先完備定義，允許用戶參與，允許需求的漸進式完善和確認，能夠適應用戶需求的變化	需求複雜、難以確定、動態變化的軟體系統
同步模型	多團隊同時進行。縮短產品開發時程，藉以提高市場競爭力	管理比一般的開發模式更為複雜，必須開發一個管理系統幫助協調。人力成本也相對提高。

不同方法之比較

模型名稱	技術特點	適用範圍
增量模型	軟體產品是增量式一塊塊開發，允許開發活動並行和重疊	技術風險較大、用戶需求較為穩定的軟體系統
螺旋模型	結合瀑布模型、快速原型模型和迭代模型的思想，並引進了風險分析活動	需求難以獲取和確定、軟體開發風險較大的軟體系統
RUP	可改造、擴展和剪裁：可以對它進行設計、開發、維護和發佈；強調迭代開發	複雜和需求難以獲取和確定的軟體系統
敏捷開發	強調適應性而非預見性，注互動溝通，注重市場快速反應能力，也即具體應對能力，客戶前期滿意度高，減少中介過程的無謂資源消耗。	項目的規模。規模增長，面對面的溝通就愈加困難，錯誤會迅速膨脹，注重人員的溝通，較忽略文件的重要性。