國立屏東大學 資訊工程學系 物件導向軟體工程 (last updated September 22, 2017)

1. 軟體工程概述

本章將就軟體工程(Software Engineering，SE)加以簡介，其相關主題包含軟體與軟體工程、軟體專案、軟體品質、軟體塑模以及各種常見的軟體生命週期模型。身為一個開發軟體的軟體工程師，除了應熟悉軟體工程的各項程序與方法外，本章末也列舉身為一個好的軟體工程師所應遵從的職業道德規範。

τ_i $a+b$ $\frac{i^2}{x}$

1.1 軟體(Software)

1.1.1 何謂軟體?

何謂軟體(Software)? 相較於有形的「硬體(Hardware)」，軟體(Software)」則是無形的。舉例來說，XBox游戲機以及其相關的感測裝置、操作搖桿等就是硬體裝置；而「NBA 2K18」、「Age of Empires」等游戲就是軟體。同樣的，音響設備是硬體，音樂作品則是軟體。硬體與軟體必須搭配使用，才能發揮功能或提供使用者相關的服務。例如單有遊戲主機，而無遊戲光碟；或是單有音響主機，卻沒有音樂光碟。反之，單有遊戲或音樂光碟，但卻沒有對應的執行或播放硬體，也都是沒有意義的。

<note> 用來存放軟體或是音樂作品的光碟則被稱為載具(carrier)。 </note>

本書著重電腦軟體(computer software)，意即必須在電腦系統上才能執行並提供服務的軟體。電腦軟體可以被視為是一些電腦程式(Computer Program)的集合，當其被加以執行時可以提供特定的功能或服務。有時候，在電腦程式中用以儲存資料的資料結構、處理資料的演算法以及程式的說明文件，也都會被視為是軟體的一部份。

<note> 一個電腦程式又可以被視為是一些程式碼(code)的集合，用以完成特定的功能。 </note>

綜合上述，我們可以將「軟體」定義為：為了在電腦系統上執行以達成特定目的一組電腦程式、數位資料及其文件的組合。

1.1.2 軟體的特性

基本上，軟體是無形的(intangible)，它沒有一般商品存在的耗損(wear out)問題，不過它可能存在效能退化(deteriorate)的問題（你有沒有遇過隨著時間的推移，軟體執行效率退化的情況？）。另一方面，軟體是極為容易再製的(reproduce)，不管製做多少份複本，其成本不會隨之增加（只要一個複製的指令即可完成)。一般而言，軟體的成本集中在其開發階段，一但開發完成，其再製成本極低，不過後續的軟體維護也必須投入相當的人力與物力。整體而言，軟體產業可算是一個勞力密集(labor-intensive)的產業，不像其它的製造業，其製作或開發的過程很難加以自動化。

1.1.3 軟體的分類

依造軟體的用途與使用者的不同，軟體可以分為幾種不同的類別，例如「客製型軟體(Custom Software)」、

是針對特定使用者、特定的用途所加以製作，通常此類軟體只有有少數的複本存在。另一方面，非客製型的軟體則被稱為「通用型軟體(Generic Software)」，它是由軟體廠商針對特定市場所開發的軟體產品(Product)，通常在市場上公開向所有使用者販售，有時又稱為「Commercial Off-the-Shelf」COTS」(開架式產品)。這兩種軟體的差別在於「Custom Software是針對明確已知的使用者及其需求客製化設計開發」Generic Software則是由軟體開發者預想使用者可能的需求而加以設計開發；就如同量身定製的西裝和大量生產的成衣西裝的差別。

1.2 軟體工程(Software Engineering)

軟體工程(Software Engineering)一詞，首見於1968年的NATO的Software Engineering Conference.[1)2)]，用以面對當時已注意到的軟體危機。在當時，軟體產業以客制型的軟體專案開發為主，但許多大型的軟體專案，往往會面臨開發成本不對追加且難以再利用(reuse)的困境。不像其它的製造業或營建業，已存在許多自動化或是專案管理的方法，軟體產業在當時還沒有任何方法可以解決這些困境。

所謂的軟體工程就是關於軟體從無到有的過程。希望透過一些工程科學的方法，讓軟體的開發能夠如同其它商品的開發一樣，具有有效的方法來確保軟體開發的效率、品質以及成本的控管。換具話說，軟體工程被視為是一種專業技能，可以應用在軟體的設計、開發、實作以及修改、維護等過程，並希望能達成軟體開發的品值保證、降低成本、易於維護，同時也希望能透過軟體工程的方法來縮短開發的時程[3)]。同時，軟體工程也被期望能向許多其它領域一樣，能夠提供包含分析、設計、評量、實作、測試、維護與精進的工程方法[4)]。這些所謂的工程方法必須是易於理解、管理與應用的方法，同時也被期待是易於再製且能夠具有標準化程序的方法。用更簡單的話來說，軟體工程就是採用系統化、規章化、可量測的方法，來進行軟體的開發與維護[5)]。

用更為簡單的方法來說，軟體工程就是應用工程方法在軟體開發上。通常軟體的開發過程，就是不斷提升抽象化層次的過程，從最高抽象層次的系統分析、設計，到逐漸具體化的程式碼撰寫與測試等過程。我們必須建構出完善的工程原理，來幫助軟體的開發獲致具有經濟效益、且能夠在真實機器上提供穩定與效率的高品質軟體[6)]。為了回應這些需求，目前的主流做法之一就是將軟體工程視為是建立與使用模型的過程[7)]，透過一些容易理解的模型，針對不同抽象層次建立並使用這些模型，可在真正開發軟體之前得到可以呈現最終軟體產品的需求與使用者界面模型，有助於軟體開發團隊與使用者進行相關的討論：更具體的設計模型，則可以用來描述實際需要開發的各項軟體組件之功能與設計準則；建置模型則可以應用在軟體的配置與維護等方面的工作。本書後續也將就這些不同的模型加以討論，其中將以UML模型為主要的探討對象。

另一方面，現在的軟體開發愈趨大型化、複雜化，往往不再是一個人或少數人可以獨立完成的，所以軟體工程也與軟體團隊的管理緊密相關。如何在團隊間適切地分工、溝通合作，也成為軟體工程相關領域的重點之一。主要的困難在於以下數點：

- 如何將整個軟體切割為適切的組件，讓團隊成員得以分工合作，同時各自完成的軟體組件也必須確保能夠正確地組合運作。
- 最終完成的軟體必須具有一定的品質。
- 軟體的開發必須滿足經費、時間等成本限制。
- 軟體開發的過程中，其可運用的各項資源往往都是有限的。
- 軟體開發的過程中，還必須確保所投入的各項成本都能夠帶來適切的助益。
- 有時軟體的開發還必須與其它團隊在成本與時程上競爭。
- 不正確的成本控管往往會造成軟體專案的失敗，但成本的控管有許多不可預知的因素。

1.2.1 軟體工程的目標

軟體工程的主要目標可列示如下：

- 使用適當的方法來描繪使用者的需求。
- 確保軟體的品質。
- 用以開發小型或大型、簡單或複雜的軟體產品。

- 縮短開發時程、降低開發成本。
- 進行軟體開發的風險管理。
- 用以在不同的方案中，尋求適合的解決方案。
- 支援軟體的精進。

1.2.2 軟體工程的範疇

以最簡單的方式來說，軟體工程就是從無到有將軟體開發出來的過程，其相關的範疇如下：

- 分析(Analysis)
- 規格化(Specification)
- 設計(Design)
- 開發(Development)/實作(Implementation)
- 測試(Testing)
- 建置(deployment)
- 維護(Maintenance)

1.2.3 Object-Oriented Software Engineering

以物件導向技術進行軟體工程各項活動，包含

- 物件導向分析(Object-oriented analysis, OOA)
- 物件導向設計(Object-oriented design, OOD)
- 物件導向程式設計(Object-oriented programming, OOP)
  - 使用物件導向程式語言(Object-oriented programming language, OOPL)進行程式之設計
  - 例如使用Java, C++, C#, Object C/C++等程式語言

1.3 軟體專案(Software Projects)

專案[8)]係指一項暫時性的工作，其目的在於創造出特定的產品或服務。專案通常具有以下的特性[9)]

- 有預定的目標與時程
- 具有獨特性
- 是暫時性的工作
- 是資源的整合
- 成敗受外在因素影響甚大
- 變動性大

所謂的軟體專案(Software Project)即為以開發軟體為主要目的的專案。就軟體產業而言，不論是客製化的軟體、亦或是通用型軟體，其開發通常都被視為是一項軟體專案。每一項軟體專案都必須針對有限的資源(包含人力、設備、時程等)進行的管理，相關工作稱為軟體專案管理(Software Project Management)。當然也不是每一項軟體專案，都是以開發新軟體為目的，有些軟體專案是針對既有的軟體進行改善或維護等工作。例如有的軟體專案是為了找出既有軟體的缺失並加以修正，又或者有的軟體專案是為了將既有的系統進行修改，使其能適用於更新版本的作業系統、資料庫系統、或是新的商業模式或法規。有些軟體專案則是為了幫既有的軟體增加或修改功能。有時候是由開發人員針對既有的功能進行新的設計，以提升軟體的效能、或透過模組的重新設計提升軟體的可維護性。

軟體專案涉及許多相關人員(Stakeholders)，包含：

- 使用者(Users):軟體的使用者。
- 客戶(Customers):支付或購買軟體的人。
- 軟體開發的相關人員:
  - 系統分析師
  - 軟體工程師
  - 程式設計師
  - 測試工程師
  - 維護工程師
- 專案經理(Project Manager，PM)，負責軟體專案成敗的決策者、領導者。
- 軟體的銷售與發行人員

1.4 軟體品質(Software Quality)

如何評定軟體的品質是一件相當重要的課題，首先必須就軟體的評量指標加以討論，常見的軟體品質評量指標如下：

- 可用性(Usability):軟體能夠提供使用者所需的服務，或是讓使用者容易地完成工作。
- 效率性(Efficiency):軟體的執行不會浪費如CPU或網路頻寬等資源。
- 穩定性(Reliability):軟體可以正確地執行，而不會發生錯誤。
- 可維護性(Maintainability):軟體功能可以容易地被修改。
- 可重用性(Reusability):既有的軟體功能可以套用在其它軟體專案的開發上。

Fig 1與Fig 2.提供了一些可能的軟體評量指標。

Fig. 1: An example ontology of quality

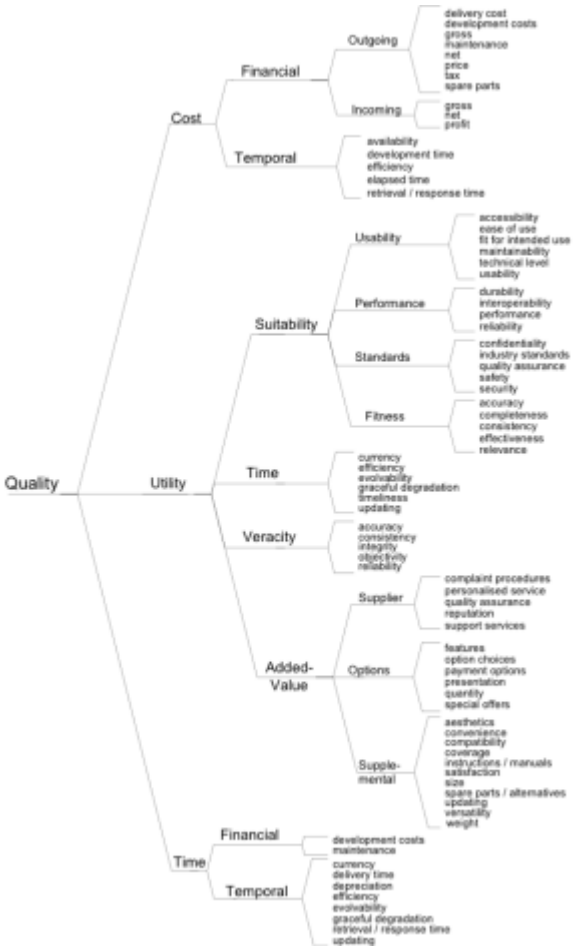
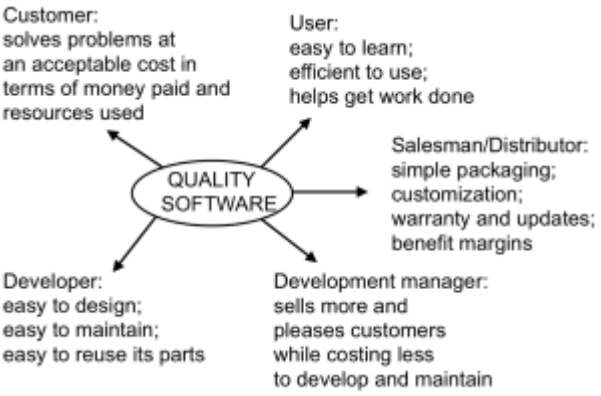


Fig. 2: Software quality at stakeholders

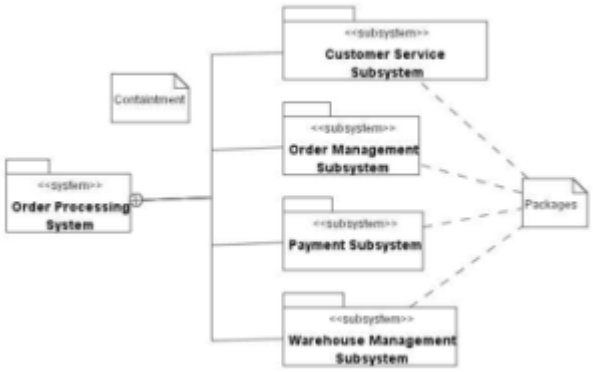


在相關的評量指標中，有一些指標彼此間是互斥的，例如透過程式碼的精簡可以提升效能，但卻也會降低可維護性與可重用性；反之，若提升了可重用性，則可能會造成效能的下降。所以我們必須在眾多的效能指標中，挑選適合的指標項目來進行評比。

1.5 軟體塑模(Software Modeling)

- To develop software is to build a MACHINE, simply by describing.[10)]
- A software engineer needs only make a description of the required machine. Describing the required machine is usually not a simple task.
- A system is “a set or arrangement of elements that are organized to accomplish some predefined goal by processing information.”[11)]
    - 舉例來說：學生選課系統、銷售系統或是一輛車等等
- 一個系統(System)可分解成數個子系統(Subsystem)的集合

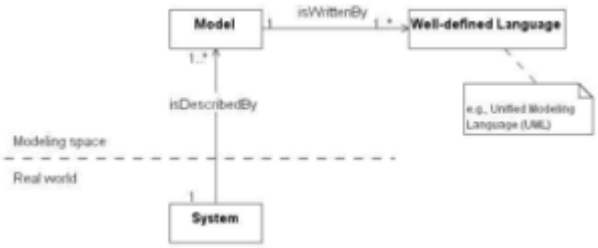
Fig. 3: 系統與子系統



- A model is a simplified representation of (part of) a system. It is written in a language with well-defined syntax and semantics, and represents certain specific aspects of a system.
- A model is a simplification of reality.
- We build models so that we can better understand the system we are developing.
- We build models of complex systems because we cannot comprehend such system in its entirely.
- The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.
- Every model may be expressed at difference levels of precision.
- The best models are connected to reality.
- No single model is sufficient. Every nontrivial system is best approached through a small set of nearly independent models.

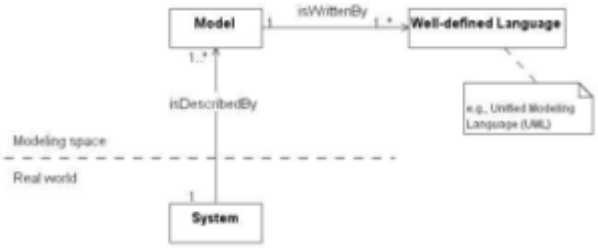
- Models are not right or wrong; they are more or less useful. [12)]

Fig. 4: 系統與模型



- 一個特定的系統可以被各式各樣的模型的從不同的觀點呈現，例如：靜態和動態的模型等等

Fig. 5: 系統可以擁有不同的模型



- Modeling(塑模)就是軟體工程的活動
  - 塑模允許我們去使用某些比現實簡單、可靠或便宜的東西。
  - 塑模允許我們以簡單的方式處理，來避免複雜性和危險。
- Modeling的構思[13)]
  - 物件導向設計就是關於塑模
  - 模型中的基本元件應該描述“things” 而非”action”或 “ processing”
  - Modeling allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality.[14)]
  - So, modeling is a means for dealing with the complexity. Complex systems are generally described by more than one model from the various kinds of views. Modeling then means constructing an abstraction of a system that focuses on interesting aspects and ignore irrelevant details.

1.6 軟體生命週期模型(Software Lifecycle Model)

- 軟體生命週期(lifecycle)
  - 軟體經歷的一連串過程，從問題的定義、系統開發到系統操作與維護。
  - 如果能使用適當的模型來表達軟體生命週期中的過程，那麼就可以將軟體的構思、設計、開發、測試與維護等工作加以抽象化、簡單化。
- 軟體生命週期模型(Software Lifecycle Model)
  - 定義軟體流程不同階段的步驟, 像是需求收集、分析、設計、執行及測試、操作與維護
  - 嚴謹法(rigorous method) – 發展生命週期的共通概念

Fig. 6: Rigorous Method(嚴謹法)

- The 7 goals of software lifecycle models [15)]
  - Effectiveness – the model must help us produce the right product;
  - Maintainability – we can quickly and easily find and remedy faults or work out where to make changes;
  - Predictability – a good model will help us to plan and estimate the work;
  - Repeatability – if a model is discovered to work, it should be replicated in future project;
  - Quality – “The totality of features and characteristics of a product or service that bear on its ability to satisfy a given need.”
  - Improvement – the models will always be running to catch up the quickly changing development environments and requested products;
  - Tracking – a defined model should allow the management, developers and customer to follow the status of a project.

1.6.1 Waterfall Model

- Winston W. Royce於1970年首先提出瀑布模型[18]
- The model suggests that software engineers should work in a series of stages.
- Before completing each stage, they should perform quality assurance (verification and validation).
- 瀑布模型的發展階段:
  - 需求收集階段
  - 分析階段
  - 設計階段
  - 實作階段
  - 測試階段
  - 操作及維護階段

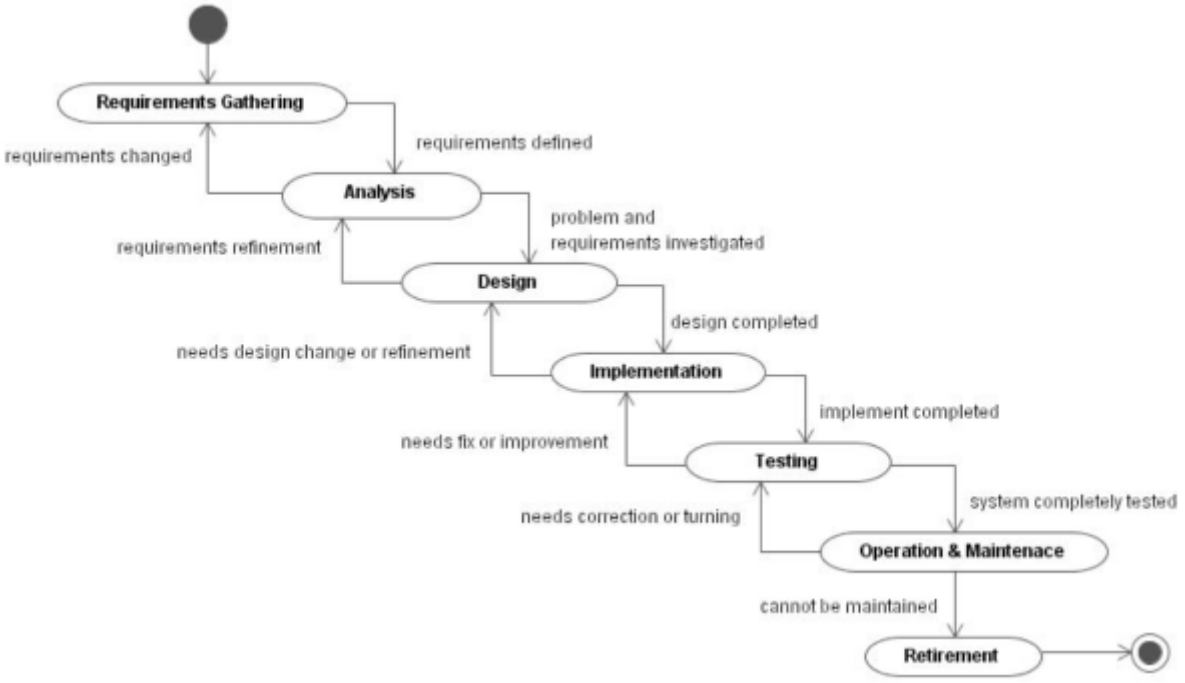


Fig. 7: Waterfall Model

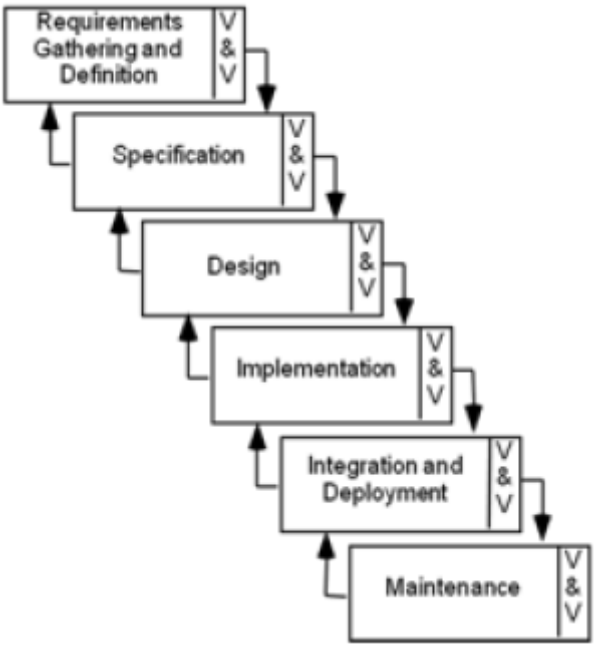


Fig. 8: Waterfall Model

- Waterfall Model 分析
  - 優點
    - 嚴謹法
    - 於每個階段提供文件及結果確認
    - 階段性審核(Milestone approval)
  - 缺點
    - The model implies that you should attempt to complete a given stage before moving on to the next stage
    - Does not account for the fact that requirements constantly change.
    - It also means that customers cannot use anything until the entire system is complete.
    - The model makes no allowances for prototyping.
    - It implies that you can get the requirements right by simply writing them down and reviewing them.
    - The model implies that once the product is finished, everything else is maintenance.

1.6.2 Phased-Release Model

- It introduces the notion of incremental development.
- After requirements gathering and planning, the project should be broken into separate subprojects, or phases.
- Each phase can be released to customers when ready.
- Parts of the system will be available earlier than when using a strict waterfall approach.
- However, it continues to suggest that all requirements be finalized at the start of development.

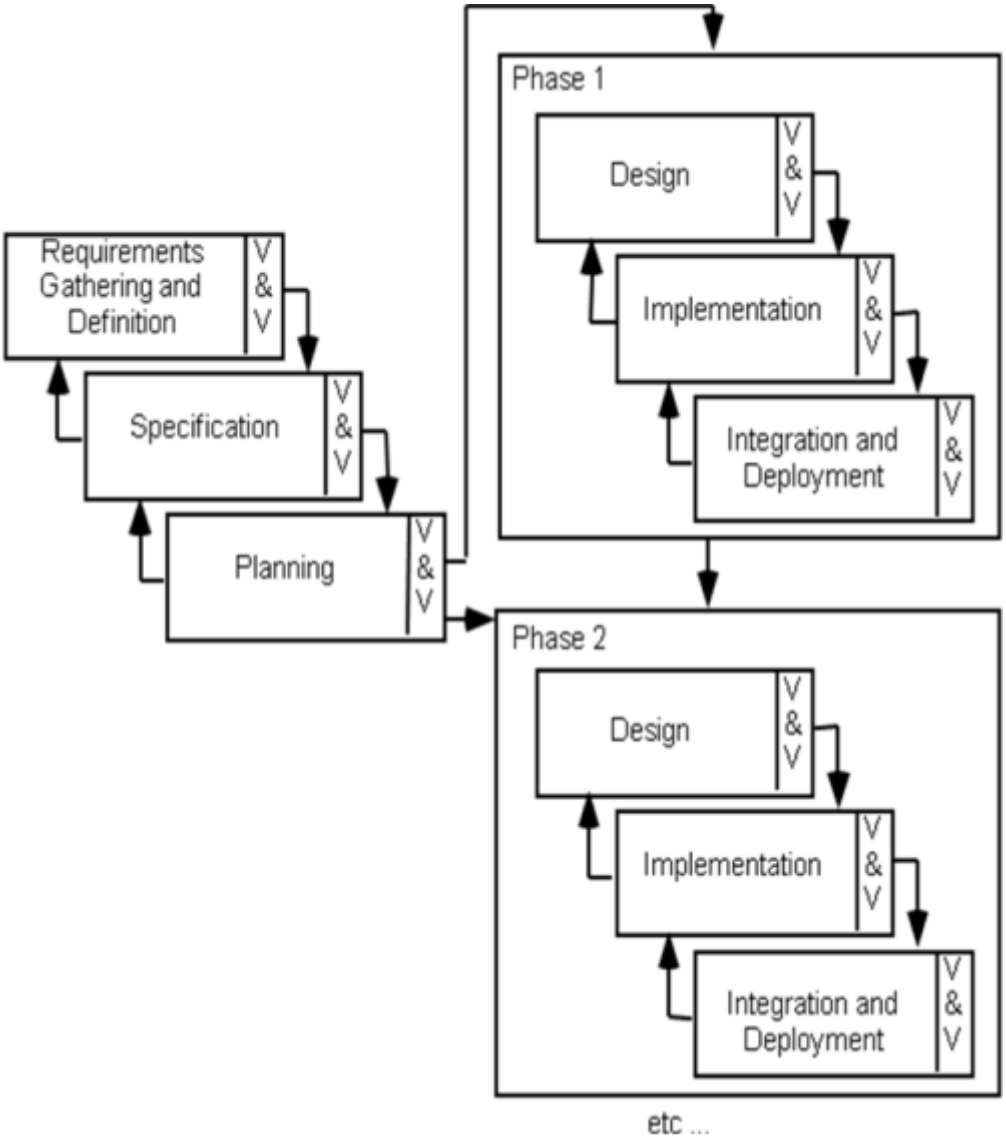


Fig. 9: Pahsed-Release Method

1.6.3 Rapid Prototyping Model

- Software Prototype.
  - 較初步的軟體系統版本(系統雛型)
  - 協助用戶了解軟體該做些什麼以及如何做
- Rapid Prototype.
  - 部份完成的目標應用程式（如GUI元件），可透過客戶與開發者確認需求。
  - 有效且實際的方式以得知客戶的需求
- Rapid Prototyping Model 優勢
  - 確認後將不拾棄.
  - 較少的回饋(Feedback)需求
  - 快速建立及快速修正來反映客戶需求

1.6.4 Spiral Model

- 由 Barry Boehm 於 1986 年所提出[16)]，為waterfall model 替代方法

- 降低風險的好方法
- It explicitly embraces prototyping and an iterative approach to software development.
- Start by developing a small prototype.
- Followed by a mini-waterfall process, primarily to gather requirements.
- Then, the first prototype is reviewed.
- In subsequent loops, the project team performs further requirements, design, implementation and review.
- The first thing to do before embarking on each new loop is risk analysis.
- Maintenance is simply a type of on-going development.
- Spiral Model 分析
  - 優點
    - 支援現存軟體的再應用
    - 不需做太多的測試
  - 缺點
    - 僅適用於較大規模的軟體開發

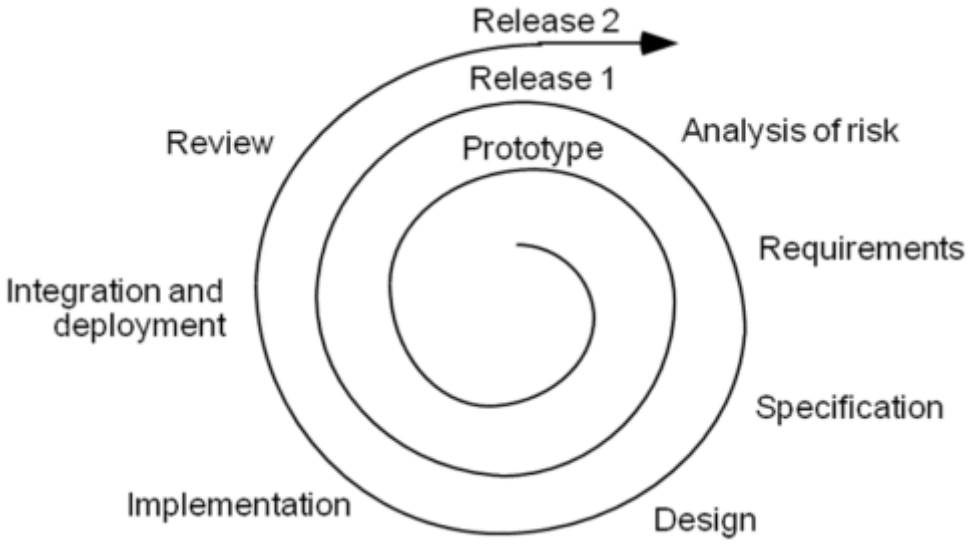


Fig. 10: Spiral Model

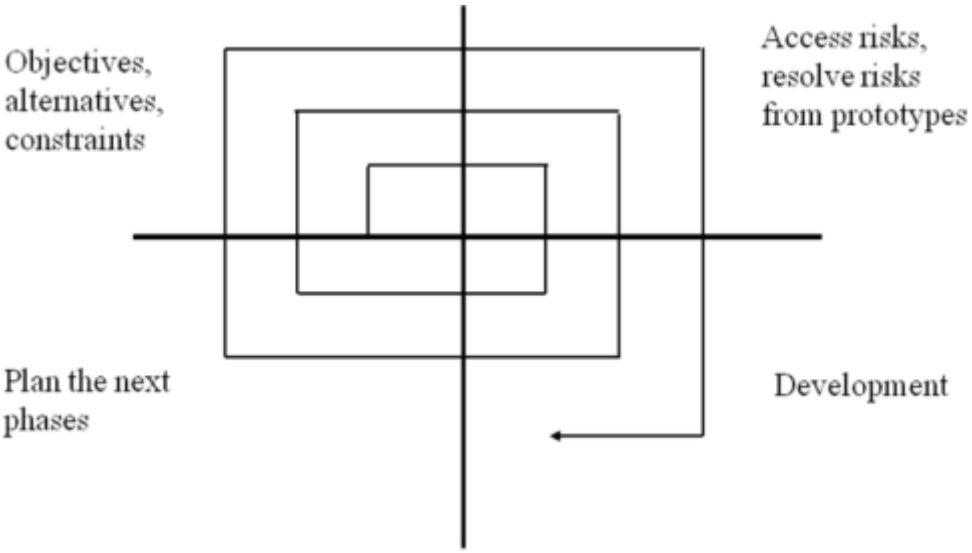


Fig. 11: Spiral Model (Another Version)

1.6.5 Concurrent Engineering Model

- It explicitly accounts for the divide and conquer principle.

- Each team works in parallel on its own component, typically following a spiral or evolutionary approach.
- There has to be some initial planning, and periodic integration.

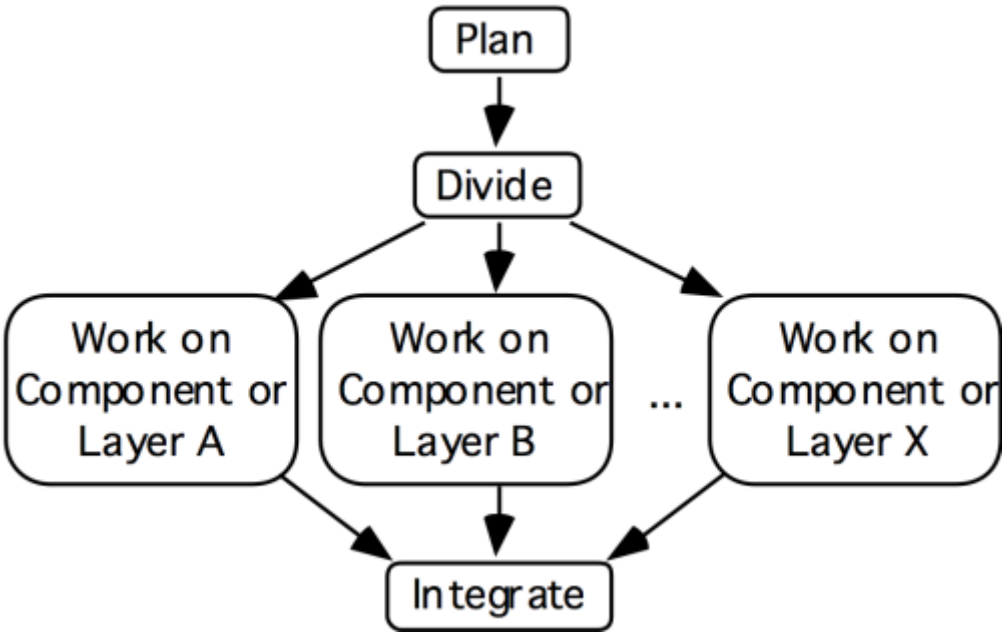


Fig. 12: Concurrent Engineering Model

1.6.6 Unified Process - An Object-Oriented Lifecycle Model

- Unified Software Development Process常簡稱為Unified Process(UP)是一種軟體開發程序架構(Software Development Process Framework) [17)18)]
  - An iterative and incremental software development process framework.
  - UP is not simply a process, but rather an extensible framework which should be customized for specific organizations or projects.
  - The first book to describe the process was titled The Unified Software Development Process (ISBN 0-201-57169-2) and published in 1999 by Ivar Jacobson, Grady Booch and James Rumbaugh.[19)]
  - UP是一種程序架構，可用以設計客製化的程序
  - 以反覆的(Iterative)漸進的(Incremental)方式完成軟體系統的開發
  - 定義何人(Who)□何事(What)□何時(When)與如何(How)開發軟體

- UP and RUP
  - The best-known and extensively documented refinement of the Unified Process is the Rational Unified Process (RUP).
  - The name Unified Process as opposed to Rational Unified Process is generally used to describe the generic process, including those elements which are common to most refinements. The Unified Process name is also used to avoid potential issues of trademark infringement since Rational Unified Process and RUP are trademarks of IBM.

- UP 由階段(Phases)與流程(Processes)組成：
  - Phases
    - Inception (初始): 訂定專案願景(Vision)(專案範圍)
    - Elaboration (詳述): 決定要做什麼與需要什麼，以及決定系統架構
    - Construction (建構): 進行產品開發建構

- Transition (轉換/轉移): 將產品發佈給使用者

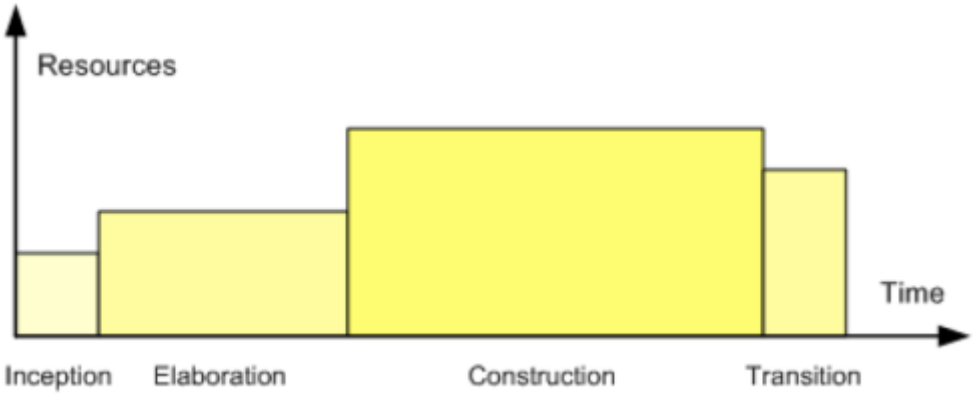


Fig. 13: Four Phases of the Unified Process

figure 13顯示了典型的UP四個階段所需的資源與耗費的時間。

- The Elaboration, Construction and Transition phases are divided into a series of time boxed iterations. (The Inception phase may also be divided into iterations for a large project.)
- Each iteration results in an increment, which is a release of the system that contains added or improved functionality compared with the previous release.

- 程序規範 (Process Disciplines):

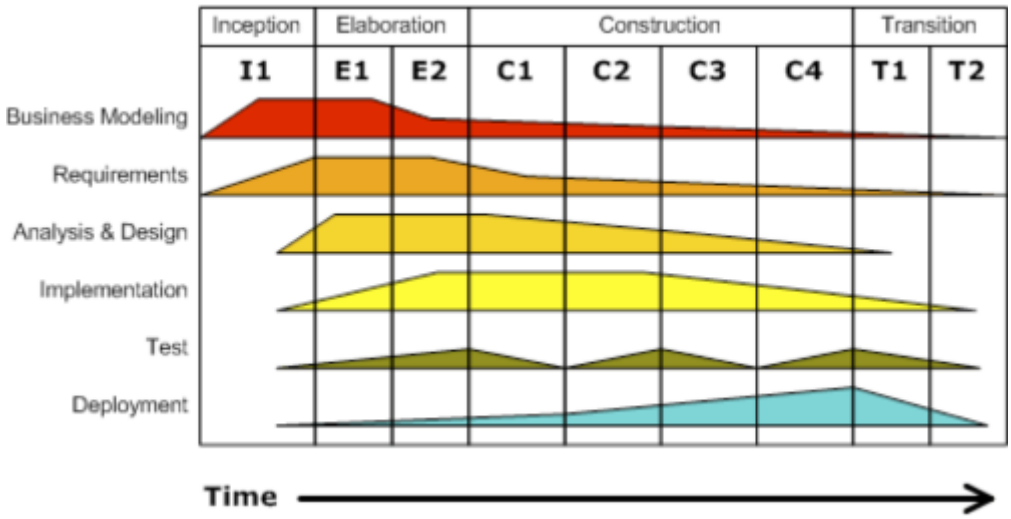


Fig. 14: Iterative Development of UP

- Business Modeling (企業塑模/塑模)
  - 瞭解企業的環境
  - 建立系統的願景
  - 建立企業的模型
- Requirements (需求)
  - 蒐集詳細的資訊
  - 定義功能性需求
  - 定義非功能性需求
  - 訂定需求的優先等級
  - 發展使用者介面
  - 與使用者評估需求
- Analysis and Design (分析與設計)

    - 分析與設計支援服務的架構與部署環境
    - 分析與設計軟體的架構
    - 分析與設計使用案例的實現
    - 分析與設計資料庫
    - 分析與設計系統與使用者介面
    - 分析與設計系統的安全與控管機制
- Implementation (實作)
- Test (測試)
    - 定義與執行單元測試
    - 定義與執行整合測試
    - 定義與執行可用性測試
    - 定義與執行使用者驗收測試
- Deployment (部署)
    - 取得硬體與系統軟體
    - 封裝與安裝元件
    - 教育訓練
    - 資料的轉換與初始化

- UP consists of several models [20)]

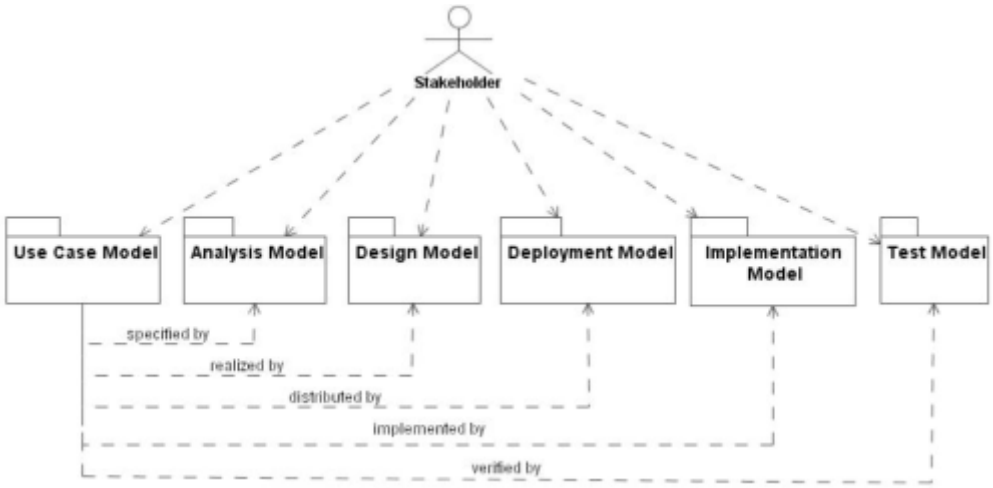


Fig. 15: UP的Model集合

- Those models are represented by UML(Unified Modeling Language)

1)

P. Naur and B. Randell, (Eds.), “Software Engineering: Report of a conference sponsored by the NATO Science Committee”, Garmisch, Germany, 7-11 Oct. 1968

2)

B. Randell and J.N. Buxton, (Eds.), “Software Engineering Techniques: Report of a conference sponsored by the NATO Science Committee”, Rome, Italy, 27-31 Oct. 1969

3) , 4)

Phillip A. Laplante, “What Every Engineer Should Know about Software Engineering”, ISBN 9780849372285, 2007.

5)

Alain Abran, James W. Moore, Pierre Bourque, Robert Dupuis, Pierre Bourque and Robert Dupuis, “Guide to the Software Engineering Body of Knowledge”, 2004 Version, IEEE Computer Society, ISBN 0-7695-2330-7, 2004.

6)

P. Naur and B. Randell, (Eds.), “Software Engineering: Report of a conference sponsored by the NATO Science Committee”, Garmisch, Germany, 7-11 Oct. 1968.
7)

I. Jacobson, M. Ericsson, and A. Jacobson, “The Object Advantage: Business Process Reengineering With Object Technology”, Addison-Wesley Professional, ISBN: 978-0201422894, September 30, 1994.
8)

William R. Duncan, “A Guide to the Project Management Body of Knowledge”, Project Management Institute, 1996.
9)

林信惠, 黃明祥, 王文良, “ 軟體專案管理 ” ,初版,智勝文化, 2002.
10)

Michael Jackson, “The World and the Machine”, ICSE, pp. 283-292, 1995.
11)

Webster Dictionary
12)

Martin Fowler, “Analysis Patterns: Reusable Object Models”, Addison-Wesley, ISBN 0-201-89542-0, 1997.
13)

John Daniels, “Models and Abstraction”, Object Expert, Vol.1(3), Mar./Apr. 1998.
14)

Jeff Rothenberg, “The Nature of Modeling”, Artificial Intelligence, Simulation, and Modeling, John Wiley & Sons, New York, pp. 75-92, 1989.
15)

Sebastiá Tyrrell, “The Many Dimensions of the Software Process”, ACM Crossroads, 2001.
16)

B. Boehm , “A Spiral Model of Software Development and Enhancement”, ACM SIGSOFT Software Engineering Notes, 11(4):14-24, August 1986.
17)

Wikipedia, http://www.wikipedia.org
18), 19), 20)

Ivar Jacobson, Grady Booch, and James Rumbaugh, “The Unified Software Development Process”, Addison-Wesley Professional, 1 edition, ISBN: 978-0201571691, February 14, 1999.