ISBN 978-957-729-659-7

第1章 爲什麼需要軟體工程？

ISBN 978-957-729-659-7

本章大綱

- 1.1　何謂軟體工程？
- 1.2　軟體工程的內涵
- 1.3　軟體工程的歷史
- 1.4　軟體工程的現況
- 1.5　結語

ISBN 978-957-729-659-7

學習目標

- 瞭解軟體開發所遭遇的問題
- 知道軟體工程演變的歷史
- 能夠定義何謂「工程」與「軟體工程」
- 瞭解軟體開發是一項專業且需要被管理
- 瞭解軟體工程的原則及其基礎

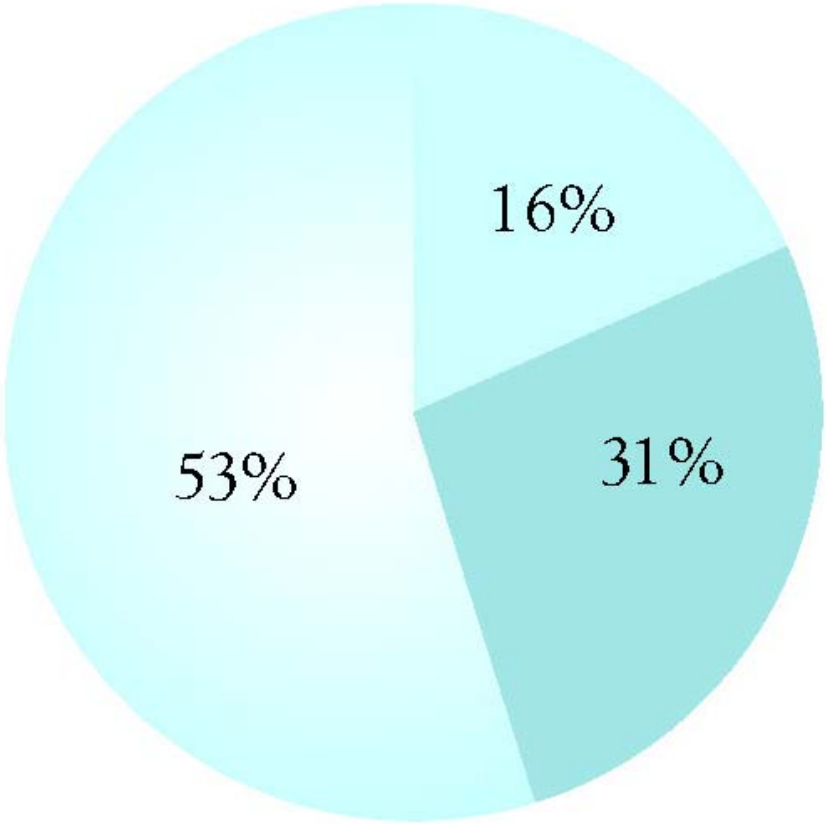
ISBN 978-957-729-659-7

何謂軟體工程？(1/4)

- 何謂「軟體」？
  - 由一組物件所構成，包括：程式、文件以及資料。
  - 軟體重要的是運作在電腦上，所呈現之動態行爲。
  - 軟體是真實世界的模型。
  - 軟體是數位形式的知識。
  - 軟體是儲存知識的媒體。
- 軟體開發是一項高難度、高風險的活動，由於高失敗率，故有所謂的「軟體危機」之說。The Standish Group在1995年發表的一份著名研究報告（如圖1.1），發現其中53%的專案預算超出原訂金額189%，時程超出原訂日程221%，而系統卻只能提供預定功能的61%。 如圖1.1所示。

ISBN 978-957-729-659-7

圖1.1 軟體開發專案的成功率



ISBN 978-957-729-659-7

何謂軟體工程？(2/4)

- 軟體工程的由來
  - 鑑於軟體開發所遭遇到的困境，北大西洋公約組織（North Atlantic Treaty Organisation, NATO）於1968年舉辦了首次的軟體工程學術會議，並於會中提出以「軟體工程」來界定軟體開發所需之相關知識，並且建議：「軟體開發應該是類似工程的活動」。

ISBN 978-957-729-659-7

何謂軟體工程？(3/4)

- 綜合而言，軟體工程可被描述如下：
  - 範圍：研究軟體流程、開發原則（principle）、工程技術與管理以及相關的表示法。
  - 目標：生產優質軟體、準時交付、符合預算、滿足顧客的要求與需要。
  - 內容：一門知識或學科其中包含有工程原理、方法學、軟體流程等。

ISBN 978-957-729-659-7

何謂軟體工程？(4/4)

- 意涵：
  - 優質軟體：包含屬性有簡單、彈性、可維護性、可讀性等。
  - 準時交付：包含項目有時程安排、進度追蹤、專案控管等。
  - 符合預算：包含項目有成本分析、軟體規模估算等。
  - 滿足用戶需要：包含項目有需求分析、變更管理、版本管理等。

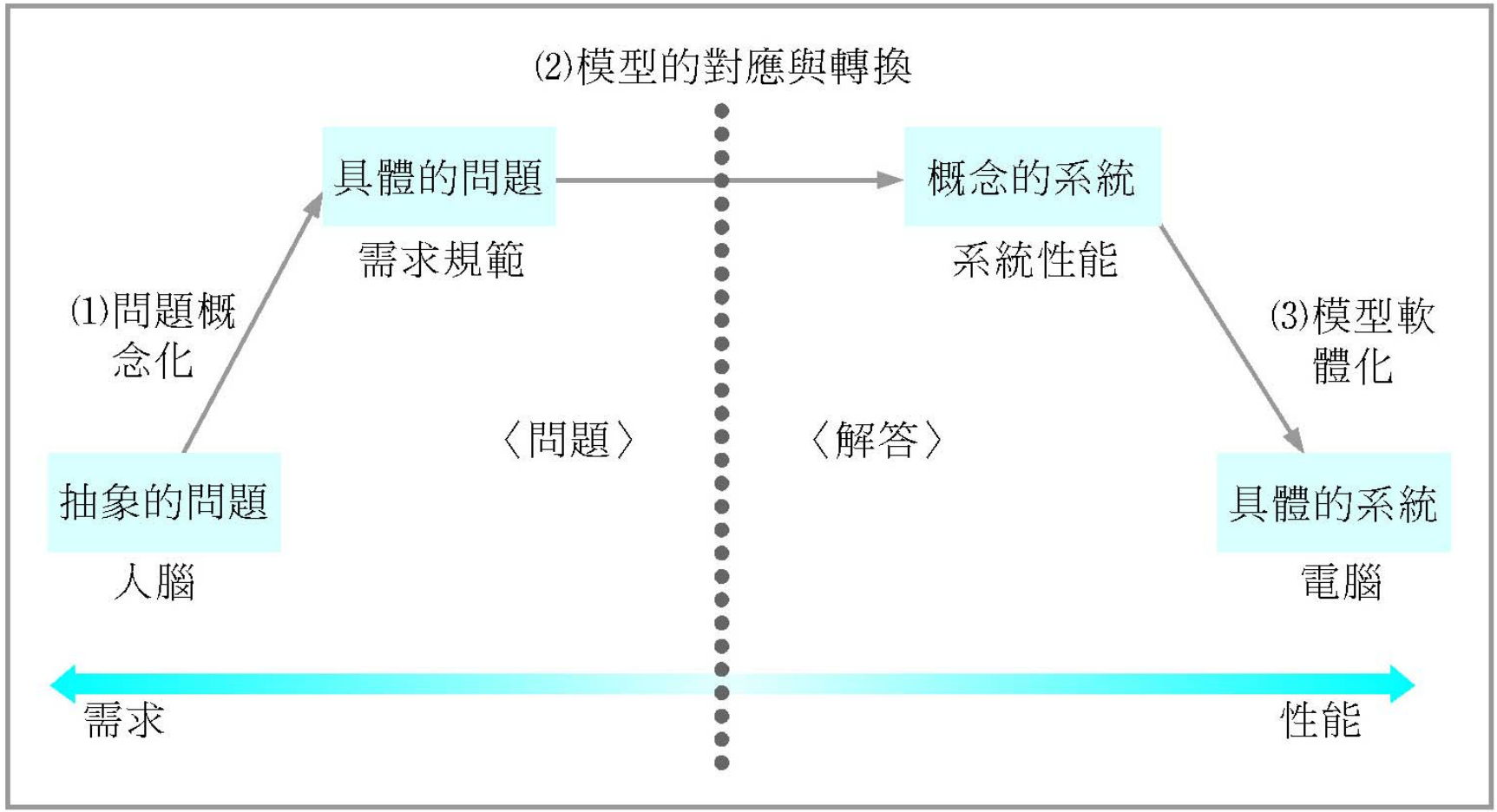
ISBN 978-957-729-659-7

軟體工程的內涵(1/4)

- 本書的定義。所謂「軟體工程」，就是研究如何「建立與運用工程原則，以便從一個抽象的『問題』，推導（或創造）出具體的軟體『解答』」，參考圖1.2。

ISBN 978-957-729-659-7

圖1.2軟體工程：從問題到解答



ISBN 978-957-729-659-7

軟體工程的內涵(2/4)

- 一般而言，從問題到解答的過程可依序分爲五個步驟：
  - 定義及描述問題
  - 分析問題
  - 尋找問題之解答
  - 選擇最適當之解決方案（亦即最佳化）
  - 實施解決方案
- 軟體工程解決問題的方法
  - 首先，是以適當的工具，將問題概念化與模型化。
  - 其次，是模型的轉換與對應。從定義問題的模型，轉換到解決問題的模型，其中包含了構思、尋找、評估，以及最佳化的過程。
  - 最後，是模型軟體化。將設計好的概念模型，透過適當的工具以軟體實作出來。

ISBN 978-957-729-659-7

軟體工程的內涵(3/4)

- 軟體工程的基本原則
  - 原則一：焦點分離
  - 原則二：嚴謹且正式
  - 原則三：抽象化
  - 原則四：模組化
  - 原則五：通用性
  - 原則六：預視改變
  - 原則七：遞增法

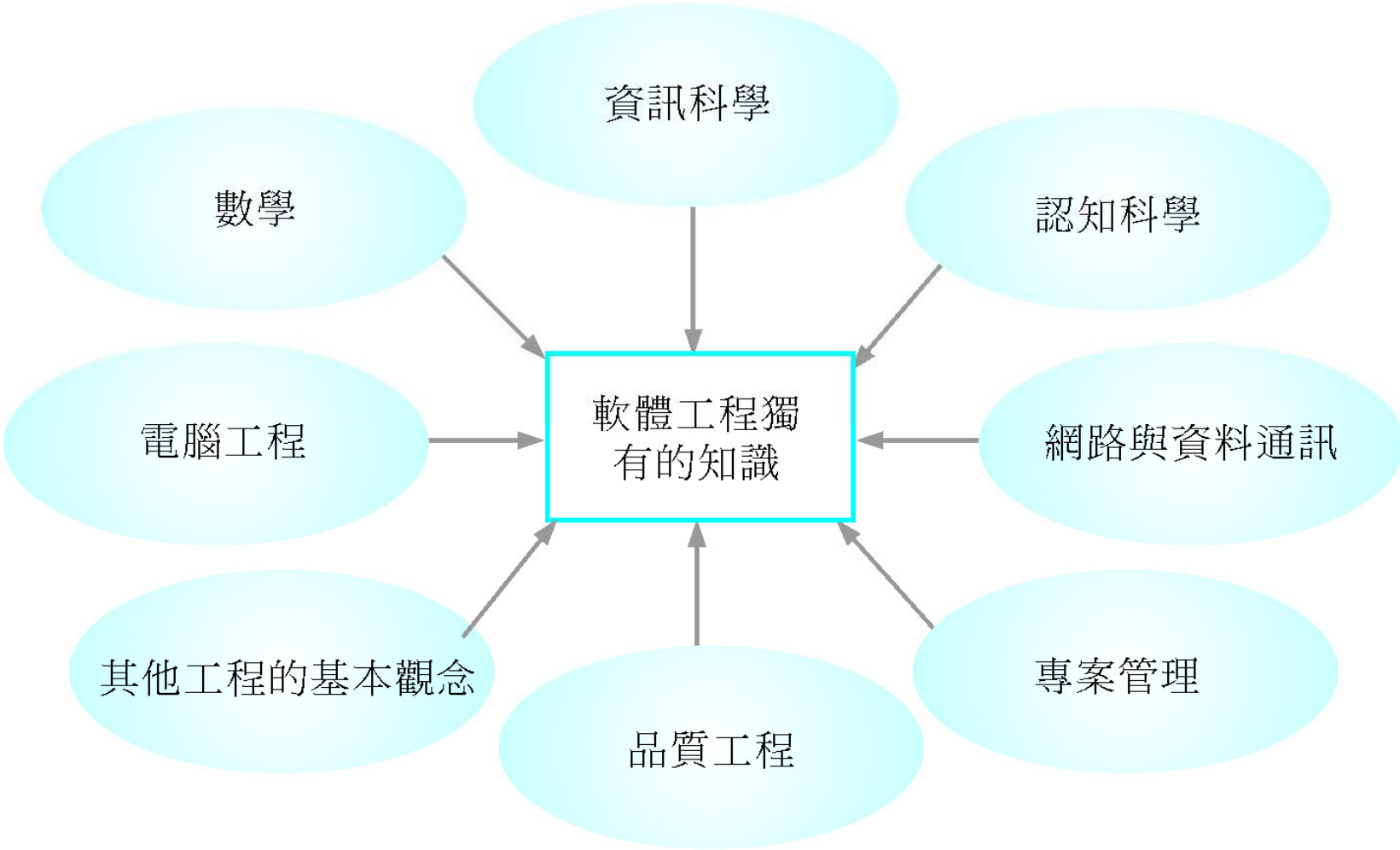
ISBN 978-957-729-659-7

軟體工程的內涵(4/4)

- 軟體工程的核心知識
  - 軟體需求
  - 軟體設計
  - 軟體建構
  - 軟體測試
  - 軟體維護與更新
  - 軟體構型管理
  - 軟體工程管理
  - 軟體開發流程
  - 軟體工程工具與方法
  - 軟體品質
- 軟體工程的周邊知識以及與電腦科學的差別，如圖1.3及表1.2。

ISBN 978-957-729-659-7

圖1.3 軟體工程周邊學門的知識



ISBN 978-957-729-659-7

表1.2 軟體工程與電腦科學的差別

	軟體工程	電腦科學
目標	構建易使用且高效的軟體，從而提高人們的工作效率與生活的舒適度	探索正確的計算與建模方法，從而改進計算方法本身
產品	軟體（如辦公套件與編譯器）	演算法（如各種排序法）或抽象的問題（如哲學家進餐問題）
進度與時間表	軟體專案都有特定的進度與時間表	研究專案一般不具有設定的進度與時間表
關注點	軟體工程關注如何為用戶實現價值	軟體理論關注的是軟體本身運行的原理，如時間複雜度、空間複雜度，以及演算法的正確性
變化程度	隨著技術和用戶需求的不斷變化，軟體開發人員必須時刻調整自己的開發技能，以適應當前的需求。同時軟體工程本身也應處於不斷的發展中	對於某種特定問題的正確解決方法永遠不會改變
需要的其他知識	相關領域的知識	數學

ISBN 978-957-729-659-7

軟體工程的歷史(1/2)

- 程式設計的演化
  - 軟體工程的史前時代，始於1950年代初期，此時的電腦編程很簡單，沒有什麼系統化的方法，軟體開發也沒有任何管理，此時屬於探索性編程的年代。
  - 1960年代初期，高階程式語言相繼出現，大幅地減少了軟體開發所需的心力。
  - 1960年代晚期，軟體工程誕生。此時主要的思維，是以控制結構爲重心的程式設計。
  - 1970年代初期，人們又發現程式的「資料結構」也很重要，到了晚期，進一步發展成針對資料流（dataflow）來做設計。

ISBN 978-957-729-659-7

軟體工程的歷史(2/2)

- 1980年代初期提出「物件導向設計」的概念，其後逐漸成爲軟體設計的主流直至今日。

❑ 軟體工程的演進

- 除了程式設計之外，其它相關軟體工程的技術，如軟體生命週期模型、需求分析、專案管理、軟體測試、除錯技巧、品質管理、軟體測量，以及軟體設計輔助工具等，也都有顯著的進展。
- 既然是工程，軟體的度量也是不可或缺的，軟體尺度（software metric）的技術因此逐漸發展，以協助管理軟體專案與系統品質。

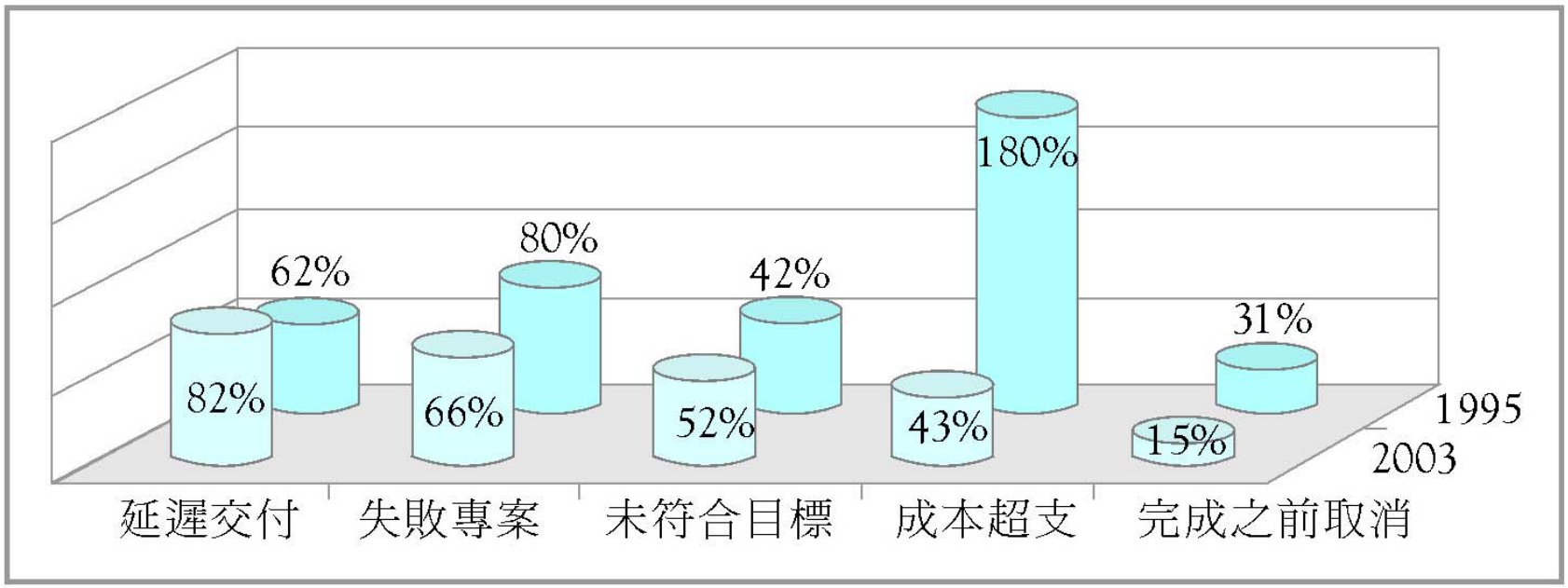
ISBN 978-957-729-659-7

軟體工程的現況(1/4)

- 今日軟體產業的現況，多數組織依然遵循舊有的習慣，仍舊忙著將軟體送出門，以致於鮮少花費時間在教育或培訓如何有效解決問題。
- 過時的作業模式變得組織化，很少人知道（或是能夠整合）最佳的實務以改善現況。雖然在某些特定領域有所進展，但軟體這一行業快速地演變，導致整體情況幾乎沒有改善，如圖1.5所示。

ISBN 978-957-729-659-7

圖1.5 GHAOS 1995年與2003年的比較



ISBN 978-957-729-659-7

軟體工程的現況(2/4)

- 軟體本身的變化
  - 程式的規模從數百行到數千行，再進展到數萬乃至數十萬行。
  - 作業環境從簡單的批次運算，演變成複雜的多工系統（multi-programming）、分時作業、分散式計算、視窗介面，以致於今日網際網路、多層式的系統架構等。
  - 軟體所處的環境也不同，現今改由強烈的市場力量所驅動，造成軟體開發受到持續的壓力。
  - 人們對於軟體的需求有增無減，也導致人才短缺，軟體工程所需的技術經常供給不足。
  - 莫爾定律（Moore's Law）不斷地擴展，導致軟體變得更加複雜。

ISBN 978-957-729-659-7

軟體工程的現況(3/4)

- 教育訓練
  - 當前的教育趨勢：如今軟體工程的相關知識已經逐漸發展成熟，可成爲獨立學門。
  - 認證與執照：認證是一個志願的流程，目的是幫助大眾瞭解，誰在專業上是合格的。至於執照則是非自願的，由政府發給，必須通過後才能執業。
- 流程與品管
  - 軟體專案的失敗多數時候是由於管理不良（或根本不存在）所致；若沒有應用合理的管理原則，軟體開發在技術面上就很難成功。因此，管理才能與技術技能一樣重要。

ISBN 978-957-729-659-7

軟體工程的現況(4/4)

- 1984年，美國國防部與卡內基美隆大學（Carnegie Mellon University）的軟體工程學院（Software Engineering Institute, SEI）聯手，共同推動軟體品質改善計畫。使用「軟體能力成熟度整合模型」（Capability Maturity Model Integration, CMMI）來改善軟體開發流程。
- 流程的改善無法速成，因爲能力成熟度的改善，牽涉到組織內人員的思維與工作文化，特別費時與困難。CMMI的理想若能實現，則透過軟體流程的改善，可降低成本並提高生產力。

ISBN 978-957-729-659-7

結語

- 軟體工程發展至今，依然存在著許多問題，尚未能有標準的成功法則。所以，學習軟體工程這門科學，不能一知半解。否則，或許差之毫釐，卻失之千里。
- 不只要學習Know-what與Know-how，還必須要瞭解Know-why。應透過思辨的過程，深入瞭解各種方法背後的假設與邏輯，以避免掉入各種軟體開發陷阱。
- 看問題應有全面周延的認識，避免直線式思考或斷章取義。管理學、心理學與社會科學，相對於工程技術而言，也同樣重要。
- 工程師們須培養多樣化的技能，尤其是瞭解與分析用戶需求的能力。此外，在團隊合作的過程中，溝通與管理的技能也不可或缺的；在可預見的未來，人的素質依然決定軟體產品的品質。