第 1 章

導論

本章的目標是介紹軟體工程這個主題，並描繪出本書的整體架構。讀完本章後你將能夠：

- 瞭解何謂軟體工程以及軟體工程的重要性
- 回答軟體工程基本觀念的重要問題
- 瞭解對軟體工程師而言非常重要的專業與道德素養

目前幾乎全世界所有國家，都必須依賴複雜的電腦化系統。國家基礎建設與水電等公用事業的運作，也都是依靠電腦化系統。而且大部分的電子產品，都會包含電腦和控制軟體。現在的製造業與物流業，已經和金融業一樣完全電腦化了。因此，如何以合乎成本效益的方式生產和維護軟體，對整個國家、甚至整個國際的經濟都是非常重要的。

軟體工程是一門工程學科，目的是要以合乎成本效益的方式，開發出高品質的軟體系統。軟體是一種抽象而且無形的東西，它不受物質的限制、也不會受到物理定律或製造過程的控管。就某些方面來看，這樣的特質可以簡化軟體工程，因爲軟體的潛力不會受到實體的限制。然而從另外一方面而言，缺乏自然的限制表示軟體很容易變得極度複雜，因而令人非常難瞭解。

軟體工程（software engineering）這個觀念是在 1968 年一場名爲「軟體危機」的研討會中首次提出。當時的軟體危機是因爲出現了由積體電路所組成的新一代電腦硬體，它們的能力讓原本不可能實現的電腦應用程式變得可行，因而導致軟體的大小與複雜度呈現級數式的成長。

早期建置這類系統的經驗顯示，非正規的軟體開發方法其結果並不夠好。當時的大型專案開發有時會延遲好幾年，專案的成本也會比原先的預估超出許多，而且軟體不可靠、難以維護，執行效能也很差。軟體開發在當時已呈現危機。雖然硬體的成本一直下降，但軟體的成本卻快速攀升，這時候就需要新的技術與方法來控制大型軟體系統的複雜度。

這些技術後來便成爲軟體工程的一部份，而且現在已經普遍使用。雖然軟體的生產力提高，但是軟體系統的需求也越來越複雜。例如整合電腦與通訊系統所產生的新技術，以及複雜的圖形化使用者介面，這些對於軟體工程師都是新的挑戰。由於許多公司仍然尚未有效運用軟體工程技術，因此還是有非常多專案所開發出來的軟體並不可靠、時間拖延和超出預算。

筆者認爲自 1968 年以來，軟體工程已經有了非常長足的進步，它的發展有助於軟體品質的改善。現在我們已經比較瞭解軟體開發過程中包含的一些主要

活動，對於軟體的需求規格、設計與建置等活動也發展出許多有效的流程方法。利用一些新的符號表示法與工具，可減少開發大型複雜系統所花的人力和時間。

目前軟體工程界還沒有出現任何單一的「完美」產品。目前市面上有非常多種不同類型的系統，而各種系統都有不同的組織在使用，這意味著我們需要有多種方式來開發軟體。不過無論是哪種技術，其軟體程序和系統架構的觀念是大同小異的，而這些都是軟體工程的基本要素。

軟體工程師可以對他們的成就感到驕傲，如果沒有這些複雜的軟體，人類就無法探索太空，也沒有網際網路和現代的電信科技，而且所有的旅行方式都會變得更危險而且更昂貴。軟體工程的貢獻非常大，而我深信隨著這個領域越來越成熟，它在 21 世紀將會有更大的貢獻。

1.1 軟體工程常見問題

本節將回答一些與軟體工程有關的基本概念問題，以及表達我對這個領域的一些觀點。本節內容將以「FAQ」（常見問答集）的形式呈現。這種方式普遍應用在網際網路的新聞群組中，提供初學者某些常見問題的解答。我認爲這種方式是介紹軟體工程最有效的方式。

圖 1.1 摘要整理出本節所有問題的答案。

1.1.1 何謂軟體？

許多人把**軟體**（software）這個術語和電腦程式劃上等號，事實上這個說法太狹隘了。軟體不只是程式而已，它還包括了可以讓這些程式正常運作所需的相關說明文件和組態設定資料。軟體系統通常包含許多獨立的程式、用來設定這些程式的組態設定檔、描述系統結構的系統說明文件、說明如何使用系統的使用說明文件，以及讓使用者知道如何下載最新產品資訊的網站等。

問題	解答
何謂軟體？	電腦程式和相關的說明文件。可以是為某位客戶或一般大眾市場所開發的軟體產品
何謂軟體工程？	軟體工程是一門著重在軟體生產的各類知識的工程學科
軟體工程與電腦科學有何不同？	電腦科學是與電腦基本觀念和理論有關的學科；軟體工程則是著重在開發和發行有用軟體相關的實用知識
軟體工程與系統工程有何不同？	系統工程是涵蓋電腦系統開發過程的各面向，包括硬體、軟體和程序等。軟體工程是這個程序的一部分
何謂軟體程序？	軟體開發或演進的一系列活動
何謂軟體程序模型？	軟體程序的簡化表示方式，以特定的觀點來表示
何謂軟體工程的成本？	開發成本大約佔 60%，測試成本約佔 40%。對於客製化的軟體而言，其軟體演進成本經常會超過開發成本
何謂軟體工程方法？	軟體開發的結構性方法，包括系統模型、符號表示法、規則、設計建議以及程序指南等
何謂 CASE（電腦輔助軟體工程）？	針對軟體程序的活動提供自動化支援的軟體系統。CASE 系統通常用來支援某種軟體工程方法
好的軟體有哪些特性？	軟體必須具有使用者要求的功能和效能，而且應該具有可維護、可信任以及可用的特性
軟體工程面臨的主要挑戰為何？	要應付日益增加的多樣性、縮短開發時間的要求，以及開發出的軟體必須值得信任的要求

圖 1.1 軟體工程常見問題集

軟體工程師的主要工作就是開發這些軟體產品，也就是可以賣到顧客手上的軟體。這些軟體產品可以分成兩種類型：

1. **通用產品**（generic product）：它是由軟體開發公司所生產的獨立系統，公開銷售給市場上的任何顧客。這類產品最常見的是 PC 上的軟體，例如資料庫、文書處理程式、繪圖套裝軟體和專案管理工具等。

2. **客製化產品**（customized product）：這類系統是專門為某個客戶特別訂做的系統，軟體承包商會針對這個客戶量身訂做它要的軟體。例如電子設備的控制系統、支援某個商業程序的系統，以及航管控制系統等都是這類軟體。

這兩種不同類型的軟體最主要的差異在於：開發通用軟體產品的公司可以控制軟體的規格，而客製化產品的規格通常是由購買此軟體的客戶來制訂與控制，軟體開發人員必須遵循客戶所制訂的規格進行開發。

不過，現在這兩種軟體之間的界線已經越來越模糊。有越來越多的公司是採用由某種通用產品著手，然後再根據某個客戶的需求客製化。像企業資源系統（Enterprise Resource Planning, ERP）就是最好的例子，例如 SAP 系統是個龐大而複雜的系統，開發者會把客戶公司的企業法則、程序、所需要的報表等資訊輸入系統中以進行客製化。

1.1.2 何謂軟體工程？

軟體工程（software engineering）是一門著重在軟體生產的各類知識的工程學科，範圍從最開始的系統規格制訂，到系統上線後的維護階段都包括在內。在此定義中有兩個重要的詞語：

1. **工程學科**：工程師會讓事情運作，他們會應用適當的理論、方法和工具，提出問題的解決方案，即使沒有適當的理論和方法也會嘗試找出方法。工程師也體認到他們的工作必須受組織與財務的限制，所以會在這些限制條件下尋找解決方案。
2. **軟體生產的各類知識**：軟體工程不只包含軟體開發時的技術性過程，還有一些相關的活動。例如軟體的專案管理以及支援軟體生產的開發工具、方法和理論。

一般而言，軟體工程師會採用有系統、有組織的方式來進行工作，因爲這些方式通常是生產高品質軟體的最有效方法。不過，工程的方法論通常是針對當時的情況選擇最適合的方法，以及選擇在某些情況下可能有效率、或是非正

規但更具創意的開發方式。非正規的開發方式特別適合開發網站系統，因爲它必須融合軟體和繪圖設計技巧。

1.1.3 軟體工程與電腦科學有何不同？

基本上，**電腦科學**（computer science）是與電腦和軟體系統基本理論和方法有關的學科，而軟體工程則是與生產軟體時所面臨的實際問題有關。電腦科學領域中某些知識對軟體工程師而言非常重要，就像某些物理知識對電機工程師的重要性一樣。

在理想情況下，所有的軟體工程都應該以電腦科學的理論爲基礎，但實際上並非一定如此。軟體工程師經常會使用某些特別的（ad hoc）方法來開發軟體。電腦科學的完美理論不可能永遠都可以解決真實世界的複雜問題，這些問題有時候必須靠軟體來解決。

1.1.4 軟體工程與系統工程有何不同？

系統工程（system engineering）涵蓋電腦系統開發與演進過程的各個層面，而其中軟體扮演著主要的角色。因此系統工程會與硬體開發、企業規範與程序制訂、系統部署及軟體工程有關。系統工程師的工作包括指定系統、定義整體架構以及將不同部分整合成最後的完成系統等。他們與系統的個別元件（如硬體、軟體等）比較無關。

系統工程是一門比軟體工程更具歷史的學問。人們在傳統複雜的工業系統裝配組合上，已經有超過 100 年以上的經驗（例如飛機和化學工廠之類的系統）。然而，隨著系統中軟體的比例逐漸升高，系統工程的程序中就會開始採用如「使用個案塑模」（use-case modeling）與「組態管理」（configuration management）等軟體工程技術。第 2 章將會探討系統工程。

1.1.5 何謂軟體程序？

軟體程序（software process）是生產軟體產品的一連串活動與相關的成果。下列所示爲所有軟體程序所共有的 4 個基本程序活動（本書稍後說明）：

1. **軟體規格制訂**（software specification）：定義軟體的功能以及運作的限制。
2. **軟體開發**（software development）：設計與撰寫軟體。
3. **軟體確認**（software validation）：必須確認軟體是否符合客戶的需求。
4. **軟體演進**（software evolution）：軟體必須持續修訂，以符合客戶和市場的需求變化。

不同類型的軟體需要不同的開發程序。舉例而言，飛機裡的即時軟體在開發程序開始前就必須制訂好完整的規格，而電子商務系統的規格制定通常是與程式撰寫一起進行的。因此針對不同種類的軟體，以上活動可能會有不同的組合方式，描述的詳細程度也不同。如果使用不適合的軟體程序，可能會導致軟體品質降低、開發成本提高，或是開發出沒有用的軟體產品。

軟體程序將在第 4 章討論，而有關軟體程序改善的一些重要議題則會在第 28 章介紹。

1.1.6 何謂軟體程序模型？

軟體程序模型（Software Process Model）是以某個特定觀點呈現的軟體程序簡化描述。程序模型可能會包含軟體程序的部分活動、軟體產品以及與軟體工程有關的人員。軟體程序模型的類型有下列幾種：

1. **工作流程模型**（workflow model）：它可以展示程序中的輸入、輸出與相依關係的活動順序。此模型中的活動代表人為的動作。
2. **資料流程模型**（dataflow model）或**活動模型**（activity model）：以一組活動來表示程序，每一個活動都可以執行某些資料轉換的動作。它將展示此程

序的輸入（例如規格）如何轉換成輸出（例如設計）。這裡的活動所表示的轉換動作，可能是由人員或電腦來執行。

3. **角色／動作模型**（role/action model）：用來表示軟體程序相關人員的角色以及各自負責的活動。

大部分的軟體程序模型，都是根據下列其中一種軟體開發模式而來：

1. **瀑布式**（waterfall approach）：這種方式是將瀑布上層的活動表示成分開的程序階段，例如制訂需求規格、軟體設計、實作和測試等階段。每個階段在確認「簽結」（sign-off）後，開發動作就會進入到下一階段。
2. **反覆式開發**（Iterative development）：這種方式會反覆進行規格制訂、開發與確認等活動。系統一開始是以非常抽象的規格快速開發而成，然後再根據客戶的意見做調整，產生滿足客戶需求的系統。接下來可能是直接交付此系統，或者是使用更結構化的方式來實作，產生更堅固而且容易維護的系統。
3. **元件式軟體工程**（component-based software engineering, CBSE）：這項技術是假設系統的各部分組成元件都已經存在，系統開發的過程主要是整合這些元件，而不是從無到有。CBSE 在第 19 章詳細探討。

在第 4 章和第 17 章將會再探討這些通用的程序模式。

1.1.7 何謂軟體工程的成本？

這個問題並不容易回答，因爲軟體程序中各個活動之間正確的成本分配情形，必須根據使用的程序和開發的軟體種類而定。舉例而言，即時軟體通常會比網站系統需要更詳盡的驗證和測試。而不同的開發模式在各個軟體程序活動之間的成本分配也不同。如果假設開發一個複雜系統的總成本爲 100 個成本單位，則這些成本單位的分佈情形可能會如圖 1.2 所示。

假如是瀑布式開發模式，規格制訂、設計、開發以及整合的成本是分開評估的。請注意，此時系統整合與測試是成本最高的開發活動，通常這部分活動大約佔總體開發成本的 40%，但是有些關鍵任務系統則可能高達系統總體成本的 50%。

如果使用反覆式方法開發，那麼在規格制訂、設計與開發這 3 個階段之間就沒有明顯的界線存在。這裡的規格制訂成本會降低，因爲這個方法在開發前只會制訂出一些初步的大略規格，然後在開發過程中才同時進行詳細規格的制訂、設計、實作、整合與測試等所有活動。不過，在初步的實作完成後，仍然需要進行獨立的系統測試活動。

元件式軟體工程是最近才開始廣泛使用，因此還沒有各種軟體開發活動的精確成本數據。不過可以得知的是，它的開發成本相對於整合和測試成本是降低的，而整合和測試成本會增加，這是因爲要確認這些元件組合起來真的可以符合規格需求，而且運作正常。

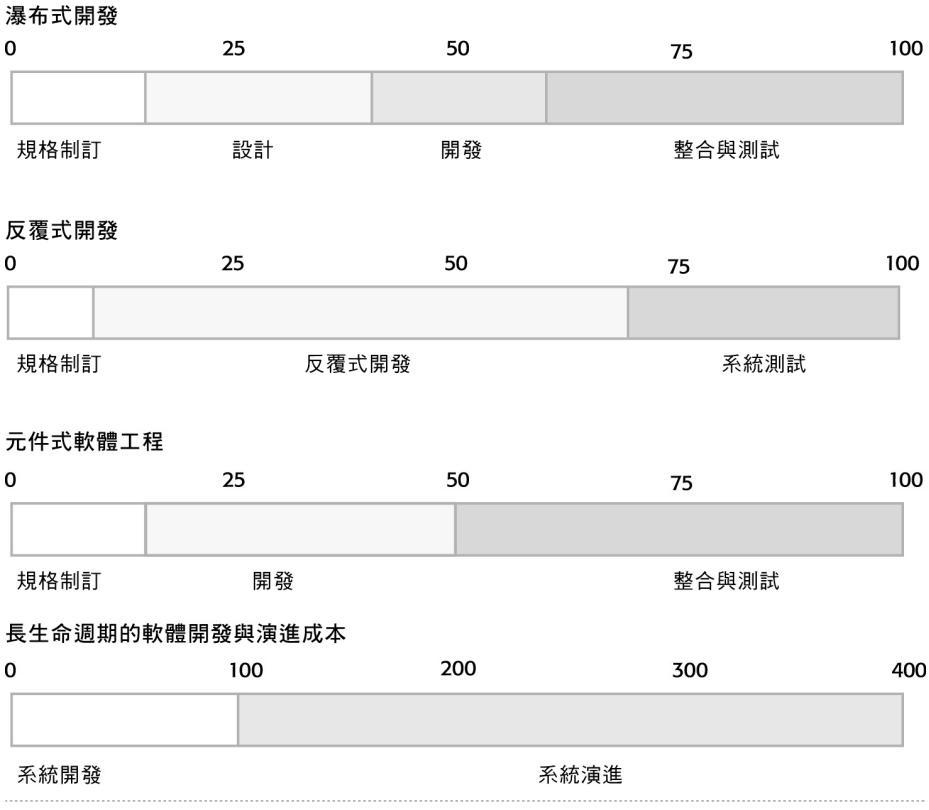

圖 1.2　軟體工程各活動的成本分配情形

在開發成本方面，還必須考慮在軟體正式上線後所發生的軟體修改成本。軟體演進成本會隨著軟體種類不同而有很大差異。對於有較長生命週期的軟體系統，例如可能會使用長達 10 年以上的控制系統，這些成本可能會超過開發成本的 3 到 4 倍之多（參見圖 1.2）。

以上的這些成本分佈資訊，是適用於客製化的軟體，也就是由客戶指定規格，然後由承包商開發的軟體。然而對於大部分的 PC 軟體產品，它們的成本特性都不一樣，這些軟體產品通常是根據大綱式的規格開發而成，使用的是演進式的開發方式，所以其制訂規格的成本比較低。不過，因爲這種軟體的目的是要用在許多種不同的電腦設定環境下，所以必須經過非常廣泛的測試。圖 1.3 是這類產品可能的成本分佈。

通用型軟體產品的演進成本尤其難以估算，許多軟體產品不會有太多的正式改版動作。軟體產品一旦公開發行之後，下一個版本就已經開始研發了。基於市場行銷的因素，廠商應該會以新產品（但與舊版相容）的方式發行，而不會發行舊版產品的修正版。因此，這種軟體的開發與演進成本是無法分開評估的（不像客製化軟體），而是直接算在系統下一個版本的開發成本上。

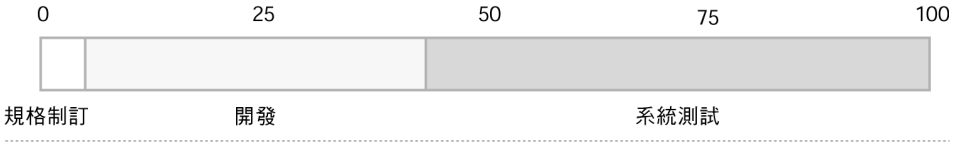

圖 1.3 產品開發成本

1.1.8 何謂軟體工程方法？

軟體工程方法是開發軟體的一些結構性方法，它們的目的是要以合乎成本效益的方式生產出高品質的軟體產品。早在 1970 年代就已經發表了「結構式分析」（DeMarco, 1978）與 JSD（Jackson, 1983）等方法。這些方法主要是想要嘗試找出系統的基本功能元件，目前功能導向（function-oriented）的方法仍然有人在使用。

從 1980 到 1990 年代，這些功能導向的方法又增加了一些物件導向（Object-Oriented, OO）的功能，例如由 Booch（Booch, 1994）和 Rumbaugh（Rumbaugh, et al., 1991）所提出的方法。現在這些方法已經整合成一個統一的方法，稱爲「統一塑模語言」（Unified Modeling Language），簡稱 UML（Booch, et al., 1999; Rumbaugh, et al., 1999a; Rumbaugh, et al., 1999b）。

目前爲止，還沒有一個完全理想的方法出現，而且不同的方法各有不同的適用領域。例如，物件導向方法通常適用於互動式系統，但不適合有嚴格即時需求的系統。

這些方法都是以圖形方式來表示開發系統的模型，然後再以模型當成系統的規格或設計。這種表示法包括多種不同的組成元件（圖 1.4）。

組成元件	說明	範例
系統模型描述	描述要開發的系統模型以及用來定義這些模型的符號表示法	物件模型、資料流程模型、狀態機模型等
規則	套用在系統模型上的限制	系統模型中的每一個實體都必須有一個唯一的名稱
建議	良好的設計經驗，遵循這些建議應該可以產生有良好組織架構的系統模型	任何一個物件不能有 7 個以上的子物件
程序指南	描述開發此系統模型時可能的後續活動，以及這些活動的架構	在定義與物件相關的運算動作之前，應該先將物件的屬性記錄成文件

圖 1.4　方法的組成元件

1.1.9 何謂 CASE？

CASE 是**電腦輔助軟體工程**（Computer-Aided Software Engineering）的縮寫，它包含許多不同種類而且範圍廣泛的程式，這些程式是用來支援軟體程序中的各項活動，包括需求分析、系統塑模、除錯及測試等。這些方法目前都

有相關的 CASE 技術可用，例如可使用對應符號表示法的編輯器、可根據規則檢查系統模型的分析模組，以及輔助建立系統文件的報表產生器等。CASE 工具還可能包含程式碼產生器，它可以根據系統模型與軟體工程師輸入的程序指南，自動產生原始程式碼。

1.1.10 好的軟體有哪些特性？

軟體產品的品質除了反應在它所提供的服務之外，還有其他許多相關的特性。這些特性不是直接衡量軟體的功能，而是反映軟體執行時的一些行爲、原始程式的架構，以及相關的說明文件。這些特性有時候也稱爲「非功能特性」（non-functional attribute），例如軟體對使用者查詢的回應時間快慢，以及程式碼的易讀性。

每個軟體系統的特性，顯然必須依據它的應用程式而定。例如，銀行系統一定要夠安全、互動式遊戲必須夠快、電話交換系統必須夠可靠等。這些特性歸納在圖 1.5，我個人認爲這些是成爲好軟體的必要特性。

產品特性	說明
可維護性（maintainability）	完成後的軟體必須能夠針對客戶的需求改變進行軟體的演進與維護。這是一項非常重要的特性，因為在變動的商業環境中，軟體的改變是無可避免的結果
可信賴度（dependability）	軟體的可信賴度包括可靠性、防護性及安全性。系統發生故障時，可信任的軟體不應該會造成實體或經濟上的損失
效率（efficiency）	軟體不應該浪費系統資源，例如記憶體和處理器時間。因此，效率包含了回應速度、處理時間、記憶體的使用率等因素
可用性（usability）	軟體必須能夠讓使用者容易上手，不用花太多功夫。這表示它必須有適當的使用者介面與說明文件

圖 1.5 良好軟體的必要特性

1.1.11 軟體工程面臨的主要挑戰為何？

21 世紀的軟體工程面臨下列 3 項主要挑戰：

1. **異質性的挑戰**：軟體系統逐漸變成跨網路的分散式系統，這類系統可能包括各種不同的電腦以及不同的支援系統，它經常需要將新軟體與舊系統（可能是以不同的程式語言撰寫而成）加以整合。這些異質性（heterogeneity）系統所面臨的挑戰是：該使用何種開發技術來建立值得信賴同時可處理異質問題的軟體。
2. **開發時間的挑戰**：許多傳統的軟體工程技術都很費時，而花費的時間主要是為了確保軟體能夠具有一定的品質。不過，今日的企業必須具備即時回應和應變的能力，因此它們使用的軟體也必須同樣能夠快速應變。這方面的挑戰是如何讓大型而複雜的系統，能夠在不降低系統品質的條件下，縮短開發的時間。
3. **信任度的挑戰**：隨著軟體深入我們生活的各個層面，軟體也必須讓我們更能信任才行，而那些透過網頁或網站服務介面所存取的遠端軟體系統更是如此。信任度的挑戰在於如何開發出能讓軟體展示出它值得使用者信任的技術。

當然，以上這些挑戰並非互不相干的。例如，有時為了讓舊系統具有網站服務的介面，可能需要進行快速的修改。為了克服這些挑戰，我們必須使用新的工具與技術，並以創新的方式結合與使用現有的軟體工程方法。

1.2 專業與道德上的責任

軟體工程師與其他領域的工程師一樣，他們的工作必須符合法律與社會的規範。軟體工程師的工作，所牽涉的不只是如何應用個人的技能，還包含了更廣大的責任。軟體工程師若要讓別人以專業人士來看待，則行為上必須謹守倫理與道德的規範。

軟體工程師必須具備誠實與正直的一般標準，他們不能利用本身具有的能力與技術從事不法的事，更不能有破壞軟體工程專業的行爲。不過，在可接受的行爲標準裡，有一些模糊地帶並沒有受到法律的規範，而純粹是職業上的道德。例如：

1. **保密**（confidentiality）：不管有沒有簽署保密合約，工程師都應該遵守雇主或客戶的保密要求。
2. **稱職**（competence）：工程師必須根據自己的能力做稱職的表現，不應該故意接受無法勝任的工作。
3. **智慧財產權**（intellectual property right）：工程師應該注意有關智慧財產權的法律相關規定，例如專利或版權。他們應該確保雇主和客戶的智慧財產受到保護。
4. **濫用電腦**（computer misuse）：軟體工程師不應該利用本身的技術與能力濫用其他人的電腦設備。濫用電腦的情況從單純的利用雇主機器玩電腦遊戲，到非常嚴重的散播病毒等都算。

有些專業性的社團和機構，在制定職業道德標準方面扮演了非常重要的角色，例如 ACM、IEEE 以及英國電腦協會等組織，都有發行專業人員應該遵守的行爲準則或道德規範，加入這些組織的會員必須接受並遵守這些規範。這些行爲準則或規範通常和基本的道德行爲有關。

ACM 和 IEEE 也合作制訂出聯合的道德準則和專業守則。這些準則有兩種版本，第一種比較簡短（圖 1.6），另一種則是在簡短版本中再加入詳細說明與主旨的完整版（Gotterbarn, et al., 1999）。這些準則背後的緣由闡述在完整版最前面兩段：

> 電腦對於商業、工業、政府、醫療、教育、娛樂以及整體社會而言，扮演了非常重要的角色，且其重要性與日俱增。軟體工程師可以透過直接參與或講授教學的方式，對軟體系統的分析、規格制訂、設計、開發、驗證、維護與測試等做出貢獻。由於軟體工程師參與了軟體系統的開發工作，所以他們可以對軟體系統做出好的貢獻或是造成傷害，也可能對別人或影響

別人做出好的貢獻或造成傷害。為了儘可能確保他們的努力是用在正途，軟體工程師必須承諾讓軟體工程成為有益的且受尊重的專業。根據這項承諾，軟體工程師應該支持下列的道德規範和專業守則。

「規範」（code）部分包含與專業軟體工程師的行為和決策有關的 8 項「準則」（principle），包括對開業者、教育人員、經理人、主管、政策制訂者以及受專業訓練的學員或學生等的規範。這些「準則」明示了參與的個人、群組和組織在道德上的責任關係，以及在這些關係中的主要義務。每個「準則」中的「條款」（clause）是用來說明這些關係的某些責任與義務。這些責任與義務是以軟體工程師的人性為出發點，尤其著重在受到軟體工程師的工作影響的人員，以及軟體工程實務上的一些獨特要素。「規範」部分則是將這些準則規定為軟體工程師或有志於成為軟體工程師者的責任與義務。

不同人可能有不同的觀點和目的，有時你可能會面臨道德上的兩難。例如，如果公司採用管理高層的政策，但是你不同意這樣的做法時，你要如何因應？當然，這必須視每個人的情況和不同意的內容而定。你覺得最好是根據你在組織內的職位權限提出異議還是服從？假如你對某個軟體專案覺得有問題，會在什麼時候向管理階層揭露這些問題呢？如果你只是懷疑就向主管提出，可能會被說成反應過度；但是如果太晚提出，則又可能會讓問題無法解決。

在專業的生涯中，我們時常會遇到這類道德上的兩難。幸好大部分情況都不是太糟糕，或是可以用簡單的方法解決。如果遇上無法解決的問題，工程師可能會選擇辭職，但是這樣又可能會影響到同事或家人。

最麻煩的情況是當他的雇主不遵守道德時。假設某家公司負責開發一個非常重視安全性的系統，但是由於時間的壓力，他們僞造安全驗證紀錄。此時工程師的責任是爲公司保密，還是要趕緊警告客戶這個系統可能不安全？

就安全性來看，這個問題沒有絕對正確的做法。雖然這個系統沒有根據事先設定的條件進行驗證，但是這些條件有可能太過嚴苛。事實上，說不定系統還是可以安全的運作。反過來看，即使經過適當的驗證，系統還是有可能會失

敗，或是造成意外。太早揭發問題可能會傷及雇主和其他員工；但是如果沒有揭露，又可能會傷害到其他人。

軟體工程的道德規範與專業守則

ACM/IEEE-CS 聯合工作小組的軟體工程道德與專業守則

前言

這個簡短版本的規範是以較高階抽象的方式，摘要說明軟體工程專業人士應有的抱負。完整版本中的條文有加入範例和詳細說明，闡述這些抱負如何影響專業人士的行為。如果沒有這些抱負，詳述部分就變成只是冗長的法律條文；但是如果沒有詳述的內容，這些抱負又會變成空談。因此，這些抱負和詳述形成了一個緊密的規範。

軟體工程師應該承諾讓軟體的分析、規格制訂、設計、開發、測試及維護等成為有益大眾且受尊重的專業。依照軟體工程師對大眾的健康、安全和福利的承諾，他們應該遵守下列 8 個準則：

1. PUBLIC：軟體工程師應該維護大眾的利益。
2. CLIENT AND EMPLOYER：軟體工程師應該以讓他的客戶和雇主得到最佳利益為職責，並且維護大眾利益。
3. PRODUCT：軟體工程師應該確保他的產品和相關的修改能夠儘可能符合最高的專業標準。
4. JUDGMENT：軟體工程師在專業判斷上應該維持正直與中立。
5. MANAGEMENT：軟體工程師的經理人和主管應該在軟體開發與維護上支持與提倡合乎道德的管理方法。
6. PROFESSION：軟體工程師應該提升專業的誠實與名譽，並符合大眾利益。
7. COLLEAGUES：軟體工程師應該公平的支援他們的同事。
8. SELF：軟體工程師在其專業領域上應有終身學習的態度，並且應該在專業領域上提倡合乎道德的方法。

圖 1.6 ACM/IEEE 道德規範（©IEEE/ACM 1999）

在這個情況下，你必須自己做決定。考量可能造成的傷害、造成傷害的範圍以及被波及的人員，這些都會影響到你的決定。如果情況非常危急，甚至可

以考慮透過全國性的媒體對外公佈。然而爲了尊重雇主的權益，你還是應該先試著解決問題。

其他的道德問題可能還有參與軍事和核武有關的系統開發。有些人對這類問題非常敏感，他們不希望參與任何和軍事有關的系統開發工作。有些人則會願意開發軍事系統，但是排斥開發武器系統。另外有一些人則會認爲國防安全重於一切，參與武器系統的開發並無不妥。這類道德上的認定標準，完全視當事人的觀點而定。

在這種情況下，最重要的是雇主與員工之間必須事先溝通彼此的觀點。當有類似軍事或核武系統的開發工作時，雇主才能夠指派願意接受這類工作的人員。同樣的，如果員工事先清楚表明不願意參與這類系統的開發，雇主也不能對其施加壓力強迫他接受。

過去幾年來，一般領域中的道德與專業責任逐漸受到重視。它是先從哲學的觀點來考量道德上的基本準則，參考這些基本準則之後再來探討軟體工程的職業道德。這是由 Laudon（Laudon, 1995）所採用的方法，然後由 Huff 和 Martin（Huff and Martin, 1995）再加以擴充。

不過，我個人發現這個方法太過抽象，很難應用在日常的工作上。我比較喜歡收錄在行爲準則和專業守則裡的具體方法。我認爲道德規範最好是在與軟體工程有關的文章中討論，不應該自成一個主題。因此本書雖然不包含抽象的道德探討，但是會適當的在習題中加入一些範例，在討論道德議題時可由這些範例開始著手。

摘要

- **軟體工程**是一門著重在生產軟體各方面知識的工程學科。
- 軟體產品包含開發完成的程式以及相關的說明文件。主要的產品特性有：**可維護性**、**可信賴度**、**效率**以及**可用性**。

- **軟體程序**包含與開發軟體產品有關的各項活動。所有的軟體程序都包含如軟體規格制定、開發、確認以及演進等高階活動。
- 生產軟體必須透過有系統、有組織的方法。這些方法包括提出後續的程序、即將使用的符號表示法、系統模型和掌管模型的規則說明與設計指南等。
- CASE **工具**是一組軟體系統，設計用來支援軟體程序中常見的活動，例如編輯設計圖、檢查圖表的一致性，以及追蹤程式的測試結果等。
- 軟體工程師必須對其專業和社會負責，不應該只著重於技術性的問題。
- 某些專業團體會發表一些**行爲準則**，作為其所屬會員的行為標準。

參考資料

Fundamentals of Software Engineering：這是一本探討軟體工程概論的教科書，它的看法與本書頗為不同（C. Ghezi, et. al., Prentice Hall, 2003）。

Software engineering: The state of the practice：這一期的 IEEE Software 期刊包含多篇探討軟體工程目前的實用經驗、變化以及如何應用新興軟體技術的論文（IEEE Software, 20 (6), November 2003）。

Software Engineering: An Engineering Approach：軟體工程概論教科書，書中包括許多有用的案例研討（J. F. Peters and W. Pedrycz, 2000, John Wiley & Sons）。

Professional Issues in Software Engineering：這是一本討論法律、專業議題以及專業道德很好的書籍，我個人比較喜歡它的實用觀點（F. Bott, et al., 3rd edition, 2000, Taylor & Francis）。

Software Engineering Code of Ethics is approved：這是一篇討論 ACM/IEEE 道德規範發展背景的文章，文中包含簡短版與完整版的兩種規範（Comm. ACM, D. Gotterbarn, et al., October 1999）。

No silver bullet: Essence and accidents of software engineering：這篇文章雖然很舊了，但是它對軟體工程的問題而言是很好的一般性導論。文中表達的主

要訊息到目前為止並沒有改變（F. P. Brooks, IEEE Computer, 20 (4), April 1987）。

學習評量

複習題

1. 何謂軟體危機？
2. 軟體產品可分成哪兩種基本類型？
3. 何謂軟體工程？
4. 軟體程序中有哪些基本的程序活動？
5. 軟體開發模式可分成哪 3 大類？
6. 軟體工程方法是由哪些主要的元件所組成？
7. CASE 這個縮寫字代表的意義為何？
8. 為什麼可維護性（maintainability）是良好軟體的必要特性之一？
9. 軟體工程面臨的 3 個主要挑戰為何？
10. 何謂軟體工程道德準則？

問題與練習

1. 請參考 1.1.6 節中的軟體成本分佈，並且說明為何最好將軟體（software）視為不只是系統終端使用者所執行的一些程式（program）而已。
2. 請說明通用產品與客製化產品兩者之間的差異。
3. 所有的軟體產品都必須有哪 4 項重要特性？並請另外提出其他 4 項有時候也很重要的特性。

4. 軟體程序模型與軟體程序有何不同？請提出兩種能幫助軟體程序模型找出如何改善程序的方法。

5. 請說明在廣大市場銷售的通用軟體產品，為何其系統測試成本特別高。

6. 只有透過 CASE 技術的支援，軟體工程的方法才能被廣泛使用。請提出 5 種可以由 CASE 工具提供的支援方法。

7. 軟體工程除了異質性、開發時間與信任度的挑戰之外，請再提出它在 21 世紀會面臨的其他問題和挑戰。

8. 請探討專業工程師是否應該和醫師或律師一樣經過認證。

9. 請根據圖 1.6 的 ACM/IEEE 道德準則中的每一個條款，舉出可以說明這些條款的適當範例。

10. 為了反擊恐怖主義，很多國家正在計畫開發能夠記錄大量的市民與其活動的電腦系統。顯然這樣做會觸及隱私權的爭議，請闡述開發這類系統的道德問題。