Chapter 4

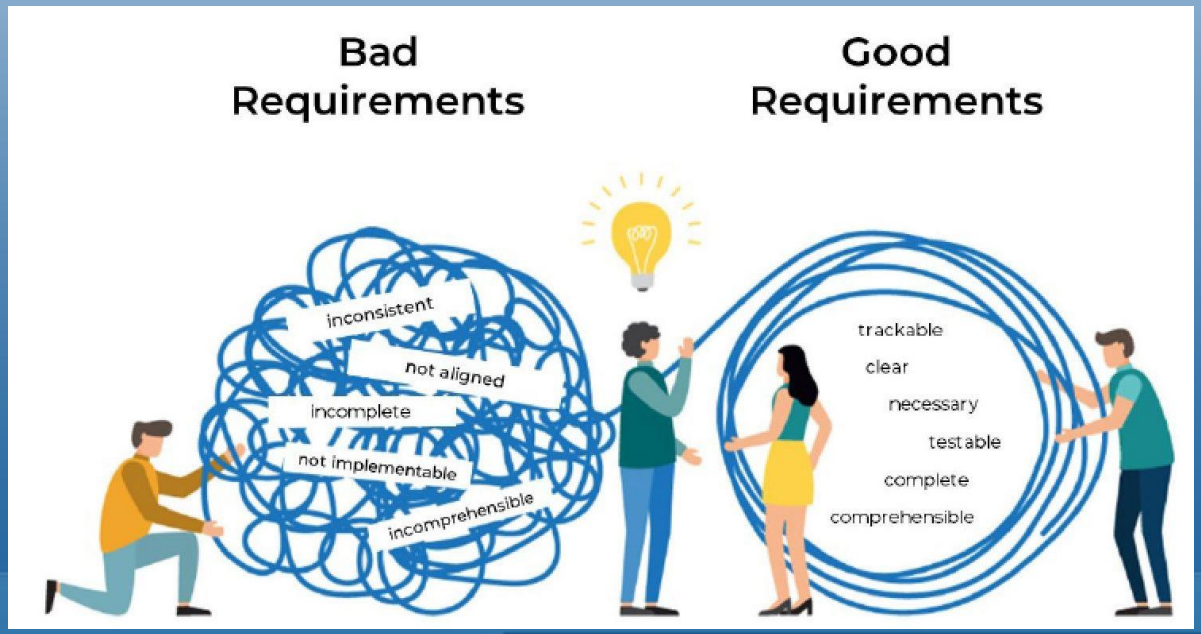
需求工程

本章內容

4.1 功能需求與非功能需求
4.2 需求工程程序
4.3 需求抽取
4.4 需求規格制訂
4.5 需求確認
4.6 需求變更

需求工程

- 系統需求與需求工程的定義
  - 系統需求是描述系統應該提供的服務，以及它的運作限制(Constraint)
  - 需求工程(Requirements Engineering, RE)則是找出、分析、規格說明、驗證、和管理這些軟體需求服務與限制的過程。



需求工程

- 一般而言，軟體需求工程的內容如下：
  1. 了解用戶對軟體的期待。
  2. 描述功能性軟體需求及非功能性軟體需求。
  3. 描述主要的軟體需求工程中各項子任務。
  4. 使用軟體需求引出及分析所需之技術。
  5. 組織整體的軟體需求，並在軟體需求文件中表達。

系統需求與需求工程的差異

- Davis (Davis 1993) 曾說明為何這兩者會有這樣的差異存在：

> 假如公司希望將大型軟體開發專案外包，它必須以夠抽象的方式來定義需求，不能事先定義解決方案。訂出的需求必須能夠讓多家承包商參與競標，以便比較符合客戶公司需要的各種不同解決方案。得標的承包商必須為客戶撰寫更詳細的系統定義，讓客戶瞭解並進而可以確認軟體所提供的功能。這兩種文件都可以稱為系統的「需求文件」。

- 使用者需求 (user requirement)
- 系統需求 (system requirement)

1. 是一些以自然語言加上圖表所形成的敘述，用來描述系統預期提供的服務，以及運作時的限制。
2. 更詳細的定義軟體系統的功能、服務與運作限制。也稱為「功能規格」(functional specification)，它可能會成為系統採購單位與軟體開發公司之間合約的一部分。

使用者需求定義

1. Mentcare 系統每個月必須產出每個心理診所開出的處方藥品成本的月管理報表。

系統需求規格

1.1 每個月的最後一個工作天，系統必須產出一份列出每種處方藥品、它們的成本，以及開此藥品的診所名稱的摘要。

1.2 每個月最後一個工作天的下午 17.30，系統必須自動產生並列印此份月報表。

1.3 針對每個診所產生一份列出每種藥品名稱、處方總數、處方總劑量，以及該藥品總成本的報表。

1.4 假如藥品提供數種不同劑量（如 10 mg、20 mg），則針對個別劑量應建立分開的報表。

1.5 所有與成本相關的報表，只限有被列入管理存取控制清單中有取得授權的使用者方可存取。

圖 4.1　使用者需求和系統需求

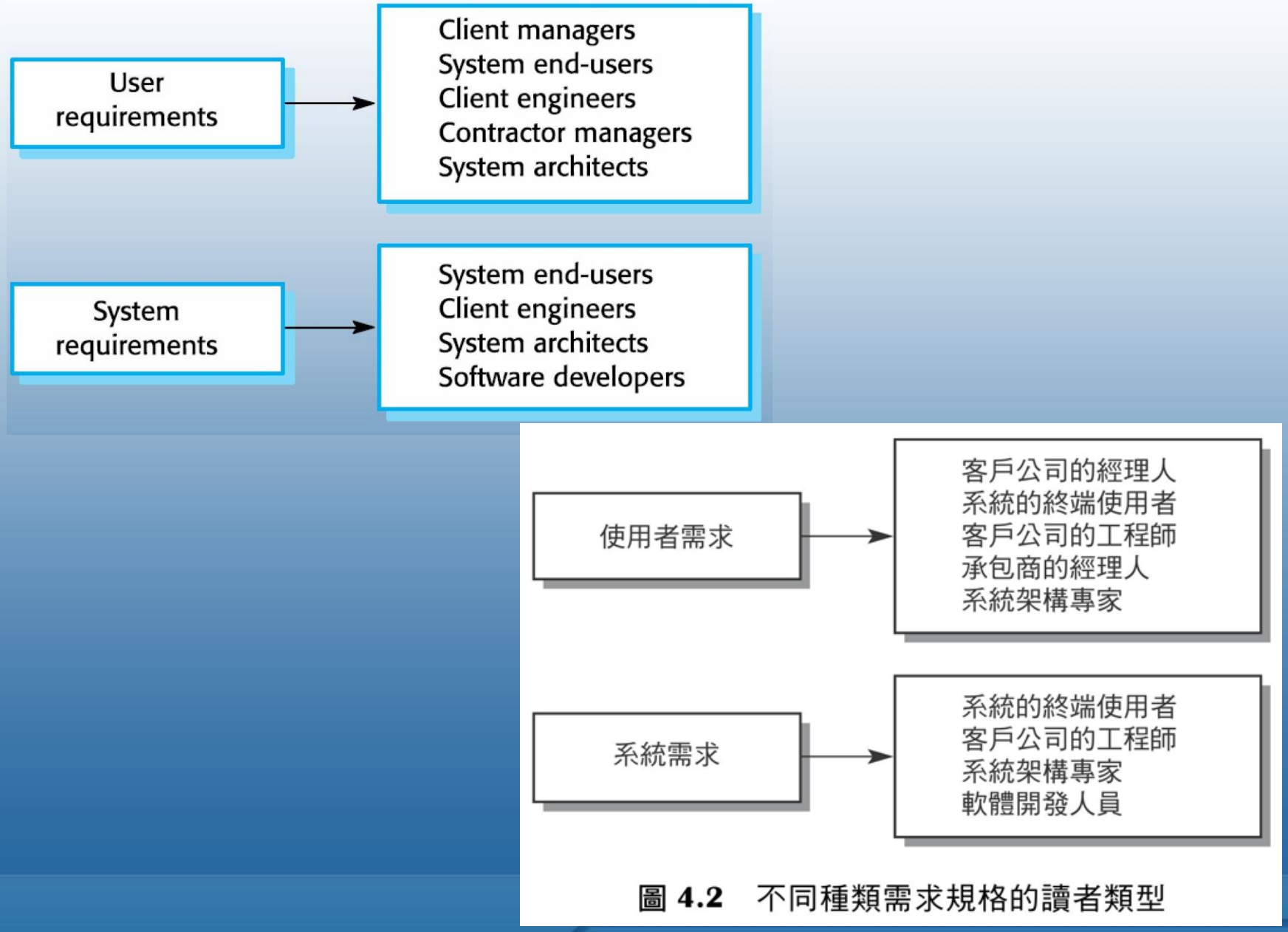


圖 4.2　不同種類需求規格的讀者類型

- 系統關係人(Stakeholder)指的是會被系統以某種方式所影響，或是對系統有合法利益的人。包括從系統的終端使用者一直到經理人，還有外部的關係人如主管機關(負責認證系統的可接受度)等。
- 例如Mentcare系統的系統關係人會包括：
  1. **病患和家屬**：系統中紀錄的是他們的資訊。
  2. **醫生**：負責評估病情並進行治療。
  3. **護理人員**：協調各位醫師的會診並協助某些治療。
  4. **醫療行政人員**：管理病人的預約資料。
  5. **IT人員**：負責架設和維護系統。
  6. **醫療行為主管機關**：確保此系統符合目前的醫療道德原則。

RE Process: https://www.youtube.com/watch?v=_llqRnlrzWw

7. **醫療行政主管**：可從系統取得管理方面的資訊。
8. **醫療病歷管理人員**：負責確保系統資料的維護與保存，以及病歷保管程序有被確實遵守。

- 本章我是以「傳統」觀點來看需求，而不是第3章探討的敏捷式程序裡的需求。

4.1 功能需求與非功能需求

- 軟體系統的需求通常可以分成兩大類。
  1. **功能需求 (functional requirement)**：描述系統應該提供的服務、系統對特定輸入的反應，以及系統在特定情況下的行為等敘述。有些時候也會在功能需求中明確描述系統不應該做什麼。
  2. **非功能需求 (non-functional requirement)**：系統提供的服務或功能的限制。這包括時間上的限制、開發程序上的限制，以及由標準所衍生的限制。需求經常是套用在整個系統上。

4.1.1 功能需求(Functional Requirements)

- 描述系統應該做的事。依據軟體類型、該軟體的預期使用者，以及使用單位所採取的一般做法而定。
- 系統需求層次是把使用者需求擴充，而且應該由系統開發人員來撰寫。他們應該仔細描述系統的功能。

領域需求 (domain requirement) 是衍生自系統應用領域的需求，而不是來自系統使用者的指定要求。

4.1.1 功能需求

- 以Mentcare系統的功能需求為例：
  1. 使用者必須能夠搜尋全部診所的預約清單。
  2. 系統必須每天針對每個診所產生一份當天預約的病患名單。
  3. 使用系統的每位工作人員，都應該使用自己獨一無二的8位數員工編號以供識別。
- 需求規格的不精確可能導致客戶和開發團隊的爭執。系統開發人員在遇到這些模棱兩可的需求時，很自然的會朝向比較容易實作的方向來解讀。

4.1.1 功能需求

- 以Mentcare系統需求清單中的第一個需求為例。
- 醫療人員在敘述這項需求時，可能心裡想像的「搜尋」功能是只要輸入病患姓名，系統就會在全部診所的預約名單中搜尋。開發人員可能會以不同方式來解讀，將這個搜尋功能設計成使用者必須先選擇某家診所再開始搜尋。
- 系統的功能需求規格，理想上應該具備完整性與一致性。
  - 完整性 (completeness) 代表使用者要求的所有服務和資訊都應該要定義。
  - 一致性 (consistency) 則表示需求之間不能有相互矛盾的定義。
- 不過實際上，只有非常小型的軟體系統才有可能達到需求的完整性和一致性。

4.1.2 非功能需求

- 非功能需求通常是指定或限制整個系統的一些特性，它們可能與系統的外顯性質 (emergent property) 有關，例如可靠性、回應時間和儲存空間等。它們也可能用來定義系統實作上的限制，例如I/O裝置的功能或與其他系統的溝通介面所使用的資料表示方式。
- 非功能需求通常比個別的功能需求更重要。但如果是非功能需求不符合時，可能會讓整個系統無法使用。

4.1.2 非功能需求(Nonfunctional Requirement)

- 非功能需求的實作部分可能散佈在系統各處，原因有二：
  1. 通常會影響系統的整體架構，而不只是個別元件。比如說，為了確保嵌入式系統能符合效能需求，可能必須組織系統讓元件之間的通訊越少越好。
  2. 單一的非功能需求(例如保全需求)可能會產生出多個相關的功能需求，它們負責定義要實作這個非功能需求必備的新服務。此外，它也可能產生用來限制現有需求的新需求，例如限制對系統中資訊的存取。

4.1.2 非功能需求

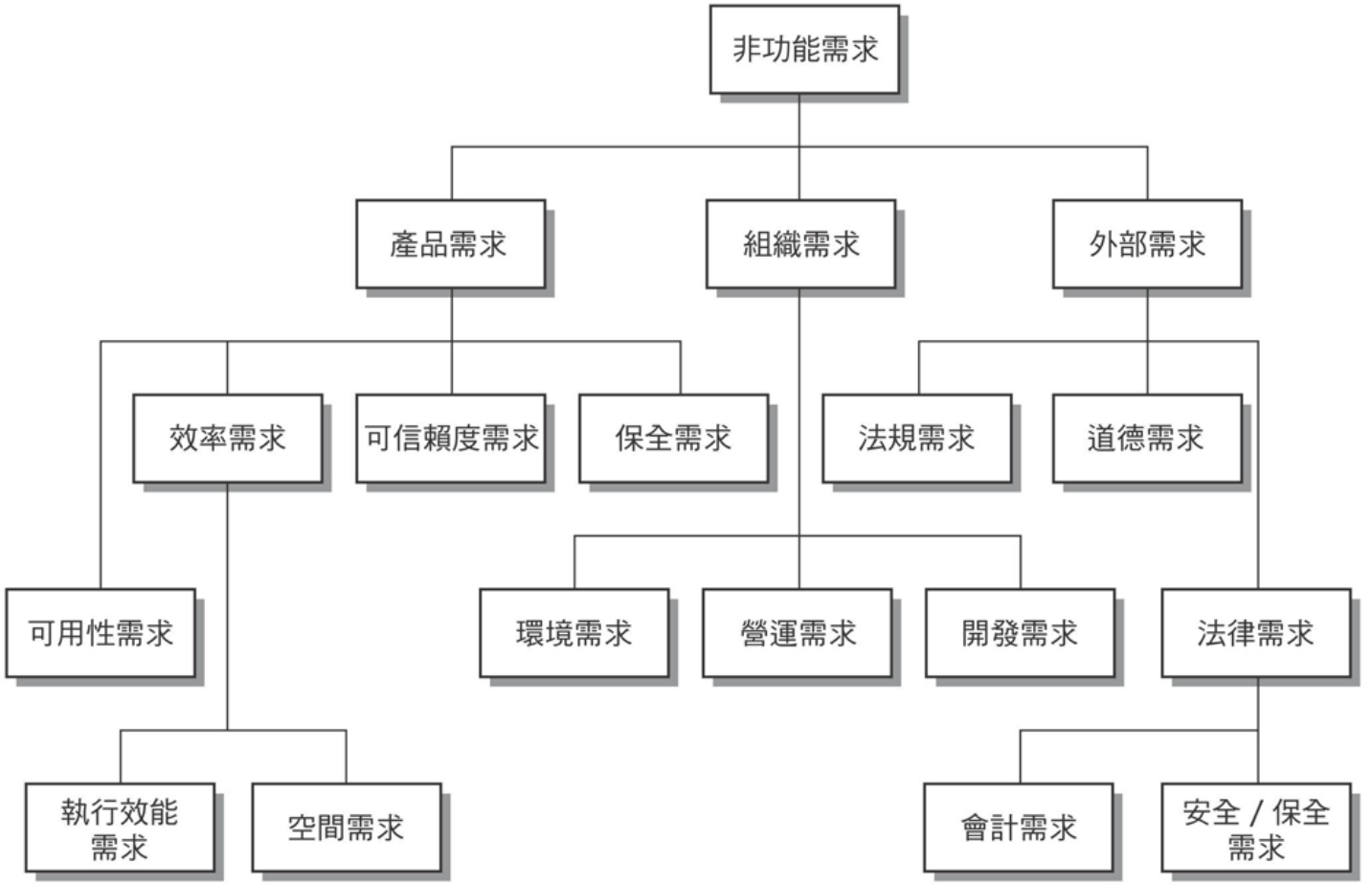


圖 4.3　非功能需求的類型

4.1.2 非功能需求

1. **產品需求 (product requirement)**：這類需求是指定或規範軟體在執行期間的行為。例如執行效能需求、可靠度需求。
2. **組織需求 (organizational requirement)**：這類需求是廣泛的系統需求，衍生自客戶公司和軟體開發公司的組織政策和程序。如何使用的營運程序需求、開發環境或程序標準的開發程序需求。
3. **外部需求 (external requirement)**：這類需求包括系統與其開發程序之外的其他因素所衍生的所有需求。確保系統合法運作的法律需求。

4.1.2 非功能需求

產品需求

Mentcare 系統在一般上班時間（星期一至五，08.30–17.30），應該讓所有診所都能使用；發生在一般上班時間的停機狀況，時間不能超過一天 5 秒鐘。

組織需求

Mentcare 系統的使用者，應提供他們的衛生機關識別卡讓系統驗證身分。

外部需求

系統應實作 HStan-03-2006-priv 所規定的病患隱私條款。

4.4 Mentcare 系統可能的非功能需求範例

4.1.2 非功能需求

- 非功能需求的共同問題是它們有時太過籠統。
- 這種模糊的目標會造成系統開發者的問題。
- 經理人在表達使用方便性的需求時可能會列出的系統目標敘述：
  - 此系統應該讓醫療人員很容易使用，並且其架構方式應該盡量減少使用者發生錯誤的可能性。
- 下面我重寫這個系統目標敘述，展示如何將目標敘述表達成「可測試」的非功能需求。
  - 醫療人員在經過2小時的訓練後，就應該能使用所有的系統功能。並且有經驗的使用者在經過訓練後，每小時內可能發生的錯誤個數平均不得超過2次。

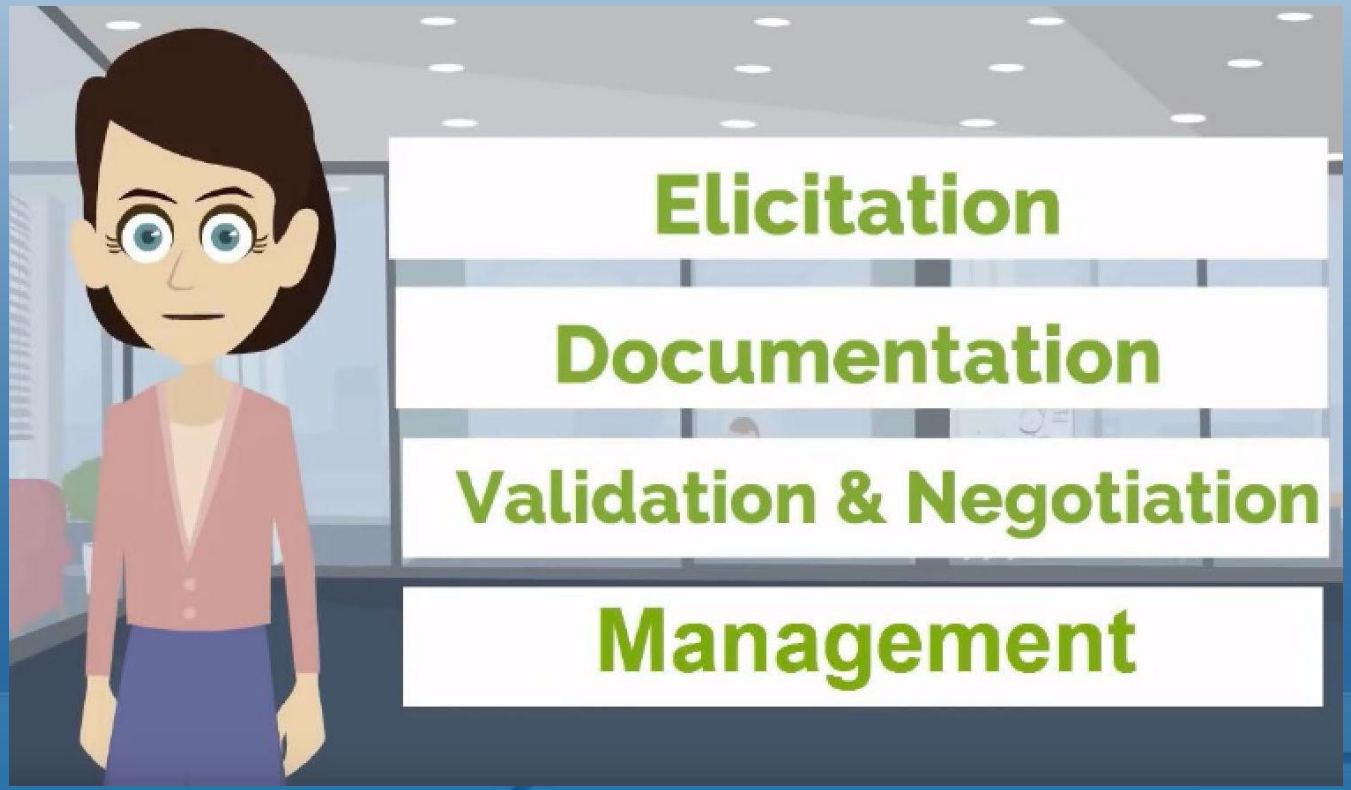
4.1.2 非功能需求

特性	測量指標
速度	已處理的異動個數 / 秒 使用者 / 事件回應時間 畫面重新整理時間
大小	百萬個位元組 (Megabytes) ROM 晶片的數量
容易使用	訓練時間 輔助說明視窗的數量
可靠性	平均的故障間隔時間 無法使用的機率 故障出現率 可用性
穩固性	故障後重新開機所需的時間 造成故障的事件比例 故障時的資料毀損機率
可移植性	與目標系統相依的敘述個數百分比 目標系統的個數

圖 4.5 可用來設定非功能需求的測量指標

4.2 需求工程程序

- 需求工程程序 (requirements engineering process) 是由3個關鍵活動所組成，關係人互動找出需求 (抽取與分析)、將需求轉換成某種標準格式 (規格制訂)，以及檢查需求真的有成功定義出客戶所要的系統 (確認) 等。



4.2 需求工程程序

- 圖顯示了這個交錯進行的關係。這裡的活動是組織成一個環繞螺旋狀的反覆式程序，RE程序的輸出是一份系統需求文件。

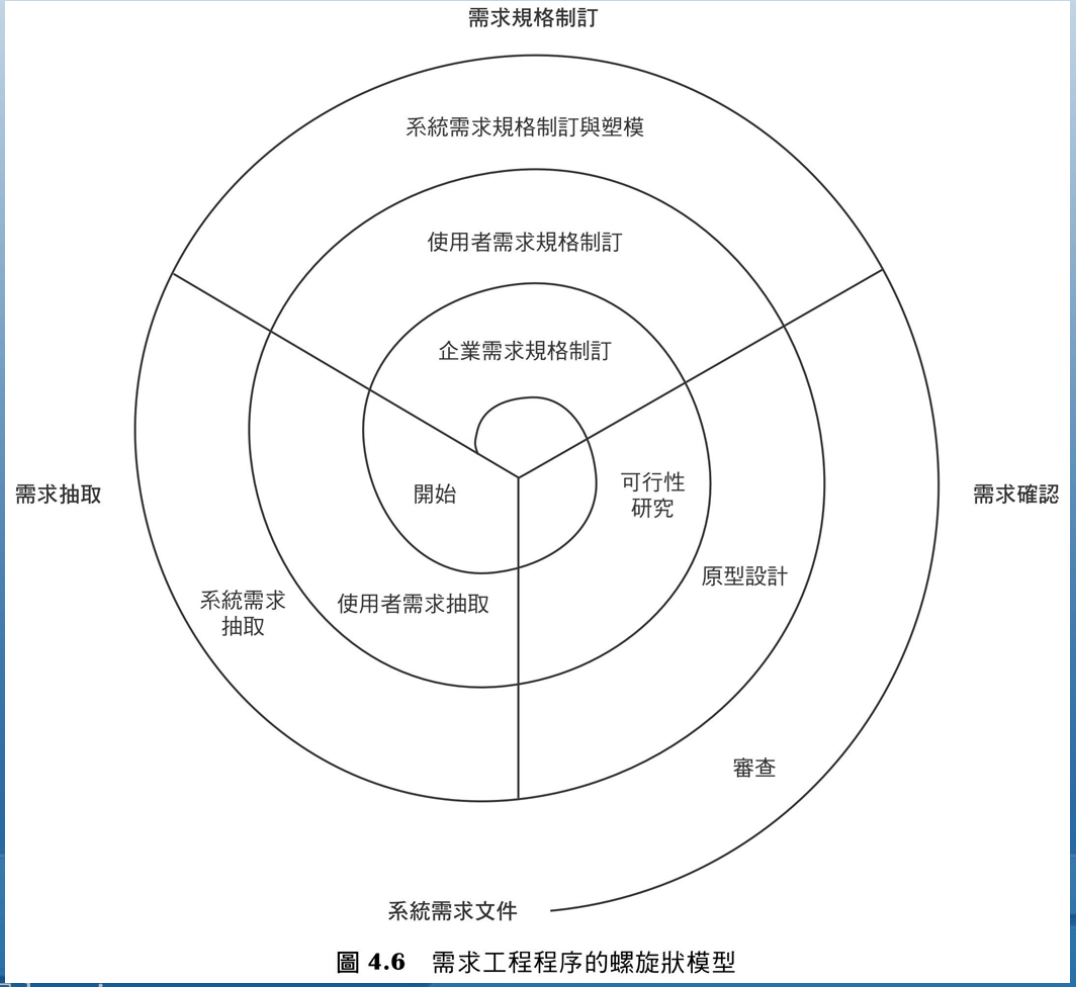


圖 4.6 需求工程程序的螺旋狀模型

4.3 需求抽取

- 需求抽取 (requirements elicitation) 程序的目標是瞭解關係人的工作內容，以及他們可能會如何使用新系統來輔助工作。

- 下面這幾個原因造成要抽取和瞭解系統關係人的需求並不容易：
  1. 關係人經常不知道可以要求電腦系統為他們做什麼事。
  2. 關係人很自然的會使用自己的術語和工作上的知識來表達需求。
  3. 不同的關係人有不同的需求。
  4. 政治因素可能會影響系統的需求。
  5. 被分析的經濟和企業環境是動態的。新關係人也可能會提出新的需求。

4.3 需求抽取

- 這個程序的活動如下：

1. **發現與瞭解需求 (requirements discovery and understanding)**：與系統的關係人進行互動，收集他們的需求。也會找出關係人的領域需求與相關文件。
2. **需求分類和組織 (requirements classification and organization)**：此活動將雜亂無章的需求依照相關性分組，組織成幾組關係密切的叢集。
3. **排列需求優先順序與協調 (requirements prioritization and negotiation)**：需求安排優先順序，並發現衝突再經由協調來解決這些衝突。
4. **製作需求文件 (requirements documentation)**：將需求記錄成文件，當作螺旋狀下一圈的輸入。

4.3 需求抽取

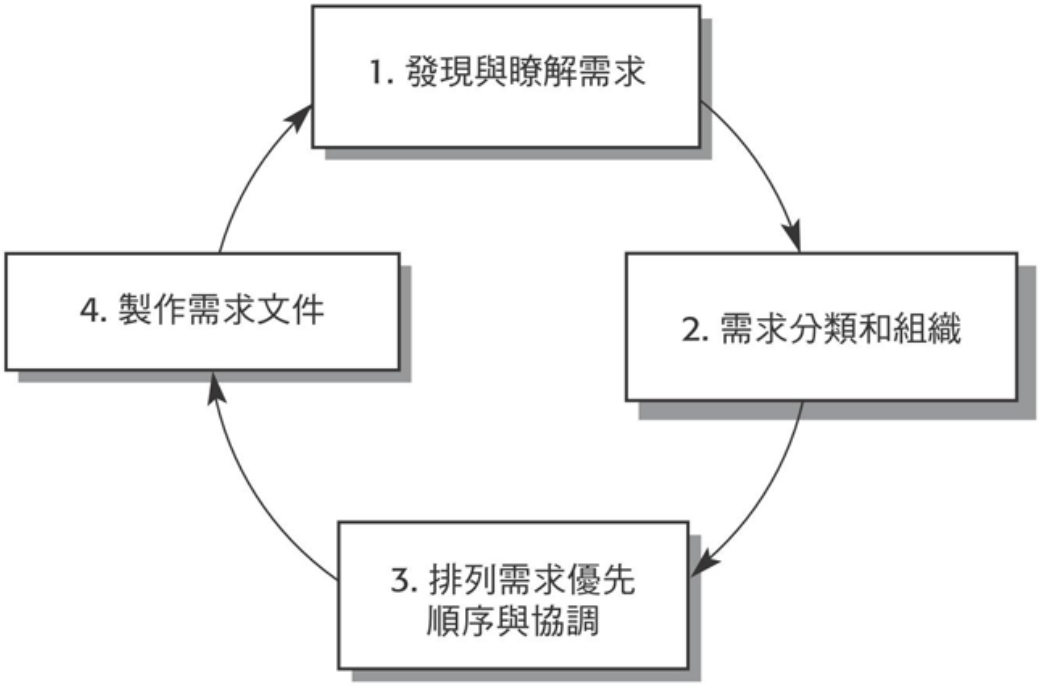


圖 4.7 需求抽取與分析程序

- 整個程序循環是從發現需求開始，一直到製作需求文件後結束。

4.3.1 需求抽取技術

- 需求抽取階段是與不同類型的關係人開會，找出有關提案系統的資訊。

4.3.1.1 訪談(Interviewing)

- 與系統關係人的正式或非正式訪談，大部分需求工程程序都含有這部分。
- 訪談可能的形式有兩種：
  1. **限定的訪談 (closed interview)**：關係人所回答的是一些事先定義好的問題。
  2. **開放式訪談 (open interview)**：沒有事先定義問題和議程。需求工程團隊會與關係人一起討論某些議題，因而更瞭解關係人的需求。

- 在實務上，與關係人的訪談通常是兩種方式合併進行。

4.3.1.1 訪談

- 在訪談過程不容易抽取出領域知識，其原因有二：
  1. 應用領域專家很自然的會使用他們那一行的術語和行話。這很容易讓需求工程師誤解。
  2. 有些領域知識對於關係人而言太熟悉了，因此他們可能會覺得很難解釋，或者覺得這是基本常識根本不值得一提。
- 想成為有效率的訪談人員，要牢記兩件事：
  1. 他們心胸開放，對於需求儘量避免有先入為主的心態，並且願意傾聽關係人的陳述。
  2. 他們會提示受訪者從某個問題或需求提案開始討論，或者建議在某個原型系統上一起工作。

4.3.1.2 民族誌法

- 民族誌法 (ethnography) 是一種觀察的技術，可用來瞭解作業程序，並協助產生出能支援這些程序的軟體需求。分析人員必須將自己融入在未來將使用此系統的工作環境中，觀察與記錄日常的實際工作。
- Suchman (Suchman 1983) 率先使用民族誌法來研究辦公室的工作生態。她發現實際工作的情形，比辦公室自動化系統所假設的簡單模型更豐富、更複雜也更動態得多。

4.3.1.2 民族誌法

- 民族誌法可以和系統原型的開發加以合併 。
- 民族誌法對於瞭解現有系統有幫助，但這些瞭解未必有助於創新。

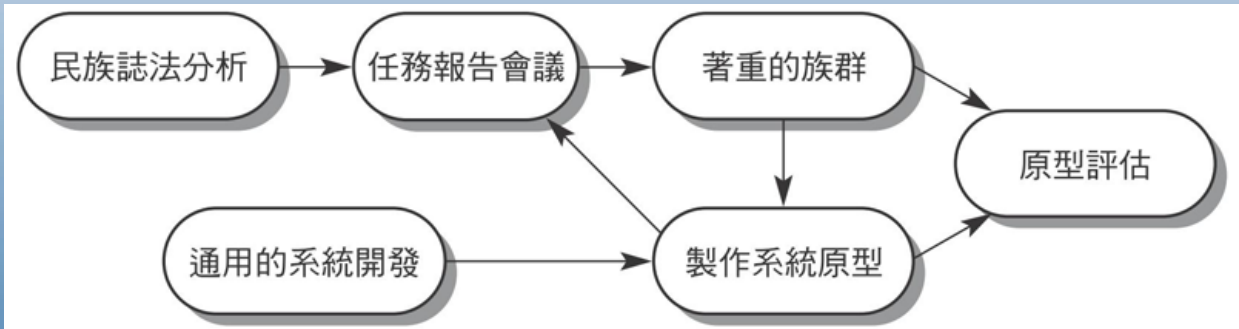


圖 4.8 使用民族誌法和原型法進行需求分析

- 使用民族誌法可發現一些使用其他需求抽取技術通常不會發現的關鍵程序的詳細資訊。不過，因為民族誌法只針對終端使用者，所以不適合用來尋找範圍較廣的組織或應用領域的需求，或者建議應該增加的創新功能。

4.3.2 故事和情境

- 人們通常會發現透過真實生活的例子會比抽象描述容易瞭解。
- 故事和情境在本質上是相同的東西，它們都是描述如何使用系統來完成某件工作。
- 用故事來描述的好處是大家都能很容易聯想。
- 情境最好是以結構化方式表達，而不要用文章敘述。

4.3.2 故事和情境(Stories and Scenarios)

課堂上的照片分享

Jack 是蘇格蘭北部 Ullapool 村莊的小學教師。他決定要讓學生進行以當地漁業為主題的作業，研究它的歷史、發展過程和經濟影響。他要求學生訪問親友收集回憶往事，查閱舊報紙封存檔案，還有收集與當地漁業和漁村相關的老照片。學生使用 iLearn 的維基網站 (wiki) 收集漁業故事，還有用 SCRAN（一個歷史資源網站）取得報紙封存檔和照片。不過 Jack 還需要一個相片分享網站，因為他要求學生如果找到舊照片要翻拍上傳，分享給大家並一起評論。

Jack 送郵件到小學教師的群組（他也是成員之一），詢問是否有合適的系統請大家推薦。有兩位教師回應，兩人都推薦他使用 KidsTakePics 這個相片分享網站，它能讓教師事先檢查和審核內容。因為 KidsTakePics 並未與 iLearn 驗證服務整合，所以他建立一個教師帳號和一個班級帳號。他使用 iLearn 設定服務將 KidsTakePics 加入他班級的學生能看到的服務之一。這樣一來他班上的學生一登入，就能馬上使用這系統，把原來在行動裝置或教室電腦的相片上傳過去。

圖 4.9　iLearn 系統的使用者故事

4.3.2 故事和情境

- 一個情境可能會包含：
  1. 情境一開始時系統和使用者所預期的情況描述。
  2. 情境中事件的正常流程描述。
  3. 可能發生的問題及解決方法的描述。
  4. 同一時間可能發生的其他活動的相關資訊。
  5. 情境完成時的系統狀態描述。

4.3.2 故事和情境

上傳相片到 KidsTakePics

初始假設：假設有一或多位使用者要上傳一或多張數位相片到相片分享網站。這些相片原本是儲存在平板或筆電上。使用者已經成功登入 KidsTakePics。

正常情況：使用者選擇要上傳相片，系統提示先在電腦上選取準備要上傳的相片，然後選取作業名稱，之後會將相片儲存在裡面。每張上傳的相片使用者都可選擇輸入與它有關的關鍵字。上傳的相片它的檔名會用使用者名稱加上相片原本在電腦上的檔名組合而成。

上傳完成後，系統會自動傳送電子郵件給作業的審核者，通知有新的內容等待審查。審核後會以螢幕訊息通知使用者審核完成。

可能發生的問題：被選取的作業沒有指定審核者。這時會自動產生一封電子郵件傳送給學校行政人員，要求他們指派一個審核者。同時使用者應該要收到他們的相片可能要延遲一些時間才能上線的通知。

假如同一位使用者名下有同名稱的相片已經上傳過，應該詢問使用者是要用相同名稱再次上傳相片、重新命名相片，或者取消上傳動作。如果使用者的決定是再次上傳，原本的相片就會被新上傳的覆蓋。假如決定重新命名相片，就會用原來的舊檔名後面附加一個數字編號做為新檔名。

其他動作：審核者可能會登入系統，審查目前上傳過來的相片決定是否核准。

完成時的系統狀態：使用者已經登入。選好的相片已經上傳完成，並指派狀態為「等待審核」。這些相片目前只有審核者和上傳相片的人看得到。

圖 4.10 上傳相片到 KidsTakePics 的情境描述

4.4 需求規格制訂

- 需求規格制訂是指把使用者和系統需求撰寫成需求文件的過程。
- 系統的使用者需求應該描述功能需求與非功能需求兩者，以便能夠讓沒有專業技術背景的系統使用者也能瞭解這些需求。
- 它也可以當作承包系統實作合約的一部分，因此應該是完整的，而且與整個系統的細部規格一致。

4.4 需求規格制訂 (Requirements Specification)

表示法	說明
自然語言敘述句	將需求以自然語言寫成加上編號的敘述句，每個敘述句表達一個需求
結構化自然語言	以自然語言將需求填寫在標準的表單或範本上，每個欄位提供需求的某方面資訊
圖形表示法	加上文字標記的圖形化模型，用來定義系統的功能需求。最常用的是 UML 使用案例和序列圖
數學式規格	它是以數學觀念（如有限狀態機或集合論）為主的符號表示法。這類規格的表達一絲不苟，可以減少客戶和承包商對系統功能的爭執。不過，大部分客戶都不瞭解這些正規化的規格，所以不願意使用它當作系統的合約（參見第 10 章）

圖 4.11　撰寫系統需求規格的各種表示法

4.4 需求規格制訂

- 若要完整的描述一個複雜的軟體系統，需要某種程度的詳細資訊，因此不太可能也不應該排除所有的設計資訊。原因如下：
  1. 系統的初始架構定義可以用來幫助建構需求規格，而系統的需求則會根據它的各個子系統來組織。
  2. 大部分的情況下，系統都必須與其他現有系統相互溝通。
  3. 為了符合非功能需求，可能會需要使用某種架構。

4.4.1 自然語言規格

- 從1950年代開始，自然語言就一直被用來撰寫軟體的需求。它的好處是表達容易、直覺而且通用，但是缺點是容易模糊不清、模稜兩可，而且它的意義可能因解讀者的背景而異。
- 減少誤解，下面建議幾項簡單的原則：

1. 創造一個標準的格式，並確定所有的需求定義都遵照這個格式。
2. 語言的使用要一致。例如把強制和期望的需求區分清楚。
3. 利用字型功能(粗體、斜體或色彩)強調需求的重要部分。
4. 因此應該儘量避免使用電腦術語、簡稱或縮寫字等。
5. 應該儘可能嘗試將每個使用者需求記錄一份理由敘述，裡面解釋為何要納入這項需求，還有是誰提出的。

4.4.1 自然語言規格

3.2 系統必須每 10 分鐘測量一次血糖，必要時注射胰島素。(血糖值的變化比較慢，所以不需要比 10 分鐘更密集的測量，但是也不能間隔更長，否則可能導致血糖值飆高。)

3.6 系統必須每分鐘執行一次自我檢測程序，測試表 1 中定義的各種條件和相關聯的行動。(自我檢測程序可以發現硬體和軟體的問題，並警示使用者系統運作會變得不正常。)

圖 4.12　胰島素給藥系統的軟體規格範例

4.4.2 使用結構化語言撰寫規格

- 結構化自然語言是一種撰寫系統需求的方式，它是一種形式有限制的自然語言，而且所有的需求都是以標準方式來撰寫。
- 使用結構化方式來描述系統需求時，必須先定義一或多個可以表達需求的標準範本 (Template)，並將這些範本表達成結構化表單。

4.4.2 使用結構化語言撰寫規格

Insulin Pump/Control Software/SRS/3.3.2

功能	計算胰島素劑量：安全的血糖值。
說明	當目前測量到的血糖值在安全範圍內（3 至 7 單位）時，計算應該注射的胰島素劑量。
輸入	目前的血糖讀數 (r2)，之前兩次的血糖讀數 (r0 和 r1)。
來源	目前的血糖讀數從感應器讀取，其餘從記憶體中讀取。
輸出	CompDose：注射的胰島素劑量。
目標	主要控制迴圈。
行動	假如血糖值是穩定或下滑，或者增加的幅度逐漸降低，則 CompDose 的值等於零。假如血糖值正在增加，而且增加的幅度越來越快，則 CompDose 的值等於前一次的血糖讀數減去目前的血糖讀數的結果除以 4，再四捨五入。假如四捨五入後的結果等於零，則 CompDose 的值設定為最小劑量。
必要條件	前兩次的血糖讀數，這樣才可以算出血糖值變化的速率。
前置條件	胰島素儲存區中至少要有單次最大劑量的存量。
後續條件	將 r1 的值取代 r0，然後用 r2 的值取代 r1。
副作用	無。

圖 4.13 胰島素給藥系統某項需求的結構化規格範例

4.4.2 使用結構化語言撰寫規格

- 當使用標準的表單來描述功能需求時，應該包含下列這些資訊：
  1. 欲指定的功能或實體的描述。
  2. 系統的輸入和輸入來源的描述。
  3. 系統的輸出和輸出去向的描述。
  4. 計算動作所需的資訊或會使用到的系統其他實體（「必要條件」部分）。
  5. 描述所採取的行動。
  6. 假如是採用功能性方法，則在前置條件處規定呼叫此功能之前必須為真的條件，並且在後續條件處設定呼叫此功能之後必須為真的條件。
  7. 描述運算動作可能的副作用（如果有的話）。

4.4.2 使用結構化語言撰寫規格

- 規格書內容還是會有一些模糊不清的地方。
- 針對這種問題，你可以使用表格或圖形模型，替自然語言需求補充其他資訊。
- 當可能的情況有好幾種，而你必須針對每一種描述要採取什麼行動時，表格方式會特別好用。例如注射的胰島素劑量是根據血糖值的變化速率計算出來的，而變化的速率是利用前兩次的血糖讀數計算而得。

4.4.2 使用結構化語言撰寫規格

條件	動作
血糖值下滑 (r2 < r1)	CompDose = 0
血糖值穩定 (r2 = r1)	CompDose = 0
血糖值正在增加，且增加的幅度逐漸降低 ((r2 – r1) < (r1 – r0))	CompDose = 0
血糖值正在增加，而且增加的幅度穩定或越來越快 r2 > r1 & ((r2 – r1) ≧ (r1 – r0))	CompDose = round ((r2 – r1)/4) If 四捨五入後結果 = 0 then CompDose = MinimumDose

圖 4.14 描述胰島素劑量計算公式的表格式規格

4.4.3 使用案例

- 使用案例 (use case) 是一種以圖形模型和結構化文字，描述使用者與系統間互動的方式。
- 現在已經變成UML符號表示法的基本功能之一。
- 從每個使用案例可以辨識出互動的類型和互動中的行動者 (actor)。
- 使用案例的文件紀錄是使用高階的使用案例圖，裡面包含一組使用案例集合，記錄將在系統需求中描述的所有可能的互動動作。
- 每個使用案例都應該用文字敘述加以記錄，然後連結到UML的其他模型，來獲得關於此情境的詳細資訊。

4.4.3 使用案例
(Use cases for the Mentcaresystem)

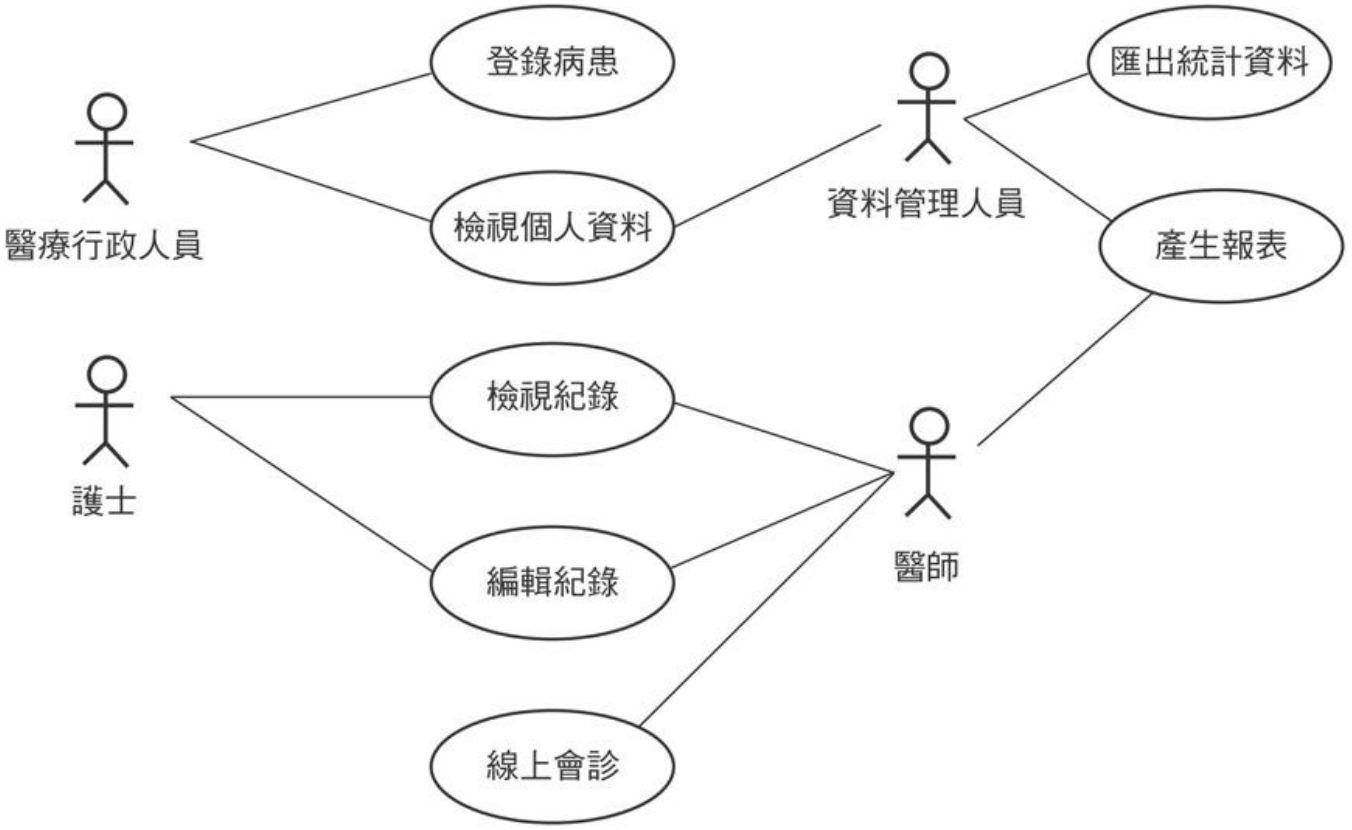


圖 4.15 Mentcare 系統的使用案例

4.4.4 軟體需求文件

- 軟體需求文件 (software requirements document) 有時候也稱為軟體需求規格書 (software requirements specification, SRS)。它是一份用來說明系統開發者應該實作什麼功能的正式文件，其內容可能涵蓋系統的使用者需求，以及系統需求的詳細規格。

4.4.4 軟體需求文件

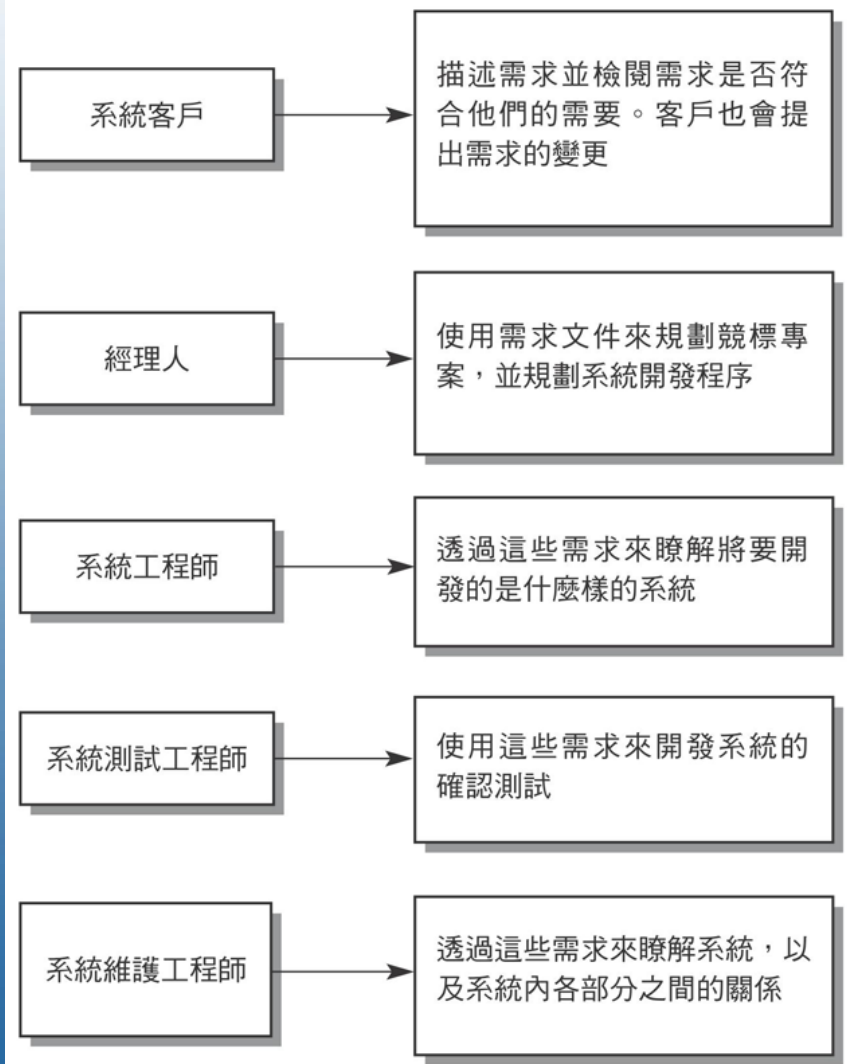


圖 4.16　需求文件的使用者類型

4.4.4 軟體需求文件

- 需求文件的詳細程度，是依照系統類型與使用的開發程序而定。
- 圖展示需求文件的一種可能的結構，這是根據IEEE的需求文件標準 (IEEE 1998)。這個標準是一種通用標準，可調整用在各種特定的用途上。

章節	說明
前言	在此定義此文件的適用對象並描述版本歷史，包括建立新版的原因和每一個版本的變更摘要
簡介	描述為何需要此系統，簡單描述它的功能，並說明如何與其他系統一起運作。另外，還應該描述系統如何與組織的商業或策略目標相配合
詞彙	定義此文件中會用到的術語，此處不應該預設讀者的經驗或專長
使用者需求定義	描述提供給使用者的服務與非功能系統需求。這裡的描述可使用自然語言、圖表或其他能讓客戶瞭解的表示法。如果必須遵守任何產品或程序標準，也應該在此規定
系統架構	簡單的整體介紹預期的系統架構，展示系統各模組的功能。再利用的架構元件則應該特別說明
系統需求規格	這裡應該更深入描述功能需求與非功能需求。必要時可針對非功能需求做更詳盡的說明。還可定義它與其他系統之間的介面
系統模型	建立圖形化的系統模型以展示系統元件之間、元件與系統，以及系統與環境之間的關係。這些模型可能是物件模型、資料流程模型或語意資料模型
系統演進	描述系統所根據的基本假設，以及預期因為硬體進步或使用者需求改變等因素所造成的改變。這章節對於系統設計人員很有用，因為它可協助避免下達對未來改變會產生束縛的決策
附錄	此處提供與開發中的應用程式有關的詳細而專門的資訊，比如說對某種硬體或資料庫的說明。例如硬體需求是定義系統最簡或最佳的組態，而資料庫需求則是定義資料的邏輯組織架構與資料間的關係
索引	文件的索引可能有好幾種，除了一般的字母索引外、可能還有圖表索引或功能索引等

圖 4.17　需求文件的結構

4.5 需求確認 (Requirements Validation)

- 需求確認 (requirement validation) 這個程序是負責檢查需求，確認它們是否真正定義出顧客想要的系統。
- 需求確認是非常重要的程序，因為如果需求文件有錯誤，等到開發期間或甚至系統上線時才發現的話，可能會導致很多工作要重做而浪費成本。
- 由於需求的問題而修改系統所花的成本，會比修改設計或程式撰寫錯誤所花的成本多很多。

4.5 需求確認

- 執行以下幾種檢查：
  1. **有效性檢查 (validity check)**：檢查需求是否真的反映出系統使用者真正的需要。
  2. **一致性檢查 (consistency check)**：在文件中的需求不能相互衝突。
  3. **完整性檢查 (completeness check)**：需求文件中定義的需求必須包含系統使用者想要的所有功能和限制。
  4. **實現性檢查 (realism check)**：利用對現有科技的瞭解，檢查需求是否能夠真正的被實作出來。
  5. **可驗證性 (verifiability)**：系統需求應該以能夠驗證的方式來撰寫。

4.5 需求確認

- 目前有多種需求確認技術，可以個別或聯合使用：

  1. **需求審查 (requirements review)**：
  2. **原型法 (prototyping)**：必須向終端使用者和客戶展示一個可執行的系統模型。確認是否符合他們的真正需求。
  3. **產生測試案例 (test-case generation)**：在撰寫程式碼之前，就先開發使用者需求的測試案例。

4.6 需求變更

- 大型軟體系統的需求會一直持續變動。
- 在軟體開發過程中，關係人對問題的理解會一直改變，接著這些改變一定會反應到系統需求上。

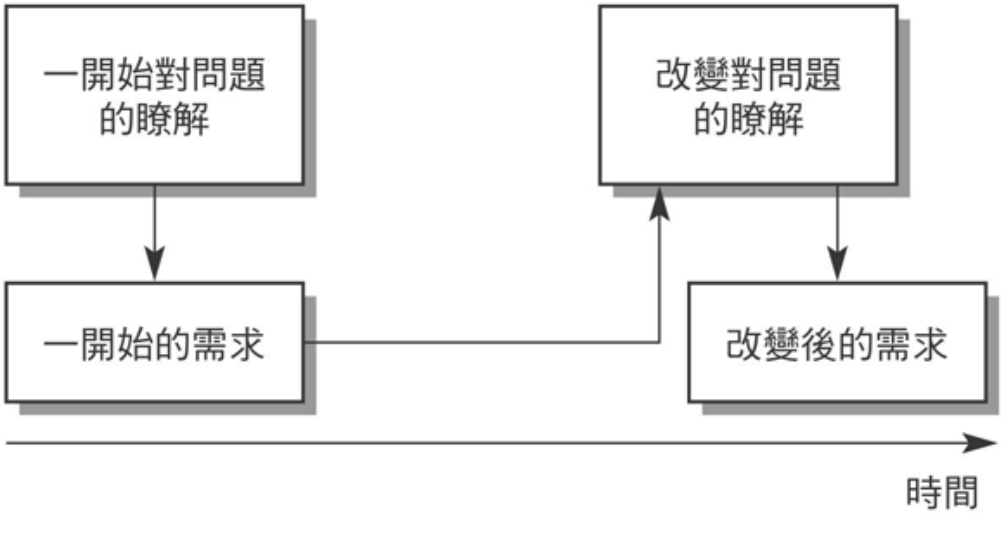


圖 4.18 需求的演進

4.6 需求變更

- 大部分對系統需求的變更經常是因為企業環境的變動引起的：
  1. 系統上線後因為企業和技術環境的改變，導致系統也必須跟著改變。例如加入新硬體或更新舊硬體時，系統可能必須與其他系統溝通。另外還可能出現系統必須遵守的新法律和新規定。
  2. 系統出錢的人和使用的人通常不是同一批人。採購系統時所提的需求大都受限於組織和預算的限制，但這些需求可能會與終端使用者的需求產生衝突。結果等到系統交付後，卻不得不加上新的功能，否則就無法符合系統目標。
  3. 大型系統通常有各種不同的關係人群組，而他們有不同的需求和優先順序，這些需求和優先順序可能會產生衝突或矛盾。最後的系統需求必須是大家的協商方案，而且根據經驗，提供給不同關係人的支援也會跟著改變，需求的優先順序也可能會調動。

4.6.1 需求管理的規劃

- 需求管理規劃 (planning) 階段著重在如何管理一整組持續演進的需求。必須決定下列幾個主題：

1. **識別需求**：每一項需求都必須能夠唯一的被識別，以便讓其他需求交叉參考並因此而評估可追蹤性。
2. **變更管理的程序**：這是一組用來評估變更的影響與成本的活動。
3. **建立可追蹤性的原則**：這些原則是用來定義需求之間的關係、需求與系統設計之間的關係，以及這些紀錄應該如何維護。
4. **工具的支援**：支援的工具種類從專門的需求管理系統，到共用的試算表和簡單的資料庫系統都有。

4.6.2 需求變更管理

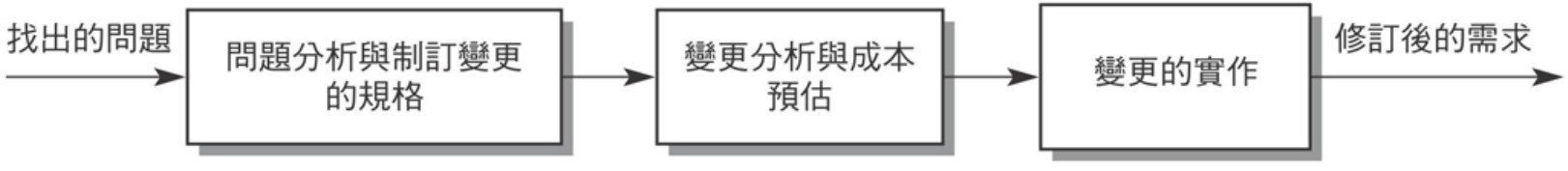


圖 4.19 需求變更管理

4.6.2 需求變更管理

- 變更管理程序有3個主要階段：
  1. **問題分析與制訂變更的規格**：這個階段會對問題或變更提案進行分析，檢查其有效性，分析結果會回應給變更的提案者，提案者可能會回覆更確定的需求變更提案，或者決定撤銷變更需求。
  2. **變更分析與成本預估**：這個階段會利用可追蹤性資訊和系統需求的知識來評估變更的影響。這些分析完成後，便決定是否繼續進行需求的變更。
  3. **變更的實作**：這個階段將修改需求文件，必要時連系統設計與實作部分都一起修改。若要讓文件容易變更，儘量模組化，使得個別章節可方便的變更和取代。