

書名

使用案例最佳實務-寫作指南、秘訣與範本

Writing Effective Use Cases

作者：Alistair Cockburn

英文版出版商：Addison-Wesley

繁體中文版譯者：趙光正

繁體中文版出版商：碁峰出版社

繁體中文版預計出版日期：2003/3

譯者介紹

趙光正，政治大學資管系碩士班畢業。目前從事物件導向顧問工作，曾經參與通訊、金融、醫療等方面的專案，同時也是點空間 www.dotspace.idv.tw 的版主之一。翻譯作品有 UML 精華第二版 (UML Distilled: A Brief Guide to the Standard Object Modeling Language 2nd)、Rational 統一流程第二版 (The Rational Unified Process: An Introduction 2nd)、Java 經典範例第二版 (Java Examples in a Nutshell: A Tutorial Companion to Java in a Nutshell 2nd) 與 UML 與樣式徹底研究 (Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process 2nd)。

封底

使用案例是用來捕捉軟體系統與企業流程中行爲需求的一種工具，而且這種做法很快就廣爲大家所接受了。它對專案規劃工作非常有幫助，我們可用它清楚展現出人們最後是如何使用被設計的系統。表面看來，使用案例只是一種直覺、簡單的概念。然而，一旦我們有需要寫出一整組的使用案例時，問題就來了：「該如何寫使用案例呢？」因爲使用案例本質上是一種散文，所以很難回答這個問題；也因爲它是散文，所以寫使用案例是一件很難的工作。

在*使用案例最佳實務*一書中，物件技術專家 Alistair Cockburn 以最新、最實際的寫作指引，教我們使用案例的寫法。作者依據他豐富的使用案例寫作經驗，延續傳統的使用案例寫法，提供我們一份巨細靡遺的使用案例寫作指南。書中完整涵蓋了入門、中級與進階的概念，提供我們各種不同程度、適當的知識。爲了強化教學效果，書中更分別示範了好的跟壞的使用案例範例。此外，本書還有許多對大家的學習很有幫助的習題（同時附上解答），以強調出書本中所提到的一些重要觀念。

本書重點包括：

- ◆ 完整討論使用案例中的主要組成元素 — 參與者、關係人、設計範圍、情節等。
- ◆ 提供使用案例寫作指南，其中包含寫作步驟以及建議大家採用的一些格式。
- ◆ 教我們節省使用案例寫作時間的大量秘訣。
- ◆ 以最有效的方式秀出使用案例的寫作範本，並附上一些評論，說明我們該何時或何地採用這些範本。
- ◆ 說明證實過有用、應用使用案例的方法論。

經由本書的教導，我們將學到如何寫出使用案例中的一些必要組成元素、改善我們的使用案例寫作技巧，讓大家可以下一個開發專案中，有效地應用使用案例。Alistair Cockburn 是著名的使用案例專家，他也是 Humans and Technology 公司的顧問之一，負責協助客戶成功完成物件導向專案。在保險、零售與電子商務公司以及一些大型組織（例如挪威央行與 IBM）中，他的軟硬體開發專案經驗已超過 20 年。同時，他也是 *Surviving Object-Oriented Projects* (Addison-Wesley, 1998) 一書的作者。

譯者簡介 趙光正 (kcchao@hotmail.com) 政治大學資管系碩士班畢業。目前從事物件導向顧問工作，曾經開發過醫療、金融方面的專案，也是 *UML 精華* 第二版、*Rational 統一流程* 第二版以及 *UML 與樣式徹底研究* 第二版等書的譯者。

封面內頁

使用案例的一些寫作提示

寫出具有可讀性的東西

使用案例縱然是採用非正式格式所寫的，只要具有可讀性，還是很有用的；另一方面，沒有人會去看不好閱讀的使用案例。

請先求廣度。先寫出低精確度的內容，再逐步寫出高精確度的內容。

精確等級 1：主要參與者名稱與其目標

精確等級 2：使用案例簡介或主要成功情節

精確等級 3：擴充情況

精確等級 4：擴充情況的處理步驟

針對每個動作步驟：

請寫出次一層的目標。

請捕捉參與者的意圖，而不要寫出使用者介面細節。

由參與者來負責傳遞資訊、證實條件成功或更新狀態。

在步驟間加上一些註釋，說明動作步驟的先後順序（或補充不足之處）。

問自己：「為何要這樣做」的理由，以找到更高一層的目標。

資料描述的精確度可分成下列三種等級（請只把精確等級 1 的資料描述寫在使用案例的說明文字中）：

精確等級 1：資料暱名

精確等級 2：跟資料暱名有關的資料欄位

精確等級 3：欄位型態、長度與有效性

圖示

設計範圍	目標等級
☐ 企業組織（黑箱）	⊙ 非常高階的目標摘要
☐ 企業組織（白箱）	∩ 目標摘要
▣ 系統（黑箱）	⋈ 使用者目標
▣ 系統（白箱）	⊗ 子功能
⊞ 元件	⊖ 非常低

針對目標等級，除了用圖來標示它之外，我們還可以在使用案例名稱之前用字元來標示它：

在目標摘要等級的使用案例名稱前面加上「+」

在使用者目標等級的使用案例名稱前面加上「！」或不加任何字元

在子功能等級的使用案例名稱前面加上「-」

使用案例的寫作步驟

1. 明確指出設計範圍與系統邊界的名稱。
用專案範圍內／外清單來追蹤最初情境圖的變動情形。
2. 用腦力激盪法列出主要參與者。
在系統的整個使用過程中，找出所有可能會有的人類或非人類主要參與者。
3. 用腦力激盪法窮舉出系統中的使用者目標。
這時候，我們將擁有最初的參與者 — 目標清單。
4. 找出最外層的目標摘要等級使用案例，以了解誰真的關心這個系統。
針對每個主要參與者，檢查看看哪個使用案例是他的最外層使用案例。
5. 重新討論並修正目標摘要等級的使用案例。我們可以新增、刪除或合併一些目標。
請再次檢查一下位於系統邊界、由時間所觸發的事件或其他事件。
6. 選擇一個使用案例，然後展開它。
可以的話，請練習寫出一段使用案例描述，學習如何寫出使用案例中的素材。
7. 找出使用案例的關係人與其利益、事先條件與事後保證。
系統在執行前必須先確認事先條件有成立，而且對關係人的利益要做出事後保證。
8. 寫出使用案例的主要成功情節（MSS）。
經過寫作步驟3-9之後，主要成功情節中所寫的內容必須符合所有關係人的利益與事後保證。
9. 用腦力激盪法窮舉出主要成功情節中可能會發生的擴充情況。
包含系統中所有能偵測到、必須處理的擴充情況。
10. 針對擴充情況，寫出它們的處理步驟。
在擴充情節中，所有步驟最後都會回到MSS，有可能是以成功傳回、也有可能是以失敗傳回。
11. 把比較複雜的流程粹取成子使用案例；不重要、比較小的子使用案例則合併回到呼叫它的使用案例中。
粹取出子使用案例是很簡單的事，不過它會增加專案的維護成本。
12. 重新調整整組使用案例：必要時新增、粹取或合併一些使用案例。
檢查可讀性、完整性，以及是否符合關係人的利益。

封底內頁

針對單一使用案例所做的欄位測驗：合格／不合格

下列所有問題的測驗結果都必須是「合格」的。

欄位	問題
使用案例標題	它是由主動動詞所構成、代表主要參與者目標的目標片語嗎？ 系統能達到這個目標嗎？
設計範圍與目標等級	有填寫這兩個欄位嗎？
設計範圍	使用案例有把設計範圍中所提到的系統視為黑箱嗎？（如果使用案例代表系統需求文件的話，答案就一定是「肯定的」。不過，如果使用案例是屬於白箱式企業使用案例的話，答案就可以是「否定的」。） 如果設計範圍中的系統就是我們所設計的系統，設計師所設計出來的東西有在設計範圍內、沒超出設計範圍嗎？
目標等級	使用案例中所描述的內容符合我們想要的目標等級嗎？ 目標真的是我們所想要的目標等級嗎？
主要參與者	主要參與者（他／她／它）會對 SuD 發生行為嗎？ 主要參與者（他／她／它）對 SuD 有目標，而且希望藉此獲得系統所承諾的服務嗎？
事先條件	這些事先條件具有強制性嗎？我們是否可在 SuD 中找到合適的地方設定這些條件？ 使用案例中是否永遠不需要測試這些條件有沒有成立？
關係人與其利益	關係人有名稱嗎？系統是否必須滿足他們所表達出來的利益呢？（關係人所表達的系統使用情形【usage】會隨著正式性與可容忍程度而異。）
最小事後保證	有保障關係人的所有利益嗎？
成功事後保證	有滿足關係人的所有利益嗎？
主要成功情節	是按照寫作步驟 3-9 所寫出來的嗎？

	它是由某個觸發事件所驅動，達到成功的事後保證嗎？
	它可以有其他不同、正確的執行順序嗎？
情節中的任何動作步驟	它是用次一層目標的片語所寫成的嗎？
	情節中所描述的動作流程，在成功完成某個動作步驟後，會直接執行下個動作步驟嗎？
	動作步驟中有很清楚表明是由那個參與者使用系統來達成目標嗎（他是「擊球者」）？
	參與者的意圖很清楚嗎？
	動作步驟的目標等級是否比整個使用案例的目標等級還低嗎？它是否如我們所願，只比使用案例的目標等級低一點點？
	你確定動作步驟中沒有描述到系統的使用者介面設計方式嗎？
	傳給動作步驟的資訊有很明確嗎？
	動作步驟是在「證實」某個條件「成立」，而不是在「檢查」這個條件嗎？
擴充情況	系統可偵測並處理這些擴充情況，而且這些情況是必要的嗎？
	這是系統實際上有需要的擴充情況嗎？
技術與資料變異情形清單	對主要成功情節來說，你確定這些不是常態行為的擴充情節嗎？
使用案例的整體內容	對贊助者與使用者來說，「這是你要的嗎？」
	對贊助者與使用者來說，「你能夠據此判別所交付的系統是自己想要的系統嗎？」
	對開發人員來說，「你能實作這個使用案例嗎？」





目錄

書名.....	1
譯者介紹.....	1
封底.....	2
封面內頁.....	3
使用案例的一些寫作提示.....	3
圖示.....	3
使用案例的寫作步驟.....	4
封底內頁.....	5
針對單一使用案例所做的欄位測驗：合格／不合格.....	5
目錄.....	7
譯序.....	16
緣起.....	16
電腦翻譯書.....	16
軟體工程.....	18
前言.....	21
本書所預設的讀者.....	21
本書結構.....	21
本書中的概念由來.....	22
書中的範例.....	23
本書是水晶系列叢書中關於使用案例方面的一本書.....	23
致謝.....	25
第 1 章 簡介.....	26
1.1 何謂使用案例（多一點或少一點）.....	26
使用案例 1 <input type="checkbox"/> 在網路上買股票 <input checked="" type="checkbox"/>	30
使用案例 2 <input checked="" type="checkbox"/> 拿到車禍理賠 <input checked="" type="checkbox"/>	31
使用案例 3 <input type="checkbox"/> 記錄某個貨箱送達 <input checked="" type="checkbox"/>	32
1.2 你用的使用案例不等於我的.....	33
使用案例 4 <input checked="" type="checkbox"/> 買東西（非正式格式版本） <input checked="" type="checkbox"/>	37
使用案例 5 <input checked="" type="checkbox"/> 買東西（正式格式版本） <input checked="" type="checkbox"/>	38
Steve Adolph：在新領域中「發掘」需求.....	41
1.3 需求與使用案例.....	42
把使用案例當成連接專案用的結構.....	44
1.4 什麼時候使用案例會更有價值.....	45
1.5 管理自己的精力.....	47
1.6 以系統使用情形簡述暖身.....	49

系統使用情形簡述：使用「快速提領現金」	50
1.7 習題	51
第一部分 使用案例的主體部分	53
第 2 章 把使用案例當作系統的行為合約	54
2.1 對系統有目標的參與者間的互動情形	54
參與者對系統是有目標的	54
目標可能會失敗	56
互動情形可以是合成的	56
把一些情節集合在一起，以形成使用案例	59
2.2 有利益的關係人間的合約	61
2.3 （用 UML 畫出來、跟使用案例有關的）圖形模型	62
第 3 章 設計範圍	66
3.1 功能範圍	67
參與者 — 目標清單	67
使用案例簡介	68
3.2 設計範圍	69
用圖形化的圖示來強調設計範圍	71
各種不同設計範圍的範例	72
3.3 最外層使用案例	82
3.4 使用定義專案範圍用的工作成果	83
3.5 習題	84
第 4 章 關係人與參與者	85
4.1 關係人	85
4.2 主要參與者	86
為何主要參與者有時候不重要、有時候又很重要	87
參與者 v.s. 角色	89
列出主要參與者的特徵	90
4.3 支援性參與者	91
4.4 討論中系統	91
4.5 內部參與者與白箱式使用案例	91
4.6 習題	92
第 5 章 三個命名過的目標等級	
5.2 使用者目標等級（藍色、海平面 ^ㄤ ）	
5.3 目標摘要等級（白色、白雲 [☁] ／風箏 ^ㄉ ）	
重新思考：如何找到最外層使用案例	
5.4 子功能目標等級（靛藍色／黑色、海平面下 ^ㄨ ／蚌 ^ㄉ ）	
把等級比較低的目標彙總起來	
5.5 用圖形化的圖示來強調目標等級	

5.6	找到合適的目標等級.....
	找到使用者目標.....
	提升或降低目標等級.....
5.7	比較長的寫作範例：分別屬於不同目標等級的「處理索賠」使用案例.....
5.8	習題.....
第 6 章	事先條件、觸發事件與事後保證.....
6.1	事先條件.....
6.2	最小事後保證.....
6.3	成功事後保證.....
6.4	觸發事件.....
6.5	習題.....
第 7 章	情節與動作步驟.....
7.1	主要成功情節.....
	環繞在情節周圍的相似結構.....
	使用案例 在網路上買股票.....
	情節中的主體.....
7.2	動作步驟.....
	寫作指引.....
	寫作指引 1：使用簡單句.....
	寫作指引 2：很清楚寫出「球在誰的手上」.....
	寫作指引 3：用鳥瞰觀點來寫.....
	寫作指引 4：每個動作步驟對目標達成都要有所進展.....
	寫作指引 5：寫出參與者的意圖，而非動作.....
	寫作指引 6：在動作步驟中放入「合理的」一組動作.....
	寫作指引 7：用「證實」的語調來寫，而不要用「檢查某個東西是否有成立」的語調來寫.....
	寫作指引 8：選擇性提到動作步驟的執行時間.....
	寫作指引 9：慣用寫法：「使用者要系統 A 跟系統 B 做某件事」... ..
	寫作指引 10：慣用寫法：「重複執行動作步驟 x-y 直到...為止」... ..
	動作步驟要編號？還是不要？.....
7.3	習題.....
第 8 章	擴充情節.....
8.1	擴充情節的基本概念.....
8.2	擴充情況.....
	用腦力激盪法找出所有可想到的失敗情形與替代流程.....
	寫作指引 11：描述擴充情況時，請說出可偵測到的東西.....
	把擴充情況清單合理化.....

	把失敗情況合併起來.....	
8.3	擴充情節中的處理方式.....	
	寫作指引 12：把擴充情況的處理方式做適當的縮排.....	
	失敗情況中的失敗情況.....	
	把擴充情節獨立成新的使用案例.....	
8.4	習題.....	
第 9 章	技術與資料變異情形.....	
第 10 章	使用案例間的連結關係.....	
10.1	子使用案例.....	
10.2	擴充使用案例.....	
	何時該寫出擴充使用案例.....	
10.3	習題.....	
第 11 章	使用案例格式.....	
11.1	我們可選用的使用案例格式.....	
	正式格式.....	
	非正式格式.....	
	單欄表格格式.....	
	兩欄表格格式.....	
	RUP 的使用案例寫作風格.....	
	採用條件子句的使用案例寫作風格.....	
	採用 Occam 程式語言的使用案例寫作風格.....	
	用圖形呈現出來的使用案例寫作風格.....	
	UML 中的使用案例圖.....	
11.2	影響使用案例寫作風格的一些力量.....	
	一致性.....	
	複雜度.....	
11.3	五種不同專案型態適用的寫作標準.....	
	爲了引導需求.....	
	爲了建立企業模型.....	
	爲了評估系統大小規模.....	
	爲了短期、有高度時間壓力的專案.....	
	爲了寫出詳細的功能需求.....	
11.4	結論.....	
11.5	習題.....	
第二部分	經常會討論到的一些相關主題.....	
第 12 章	何時才算寫完使用案例.....	
	關於寫完使用案例這檔子事.....	
第 13 章	如何擴大規模處理大量的使用案例.....	

	把每個使用案例描述得少一點（採用低精確度呈現方式）
	把使用案例分群.....
第 14 章	CRUD 使用案例與可參數化使用案例
14.1	CRUD 使用案例
使用案例 32	管理報表 
使用案例 33	儲存報表 
14.2	可參數化使用案例.....
第 15 章	建立企業流程模型.....
15.1	建立模型 v.s. 進行設計
從核心業務開始思考.....	
從（中間的）企業流程開始往技術層面思考.....	
直接從技術往回看企業流程.....	
15.2	把企業使用案例跟系統使用案例連接起來.....
	Rusty Walters：建企業模型 v.s. 系統需求
第 16 章	被漏掉的需求.....
16.1	資料需求的精確度.....
16.2	從使用案例連到其他需求.....
第 17 章	在整個開發流程中的使用案例.....
17.1	專案組織中的使用案例.....
根據使用案例標題來組織專案.....	
處理橫跨不同發行版本的使用案例.....	
每次實作出完整的情節.....	
17.2	從使用案例對應到設計工作清單或系統特性清單.....
使用案例 34	記錄舊換新活動 
17.3	從使用案例到設計.....
物件導向設計師要特別注意的地方.....	
17.4	從使用案例到使用者介面設計.....
17.5	從使用案例到測試案例.....
使用案例 35	下貨品訂單、產生發票（測試範例） 
17.6	實際可行的使用案例寫作過程.....
個人與團體並重的寫作過程.....	
每個使用案例所需花費的寫作時間.....	
大型團體中的使用案例蒐集方式.....	
	Andy Kracus：跟大量、各式各樣的非技術人員一起合作 蒐集使用案例.....
第 18 章	使用案例簡介與終極開發流程.....
第 19 章	常見的使用案例寫作錯誤.....
19.1	沒描述到系統的行為.....

19.2 沒描述到主要參與者的行為.....
19.3 描述太多使用者介面細節.....
19.4 所描述的目標等級太低.....
19.5 寫作目的跟寫作內容不一致.....
19.6 進階範例：描述過多的使用者介面細節.....
使用案例 36 ■找出解決方案 — 改善前的版本 ㄊ
使用案例 37 ■找出可行的解決方案 — 改善後的版本 ㄊ
第三部分 給忙碌者的一些寫作提示.....	94
第 20 章 跟單一使用案例有關的寫作提示.....	95
寫作提示 1：使用案例是一種散文.....	95
寫作提示 2：把使用案例寫的很容易讀.....	95
寫作提示 3：只用一種句型.....	97
寫作提示 4：「包含」某些子使用案例.....	98
寫作提示 5：球在誰的手上？.....	99
寫作提示 6：寫出合適的目標等級.....	99
寫作提示 7：不要寫出 GUI.....	101
寫作提示 8：使用案例的兩種結尾.....	102
寫作提示 9：需要對關係人做出事後保證.....	103
寫作提示 10：事先條件.....	106
寫作提示 11：針對每個使用案例所做的欄位合格／不合格測驗.....	106
第 21 章 跟整組使用案例有關的寫作提示.....	109
寫作提示 12：可不斷展開的敘事情節.....	109
寫作提示 13：公司設計範圍與系統設計範圍兩者.....	110
寫作提示 14：使用案例格式的核心價值與替代性價值.....	112
核心價值.....	112
恰當的替代性價值.....	114
不恰當的替代性價值.....	116
寫作提示 15：跟整組使用案例有關的品質問題.....	118
第 22 章 進行跟使用案例相關工作時的寫作提示.....	119
寫作提示 16：使用案例只是需求文件中的第三章（第四章在哪裡呢？）.....	119
寫作提示 17：寫使用案例時先求廣度.....	119
寫作提示 18：使用案例的 12 步寫作訣竅.....	122
寫作提示 19：了解犯錯成本.....	123
寫作提示 20：偏好藍色牛仔褲（譯註：簡單方便的意思）.....	124
寫作提示 21：處理失敗情況.....	126
寫作提示 22：專案剛開始與快結束時，工作職稱是很重要的.....	127
寫作提示 23：由參與者扮演角色.....	129

寫作提示 24：偉大的畫圖騙局.....	130
寫作提示 25：偉大的工具之爭.....	133
寫作提示 26：用使用案例標題與使用案例簡介來規劃專案.....	136
附錄 A UML 中的使用案例.....	138
A.1 橢圓形與簡單棒形人偶圖示.....	138
A.2 UML 中的包含關係.....	139
寫作指引 13：把等級比較高的目標畫得高一點.....	140
A.3 UML 中的擴充關係.....	141
寫作指引 14：把擴充使用案例的位置畫得低一點.....	143
寫作指引 15：對 UML 中的各種使用案例關係，請用不同的箭頭形狀.....	144
正確的擴充關係用法.....	145
擴充點.....	146
A.4 UML 中的一般化關係.....	148
正確的一般化關係用法.....	149
寫作指引 16：把一般性的目標畫得高一點.....	150
使用一般化關係時，可能會發生的錯誤.....	151
A.5 下層使用案例 vs. 子使用案例.....	153
A.6 畫使用案例圖.....	155
寫作指引 17：不要在情境圖中畫出目標等級比使用者目標等級更低的目標.....	155
寫作指引 18：把支援性參與者放在圖的右邊.....	155
A.7 用以文字為基礎的使用案例代替以圖形為主的_UC.....	156
附錄 B 部分習題解答.....	158
第 3 章.....	158
習題 3.1.....	158
習題 3.2.....	158
第 4 章.....	159
習題 4.2.....	159
習題 4.3.....	160
第 5 章.....	161
習題 5.1.....	161
習題 5.2.....	161
第 6 章.....	161
習題 6.1.....	161
習題 6.4.....	162
第 7 章.....	162
習題 7.1.....	162

習題 7.2.....	163
習題 7.4.....	163
第 8 章.....	164
習題 8.1.....	164
習題 8.5.....	165
第 11 章.....	166
習題 11.1.....	166
附錄 C 字彙表.....	167
主要術語.....	167
使用案例的分類方式.....	168
跟使用案例相關的一些圖.....	169
附錄 D 參考資料.....	171
參考書籍.....	171
參考文章.....	171
有用的網路資源.....	171
英文索引.....	173
A.....	173
B.....	175
C.....	176
D.....	178
E.....	179
F.....	180
G.....	183
H.....	185
I.....	186
J.....	186
K.....	187
L.....	187
M.....	188
N.....	189
O.....	189
P.....	189
Q.....	191
R.....	191
S.....	195
T.....	199
U.....	199
V.....	203

W.....	203
X.....	204
Y.....	204

譯序

緣起

這篇譯序緣起於一個輾轉難眠的週末夜晚，小弟腦海裡有一些揮之不去的想法，不得不把它們記錄下來。既然各位都是**有緣**看到本書的人，不如就跟大家分享一下這些想法。不過，由於接下來所說的跟本書沒有很大的直接關係，主要是個人針對軟體業現況所做的一些觀察，所以不想聽小弟贅言者，可直接跳過不看。

電腦翻譯書

既然這是一本電腦翻譯書，就從這裡開始談起吧！

品牌市場 v.s. 量販市場

老實說，以國內繁體中文書的市場規模，在比較艱深的議題上，實在養不起好的作者。所以這方面的需求只好依賴非繁體中文來源，目前是以英文書為主、日文書為輔。

翻譯書需要做市場區隔，我們可簡單分為品牌市場、量販市場兩種。每本書出版時都希望獲取最大利潤。不過，價格不是決定「利潤」的唯一因素。為了獲取該有的利潤，國內電腦翻譯書市場最後應該會區分成高價位的「品牌」市場跟大眾化的「量販」市場。而所謂的品牌市場，指的是門檻比較高的市場（例如 Design Pattern、CMM），這樣的書影響定價的是它本身所具有的稀有性，用比較易衡量的因子來說，就是：「原作+譯者」。例如四人幫的 Design Pattern 一書就是該領域的聖經本，本身具有相當的稀有性，加上葉秉哲所翻譯的中文品質，其實是足以跟英文版定相同價格的；相反地，決定量販市場定價的是元／頁，目前國內的元頁比約在 1.2 元／頁上下，也就是說 600 頁的翻譯書，定價大概是 720 元。

談到翻譯書的出版，就不能忘了「出版社」。出版社重不重要呢？答案是肯定的。好的版面、索引、紙張、校稿，甚至是美觀的印刷，都跟出版社出好書的決心有關。同時，唯有它擔任稱職的守門人，才能阻絕「低劣」的譯者出現。此外，出版社是書本定價、版稅的決定者，適當的定價、合理的版稅才能產生良性循環，不至於斷送好譯者的生路，甚至趕盡殺絕。不過，現有的翻譯書版稅水準為 4%-10%，除非它能調高到 6%（量販市場下限）至 15%（品牌市場上限）之間，否則將很難看到專職譯者存活下來。

然而，這樣高的版稅對出版社，有什麼好處呢？首先，好的譯者才能存留下來，畢竟在版稅收入不佳的情況下，單單靠譯者理想是無法維持太久的。其次，出版

社在成本壓力下勢必越來越走向專業。過去，我們是盡量減少虧損成本，現在則是要減少虧損機會，這樣的 Know-How，勢必讓一些不稱職的出版社退出競爭行列。出版業向來有一本書賣座養九本書的現象存在。過去，我們可能有機會出九本不賣錢書。現在，每做一本書，在所需花費成本越來越高的情形下，可能要壓縮在只有四本書不賣錢的範圍內。唯有事先做好評估工作，才能減少虧損發生。過去大家總是把目標朝向降低虧損成本，而去想辦法減少費用（例如降低版稅、減少校稿次數等），現在則需要想辦法減少虧損機會。

如何挑選一本好的翻譯書

出版社、譯者的責任是做出好書，那麼讀者的責任就是對自己的荷包負責：慎選好書。由於目前繁體中文翻譯書市場良莠不齊，所以各位看官買書時可要睜大眼睛。以下就個人淺見略述選書要點。

好的譯者等於品質保證

之前提過「作者+譯者」可用來衡量稀有性。同樣地，根據這點，我們也可以讓譯者扮演兩種角色：挑選好書、把書翻譯好。

好的譯者會慎選好書翻譯。翻譯書是曠日費時、替人作嫁的工作。當我們看到一本好的翻譯書時，絕大部分功勞都會記在作者身上，譯者往往只是個無名的幕後英雄。在無名的情況下，當然要有利可圖，而好書就是的市場獲利保證之一。巧婦難為無米之炊，沒有好書，譯者是不可能出一本好翻譯書的。

其次是好的譯者會愛惜自己的羽毛，把書翻譯好。翻譯品質的良莠要看譯者的做事態度而定。由於消費者是：看到好翻譯時，不見得記得翻譯者是誰；看到爛翻譯時，卻可能對此人印象深刻。書賣不好事小，個人將來在職場上的聲譽可不能小覷。

講了這麼多，還是漏了一點，誰才是好的譯者呢？其實消費者只要多問一下，就不難在各個專業領域中發現「一兩位」好譯者。好譯者是不可能十八般武藝樣樣精通的，大家可尋找「只鎖定某個領域翻譯」、「翻譯數目多」或「有著作」者，把這樣的人挑選出來作為初始名單，再用口碑刪去法，相信好的譯者就呼之欲出了。例如侯捷專精在 C++ 領域、有多本翻譯作品，且有著作，就可列入好譯者的候選名單之列。

從蛛絲馬跡選好書

只挑譯者的話，相信還是會遺漏許多初出茅廬、品質卻還不錯的翻譯作品。下面列出幾點作為直接從作品挑選好書的指南。

1. 在必要處加上譯註

譯者對此書的相關領域具有相當程度時，才有能力加譯註。此外，從此處也可看出譯者試圖補充書中內容的不足之處。

2. 保留英文頁碼

由於中文排版軟體在功能上的不足，重新製作中文索引是件很困難的工作。既然如此，我們只好退而求其次，保留英文索引與相關頁碼。不過，除非譯者要求，否則出版社是不會自找麻煩，排版時保留英文頁碼的。由此可看出譯者的用心。

3. 在英文書目中，列出已有中文翻譯版之中文書名。

連這麼細微的地方都不放過，可見譯者是很細心的。

4. 曾經向原作者討教譯者不了解之處

有時候，譯者不見得能了解一些口語說法或不清楚某個講法的來源或出處，這時候是有必要跟原作者請教的。由此可看出譯者不會隨意翻譯自己不了解的地方。

5. 有具名校編

這點可看出出版社是否把關嚴格。多一個人看，至少可發現一些寫得不清楚、打字或錯別字，以及翻譯有問題或譯詞不佳之處。

6. 中英對照

中英對照是讀者對翻譯作品的最嚴格評斷。兩相對照之下，譯者的文筆好壞或正確性都可以馬上赤裸裸呈現出來。願意採用這種方式的譯者，對自己的作品是相當有自信的。

7. 是否有常見的錯別字

錯別字並非不可原諒的錯誤，不過比較少的錯別字既可看出校稿的用心，又可增加我們對譯者在中文造詣方面的信心。常見的錯別字有：把「身分」寫成「身份」、搞不清楚作為動詞的「記錄」跟名詞的「紀錄」、把「關聯」寫成「關連」（這也是譯者在 *UML 精華* 第二版中所犯的錯誤，特此向大家致歉）。

8. 句子是否通順

這點可看出譯者的翻譯經驗。一般來說，有經驗的譯者會避免濫用句中的助詞，例如「的」、「會」。還有，避免一些贅字的存在，例如把「遺漏掉」改寫成「漏掉」。

軟體工程

在軟體工程方面，國內的資源相當少。點空間 (<http://www.dotspace.idv.tw>) 的興起，一方面是有市場上的需要，另一方面則是它比較具有結構性，分別以技術面、管理面與開發流程面來討論相關主題。以下僅就小弟比較熟悉的開發流程部分說明。

開發流程

由於小弟所翻譯的作品都集中在 UML、RUP 上，所以自己對重量級開發流程有比較深的體認。學習過重量級開發流程的人，在採用這種開發流程時，不可避免地都會自行「剪裁」原本厚重的開發步驟，形成「個人版」的輕量級開發流程，而 XP 的出現恰好反映了大家的內心渴望。

RUP v.s. XP

這幾十年來，軟體開發流程歷經兩個不同階段。在 1990 年代初期，重量級開發流程相當興盛，這個時期的成果以 CMM、RUP 等為指標。接下來，在 1990 年代末期，輕量級開發流程逐漸受到矚目，以 XP 為代表。

RUP 非常重視建模型這項工作，它會產生大量以 UML 為基礎的使用案例模型與設計模型。這時候，蒐集需求、設計與寫程式等工作，通常會由不同角色的人所負責。這些扮演不同角色的人在交接工作時都是以文件為主的。

XP 則非常重視溝通。在需求方面，它要求顧客跟開發團隊要一起工作；設計方面則透過程式重整達成目的。RUP 與 XP 間的差異，請參見序表 1

序表 1 RUP v.s. XP

	RUP	XP
理念	<ul style="list-style-type: none">◆ 風險驅動◆ 以架構為中心	<ul style="list-style-type: none">◆ 溝通◆ 簡單 (simplicity)◆ 回饋◆ 勇氣
工具	<ul style="list-style-type: none">◆ 使用案例(由分析師撰寫，它是關係人間的行為合約)◆ 架構觀點(系統整體設計方式)	<ul style="list-style-type: none">◆ 使用者故事(由使用者撰寫、近似於使用案例簡介【use case brief】，不過前者所涵蓋的範圍較小，而且它是開發人員對顧客承諾以後會繼續討論的東西)◆ 突圍性解決方案 (spike solution)◆ 系統隱喻 (metaphor) (以類比方式來描述系統架構，它可讓命名方式具有一致性)◆ 單元測試◆ 驗收測試 (因為採用持續整合【continuous integration】，所以沒有整合測試)
解決方案	<ul style="list-style-type: none">◆ 在初始階段用主要使用案例	<ul style="list-style-type: none">◆ 用突圍性解決方案 (spike solution) 降低風險、提高專案時程估計值的準確性。

	評估風險	◆	測試優先 (test first) 以保護系統重整過程。
◆	在詳述階段建立健全的架構基礎，以消除專案高風險元素	◆	程式重整 (refactoring) 以逐漸形成系統架構。
		◆	搭檔編程 (pair programming) 加強設計上的溝通。
		◆	顧客駐點 (customer on site) 以補足使用者故事的不足。
規畫方式	◆ 階段計畫	◆	發行計畫 (由管理人員決定 4 個變數【成本、時間、品質與設計範圍】中的 3 個變數，而開發人員則決定最後一個變數)
	◆ 反覆計畫	◆	反覆計畫
	◆ 固定反覆期間的長度	◆	專案速度可以根據時間 (固定時間長度) 或範圍 (固定使用者故事數目) 決定。

UML 已死？

RUP 主張計畫設計 (planned design) (先有設計模型，再寫程式)、彈性設計 (例如樣式、框架等)，不過這種做法往往容易造成過度設計問題 (譯註：planned design 之所以翻譯成「計畫設計」是模仿計畫經濟【planned economy】的譯法)。另一方面，XP 則主張簡單設計 (simple design) 與程式重整 (refactoring)。也就是說，為了避免過度設計，我們只會針對目前需求，設計出夠用的東西。既然設計工作簡化了，就不用把設計跟寫程式工作獨立開來。這麼一來，是否就不用再畫 UML 圖呢？事實上，簡單設計最後還是會把專案導引到樣式方向。設計不是不存在，只是在必要時發生，而且會跟寫程式工作同時進行。目前有一種趨勢是：建模型的工具跟整合開發環境 (IDE) 整合在一起，例如 Together ControlCenter、Rational XDE 等。也就是說，過去我們會先用建模型的工具 (例如原先的 Rational Rose) 畫出模型，然後再用正向工程 (forward engineering) 把設計結果轉成程式碼。在寫程式的過程中，如果有改變設計的話，就以反向工程 (reverse engineering) 修改設計模型。現在的做法就比較極端 (extreme) 一點：寫程式時，我們可能用跟 IDE 整合在一起的建模型工具畫 UML 圖，這時候同時也會產生程式碼，反之亦然。我們再也不需要事後做多餘的正向、反向工程。如果目前的設計結果不敷使用，就用程式重整去變更設計方式。現在，大家應該可以很清楚了解到「設計工作不是消失，而是跟寫程式工作更緊密結合在一起」這個概念吧！換句話說，畫 UML 圖跟寫程式是時時刻刻同步進行的。最後，由於一張圖可勝過千言萬語，所以如果畫圖跟寫程式工作可緊密結合的話，我們在討論時還是需要用可提高認知程度的 UML 圖來溝通。所以 XP 的開發方式應該是「以程式碼為主、UML 圖為輔」的。

前言

爲了捕捉行爲需求、軟體系統或描述企業流程，有越來越多的人開始寫使用案例。它看起來很簡單——只要寫出系統的使用情形就可以了。不過，一旦開始寫使用案例時，我們馬上就面臨一個問題：「使用案例中到底該寫些什麼東西——寫多少、多詳細呢？」這是很難回答的一個問題。問題根本在於寫使用案例這件事，基本上相當於寫散文，因此想要把使用案例寫清楚、寫好，所面臨的困難跟寫一般性的散文是一樣的。說明怎樣的使用案例才是好的使用案例已經很難了。不過，我們要知道的卻是更難的東西：如何寫出很好的使用案例。

本書內容是我在寫使用案例或進行教學時的一些寫作指引，裡面說明人們該如何思考或觀察什麼，最後才能寫出比較好的使用案例、適當切割使用案例。

本書中同時有秀出一些好的跟不好的使用案例範例，而且以不同方式寫出很具真實感的使用案例。最棒的一個消息是：縱然不是最好的使用案例也是*有用的*。就算是好壞屬於中等程度的使用案例，也比跟它競爭的其他需求文件還有用。所以，請大家輕鬆一點，我們只要寫出具有可讀性的東西，對組織就有幫助。

本書所預設的讀者

本書主要是作爲業界專家自修時的教材，所以它是以適合自修的方式來組織這本寫作指南的。書中有入門到進階的教材，包括：概念、範例、寫作提示與習題（有些有解答，其他則否）。

使用案例的寫作指導老師應該可從本書找到一些合適的說明與範例給開發團隊看，而課程助教也可用本書來準備課程資料，必要的話還可以把本書當成指定教材。（然而，因爲許多習題都有解答，所以課程助教要寫出自己的考題。☺）

本書結構

本書的組織方式是：一開始先對使用案例做簡介，然後再仔細說明使用案例的主要內容、常見問題、給忙碌者看的寫作提示等，最後是附錄部分。

在**簡介**中，我們會簡單介紹一些很重要的概念，依序討論到：「使用案例看起來該像什麼？」、「何時該寫使用案例？」以及「哪些是合法的寫法？」這些問題的簡單答案是：根據寫使用案例的時間、地點、跟誰寫與寫的原因，使用案例會有不同寫法。我們會在簡介這一章中開始討論這些問題，而在整本書中延續這些討論。

在**第一部分、使用案例的主要內容**中，我們針對大家需專精的每個主要概念，分別用一章來解釋它們，同時也會提供一些範本，讓大家知道裡面該寫些什麼。這一部分包括「把使用案例當成系統的行爲合約」、「設計範圍」、「關係人與參與

者」、「三個命名過的目標等級」、「事先條件、觸發事件與事後保證」、「情節與動作步驟」、「擴充情節」、「技術與資料變異情形」、「使用案例間的連結關係」與「使用案例格式」。

在**第二部分、經常會討論到的一些相關主題**中，我們討論會經常不斷發生的一些特殊主題：「何時才算寫完使用案例」、「如何擴大規模處理大量的使用案例」、「CRUD 與可參數化使用案例」、「建立企業流程模型」、「被漏掉的需求」、「在整個開發流程中的使用案例」、「使用案例簡介與終極開發流程」與「常見的使用案例寫作錯誤」。

針對已讀完本書或了解本書內容、希望可時時複習重要概念的人，我們提供**第三部分、給忙碌者的一些寫作提示**。這一部分包括「跟單一使用案例有關的寫作提示」、「跟整組使用案例有關的寫作提示」與「進行跟使用案例相關工作時的寫作提示」。

本書最後還包括了四個附錄：附錄 A 討論「UML 中的使用案例」、附錄 B 是「部分習題解答」，而附錄 C 則是「字彙表」；附錄 D 是寫使用案例時可用到的「參考資料」清單。

本書中的概念由來

在 1960 年代末期，當 Ivar Jacobson 還在易利信公司做電話系統時，發明了後來稱為使用案例的東西。到了 1980 年代末期，他把這些概念引進到物件導向程式設計的社群中，這些人把使用案例視為開發流程中可用來填補需求跟設計或實作間明顯差異的東西。我在 1990 年代初期上過 Jacobson 的課。當時，雖然他跟他的團隊所用的字眼跟我用來描述 *目標*（譯註：主要成功情節）或 *目標失敗情形*（譯註：擴充情節）的字眼不同，不過後來我終於了解到他們也正在使用這些概念。彼此討論過幾次之後，Jacobson 與我發現到彼此的模型沒有明顯差異在。後來，我慢慢擴充他的模型，融入自己在使用案例方面的一些深入的了解。

當我在 1994 年替 IBM 的顧問團隊製作使用案例寫作指南時，建立了參與者與目標概念模型。這個模型揭開使用案例中的許多神秘之處，並且教大家如何組織使用案例與寫使用案例。從 1995 年開始，參與者與目標模型以非正式方式流傳，先是公佈在 <http://Alistair.Cockburn.us>，稍後在 <http://www.usecases.org>，最後則是出現我在 *Journal of Object-Oriented Programming*（1997 年）中的一篇文章：

「Structuring Use Cases With Goals。」

雖然這個概念有些鬆散，不過從 1994 到 1999 年間，整個概念還是相當穩定。最後，經由教學與訓練過程，我找到人們學習使用案例這個簡單概念時需經歷相當學習曲線的原因所在（請不要太難過，我第一次用使用案例時，也犯過許多同樣的錯誤！）。由於對使用案例的用法有一些深入了解，再加上參與者與目標模型中的一些缺點，最後促成了我寫出本書做一些解釋，並加入關係人與利益模型。後者是我第一次公開的新概念。

統一模型語言（Unified Modeling Language，UML）對這些概念有一點影響在，反之亦然。Jacobson 的前同事 Gunnar Overgaard 遵照他的想法，寫出 UML 中大部分的使用案例內容。然而，UML 標準團隊有強烈的繪圖工具傾向，所以標準中喪失了使用案例在文字方面的本質。Gunnar Overgaard 跟 Ivar Jacobson 討論過我的概念，確認我對使用案例的大部分概念都可涵蓋在 UML 中代表使用案例的橢圓形中，不會影響到 UML 標準中所提到的東西或受其影響。這代表本書概念跟 UML 1.3 版中的使用案例標準很相容，大家可以一起使用它們。另一方面，如果你只讀過 UML 標準，由於裡面沒有討論到使用案例的內容或寫法，所以你將不了解使用案例是什麼或該如何善用使用案例。結果可能會很危險，你可能會把使用案例視為圖形化的東西，而不是文字或由文字所構成的結構。因為本書的目標是告訴你一些有效的使用案例寫法以及 UML 標準中很少說明的事，所以我把自己對 UML 的評論獨立開來，放在附錄 A 中。

書中的範例

本書的寫作範例都盡量引用真實專案中的一些例子，其中有些可能會有不完美之處。我意圖藉此告訴大家：對寫出這些使用案例範例的專案開發團隊來說，它們已經能滿足其需要。至於那些不完美之處，對寫使用案例來說，都在差異性與經濟效益的允許範圍之內。

Addison-Wesley 的編輯大人們說服我稍微整理了一下這些範例，讓它們以比我原本想要呈現還工整的方式出現在本書中。這樣做的目的是在實際與夠用的使用案例風貌之外，強調出正確寫法。我希望大家既能覺得這些範例很有用，也能了解專案中的使用案例實際寫法。你可以把我的一些寫作規則應用在這些範例上，找出可改善這些範例的地方。事實上，這樣的事永遠都會存在，因為改善某人的寫法是一件永無止境的事，本人也願意接受大家的質問與批評。

本書是水晶系列叢書中關於使用案例方面的

一本書

本書是針對軟體專家所寫的水晶系列叢書中的一本書。這一系列叢書強調輕量級、以人為主的軟體開發技術。有些書可能是討論某個技術，有些則是討論專案中的某個角色，有些則是討論團隊合作方面的議題。

水晶系列叢書都符合下面兩項基本原則：

- ◆ 軟體開發過程是創造與溝通的團隊合作遊戲。大家拓展個人的開發技能並增進團隊合作的效果之後，就能改善軟體開發過程。
- ◆ 不同專案對軟體開發過程會有不同的需要。系統具有不同特性，而且是由不

同大小的開發團隊所建構出來的。此外，開發團隊成員對系統也會有不同的價值觀與開發優先順序。所以，我們是沒有辦法知道哪個方式是製造軟體的最佳方式。

Software Development 是水晶系列叢書中的一本基本書籍，裡面詳述一些概念：把軟體開發過程視為團隊合作遊戲、把方法論視為文化交流以及一整族的方法論等。這本書中分出方法論、技術與開發活動、工作成果以及標準幾個構面。針對這些內容，我們把跟使用案例有關的基本想法寫在本書的 1.2 *你的使用案例不等於我的使用案例* 中。

使用案例最佳實務-寫作指南、秘訣與範本 是一本技術方面的寫作指南，裡面鉅細靡遺說明了使用案例的寫法。雖然這些技術都可適用在大部分專案中，不過我們還是建議你要根據專案實際所需、選擇合適的範本與寫作標準來寫使用案例。

致謝

這裡要感謝許多人。感謝他們審查本書草稿，要求我釐清會對他們的客戶、同事與學生造成困擾的一些主題。特別感謝 Russel Walters，他是有經驗的人，對開發團隊的直接、實際需要也有銳利眼光，感謝他的鼓勵與特殊回饋。感謝 FirePond & Fireman 基金保險公司提供實際的使用案例範例。Pete McBreen 是第一位嘗試去用關係人與利益模型的人，本書加入了他的看法、獨到眼光與改善建議。感謝矽谷樣式團隊（Silicon Valley Patterns Group）細心閱讀本書的早期草稿，也謝謝他們對許多內容與概念提供了很有用的評論。在 Fort Union Beans & Brew 工作的 Mike Jones 則提供了用來代表子系統使用案例的螺絲圖示。

特別要感謝 Susan 這麼確實地讀本書，她修正了我們可想到的任何東西，包括：章節順序、內容、編排格式，甚至是使用案例的範例。她貢獻的大量成果很顯著地改善了本書的品質，這些貢獻都反映在本書最後一版上。

還有許多其他審查過本書的人提供了許多詳細的評論與鼓勵，包括 Paul Ranney、Andy Pols、Martin Fowler、Karl Waclawek、Alan Williams、Brian Henderson-Sellers、Larry Constantine 與 Russel Gold 等人。Addison-Wesley 的編輯們也費心修正了本人不優雅的句子與打字錯誤。

感謝我的學生幫我找出本書中所提到概念的一些缺失。

再次感謝我們家人、Deanna、Cameron、Sean 與 Kieran 以及在 Fort Union Beans & Brew 工作的人，他們提供我一個充滿鬥志、愉悅的環境。

有更多跟使用案例有關的東西放在之前提過的兩個網站：Alistair.Cockburn.us 與 www.usecases.org。為了避免以後發生尷尬，提醒大家，在小弟的姓名中 Co-burn 的 o 是發長音的。

第1章簡介

使用案例看起來應該像什麼？

為何不同的專案開發團隊需要不同的使用案例寫作風格？

使用案例適合用在需求蒐集工作中的什麼地方？

寫使用案例時該如何暖身？

開始寫使用案例的細節之前，心裡面對上述問題有一些答案是很有幫助的。

What do use cases look like?

Why do different project teams need different writing styles?

Where do use cases fit into the requirements gathering work?

How do we warm up for writing use cases?

It will be useful to have some answers to these questions before getting into the details of use cases themselves.

1.1 何謂使用案例（多一點或少一點）

我們用使用案例來捕捉系統行爲，把它當作關係人間的合約（譯註：如果是終極開發流程所採用的使用者故事的話，它就是開發人員跟顧客承諾以後會繼續討論的東西）。使用案例中會描述系統回應某個關係人的請求時，在不同條件下所發生的系統行爲。我們把發出請求的關係人稱爲**主要參與者**（primary actor）。主要參與者會啓動跟系統間的一段互動關係，以達成某個目的。而系統則負責回應主要參與者的請求，且要保障所有關係人的利益（譯註：從這一段話，我們可以發現作者除了原先的參與者與目標模型之外，又加入了關係人與利益模型）。根據特定的請求與條件，我們可能會得到一些不同的行爲執行順序或情節。某個使用案例中會集合好幾個不同的相關情節（譯註：我們可以把使用案例視爲類別，情節視爲實例）。

A use case captures a contract between the stakeholders of a system about its behavior. The use case describes the system's behavior under various conditions as the system responds to a request from one of the stakeholders, called the primary actor. The primary actor initiates an interaction with the system to accomplish some goal. The system responds, protecting the interests of all the stakeholders. Different sequences of behavior, or scenarios, can unfold, depending on the particular requests made and the conditions surrounding the requests. The use case gathers those different scenarios together.

雖然使用案例可用流程圖、循序圖、派翠網路或程式語言呈現出來，不過，基本上使用案例是由文字所構成的。在正常情況下，我們把使用案例當成與人之間的一種溝通工具，而且開發人員通常會用它跟沒受過訓練的人溝通。這時候，簡

單的文字通常是最好的溝通工具。

Use cases are fundamentally a text form, although they can be written using flow charts, sequence charts, Petri nets, or programming languages. Under normal circumstances, they serve as a means of communication from one person to another, often among people with no special training. Simple text is, therefore, usually the best choice.

使用案例是一種寫作結果，它可增進開發團隊對未來系統的討論。開發團隊可能會用使用案例來記錄實際需求，也可能不會。當然，也有些開發團隊可能會用使用案例來記錄他們的最後設計結果。做這些事時，我們有可能是針對整個公司這麼大的系統來做的，也有可能只針對一個軟體應用程式這麼小的系統做的。有一個很有趣的地方是：縱然開發團隊採用不同的嚴謹程度、寫出不同等級的技術細節，某些基本的寫作規則卻可適用於各種情況。

↑ 英 1

The use case, as a form of writing, can stimulate discussion within a team about an upcoming system. The team might or might not document the actual requirements with use cases. Another team might document their final design with use cases. All of the above might be done for a system as large as an entire company or as small as a piece of a software application program. What is interesting is that the same basic rules of writing apply to all of these situations, even though the teams will write with different amounts of rigor and at different levels of technical detail.

當我們用使用案例來記錄組織的企業流程時，*討論中系統*（system under discussion, SuD）就等於組織本身，而關係人則是公司股東、顧客、賣東西給這個公司的人與政府管理機構。主要參與者包括公司顧客，可能還會有他們的供應商（譯註：讀者在閱讀時要謹記「關係人跟參與者」是不同的，「利益與目標」也是有差異的）。

當我們用使用案例來記錄軟體的行為需求時，SuD 就等於電腦程式，而關係人則是使用程式的人、擁有程式的公司、政府管理機構與其他電腦程式等。主要參與者包括坐在電腦螢幕前面的使用者或其他電腦程式。

When use cases document an organization's business processes, the system under discussion (SuD) is the organization itself. The stakeholders are the company shareholders, customers, vendors, and government regulatory agencies. The primary actors include the company's customers and perhaps their suppliers.

When use cases record behavioral requirements for a piece of software, the SuD is the computer program. The stakeholders are the people who use the program, the company that owns it, government regulatory agencies, and other computer programs. The primary actor is the user sitting at the computer screen or another computer system.

寫得好的使用案例將是很容易閱讀的，而且裡面的句子只會用一種句型來表達

— 用簡單句所寫成的動作步驟（action step） — 經由這些動作步驟，參與者可達成某個結果或把資訊傳給其他參與者。學習閱讀使用案例所需花費的時間應該不用幾分鐘。

不過，學會寫出使用案例則難多了。寫使用案例的人必須精通三種概念，並且要把它們應用在使用案例中的每個句子或整個使用案例上。剛開始聽到我這樣講時，可能會覺得有點奇怪，不過想要讓這三種概念可直覺用在使用案例，並不是件簡單的事。一旦你開始寫使用案例，馬上就會遇到一些困難。這三種概念分別是：

A well-written use case is easy to read. It consists of sentences written in only one grammatical form—a simple action step—in which an actor achieves a result or passes information to another actor. Learning to read a use case should not take more than a few minutes.

Learning to write a good use case is harder. The writer has to master three concepts that apply to every sentence in the use case and to the use case as a whole. Odd though it may seem at first glance, keeping these three concepts straight is not easy. The difficulty shows up as soon as you start to write your first use case. The three concepts are

- ◆ **設計範圍**：什麼是真正的討論中系統？
Scope: What is really the system under discussion?
- ◆ **主要參與者**：誰對系統具有目標？
Primary actor: Who has the goal?
- ◆ **目標等級**：目標等級有多高或低？
Level: How high- or low-level is that goal?

本章後面有幾個使用案例的範例。至於使用案例是由把些主要內容所構成的，我們會在下一章加以說明。現在，請先記住下列一些定義：

Several examples of use cases follow. The parts of a use case are described in the next chapter. For now, remember these summary definitions:

- ◆ **參與者**：會對系統產生行為的任何人或東西。
Actor: anyone or anything with behavior.
- ◆ **關係人**：討論中系統（SuD）會對它發生利益的任何人或東西。
Stakeholder: someone or something with a vested interest in the behavior of the system under discussion (SuD).
- ◆ **主要參與者**：會啟動跟 SuD 之間的一段互動關係、以達成某個目標的人或東西。
Primary actor: the stakeholder who or which initiates an interaction with the SuD to achieve a goal.
- ◆ **使用案例**：針對 SuD 行為所寫出來的合約。
Use case: a contract for the behavior of the SuD.

- ◆ **設計範圍**：找出我們所要討論的系統。
Scope: identifies the system that we are discussing.
 - ◆ **事先條件與事後保證**：使用案例執行前後必須成立的東西。
Preconditions and guarantees: what must be true before and after the use case runs.
- ↑ 英 2
- ◆ **主要成功情節**：在情節中，我們會描述每個動作步驟都成功的情況。
Main success scenario: a case in which nothing goes wrong.
 - ◆ **擴充情節**：情節中所發生的一些例外情況。
Extensions: what can happen differently during that scenario.
 - ◆ 擴充情節中會參考到主要成功情節中的動作步驟編號，而且它會說明需要偵測到的各種不同條件（例如，動作步驟 4a 與 4b 分別代表跟動作步驟 4 有關的兩個不同條件）。
Numbers in the extensions refer to the step numbers in the main success scenario at which each different situation is detected (for instance, steps 4a and 4b indicate two different conditions that can show up at step 4).
 - ◆ 當某個使用案例需要參考其他使用案例時，我們會把被參考到的使用案例加上底線。
When a use case references another use case, the referenced use case is underlined.

本書的第一個使用案例範例中描述某人在網路上買股票的情形。為了表示我們在某段期間內達成某個目標，我會把這個使用案例加上**使用者目標等級**（user goal level）的圖示，也就是在使用案例名稱旁邊加上海平面符號¹。第二個使用案例範例中則描述某人試著做車禍求償的情形，這個目標會延續一段比較長的期間。為了表示這個情況，我會把這個使用案例加上**目標摘要等級**（summary goal level）的圖示，也就是在使用案例名稱旁邊加上海平面以上的符號²。第五章會說明這些符號，而且這些符號都彙整在本書封面的內頁當中。

The first use case describes a person about to buy some stocks over the web. To signify that we are dealing with a goal to be achieved in a single sitting, I mark the use case as being at the user-goal level, and tag it with the sea-level symbol, . The second use case describes a person trying to get paid for a car accident, a goal that takes longer than a single sitting. To show this, I mark the use case as being at the summary level, and tag it with the above-sea-level symbol, . These symbols are explained in Chapter 5, and summarized inside the front cover.

第一個使用案例中說明某人跟程式（「PAF」）互動、用工作站連到網路的情形。黑箱符號³代表討論中系統是一個電腦系統（譯註：而且這時候會假裝成看不到討論中系統內的東西）。第二個使用案例中則描述某人跟一家公司互動的情形，我用建築物符號⁴標示它（譯註：在這兩個例子中，雖然設計範圍不同，一個是

電腦程式，另一個是公司，不過它們都把 SuD 視為黑箱，所以都用灰色的符號來表示)。說明使用案例的設計範圍與目標等級是必要的，不過要不要用符號來標示它則完全是選擇性的。

下面分別是使用案例 1 與使用案例 2。

↑ 英 3

The first use case describes the person's interactions with a program ("PAF") running on a workstation connected to the Web. The black-box symbol, , indicates that the system being discussed is a computer system. The second use case describes a person's interaction with a company, which I indicate with a building symbol, . The use of symbols is completely optional. Labeling the scope and level is not.

Here are Use Cases 1 and 2.

使用案例 1 ◻在網路上買股票

主要參與者：購買者

設計範圍：個人化的顧問／理財套件 (PAF)

目標等級：使用者目標等級

關係人與利益：

購買者 — 希望買股票，並且自動把它們加到 PAF 的投資組合中。

股票經紀商 — 希望完整的購買資訊。

事先條件：使用者已經有開啓的 PAF。

最小事後保證：要有足夠的登入資訊，當 PAF 偵測到錯誤情況發生時，會告知使用者提供一些細節 (譯註：根據 *UML 與樣式徹底研究第二版*【Applying UML and Patterns 2nd】一書的建議，我們這裡會用宣告式、被動式、過去式的時態 (被...) 寫出事後保證，強調我們是在宣告狀態的變化而不是設計如何達成這些變化)。

成功事後保證：被遠端網站告知這筆購買已經成功；歷史紀錄與使用者的投資組合都會被更新。

主要成功情節：

1. 購買者在網路上選擇想買的股票。
2. PAF 從使用者得到他想用的網站名稱 (E*Trade、Schwab 等)。
3. PAF 開啓跟這個網站之間的網路連線，保持控制權。
4. 購買者瀏覽網站買股票。
5. PAF 攔截網站回應並更新購買者的投資組合。
6. PAF 排列新的投資組合、秀給使用者看。

擴充情節：

2a. 購買者想要進入 PAF 沒有支援的網站：

2a1. 系統給購買者一個新建議，使用者可以選擇要不要取消這個使用案例。

- 3a. 在建立網路連線時，發生任何的網站失敗情形：
 - 3a1. 系統回報失敗情形，並且給購買者一些建議，看看是否要回到上一步。
 - 3a2. 購買者可回到這個使用案例的上一步或再連線一次。
- 4a. 在購買交易動作過程中，電腦當機或被關機：
 - 4a1. (我們這時候能做些什麼？)(譯註：其實這裡可以寫一些重開機時的應對措施)。
- 4b. 網站沒有告知交易是否成功，不過這筆交易動作會被延遲處理：
 - 4b1. PAF 把延遲處理登錄到歷史紀錄中，並且設定計時器，到時候再跟購買者詢問交易結果。
- 5a. 網站沒有傳回交易的必要資訊：
 - 5a1. PAF 在歷史紀錄中登錄缺乏資訊，要求購買者更新有問題的購買交易動作。

↑ 英 4

使用案例 2 拿到車禍理賠

主要參與者：索賠者

設計範圍：保險公司(「MyInsCo」)

目標等級：目標摘要等級

關係人與利益：

索賠者 — 盡可能得到賠償。

MyInsCo — 付出最少的合理金額。

保險部門 — 知道所有要遵守的條文。

事先條件：無。

最小事後保證：MyInsCo 要把索賠與所有活動登錄到歷史紀錄中。

成功事後保證：索賠者跟 MyInsCo 都同意一個賠償金額；索賠者得到這個金額。

觸發事件：索賠者要求索賠。

主要成功情節：

1. 索賠者根據具體資料索賠。
2. 保險公司檢查索賠者是否擁有有效保單。
3. 保險公司指定經銷商檢查這個個案。
4. 保險公司檢查所有相關細節都在保單條文範圍內。
5. 保險公司付錢給索賠者並結束這個個案。

擴充情節：

- 1a. 索賠資料不完整：
 - 1a1. 保險公司要求補足缺少的資訊。
 - 1a2. 索賠者提供不足資訊。
- 2a. 索賠者沒有一份有效保單。

- 2a1. 保險公司回絕索賠、告知索賠者、記錄所有相關活動、終止這項訴訟。
- 3a. 這時候沒有經銷商有空。
 - 3a1. (保險公司這時候會做什麼?)
- 4a. 這件意外違反保單中的基本條文：
 - 4a1. 保險公司回絕索賠、告知索賠者、記錄所有相關活動、終止這項訴訟。
- 4b. 這件意外違反保單中某些比較次要的條文：
 - 4b1. 保險公司開始跟索賠者討價還價定出賠償金額。

本書中的大部分使用案例範例都來自真正的專案，而且我們很小心避免更改這些使用案例（唯一例外是在使用案例中加上本來沒有的設計範圍與目標等級）。我希望你看到的是實務上可行的一些範例，而不是課堂上那些看起來漂漂亮亮的東西。大家不太有時間寫出很正式、完整或工整的使用案例。我們通常只有時間寫出「夠用」的使用案例而已，裡面只會包含一些必要的東西。之所以想秀出真實的範例給大家看，主要是因為：除非在別人的指導之下，否則大家是很難寫出完美使用案例的。甚至是我，大部分時候也很難寫出完美的使用案例。

↑ 英 5

Most of the use cases in this book come from live projects, and I have been careful not to touch them up (except to add the scope and level tags if they weren't there). I want you to see samples of what works in practice, not just what is attractive in the classroom. People rarely have time to make the use cases formal, complete, and pretty. They usually only have time to make them "sufficient," which is all that is necessary. I show these real samples because you rarely will be able to generate perfect use cases yourself, despite my coaching. Even I can't write perfect use cases most of the time.

使用案例 3 是由挪威央行的 Torfinn Aas 為他的同事、使用者代表跟他自己所寫的使用案例。從這個範例中，我們可知道：縱然用不同的使用案例格式，也不會失去使用案例的價值所在。寫這個使用案例的人在敘事情節中加入一些額外的企業情境，以說明在某個工作天當中，電腦應用程式是如何運作的。這種做法是很實際的，因為這樣一來我們就不用另外寫一份文件來描述企業流程。這些內容既不會造成混淆，對相關人員來說也是很有資訊價值的。

Use Case 3 was written by Torfinn Aas of Central Bank of Norway for his colleague, his user representative, and himself. It shows how the form can be modified without losing value. The writer added additional business context to the story, illustrating how the computer application operates in the course of a working day. This was practical, as it saved having to write a separate document describing the business process. It confused no one and was informative to the people involved.

使用案例 3 □記錄某個貨箱送達

RA — 「收貨員 (receiving agent)」

RO — 「登記作業員 (registration operator)」

主要參與者：RA

設計範圍：夜間收貨登記軟體

目標等級：使用者目標等級

主要成功情節：

1. RA 收到從貨運公司 (transport company, TC) 送來的貨箱 (箱子會有貨箱 ID, 如果箱子裡面還有袋子, 還會有袋子 ID) 並打開它。
2. RA 證實貨箱 ID 與 TC 的登錄 ID。
3. RA 可能會在送貨者給的送貨單上簽名。
4. RA 把送抵的貨箱登錄到系統裡面, 裡面會記錄：

RA ID

日期、時間

貨箱 ID

貨運公司

(送貨者姓名?)

袋子編號 (有沒有袋子 ID?)

<估算價值>

5. RA 把袋子從貨箱中拿出來。

擴充情節：

- 2a. 貨箱 ID 跟貨運公司 ID 不吻合。
- 4a. 火警聲響起, 中斷登錄作業。
- 4b. 電腦當機。
把貨箱放在桌上, 等電腦重新開機。

變異情節：

- 4'. 送貨者 ID 是可有可無的。
- 4''. 估算價值是可有可無的。
- 5'. RA 把袋子留在貨箱裡面。

↑ 英 6

1.2 你用的使用案例不等於我的

使用案例是一種寫作的形式, 我們可以把它用在不同的情況下, 包括：

- ◆ 用它描述企業中的工作流程。
To describe a business's work process.
- ◆ 把焦點放在未來的軟體系統需求上, 不過我們不一定要把需求說明文字寫在裡面 (譯註: 我們可以把它當作促進需求討論的一種工具, 而把需求記錄在其他種類的需求文件中)。

To focus discussion about upcoming software system requirements, but not to be

the requirements description.

- ◆ 作為系統的功能需求。

To be the functional requirements for a system.

- ◆ 記錄系統的設計方式（譯註：這裡的設計並非是指系統「設計」，而是寫出子系統的互動情形）。

To document the design of the system.

- ◆ 使用案例可用在小型、緊密編組的組織，也可用在大型、分散的組織。

To be written in a small, close-knit group, or in a large or distributed group.

在各種情況下，使用案例的寫作風格會有點不同。下面是最常見的使用案例次要寫作型態（subform of use case），它們都是針對不同目的調整的。

- ◆ 緊密編組的組織在蒐集需求或稍微大一點的組織在討論未來需求時，都會用非正式格式來寫使用案例，而不會用正式格式來寫，後者是大型、地理位置分散或有正式格式傾向的團隊才會採用的。非正式、簡化過的使用案例範本可以讓使用案例的寫作過程快一點（我們稍後會看到更多用非正式格式所寫的範例）。使用案例 1 到使用案例 3 都是用正式、完整的使用案例範本所寫的，而且動作步驟有編號過。使用案例 4 則是用非正式格式所寫的範例。

A close-knit group gathering requirements, or a larger group discussing upcoming requirements, will write casual as opposed to fully dressed use cases, which are written by larger, geographically distributed, or formally inclined teams. The casual form "short-circuits" the use case template, making the use cases faster to write (see more on this later). Use Cases 1 through 3 are fully dressed, using the full use case template and step-numbering scheme. An example of casual form is shown in Use Case 4.

- ◆ 負責企業流程的人會寫出企業使用案例（business use case）說明企業的運作方式，而硬體或軟體開發團隊則會用系統使用案例（system use case）寫出他們要的需求。設計團隊也可能寫出其他系統使用案例，以記錄他們的設計方式，或者寫出由系統分解下來、比較小的子系統需求。

Business process people write business use cases to describe the operations of their business, while a hardware or software development team writes system use cases for their requirements. The design team may write other system use cases to document their design or to break down the requirements for small subsystems.

- ◆ 在不同時期，根據當時的觀點所需要的目標等級，寫使用案例的人可寫出多重目標或目標摘要（summary goal），也可寫出單一目標或使用者目標（user goal），或者只寫出部分使用者目標或子功能（sub-function）。說明使用案例中所描述的內容是哪個目標等級是件很重要的事，我的學生想出兩種不同的方式來呈現目標等級：一種是用海平面的相對高度（高於海平面、海平面與低於海平面）來表示，另一種則是用顏色加以區分（白色、藍色與靛藍）。

Depending on the level of view needed at the time, the writer will choose to describe a multi-sitting, or summary, goal; a single-sitting, or user goal; or a part of a user goal, or subfunction. Communicating which of these is being described is so important that my students have come up with two different gradients to describe them: by height relative to sea level (above sea level, at sea level, underwater), and by color (white, blue, indigo).

- ◆ 不論系統是企業流程或電腦系統，當某人要寫出他所要設計的新系統需求時，都會寫出黑箱式使用案例（black-box use case）——裡面不會討論到系統內部。企業流程設計者可能會寫出白箱式使用案例（white-box use case），以秀出公司或組織是如何執行內部流程的。技術開發團隊則可能會用黑箱式使用案例來記錄他們所要設計的系統操作情境，而用白箱式使用案例記錄系統內的運作方式。

↑ 英 7

Anyone writing requirements for a new system to be designed, whether business process or computer system, will write black-box use cases—those that do not discuss the innards of the system. Business process designers will write white-box use cases, showing how the company or organization runs its internal processes. The technical development team might do the same to document the operational context for the system they are about to design, and they might write white-box use cases to document the workings of the system they just designed.

讓人驚訝的是使用案例這種寫作形式竟然可用在各種不同情況下，不過它也會造成我們混淆。由於我們的使用案例可能根據幾種不同目的所寫出來的，所以有可能會發現到好像有幾個自己坐在一起，爭論所寫出來的內容。隨著時間過去，我們可能會遇到幾個特性混合在一起的情形。

針對不同的使用案例寫作目的，如果想用一種方式同時談論各種不同形式的使用案例，那麼我們可能會毀了整本書。我所能做的就是先說明議題，大家再由範例去理解它們。

It is wonderful that the use case writing form can be used in such varied situations, but it is confusing. Several of you sitting together are likely to find yourselves disagreeing on some matter of writing, just because your use cases are for different purposes. And you are likely to encounter several combinations of those characteristics over time.

Finding a general way to talk about use cases, while allowing all those variations, will plague us throughout the book. The best I can do is outline the issue now and let the examples speak for themselves.

你可能想試試看自己能否判斷出本章中一些使用案例的寫作目的為何。使用案例 1、3 與 5 都是為系統需求而寫的，所以它們都是採用正式格式、黑箱式、設計範圍為系統、屬於使用者目標等級的使用案例。使用案例 4 也是如此，不過它是

用非正式格式所寫的。至於使用案例 2 則是說明企業流程、為情境目的而寫的使用案例，它是正式格式、黑箱式，屬於目標摘要等級的使用案例。

使用案例格式的最大差異在於它們有多「正式」。請考慮下面兩種非常不同的情況：

You may want to test yourself on the use cases in this chapter. Use Cases 1, 3, and 5 were written for system requirements purposes, so they are the fully dressed, black-box, system type, at the user-goal level. Use Case 4 is the same, but casual, not fully dressed. Use Case 2, written as the context-setting use case for business process documentation, is fully dressed, black-box, and at the summary level.

The largest difference between use case formats is how "dressed up" they are.

Consider these quite different situations:

- ◆ 有一個開發團隊準備開發大型、處理具關鍵性任務的專案軟體。他們覺得為額外的儀式性付出成本是值得的，所以(a)使用案例範本需要寫的比較長、詳細一點、(b)寫使用案例的開發團隊必須採用相同的使用案例風格，以減少含糊不清的地方或造成誤解、(c)審查時要嚴謹一些，需要細看使用案例中是否有漏掉或寫得不清楚的地方。他們不太容許有錯誤發生，並且決定要減少不同人寫使用案例時的差異。

A team is working on software for a large, mission-critical project. They decide that the extra ceremony is worth the extra cost, so (a) the use case template needs to be longer and more detailed, (b) the writing team should write in the same style to reduce ambiguity and misunderstanding, and (c) the reviews should be tighter to more closely scrutinize the use cases for omissions and ambiguities. Having little tolerance for mistakes, they decide to reduce tolerances (variation between people) in the use case writing as well.

- ◆ 由三到五人所組成的開發團隊負責建構某個系統，這個系統可能會造成的最大傷害是失去舒適感，我們可以很容易地用電話來補救它。他們認為所有的儀式性都是浪費時間、精力與金錢。因此，他們選擇(a)簡單一點的範本、(b)可容忍不同寫作風格差異、(c)比較少的審查工作，同時也能原諒一些錯誤存在。寫作結果中的錯誤與遺漏處可以靠其他的專案機制來補救，例如透過開發團隊成員或使用者間的交談。在溝通用的文件上，他們容許比較多的錯誤發生、比較非正式的寫作方式，以及不同人的寫作差異。

A team of three to five people is building a system whose worst damage is the loss of comfort, easily remedied with a phone call. They consider all the ceremony a waste of time, energy, and money. They therefore choose (a) a simpler template, (b) to tolerate more variation in writing style, and (c) fewer and more forgiving reviews. The errors and omissions in the writing are to be caught by other project mechanisms, probably conversations among teammates and with users. They can tolerate more errors in their written communication

and so more casual writing and more variation between people.

這兩種作法都沒有錯。選擇哪一種作法完全要根據專案而定。作為一個方法論者，這是在過去五年來所學到的最重要一課。當然，我們已說過「單一尺寸是無法適合所有情況」這句話很多年了，不過怎樣把它變成一個具體建議還是一個秘密。

Neither is wrong. Such choices must be made on a project-by-project basis. This is the most important lesson that I, as a methodologist, have learned in the last five years. Of course we have been saying, "One size doesn't fit all" for years, but just how to translate that into concrete advice has remained a mystery.

有問題的地方可能是使用不必要的精確度與嚴格程度，而浪費很多的專案時間與精力。就像 Jim Sawyer 在一封討論相關問題的電子郵件中所寫的：「...只要範本的正式程度沒有高到讓你覺得掉入設計世界中的無底深淵就好。發現這個情形時，我會去掉所有的儀式性，而用餐巾紙來描繪出比較概略的東西。」

↑ 英 8

The mistake is getting too caught up in precision and rigor when they are not needed, which will cost your project a lot in time and energy. As Jim Sawyer wrote in an email discussion, "... as long as the templates don't feel so formal that you get lost in a recursive descent that worm-holes its way into design space. If that starts to occur, I say strip the little buggers naked and start telling stories and scrawling on napkins."

從上面我們可以得到一個結論：只有一種使用案例範本是不夠的。我們至少要有兩種範本：低儀式性專案用的非正式格式與高儀式性專案用的正式格式。任何一個專案可針對自己的情況採用其中一種範本。下面兩個使用案例中的內容是一樣的，不過分別是用兩種正式程度不同的格式所寫的。

I have come to the conclusion that just one use case template isn't enough. There must be at least two: a casual one for low-ceremony projects and a fully dressed one for higher-ceremony projects. Any one project will adapt one of the two forms for its situation. The next two use cases are the same but written in the two styles.

使用案例 4 買東西（非正式格式版本）

申請人開始一項申請，並且把申請資料送到她或他的同意人手上。同意人會檢查金額是否在預算之內、檢查貨物價格、把申請資料填寫完，並且把它交給採購人。採購人會檢查倉庫庫存、找尋貨物的最好供應商。授權者會證實同意人的簽名。採購人完成下訂單請求、對供應商發出 PO。供應商送出要送過來的貨品、拿到已送貨收據（超出討論中系統範圍）。收貨人登錄貨品送達、把貨品送給申請人。申請人標示已經收到東西。

在收到貨品之前，申請人都可變動或取消申請。如果是取消的話，會把申請從任何進行中的處理事項移除（要把它從系統移除嗎？）。如果降價的話，還是把它

留在處理事項不變。如果價格升高的話，把它送回給同意人。

使用案例 5 買東西（正式格式版本）

主要參與者：申請人

情境中的目標：申請人藉由系統買東西，得到它。不過裡面不包括付款動作。

設計範圍：企業 — 整個購買機制、可能是電子或非電子的、就像我們在公司看到的一樣。

目標等級：目標摘要等級

關係人與利益：

申請人：希望訂購東西、可用很簡單的方式完成。

公司：希望控制花費，不過還是允許必要物品的購買。

供應商：希望能收到任何已送出貨品的貨款。

事先條件：無。

最小事後保證：每個被送出的訂單都要被有效的授權者認可過。訂單可被追蹤，所以公司收到的有效貨品才会有帳單。

成功事後保證：申請人拿到貨品、正確的花費被記入借方。

觸發事件：申請人決定要買某個東西。

主要成功情節：

1. 申請人：啟動一項申請。
2. 同意人：檢查金額在預算內，檢查貨物價格、完成要發出的申請。
3. 採購人：檢查倉庫庫存，找尋貨品的最佳供應商。
4. 授權者：證實同意人的簽名。
5. 採購人：完成下訂單請求、對供應商發出 PO。
6. 供應商：送出我們要收到的貨品，取得已送出的收據。

↑ 英 9

擴充情節：

- 1a. 申請人不知道供應商或價格：把這些部分空白，繼續下一步。
- 1b. 在收到貨品之前，申請人都可改變或取消申請：
取消它會把它從進行中的處理事項移除（要把它從系統移除嗎？）。
如果降價的話，還是把它留在處理事項不變。
如果價格升高的話，把它送回給同意人。
- 2a. 同意人不知道供應商或價格：把這些部分空白，留給採購人填或退回。
- 2b. 同意人不是申請人的經理：如果同意人願意簽名的話，還是 OK。
- 2c. 同意人回絕申請：把它送回給申請人，請他改變申請或取消它。
- 3a. 採購人在倉庫庫存中找尋申請所需貨品：發貨、根據現有的量扣掉申請量，然後繼續下一步。
- 3b. 採購人填入之前沒有填的供應商與價格：把申請重送給同意人。

4a. 授權者否決同意人的認可：把申請資料送回給申請人，並且把它從進行中處理事項移除。（這代表什麼意思呢？）

5a. 申請資料中包括好幾個供應商：採購人做出多份 PO。

5b. 採購人合併好幾份申請資料：屬於同一個流程，不過在 PO 上註明被合併的申請資料為何。

6a. 供應商沒有準時送來東西：系統發出未送貨警訊。

7a. 只有部分貨品送到：收貨人在 PO 上註明已送到的部分貨品，然後繼續。

7b. 針對包含多份申請資料的 PO，當部分貨品送到時，收貨人把數量分配到某幾份申請上，然後繼續。

8a. 貨品不對或品質不佳：申請人拒收送來的貨品。（這代表什麼意思呢？）

8b. 申請人已經離職：採購人向申請人的經理確認，重新指定申請人，或者退貨並取消申請。

技術或資料變異情形清單：無。

優先順序：不定。

發行版本：包含好幾個。

回應時間：不定。

使用頻率：3/天

跟主要參與者溝通管道：網路瀏覽器、電子郵件系統或類似的東西。

次要參與者：供應商。

↑ 英 10

跟次要參與者溝通管道：傳真機、電話、汽車。

開放議題：

何時要把已取消的申請資料從系統中刪除？

取消申請需要怎樣的權限？

誰能變更申請內容？

要維護怎樣的申請變更歷史？

當申請人拒收送來的貨品時，會發生什麼事？

申請單跟訂單的作用有何不同？

下訂單時該如何查看並使用內部庫存？

我希望大家都很清楚只表示「我們在專案中會寫出使用案例」是不夠的。同時，也要很清楚知道在任何建議或開發流程的定義中簡單說明「要寫使用案例」也是不完整的。對某個專案有效的使用案例，對其他專案來說可能是無效的。大家必須說明：使用案例是用正式格式或非正式格式所寫的、範本中有哪些部分是一定要寫的，以及允許不同寫作者間可以有多大的差異存在。

I hope it is clear that simply saying, "We write use cases on this project" does not say very much, and that any recommendation or process definition that simply says, "Write use cases" is incomplete. A use case valid on one project is not valid on another project. More must be said about whether fully dressed or casual use cases are

being used, which template parts and formats are mandatory, and how much tolerance across writers is permitted.

在 *Software Development as a Cooperative Game* (Cockburn, 2001) 一書中，我們有完整討論到不同專案可容忍的差異所在。不過，學習如何寫使用案例並不需要做這麼詳細的討論。只要把寫作技巧跟使用案例品質與專案標準的意義分開就可以了。

The full discussion of tolerance and variation across projects is described in *Software Development as a Cooperative Game* (Cockburn, 2001). We don't need the full discussion to learn how to write use cases. We do need to separate the writing technique from use case quality and project standards.

當人們在寫使用案例時，「寫作技巧」是時時會用到的思考方式或動作。本書把重點放在寫作技巧上，也就是：如何思考、如何寫出句子、用怎樣的步驟寫出使用案例。關於寫作技巧，很幸運的一點是：它們大部分都跟專案的規模大小無關。熟練某個寫作技巧的人，可以把它用在大型或小型專案上。

"Techniques" are the moment-to-moment thinking or actions people use while constructing use cases. This book is largely concerned with technique: how to think, how to phrase sentences, in what sequence to work. The fortunate thing about techniques is that they are largely independent of the size of the project. A person skilled in a technique can apply it on projects both large and small.

「品質」說明我們該如何判別使用案例是否符合它們的寫作目的。在本書中，針對使用案例中的每個部分、是否橫跨不同使用案例或針對不同目的寫出使用案例，我都會說明自己曾看過的最好寫法。不過，最後，當我們在評估使用案例的品質時，主要還是根據自己所選擇的寫作目的、差異容忍度與儀式性程度而定的。"Quality" says how to tell whether the use cases that have been written are acceptable for their purpose. In this book, I describe the best way of writing I have seen for each use case part, across use cases, and for different purposes. In the end, though, the way you evaluate the quality of your use cases depends on the purpose, tolerance, and amount of ceremony you choose.

「使用案例寫作標準」說明專案中成員在寫使用案例時所同意遵守的東西。在本書中，我會討論不同的合理標準、給大家看各種不同範本，以及不同的句子或標題風格。同時，我也會提供一些寫作建議。不過，要不要訂定或採用這些標準是組織或專案的事，大家可自行決定要用多強烈的方式來強化這些標準。

在本書大部分內容中，我都在處理最需要關心的問題 — 寫出精確的需求。下面是我的客戶 Steve Adolph（他是一位顧問）對「使用案例可用來發掘需求，不只是記錄需求而以」的見證。

↑ 英 11

"Standards" say what the people on the project agree to when writing their use cases. In this book, I discuss alternative reasonable standards, showing different templates

and different sentence and heading styles. I come out with a few specific recommendations, but ultimately it is for the organization or project to set or adapt the standards and to decide how strongly to enforce them.

In most of this book, I deal with the most demanding problem-writing precise requirements. In the following eyewitness account, Steve Adolph, a consultant, describes using use cases to discover requirements rather than to document them.

◆ Steve Adolph：在新領域中「發掘」需求

一般來說，使用案例是用來捕捉功能需求、建立模型的。跟傳統、跟購物清單一樣的需求比起來，人們發現到這種跟故事一樣的格式是很容易了解的。我們真的可藉此了解系統該支援什麼東西。

不過，如果沒人知道系統要支援什麼東西時該怎麼辦？企業流程在自動化時通常會被改變。例如，印表機工業最近歷經了一次最大的變動，有人發明直接設定版面／列印的技術。在此之前，設定印表機版面是很花費功夫、需要好幾個步驟的事。直接設定版面／列印讓我們可以直接把文字處理器中的文件送出去列印。

針對直接設定版面系統這個工作流程，身為分析師的你該如何蒐集這種全新東西的需求呢？你可以先找到現存系統中的使用案例，再尋找系統的參與者與服務。然而，這樣做只能告訴我們現有系統為何。因為還沒有人有新工作流程的經驗，所以全部的領域專家也都要跟著你一起學。你需要同時設計新企業流程與新軟體。這真是一件很棒的事！不過，你該如何找到新事務的蛛絲馬跡呢？是以現存模型為基礎，詢問「有什麼需要」嗎？你會得到「什麼都可能」的答案。

當你用使用案例來記錄需求時，必須有人先創造出系統願景。我們只是用使用案例簡單描述出系統願景，讓大家很清楚了解它而已。然而，當你需要「發掘」需求時，代表你正在創造新的系統願景。

我們可以把使用案例當成腦力激盪時的工具，問這樣的問題：「假設你有了新科技，對使用案例目標而言，使用案例中的哪個動作步驟將不再具有價值？」這時候，我們必須創造出一個新的故事，說明參與者如何達到它們的目標。主要參與者的目標還是一樣，不過某些支援性參與者的目標則可能會消失或改變。

如果我們採用浮潛式方案（dive-and-surface approach）的話，就可產生涵蓋範圍很廣、高階的模型，描述我們是如何思考新系統的工作方式。因為這是一個新領域，所以請讓事情保持簡單。發掘出主要成功情節可能會像什麼東西之後，請跟原有的領域專家討論這樣的情節是否可行。

接下來，我們會潛入（深入）某個使用案例中的細節，考慮不同替代方案。請善用人們容易理解敘事情節這個事實，用這些情節找出被漏掉的需求。閱讀使用案例中的某個動作步驟，然後問這樣的問題：「嗯！如果客戶端要用硬拷貝而不用數位拷貝方式的話，結果會怎樣？」跟嘗試直接由腦海中蒐集系統如何運作的模型比起來，這樣做會比較容易。

最後，我們會浮出來到使用者目標等級。之前所了解的細節組合起來後，會改變原先使用者目標中的情節嗎？調整模型之後，再潛入（深入）另一個使用案例。我的經驗是：用使用案例去發掘需求可讓我們獲得高品質的功能需求。這樣的需求比較有組織性、也比較完整。

↑ 英 12

1.3 需求與使用案例

如果你把使用案例當成需求，心裡請記得兩件事：

- ◆ 它們真的是需求。你不用把使用案例轉成其他形式的行為需求。只要寫的適當，它們可精確表達出系統應該要做的細節。

They really are requirements. You shouldn't have to convert them into some other form of behavioral requirements. Properly written, they accurately detail what the system must do.

- ◆ 它們不代表全部需求。裡面不會寫出詳細的外部介面、資料格式、企業規則與複雜公式，只會有你需要蒐集的全部需求中的一部分 — 它雖然是非常重要的一部分，不過仍然不能代表全部需求。

They are not all of the requirements. They don't detail external interfaces, data formats, business rules, and complex formulae. They constitute only a fraction (perhaps a third) of all the requirements you need to collect—a very important fraction but a fraction nonetheless.

每個組織會蒐集它所需的合適需求。我們甚至可找到適合作為需求的一些標準。不管是什麼，使用案例都只代表全部被記錄需求中的一部分。

下面的需求大綱是我所發現的一種很有用大綱，它是 Suzanne Robertson 與 Atlantic 系統指引中的範本，我把它修正後得到下面的需求大綱。這些指引可在他們的網站與 *Managing Requirements* (Robertson 與 Robertson, 1999) 一書中找到。他們的範本完整到讓人害怕，所以我就以他們的範本為基礎稍加裁減，變成下面的大綱。不過，這份大綱對我曾遇過的大部分專案來說還是太大了，所以如果有需要的話，還可以進一步裁減它。不論這份範本的大小如何，裡面還是問了一些有趣的問題，例如「當系統遭遇失敗情況時，由人所執行的備援方式是什麼？」或「有什麼政治考量會引發任何需求嗎？」

Every organization collects requirements to suit its needs. There are even standards available for requirements descriptions. In any of them, use cases occupy only one part of the total requirements documented.

The following requirements outline is one that I find useful. I adapted it from the template that Suzanne Robertson and the Atlantic Systems Guild published on their Web site and in the book *Managing Requirements* (Robertson and Robertson, 1999). Their template is intimidating in its completeness, so I've cut it down to the form

shown in the outline that follows, which I use as a guideline. This is still too large for most projects I encounter, and so I tend to cut it down further as needed. Whatever its size, it asks many interesting questions that otherwise would not be asked, such as "What is the human backup to system failure?" and "What political considerations drive any of the requirements?"

雖然本書的角色不是訂出你要交付的需求標準，不過我曾遇過有許多人都從未看過類似的需求大綱。爲了讓大家了解一下需求標準，我在下面秀出這份需求大綱，告訴你在整個需求中使用案例所佔的地位，也讓你知曉使用案例中是無法記錄所有需求的，它只能說明需求中的行爲部分，以代表系統所需功能。

While it is not the role of this book to standardize your requirements deliverable, I have run into many people who have never seen a requirements outline. I pass along this one for your consideration. Its main purpose is to illustrate the place of use cases in the overall requirements and to make the point that use cases will not hold all the requirements, but only describe the behavioral portion, the required function.

看起來很嚇人的需求文件大綱

第一章、系統目的與專案範圍

- 1a. 什麼是整體的設計範圍與目標？
- 1b. 關係人（誰會關心？）
- 1c. 什麼東西在專案範圍內，什麼不在專案範圍內？

第二章、所用術語／字彙

第三章、使用案例

- 3a. 主要參與者跟他們對系統的一般性目標
- 3b. 企業使用案例（操作系統時的概念）
- 3c. 系統使用案例

↑ 英 13

第四章、所使用技術

- 4a. 這個系統的技術需求是什麼？
- 4b. 這個系統跟什麼系統之間會有介面、會有怎樣的需求？

第五章、其他需求

- 5a. 開發流程
 - Q1. 誰是專案的參與人員？
 - Q2. 開發流程會反映出什麼價值（簡單、立即反應、快速或有彈性）？
 - Q3. 使用者與贊助者希望得到什麼回饋或專案可見性？
 - Q4. 什麼可以用買的、什麼必須自行建構、什麼是我們的競爭對象？
 - Q5. 需要哪些其他的開發流程需求（測試、安裝等）？
 - Q6. 專案運作時會有什麼相依性？
- 5b. 企業規格
- 5c. 效能

- 5d. 操作性、安全性與文件
- 5e. 系統的使用與可用性
- 5f. 系統的維護與可攜性
- 5g. 無法解決或稍後再解決的東西

第六章、跟人的備援方式、法律、政治與組織相關的議題

- Q1. 系統操作時發生失敗情形時，人的備援方式為何？
- Q2. 有哪些法律或政治需求？
- Q3. 完成系統時要考慮的人性推論結果為何？
- Q4. 會有怎樣的訓練需求？
- Q5. 系統在人的環境中，會有怎樣的假設、相依性？

要注意的是：使用案例只是我們所列出的需求大綱中的第三章。它們不能代表全部需求 — 只能代表行為需求，不過它可代表全部行為需求。企業規則、字彙表、效能目標、開發流程需求，以及許多其他東西都不在行為範疇之內。使用案例會自己獨立成一章（請參見圖 1.1）（譯註：使用案例以外的其他需求通常都記錄在專案願景、字彙表或輔助規格書中。）

The thing to note is that use cases occupy only Chapter 3 of the requirements. They are not all of the requirements—they are only the behavioral requirements but they are all of the behavioral requirements. Business rules, glossary, performance targets, process requirements, and many other things do not fall into the category of behavior. They need their own chapters (see Figure 1.1).

把使用案例當成連接專案用的結構

使用案例可以幫我們連接許多其他需求細節。它提供我們當腳架的材料，以連接需求中不同部分的資訊，幫助我們連接使用者相關說明資訊、企業規則與資料格式等需求。

除了需求文件之外，使用案例也可幫我們組織專案開發資訊，例如發行日期、開發團隊、優先順序與專案開發狀態等。此外，使用案例也可幫助設計團隊追蹤特定結果，特別是使用者介面設計與系統測試。

↑ 英 14

Use cases connect many other requirements details. They provide a scaffolding that connects information in different parts of the requirements and they help crosslink user profile information, business rules, and data format requirements.

Outside the requirements document, use cases help structure project planning information such as release dates, teams, priorities, and development status. Also, they help the design team track certain results, particularly the design of the user interface and system tests.

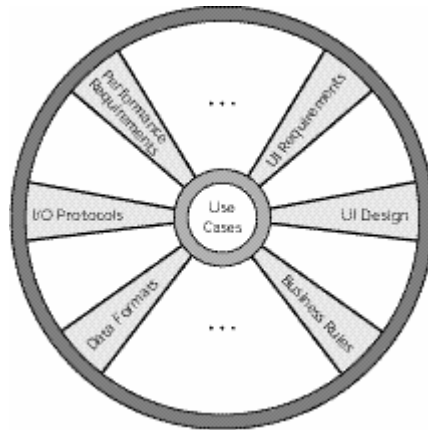


圖 1.1 需求的「輪軸與輪輻」模型

除了功能需求之外，其他需求雖然不會寫在使用案例中，不過所有需求都會跟使用案例連結在一起。如圖 1.1 所示，使用案例就像是輪子的軸心，而其他需求則像不同方向的軸輻。也因為如此，大家會把使用案例視為需求的核心元素，甚至是專案開發流程的核心元素（譯註：這樣的開發流程就是以使用案例驅動的開發流程【use-case-driven process】，例如 RUP）。

While not in the use cases, all these requirements are connected to them. The use cases act as the hub of a wheel, as in Figure 1.1, and the other information acts as spokes leading in different directions. It is for this reason that people seem to consider use cases to be the central element of the requirements or even the central element of the project's development process.

1.4 什麼時候使用案例會更有價值

使用案例之所以很流行，絕大部分是因為我們會在使用案例中寫出具有一致性的敘事情節，以說明使用中的系統行為究竟為何。系統的使用者可藉此看到新系統中有什麼東西，也可及早反應，以調整或拒絕某些敘事情節（例如開發人員可以問使用者說：「你的意思是我們必須這樣做嗎？」）。然而，使用案例不只可帶給我們這些價值，而且這些也不是它所帶來的最重要東西。

Use cases are popular largely because they tell coherent stories about how the system will behave in use. The users of the system get to see just what this new system will be. They get to react early to fine-tune or reject the stories ("You mean we'll have to do what?"). That is, however, only one of the ways use cases contribute value and possibly not the most significant.

使用案例創造價值的第一個時間點是：它把系統要支援的東西以使用者目標加以命名，並且把這些使用者目標集合起來變成一份清單。清單中會秀出系統要做什麼、設計範圍，以及系統的存在目的。在專案中，使用案例可作為不同關係人間

的溝通工具。

譯註：對很多關係人來說，只要花幾分鐘就可以了解使用案例。因此，它很適合用來作溝通工具。

The first moment at which use cases create value is when they are named as user goals that the system will support, and are collected into a list. This list announces what the system will do, revealing the scope of the system, its purpose in life. It becomes a communication device between the different stakeholders on the project. 使用者代表、主管、專家級的開發人員與專案經理會檢查這份目標清單，而且專案經理也可藉此評價出系統的成本與複雜性。他們會協商出要先建構什麼功能、怎麼組織開發團隊等。我們能夠以這份清單為基礎，再加上複雜性、成本、時間與狀態等度量屬性。它可蒐集涵蓋整個專案生命週期的各種不同資訊。

↑ 英 15

The list of goals will be examined by user representatives, executives, expert developers, and project managers, who will estimate the cost and complexity of the system starting from it. They will negotiate which functions get built first and how the teams are to be set up. The list is a framework to which to attach complexity, cost, timing, and status measures. It collects diverse information over the life of the project. 使用案例第二個特別有價值的時候是：使用案例撰寫人員可以用腦力激盪方式找出成功情節中可能出錯的地方、列出這些地方，並記錄系統該如何回應它們。在這個時候，開發團隊可能會發現到一些很讓人驚訝的東西或需求提供者沒想到的東西。

The second particularly valuable moment is when the use case writers brainstorm all the things that could go wrong in the successful scenario, list them, and begin documenting how the system should respond. At that moment, the team is likely to uncover something surprising, something that they or their requirements givers had not thought about.

當我開始覺得寫使用案例很無聊時，我會堅持下去直到找到失敗情況為止。在記錄失敗情況的處理方式時，我經常可發現新的關係人、系統、目標或企業規則。當我們在一起工作、想辦法找出好的方式以處理這些情況時，經常可看見企業專家聚在一起或打電話，以了解系統在這個時候該做些什麼事。

When I get bored writing a use case, I hold out until I get to the failure conditions. There I regularly discover a new stakeholder, system, goal, or business rule while documenting the failure handling. As we work out how to deal with one of these conditions, I often see the business experts huddled together or making phone calls to resolve what the system should be doing here.

如果我們沒有針對使用案例中的每個動作步驟，一個個用腦力激盪方式找出所有的失敗情形，等到程式設計師嘗試寫某一段程式時，可能才會發現到系統中許多的錯誤情況。這時候發現新功能或新企業規則已經太晚了，因為企業專家通常已

經走了，而且時間也很急迫，於是程式設計師很可能會自行想像，而不研究系統想要的行為究竟是什麼。

對於寫使用案例時只寫出一段描述的人，因為他們寫得很少，所以可節省很多時間，馬上得到使用案例的好處。另一方面，嘗試寫出失敗情況處理方式的人，由於他們可及早發現一些細微需求，可節省許多開發時間。

Without discrete use case steps and failure brainstorming, many error conditions go undetected until some programmer discovers them while typing a code fragment. That is very late to be discovering new functions and business rules. The business experts usually are gone and time is pressing, and so the programmers type whatever they think up at the moment instead of researching the desired behavior.

People who write one-paragraph use cases save a lot of time by writing so little and already reap one of the benefits of use cases. People who perservere through failure handling save a lot of time by finding subtle requirements early.

1.5 管理自己的精力

保持你的精力或至少管管自己的精力。如果一開始就寫出使用案例中的所有細節，我們將無法適時從某個主題轉移到另一個主題。如果剛開始時我們只先寫下大綱，然後再寫出每個使用案例中的必要部分，我們就能夠：

Save your energy. Or at least manage it. If you start writing all the details at the first sitting, you won't move from topic to topic in a timely way. If you write down just an outline to start with and then write just the essence of each use case, you can

- ◆ 讓關係人有機會及早正確、深入地了解使用案例的優先順序。
Give your stakeholders a chance to offer correction and insight about priorities early.
- ◆ 把工作分給不同團隊做，以增加平行性與生產率。
Permit the work to be split across multiple groups, increasing parallelism and productivity.

人們通常會說：「給我五萬英尺遠的看法」、「先給我一個概述」或「我們稍後再寫出細節」。他們的意思是說：「現在先用精確度低的方式工作，稍後再寫出細節」。精確度代表你要寫出多少東西。當你說「顧客想租錄影帶」時，雖然沒有用到許多字，不過卻可以給聽者一個很重要的基本想法。當你秀出系統中該有的所有目標名稱清單時，等於用一小撮字就讓關係人知道一大堆資訊。

↑ 英 16

People often say, "Give me the 50,000-foot view," "Give me just a sketch," or "We'll add details later." They mean "Work at low precision for the moment; we can add more later."

Precision is how much you care to say. When you state, "A 'Customer' will want to

rent a video," you are not using very many words, but you are actually communicating a great deal to your readers. When you show a list of all the goals that your proposed system will support, you have given your stakeholders an enormous amount of information from a small set of words.

精確度不等於正確性。如果有人告訴你「 $\pi=4.141592$ 」，那麼他們就有很高的精確度，不過正確性卻很低（譯註：4.141592 精確到小數點下第 6 位，所以精確度高，不過卻不正確，因為正確值=3.141592）。如果他們說「 π 約等於 3」，那麼他們的精確度雖然不高，不過說多少就正確多少。使用案例也是同樣的概念。

最後，我們可能會寫出每個使用案例的細節，以增加它們的精確度。如果一開始最原始、低精確度的目標是錯的（不正確的），那麼花時間寫出精確的描述只是在浪費力氣而已。我們先要有正確的目標清單，接下來才花數十個工作人月，寫出完全詳述的一組使用案例。

根據所需花費精神與每個階段所具備的一定價值，我會把寫使用案例分成四種精確度：

Precision is not the same as accuracy. If someone tells you, " π is 4.141592," they are using a lot of precision. They are, however, quite far off or inaccurate. If they say, " π is about 3," they are not using much precision (there aren't very many digits), but they are accurate for as much as they said. The same ideas hold for use cases. You will eventually add details to each use case, increasing precision. If you happen to be wrong (inaccurate) with your original, low-precision statement of goals, then the energy put into the high-precision description is wasted. Better to get the goal list correct before expending the dozens of work-months of energy required for a fully elaborated set of use cases.

I divide the energy of writing use cases into four stages of precision, according to the amount of energy required and the value of pausing after each stage:

1. **參與者與其目標。**列出系統要支援的參與者跟他們的目標。請審查這份清單的正確性與精確度。定出目標的優先順序，並且以不同發行順序指定給開發團隊。這時候，你將具有最初級精確度。

Actors and goals. List what actors and which of their goals the system will support. Review this list for accuracy and completeness. Prioritize and assign goals to teams and releases. You now have the functional requirements to the first level of precision.

2. **使用案例簡介或主要成功情節。**對於你有興趣、選出來的使用案例，我們會概述它的主要成功情節。請審查這些草稿，確認系統所提供的東西真的是關係人的利益。這是功能需求的第二級精確度。我們可以很快寫出這些草稿內容，不像接下來的兩個等級要花費許多功夫。

Use case brief or main success scenario. For the use cases you have selected to pursue, sketch the main success scenario. Review these in draft form to make

sure that the system really is delivering the interests of the stakeholders you care about. This is the second level of precision on the functional requirements. It is fairly easy material to draft, unlike the next two levels.

3. **失敗情形。**把主要成功情節寫完整，並且用腦力激盪方式找出所有會發生的失敗情形。在決定系統該如何處理失敗情形之前，我們要先確認自己已經大約列出這份清單了。寫出失敗處理方式比列出失敗情形還要花不少精神。大家通常會在列出所有失敗情形之前，馬上就花太多精神寫出失敗處理方式。
Failure conditions. Complete the main success scenario and brainstorm all the failures that could occur. Draft this list completely before working out how the system must handle them. Filling in the failure handling takes much more energy than listing the failures. People who start writing the failure handling immediately often run out of energy before listing all the failure conditions.
4. **失敗情形的處理方式。**寫出系統將如何回應每個失敗情形。這件事通常很難處理、麻煩、讓人吃驚。之所以讓人吃驚，主要是因為寫處理方式時，經常會出現跟某個很模糊的企業規則有關的問題，或者突然發現需要支援的新參與者或目標。

Failure handling. Write how the system is supposed to respond to each failure. This is often tricky, tiring, and surprising work. It is surprising because quite often a question about an obscure business rule will surface during this writing, or the failure handling will suddenly reveal a new actor or new goal that needs to be supported.

大部分專案的時間與人力都不足。因此，我們應該把「管理工作的精確度」這件事列入專案的優先考量當中。我強烈推薦大家按照這裡所描述的精確度依序工作。

Most projects are short on time and energy. Managing the precision level to which you work should therefore be a project priority. I strongly recommend working in the order given here.

1.6 以系統使用情形簡述暖身

系統使用情形簡述 (usage narrative) 代表某個特定情況下，實際運作的一個使用案例實例 — 這是參與者使用系統時，一個高度特殊化的例子。它不等於使用案例(譯註：如果把使用案例視為參數化類別【parameterized class】或範本【template】的話，系統使用情形簡述就像繫結元素【bound element】)，而且在大部分專案中，我們不會把它放在正式需求文件中。然而，這是一種非常有用的機制，值得你花時間描述它、寫下來。

↑ 英 17

A usage narrative is a situated example of the use case in operation—a single, highly

specific example of an actor using the system. It is not a use case, and in most projects it does not survive into the official requirements document. However, it is a very useful device that is worth my describing and your writing.

在專案一開始時，我們或企業專家可能對使用案例的寫作只具有一點經驗或對系統的詳細操作方式沒有完整了解它。爲了熟悉這些內容，我們可以先簡單描述它。換言之，先說明一個參與者在它的生命中某一天的一段時間中是如何過的。On starting a new project, you or the business experts may have little experience with use case writing or may not have thought through the system's detailed operation. To become comfortable with the material, sketch out a vignette, that is, a few moments in the day of the life of one of the actors.

在我們所描述的某種系統使用情形簡述中，會想像並簡短寫出某個特定參與者、的心理狀態 — 他希望擁有某些東西的原因爲何，或者在什麼情況下他會去做某些事。當你在寫所有使用案例時，不要一下子寫太多東西出來。有一件事很讓人驚訝，我們可以只用幾個字就表達出許多資訊。請針對特殊情況，從頭到尾寫出這個情況下世界中會發生的事。

In this narrative, invent a fictional but specific actor and briefly capture the mental state of that person-why he wants what he wants or what conditions drive him to act as he does. As with all use case writing, you needn't write much. It is astonishing how much information can be conveyed with just a few words. Capture how the world works, in this particular case, from the start of the situation to the end.

讓系統使用情形簡述維持簡潔是很重要的事。這樣一來，閱讀者可一眼看完整個敘事情節。另一方面，系統使用情形中的細節、動機或跟感情有關的內容也很重要。有了這些東西之後，每個閱讀者（從需求驗證人員到軟體設計師、測試撰寫人員與訓練教材撰寫人員等）都可知道系統如何在最佳狀態下爲使用者帶來價值。

我們只需花一點力氣、寫出系統使用情形的一點點描述，就可讓閱讀者以很簡單、優雅的方式了解使用案例。下面是一個例子。

Brevity is important so the reader can get the story at a glance. Details and motives, or emotional content, are important so that every reader, from the requirements validator to the software designer, test writer, and training materials writer, can see how the system should be optimized to add value to the user.

A usage narrative takes little energy to write and little space, and leads the reader into the use case itself easily and gently. Here is an example.

◆ 系統使用情形簡述：使用「快速提領現金」

瑪莉在去工作的路上，準備帶她的兩個女兒到托兒所。她把車開到 ATM 前面、把卡刷過讀卡機、輸入她的個人識別碼、選擇**快速提領現金**，然後輸入金額 35

元。ATM 送出一張 20 元、3 張 5 元紙鈔，並且印出一張收據，裡面記載她的帳戶扣掉 35 元後的餘額。ATM 完成**快速提領現金**的每筆交易動作之後，會重新設定螢幕，所以瑪莉拿錢後可以馬上離開，不用擔心下個人存取到她的帳戶。她很喜歡用**快速提領現金**，因為 ATM 不會問很多問題而降低整個互動速度。她之所以到這個特定的 ATM，因為這裡有 5 元紙鈔，而她可以用 5 元紙鈔付錢給托兒所，此外，她也不用下車就可領到錢。

人們寫出系統使用情形簡述以幫助他們了解使用中系統究竟為何。在寫使用案例的細節之前，我們可用這種方式暖身。有時候，開發團隊可能會在一開始時就針對所有使用案例寫出這些簡略的系統使用情形簡述。有時候，則會先針對特定的使用案例做這件事。當大家要寫出這個東西時，會把使用者、分析師與需求撰寫人員聚在一起、模擬系統使用情形，幫大家定出專案範圍、對系統使用情形產生共識。

系統使用情形簡述並不等於需求；相反地，它可作為更詳細、更一般化需求的基礎。我們用它決定使用案例的內容為何。使用案例本身是系統使用情形的正式格式 — 使用案例代表一種方案 — 它用一般化的參與者名稱，而系統使用情形簡述則是以參與者的實際名稱寫成的。

↑ 英 18

People write usage narratives to help them envision the system in use. They also use it to warm up before writing a use case, to work through the details. Occasionally, a team publishes the narratives at the beginning of all the use cases or just before the specific use cases they illustrate. One group wrote that they get a user, an analyst, and a requirements writer together and animate the narrative to help scope the system and create a shared vision of it in use.

The narrative is not the requirements; rather, it sets the stage for more detailed and generalized descriptions of the requirements. The narrative anchors the use case. The use case itself is a dried-out form of the narrative—a formula—with a generic actor name instead of the actual name used in the usage narrative.

1.7 習題

需求文件

- 1.1. 在需求文件大綱中，哪些小節會受到使用案例影響，哪些不會？請跟別人一起討論，並請思考一下你為何會想出不同的答案。
- 1.2. 請設計出另一種真正可用的需求大綱，並且把它用 HTML 超連結放在內部網路中。請注意你的子目錄結構與時間戳記慣例 (date stamping convention) (為何我們需要時間戳記慣例呢？) (譯註：時間戳記可替文件提供時間證明)。

系統使用情形簡述

- 1.3. 針對 ATM，請寫出兩段系統使用情形簡述。為何你所寫的使用情形簡述會跟我們的快速提領現金範例不同呢？對於設計系統的設計師來說，這些差異會有顯著不同嗎？
- 1.4. 假設某人走進一家新開的錄影帶出租店，想要租一捲小紅娘（The Parent Trap）的最早版本，請寫下出租情形。
- 1.5. 針對你目前的專案，寫下所設計系統的使用情形簡述。同時請其他人針對同樣情況寫出系統使用情形簡述。注意並討論兩份結果間的差異。為何會有差異存在、你在乎哪些差異 — 這些差異是在可容許範圍內，還是明顯的差異呢？

第一部分使用案例的主體部分

第2章把使用案例當作系統的行為合約

討論中系統是實現關係人間合約的一種機制。使用案例可說明合約的行為部分，我們之所以把某些句子放在使用案例中，是因為它所描述的動作會保障或促進某個關係人的利益。為了保障關係人的利益，句中可能會描述兩個參與者間的互動情形或描述系統內部要做的事（譯註：「描述系統內部要做的事」是在說明子系統間的互動情形）。

一開始，我們會先把使用案例單純用來捕捉對系統有目標的參與者間的互動情形。一旦有了這個東西之後，我們就可擴大討論範圍把使用案例視為系統對其有利益的關係人間的合約。前面的做法稱為**參與者與目標概念性模型**（actors and goals conceptual model），第二種做法則稱為**關係人與利益概念性模型**（stakeholders and interests conceptual model）。

2.1 對系統有目標的參與者間的互動情形

參與者對系統是有目標的

想像有一位職員負責接聽顧客用電話打來的服務請求（這位職員是圖 2.1 中的主要參與者）。當有一通電話打進來時，職員的目標為：用電腦記錄並啟動這個請求。

在這個例子中，系統負有一個責任：記錄並啟動這個請求。（系統有責任保障所有關係人的利益，而職員 — 主要參與者 — 只是其中一位關係人而已。）

為了負起這個責任，系統必須有系統地達到一些子目標。我們可讓系統內部自行達成子目標，也可能需要**支援性參與者**（supporting actor）的協助才能達成子目標（譯註：系統會幫助主要參與者達成其目標，卻可能需要支援性參與者達成系統內部所需子目標，這是區分主要參與者與支援性參與者的一種很好說法）。支援性參與者可能是負責列印的子系統，也有可能是其他組織，例如跟我們有合作關係的公司或政府行政機構。

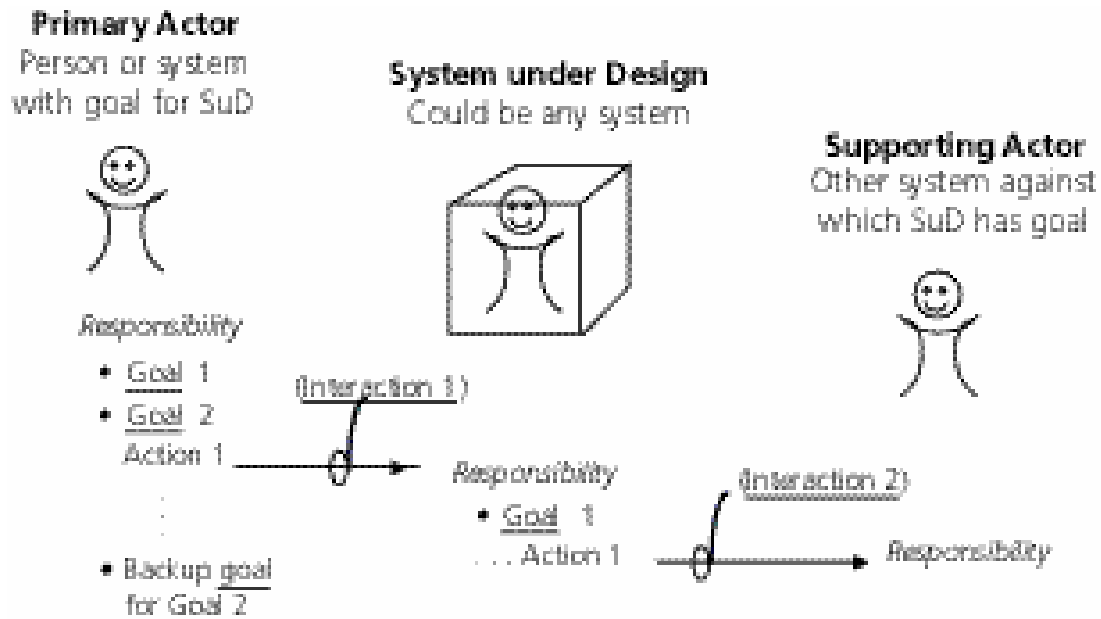


圖 2.1 對系統有目標的參與者會呼叫到由其他參與者

所負責的一些工作

支援性參與者通常會實現它的承諾，完成討論中系統（SuD）所交付的子目標。SuD 會跟外部參與者互動，依序達成某些子目標以實現它所被賦予的責任，換言之，SuD 的服務承諾。

實現某個服務承諾是系統的最高層目標，它會透過子目標達成這個承諾，而子目標還可以永無止境分解成更細的子子目標。如果我們把參與者的動作步驟也分解得很細，可能就能無限寫出子—子（...子）目標。就像諷刺作家 Irish 與詩人 Jonathan Swift 在詩文集中所寫的狂詩一樣（當然，他們不是針對使用案例而寫的）：

So, naturalists observe, a flea (在博物學家眼中的跳蚤)

Hath smaller fleas that on him prey (不管他捉到的是多小的跳蚤)

And these have smaller still to bite'em (還是會有跳蚤的跳蚤去咬它)

And so proceed ad infinitum (依據這個邏輯，始終會有更小的東西存在)

想寫出好的使用案例，其中的最大困難可能是如何管理跳蚤的跳蚤 — 使用案例中的子子目標。在第 5 章三個命名過的目標等級、第 7 章中的寫作指引 6：在動作步驟中放入「合理的」一組動作，以及第 20 章中的寫作提示 6：寫出合適的目標等級，我們會有更詳細的說明。

參與者與目標概念性模型是很好用的東西，它所描述的東西相當於企業或電腦系統（譯註：如果是企業使用案例，所描述東西的就是企業；如果是系統使用案例，所描述東西的就是電腦系統）。其中，參與者可以是個人、組織或電腦。模型中

可能會描述到好幾個系統，這些系統可能是人、組織或電腦。模型中也可能會描述到由其他電腦系統所驅動的軟體系統，而這個系統再呼叫由人類所扮演的支援性參與者，以提供所需服務；模型中也可描述某個組織呼叫電腦或個人的情形。這是一個非常有用、很一般性的模型。

↑ 英 24

目標可能會失敗

客服人員在處理客戶打來的申訴電話時，如果遇到電腦當機，她／他應該要怎麼辦呢？如果系統沒有提供它所承諾的服務，那麼使用系統的人就必須自行準備一個**備份目標**（**backup goal**）——在這個例子中，客服人員可能會用筆跟紙把申訴內容記錄下來。不論情況如何，客服人員都要完成主要的工作責任，對於系統失敗情況，也要準備**B 計畫**，以完成部份工作。

同樣地，系統的子目標之一也可能發生錯誤情況。或許主要參與者餵給系統的是不好的資料、或許系統有內部錯誤情況、或許支援性參與者沒有實現它所承諾的服務。這時候，系統應該要有什麼行為呢？這是 **SuD** 的行為需求中真正會讓人感到有趣的地方（譯註：這裡討論擴充情節中的失敗情況與其處理方式）。

在某些情況下，系統可修復失敗情況，回到系統行為的正常流程上。有時候，系統可能會放棄目標。如果你走到 **ATM** 前面，嘗試提出比你存款還要多的錢時，你的提錢目標只會失敗而已。如果 **ATM** 跟網路電腦間的連線失敗，系統也會以失敗結尾。如果你僅僅是打錯個人密碼，那麼系統就會給你第二次機會，以修正密碼。

爲了讓使用案例能夠成爲好的系統行為描述、很棒的功能需求，我們應該要把焦點放在目標失敗情況與系統對失敗的回應方式。對功能分解與資料流分解有經驗的人說過：目標失敗情況與系統對失敗的回應方式是使用案例對他們最有幫助的地方。

互動情形可以是合成的

我們可能遇到的最簡單互動情形是送出一個訊息。當我跟珍從大廳擦身而過時，我對她說：「嗨！珍。」這應該是我們兩人間的一個簡單互動情形。在程序導向程式設計中，呼叫像 `print(value)` 這樣的函式就是一個簡單的互動情形。在物件導向程式設計中，某個物件傳訊息給另一個物件（例如 `objectA->print(value)`）也是一個簡單的互動情形。

一連串的訊息或情節則代表一個合成的互動情形。假設我走到賣汽水的販賣機前面、放進一張一元美金紙幣買 80 美分的飲料，而販賣機卻告訴我無法找零。我跟機器間的互動情形如下：

1. 我放進一張一元美金紙幣。

2. 我按下「可樂」按鈕。
3. 販賣機說「無法找零」。
4. 我邊咒罵邊按下退錢按鈕。
5. 販賣機退回相當於 1 元美金的零錢。
6. 我拿著零錢離開（嘴巴一面嘟囔著）。

↑ 英 25

我們可以把*動作序列*（sequence）濃縮成單一動作步驟（例如上面的互動情形可簡述成「我試著從販賣機買一瓶飲料，不過它卻無法找零。」），然後再把這個濃縮過的動作步驟放到範圍更大的動作序列中：

1. 我走到公司銀行，提了一些錢出來。
2. 我試著從販賣機買一瓶可樂，不過它卻無法找零。
3. 我走回自助餐廳，在那裡買了一瓶可樂。

因此，互動情形可依我們的需要濃縮或分解下去，就像目標一樣。情節中的每個動作步驟都是為了某個目標而做的（譯註：每個動作步驟的目標代表使用案例所要捕捉目標的子目標），也因此它可展開，自己成為使用案例（譯註：子使用案例）。互動情形就像跳蚤中還會有跳蚤一樣，目標也是如此。

這件事聽起來有點糟，不過好消息是我們可以把系統行為用非常高階、濃縮起來的目標與互動情形表現出來。（譯註：作者反向思考說明分解互動情形雖然可能是一件壞事，濃縮互動情形則有助於我們用更抽象地方式表達事情）。只要一點、一點地分解動作步驟，我們就可根據所需精確度詳細說明系統行為。我通常把使用案例視為*尚未分解的敘事情節*。我們所要做的工作就是寫出閱讀者可用很舒適方式去讀的敘事情節。

精明的讀者可能已經看出來：我以很寬鬆的態度來使用*動作序列*（sequence）這個詞。在許多情況下，互動情形並不會以某個特定的動作先後順序發生。例如當我要買一罐 80 美分的飲料時，我可投入 8 個十美分，也可投入 3 個 25 美分、加上 1 個 5 美分（你可以有各種組合）。不論哪個硬幣先投都沒有關係。

嚴格來說，*動作序列*並不是正確的詞。從數學來看正確的詞應該是*部份有順序性的動作序列*（partial ordering）（譯註：也就是有部分的動作是要依照一定的先後順序來執行的，而其他則否）。然而，*動作序列*比較短、更接近我們要講的重點，也更容易被寫使用案例的人所了解。如果有人跟你說某些訊息是可平行發生的，你就可以說：「嗯，很好！那麼我們再多寫一些平行發生的狀況」，然後再看看我們可寫出怎樣的東西。我的經驗是：只要經過非常少的訓練，大家就可寫出非常清楚的說明描述。因此，我將繼續採用*動作序列*這個詞。相關範例請看使用案例 22 登錄損失，這是動作序列非常複雜的一個範例。

如果你想創造出一個寫使用案例的正式語言，那麼我們很容易就會陷入某種困境中。大部分的語言設計者可能會強迫寫作者列出所有可能發生的事件先後順序，或者發明複雜的表示法，以允許寫作者可寫出依任意順序發生的事件。因為我們所寫的使用案例是給其他人看而不是要給電腦看的，所以很幸運地我們只要簡單

寫出像「購買者可依任何順序放入總金額為 80 美分的五分鎊幣、十分鎊幣或二十五分鎊幣」這樣的描述就可以了。

用動作序列來描述過去發生的互動情形是很好的事，因為過去事件已完全確定。不過，如果是在描述未來可能發生的互動情形，我們就需要有一組*可能會發生的動作序列*。在這個世界中，未來可能發生的每個條件都會產生一個相對應的動作序列（譯註：作者用描述過去發生、已確定的互動情形來表示「沒有」條件子句的說明文字；另一方面，他用描述未來可能發生的互動情形代表「有」條件子句的說明文字）。例如，如果我想跟你說昨天是如何跟老闆要求加薪的，我可能會說：

「今天，我跟老闆進行一次很嚴肅的談話，我說：...她說...我說...等等。」

↑ 英 26

不過如果我想描述的是未來跟老闆間的交談過程，我可能會這樣說：

「待會要跟老闆談一些事，我真的非常緊張。」

「為什麼？」

「我將要跟老闆要求加薪。」

「如何？」

「嗯！首先我將這麼說...」，然後如果她這麼說「...」，我將回應說「...」不過，如果她這麼說「...」，我將試著這麼說「...」，像這樣子下去。

同樣地，如果我跟其他人說如何買汽水，我會這麼說：

「首先，請先準備好你的錢。」

「如果你有剛好的零錢，請把錢投入販賣機，按下可樂的按鈕。」

「如果沒有剛好的零錢，把錢投入販賣機，看看它是否能找零。如果它能找零的話...」

爲了描述未來發生的互動情形，我們必須處理不同條件下所產生的不同組動作序列。針對每一組動作序列（譯註：一組動作序列代表一個情節，而一個使用案例中又會包含好幾個情節），我們會說明發生條件爲何、相關的動作序列爲何，以及輸出結果爲何。

我們可以把一組動作序列濃縮成單一句子。例如「先往前走、然後從販賣機中買飲料」或「然後你跟老闆要求加薪」。有了這些動作序列之後，我們可根據自己的需要，把這些句子濃縮成簡短、高階的描述，也可以把它們分解成更詳細的描述。

到目前爲止，我們已經看到使用案例中會包含一組已達成（或未達成）目標的各種不同情節。不過，爲了讓使用案例更完整些，裡面還需要有：

- ◆ 跟相同主要參與者、相同目標有關的所有互動情形（譯註：我們一開始可能會先有主要成功情節，之後還需要靠腦力激盪法找出其他替代情節）。
All the interactions relate to the same goal of the same primary actor.
- ◆ 使用案例必須由觸發事件啓動，持續到目標完成或放棄爲止，而且系統必須完成跟互動情形相關的責任

譯註：我們剛開始寫使用案例時，可能只會寫出跟主要參與者的目標有關的情節，之後則需要針對幕後參與者的利益寫出相關情節。

把一些情節集合在一起，以形成使用案例

主要參與者會對系統有目標；系統應該幫主要參與者達成目標。某些情節會達成目標；某些情節結束後則會放棄目標。每個情節中都會有一連串的动作步驟，以展示分解後的動作與互動情形。使用案例中會把所有相關情節集合在一起，展現出目標成功或失敗的過程。

我們可用條紋褲的形狀來象徵使用案例中的情節（請參見圖 2.2）。褲子的皮帶上會寫出把所有情節聚集在一起的目標名稱。至於那兩條褲管，其中一條褲管上所寫的都是以成功收尾的情節，而另一條褲管上所寫的都是以失敗結尾的情節。不論是成功褲管或失敗褲管上的條紋，每一條都代表一個情節。我們會把成功褲管上的第一個條紋稱為**主要成功情節**（main success scenario）。成功褲管上的其他條紋則是最終以成功結尾的情節 — 某些情節會走其他的成功替代路徑，某些則是會從情節中的某個失敗情況復原。至於在失敗褲管上的所有條紋，都代表以失敗結尾的情形，其中有些可能是先從失敗情況中復原，最後還是失敗的情節。

↑ 英 27

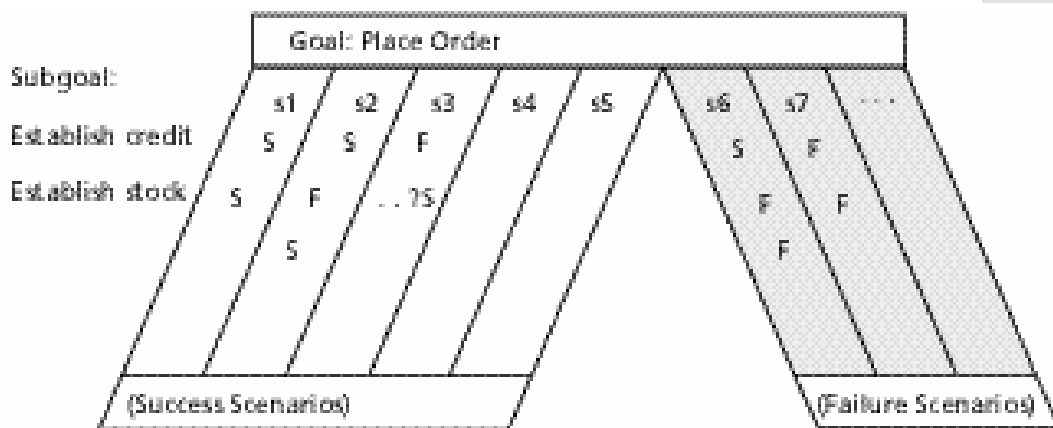


圖 2.2 條紋褲：成功或失敗的情節

事實上，我們不會從頭到尾個別寫出每個情節。這是一種很差的寫作策略，因為這樣做太繁瑣、重複，也很難維護它。不過，條紋褲這個形狀有助於我們心裡記得幾件事：每個使用案例都會有兩種結果（譯註：條紋褲有兩條褲管）、主要參與者的目標會把所有相關情節綁在一起（譯註：條紋褲的皮帶上會寫出目標名稱）、每個情節都是目標成功或失敗的簡單說明（譯註：條紋褲上的每個條紋分別代表一個情節）。

在圖 2.3 中，我們把條紋褲形狀用在子使用案例上，在外圍的使用案例中放進子條紋褲的形狀。例如，顧客希望下訂單，而下訂單的子目標之一是建立信用資料。

這個子目標很複雜，有可能會成功、也可能會失敗：它是一個使用案例，不過我們把它濃縮成單一動作步驟。動作步驟**顧客建立信用資料**代表另一條褲子的皮帶（譯註：子使用案例的目標）。至於包含這個動作步驟的條紋或情節，在這個動作步驟上的子目標有可能會成功、也可能不會。在圖 2.3 的情節 1 與 2 中，**顧客建立信用資料**這個子目標都會成立，而在情節 3 與 7 中則不會成立。然而，在情節 3 中建立信用資料雖然沒有成功，不過下一個子目標建立所需庫存資料則會成功，而且情節最後是以成功結尾的。在情節 7 中，第一個子目標、第二個子目標都會失敗，而且整個使用案例**下訂單**是以失敗結尾的。

我們在圖 2.3 的條紋褲中再秀出另一個子使用案例條紋褲的重點是：外圍的使用案例並不會去關心子使用案例裡面到底執行了什麼東西才會變成最後狀態。子使用案例可能成功、也可能會失敗。而外圍的使用案例或呼叫子使用案例的使用案例只是很單純考量呼叫子使用案例的那個動作步驟是否成功。

↑ 英 28

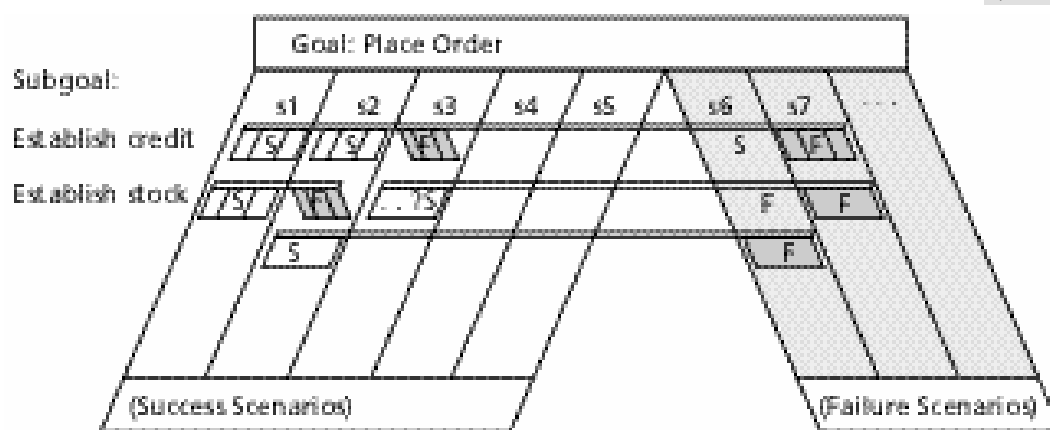


圖 2.3 在大條紋褲中再秀出子目標的小條紋褲

條紋褲形狀可讓我們領略出下列幾點寫作原則：

- ◆ 某些系統會以成功結尾；某些則以失敗結尾（譯註：在一條褲管中都是以成功結尾的情節；另一條褲管中則是以失敗結尾的情節）。
- ◆ 使用案例中會結合所有相關的成功或失敗情節（譯註：條紋褲會用皮帶結合這些情節）。
- ◆ 每個情節會純粹描述在一組特定環境下的某個輸出結果（譯註：褲管上的每個條紋代表一個情節）。
- ◆ 使用案例中會包含一些情節（譯註：褲子上有許多條紋），情節中的動作步驟可能是另一個子使用案例（譯註：大條紋褲中的小條紋褲）。
- ◆ 情節中的動作步驟並不關心條紋中所使用子使用案例的內容，它只關心結果是成功或失敗（譯註：條紋褲的褲腳分別代表以成功或失敗結尾）。

我們將在所有的使用案例寫作中使用這些寫作原則。

2.2 有利益的關係人間的合約

參與者與目標模型解釋如何寫出使用案例中的句子，不過它卻無法滿足「描述出討論中系統之內部行為」的需要（譯註：SuD 所需的內部行為會影響到關係人的利益）。因此，我們需要擴充參與者與目標模型，把使用案例視為有利益的關係人間的合約，並且把擴充後的模型稱為*關係人與利益概念性模型*（stakeholders and interests conceptual model）。模型中的關係人與利益部分可告訴我們什麼東西應該要放在使用案例裡面、什麼東西不要放在使用案例中。

譯註：合約代表兩造間的權利義務，也就是這裡的關係人與利益模型；合約的執行方式則屬於合約的行為部份，也就是參與者與目標模型。換言之，關係人與利益模型會包含參與者與目標模型。另一方面，目標是某些動作要達成的；利益則是進行這些動作時應該避免違背它、系統要保障的。

SuD 必須履行關係人間的合約，而使用案例中會詳述合約的行為部分。不過，當系統在執行時，並不是所有關係人都會在場。主要參與者通常會在，不過還是有可能會不在場。我們把其他不在場的關係人稱為*幕後參與者*（offstage actor）。系統必須滿足這些幕後參與者的利益（請參見圖 2.4），例如蒐集資訊、執行資料確認檢查與更新系統紀錄等。

↑ 英 29

舉例來說，ATM 必須保有所有互動情形的紀錄，如果發生爭執的話，我們才可以保障關係人的利益。因此，除了必要的交易資料之外，我們必須記錄其他資訊，這樣一來，我們才可找出交易動作失敗之前，最多曾經做到什麼地步（譯註：交易動作通常是由一連串步驟所組成。當交易動作失敗時，我們會取消整個交易動作回復到進行交易前的狀態，所以如果沒有回復用的相關系統紀錄存在，我們將很難了解交易動作在失敗前，曾經做到什麼步驟）。發出現金前，ATM 與銀行系統會證實帳戶持有者的帳戶中是否有超出提錢者所要提的錢，以免出現提出金額超過顧客在銀行中實際存款金額的情況。

代表合約中行為部分的使用案例會捕捉*所有*跟關係人利益有關的行為需求，而且*只*包含行為需求（譯註：我們也可藉由關係人利益，檢驗是否有遺漏的行為需求）。

為了小心完成使用案例，我們會列出所有關係人，也會明確指出使用案例執行時跟關係人相關的利益，利益中會陳述使用案例成功時對每個關係人的意義為何（譯註：成功事後保證）、失敗時系統該如何保障關係人的利益（譯註：最小事後保證）。有了關係人與利益之後，當我們在寫使用案例中的動作步驟時，就可確保使用案例從觸發到結束之前，各式各樣的利益都有被滿足（譯註：與其說滿足，不如說保障這些利益）。這也是我們之所以知道什麼時候要再多寫些使用案例、什麼時候可以停手、什麼東西要放進使用案例中或什麼東西不放進去等事情的原因。

大部分的人都不會這麼小心寫出使用案例，很幸運地，他們通常都會僥倖成功。好的使用案例寫作人員在寫非正式格式的使用案例時，會在他們的腦海中思考關係人與其利益。他們可能沒有寫出某些東西，等到系統開發時再用其他方式寫出這些漏掉的東西。在大部分專案中，這種做法還不錯。然而，有時候某些被漏掉的利益可能會引發巨大成本。請參見圖 4.1，裡面提到一個故事，說明遺漏某些關係人利益時所發生的事。

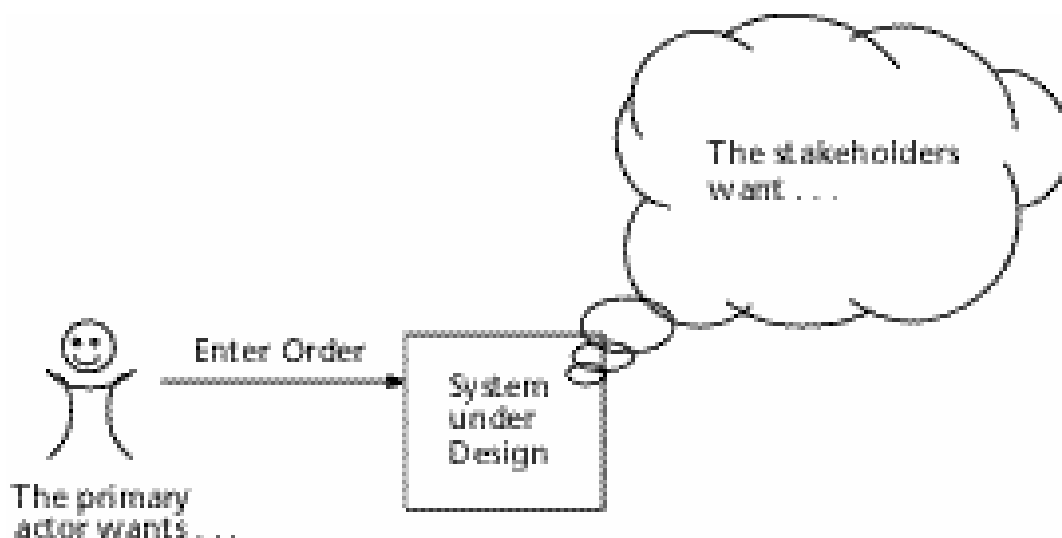


圖 2.4 SuD 會服務主要參與者並保障幕後關係人

↑ 英 30

爲了滿足關係人的利益，我們需要在關係人與利益模型中描述三種動作：

- ◆ （爲了促進目標的達成）我們會描述兩個參與者間所發生的互動情形。

譯註：在 UML 1.3 版中，SuD 也被視爲參與者，所以這裡所謂的「參與者間的互動情形」應該包括主要參與者、支援性參與者與 SuD 間的互動情形。

- ◆ 證實性動作（以保障關係人）。
- ◆ （爲了關係人利益而做的）系統內部狀態變動。

譯註：證實性動作與內部狀態變動不一定跟參與者有關，卻跟關係人的利益有關。請看圖 2.6，在圖中，私有動作跟關係人的利益間有關聯存在，參與者與互動情形也有關聯存在。

關係人與利益模型只會改變整個使用案例寫作過程中的一小部分（譯註：步驟 8、12）：列出關係人與其利益、用這份清單審查使用案例的主體是否漏掉某些東西。雖然這只是小小的變化，不過對使用案例的品質來說，卻有很大的影響。

2.3 （用 UML 畫出來、跟使用案例有關的）圖

形模型

本節主要是針對想了解抽象模型的人而寫的（譯註：本節說明跟使用案例相關的超模型【meta model】）。如果你不是這樣的人請安心跳過本節。

就像我們之前所提過的一樣，使用案例中會描述有利益的關係人間的行為合約。我們組織行為的方式是根據一組挑選過關係人（這些關係人會要求系統為他們做某些事，我們把這些關係人稱為**主要參與者**）的操作型目標（譯註：「操作型」是非常重要的形容詞，代表這些目標是系統可履行、參與者可衡量的）。使用案例的名稱就是主要參與者的目標。使用案例中會包含描述合約需要的所有行為。

譯註：這裡忍不住再重述一遍，合約中的行為必須滿足參與者的目標並保障關係人的利益。這也是本書作者從參與者與目標概念性模型擴充成關係人與利益概念性模型的最大改變。

系統必須經由某些動作、盡到滿足同一群關係人一致利益的責任。它會有下列三種動作之一：

- ◆ 參與者間的互動情形，資訊會在參與者間往返。
- ◆ 保障某個關係人利益的證實性動作。
- ◆ 內部狀態變化，其目的也是為了保障或促進某個關係人的利益。

情節由動作步驟所構成。在**成功情節**中，針對系統必須負責的服務，關係人的所有一致利益都要被滿足。在**失敗情節**中，系統要保證所有利益都有受到應有的保障。當關係人的所有利益被滿足或受保障之後，情節才會結束。

三種要求系統達成目標的觸發事件分別是：(1).由主要參與者所啟動、跟系統之間所發生的**互動情形**(2).主要參與者經由別人所啟動的**互動情形**，或者(3).由時間或狀態變化所啟動的**互動情形**。

圖 2.5 到圖 2.8 是用統一模型語言（UML）所畫出來、代表本章所描述使用案例的模型。

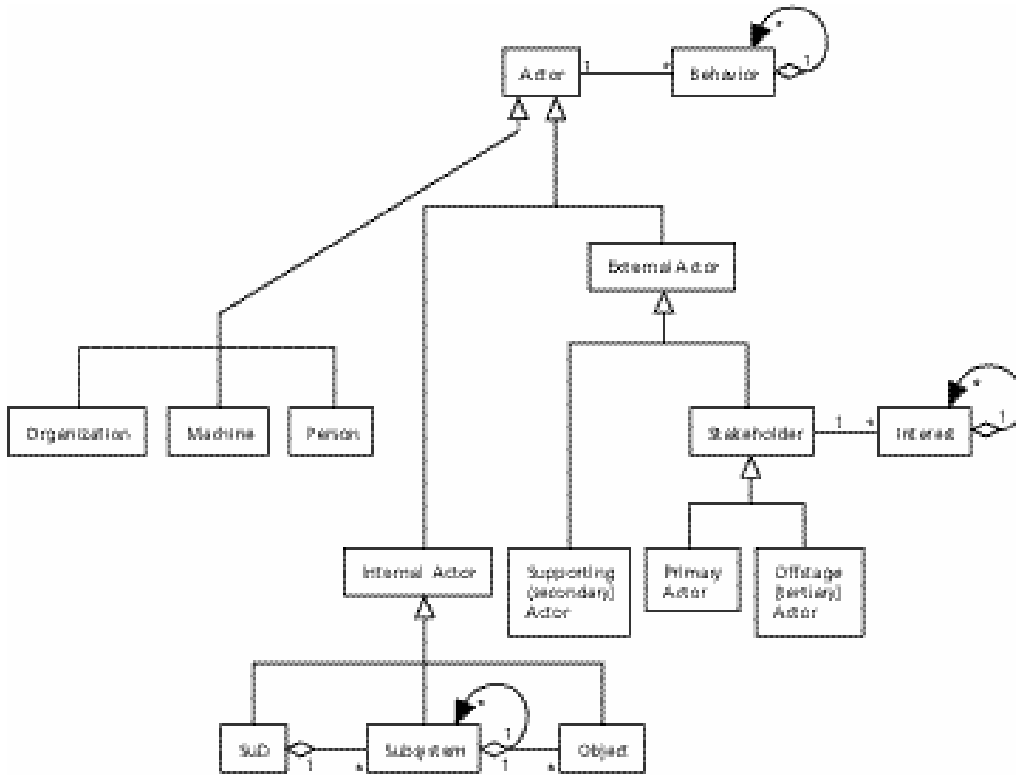


圖 2.5 參與者與關係人。

系統會保障關係人利益。參與者則會對系統發生行為。主要參與者同時也是關係人。

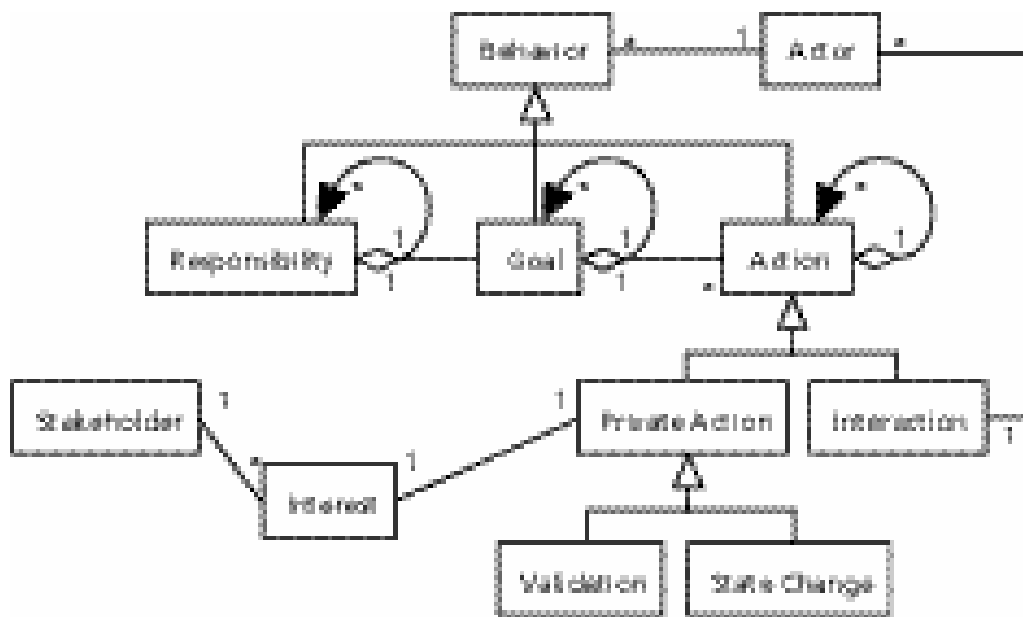


圖 2.6 行為。

目標導向的行為是由責任、目標與動作所構成的。其中，私有動作是為了促進或保障關係人利益而發生的。互動情形則代表某個參與者連到另一個參與者的一些動作。

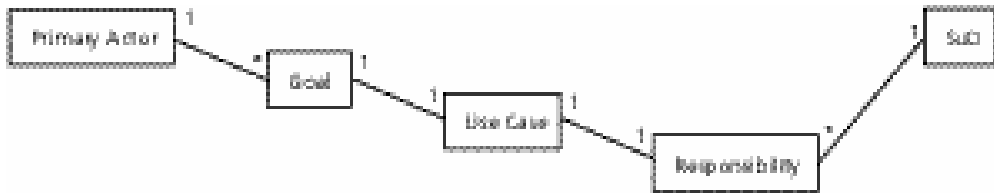


圖 2.7 把使用案例視為要求系統負的責任。

使用案例中會捕捉主要參與者的目標、要求 SuD 負的責任。

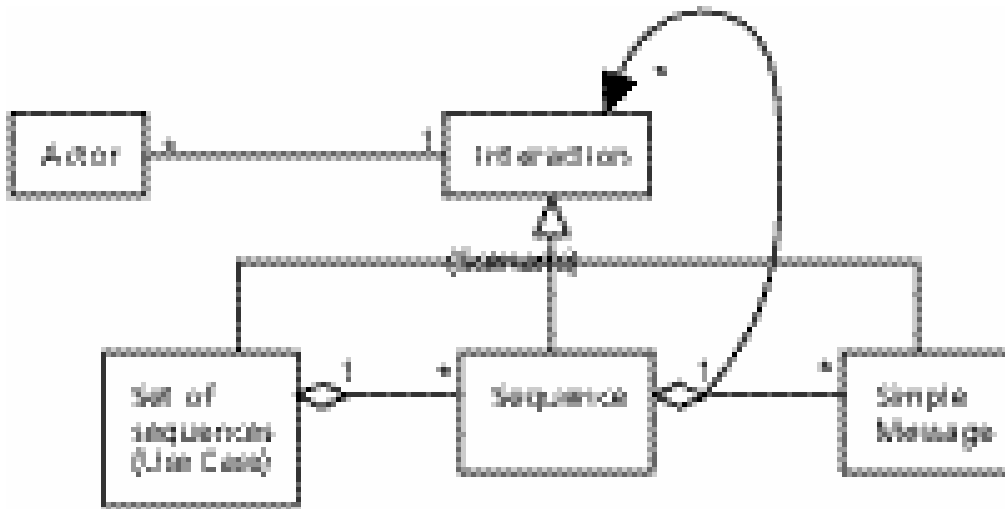


圖 2.8 互動情形可以是合成的。

互動情形中會有 N 個參與者參與互動。互動情形可被分解成（子）使用案例、情節或簡單訊息。同樣地，為了方便起見，我們這裡用「動作序列」而不說「部份有先後順序的動作序列」。

這裡跟各位說明一些事實。我需要有一些專案來測試本章所提到的模型，測試的專案要用以模型為基礎的工具。如果沒有這些經驗的話，我將無法修正模型中的錯誤。換言之，這個模型中可能會有一些細微錯誤。我希望能從願意測試這個模型的人當中找到想法以修正模型，或許我們可創造出一個以模型為基礎的工具來實踐這個模型。

第3章設計範圍

關於設計範圍這個詞，它代表我們要設計東西的邊界，以便跟其他人的設計工作內容或現存的設計結果區隔開來。

持續追蹤專案或討論中的範圍是件很難的事。顧問 Rob Thomsett 先生曾經跟我介紹過一種很棒的工具 — 專案範圍內／外清單 (in/out list)，我們可用它來追蹤或管理範圍方面的問題。這份清單很簡單且有效，你可能會覺得它非常不可思議。我們可用它來控制日常會議或專案需求的範圍討論部分。

這樣的表格只需要用到很簡單的三欄就好。左邊一欄中放主題，其餘兩欄則說明主題是在專案範圍之「內」或「外」。當我們感到混淆、不知道某個主題是否在專案範圍之內時，就可把這個主題加到清單中，然後問大家這個主題是在專案範圍內或外。就像 Rob 曾說過、本人所看到的一樣，採用這種做法的結果非常讓人驚訝。在同一個房間中，每個人雖然可能都很清楚某個主題是否在「自己」的專案範圍之內，不過大家的觀點卻可能不同，有人可能認為它是在專案範圍內、有人則否。Rob 曾說過，我們有時候會需要由專案指導員來進行仲裁，以決定某個特殊主題是否在工作範圍之內。把某個主題決定放在工作範圍之內或外，很可能會影響到好幾個工作人月的工作進度。請試著把這項技術用在下個專案或下次會議中。

表 3.1 是專案範圍內／外清單的一個範例，我們把它用在採購申請追蹤系統上。一開始進行需求或使用案例寫作的開發活動時，請馬上用這份專案範圍內／外清單區分什麼東西在工作範圍之內、什麼不是。一旦討論內容快要失控或某些不相關需求不知不覺進入討論範圍時，我們就可以趕快參考這份清單。隨著討論進行下去，我們會不斷更新這份清單。

請把這份專案範圍內／外清單用在討論中系統裡面跟功能範圍與設計範圍相關的主題上。

↑ 英 35

表 3.1 專案範圍內／外清單範例

主題	內	外
開出任何格式的發票 (根據供應商、零件、個人等) 產生申請的相關報告	內	外
把幾份申請合併到一份 PO 中	內	
有部分東西送達、部分東西延遲送達或送錯東西	內	
所有的新系統服務、軟體	內	
系統中任何非軟體部分		外

找出現有的軟體中，哪些是可用的
正式申請

內
內

3.1 功能範圍

功能範圍代表系統將提供的服務，我們最後都會把這些服務寫在使用案例裡面。然而，當專案一開始時，我們可能不是很明確知道哪些服務將是系統會提供的。因此，我們可以一面找尋使用案例、一面決定功能範圍為何——這兩件工作是交互進行的。專案範圍內／外清單對找尋功能範圍是很有幫助的，因為它可幫我們畫出系統的功能邊界，知道什麼服務在範圍裡面、什麼在範圍之外。除了專案範圍內／外清單之外，我們還可用其他兩種工具找出功能範圍：參與者 — 目標清單（actor-goal list）與使用案例簡介（use case brief）。

譯註：當我們還無法明確指出系統邊界時，專案範圍內／外清單是比參與者 — 目標清單好用的工具，因為它可以讓我們把不確定會有的主題先標示成「外」，反悔時再改成「內」，這樣就不會有所遺漏。

參與者 — 目標清單

參與者 — 目標清單中會列出系統所支援的全部使用者目標，秀出系統的功能內容。參與者 — 目標清單跟專案範圍內／外清單不同的地方是：後者同時會有範圍內、外的項目，而前者則只會有系統實際支援的服務。表 3.2 是採購申請追蹤系統中專案的參與者 — 目標清單。

我們用三欄式表格記錄這份清單。在表格的左欄中放入主要參與者的名稱（主要參與者對系統會有目標）；然後把跟系統相關、每個參與者的目標放在中間欄；最後再把優先權或系統在哪個發行版本中可能會支援這個目標的猜測放在第三欄。請在專案進行中，持續更新這份清單，讓它時時反映出系統的功能邊界狀態。有些人會加入一些額外的欄——*觸發事件*（找出哪些使用案例是由時間所觸發，而不是由人或其他系統所觸發）、*企業優先順序*、*開發複雜性與開發優先順序*，以便讓我們區分(1)企業需要跟(2)影響開發優先順序的開發成本兩者間的不同。參與者 — 目標清單是使用者代表、財務贊助者與開發團隊間最初的協商點。這時候，我們會把焦點放在專案的功能配置與內容上。

↑ 英 36

譯註：如果我們採用 RUP 或 XP 開發流程，在考慮參與者 — 目標清單中的優先順序時，也應該要考量階段計畫（RUP）或發行計畫（XP）中的反覆該如何配置。

表 3.2 參與者 — 目標清單範例

參與者	屬於工作等級的目標	優先權
任何參與者	檢查申請	1
授權人	改變授權	2
採購人	改變供應商合約	3
申請人	啟動申請	1
	改變申請	1
	取消申請	4
	標示所申請東西已送達	4
	拒收已送達貨品	4
批准人	完成要發出的申請	2
採購人	完成要下訂單的申請	1
	啟動跟供貨商之間的 PO	1
	發出未送達的警訊	4
授權人	證實批准人的簽名	3
收貨人	登錄東西送達	1

使用案例簡介

這裡重複強調「管理自己的精力、盡可能在低精確度中工作」的重要性。參與者 — 目標清單就是以最低的精確度來描述系統行爲，而且工作時綜觀系統全局是非常有幫助的。比參與者 — 目標清單還要詳細一點的下一層精確度可能是主要成功情節或*使用案例簡介* (use case brief)。

使用案例簡介是用 2 到 6 個句子來描述使用案例中的行爲，裡面只會提到最重要的活動與失敗情形。它可幫人們記得使用案例中會發生的事，也非常有助於評估工作複雜性。有些以商業現成元件來建構系統的開發團隊就會用這樣的簡介來選擇元件。某些專案開發團隊（例如一些擁有非常好的內部溝通、會跟使用者持續討論的團隊）在寫需求時，只會寫出這些使用案例簡介；其餘需求則會在後續討論、系統原型與經常交付給顧客的增量版中再補足它。

↑ 英 37

譯註：在終極開發流程 (extreme programming, XP) 中，我們會先請「顧客」寫下使用者故事 (user story)，然後請顧客駐點 (customer on site)。當開發人員覺得需求不清楚時，可以馬上詢問顧客，並且會請顧客持續做驗收測試 (acceptance test)，以盡快獲得回饋。

我們可以用表格來整理這些使用案例簡介，把它視爲參與者 — 目標清單的擴充結果，或者把使用案例簡介直接寫在使用案例的主要內容中，作爲使用案例的第

一版草稿。表 3.3 就是使用案例簡介的一個範例，感謝 Navigation Technologies 公司的 Paul Ford、Steve Young 與 Paul Bouzide 提供這個範例。

表 3.3 使用案例簡介範例

參與者	目標	使用案例簡介
生產人員	修改管理層面的資料格	生產人員把管理層面的超資料（管理階層、幣別、語言碼、街道種類等）加到參考用資料庫中。我們會分門別類把原始資料的聯絡資訊記錄下來。這是更新參考資料的一個特例。
生產人員	準備數位製圖用的原始資料	生產人員把外部數位資料轉成標準格式，然後證實並更正它，準備把它跟操作用資料庫結合在一起。這些資料都會分門別類儲存在數位原始資料庫中。
生產人員或外勤人員	對於共享、被登出的資料，完成更新操作用資料庫的交易動作	工作人員把累積的更新交易動作應用在操作用資料庫上。不會有衝突的交易動作會被完成，寫到操作用資料庫中。應用程式的情境會跟操作用資料庫保持同步。在應用程式的情境中，已完成的交易動作會被清除、跟操作用資料庫保持一致，而會發生衝突的交易動作則留待手動或互動式方式解決它。

3.2 設計範圍

設計範圍代表系統邊界 — 如果軟體會佔據空間的話，我會把它視為「空間上的

邊界」。它代表我們所設計或討論的整組系統，包括硬體與軟體。如果我們負責設計 ATM，那麼就應該要生產出 ATM 箱子裡面的硬體與軟體 — 這時候，箱子跟箱子裡面的所有東西都是我們需要設計的。而這個箱子所用到的電腦網路則不是由我們負責設計的 — 它在我們的設計範圍之外。

↑ 英 38

在原文書中，作者用「範圍」兩字代表「設計範圍」，中文版中則在整本書中完整寫出「設計範圍」。作者之所以做這樣的簡寫，主要是因為參與者 — 目標清單跟使用案例已經夠定出功能範圍了，至於設計範圍，則是每個使用案例中都應該考量的一個主題（譯註：這也是作者的參與者與目標概念性模型中要求每個使用案例都要標示出設計範圍的原因）。

就像下面小故事中所展示出來的：寫跟看使用案例的人對使用案例的設計範圍有一致、正確認知是非常重要的。認知錯誤的代價包括：以兩個或更多認知不同的數量為因子的成本，加上對合約造成毀滅結果的可能性。使用案例的閱讀者必須要能夠很快看出哪些東西是在你想要的系統邊界之內。單單從使用案例名稱或主要參與者，我們很難明顯看出設計範圍。不同大小規模的系統甚至可能會有一組相同的使用案例。

一般來說，寫使用案例的人會認為系統邊界這麼明顯，根本不需要提它。然而，一旦寫或讀的人有好幾個，使用案例的設計範圍就不是那麼明顯了。其中一位寫作者可能會把整個企業當成設計範圍（請參見圖 3.1）、另一位則可能會把公司的所有軟體當成設計範圍、其他人可能會把它視為新的客戶端加上伺服器系統，也有人可能會把它視為只包括客戶端或只包括伺服器。閱讀者沒有線索可知道使用案例的設計範圍究竟為何，他可能會迷失在文件中或誤解文件的意思。

◆ 一個很短卻真實的故事

為了協助一個想花固定開發時間與成本建構出來的大型系統，我們看了一些示範設計結果。針對印表機部分，我跟大家討論它的功能。結果，IS 專家笑著說：「我被你們這些滿腦子都是個人電腦的人給打敗了，你以為我們會用這樣的小印表機來列印發票嗎？事實上，我們有一個大型的列印系統，裡面會有列表機串鏈、批次處理 I/O，以及所有跟列印有關的東西。我們需要印出一箱箱的發票！」

我被他的話嚇到。「你的意思是說印表機不在系統的設計範圍之內嗎？」

「當然不是！我們將使用現存的列印系統。」

最後，我們發現要設計的系統會跟列印系統間有複雜的介面存在。系統必須替要列印的東西準備磁帶。從晚上到早上間，列印系統會讀入磁帶，並印出它所能列印的東西。它也會準備一個回覆用磁帶，裡面記錄列印工作結果，說明哪些東西沒辦法列印。隔天，我們的系統會讀回這些結果，標示出哪些東西沒有正確列印出來。設計出跟磁帶間的連接方式是很重要的工作，而且這樣的設計方式跟我們之前所預期的東西完全不同。

列印系統不是我們要設計的東西，不過它卻是我們會用到的。它在我們的設計範圍之外。（如同 3.3 節中所述，它是一個支援性參與者【supporting actor】。）剛

開始，我們沒有發現這個錯誤，因此把列印功能寫在使用案例中、放入設計範圍之內，也花了一些功夫寫出系統不需要的東西。

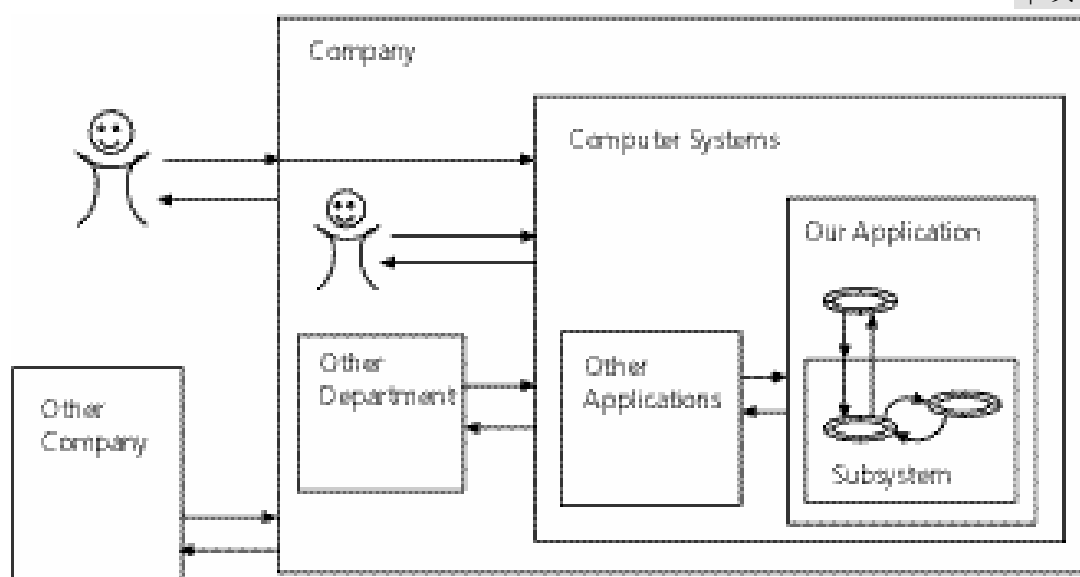


圖 3.1 設計範圍是可大可小的

要做什麼才能減少誤解呢？

針對這個問題，我能找到的唯一答案是替每個使用案例標示設計範圍，以特定的名稱來代表有意義的設計範圍。為了方便說明起見，我假設 MyTelCo 公司正在設計某個 NewApp 系統，裡面會包含一個 Searcher 子系統。這時候，設計範圍名稱分別為：

- ◆ 企業（換言之就是 MyTelCo）。針對這樣的設計範圍，我們會討論可達成主要參與者目標的整個組織或企業的行爲。請在使用案例的設計範圍欄位標示出組織名稱 — MyTelCo — 而不要只寫出「公司」兩個字。如果我們是討論某個部門的行爲，請寫出部門名稱。企業使用案例（business use case）就是設計範圍為企業的使用案例。
- ◆ 系統（換言之，NewApp）。它代表你負責建構的硬體或軟體部分。系統外的東西就是跟我們所設計系統有接觸的所有硬體、軟體或人類。
- ◆ 子系統（換言之，Searcher）。這時候，我們必須把主要系統打開，此時的設計範圍是在討論系統中某部分的運作情形。






用圖形化的圖示來強調設計範圍

請考慮在使用案例標題的左邊加上一個圖示，讓閱讀者在還沒有看使用案例中的內容之前，先知道使用案例的設計範圍為何。到目前為止，還沒有工具可管理這些圖示，不過我發現畫出這些圖示可減少我們對使用案例發生混淆的情形。在本

書中，我會替每個使用案例標示出適當的圖示，讓你很容易就可以注意到它的設計範圍為何。

當你在看下面的清單時，請記得黑箱式使用案例中不會描述到討論中系統的內部結構，而白箱式使用案例則會。

↑ 英 40

- ◆ 企業使用案例的設計範圍是企業。它的圖示是一棟建築物。如果你把整個企業視為黑箱，那麼這個圖示就是灰色的（）。如果你會提到企業內的部門與人員，那麼這個圖示就是白色的（）。
- ◆ 系統使用案例的設計範圍是電腦系統。它的圖示是一個箱子。如果你把系統視為黑箱，那麼這個圖示就是灰色的（）。如果你會提到系統內元件的運作情形，那麼這個圖示就是白色的（）。
- ◆ 元件使用案例的設計範圍是討論中系統的子系統或元件。它的圖示是螺絲（）。範例請參見使用案例 13 到使用案例 17。

各種不同設計範圍的範例

下面提供三個範例以展示出系統的不同設計範圍。

(1) 從企業設計範圍到系統設計範圍的範例


假設我們在一家 *MyTelCo* 電話公司工作，它要設計一個新系統 *Acura* 以記錄顧客要新增服務或更新服務組合的訂單。*Acura* 中包括伺服器以及連到伺服器的工作站，而且伺服器還會連到主機以執行舊系統 *BSSO*。對我們來說，*BSSO* 只是連到主機的一台終端機而已。我們無法改變它，只能使用它現存的介面。

Acura 的主要參與者包括顧客、雇員、經理與 *BSSO*（我們很清楚知道設計範圍中並不包括 *BSSO*）。

我們先找出系統該支援的目標。最明顯的目標是「新增服務申請」。我們決定它的主要參與者是公司雇員，雇員會負責服務顧客。坐下來寫一些使用案例吧！

我們馬上就面臨到的一個問題是「討論中系統究竟為何？」很明顯地，有兩個東西是我們很感興趣的：

- ◆ *MyTelCo*。針對整個公司，我們對一個問題很感興趣：「對顧客來說，*MyTelCo* 所提供的服務看起來為何？我們希望完整秀出新增服務申請的整個過程，其中包括從客戶一開始申請服務到完成申請、提供顧客服務的完整流程。」這樣的問題讓人很感興趣。一方面，公司經理希望知道外面世界是如何看待新系統的，而負責完成顧客服務申請的開發團隊也希望知道新系統的使用情境為何。

這個使用案例是針對企業設計範圍（）而寫的，它的設計範圍欄位會標示出 *MyTelCo*，而且在寫使用案例時也不會提到公司內部的工作人員（換

句話說，我們不會提到雇員、部門或電腦)。這類型的**使用案例**通常稱為**企業使用案例**，因為它會說明企業跟外界間的**互動情形**。

↑ 英 41

- ◆ **Acura**。同時，我們也對另一個問題感到興趣：「**Acura** 的服務看起來如何？一方面，它會跟雇員或顧客有關係，另一方面則跟 **BSSO** 系統有關係。」這個使用案例是設計師心裡最關心的東西，因為裡面會描述出他們所要建構的東西究竟為何。它是針對系統設計範圍 (▣) 而寫的，我們會在它的設計範圍欄位標示「**Acura**」。使用案例中可任意提到雇員、部門與其他電腦系統，不過我們不會提到工作站與伺服器這兩個子系統 (譯註：系統是由這兩個系統所構成，我們在描述黑箱式系統使用案例時，並不會寫到系統中的子系統或元件)。

針對上面兩個問題，我們會分別寫出兩個使用案例。為了避免重複描述到相同資訊，我們會先寫出目標等級比較高的企業使用案例 (它的目標等級圖示為風箏)，描述 **MyTelCo** 如何回應顧客申請、提供顧客服務，或許還會包括向顧客計價與收費的情形。企業使用案例的目的是秀出新系統運作的情境為何。而在系統使用案例中，我們會描述出要花 5 到 20 分鐘、處理顧客申請服務的使用者目標等級使用案例，它會以 **Acura** 整個系統為設計範圍。

譯註：一般來說，設計範圍指的是系統邊界。決定設計範圍之後，我們才能替使用案例找到合適的主要參與者與目標。至於目標等級為何，主要是看系統設計範圍的主要參與者，先找出他們的使用者目標，再以這個目標等級為基準 (海平面) 決定出比較高或低的目標等級。

使用案例 6 ◀ (企業的) 新增服務申請 ↗

主要參與者：顧客

設計範圍：MyTelCo

目標等級：目標摘要等級

主要成功情節：

1. 顧客打電話到 MyTelCo，申請新增一項公司所提供的服務...
2. MyTelCo 提供顧客新申請的服務...等...。

使用案例 7 ◀ (Acura 的) 新增服務申請 ↗

主要參與者：服務外部顧客的雇員

設計範圍：Acura

目標等級：使用者目標等級

主要成功情節：

1. 顧客打電話進來，雇員跟顧客討論他的服務申請。

2. 雇員在 Acura 系統中找尋顧客資料。
3. Acura 秀出顧客現有的服務組合...等...

這裡沒有寫出以 Acura 工作站或 Acura 伺服器為設計範圍的使用案例，因為我們對這樣的使用案例沒有興趣。稍後，在設計開發團隊中可能會有人用使用案例來記錄 Acura 的子系統。這時候，他們可能會寫出另外兩個使用案例，一個是以 Acura 工作站為設計範圍，另一個則是以 Acura 伺服器為設計範圍。根據個人經驗，我們應該不需要寫出這些使用案例，因為有其他合適的技術可以把子系統架構記錄下來。

↑ 英 42

(2)從包含多個電腦的設計範圍到單一應

用程式的設計範圍

接下來是比較罕見情況，不過卻是非常難的情況。這裡先說明一下 MyTelCo 的情境。

Acura 會慢慢取代掉 BSSO。新的服務申請會放到 Acura 中，然後再由 BSSO 去修改它。隨著時間過去，Acura 會接手越來越多的功能。這兩個系統必須並存且同步。因此，使用案例的設計範圍要包含這兩個系統：其中的 Acura 整個是新的，而 BSSO 會被修改以達成同步的目的。

在上面所描述的情境中，困難點在於我們可能會有四個內容重複的使用案例，兩個是 Acura 的、另外兩個是 BSSO 的。在兩兩成對的使用案例中，其中一個使用案例的主要參與者是雇員，另一個使用案例的主要參與者其他電腦系統。我們沒有辦法避免寫出這四個使用案例，不過看到這些使用案例的人卻會感到非常迷惑，因為這些使用案例看起來其內容都是重複的。

爲了處理這種情況，我會先寫出一個目標摘要等級的使用案例，讓它的设计範圍同時涵蓋兩個電腦系統。這樣一來，我就可以記錄它們在不同時間點的互動情形。然後在這個使用案例中參考到四個特定的使用案例，這四個特定的使用案例中又會寫出每個系統的需求。因此，第一個被寫出來的使用案例會是白箱式的(請注意，我們用白色箱子符號)。

因爲這裡的情況很複雜，所以我還在使用案例中加入附圖，秀出每個使用案例的實際設計範圍。

使用案例 8 □（聯合兩個系統的）輸入與

更新服務申請

主要參與者：負責服務外部顧客的雇員

設計範圍：電腦系統，包括 Acura 與 BSSO

（請參見附圖）

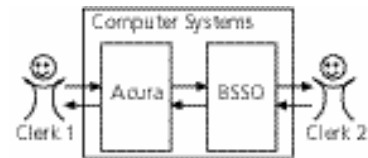
目標等級：目標摘要等級

主要成功情節：

1. 雇員在 Acura 中新增服務申請。
2. Acura 對 BSSO 通知新增服務申請。
3. 稍後某個時刻，雇員在 BSSO 中更新服務申請。
4. BSSO 對 Acura 通知更新服務申請。

這四個子使用案例都是使用者目標等級的使用案例，所以我們替它們加上海平面的符號。雖然它們都是系統使用案例，不過是針對不同系統所寫的，所以使用案例中的附圖也是針對不同系統的。在每個附圖中，我會畫出主要參與者，並且把 SuD 加上陰影。這時候，使用案例是黑箱式的，因為它們都是新工作的需求。此外，我也會用不同的動詞替使用案例命名，例如以動詞「通知」代表某個系統跟其他系統會同步。

↑ 英 43



使用案例 9 □（Acura 內的）新增服務申

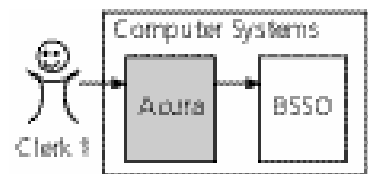
請

主要參與者：負責服務外部顧客的雇員

設計範圍：Acura

目標等級：使用者目標等級

接下來是使用案例的主體...



使用案例 10 (BSSO 內的) 通知新增

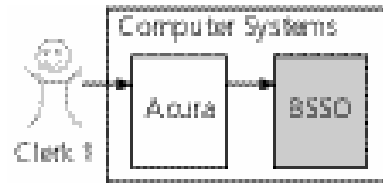
服務申請

主要參與者：Acura

設計範圍：BSSO

目標等級：使用者目標等級

接下來是使用案例的主體...



使用案例 11 (BSSO 內的) 更新服務申

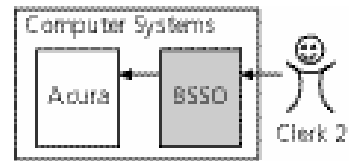
請

主要參與者：負責服務外部顧客的雇員

設計範圍：BSSO

目標等級：使用者目標等級

接下來是使用案例的主體...



使用案例 12 (Acura 內的) 通知更新

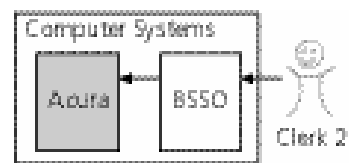
服務申請

主要參與者：BSSO

設計範圍：Acura

目標等級：使用者目標等級

接下來是使用案例的主體...



如果你用 UML 中的使用案例圖，你可能會畫出目標摘要等級的使用案例(譯註：使用案例 8)而不會畫這些附圖。不過，縱然有畫出使用案例圖，我們還是不能不標出這四個使用者目標等級的使用案例中讓人感到混淆之處，所以我們還是要小心標示出主要參與者、設計範圍與目標等級，必要時盡可能在使用案例中加上設計範圍附圖。

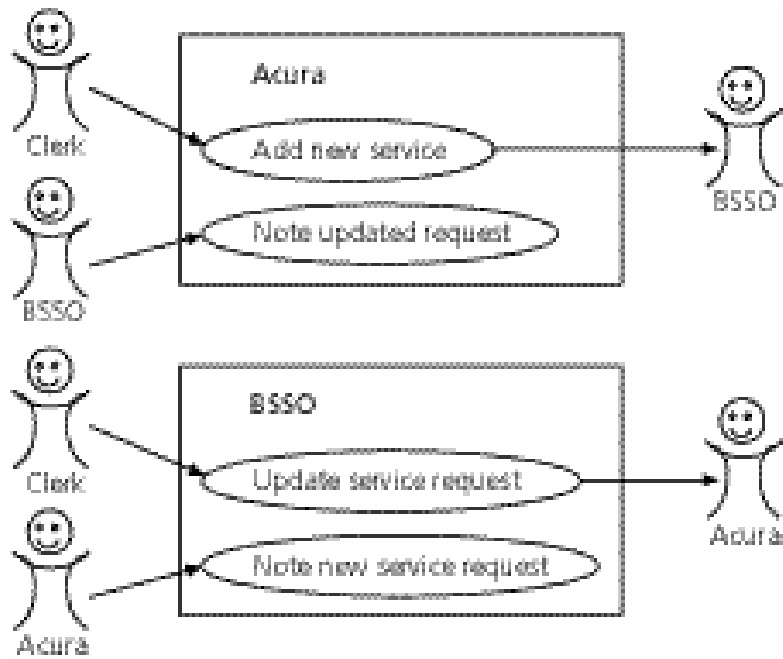


圖 3.2 針對 Acura—BSSO 而畫的使用
案例圖

這是 UML 中用來展示兩個系統間互動情形的一種畫圖風格。上方的圖展示 BSSO 是 Acura 中某個使用案例的支援性參與者，同時也是另一個使用案例的主要參與者。在下方的圖中，其角色則互換。

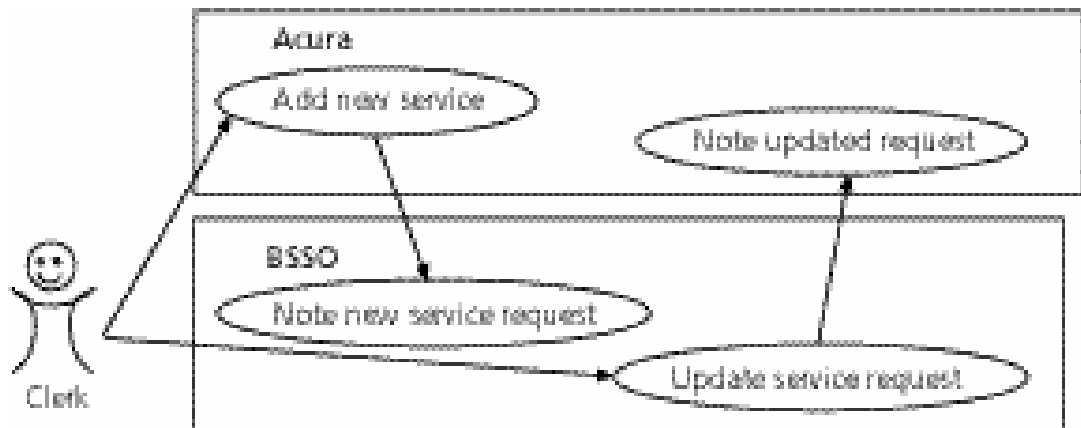


圖 3.3 把 Acura—BSSO 合併後的使用
案例圖

這裡的畫法把四個使用案例間的關係畫得非常清楚，不過畫法不是很標準，因為在圖中，某個系統的使用案例會觸發其他系統的使用案例（譯註：觸發使用案例的是參與者

或時間，不會是其他系統中的使用案例)。

就個人來說，我發現使用案例圖並無法減少多少混淆之處。因此，我可能會畫出像圖 3.3 這樣的非標準使用案例圖，以秀出兩個系統間的連接關係。這樣的圖雖然比較清楚，不過稍後卻很難維護。請畫出自己跟閱讀者間最容易溝通的圖。

↑ 英 45

(3) 描述具體細節的使用案例

這裡以比較複雜的狀況為例子，看看某個團隊是如何用使用案例把他們的設計框架記錄下來的。針對框架，他們一開始有 18 頁長、包含附圖的規則說明。他們認為這樣的文件很難閱讀，因此決定試著以使用案例來說明框架。

這個團隊準備花一個星期完成這個工作。首先，他們先粗略寫出 40 個使用案例，確定有捕捉到他們框架所能處理的所有需求。然後，靠著把部分使用案例改寫成使用案例中的擴充情節或資料變異情形清單，他們把使用案例數目減少到 6 個。因為大部分人的工作不會跟這個框架有關，所以可能無法很容易了解這些使用案例。不過，我假設某些讀者是技術方面的程式設計師，可能會想找到可以記錄他們設計結果的方法，所以我在書裡面秀出這些使用案例，讓大家知道這個團隊是如何記錄內部架構與使用變異情形清單的。以這些使用案例的問題複雜度來說，它們算是很容易讀的。請注意，在使用案例的說明文字中，所有參考到的子使用案例都會加上底線。謝謝 Calgary 公司的 Dale Margel 寫出這些使用案例。

總論：

整個架構必須要能處理並行執行的工作。為了達到這個目的，架構必須提供可處理執行緒 (thread) 與鎖定資源用的相關服務。並行服務框架 (CSF) 會提供這些服務。客戶端物件可以用 CSF 保護程式碼中的臨界區域，預防多重執程序 (process) 做出不安全的存取動作。

使用案例 13 對資源做可序列化的存取

動作

主要參與者：服務客戶端物件

設計範圍：並行服務框架 (CSF)

目標等級：使用者目標等級

主要成功情節：

1. 服務客戶端會跟資源鎖定服務要求取得某個資源的特殊存取權。
2. 資源鎖定服務會把資源的控制權傳給服務客戶端，讓它可以使用資源。
3. 服務客戶端使用這個資源。

4. 服務客戶端告知資源鎖定服務，表示它已經不再使用這個資源了。
5. 資源鎖定服務在服務客戶端用完資源之後會整理一下資源。

擴充情節：

- 2a. 資源鎖定服務發現服務客戶端對這個資源已經有存取權了：
 - 2a1. 針對這項請求，資源鎖定服務會應用資源鎖定轉換政策(使用案例 14)。
- 2b. 資源鎖定服務發現這個資源已經有人在使用了：
 - 2b1. 資源鎖定服務應用存取權相容性政策(使用案例 15)以決定是否要授權給服務客戶端。
- 2c. 資源鎖定服務的資源持有時間限制不等於零：
 - 2c1. 資源鎖定服務會啟動資源之持有計時器。

↑ 英 46

- 3a. 在客戶端還沒有告知資源鎖定服務它已經完成工作之前，資源之持有計時器的計時時間終了：
 - 3a1. 資源鎖定服務把例外事件丟給客戶端的處理程序。
 - 3a2. 執行失敗！
- 4a. 資源鎖定服務發現服務客戶端對資源的鎖定參考計數值大於零：
 - 4a1. 資源鎖定服務把服務請求的參考計數值減一。
 - 4a1. 執行成功！
- 5a. 資源鎖定服務發現沒有人在使用這個資源：
 - 5a1. 資源鎖定服務會應用存取權選擇政策(使用案例 16)以決定要授權給任何等待中的服務客戶端。
- 5b. 資源之持有計時器還在倒數計時中：
 - 5b1. 資源鎖定服務會取消資源之持有計時器的倒數動作。

技術與資料變異情形清單：

1. 客戶端所要求的存取權可以是：
 - 排它性的存取權
 - 共享的存取權
- 2c. 資源鎖定之持有時間的終止值可由下列指定：
 - 服務客戶端
 - 資源鎖定政策
 - 全域的預設值

使用案例 14 應用資源鎖定轉換政策

主要參與者：客戶端物件

設計範圍：並行服務框架 (CSF)

目標等級：子功能

主要成功情節：

1. 資源鎖定服務會證實資源請求所要求的是排它性的存取權。
2. 資源鎖定服務會證實服務客戶端已經擁有共享的存取權。
3. 資源鎖定服務會證實沒有其他服務客戶端正等著要升級存取權。
4. 資源鎖定服務把資源的排它性存取權授權給服務客戶端。
5. 資源鎖定服務會增加服務客戶端的鎖定計數值。

擴充情節：

- 1a. 資源鎖定服務發現請求是要求共享的存取權：
 - 1a1. 資源鎖定服務會增加服務客戶端的鎖定計數值。
 - 1a2. 執行成功！
- 2a. 資源鎖定服務發現服務客戶端已經具有排它性的存取權：
 - 2a1. 資源鎖定服務會增加服務客戶端的鎖定計數值。
 - 2a2. 執行成功！
- 3a. 資源鎖定服務發現其他服務客戶端正等著要升級存取權：
 - 3a1. 告知服務客戶端無法授權它所要求的存取權。
 - 3a2. 執行失敗！
- 4a. 資源鎖定服務發現其他服務客戶端正在使用這個資源：
 - 4a1. 資源鎖定服務讓服務客戶端等待資源存取權（使用案例 17）。

↑ 英 47

使用案例 15 應用存取權相容性政策

主要參與者：服務客戶端物件

設計範圍：並行服務框架（CSF）

目標等級：子功能

主要成功情節：

1. 資源鎖定服務會證實請求是要求共享的存取權。
2. 資源鎖定服務會證實資源的目前所有使用情形都是共享的存取權。

擴充情節：

- 2a. 資源鎖定服務發現請求是要求排它性的存取權：
 - 2a1. 資源鎖定服務讓服務客戶端等待資源存取權（使用案例 17）。

（這個執执行程序稍後會由資源鎖定服務策略恢復執行）

譯註：一般來說，當執执行程序需要等待資源存取權時，通常會從執行中（running）狀態進入待命中（waiting）狀態。

- 2b. 資源鎖定服務發現資源已經處於排它性的使用情況中：
 - 2b1. 資源鎖定服務讓服務客戶端等待資源存取權（使用案例 17）。

變異情形清單：

1. 相容性的準則可能會改變。

使用案例 16 應用存取權選擇政策

主要參與者：客戶端物件

設計範圍：並行服務框架（CSF）

目標等級：子功能

主要成功情節：

1. 資源鎖定服務會選出最早的等待中請求。
2. 資源鎖定服務會針對所選出來的請求，讓它的處理程序進入可執行狀態，然後把資源授權給它。

擴充情節：

- 1a. 資源鎖定服務發現目前並沒有等待中的請求：
 - 1a1. 執行成功！
- 1b. 資源鎖定服務發現等待中的請求是要從共享的存取權升級到排它性的存取權：
 - 1b1. 資源鎖定服務會選出這個要升級的請求。

↑ 英 48

- 1c. 資源鎖定服務發現選出要求共享存取權的請求：
 - 1c1. 我們會重複執行〔動作步驟 1〕直到發現下個請求是要求排它性的存取權為止。

變異情形清單：

1. 請求之選擇先後順序準則可能會變。

使用案例 17 讓服務客戶端等待資源存

取權

主要參與者：客戶端物件

設計範圍：並行服務框架（CSF）

目標等級：子功能

主要成功情節：

1. 資源鎖定服務會暫停服務客戶端的處理程序。
2. 服務客戶端會進入等待狀態直到它被恢復成可執行狀態為止。
3. 把服務客戶端的處理程序恢復成可執行狀態。

擴充情節：

- 1a. 資源鎖定服務發現等待時間的終止值有被指定：
 - 1a1. 資源鎖定服務會啟動計時器。

2a. 等待用的計時器計時終了：

2a1. 通知服務客戶端說我們無法授權給它所要求的存取權。

2a2. 執行失敗。

技術與資料變異情形清單：

1a1. 資源鎖定之持有時間的終止值可由下列指定：

服務客戶端

資源鎖定政策

全域的預設值

3.3 最外層使用案例

在前面的從企業設計範圍到系統設計範圍這個小節中，我曾建議大家可寫出兩個使用案例：其中一個是針對討論中系統而寫的、另一個則是針對最外層的設計範圍而寫的。現在，我們可以更明確指出：針對每個使用案例，請找出可包含它的最外層設計範圍，然後再針對這個最外層的設計範圍，寫出目標摘要等級的使用案例。

這個最外層使用案例是針對某個特定設計範圍所寫的。一般來說，針對原來使用案例的設計範圍，我們可找到範圍更大一點的設計範圍，讓原本使用案例的主要參與者還能處於這個設計範圍之外。如果持續擴大設計範圍的話，最後將到達某種程度的設計範圍。在這個情況下，如果設計範圍更大些，主要參與者就會變成在設計範圍之內。這時候的設計範圍就是我們所說的最外層設計範圍。有時候，最外層設計範圍是企業；有時候，它則是企業內的某個部門；又有些時候，可能只是電腦而已。通常，電腦部門是跟電腦安全相關的使用案例的主要參與者，行銷部門是跟廣告相關的使用案例的主要參與者，而顧客則是跟主要系統功能相關的使用案例的主要參與者。

↑ 英 49

一般來說，整個系統只會有 2 到 5 個最外層使用案例，有了這些最外層使用案例之後，我們就不需要把某些東西重複寫在兩個使用案例中。之所以會有這麼少的最外層使用案例，是因為每個最外層使用案例都是合併一些具有相同設計範圍、類似目標的主要參與者們，然後再把這些主要參與者的比較低層使用案例拉在一起。

本人極力建議大家寫出最外層使用案例，因為只要花非常少的時間就能替整組使用案例描繪出非常棒的情境。在最外層使用案例中，我們會秀出系統最後是如何幫助系統最外層的使用者；它們也能提供我們一份清單，讓我們可以瀏覽系統行為。

現在，讓我們看一下 MyTelCo 與它的 Acura 系統的最外層使用案例。

MyTelCo 決定讓顧客透過網頁直接存取 Acura 系統，以降低雇員的工作負荷。Acura 系統也會提供雇員的銷售績效相關報告。還有，我們需要由某人

來設定顧客與雇員的安全存取權等級。因此，我們將會有四個使用案例：（由顧客做的）*新增服務申請*、（由雇員做的）*新增服務申請*、*產生銷售績效報告*與*管理安全存取權*。

我們將會寫出四個以 Acura 系統為討論中系統的使用案例。這時候，我們需要找出每個使用案例的最外層設計範圍。

很明顯地，顧客是在 MyTelCo 之外，所以我們會有一個最外層使用案例是以顧客為主要參與者、MyTelCo 為設計範圍。這個最外層使用案例將是目標摘要等級、把 MyTelCo 視為黑箱、會回應顧客申請、完成申請服務等。事實上，這個使用案例就是我們之前所寫的使用案例 6（企業的）*新增服務申請*。

雇員是在 MyTelCo 裡面。（職員的）*新增系統特性*使用案例的最外層使用案例會以所有電腦系統作為設計範圍。這個最外層使用案例中將匯總雇員跟所有電腦系統間的所有互動情形。我預期雇員的所有使用者目標等級的使用案例都將涵蓋在這個最外層使用案例中。

*產生銷售績效報告*使用案例則會以行銷部門作為最終主要參與者。它的最外層使用案例會以服務部門作為設計範圍，秀出行銷部門跟(1).所有電腦系統與(2).服務部門間的互動情形，以設定績效獎金的計算方式、產生銷售績效報告等。

*管理安全存取權*使用案例則會以安全部門或 IT 部門作為最終主要參與者，而且它的最外層使用案例會以 IT 部門或所有電腦系統作為設計範圍。這個最外層使用案例中會參考安全部門用所有電腦系統的方式，以設定或追蹤安全相關議題。請注意，在這四個最外層使用案例中，我們已涵蓋所有在安全性、行銷、服務與顧客各方面使用 Acura 的情形。縱然 Acura 系統中會有數以百計的低階使用案例要寫，不過，除了這四個使用案例之外，我們不太可能會寫出其他的最外層使用案例。

↑ 英 50

3.4 使用定義專案範圍用的工作成果

當你在定義未來系統的功能範圍時，可能會用腦力激盪方式找出系統的所需功能，也會在白板上各種不同工作成果間來回工作。在白板某個地方，你可能會用專案範圍內／外清單來追蹤跟專案範圍討論相關部分（例如「不，Bob，我們認為新的列印系統應該在專案範圍之外 — 或者我們應該重新看看專案範圍內／外清單中的這個項目嗎？」）。這份清單上會有參與者跟它們的目標。此外，你也會有一張畫出設計範圍的圖，裡面秀出在討論中系統互動的人、組織與系統。當你在這些工作成果間來回工作時，你會發現自己的想法會逐漸成熟，知道新系統中應該要做哪些事。你會認為自己知道設計範圍究竟為何，不過專案範圍內／外清單中的變動可能會影響到設計範圍邊界。當我們有一個新的主要參與者時，目標清單也會跟著變動。

你可能會很快或稍後才發現到你還需要第四種工作成果：新系統的 *專案願景聲明*

(vision statement)。專案願景聲明可用來維繫我們的整個討論內容。它是可用來幫我們決定某個東西該在專案範圍之內或外的第一個東西。

當你定義好未來系統的專案範圍之後，會有四種跟系統的專案範圍相關的工作成果：

- ◆ 專案願景聲明
- ◆ 設計範圍附圖
- ◆ 專案範圍內／外清單
- ◆ 參與者 — 目標清單

從這小節的簡短討論中，我希望大家能夠了解到這四種工作成果是會交互影響的，而且當你在產生工作的設計範圍時，很可能會改變它們。

3.5 習題

設計範圍

3.1. 根據後面的使用者故事片段，請至少找出五個可能的系統設計範圍：

「...Jenny 站在銀行的 ATM 前面。那裡很暗。她輸入 PIN 號碼，並試著找到 Enter 鍵...」

3.2. 替 ATM 畫出多種可能會有的設計範圍圖，圖中會包括硬體與軟體。

↑ 英 51

3.3. 你所寫出來的需求是為什麼系統而寫的？它的邊界為何？裡面會有什麼東西？什麼東西會在它的外面，跟它溝通？什麼系統會包含這個系統，什麼東西又會在前者的外面，跟它溝通？請替包含這個系統的外層系統命名。

3.4. 請替個人顧問／理財（PAF）系統畫出多種可能的設計範圍圖（請參見習題 4.4）。

3.5. 請替網頁應用程式畫出多種可能的設計範圍圖。這個應用程式會從使用者的工作站、經由網路連到你公司的網站伺服器，伺服器又會連到既有的主機系統。

3.6. 請說明設計範圍為企業、白箱式的企業使用案例跟設計範圍為企業、黑箱式的企業使用案例兩者間的差異。

↑ 英 52

第4章關係人與參與者

關係人代表跟合約有關係的人。參與者則代表有行為的東西 — 就如某個學生所說的：「如果它聲稱具有某個行為的話，就必須要能執行這個行為。」參與者有可能是人、公司或組織、電腦程式，或者是電腦系統 — 包括硬體、軟體或兩者都有。

請找找看下列的參與者是否存在：

- ◆ 系統的**關係人** (stakeholder)
- ◆ 某個使用案例的**主要參與者** (primary actor)
- ◆ **討論中系統** (system under discussion) 本身
- ◆ 某個使用案例的**支援性參與者** (supporting actor)
- ◆ **內部參與者** (internal actor) — 它是 SuD 內的某個元件

譯註：大家在閱讀本章內容時，把圖 2.5 牢記在心，必能保持清晰的概念。

4.1 關係人

對關係人來說，使用案例中所執行的行為必須要能保障這個人或這個東西的利益。

當然，所有的主要參與者都是關係人。不過，縱然關係人有權力關心系統該如何運作，有某些關係人可能從不會跟系統直接互動。例如系統的擁有者、公司的董事會或管理性組織（例如內部收益服務與保險部門）都是如此。

學生會用暱名來稱呼那些從未直接出現在使用案例的動作步驟中的關係人，我們把它們稱為**幕後參與者** (offstage actor)、**第三級參與者** (tertiary actor) 或**沉默參與者** (silent actor)。

把注意力放在沉默的參與者可顯著改善使用案例品質，因為這些參與者的利益可讓我們知道系統所需執行的檢查與驗證動作、所產生的系統紀錄，以及使用案例中所需執行的動作步驟。因為我們必須為關係人加強系統行為，所以才需要把企業規則記錄下來。使用案例中需展現出系統是如何保障關係人的利益。下面的小故事告訴我們漏掉關係人的利益時會付出怎樣的代價。

↑ 英 53

◆ 一個很短卻真實的故事

這家公司在賣了好幾份新系統、經過一年的運轉之後，開始收到顧客的系統變動需求。結果看起來似乎很自然。不過，當他們上過使用案例的課之後，他們被要求用腦力激盪方式，針對這個最近交給顧客的系統，找出關係人與其利益。

讓他們感到非常驚訝的是：他們在腦力激盪過程中發現，自己竟然可以不知不覺地寫出這些最近才收到的變動需求。很明顯地，當他們在開發系統時，完全忽略掉了某些關係人的一些利益。不用太久，利益被忽略掉的關係人就開始察覺到，

系統其實並沒有為他們提供適當的服務，因此變動需求才會開始進入公司。有了這次教訓之後，專案領導者開始堅信他們應該儘早發現這些關係人與其利益，以避免重複發生代價慘痛的錯誤。

我的同事跟我都發現：只到我們詢問關係人與其利益，就可及早發現一些有顯著影響、之前卻沒有提到的需求。這樣做不需要花許多時間，卻可節省往後需花費的很多功夫。

4.2 主要參與者

使用案例的主要參與者代表那些會去要求系統提供它服務的關係人。它會有跟系統相關的目標，而且系統運作結果也必須滿足這些目標。主要參與者通常是會驅動使用案例的參與者，不過有時候會有例外（譯註：使用案例可能是由時間或其他中介者所驅動）。

一般來說，使用案例之所以會被啟動是因為主要參與者送訊息給它、按下一個按鈕、按下一個鍵，或者用其他方式來啟動敘事情節。然而，有兩種常見情況是使用案例啟動者並非主要參與者。第一種情況是類似公司雇員或電話接線生這樣的主要參與者，他們是因為其他人（譯註：例如顧客）的行為而去啟動使用案例的；第二種情況是由時間來觸發使用案例。

公司雇員或電話接線生通常是某個**最終主要參與者**（ultimate primary actor）為了技術上便利而產生的代理人，後者才是真正在乎執行結果的人。隨著開發技術變遷，最終主要參與者現在已經越來越可能自己用網路或自動電話系統來啟動或觸發使用案例。用電話提出申請的顧客是最終主要參與者的一個例子。在以網路重設計的系統（如 Amazon.com）中，顧客會直接自行輸入申請內容。

↑ 英 54

同樣地，行銷或稽核部門可能會抗拒某些由雇員所操作的使用案例。執行這些使用案例並不是雇員的目標；使用案例的操作者——雇員可能只是行銷經理的便利代理人而已。在其他不同工作環境中，行銷經理可能會自行執行使用案例。

例如我前些日子所寫的「顧客的銷售代表」或「行銷部門的雇員」，這些句子裡面所寫出來的系統使用者實際上是代替其他人做事的。從這裡我們可知道一點：雖然我們會設計出雇員的使用者介面與其安全許可權限，不過顧客或行銷部門才是真正關心結果的人。

時間是另一個系統沒有直接觸發者的情況。在每天午夜或每月月底，不需要有雇員就可自動觸發某些使用案例。在這個情況下，只要看看哪些關係人會關心使用案例的執行結果，就可以很容易地找到主要參與者。

針對「使用者 vs. 最終主要參與者」這個主題，我們可以討論很久。因此，我建議你不要花太多時間在它上面，或者你也可在公開論壇中討論這個主題。當開發團隊開始調查使用者介面的設計方式時，他們將（或應該）把適量的時間花在研究真正在操作系統使用者的特徵上。等到他們重審需求時，去了解每個使用案

例的最終主要參與者（換言之，就是真正關心使用案例結果的人）是誰就是一件非常有用的事（譯註：最終主要使用者就是使用案例的關係人，使用案例必須保障其利益）。

有一位很精明的學生問過我：「如果我們這時候弄錯了主要參與者，會造成多嚴重的傷害呢？」答案是「不太多」，下一個小節會解釋原因。

為何主要參與者有時候不重要、有時候又很重

要

只有在一開始蒐集需求或把系統交給使用者時，主要參與者才會很重要。在這兩個時間點之間的時候，它們就沒有那麼重要了。

剛開始產生使用案例時

這時候，列出主要參與者可讓我們很快地在短時間內瀏覽過整個系統（這樣做可讓我們很快逃離系統需求泥沼）。我們會先以腦力激盪法替所有參與者命名，然後繼續用腦力激盪法找出所有目標名稱。這些目標都是我們感興趣的，不過如果我們想直接找出這些目標的話，很可能就會漏掉很多目標。先用腦力激盪法找出主要參與者可幫我們建立工作時的整體結構。藉由瀏覽這個工作結構，我們可得到比較好的目標清單。

產生數目稍多的主要參與者對我們並不會造成什麼傷害，因為在最壞的情況下，最多只是重複產生相同目標而已。當我們查看參與者與目標以決定工作的優先順序時，就可找到重複目標並移除它們。

然而，縱然我們很小心、重複進行兩次腦力激盪以找出目標，不過還是不能保證可寫出系統所需支援的所有目標。當我們在寫使用案例中處理失敗情況的動作步驟時，很可能還會找到一些新使用案例，這些使用案例是無法在早期就找到、避免發生這種情況的。最好的做法只能先列出所有主要參與者，然後盡可能找出所有目標而已。

↑ 英 55

豐富的主要參與者清單可帶給我們三個額外好處：

- ◆ 它讓我們把心思放在使用系統的人身上。在需求中，我們會描述哪些人是我們所預期的主要參與者、說明他們的工作內容以及他們的典型背景環境與專業技能。做這樣的事可以讓使用者介面設計師與系統設計師所設計出來的系統符合這些人的專業需要。
- ◆ 它讓我們設定好參與者 — 目標清單用的結構，而參與者 — 目標清單又可用來決定開發工作的優先順序，並藉此切割工作。

- ◆ 它可讓我們把一大群使用案例切割成不同套件，指定給不同設計團隊做。

在寫使用案例與設計時

一旦我們開始發展使用案例的細節時，主要參與者就變得不太重要了，這點讓人感到很驚訝。之所以如此，其原因是：隨著時間過去，使用案例寫作人員可能會發現到有許多參與者都會用到某個使用案例。例如，任何比雇員職位更高的人都可接電話、跟顧客講話。因此，這時候寫作人員通常會開始用比較一般化的方式替主要參與者命名，例如他們可能會用一些像損失申請收件人、訂單收件人或開發票人的角色名稱。我們的使用案例唸起來可能會像這樣：開發票人開出一張發票或訂單收件人收到一張訂單...（這樣的主要參與者名稱並不能給我們什麼提示）。

譯註：一開始，我們通常會先以工作職稱作為某個使用案例的主要參與者。不過，實際上可能會有多個工作職稱都可執行某個使用案例。為了解決這個問題，可以把主要參與者當成這些人戴的帽子，找出比較一般化的角色，就不會受到工作職稱的刻板印象影響。

以工作職稱作為角色名稱的主要參與者並無法涵蓋能執行某個使用案例的所有角色。針對這個問題，我們可以有幾種方式來處理它，每種方式都各有優缺點，沒有說哪一種策略絕對最好，你必須自行選一種來用。

方案 1。根據原先以工作職稱為主的角色找出它們所潛藏扮演的角色，作為新的主要參與者。產生一個參與者 — 角色對照表（actor-role table），裡面的角色欄是所有不同的人與系統（它們是任意使用案例的原先主要參與者），而主要參與者欄放原先主要參與者所潛藏扮演的一般化角色（譯註：新的主要參與者），然後使用主要參與者欄中的角色名稱。用這樣的參與者 — 角色對照表建立使用案例跟現實世界中的人與系統間的關係。

這個策略允許寫作人員忽略錯綜複雜的工作職稱，用一般化的角色簡化他們的寫作工作。有些人（有可能是使用者介面設計師或軟體套件的封裝人員）可能會用這份參與者 — 角色對照表找到使用案例的實際使用者。使用方案 1 的缺點是我們要維護並閱讀一份額外的表格。

方案 2。在寫全部使用案例之前的某個小節中，我們可寫出類似這樣的說明：「經理除了可做雇員可執行的使用案例之外，還可執行更多的使用案例。區域經理除了可做經理可執行的使用案例之外，還可執行更多的使用案例。因此，當我們表示某個使用案例的主要參與者是雇員（舉例）時，我們應該知道任何比它的職位更高的人 — 在這個例子中是經理與區域經理 — 也可執行這個使用案例。」

↑ 英 56

寫出這樣的東西比維護一份參與者 — 角色對照表還容易些，因為它比較不容易變。不過，它的缺點是大家要花比較多的時間記住：當雇員是某個使用案例的主要參與者時，經理也將可執行這個使用案例。

大家可綜合這兩種策略達到比較可接受的結果。如果可接受的話，我會採用第二種策略，因為這樣可寫出審查與維護工作量比較少的工作成果。

重點是：使用案例範本中的**主要參與者**欄位隨著時間過去會越來越沒有價值。這是正常的，不用太擔心它。

系統設計之後、準備配置時

把系統交給使用者之前，主要參與者再度變得很重要。這時候，我們需要列出所有的人以及他們可執行的使用案例。因為這時候我們需要：

- ◆ 把系統封裝成不同套件，以便讓它們可載入到使用者的各式各樣機器中
- ◆ 訂定每個使用案例的安全等級（網路使用者、內部使用者與管理者等）
- ◆ 產生不同使用者群體所需的訓練內容

參與者 v.s. 角色

參與者這個字隱含對系統有動作的某個人。有時候，它代表使用案例裡面的一個人；有時候，它可能代表同一類型的很多人，這些人都扮演相同的角色。

假設 Kim 是 MyTelCo 的一位顧客、Chris 是公司裡的雇員，而 Pat 則是銷售經理。他們其中任何一個人都可以下訂單。如果用**參與者**這個術語來說明，我們可說 Kim、Chris 與 Pat 都是下訂單使用案例的主要參與者（譯註：個人 vs. 主要參與者），也可以說顧客、雇員與銷售經理都是下訂單使用案例的主要參與者（譯註：工作職務角色 vs. 主要參與者）。我們還可能會說：「銷售經理可執行雇員所能執行的任何使用案例。」這些都是很好的講法。

譯註：這段表示個人、工作職務角色都可當做主要參與者的名稱。不過，就像前面所提過的一樣，這樣的角色並無法涵蓋所有可能執行使用案例的所有角色，可是如果用比較一般化的角色，我們又必須維護一份參與者 — 角色對照表。

如果以**角色**這個詞來說明，我們可以說 Kim、Chris 與 Pat 都是個人參與者（譯註：個人 vs. 主要參與者），他們其中任何一人都可扮演顧客的角色，不過只有 Chris 與 Pat 可扮演雇員的角色，而 Pat 才可扮演銷售經理的角色（譯註：個人 vs. 角色）。然後，我們可以說驅動下訂單使用案例的角色是**訂單收件人**，而顧客、雇員或銷售經理又都可扮演**訂單收件人**這個角色（譯註：工作職務角色 vs. 一般化角色）。這種說法可能比上一段的說法更精確點，有些人也比較喜歡採用這種說法。不過，在使用案例的世界中，並沒有所謂的標準說法。

重點是：你應該用自己開發團隊比較喜歡的術語。在稍早「為何主要參與者有時候不重要、有時候又很重要」這個小節中，我曾經解釋過不應該放太多精神在這個議題上，也說過發生某些情況時該如何處理它。目前，**參與者**是業界能接受的術語，而且它也夠用，所以我在本書中採用參與者而不用角色。

用統一模型語言畫的圖與參與者／角色的特殊化關係

在 UML 中，我們可以用中空的箭頭建立某個參與者跟另一個參與者間的**特殊化關係** (specialization) (請參見圖 A.6 修正過的關閉一筆大交易使用案例)。

這種箭頭的好處是它可讓你很簡潔地表達出經理可以做雇員可做的任何事：我們只要簡單畫出一個箭號就可以了，讓箭頭指向**雇員**、箭尾指向**經理**。

壞處是對許多人來說，這種畫法的結果讓人感到卻步。大家會以為圖中想表達的是：經理是一種特殊的雇員或者把雇員當成一種特殊的顧客（實際上，這種畫法是在宣稱某個人可以做另一個人可做的任何事）。他們會誤認為經理**不只是**雇員（譯註：經理這個職稱不應該是特殊的雇員，他們的工作性質完全不同，只是前者允許做後者的工作，所以作者不認為經理「不只是」雇員）。對圖有這樣的反應並不是什麼大事，不過你還是要想清楚，不要發生誤解。

特殊化箭頭並不是萬能的，有時候它無法解決主要的參與者 — 角色問題。例如，**銷售雇員**跟**稽核雇員**可執行一些同樣的使用案例，我們不能在它們之間畫上特殊化箭頭，因為它們彼此都不能做所有對方能做的事。因此，我們又會回到參與者 — 角色的爭論當中。

列出主要參與者的特徵

對設計師來說，只有參與者清單是不夠的，我們還需要知道使用者的技能為何，這樣一來，所設計出來的系統行為與使用者介面才能符合使用者需求。手頭上握有一份參與者描述表 (actor profile table) 的開發團隊會比較清楚自己所設計出來的軟體是如何符合最終使用者的需要。因為當他們在進行開發工作時，知道最終使用者的技能為何。

最簡單的參與者描述表只會有兩欄，就像表 4.1 所示。有些人還會多列出大家所熟知的參與者其他名稱或**別名**。其他不同版本的參與者描述表可看 *Software for Use* (Constantine and Lockwood, 1999) 中的討論。

表 4.1 參與者描述表範例

名稱	概況：背景環境與技能
顧客	人們在街上可用觸控式螢幕操作系統，不過我們不能預期顧客都可輕易操作精細的 GUI。他的閱讀能力可能會有問題、有近視或有色盲等。

負責處理退貨的雇員	這樣的人會持續使用這個軟體。他是習慣打字、會使用複雜功能的使用者，可能會希望能自定使用者介面。
經理	偶而使用系統的使用者，曾經用過 GUI，不過並不熟悉任何特定的軟體功能。這些人的個性通常很急。

↑ 英 58

4.3 支援性參與者

使用案例中的支援性參與者是外部參與者，它會提供服務給討論中系統。它可能是高速的列表機、網路服務，或者是進行某些研究並把結果交回給我們的某個人（例如，驗屍機構會提供保險公司服務，以確認某人已經死亡）。我們過去把這樣的參與者稱為次要參與者（secondary actor），不過大家發現這樣的術語會造成混淆。現在，有越來越多人使用支援性參與者這個術語，這是比較自然的術語。之所以要找出支援性參與者是為了辨別出系統將使用的外部介面，以及橫跨這些介面的通訊協定為何。它會為我們帶來其他需求，例如資料格式與外部介面（請參見圖 1.1）。

參與者有可能既是某個使用案例的主要參與者，又是另一個使用案例的支援性參與者。

4.4 討論中系統

討論中系統本身也是參與者 — 它是一種特殊的參與者。我們通常會用它的名稱來稱呼它，例如 Acura，也可能會直接稱它為討論中系統（system under discussion）、設計中系統（system under design）或 SuD。我們會把這個名稱或描述寫在使用案例中的設計範圍欄位。

雖然 SuD 是參與者，不過它並不是任何使用案例的主要參與者或支援性參與者。前面的「設計範圍」小節中對 SuD 做過不少討論，大家可以回顧一下。

4.5 內部參與者與白箱式使用案例

大部分時候，我們會把討論中系統視為黑箱。這時候，我們看不到討論中系統的內部為何，也故意不去提內部參與者。當我們用使用案例替還未設計出來的系統寫出需求時，這種做法是相當好、有意義的。

不過，我們偶而可能希望用另一種使用案例格式來記錄系統裡面的各個部分是如何合作，以達成我們所期待的正確行為。當我們在記錄企業流程時，可能會採用這種做法，就像使用案例 5 買東西（正式格式版本）與使用案例 19（企業的）

處理求償一樣。當我們想秀出包含多個電腦的系統、比較大的設計範圍時，可能也會採用這種做法，例如使用案例 8（聯合兩個系統的）輸入與更新服務申請。在這些情況中，我們會把系統中的元件視為參與者。

當我們往系統裡面看，並且替系統中的元件命名、描述它們的行為時，會把系統視為白箱。這時候，跟使用案例寫作有關的所有準則還是成立；只不過，我們會同時討論內部參與者與外部參與者的行為。在白箱式使用案例中可能不只有兩個參與者，因為除了外部參與者之外，我們還會秀出系統中的元件。

↑ 英 59

當我們在設計電腦系統時，把白箱式的使用案例當成系統的行為需求是很罕見的做法，而且通常是錯的。

4.6 習題

參與者與關係人

- 4.1 請針對販賣機找出一個使用案例，這個使用案例的主要參與者是販賣機的擁有者。
- 4.2 假設你被雇來寫出一個新 ATM 的需求文件。請試著決定下列清單中的每個項目究竟是關係人、主要參與者、支援性參與者、討論中系統，或者都不是（或者屬於上述多種參與者）：

ATM

顧客

ATM 卡

銀行

ATM 的前方面板

銀行擁有者

服務人員

印表機

銀行主電腦系統

銀行出納員

搶銀行的強盜

- 4.3 ATM 是更大系統中的一個元件。事實上，它可能是好幾個更大系統的其中一部分。請針對所有會包含 ATM 的更大系統，重複前面的習題。
- 4.4 個人顧問公司（一個想像的公司）將推出一個新產品，它可讓大家檢視自己的理財投資策略，例如退休、教育基金、土地與股票。個人顧問／理財（PAF）產品會以 CD 形式推出。使用者會用 CD 安裝它，然後執行各種理財情節，以學習如何把自己的理財計畫最佳化。PAF 也可以求助於各種稅務套裝軟體（例如 Kiplinger Tax Cut），以決定目前稅法為何。個人顧問軟體跟 Kiplinger 軟體間已經有協定存在，兩種軟體可直接交換資料。它也跟各式各樣的共同

基金與網路股票服務有協定在（例如 Vanguard 與 E*Trade），可透過網路直接買賣基金或股票。這家公司認為讓這個版本的 PAF 具有網路功能是很棒的想法，而且它會以使用量為基礎對使用者收費。

請找出 PAF 的參與者、主要參與者、支援性參與者、討論中系統與包含它的系統（會把 PAF 當成元件的系統），並替它們命名。

第三部分給忙碌者的一些寫作提示

第20章跟單一使用案例有關的寫作提示

寫作提示 1：使用案例是一種散文

還記得我們在前言提過：「使用案例基本上是一種散文寫作。想寫出好的使用案例時可能會遇到的所有困難，基本上都跟散文寫作時遇到的一樣。」

FirePond 公司的 Russell Walters 寫道：

Recall from the preface, "Writing use cases is fundamentally an exercise in prose essays, with all the difficulties in articulating good that comes with prose writing in general."

Russell Walters of FirePond Corporation wrote:

我認為這一段話可以很清楚抓出問題點。這是大家對使用案例的最大迷思所在，而且對實際寫使用案例的人來說，這可能是最大的啓發。然而，這本書還未出版、大家還沒讀過它之前，我不認為寫使用案例的人可以自行體會這個道理。就我自己而言，也是到有機會跟你一起工作之後，才了解這個根本問題。事實上，我已經用使用案例這個概念工作有四年之久了。剛開始跟你一起工作時，我還沒有辦法馬上體會這個道理，直到有機會協助你進行重寫使用案例（使用案例 36）的工作時，去分析並審查使用案例改善前後的不同版本，心裡面那盞燈才算是亮了。四年多的時間真的是一段漫長的等待啊！所以，如果有一個問題是本書讀者一定要了解的，我希望他能體會這個能幫助大家有效寫出使用案例的基本問題。

藉由這個寫作提示，我希望大家能把焦點放在使用案例的說明文字，而不是使用案例圖。同時，也希望大家能對所看到的各種寫作風格，有所了解。

Use this reminder to help keep your eyes on the text, not the diagrams, and to be aware of the writing styles you will encounter.

寫作提示 2：把使用案例寫的很容易讀

希望大家把需求文件寫得短一點、清楚些，而且很易讀。

有時候，我覺得自己有點像國中二年級的英文老師，邊走邊說：

請用現在式、主動語態寫出句子來。不要用被動語氣，請用主動語氣。句子的主詞在哪裡呢？請寫出真正的需求；如果不是需求的話，就不要把它寫出來。

上面所提的那些東西可以讓需求文件短一點、清楚些，而且很易讀。下面則是一些值得大家好好培養的良好習慣，它們可讓你的使用案例短一點、清楚些、很易讀：

You want your requirements document short, clear, and easy to read.

I sometimes feel like an eighth-grade English teacher walking around, saying,

Use an active verb in the present tense. Don't use the passive voice, use the active voice. Where's the subject of the sentence? Say what is really a requirement; don't mention it if it is not a requirement.

Those are the things that make your requirements document short, clear, and easy to read. Here are a few habits to acquire to make your use cases short, clear, and easy to read:

1. 讓東西維持短短的，寫出重點就好。長的使用案例只會導致長的需求文件，很少人會喜歡去讀它。

Keep matters short and to the point. Long use cases make for long requirements, which few people enjoy reading.

2. 從目標等級最高的使用案例開始，寫出有一致性的敘事情節。目標等級最高的是目標摘要等級使用案例。使用者目標等級與子功能目標等級的使用案例會從那裡分流出來。

Start from the top and create a coherent story line. At the top will be a strategic use case. The user goal and eventually subfunction-level use cases branch off from here.

3. 用短的動詞片語替使用案例命名，說明這個使用案例會達成的目標。

Name the use cases with short verb phrases that announce the goal to be achieved.

4. 寫使用案例時，請從觸發事件開始寫，一直寫到目標達成或放棄為止，而且針對交易行為，系統也要有必要的紀錄存在。

譯註：在使用案例中，主要成功情節會有觸發事件；另一方面，擴充情節的擴充情況就是觸發事件。

Start from the trigger and continue until the goal is delivered or abandoned and the system has done any bookkeeping it needs to with respect to the transaction.

5. 用主動動詞寫出完整句子，裡面描述欲達成的子目標。

Write full sentences with active verb phrases that describe the subgoals being completed.

6. 請確定每個動作步驟中都有明確寫出參與者與其意圖。

Make sure the actor and the actor's intent are visible in each step.

7. 請把失敗情況突顯出來、讓它的復原動作變得很易讀。同時也要讓大家清楚知道事情的發生先後順序，最好不要用寫出動作步驟編號方式說明接下來會

發生什麼事（譯註：也就是不要在情節結尾處說明接下來會執行動作步驟 xx）。

Make the failure conditions stand out and their recovery actions are readable.

Let it be clear what happens next, preferably without having to name step numbers.

8. 把替代行爲放在擴充情節中，請不要在使用案例的主要成功情節中用條件子句寫出替代行爲。

Put alternative behaviors in the extensions rather than in if statements in the main body.

9. 只在非常少、特別選過的環境下使用擴充使用案例。

Create extension use cases only under very selected circumstances.

寫作提示 3：只用一種句型

在使用案例中，請只用下面這種句型寫出動作步驟：

- ◆ 現在式。
- ◆ 主動語態下的主動動詞。
- ◆ 描述參與者成功達成子目標，讓整個過程往前邁進一步的情形。

下面是一些例子：

There is only one form of sentence used in writing action steps in the use case:

A sentence in the present tense,
with an active verb in the active voice,
describing an actor successfully achieving a goal that moves the process forward.

Here are examples:

顧客輸入 ATM 卡與 PIN 號碼。

系統證實顧客有被授權過，而且帳戶中有足夠的錢可領。

PAF 會解讀從網站來的回應資訊，以更新使用者的投資組合。

雇員用搜尋「損失」明細方式找尋損失。

不論你採用正式或非正式範本，都請用這樣的句型寫出企業使用案例、系統使用案例，或者目標摘要等級、使用者目標等級與子功能目標等級的使用案例。而且不論是寫主要成功情節或擴充情節中的情節片段都是一樣的。請精通這種句子的寫作風格。

↑ 英 206

另一方面，用其他不一樣的文法句型來寫出擴充情況是很有幫助的。因為這樣一來，我們就不會把它跟動作步驟搞混了。你可以用不完整的句子（當然，還是可以用完整句子）、（盡量使用）過去式來寫出擴充情況，並且在擴充情況結尾處用分號（：）而不用句點。

Use this sentence form in business use cases, system use cases, and summary, user, and subfunction use cases, with either the fully dressed or casual template. It is the same in the main success scenario and in the extension scenario fragments. Master this sentence style.

It is useful for the condition part of an extension to have a different grammatical form so it doesn't get confused with the action steps. Use a sentence fragment (or possibly a full sentence), preferably (but not always) in the past tense. End the condition with a colon (:) instead of a period.

輸入 PIN 號碼的等待時間終了：

錯的密碼：

沒找到檔案：

使用者在中途離開：

送出來的資料不完整：

寫作提示 4：「包含」某些子使用案例

如果沒有人告訴你其他做法的話，你應該很自然就會這麼做：寫出一個動作步驟，然後在裡面呼叫其他等級比較低的目標或使用案例名稱，例如：

雇員用搜尋「損失」明細方式找尋損失。

以 UML 術語來說，呼叫其他使用案例就是包含子使用案例。如果使用案例寫作人員或老師沒有鼓勵大家去使用 UML 中的擴充關係或特殊化關係的話（請參見附錄 A），相信就算沒有人教，大家也一定會去使用包含關係。

根據經驗法則：請只用使用案例間的包含關係，而不要去用其他擴充關係或特殊化關係。遵從這個法則的人告訴我們說他們跟讀者對只使用包含關係的使用案例比較不易迷惑，而對混合擴充關係與特殊化關係的使用案例比較容易迷惑。請參見 10.2 中的何時該用擴充使用案例小節。

What you would do quite naturally, if no one told you to do otherwise, would be to write a step that calls out the name of a lower-level goal or use case, as in

Clerk finds a loss using search details for "loss."

In UML terms, the calling use case just included the sub use case. This is so much the most obvious thing to do that it would not even deserve mention if there weren't writers and teachers encouraging people to use the UML extends and specializes relations (see Appendix A).

As a first rule of thumb, always use the includes relation between use cases. People who follow this rule report that they and their readers have less confusion with their writing than people who mix includes with extends and specializes. See the subsection When to Use Extension Use Cases on page 116.

寫作提示 5：球在誰的手上？

有時候，大家所寫的使用案例會用被動語態或從系統本身的觀點往外看。結果就產生像**信用額度已輸入**的句子，這樣的句子裡面不會提到究竟是誰做輸入這件事。

請從鳥瞰觀點來寫使用案例，注視並記錄場景。你也可用表演的形式來寫，說明這些參與者的相關動作。或者暫時假裝自己正在描述一場足球賽，比賽中參與者 1 先拿球、盤一下球，然後把球踢給參與者 2；參與者 2 再把球傳給參與者 3 等等。

請在句子的前一兩個字很快點出做動作的參與者名稱。不論發生什麼事，請確認使用案例中總是很清楚表達出球在誰的手上。

↑ 英 207

Sometimes people write in the passive voice or from the point of view of the system itself, looking out at the world. This produces sentences like Credit limit gets entered. This sentence doesn't mention who is doing the entering.

Write from the point of view of a bird up above, watching and recording the scene. Or write in the form of a play, announcing which actor is about to act. Or pretend for a moment that you are describing a soccer game, in which actor 1 has the ball, dribbles it and then kicks it to actor 2; actor 2 passes it to actor 3; and so on.

Let the first or second word in the sentence be the name of the actor who owns the action. Whatever happens, make sure it is always clear who has the ball.

寫作提示 6：寫出合適的目標等級

- ◆ 請重看一下 5.5 *找到合適的目標等級*，裡面對此有完整的討論。
Review Section 5.5, Finding the Right Goal Level, for the full discussion.
- ◆ 請確認使用案例有加上正確的目標等級標籤：目標摘要等級、使用者目標等級或子功能目標等級。
Make sure that the use case is correctly labeled with its goal level: summary, user, or subfunction.
- ◆ 請定期審查你所寫的使用案例，確認自己還很清楚知道「海平面」目標在哪裡，也知道動作步驟離海平面以下（或上）有多遠。回想一下海平面目標的測試方式：
 - 它可以由某人、在某個地方、某段時間內（2 到 20 分鐘）完成。
 - 當使用案例完成時，參與者會很快樂地離開（譯註：因為達成參與者的目標了）。
 - 完成許多海平面等級的目標之後，參與者（如果是一位雇員的話）會要

求加薪。

譯註：這一點剛看到會有點奇怪，其實就作業人員來說，他們完成許多工作就代表業務成績不錯，當然會要求加薪。

Periodically review to make sure you know where "sea level" is for your goals and how far below (or above) sea level the steps are. Recall the tests for a sea-level goal:

It is done by one person, in one place, at one time (2 to 20 minutes).

The actor can go away happily as soon as it is completed.

The actor (if an employee) can ask for a raise after doing many of these.

- ◆ 回想一下，大部分使用案例的主要成功情節都會有三到九個動作步驟，而且每個動作步驟的目標等級，基本來說都只比這個使用案例的目標等級低一點。如果你發現主要成功情節中有超過九個動作步驟的話，請找找看是否存在下面情況，並合併這個動作步驟：

- 在好幾個連續的動作步驟中，都是由相同的參與者持球。
- 在某些動作步驟中，我們會描述使用者的一些動作。這些都是典型的使用者介面動作，它們違反了寫作指引 5 寫出參與者的意圖，而非動作。
- 在某些動作步驟中，兩個參與者間有許多簡單的來回動作。試著問自己這些來回動作是否要達成某個更高一級的目標。

Recall that most use cases have three to nine steps in the main success scenario and that the goal level of a step is typically just below the goal level of the use case. If you have more than nine steps, look for steps to merge in the following places:

Where the same actor has the ball for several steps in a row.

Where the user's movements are described. These are typically user interface movements, violating Guideline 5, Show the Actor's Intent, Not the Movements, on page 92.

Where there is a lot of simple back-and-forth between two actors. Ask if they aren't really trying to accomplish something one level higher with all that back and forth.

- ◆ 問自己說：「為何使用者／參與者要做這個動作？」因此而得到的答案是更高一級的目標。你可能用這個目標合併好幾個動作步驟。圖 20.1 中告訴我們在不同目標等級中，動作步驟中的子目標是如何滿足使用案例的目標。

↑ 英 208

Ask, "Why is the user/actor doing this action?" The answer you get is the next higher goal level. You may be able to use this goal to merge several steps.

Figure 20.1 shows how the goals of the steps fit within the goals of the use cases at different levels.

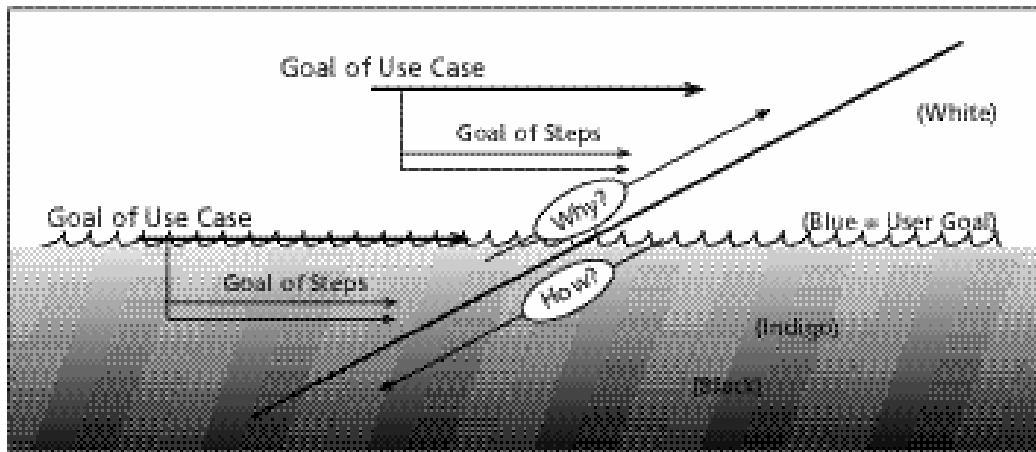


圖 20.1 詢問「為何」要提升目標等級的原因

寫作提示 7：不要寫出 GUI

寫完動作步驟之後，請證實一下裡面所捕捉到是參與者的真正意圖，而不是操作使用者介面的動作而已。毫無疑問地我們可以用使用案例記錄下使用者介面本身，不過如果我們是在寫功能需求的話，請記住不要寫出 GUI。

在需求文件中寫出使用者操作介面時的動作有下列三個缺點：

- ◆ 文件變得沒必要的長。
- ◆ 需求變得容易變動，也就是說在使用者介面設計方面的小改變都可能造成「需求」上的變動（但這不算是真正需求）。
- ◆ 這樣做不啻偷了 UI 設計師的工作，你應該相信他們可以把這個工作做好。

Verify that the step you just wrote captures the real intent of the actor, not just the movements in manipulating the user interface. This advice applies when you are writing functional requirements, since it is clear that you can write use cases to document the user interface itself.

In a requirements document, describing the user's movements in manipulating the interface has three drawbacks:

The document is needlessly longer.

The requirements become brittle, meaning that small changes in the user interface design cause a change in the "requirements" (which weren't requirements after all).

It steals the work of the UI designer, whom you should trust to do a good job.

在本書中，大部分使用案例都是寫得不錯的例子。使用案例 20 評估職業傷害的賠償就是其中一個好例子。

1. 理賠師會審查並評估醫學報告、...

2. 理賠師...以估計永久傷殘程度。
3. 理賠師會加總該理賠的永久傷殘程度、...。
4. 理賠師決定最終調解範圍。

Most of the use cases in this book should serve as good examples for you. This extract of Use Case 20, Evaluate Work Comp Claim, on page 72, is one.

1. Adjuster reviews and evaluates the reports, . . .
2. Adjuster rates the disability using . . .
3. Adjuster sums the disability owed . . .
4. Adjuster determines the final settlement range.

下面則是我們不該學習的錯誤範例：

1. 系統秀出請使用者輸入 ID 與密碼的登入畫面。
2. 顧客輸入 ID 與密碼，並按下 OK 按鈕。
3. 系統證實使用者的 ID 與密碼。
4. 系統秀出主畫面，裡面會有一些功能選單。
5. 顧客選擇一個功能，並按下「OK」按鈕。

我們很容易一不小心就寫出使用者介面動作，請多注意一下。

Here is as an example of what not to do:

1. The system displays the Login screen with fields for username and password.
2. The user enters username and password and clicks OK.
3. The system verifies name and password.
4. The system displays the Main screen, containing function choices.
5. The user selects a function and clicks OK.

It is very easy to slip into describing the user interface movements, so be on your guard.

寫作提示 8：使用案例的兩種結尾

每個使用案例都會以兩種方式結尾：成功或失敗。

心裡請記得一件事，當動作步驟呼叫某個子使用案例時，被呼叫的使用案例可能會以成功或失敗結尾。如果我們是在主要成功情節中呼叫的，那麼就要在某個擴充情節中處理這個失敗情形。如果我們是在擴充情節中呼叫的，那麼就要在同樣的擴充情節中同時描述成功與失敗處理方式（例如使用案例 22 *登錄損失情形*）。

↑ 英 209

事實上，不論目標成功或失敗，我們都具有兩種責任。第一個責任是我們必須確認：針對每個被呼叫到的使用案例，我們都有處理它的失敗情況。第二個責任是我們必須確認使用案例有滿足每個關係人的利益，特別是當目標失敗時，我們必須保障關係人的利益。

Every use case has two possible endings, success and failure.

Bear in mind that when an action step calls a sub use case, the called use case can succeed or fail. If the call is in the main success scenario, the failure is handled in an extension. If it is from an extension, describe both success and failure handling in the same extension (see, for example, Use Case 22, Register a Loss, on page 75). You actually have two responsibilities with respect to goal success and failure. The first is to make sure that you deal with the failure of every called use case. The second is to make sure that your use case satisfies the interests of every stakeholder, particularly if the goal fails.

寫作提示 9：需要對關係人做出事後保證

使用案例中不該只是記錄主要參與者跟系統間大家看得見的互動情形而已。如果真是這樣的話，我們就無法在使用案例裡面寫出令人滿意的行為需求，只是在記錄 UI 而已。

事實上，系統必須維護關係人間的合約協定。主要參與者也是關係人之一，它會保障自身利益；其他關係人則不會。使用案例中應該描述：由使用者驅動敘事情節的情況下，系統如何在不同環境下保障所有人的利益。也就是說，使用案例中應該寫出系統對關係人所做出的事後保證。

A use case does not record only the publicly visible interactions between the primary actor and the system. If it did only that, it would not make acceptable behavioral requirements, but only document the user interface.

The system enforces a contractual agreement between stakeholders. One of these is the primary actor; the others are not there to protect themselves. The use case describes how the system protects all of their interests under different circumstances, with the user driving the scenario. It describes the guarantees the system makes to them.

請花時間找出在每個使用案例中的關係人名稱與其利益。一般來說，我們應該可找到 2 到 5 種關係人：主要參與者、公司擁有者、可能會有的政府機構代表，以及其他可能會有的人。或許負責測試或維護的員工也會對使用案例操作情形有利益存在。

通常，大部分使用案例都會有相同的關係人，就算是不同使用案例，關係人的利益通常也是非常類似的。只要花一點功夫，我們很快就可以列出這些關係人的名稱與其利益。下列是典型的關係人利益：

Take the time to name the stakeholders and their interests in each use case. You should find two to five stakeholders: the primary actor, the owner of the company, perhaps a regulatory agency, and perhaps someone else. Maybe the testing or maintenance staff has an interest in the operation of the use case.

Usually, the stakeholders are the same for most use cases, and usually their interests

are very much the same across use cases. It soon takes little effort to list their names and interests. Typically, their interests are these:

- ◆ 使用案例名稱代表主要參與者的利益。它通常是取得某樣東西。
The primary actor's interest is stated in the use case name. It usually is to get something.
- ◆ 公司利益通常是確認主要參與者不是免費帶著某樣東西離開，或者說主要參與者對他所獲得的東西必須付費。
The company's interest is usually to ensure that the primary actor doesn't get away with something for free or that she pays for what she gets.
- ◆ 政府機構代表的利益通常是確認公司有遵從某些政府指引，而且通常要保有某種系統紀錄。
The regulatory agency's interest is usually to make sure that the company can demonstrate that it follows guidelines and usually that some sort of log is kept.
- ◆ 對於交易行為中所發生失敗情形的復原方式，通常會跟某一種關係人有利害關係存在。換言之，他需要更多系統紀錄才能恢復交易行為。
One of the stakeholders typically has an interest in recovering from a failure in the middle of the transaction, that is, more logging.

現在，你應該知道：主要成功情節與擴充情節都是針對關係人的利益所寫的。我們只需花費很少功夫就能達到這種程度。請試著從主要成功情節開始讀一下使用案例中的說明文字，看看關係人的利益是否有寫出來。你將會很驚訝發現到我們有多常漏掉一兩個關係人的利益。更常發生的情況是：寫使用案例的人沒有想到主要成功情節在執行過程中可能會發生某個失敗情形、應該留下系統紀錄或復原資訊。請檢查一下，看看使用案例中所有失敗情形的處理方式是否都有保障到所有關係人的全部利益。

通常，一個新的擴充情況代表主要成功情節中某個東西發生失敗情形。如果我們需要做太多的驗證動作的話，寫使用案例的人有時可以把整組檢查放在另外獨立的寫作區域中，也可以新增企業規則小節。

↑ 英 210

See that the main success scenario and its extensions address the stakeholders' interests. This takes very little effort. Read the text of the use case, starting with the main success scenario, and see whether those interests are present. You will be surprised by how often one is missing. Very often, the writer has not thought about the fact that a failure can occur in the middle of the main success scenario, leaving no log or recovery information. Check that all failure handling protects all interests of all stakeholders.

Often, a new extension condition reveals a missing validation in the main success scenario. On occasion, there are so many validations being performed that the writer moves the set of checks into a separate writing area, perhaps creating a business rules

section.

Roshi 的 Pete McBreen 寫信給我，提到他們團隊第一次試著針對某個已交給顧客的系統列出關係人利益的情形。結果在他們所列出來的關係人利益清單中，找到了顧客在第一年操作軟體後所產生的所有變動需求。他們當初成功建造並交付系統給顧客時，並沒有滿足特定關係人的某些特定需要。因此，關係人察覺到這些不足之處，對他們發出變動需求。這個結果震驚了他們的開發團隊，如果他們之前有先寫出關係人與其利益的話，就可避免掉（或減少部分）變動需求。因此，Pete 現在變成找出使用案例中關係人利益的強烈擁護者。實際上，做這項檢查工作時只需花費非常少時間，而且對所花費的時間來說，其價值是很顯著的。

Pete McBreen, of Roshi, wrote me about the first time his group listed the stakeholders' interests using a system they had already delivered. In that list they discovered all the change requests for the first year of their software's operation. They had successfully built and delivered the system without satisfying certain needs of certain stakeholders. The stakeholders figured this out, of course, and so the change requests came in. What excited this team was that, if they had written down the stakeholders and interests early on, they could have avoided (at least some of) those change requests. As a result, Pete is now a strong advocate of capturing the stakeholders' interests in use cases. Performing this check takes very little time and is very revealing for the time spent.

使用案例範本文件的事後保證小節中會記錄使用案例是如何滿足這些關係人利益的。如果我們是在人際溝通良好的開發團隊中工作，而且所進行的專案比較不用這麼吹毛求疵或儀式性比較低的話，就可省略事後保證不寫。不過，如果我們所進行的是具有關鍵緊急性的專案，或者發生傷害或誤解時的成本很高，那麼我們一定要多加注意事後保證小節。然而，不論哪一種情況，開發團隊至少都應該在心裡檢查一遍，看看使用案例的成功或失敗情形該如何結尾。

寫主要成功情節之前先寫出事後保證是很不錯的一種寫作策略。這樣一來，我們在寫主要成功情節時就可以馬上思考，做必要的驗證動作，而不用稍後再驗證它，修改使用案例中的說明文字。

請參閱 2.2 *有利益的關係人間的合約*與 6.2 *最小事後保證*這兩個小節，裡面對這個主題有更多的討論。

The guarantees section of the template documents how the use case satisfies these interests. You might skim on writing the guarantees on a less critical, low-ceremony project on which the team has good personal communications. You will pay more attention to them on more critical projects, where the potential for damage or the cost of misunderstanding is higher. However, in both cases, your team should at least go through the mental exercise of checking both the success and failure exits of the use case.

It is a good strategy to write the guarantees before writing the main success scenario,

because then you will think of the necessary validations on the first pass instead of discovering them later and having to go back and change the text.

Section 2.2, Contract between Stakeholders with Interests, and Section 6.2, Minimal Guarantees, have more details on these topics.

寫作提示 10：事先條件

我們會在使用案例的事先條件中宣告使用案例的有效運作條件。事先條件是系統確認執行前會成立的東西。我們之所以要記錄下事先條件是因為這樣一來使用案例中就不會再重新檢查它。

有兩種常見情況會變成事先條件。最常見的事先條件是：使用者已登入系統，他的身分也被證實過了。另一個經常會變成事先條件的情況是：第二個使用案例半路攔截第一個使用案例的執行活動，前者會認為第一個使用案例已設定好它可信賴的某些特定條件。例如假設使用者在第一個使用案例中已選好（部分）產品，這時候，第二個使用案例會在它的處理過程中用到已選好產品的相關知識。不論什麼時候，當我看見事先條件時，心裡知道一定還會有目標等級更高的使用案例存在，而且它會設定好我們現在所看到的這個事先條件。

The preconditions of the use case declare the valid operating conditions of the use case. The preconditions must be things that the system can ensure will be true. You document the preconditions because you will not check them again in the use case. There are two common situations that give you preconditions. The most common is that the user is logged on and validated. The other is when a second use case picks up the thread of activity partway through a first use case, expecting the first use case to have set up a particular condition that the second can rely upon. An example of this is that the user selects or partially selects a product in the first use case, and the second one uses knowledge of that selection in its processing.

Whenever I see a precondition, I know there is a higher-level use case in which the precondition was established.

寫作提示 11：針對每個使用案例所做的欄位合

格／不合格測驗

簡單的欄位合格／不合格測驗讓我們知道自己所寫的使用案例中某個部分是否正確。接下來的表 20.1 中就列出我所找到的一些測驗。這些測驗的所有結果都應該要是「合格的」。

Simple pass/fail tests let us know when we have filled in a part of the use case correctly. Table 20.1 on the next page lists the few I have found. All of them should produce a "yes" answer.

表 20.1 針對單一使用案例所做的欄位測驗：合格／不

合格

欄位	問題
使用案例標題	它是由主動動詞所構成、代表主要參與者目標的目標片語嗎？ 系統能達到這個目標嗎？
設計範圍與目標等級	有填寫這兩個欄位嗎？
設計範圍	使用案例有把設計範圍中所提到的系統視為黑箱嗎？（如果使用案例代表系統需求文件的話，答案就一定是「肯定的」。不過，如果使用案例是屬於白箱式企業使用案例的話，答案就可以是「否定的」。） 如果設計範圍中的系統就是我們所設計的系統，設計師所設計出來的東西有在設計範圍內、沒超出設計範圍嗎？
目標等級	使用案例中所描述的內容符合我們想要的目標等級嗎？ 目標真的是我們所想要的目標等級嗎？
主要參與者	主要參與者（他／她／它）會對 SuD 發生行為嗎？ 主要參與者（他／她／它）對 SuD 有目標，而且希望藉此獲得系統所承諾的服務嗎？
事先條件	這些事先條件具有強制性嗎？我們是否可在 SuD 中找到合適的地方設定這些條件？ 使用案例中是否永遠不需要測試這些條件有沒有成立？
關係人與其利益	關係人有名稱嗎？系統是否必須滿足他們所表達出來的利益呢？（關係人所表達的系統使用情形【usage】會隨著正式性與可容忍程度而異。）
最小事後保證	有保障關係人的所有利益嗎？
成功事後保證	有滿足關係人的所有利益嗎？
主要成功情節	是按照寫作步驟 3-9 所寫出來的嗎？ 它是由某個觸發事件所驅動，達到成功的事後保證

	嗎？
	它可以有其他不同、正確的執行順序嗎？
情節中的任何動作步驟	它是用次一層目標的片語所寫成的嗎？
	情節中所描述的動作流程，在成功完成某個動作步驟後，會直接執行下個動作步驟嗎？
	動作步驟中有很清楚表明是由那個參與者使用系統來達成目標嗎（他是「擊球者」）？
	參與者的意圖很清楚嗎？
	動作步驟的目標等級是否比整個使用案例的目標等級還低嗎？它是否如我們所願，只比使用案例的目標等級低一點點？
	你確定動作步驟中沒有描述到系統的使用者介面設計方式嗎？
	傳給動作步驟的資訊有很明確嗎？
	動作步驟是在「證實」某個條件「成立」，而不是在「檢查」這個條件嗎？
擴充情況	系統可偵測並處理這些擴充情況，而且這些條件是必要的嗎？
	這是系統實際上有需要的擴充情況嗎？
技術與資料變異情形清單	對主要成功情節來說，你確定這些不是常態行為的擴充情節嗎？
使用案例的整體內容	對贊助者與使用者來說，「這是你要的嗎？」
	對贊助者與使用者來說，「你能夠據此判別所交付的系統是自己想要的系統嗎？」
	對開發人員來說，「你能實作這個使用案例嗎？」

第21章跟整組使用案例有關的寫作提示

寫作提示 12：可不斷展開的敘事情節

對一個開發專案來說，整組使用案例中會有一個最上層的使用案例存在，我們會把它稱為像 ZZZ 系統這樣的名稱。這個使用案例除了當做參與者與其最高層目標的目錄之外，還有其他用途：我們可以把它拿來當作任何人第一次看這個系統時的起點。因為裡面不會寫出太多關於敘事情節的東西，所以它是可以選擇性寫出來的，不過大部分的人在閱讀一整組使用案例時，還是希望可以從某個地方開始讀它們。

For a development project, there is one use case at the top of the stack, called something like Use the ZZZ System. This use case is little more than a table of contents for the actors and their highest-level goals. It serves as a starting point for anyone looking at the system for the first time. It is optional, since it doesn't have much of a story line, but most people like to see just one starting place for their reading.

這個最上層的使用案例會去呼叫一些最外層使用案例 (outmost use case)，後者會秀出系統中最外層主要參與者的摘要目標。對公司的資訊系統來說，公司外部顧客、行銷部門或 IT (或安全) 部門都是典型的最外層主要參與者。在這些最外層使用案例裡面，我們會寫出系統的海平面等級使用案例間的交互關係。大部分讀者應該都會從其中挑一個使用案例作為閱讀「敘事情節」的起點。

我們可以把這些最外層使用案例展開變成使用者目標等級或海平面等級的使用案例。在使用者目標等級使用案例中，設計範圍等於要設計出來的系統。至於使用者目標等級使用案例中的動作步驟，則會秀出參與者跟系統間的互動情形。這些互動情形會達成使用者的「立即」目標。

This topmost use case calls out the outermost use cases, which show the summary goals of the outermost primary actors of the system. For a corporate information system, there is typically an external customer, the marketing department or the IT or security department. These use cases show the interrelationships of the sea-level use cases that define the system. For most readers, the "story" starts with one of them. The outermost use cases unfold into user-goal or sea-level use cases. In the user-goal use cases, the design scope is the system being designed. The steps show the actors and the system interacting to deliver the user's immediate goal.

如果子使用案例很複雜或者它會被用在幾個不同地方的話，我們就會把海平面等級使用案例中的動作步驟展開變成海平面下（靛藍色或子功能）的使用案例。不過，由於子功能目標等級使用案例維護起來很昂貴，所以我們只有在必要時才會寫出子功能目標等級使用案例。一般來說，我們可能會產生像*找尋顧客*、*找尋產品*等這樣子的子功能目標等級使用案例。除此之外，我們只有在非常偶然的情況下，才會再度把靛藍色使用案例展開變成深靛藍色的使用案例。

「把整組使用案例視為可不斷展開的敘事情節」這種做法的價值是：這樣一來，不論是把使用案例中某個複雜的一個小節移開獨立成一個使用案例，或者把一個簡單的子使用案例合併回去呼叫它的使用案例，都只是「很小」的動作而已。原則上，每個動作步驟都可以馬上展開自己獨立變成一個使用案例。請參見 10.1 子使用案例。

↑ 英 215

A step in a sea-level use case unfolds into an underwater (indigo or subfunction) use case if the sub use case is complicated or is used in several places. Subfunction use cases are expensive to maintain, so only use them when you have to. Usually, you will have to create subfunction use cases for Find a Customer, Find a Product, and so on. On occasion, a step in one indigo use case unfolds to a deeper indigo use case. The value of viewing the use case set as an ever-unfolding story is that it becomes a "minor" operation to move a complicated section of writing into its own use case or to fold a simple sub use case back into its calling use case. Each action step can, in principle, be unfolded to become a use case in its own right. See Section 10.1, Sub Use Cases.

寫作提示 13：公司設計範圍與系統設計範圍兩者

我們可能會對設計範圍感到混淆。關於「系統確切邊界到底在哪裡」這一點，大家對它會有不同的概念。不過，如果你是在寫企業使用案例或系統使用案例，那麼設計範圍就會變得特別清楚。

企業使用案例的設計範圍是企業業務，裡面所描述的是企業組織外的參與者如何達成跟企業組織有關的目標。企業使用案例中通常不會提到技術，因為它跟企業業務如何運作有關。

另一方面，系統使用案例的設計範圍則是我們要設計的電腦系統，裡面所描述的是參與者達成跟電腦系統有關的目標；它會跟技術有關。

Design scope can cause confusion. People have different ideas of where the exact boundaries of the system are. In particular, be very clear whether you are writing a

business use case or a system use case.

A business use case is one in which the design scope is business operations. It is about an actor outside the organization achieving a goal with respect to the organization.

The business use case often contains no mention of technology, since it is concerned with how the business operates.

A system use case is one in which the design scope is the computer system to be designed. It is about an actor achieving a goal with the computer system; it is about technology.

企業使用案例通常會寫成白箱式的使用案例，以描述人們跟公司中某些部門間的互動情形；另一方面，系統使用案例大部分時候會被寫成黑箱式的使用案例。這兩種寫法都蠻恰當的，因為大部分企業使用案例的目的都是為了描述公司現在或未來的設計方式（譯註：公司流程的設計方式），而系統使用案例的目的則是針對新的設計方式（譯註：公司流程再造後的設計方式）而寫出來的需求。白箱式企業使用案例描述企業內部；黑箱式系統使用案例則描述電腦系統外部。

譯註：譯者很早之前就開始採用白箱式企業使用案例跟黑箱式系統使用案例來捕捉企業內部流程與電腦系統的行爲需求。只不過，在還沒看過設計範圍這個概念之前，自己是知其然而不知所以然，無法告訴別人自己為何要這麼做。尤其是遇到 intranet 系統轉型成 extranet（或 internet）系統的情況，我們很不容易清楚企業使用案例到底是在描述企業內部流程還是外部流程。有了設計範圍之後，我們就可以很清楚地用黑箱式企業使用案例說明企業外部流程，而用白箱式企業使用案例說明企業內部流程。在此衷心感謝本書作者！

The business use case is often written in white-box form, describing the interactions between the people and departments in the company, while the system use case is almost always written in black-box form. This is usually appropriate because the purpose of most business use cases is to describe the present or future design of the company while that of the system use case is to create requirements for a new design. The business use case describes the inside of the business; the system use case describes the outside of the computer system.

如果你正在設計一個電腦系統，那麼你應該同時會有企業使用案例與系統使用案例兩者。企業使用案例用來描述系統功能情境，說明系統所處的企業環境為何。為了避免混淆，請務必標示出使用案例的設計範圍來。請試著想出一種圖示，以標示出它是企業使用案例或系統使用案例（請參閱 3.2 設計範圍中的用圖形化的圖示來強調設計範圍小節）。此外，也請考慮在使用案例中放入一張圖，說明系統中所包含的子系統為何（請參見 3.2 設計範圍中的聯合兩個系統的輸入與更新服務申請小節）。

If you are designing a computer system, you should have both business and system use cases in your collection. The business use case describes the context of the system's function, the place of the system in the business.

To reduce confusion, always label the scope of the use case. Consider creating a graphic icon to illustrate whether it is a business or system use case (see the subsection Using Graphical Icons to Highlight the Design Scope on page 40). Consider placing a picture of the system inside its containing system within the use case itself (see Use Case 8, Enter and Update Requests (Joint System), on page 43).

寫作提示 14：使用案例格式的核心價值與替代

性價值

雖然大家不斷發明新的使用案例格式，不過有經驗的使用案例寫作人員卻對使用案例格式的核心價值慢慢形成共識。在 1999 年、美國 TOOLS 會議上的兩篇論文（Firesmith, 1999 與 Lilly, 1999）中有提到使用案例寫作過程中最常見的十幾個錯誤。這些文章中所提到的錯誤與改善之道正好呼應到本節所描述的「使用案例格式的核心價值」。

譯註：在這個小節中，作者所用的「價值」其實是可以「做法」換掉它。所以「核心價值」相當於「主要做法」，「替代性價值」則相當於「替代性做法」。

Although people keep inventing new use case formats, experienced writers are coming to a consensus on core format values. Two papers in the 1999 TOOLS USA conference (Firesmith, 1999, and Lilly, 1999) described a top dozen or so mistakes made in use case writing. The mistakes and fixes described in those articles echo the core values.

核心價值

以目標為基礎寫出使用案例。

使用案例是以(1).主要參與者的目標與(2).其他參與者（例如 SuD）的子目標（為了滿足主要參與者的目標）這兩種目標為核心的。在使用案例中，每個句子裡面所描述的東西都是為了達成某個子目標而做的。

↑ 英 216

Use cases are centered around the goals of the primary actors and the subgoals of the various actors, including the SuD, in achieving that goal. Each sentence describes a subgoal being achieved.

從鳥瞰觀點寫出使用案例。

當我們在描述使用案例時，請從上空、鳥瞰觀點寫出這些動作，或者假裝自己正在用參與者名稱轉播一場球賽，而不是「從系統內部往外看這個世界」。

The use case is written as describing the actions as seen by a bird viewing the scene

from above, or as a play naming the actors. It is not written from the "inside looking out."

讓使用案例具有可讀性。

使用案例或功能規格書最後都是要讓人看的。如果不容易懂它的話，就失去了核心目的。我們可以犧牲掉某些精確度、甚至是正確性以換取可讀性，然後再用更多的溝通來補償它。可是，一旦我們犧牲掉可讀性之後，委託我們開發系統的人就不願意讀我們所寫的使用案例了。

Ultimately a use case, or any specification, will be read by people. If it is not easily understood, it does not serve its core purpose. You can increase readability by sacrificing some amount of precision and even accuracy, making up for the lack with increased conversation. Once you sacrifice readability, however, your constituents won't read your use cases.

讓使用案例具有不同用途。

使用案例是用來描述行為的一種形式。在專案中的不同時間點，使用案例可以有不同用途。例如使用案例可以用來：

- ◆ 提供黑箱式功能需求。
- ◆ 提供企業組織進行企業流程再造時的需求。
- ◆ 記錄企業組織的企業流程（白箱式的企業使用案例）。
- ◆ 協助我們從使用者或關係人身上引導出需求（當開發團隊用其他文件形式作為最後需求文件時，我們後來會把使用案例丟棄掉）。
- ◆ 詳述要被建置與執行的測試案例。
- ◆ 記錄系統的內部情形（白箱式的系統使用案例）。
- ◆ 記錄某個設計或設計框架中的行為。

Use cases are a form of behavioral description that can serve various purposes at various times in a project. For example, they have been used to

Provide black-box functional requirements.

Provide requirements for an organization's business process redesign.

Document an organization's business process (white box).

Help elicit requirements from users or stakeholders (to be discarded when the team writes the final requirements in some other form).

Specify the test cases that are to be constructed and run.

Document the internals of a system (white box).

Document the behavior of a design or design framework.

寫出黑箱式需求。

當我們把使用案例當作功能規格時，總是會把 SuD 視為黑箱。有一些專案開發團隊曾試著要寫出白箱式需求（裡面會猜測系統內部看起來像什麼東西），他們告訴我們說這樣的使用案例是很難讀、不太容易讓人接受的，而且有點容易改變（它會隨著設計工作進行下去而改變）。

When used as a functional specification, the SuD is always treated as a black box. Project teams who have tried writing white-box requirements (guessing what the inside of the system will look like) report that the resulting use cases are hard to read, not well received, and brittle, changing as the design proceeds.

把替代路徑放在主要成功情節後面。

把替代流程放在主要成功情節後面的最原始概念就是：讓使用案例變得很易讀。如果我們把分支情況寫在使用案例說明文字的主要成功情節中，只會把敘事情節變得更難讀而已。

The original idea of putting alternative courses after the main success scenario keeps showing up as the way to create the use cases that are the easiest to read. Putting the branching cases inside the main body of the text seems to make the story too hard to read.

不要花太多無謂精力在使用案例上。

持續漫無目的去修改使用案例並不能增加它的價值。我們所寫出來的第一版使用案例草稿很可能就能提供我們使用案例所具有價值的一半了。雖然多寫出擴充情節還能增加一些價值，不過如果我們接下來還繼續去修改句子中用字的話，其實是不能增加多少溝通效果的。這時候，我們的精力應該要轉到其他事物上，例如跟外部間的介面、企業規則等，這些東西都屬於需求文件中的其他部分。當然，持續修改使用案例所得到的報酬，其衰減率會隨著專案的關鍵緊急程度而有所不同。

↑ 英 217

Continued fiddling with the use cases does not increase their value. The first draft of the use case creates perhaps half its value. Adding to the extensions adds value, but after a short while, changing the wording of the sentences no longer improves communication. At that point, your energy should go into other things, such as the external interfaces, the business rules, and so on, which are all part of the rest of the requirements. Of course, the rate of diminishing returns varies with the criticality of the project.

恰當的替代性價值

除了核心價值之外，我們同時也發現到一些可讓人接受的替代性價值。

編號過的動作步驟 v.s. 簡單段落。

某些人會把動作步驟編號，這樣一來，他們就可以在擴充情節中參照到這些動作步驟。其他人則只會寫出一些簡單段落，同時也會以簡單段落形式寫出替代情節。這兩種解決方案都是可行的。

Some people number steps so they can refer to them in the extensions section. Others write in simple paragraphs and put the alternatives in similar paragraph form. Both

approaches seem to work well.

非正式格式 v.s. 正式格式。

有時候，花很多精神去處理功能需求是恰當的做法；其他時候，甚至是在同一個專案中，則可能變成浪費精力的事（請參見 1.2 *你用的使用案例不等於我的與 1.5 管理自己的精力*）。這是真的！面對各種不同情境，我甚至不建議你只採用一種寫作範本，而且至少讓你知道有兩種版本存在：非正式格式與正式格式。不過，不同的使用案例寫作人員有時候可能會比較偏好使用某一種格式。每一種格式都會有它自己的工作方式。請比較使用案例 25 *實際的登入動作（非正式格式版本）* 跟使用案例 5 *買東西（正式格式版本）*。

There are times when it is appropriate to allocate a lot of energy to detailing the functional requirements; at other times, even on the same project, that is a waste of energy. (See Section 1.2, *Your Use Case Is Not My Use Case*, and Section 1.5, *Manage Your Energy*.) This is true to such an extent that I don't even recommend just one template any more, but always show both the casual and fully dressed versions. Different writers sometimes prefer one over the other. Each works in its own way. Compare Use Case 25, *Actually Login (Casual Version)*, on page 120, with Use Case 5, *Buy Something (Fully Dressed Version)*, on page 9.

在先期建立企業模型時，我們該用或不用使用案例呢？

有些開發團隊喜歡在寫出系統的功能需求之前，先記錄或修正企業流程。在這些團隊裡面，有些人會選擇用使用案例來描述企業流程，其他人則可能會選擇用其他形式來建立企業流程模型。如果你是選擇採用使用案例的話，那麼從系統的功能需求來看，建立企業流程模型時所用的表示法跟建立系統模型時所用的並不會有什麼不同。

Some teams like to document or revise the business process before writing the functional requirements for a system. Of those, some choose use cases to describe the business process and some choose another business process modeling form. From the perspective of the system's functional requirements, it does not seem to make much difference which business process modeling notation is chosen.

使用案例圖 v.s. 參與者 — 目標清單。

某些人會用參與者 — 目標清單秀出將來要開發的整組使用案例；其他人則比較喜歡用使用案例圖秀出主要參與者跟他們的使用者目標等級使用案例，它的用途其實跟參與者 — 目標清單是一樣的。

Some people use actor-goal lists to show the set of use cases being developed; some prefer use case diagrams. The use case diagram, showing the primary actors and their user-goal use cases, can serve the same purpose as that of the actor-goal list.

白箱式使用案例 v.s. 合作圖。

白箱式使用案例跟 UML 中的合作圖其用途幾乎是相等的。你可以把使用案例視為以文字寫成的合作圖。其差別是：合作圖中不會描述到元件的內部動作，而使

用案例則可能會。

譯註 1：首先要說明的是：循序圖跟合作圖是等價的。所以這裡所講的東西同時也適用於循序圖。還有，這裡所說的白箱式使用案例應該是白箱式系統使用案例，因為它會秀出系統中的元件。

譯註 2：在合作圖中，除了*自身訊息*【self message】之外，我們通常只會秀出元件間的訊息，而用註解去說明一些元件內的動作；另一方面，使用案例則可視為可不斷展開的目標，所以我們是有可能在子功能目標等級使用案例中秀出元件的內部動作。

There is near equivalence between white-box use cases and UML collaboration diagrams. You can think of use cases as textual collaboration diagrams. The difference is that a collaboration diagram does not describe the components' internal actions, which the use case might.

不恰當的替代性價值

在主要成功情節中寫出「條件」子句。

如果使用案例中只有一個分支行為的話，把分支動作寫在主要說明文字中是比較簡單的做法。然而，使用案例中通常會有許多分支路徑存在，大家會迷失在這些分支路徑當中，不知道敘事情節下一步該走哪個動作步驟。某些曾經在主要成功情節中寫出*條件*子句的人表示他們很快就決定要改成「在主要成功情節之後再寫出擴充情節」的做法了。

↑ 英 218

If there were only one branching of behavior in a use case, it would be simpler to put that branching within the main text. However, use cases have many branches, and people lose the thread of the story. Individuals who have used if statements report that they soon change to the form of the main success scenario followed by extensions.

用循序圖代替使用案例的說明文字。

某些有支援循序圖的軟體開發工具宣稱他們支援使用案例。雖然循序圖也能秀出參與者間的互動情形，不過：

- ◆ 它無法捕捉到內部動作（我們需要這些內部動作，以秀出系統是如何保障關係人利益的）。
- ◆ 它比使用案例難讀多了（它是一種特殊化的表示法，而且用它來描述互動情形會佔據許多空間）。
- ◆ 參與者間的箭頭根本不夠我們放入需要寫出來的說明文字。
- ◆ 大部分工具都強迫寫使用案例的人隱藏這些說明文字，結果我們需要用彈出式對話框把說明文字秀出來，這樣一來，我們就很難沿著敘事情節一路走下去。
- ◆ 大部分工具都強迫寫使用案例的人每次只能寫出一條替代路徑，而且每次都

要從使用案例的開頭寫。這種重複花費的功夫既無聊又容易出錯，而且也不容易讀它，因為讀使用案例的人必須自新找出每個不同路徑中的差異行為究竟為何。

Some software development tools claim to support use cases because they supply sequence diagrams. Sequence diagrams also show interactions between actors, but

They do not capture internal actions (needed to show how the system protects the interests of the stakeholders).

They are much harder to read (they are a specialized notation, and they take up a lot more space).

It is nearly impossible to fit the needed amount of text on the arrows between actors.

Most tools force the writer to hide the text behind a pop-up dialog box, making the story line hard to follow.

Most tools force the writer to write each alternate path independently, starting over from the beginning of the use case each time. This duplication of effort is tiring, error prone, and hard on the reader, who has to detect what difference of behavior is presented in each variation.

循序圖不是用來表達使用案例的一種良好形式。有些人爲了得到工具的一些自動化服務好處才堅持這麼做：例如交互參照功能、會「往前、後退」連接的超連結，以及修改某個名稱時，相同名稱都會一起被修改到。這些服務雖然看起來不錯（而且我們目前可取得的文字型工具也缺乏這些功能），不過大部分的使用案例寫作人員跟讀者都同意：他們不會爲了這些服務而犧牲文字易讀、易寫的優點。

Sequence diagrams are not a good form for expressing use cases. People who insist on them do so to get the tool benefit of automated services: cross-referencing, back-and-forth hyperlinks, and the ability to change names globally. While these services are nice (and lacking in the textual tools currently available), most writers and readers agree that they are not sufficient reward for the sacrificed ease of writing and reading.

在功能規格書中描述 GUI。

寫需求時不讓使用者介面跟系統所需功能混在一起是需要一點寫作藝術的。這種寫作藝術很難學，不過它值得我們花功夫去學它。大家一致的共識是：強烈反對在使用案例中寫出使用者介面。請參見 19.6 進階範例：描述過多的使用者介面細節與 *Software for Use* (Constantine & Lockwood, 1999) 一書。

There is a small art to writing the requirements so that the user interface is not specified along with the needed function. This art is not hard to learn and is worth learning. The consensus is strong against describing the user interface in the use cases. See Section 19.6, Advanced Example of Too Much UI, and *Designing Software for Use* (Constantine and Lockwood, 1999).

寫作提示 15：跟整組使用案例有關的品質問題

針對整組使用案例，我只有三個跟品質方面有關的問題：

- ◆ 從最高等級到最低等級的目標，整組使用案例可形成具有一致性的敘事情節嗎？
- ◆ 針對每個主要參與者，會有一個位於設計範圍最外層、用來設定使用情境、目標等級最高的使用案例嗎？
- ◆ （對贊助者跟使用者來說）「這組使用案例代表所有需要開發的東西嗎？」

↑ 英 219

I have only three quality questions for the full use case set:

Do the use cases form a story that unfolds from the highest- to the lowest-level goal?

Is there a context-setting, highest-level use case at the outermost design scope possible for each primary actor?

(To the sponsors and users) "Is this everything that needs to be developed?"

第22章進行跟使用案例相關工作時的 寫作提示

寫作提示 16：使用案例只是需求文件中的第三章 (第四章在哪裡呢?)

使用案例只是整個需求蒐集工作中的一小部分，或者它只是需求文件中的「第三章」而已（譯註：在 1.3 *需求與使用案例*中，作者有秀出一個需求文件大綱，其中的第三章就是使用案例）。不過，它是整個需求蒐集工作的核心部分，我們拿它作為連接資料定義、企業規則、使用者介面設計、企業領域模型等的核心或軸心。然而，它並不能代表所有需求，只能代表行為需求。

請恕我不斷重述這件事：有些開發團隊會把使用案例加上光環，想把所有需求一股腦放進使用案例中。

Use cases are only a small part of the total requirements-gathering effort, perhaps "Chapter 3" of the requirements document. They are a central part of that effort, acting as a core or hub that links data definitions, business rules, user interface design, the business domain model, and so on. However, they are not all of the requirements, only the behavioral ones.

This has to be stated over and over, because such an aura has grown around use cases that some teams try to fit into them every piece of the requirements somehow.

寫作提示 17：寫使用案例時先求廣度

寫使用案例時，請先求廣度而非深度、先以精確度比較低再以比較高的方式寫出使用案例。逐步增加使用案例精確度（請參見圖 22.1）的做法可幫助我們管理自己的精力（請參見 1.5 *管理自己的精力*）。請依照下面的先後順序寫出使用案例：

Work breadth first, not depth, from lower to higher precision. Work expands with precision (see Figure 22.1), so this will help you manage your energy. (See Section 1.5, Manage Your Energy.) Work in this order:

1. 主要參與者。

第一步，如果所需花費時間很短的話，請就近蒐集整個系統中的所有主要參與者。大部分系統都很大，以至於我們很快就會墜入系統的五里霧中，所以把跟整

個系統相關的東西濃縮在某個地方是很不錯的，一下下也好。請以腦力激盪方式找出這些參與者，這將有助於我們第一次就找到大部分目標。

Primary actors. Collect all the primary actors as the first step in getting your arms around the entire system, if only briefly. Most systems are so large that you will soon lose track of everything, so it is nice to have the whole system in one place even for a short time. Brainstorm these actors to help you get the most goals on the first round.

2. 目標。

請注意，所有主要參與者的全部目標是我們可以用單一觀點看到整個系統的最後機會。因此，請花夠多的時間與精力來取得我們能得到的最完整、正確清單。後面的那幾個步驟都是讓更多人參與、花更多功夫在這份清單上而已。請使用者、贊助者與開發人員審查這份清單，讓大家同意配置系統的開發優先順序，並了解它。

↑ 英 221

Goals. Listing all the goals of all the primary actors is perhaps the last chance you will have to capture the entire system in one view. Spend enough time and energy to get this list as complete and correct as you can. The next steps will involve more people and much more work. Review the list with users, sponsors, and developers so that they can all agree on the priority and understand the system being deployed.

參與者	目標	成功動作	失敗情況	復原動作
				復原動作
				復原動作
		失敗情況	復原動作	
			復原動作	
			復原動作	
	目標	成功動作	失敗情況	復原動作
				復原動作
				復原動作
		失敗情況	復原動作	
			復原動作	
			復原動作	
	成功動作	失敗情況	復原動作	
			復原動作	
復原動作				
成功動作	失敗情況	復原動作		

				復原動作
				復原動作
			失敗情況	復原動作
				復原動作
				復原動作

圖 22.1 寫使用案例時逐漸增加精確度。

3. 主要成功情節。

典型的主要成功情節是很短且非常顯而易見的。它可以告訴我們系統要交付出去的敘事情節究竟是什麼。在研究各種失敗情況之前，請先確認使用案例寫作人員有寫出系統在正常情況下是如何工作的。

Main success scenario. The main success scenario is typically short and fairly obvious. It tells the story of what the system delivers. Make sure that the writers show how the system works once, before investigating all the ways it can fail.

4. 失敗條件／擴充情況。

在還未擔心如何處理失敗情況之前，請先找出所有擴充情況。這是一種腦力激盪活動，它跟研究或寫出處理擴充情況的動作步驟是很不一樣的。因而產生的清單可作為使用案例寫作人員的工作清單，這樣一來，使用案例寫作人員就可以有時候努力寫作、有時候休息一下，也不會忘記之前工作到什麼地方了。如果有人每寫出一個擴充情況就想要把它修到完整為止，那麼這種人就是典型的失敗情況清單寫不完的人。等到他寫完幾個失敗情節之後，很快就沒有力氣了。

Failure/extension conditions. Capture all the extension conditions before worrying about how to handle them. This is a brainstorming activity, which is quite different from researching and writing the extension-handling steps. The generated list serves as a worksheet for the writers, who can then write in spurts, taking breaks without worrying about losing their place. People who try to fix each condition as they name them typically never complete the failure list. They run out of energy after writing a few failure scenarios.

5. 復原用的動作步驟。

雖然我們是在使用案例寫作的蠻後面部分才開始寫出復原用的動作步驟，不過讓人訝異的是：它們經常會讓我們找到一些新的使用者目標、參與者與失敗情況。寫出這些復原用的動作步驟是整個使用案例寫作活動中最困難的部分，因為它迫使寫作人員面對企業政策，這是之前通常未被提到的地方。當我在寫復原用的動作步驟時，如果發現之前沒看見的企業政策、新參與者或新使用案例，這時候我並不會覺得之前所做的一些努力都是白費的。

Recovery steps. Although built last of all the use case steps, the recovery steps

uncover new user goals, new actors, and new failure conditions surprisingly often. Writing them is the hardest part of the use case writing activity because it forces the writer to confront business policy matters that often go unmentioned. When I discover an obscure business policy, a new actor, or a new use case while writing recovery steps, I feel vindicated for all my effort.

6. 資料欄位。

正式來說，這部分不屬於使用案例寫作活動中的一部分，不過我們通常會指派同樣的人把這些資料名稱（例如「顧客資訊」）展開變成資料欄位清單（請參閱 16.1 資料需求的精確度）。

Data fields. While formally outside the use case writing effort, often the same people have the assignment of expanding data names (such as "customer information") into lists of data fields (see Section 16.1, Precision in Data Requirements).

7. 資料欄位細節與資料欄位檢查。

當使用案例寫作人員正在審查使用案例時，某些情況下會請其他的人負責寫出資料欄位細節與資料欄位檢查。換言之，我們通常會請 IT 技術人員寫出資料欄位細節，而 IT 企業分析師，甚至是使用者則負責寫出使用案例。它是等級最低（譯註：最精細）的資料格式呈現方式。重述一次，一開始寫使用案例時，把這些細節與檢查排除在使用案例之外是恰當的，不過最終還是會把它們寫出來。

Data field details and checks. In some cases, different people write the data field details and checks while the use case writers are reviewing the use cases. Often the IT technical people will write the field details, while IT business analysts or even users write the use cases. This represents the data formats at the final level of precision. Again, these details and checks are outside the use cases proper but have to be written eventually.

寫作提示 18：使用案例的 12 步寫作訣竅

1. 找出系統邊界（情境圖、專案範圍內／外清單）。
Find the boundaries of the system (context diagram, in/out list).
2. 用腦力激盪法列出主要參與者（參與者簡述表）。
Brainstorm and list the primary actors (actor profile table).
3. 用腦力激盪法窮舉系統中主要參與者的使用者目標（參與者 — 目標清單）。
Brainstorm and list the primary actors' goals against the system (actor-goal list).
4. 找出最外層目標摘要等級的使用案例，裡面會涵蓋上述的所有東西。
Write the outermost summary-level use cases covering all of the above.
5. 重新討論並修正目標摘要等級的使用案例。新增、刪除或合併一些目標。
Reconsider and revise the strategic use cases. Add, subtract, and merge goals.
6. 選出一個使用案例，展開它或寫出其敘事情節，以熟悉使用案例的寫作材

料。

- Pick a use case to expand or write a narrative to get acquainted with the material.
7. 補上使用案例的關係人與其利益、事先條件與事後保證，並重複檢查它們。
Fill in the stakeholders, interests, preconditions, and guarantees. Double-check them.
 8. 寫出主要成功情節；重新檢查關係人的利益與事後保證。
Write the main success scenario; check it against the interests and the guarantees.
 9. 用腦力激盪法窮舉可能的失敗條件與替代路徑成功條件。
Brainstorm and list possible failure conditions and alternate success conditions.
 10. 寫出每個擴充情節中參與者跟系統該如何動作。
Write how the actors and the system should behave in each extension.
 11. 把需要自己獨立出來的東西分開變成子使用案例。
Break out any sub use case that needs its own space.
 12. 從最高層的使用案例開始重新調整整組使用案例。有必要時新增、粹取或合併使用案例。請重新檢查使用案例的完整性、可讀性與失敗條件。
Start from the top and readjust the use cases. Add, subtract, and merge as appropriate. Double-check for completeness, readability, and failure conditions.

寫作提示 19：了解犯錯成本

以比較低的品質寫出使用案例所需付出的成本要看你的系統與專案而定。某些專案只需要近似於無品質的程度寫出需求文件就好，因為使用者跟開發人員間有很好的溝通存在。

Chrysler 公司的綜合津貼專案的開發團隊，在它建構用來支付 Chrysler 公司所有薪資的軟體時採用「終極開發流程（XP）」方法論（Beck, 1999），他們就從未寫出比使用案例簡介更多的東西。他們寫得很少，而且不會寫出使用案例，只會把稱為「使用者故事（user story）」的東西寫在索引卡片上。使用者故事只是用來當作需求專家跟開發人員間對話的承諾而已。更值得注意的是，14 位開發團隊成員在兩個緊連的房間中一起工作，有很棒的團隊內部溝通存在。

↑ 英 223

The cost of lowered quality in a use case depends on your system and project. Some projects need next to no quality in the writing of the requirements document because they have such good communication between users and developers:

The Chrysler Comprehensive Compensation project team, in its building of software to pay all of Chrysler's payroll using the "eXtreme Programming"

methodology (Beck 1999), never went further than use case briefs. They wrote so little that they called them "stories" rather than use cases and put them on index cards. Each was really a promise for a conversation between a requirements expert and a developer. Significantly, the 14 team members sat in two adjacent rooms, and had excellent in-team communications.

當你的系統使用情形專家跟開發人員間的內部溝通越好，那麼把使用案例範本的某些部分拿掉的成本就越低。因為大家只要很簡單地互相講話就可以了，事情可以變得很直接。

如果你的工作環境採用分散式開發、需要跟好幾個負責承包的開發團隊一起工作、開發團隊很大，或者你所開發的是跟生命息息相關的系統，那麼需求文件品質不足所帶來的成本就比較高。如果正確寫下系統功能性是很重要的事，那麼你就必須注意關係人與其利益、事先條件，以及最小事後保證。

請研究出自己的專案究竟需要位於品質高低的哪個位置。一方面，不要對小型、非正式專案中的一個小錯誤反應過度；另一方面，如果對需求誤解的後果是很嚴重的話，就請嚴格寫出需求來。

The better the internal communications between your usage experts and developers, the lower the cost of omitting parts of the use case template. People will simply talk to each other and straighten matters out.

If you are working with a distributed development or multi-contractor development group, or one that is very large, or if you are working on life-critical systems, the cost of quality failure is higher. If it is crucial to get the system's functionality correctly written down, you need to pay close attention to the stakeholders and interests, the preconditions, and the minimal guarantees.

Recognize where your project sits along this spectrum. Don't get too worked up over relatively small mistakes on a small, casual project, but be rigorous if the consequences of misunderstanding are great.

寫作提示 20：偏好藍色牛仔褲（譯註：簡單方便的意）

聽起來雖然會有點奇怪，不過如果要比較寫太少跟寫太多需求兩者的話，一般來說，前者所造成的傷害是比較小的。當我們不曉得該寫多或少時，請用等級比較高的目標、精確度比較低、簡單的敘事格式，寫出比較少的說明文字。如果我們有短而易讀的文件，大家可能會因為不安而去讀它、提出問題。藉由發問，我們就可以發現什麼資訊被漏掉了。

另一種相反策略則行不通：如果我們巨細靡遺地寫出幾百頁、目標等級非常低的

使用案例，那麼可能只有很少、甚至沒有人會因為不安而去讀它，結果我們只是讓整個開發團隊停止溝通，而不是讓大家開口。程式設計師常犯的錯誤是寫出目標等級太低的東西，我們常看到這種錯誤。

Odd though it may sound, you will typically do less damage if you write too little rather than too much. When in doubt, write less text, using higher-level goals, with less precision, and in plain story form. Then you will have a short, readable document, which means that people will bother to read it and then ask questions. From those questions, you can discover what information is missing.

The opposite strategy fails: If you write a hundred or so low-level use cases in great detail, few or no people will bother to read them, and you will shut down communications on the team instead of opening them up. It is a common fault of programmers to write at too low a goal level, so this mistake happens quite often.

◆ 一個很短卻真實的故事

在一個有 50 位開發人員、金額達 1 仟五百萬美金的成功專案中，我們只用簡單段落形式寫出主要成功情節與一些失敗情形。因為我們有很棒的溝通，所以這種做法才是可行的。每位需求寫作人員都會跟 2 到 3 位設計師或程式設計師一起搭配工作。他們都彼此坐在附近或每天互訪好幾次。

事實上，加強開發團隊內的溝通品質對所有專案來說都是有幫助的。上述故事中所採用的團隊合作策略，請詳見 *Surviving Object-Oriented Projects* (Cockburn, 1998) 一書中的全方位團隊樣式 (Holistic Diversity pattern)。

↑ 英 224

Actually, enhancing the quality of in-team communications helps every project. The teaming strategy described in the story is the Holistic Diversity pattern from *Surviving Object-Oriented Projects* (Cockburn, 1998).

譯註：下面概述全方位團隊樣式。

◆ 團隊合作方式：全方位團隊

簡介 開發某個子系統時需要具備多種技能，不過大家都是專精於某個方面的人，所以我們集合各種專長的人組成一個團隊。

徵兆

- ◆ 聽到有人抱怨說目前的開發方式跟「牆上一團爛泥」一樣（譯註：沒有開發流程的情形）。
- ◆ 聽到有人抱怨說我們採用的是很官僚的開發流程。
- ◆ 互動溝通不良。
- ◆ 開發團隊是根據專長或由交付不同結果的開發階段來分組的。
- ◆ 移交工作時是把用特定格式所寫成的文件交給對方，而不是造訪對方以說明工作內容，在文件格式中會點出需要交接的工作內容（譯註：採用重量級開發流程時常發生的情形）。

- ◆ 開發團隊無法把工作發現所得跟開發團隊的工作策略連接起來。
- ◆ 開發團隊成員彼此間不互相尊重。
- ◆ 大家都可能會被派去做任何事，而且有人抱怨每次轉換不同工作時，需要花時間來調適心裡、很浪費時間。

建議做法

針對要交付給顧客的每個功能或一整組功能，組成一個小團隊（由 2-5 人組成），讓他們一起合作完成工作。這個團隊中可能會有需求蒐集、使用者介面設計、技術性設計與寫程式、資料庫或測試方面的專家，我們可以在一開始時就指派這些專家參加，或逐步指派不同專家參加。此外，我們會對團隊成員一視同仁，不會偏愛某種專長的人。請調整團隊大小與其工作場所，讓他們可以彼此直接溝通，而不需要透過文件溝通。

雖然這個團隊有責任寫出文件給其他非團隊的專案成員看，不過團隊內部是不需要文件存在的。而且可以自行選擇工作分配方式。

我們需要協調各個不同團隊，讓它們的交付成果（例如需求文件、使用者介面設計結果、軟體架構等）有一致性。

如果整個團隊的工作都由一個人來負責，那麼他將很難專精於不同專長，而且執行各種不同專長的工作時，要調適心理、有不同的工作重點（開會時就需要具有跟設計 OO 框架時不一樣的心情，而且前者比後者更需要專注精神【譯註：開會時很容易被不同事務打斷，自然要更加專注】）。

另一方面，如果團隊太大的話，溝通起來就很痛苦。

寫作提示 21：處理失敗情況

使用案例的最大價值之一就是裡面會指出所有的擴充情況。在許多專案中，當程式設計師在某個時刻寫出這樣的程式時：

```
if (執行條件)
  then (做某件事)
else ...?
```

這時候，他會在 else... 上面停下來沉思：「我很納悶，系統這個地方應該會做什麼事呢？由於需求文件中沒有說明這個奇特情況，而且也沒有人可問，所以...。」接下來，他很快就會照自己的想法去寫這段程式：

```
else (照程式設計師的想法做某件事)
```

One of the great values of use cases is that they name all the extension conditions. On many projects, there is a moment when the programmer has just written

```
If <condition>
```

```
then <do this>
else ..?.
```

He stops to muse on the else. . . "I wonder what the system is supposed to do here? The requirements don't say anything about this odd condition, and I don't have anyone to ask about it. Oh, well, . . ." He then types something quick into the program:

```
else <do that>
```

這裡的「else」處理方式應該是需求文件中原本要記載的某個東西，而且有許多時候，它跟某些非常重要的企業規則有關。遇到這種情況時，我經常可以看見系統使用情形專家聚在一起或找其他相關人員開會，釐清這些情況下系統該怎麼做。

找尋失敗條件並寫出處理方式時，通常可以讓我們發現一些新的參與者、目標與企業規則。一般來說，這些東西都是很細微難測的，而且需要進行一些研究工作，也可能會因此而改變系統的複雜性。

如果你向來只習慣寫出成功情節，那麼請在下個使用案例中試著找出失敗情況與其處理方式。結果一定會讓你大吃一驚，對自己的發現感到振奮不已。

The "else" handling was something that should have been in the requirements document. Very often, it involves significant business rules. I often see usage experts huddling together or calling associates to straighten out just what should be done by the system in these situations.

Finding the failure conditions and writing the failure handling often turn up new actors, new goals, and new business rules. Often, these are subtle and require some research, or they change the complexity of the system.

If you are only used to writing the success scenario, try capturing the failures and failure handling on your next use cases. You are likely to be surprised and heartened by what you uncover.

寫作提示 22：專案剛開始與快結束時，工作職

稱是很重要的

專案剛開始與快結束時工作職稱是很重要的，不過在專案進行中工作職稱就變得不是那麼重要了。

專案剛開始時，我們要蒐集系統需滿足的所有目標，並且用很易讀的方式來組織它們。把焦點放在會受系統影響的工作職稱或社會角色上可以讓我們以很有效的方式進行腦力激盪，找到最好的第一版目標名稱。找到一長串目標清單之後，工作職稱還是可作為分組時的結構大綱，讓我們更容易審查這些目標、替指定目標排出優先順序。

Job titles are important at the beginning and at the end of the project, but not in the middle.

At the beginning of the project, you need to collect all the goals the system must satisfy and put them into a readable structure. Focusing on the job titles or societal roles that will be affected by the system allows you to brainstorm effectively and make the best first pass at naming the goals. Once a long list of goals is in hand, the job titles also provide a clustering scheme to make it easier to review and prioritize the nominated goals.

工作職稱讓我們以工作時的所需技能與工作形式作為目標特徵。這項資訊也可為我們的使用者介面設計帶來線索。

一旦大家開始去開發使用案例，這時候，藉由一些討論，我們就可以知道不同角色的工作內容是如何重疊的。此時，主要參與者的角色名稱會變得比較一般化(例如「下訂單者」)或跟實際使用系統人的工作職稱更不一樣。最後這些角色名稱只是用來提醒使用案例寫作人員的關鍵而已，讓大家知道某個參與者在這裡具有一個目標而已。

↑ 英 225

譯註：這段所描述的是在專案進行中，因為工作重點轉移到目標上面，所以工作職稱就變得不是那麼重要了。

等到我們要配置系統時，工作職稱又再度變得很重要。因為開發團隊必須：

Job titles allow you to characterize the skills and work styles of the different job. This information informs your design of the user interface.

Once people start developing the use cases, discussions will surface about how roles overlap. The role names used for primary actors become more generic (e.g., "Orderer") or more distant from the people who will actually use the system, until they are only placeholders to remind the writers that there really is an actor having a goal.

Once the system is deployed, job titles become important again. The development team must

- ◆ 指定不同的權限等級給特定使用者，限制每種資訊是可更新或只是可讀取的。
Assign permission levels to specific users for updating or perhaps just reading each kind of information.
- ◆ 根據工作職稱的所需技能組合以及不同團體會用到的使用案例來準備新系統的訓練教材。
Prepare training materials on the new system, based on the skill sets of the people with those job titles and which use cases each group will be using.
- ◆ 為了配置系統而包裝它，這時候我們會把屬於同一組的使用案例實作放在一起。
Package the system for deployment, putting clusters of use case

implementations together.

寫作提示 23：由參與者扮演角色

「參與者」可代表某個系統使用者的工作職稱，也可以代表他（假設這是一個人）在使用系統時所扮演的角色。我們用哪個術語來代表它並不是很重要，所以請不要花太多精力在參與者、工作職稱或角色間的差異上。

重要之處在於「目標」，它代表系統將會做什麼事。在系統生命過程中，到底是誰在要求這個目標是可以透過不斷協商加以調整的。當我們發現店長也會扮演銷售櫃員角色時，你可以：

"Actor" means either the job title of the person using the system or the role that person is playing during system use (assuming that it is a person). It is not significant which way we use the term, so don't spend too much energy on the distinction.

The important part is the goal, which says what the system is going to do. Just exactly who calls upon that goal will be negotiated and rearranged over the life of the system.

When you discover that the store manager also can act as a sales clerk, you can

- ◆ 在使用案例中說明主要參與者可能是「銷售櫃員或店長」（如果你是 UML 迷的話：請從這兩個參與者分別畫箭頭到代表這個使用案例的橢圓形上）。

Write in the use case that the primary actor is "either Sales Clerk or Store Manager" (UML fans: Draw the arrow from both actors to the ellipse).

- ◆ 說明「店長在執行這個使用案例時，可能扮演銷售櫃員的角色」（如果你是 UML 迷的話：請從店長畫代表一般化的箭頭指向銷售櫃員）。

Write "Store Manager might be acting in the role of Sales Clerk while executing this use case" (UML fans: Draw a generalization arrow from Store Manager to Sales Clerk).

- ◆ 產生一個叫做「下訂單者」的主要參與者。說明「當銷售櫃員或店長在執行這個使用案例時，可能扮演銷售櫃員的角色」（如果你是 UML 迷的話：請分別從銷售櫃員與店長畫代表一般化的箭頭指向下訂單者）

Create an "Orderer" to be the primary actor. Write "Sales Clerk or Store Manager is acting in the role of Orderer when running this use case" (UML fans: Draw generalization arrows from Sales Clerk and Store Manager to Orderer).

上面三種做法都是對的，我們可以根據哪一種做法最容易跟聽眾溝通來選擇。

大家要了解的是：每個人都可以扮演許多角色，執行某項工作時，他只是在扮演某個角色而已。具有某個工作職稱的人可扮演許多角色，我們不管到底是哪個工作職稱在扮演這個角色。

請重讀一遍 4.2 主要參與者中的為何主要參與者有時候不重要、有時候又很重要小節，以及寫作提示 22 專案剛開始與快結束時，工作職稱是很重要的，以了解我們如何把工作職稱轉變成角色名稱，而後來角色名稱又如何跟工作職稱產生對

應關係。

↑ 英 226

None of these is wrong, so you can choose whichever you find communicates to your audience.

Recognize that a person fills many roles, that a person in a job is just a person filling a role, and that a person with a job title acts in many roles even when acting the role of that job title. Recognize that the important part of the use case is not the primary actor's name but the goal. Recognize that it is useful to settle on a convention for your team to use so that they can be consistent in their use of job titles.

Review the subsection Why Primary Actors Are Unimportant (and Important) on page 55 and Reminder 22, Job Titles Sooner and Later, on page 225, to see how actor names shift to role names and map back to actor names.

寫作提示 24：偉大的畫圖騙局

自從 Jacobson 的第一本關於使用案例寫作方面的書 *Object-Oriented Software Engineering* (1993) 出版以來，就有許多人把使用案例的寫作焦點放在代表參與者的棒狀小人跟代表使用案例的橢圓形身上，卻忽略掉使用案例基本上是由文字所構成的。為何會有這種結果，本人實在百思不解。受到強勢的 CASE 工具業影響，使用案例已經有用圖形來建立模型的工具，不過卻還沒有用文字來建立模型的工具存在，而且許多工具也都在畫使用案例圖方面極盡所能。雖然 OMG 的 UML 標準是由對文字型使用案例寫作有經驗的人所訂定的，不過裡面並沒有修正這種錯誤觀點。個人猜測是強勢的 CASE 遊說團體影響了 OMG 的所作所為。在許多重大場合中，有人對我解釋說「UML 僅僅是一種工具交換標準」而已。因此，被隱藏在橢圓形背後的說明文字不知怎麼地變成標準之外、由每個使用案例寫作人員自行決定的個人勢力範圍。

For reasons that remain a mystery to me, many people have focused on the stick figures and ellipses in use case writing since Jacobson's first book, *Object-Oriented Software Engineering* (1993) came out, and have neglected to notice that use cases are fundamentally a text form. The strong CASE tool industry, which already had graphical but not text modeling tools, seized upon this focus and presented tools that maximized the amount of drawing in use cases. This was not corrected in the OMG's Unified Modeling Language standard, which was written by people experienced in textual use cases. I suspect that the strong CASE lobby affected OMG's efforts. "UML is merely a tool interchange standard," is how it was explained to me on several occasions. Hence, the text that sits behind each ellipse somehow is not part of the standard and is a local matter for each writer to decide.

不論形成這種論點的原因為何，我們現在的處境是：雖然橢圓形只能為我們帶給

非常少的資訊，不過現在卻有許多人認為這些橢圓形就是使用案例。有經驗的開發人員可能會嘲笑這一點。個人非常感謝 Andy Hunt 與 Dave Thomas 所畫、發人醒思的戲謔之作「被簡化的需求」這個使用案例觀點，請參見圖 22.2（摘自於 *The Pragmatic Programmer*，1999）。

橢圓形並不能取代說明文字，了解到這一點是很重要的事。使用案例圖中（有意地）省略掉參與者間的事情發生先後順序、資料與收送資訊。我們可以把它當作：
Whatever the reasons, we now have a situation in which many people think that the ellipses are the use cases, even though they convey very little information.

Experienced developers can be quite sarcastic about this. I thank Andy Hunt and Dave Thomas for their lighthearted spoof of the "requirements made easy" view of use cases in Figure 22.2 (from *The Pragmatic Programmer*, 1999).

It is important to recognize that ellipses cannot possibly replace text. The use case diagram is (intentionally) lacking sequencing, data, and receiving actor. It is to be used as

- ◆ 使用案例的目錄。

 - A table of contents to the use cases.

- ◆ 系統的情境圖，裡面會秀出參與者要求系統去達成的各式各樣、部分內容會重疊的目標，也可能會秀出系統要求次要參與者去做某些事的情況。

 - A context diagram for the system, showing the actors pursuing their various and overlapping goals, and perhaps the system reaching out to the secondary actors.

- ◆ 「整體構思」，裡面會秀出目標等級比較高跟比較低的使用案例間的關係。

 - A "big picture," showing how higher-level use cases relate to lower-level use cases.

就像寫作提示 14 *使用案例格式的核心價值與替代性價值*中所說的一樣，上述這些用途都是可行的。請記住：因為使用案例基本上是以文字所寫成的，所以我們應該用橢圓形圖來輔助這些說明文字，而不是取代它。圖 22.3 與表 22.1 中秀出呈現情境圖的兩種不同做法。表中所秀出的參與者與目標都跟使用案例圖一樣，不過為了清晰起見，不同參與者所具有的相同目標，會重複出現於各個參與者的目標中。

↑ 英 227

These are all fine, as described in Reminder 14, *Core Values and Variations*, on page 216. Just remember that use cases are fundamentally a text form, so use the ellipses to augment, not replace, the text. Figure 22.3 and Table 22.1 show two ways of presenting the context diagram. The table on the next page shows the same actors and goals, with common use cases repeated for clarity.



圖 22.2 「媽媽！我希望回家。」

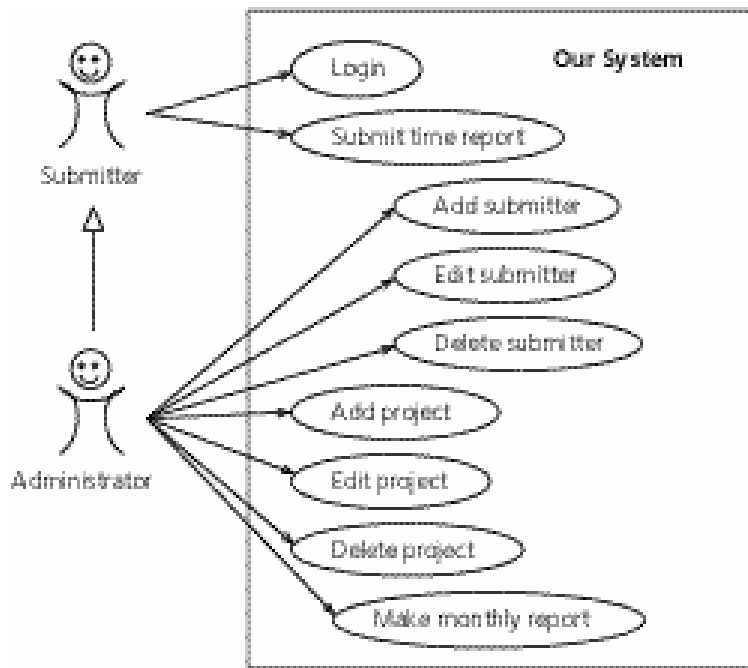


圖 22.3 用橢圓形圖這種形式所表現出來的情境圖 (摘自

Grady Booch 的手稿)。

表 22.1 用參與者 — 目標清單呈現出來的情境圖

參與者	目標
繳報告者	登入 繳交定期報告
管理者	登入 繳交定期報告 新增繳報告者

編輯繳報告者
刪除繳報告者
新增專案
編輯專案
刪除專案
做出月報告

↑ 英 228

寫作提示 25：偉大的工具之爭

有一件事令人感到很難過，那就是目前市場上任何工具對使用案例的支援都還不是很好。雖然有許多公司宣稱在文字或圖形方面都有支援使用案例，不過卻沒有一個工具真的有包含接近 2.3 圖形模型中所陳述的那個超模型，而且絕大多數都很難用。結果使用案例寫作人員只能在一堆不適用的工具中勉為其難選擇一種工具去用。

Sadly, use cases are not well supported by any tools currently on the market. Many companies claim to support them, either in text or in graphics. However, none of their tools contain a metamodel close to that described in Section 2.3, The Graphical Model, and most are quite hard to use. As a result, the use case writer is faced with an awkward choice.

Lotus Notes。

到目前為止，它還是我的最愛。雖然 Lotus Notes 中沒有包含使用案例的超模型，不過它卻支援協同工作、超連結、共通範本、文件歷史紀錄、在整組使用案例中快速觀看與編輯，以及可以很容易建構出可搜尋的 view。這些都是很重要的優點。Lotus Notes 也可讓我們用不同 view 去展開同一個資料庫中的資料描述內容。當我們更新範本時，資料庫中的所有使用案例也會同時更新。這樣的範本很容易設，也非常簡單易用。在一個以固定成本跟贊助顧客喊價的專案中，我曾經用 Lotus Notes 審查過超過 200 以上的使用案例。

Still my favorite, Lotus Notes has no metamodel of use cases but supports cooperative work, hyperlinking, common template, document history, quick view-and-edit across the use case set, and easily constructed sortable views. These are all significant advantages. Lotus Notes also allows the expanding data descriptions to be kept in the same database but in different views. When you update the template, all use cases in the database are updated. The template is fairly easy to set up and extremely easy to use. I used Lotus Notes to review over 200 use cases on a fixed-cost project bid with the sponsoring customers.

跟任何只支援簡單文字的工具一樣，Lotus Notes 的缺點在於：替主要成功情節與擴充情節中的動作步驟重新編號的功能不足，因此編輯使用案例很快就會變成

一件煩人的事。超連結最終都會過時；手動插入的「後退」連結也會很快過時。由於超連結中沒有自動化產生的「後退」連結，所以我們無法判別出究竟是哪一個目標等級比較高的使用案例呼叫你正在看的這個使用案例。

The drawback to Lotus Notes, as to any of the plain-text tools, is that renumbering the steps and extensions while editing a use case soon becomes a nuisance. The hyperlinks eventually become outdated; manually inserted backlinks become outdated very soon. There are no automated backlinks on the hyperlinks, so you can't tell which higher-level use cases invoke the use case you are looking at.

Lotus Notes 最吸引我的地方在於它可以很容易跟加過註解的參與者 — 目標清單結合在一起，由整組使用案例動態產生一個 view 給我們看。我們只要寫出一個新的使用案例，view 中馬上就會出現它的蹤跡。這個 view 同時也是具有超連結功能的目錄、參與者 — 目標清單，以及進度追蹤表。有時候，我喜歡根據優先順序、發行版本、完成狀態與使用案例名稱來看這些使用案例，其他時候則喜歡根據主要參與者或主題範圍、目標等級與使用案例名稱來看使用案例。

What makes Lotus Notes most attractive to me is its ease of use combined with the way the annotated actor-goal list provides a dynamically generated view of the use case set. Just write a new use case, and the view immediately shows its presence. The view is simultaneously a hyperlinked table of contents, an actor-goal list, and a progress tracking table. I like to view the use cases by priority, release, state of completion, and title, or by primary actor or subject area, level, and title.

具超連結功能的文字處理器。

如果某個文字處理器具有超連結功能的話，那麼我們還是有能力用它來處理使用案例。請把使用案例範本加入範本檔中。然後根據這個範本寫出每個使用案例的說明文字檔，我們應該能夠很容易就產生使用案例間的連結。不過，請記得不要任意改變檔名！使用案例寫作人員對文字處理器都很熟悉，而且用它來寫敘事情節也是很舒適的。

文字處理器也具有跟 Lotus Notes 一樣的所有缺點。更重要的是，它無法列出所有使用案例、根據發行版本或狀態排序，讓我們可以用單擊方式來開啓它。換言之，我們必須另外建構並維護一個包含全部使用案例的清單，而且這個清單很快就會過時。此外，針對使用案例範本，我們也缺乏整體更新的機制，所以隨著時間過去，我們將會有使用案例範本的好幾個不同版本。

With hyperlinking, word processors finally became viable for use cases. Put the use case template into a template file. Put each use case into its own text file using that template, and it becomes easy to create links across use cases. Just don't change the file's name! Writers are familiar with word processors and are comfortable using them to write stories.

Word processors have all the drawbacks of Lotus Notes. More significantly, they provide no way to list all the use cases, sorted by release or status, and click them

open. This means that a separate, overview list has to be constructed and maintained, and will soon be out of date. There is no global update mechanism for the template, and so multiple template versions tend to accumulate over time.

關聯式資料庫。

我曾經看過、聽過有人嘗試要把參與者、目標與動作步驟的模型放入關聯式資料庫（例如微軟的 Access）中。雖然這是一種很自然的想法，不過因此而產生的工具卻難用的可怕，逼得使用案例寫作人員回頭去用他們的文字處理器。

↑ 英 229

I have seen and heard of several attempts to put the model of actors, goals, and steps into a relational database such as Microsoft Access. While this is a natural idea, the resulting tools are awkward to use, sending the use case writers back to their word processors.

需求管理工具。

特殊化的需求管理工具（例如 DOORS 或 Requisite Pro）慢慢越來越被大家所接受。它們提供自動化的前進、後退超連結，而且它們本來就是拿來處理以文字為基礎的需求描述。壞處則是：它們都沒有提供主要成功情節與擴充情節的模型，這是使用案例的整個核心所在。我曾看過少數以這些工具所寫出來的使用案例，它們都非常長、使用大量的縮排、編號與行號，讓這些使用案例變得很不好讀（請回想寫作提示 2 把使用案例寫得很容易讀與寫作提示 20 偏好藍色牛仔褲【譯註：簡單易用的意思】）。如果你正在使用這種工具，請找出一種方式讓敘事情節更加易讀。

Specialized requirements management tools, such as DOORS or Requisite Pro, are becoming more common. They provide automated forward and backward hyperlinks and are intended for text-based requirements descriptions. On the minus side, none that I know of supports the model of main success scenario and extensions that is at the heart of use cases. The few use cases I have seen from such tools are very lengthy, with a great deal of indenting, numbering, and lines, making them hard to read (remember Reminder 2, Make the Use Case Easy to Read, on page 205, and Reminder 20, Blue Jeans Preferred, on page 224). If you are using such a tool, find a way to make the story shine through.

CASE 工具。

在好的一面，CASE 工具可讓我們一次整體改變超模型中的任何實體，也可以自動化產生「後退」連結。然而，就像之前所講的一樣，它們意圖用方塊（譯註：物件或類別）跟箭頭（譯註：訊息）畫出互動情形，這種做法不利於寫出說明文字（譯註：說明文字必須隱藏在箭頭後面）。因此，這樣的循序圖不合適作為文字型使用案例的替代品，而且大部分的 CASE 工具在輸入說明文字方面，都只提供比對話框還多一點的協助而已。我曾看過原本用 CASE 工具來寫使用案例的團隊，最後拒絕再去用它，回過頭去用文字處理器。

看完上述討論之後，我們似乎沒有找到用起來可以很愉快的工具。祝大家好運吧！

On the plus side, CASE tools support global changes to any entity in its metamodel and automated back links. However, as described earlier, they tend to be built around boxes and arrows, doing poorly with text. Sequence diagrams are not an acceptable substitute for textual use cases, and most CASE tools offer little more than a dialog box for text entry. I have seen writing teams mutiny and revert to word processing rather than use their CASE tools.

That leaves you with a less than pleasant choice to make. Good luck.

寫作提示 26：用使用案例標題與使用案例簡介

來規劃專案

關於以使用案例來追蹤專案進度，或以參與者 — 目標清單作為專案規劃框架這兩方面，請重看一下 17.1 *專案組織中的使用案例*。下面則是相關的寫作提示。

Review Section 17.1, Use Cases in Project Organization, for the pluses and minuses of use cases employed to track project progress and for an example of the actor-goal list as a project-planning framework. Here are the reminders.

使用案例規劃表。

請把參與者與目標放在表的最左邊兩欄，接下來的欄中則記錄任何有需要的東西：例如企業價值、複雜度、發行版本、負責的開發團隊、完成度、效能需求、外部介面等等。

當我們用這樣一個表來做規劃時，開發團隊可以跟贊助者協商出每個使用案例的實際開發優先順序。大家一起討論企業需要 v.s. 技術困難度、企業相依性，以及技術相依性，想出一個開發先後順序。

Put the actors and goals in the leftmost two columns of a table, and in the next columns record any of the following as needed: business value, complexity, release, team, completeness, performance requirement, external interfaces, and so on.

Using this table, your team can negotiate over the actual development priority of each use case. They will discuss business need versus technical difficulty, business dependencies, and technical dependencies, and come up with a development sequence.

只交付部分使用案例出來。

就像 17.1 *專案組織中的使用案例* 裡面的 *處理橫跨不同發行版本的使用案例* 小節中所描述的一樣，我們經常必須決定要在某個特定發行版本中交付一部分的使用案例而已。大部分開發團隊只會很簡單地用黃色色筆強調這個部分，或者用粗體

指出使用案例中的哪些部分會交給顧客。我們可能希望在規劃表中標示出：使用案例在哪個發行版本中第一次出現，以及它會在哪个發行版本中最後一次整個交給顧客。

↑ 英 230

As described in the subsection *Handle Use Cases Crossing Releases* on page 169, you will quite often decide to deliver only part of a use case in a particular release. Most teams simply use a yellow highlighter or bold text to indicate which portion of a use case that is. You will want to note in the planning table the first release in which the use case shows up and the final release in which the use case will be delivered in its entirety.

附錄 A UML 中的使用案例

統一模型語言（UML）中定義了一些規定大家去用的圖示。裡面雖然沒有提到使用案例內容或寫作風格方面的東西，不過有許多地方卻很複雜，需要跟大家討論一下（譯註：本書作者認為 UML 中對使用案例所抱持的圖形觀點是一種迷思，他認為使用案例應該是以文字為主、圖形為輔。另一方面，UML 中所定義的三種使用案例間關係也有許多值得討論之處）。請把精力放在學習如何寫出清晰的說明文字，不要花太多精力在使用案例圖上。如果你真的喜歡畫圖，那麼請把焦點放在學習使用案例間關係的基本概念，然後自行決定一些簡單的畫圖準則，讓畫出來的圖很清楚。

The Unified Modeling Language defines graphical icons that people are determined to use. It does not address use case content or writing style, but does provide lots of complexity for people to discuss. Spend your energy learning to write clear text instead. If you like diagrams, learn the basics of the relations and then set a few simple standards to keep the drawings clear.

A.1 橢圓形與簡單棒形人偶圖示

當你走到白板前面想畫出人們使用系統的圖時，很自然地，會用棒形人偶代表一個人、橢圓形或方塊代表這個人要求執行的使用案例，而且替棒形人偶加上參與者的名稱、橢圓形加上使用案例的名稱。它能提供的資訊跟參與者 — 目標清單一樣，只是呈現方式不同罷了。這樣的圖可作為使用案例目錄。

到目前為止，一切看起來都還不錯、很自然。

不過，如果你或讀者相信我們可用這樣的圖定出系統的功能需求時，問題就來了。有些人被這些圖沖昏了頭，認為它們可以把困難的東西變得很簡單（就像寫作提示 17 寫使用案例時先求廣度中的圖 22.1 一樣，我們可以很簡單地逐步秀出使用案例中的內容）。結果，他們會試著在圖中盡可能放進很多東西，希望可以不用寫出說明文字。下面是發生這種情況時的一些症狀。

When you walk to the whiteboard to draw pictures of people using the system, it is very natural to draw a stick figure for a person, and ellipses or boxes for the use cases they are calling upon. Label the stick figure with the title of the actor and the ellipses with the titles of the use cases. The information is the same as in the actor-goal list, but the presentation is different. The diagrams can be used as a table of contents.

So far, all is all fine and normal.

The trouble starts when you or your readers believe that the diagrams define the system's functional requirements. Some people become infatuated with the diagrams, thinking that they will make a hard job simple (as in Figure 22.1 on page 222). They

try to capture as much as possible in the diagram, hoping, perhaps, that text will never have to be written. Here are events that are symptomatic of this situation.

最近，我的班上有一位學生拿了一幅原本捲起來、長達好幾尺的圖給我看。他從一邊展開圖給我看，裡面畫滿了橢圓形跟四處亂飛的箭頭，混合了各種使用案例間的關係，包括包含關係、擴充關係與一般化關係（這些關係看起來雖然一樣，不過當然不同，差異只在箭頭上的小說明文字上）。他希望問我這個專案是否有正確使用所有的關係，而沒察覺到這麼龐大複雜的圖，別人幾乎都沒辦法理解系統裡面在做什麼事。

↑ 英 233

還有一個學生很自豪地秀給我看他是如何「修正」圖中的明顯缺點：一般的圖中沒有秀出呼叫子使用案例的先後順序。他在圖中新增了更多 UML 中的前導關係（precede relation），以表示某些子使用案例比其他的還先被呼叫到。結果當然是得到更複雜的圖，這樣的圖不但比同樣的說明文字佔更多空間，也更加難讀。這裡改寫一段諺語：「他把原本易讀的千言萬語畫在自己難讀的世界中」。（譯註：本來的諺語是「一張圖勝過千言萬語」）
圖形是一種二維、協助記憶的東西，它有助於提高大家的認知度，也就是說，協助大家把焦點放在使用案例間的關係上。請把使用案例圖用來協助我們了解使用案例間的關係，而不是取代說明文字。

在這個前提下，讓我們看一下 UML 中的各種使用案例間關係。

A drawing is a two-dimensional mnemonic device that serves a cognitive purpose, namely, to highlight relationships. Use them for this purpose, not to replace the text. With this purpose in hand, let us look at the individual relations in UML.

A.2 UML 中的包含關係

如果某個基本使用案例（base use case）中有一個動作步驟會呼叫自己以外的其他被包含使用案例名稱，那麼我們就把這種做法稱為：某個基本使用案例去包含另一個被包含使用案例（included use case）。這是目標等級比較高跟目標等級比較低的使用案例間一種正常、顯而易見的關係。被包含使用案例中所描述的目標等級會比基本使用案例還低（譯註：這樣的被包含使用案例稱為子使用案例）。這時候，動作步驟中的動詞片語就是潛在的子使用案例名稱。如果你沒有把這個目標獨立開、變成一個使用案例，那麼它僅僅就是一個步驟而已。如果你把它分開自己獨立成一個使用案例，那麼這個動作步驟就是在呼叫子使用案例（這是我自己的字說的）或動作步驟中包含某個被包含使用案例的行為（這是用 UML 1.3 版的字說的）。在 UML 1.3 版之前，我們會說這個動作步驟在使用（use）目標等級比較低的使用案例（不過，「使用」這個字現在已經不用了）。我們會從（目標等級比較高的）基本使用案例畫虛線箭頭到被包含使用案例上，以表示基本使用案例「知道」跟被包含使用案例有關的事，如圖 A.1 所示。

A base use case includes an included use case if an action step in the base use case calls out the included use cases's name. This is the normal and obvious relationship between a higher-level and a lower-level use case. The included use case describes a lower-level goal than the base use case.

The verb phrase in an action step is potentially the name of a sub use case. If you never break out that goal into its own use case, it is simply a step. If you do break out that goal into its own use case, then the step calls the sub use case, in my vocabulary, or it includes the behavior of the included use case, in current UML 1.3 vocabulary. Prior to UML 1.3, it was said to use the lower-level use case (that phrase is now out of date).

A dashed arrow goes from the (higher-level) base use case to the included use case, signifying that the base use case "knows about" the included one, as illustrated in Figure A.1.

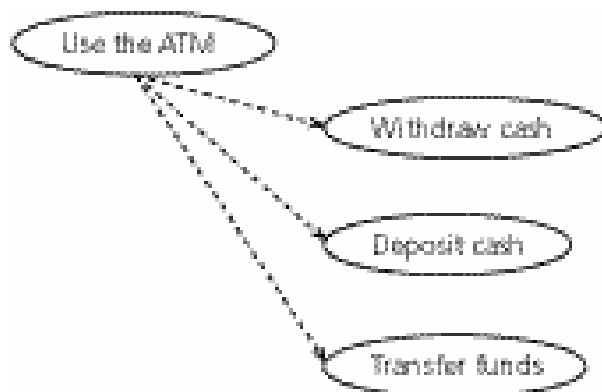


圖 A.1 畫出包含關係

寫作指引 13：把等級比較高的目標畫得高一

點

在圖中，請務必把等級比較高的目標畫得比等級比較低的高一點。這樣做可減少我們對目標等級發生混淆情形，對讀者來說也是很直覺的。當我們這麼做時，從基本使用案例畫到被包含使用案例的箭頭總是會往下畫。

UML 允許我們改變 UML 元素的圖形呈現方式。我發現大部分的人在徒手畫圖時，都只會很簡單地從基本使用案例畫一條實線箭頭到被包含使用案例上（因為畫虛線是件很討厭的事）。這樣沒什麼不好，而且也沒什麼錯。如果你用畫圖程

式的話，則可用它所提供的箭頭形式來畫。

Always draw higher-level goals higher up on the diagram than lower level goals. This helps reduce goal-level confusion and is intuitive to readers. When you do this, the arrow from a base use case to an included use case will always point down.

UML permits you to change the pictorial representation of each of its elements. I find that most people, when drawing by hand, simply draw a solid arrow from base to included use case (dashed arrows are tedious). This is fine, and now you can justify it. With a graphics program, you will probably use the arrow style that comes with the program.

對大部分程式設計師來說，包含關係就像程式語言中古老的副程式呼叫方式一樣。這樣想沒什麼問題，也沒什麼丟臉之處；相反地，我們是很自然地在用一種正常機制，它既可用在寫程式上，也可用在日常生活中。偶爾，我們可呼叫可參數化使用案例、傳功能參數給它，甚至還可以有傳回值（請參見第 14 章 *CRUD 使用案例與可參數化使用案例*）。不過，心裡面請時時記住：使用案例是拿來跟其他人溝通的，而不是跟 CASE 工具或編譯器溝通（譯註：作者的意思是請畫得很清楚、易懂）。

It should be evident to most programmers that the includes relation is the old subroutine call from programming languages. This is not a problem or a disgrace; rather, it is a natural use of a natural mechanism, which we use both in programming and in our daily lives. On occasion, it is appropriate to parameterize use cases, pass them function arguments, and even have them return values (see Chapter 14, Two Special Use Cases). Keep in mind, though, that the purpose of a use case is to communicate with another person, not with a CASE tool or a compiler.

A.3 UML 中的擴充關係

我們會在擴充使用案例（extending or extension use case）中寫出基本使用案例名稱、定出它中斷基本使用案例時的情境，以擴充基本使用案例。相反地，基本使用案例中則不會寫出擴充使用案例名稱。如果你希望有任何數量的使用案例可中斷基本使用案例，卻不會有每新增一個中斷的使用案例，就更新基本使用案例一次的維護惡夢，那麼擴充使用案例就是一種很有用的做法（請參見 10.2 擴充使用案例）。

An extending or extension use case extends a base use case by naming the base use case and defining the circumstances under which it interrupts the base use case. The base use case does not name the extending one. This is useful if you want any number of use cases to interrupt the base one without the maintenance nightmare of updating the base use case each time a new, interrupting use case is added. (See Section 10.2, Extension Use Cases.)

擴充使用案例中會詳述當我們由基本使用案例執行前者時，後者在行為方面的內部條件（譯註：相當於使用案例中的事先條件）與驅動條件（譯註：相當於使用案例中的觸發事件）。當基本使用案例執行某些行為時，如果這些條件符合的話（譯註：內部條件加上驅動條件），就會跳到擴充使用案例中執行。當擴充使用案例結束時，行為又會回到當初離開基本使用案例的地方繼續執行。

譯註：如果是呼叫子使用案例，而且是以成功情況傳回的話，那麼應該是從下個動作步驟繼續執行。

Behaviorally, the extending use case specifies some internal condition in the base use case and a triggering condition. Behavior runs through the base use case until the condition occurs, at which point it continues in the extending use case. When the extending use case finishes, the behavior picks up in the base use case where it left off.

Rebecca Wirfs-Brock 就很生動地描述擴充使用案例，他說：擴充使用案例就像基本使用案例的**补丁**（程式設計師應該可以把它類推成程式补丁【**program patch**】）。其他程式設計師則把它視為文字版的 **mock** 程式指令、*come-from* 述句。

譯註：相信很多人都沒寫過 **COME-FROM** 述句，所以下面就寫出一段 FORTRAN 程式解釋它的寫法。

```
10 J = 1
11 COME FROM 20
12 WRITE (6, 40) J STOP
13 COME FROM 10
20 J = J + 2
40 FORMAT (14)
```

這段程式執行時，第 10 行先把 J 設成 1，然後第 11 行會先跳到第 20 行執行，把 J 設成 3，第 20 行執行結束後再跳到第 11 行的下一行：第 12 行執行。第 12 行會印出 J，然後結束程式。第 13、40 行都不會執行到。

如果不看第 13、40 行的話，其中第 11 行相當於擴充點，而第 20 行相當於擴充使用案例，10 到 12 行相當於基本使用案例。

Rebecca Wirfs-Brock colorfully refers to the extending use case as a patch on the base use case (programmers should relate to the analogy of program patches!). Other programmers see it as a text version of the mock programming instruction, the *come-from* statement.

其實，使用案例中的擴充情況就是一種很自然的擴充形式。擴充使用案例相當於把擴充情況與其處理方式，從某個使用案例中移開、自己獨立變成一個使用案例（請參見 10.2 **擴充使用案例**）。我們可以把擴充使用案例視為長得太大的擴充情節，因此需要有自己的獨立空間。

在 UML 中，**擴充**（*extends*）關係的預設畫法跟**包含**（*includes*）關係一樣都是虛線箭頭，不過**擴充**關係是從擴充使用案例指向基本使用案例，而**包含**關係則是

從基本使用案例指向子使用案例，而且擴充關係會替箭頭加上「<<extends>>」造型 (stereotype)，而包含關係則會加上「<<includes>>」造型。我則會畫一個掛勾把擴充使用案例勾在基本使用案例上，以強調包含關係跟擴充關係是不同的。請參見圖 A.2。

↑ 英 235

We use the extension form quite naturally when writing extension conditions within a use case. An extension use case is just the extension condition with the handling pulled out and turned into a use case on its own (see Section 10.2, Extension Use Cases). Think of it as a scenario extension that outgrew its use case and was given its own space.

The default UML drawing for extends is a dashed arrow (the same as for includes) from extending to base use case, with the phrase "<<extends>>" set alongside it. I draw it with a hook from the extending use case back to the base use case, as shown in Figure A.2, to highlight the difference between includes and extends relations.

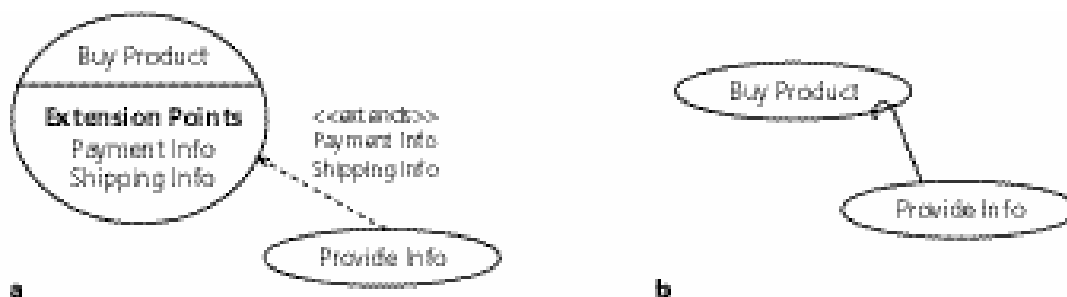


圖 A.2 畫出擴充關係

圖 A.2(a)中所秀出來的是 UML 中預設的擴充關係畫法 (這個例子是從 *UML 精華*【Fowler, 1999】中所摘錄出來的)。圖 A.2(b)中所秀出來的則是用掛勾連接的畫法。

Figure A.2(a) shows the default UML way of drawing extends (the example is from *UML Distilled* (Fowler, 1999)). Figure A.2(b) shows the hook connector.

寫作指引 14：把擴充使用案例的位置畫得低

一點

一般來說，擴充使用案例的目標等級會比它所擴充的基本使用案例還低，所以圖中應該把它的位置畫得比基本使用案例還低。然而，擴充關係中是目標等級比較低的使用案例知道目標等級比較高的使用案例。所以，我們應該把箭頭或掛勾朝

上往代表基本使用案例的橢圓形方向畫。

An extension use case is generally at a lower level than the use case it extends, so it should be placed lower on the diagram. In the extends relation, however, it is the lower use case that knows about the higher use case. Therefore, the arrow or hook should be drawn up from the extending to the base use case symbol.

寫作指引 15：對 UML 中的各種使用案例關

係，請用不同的箭頭形狀

UML 中刻意不規定連接使用案例的箭頭形狀。任何使用案例間的關係都可用開放式箭頭加上說明關係的小字來畫。之所以有這種概念是因為：不同的工具提供廠商或專案開發團隊可能希望自行決定箭頭形狀，所以 UML 標準中不禁止這種行爲。

很不幸地，大家都只是簡單在用這種沒有差異的箭頭來畫出各種使用案例間的關係，結果畫出來的圖變得很不容易讀。讀者必須細看箭頭上的小說明文字才能了解這些關係是做什麼用的，沒有簡單、易見的線索可幫大家一眼看出這些關係。由於 UML 中規定這種沒有差異的箭頭，加上缺乏其他畫圖慣例，結果畫出來的使用案例圖看起來真的都很難理解。

因此，請花一點功夫替下面這三種關係定出不同的箭頭形式：

UML deliberately leaves unresolved the style of the arrows connecting use case symbols. Any relation can be drawn with an open-headed arrow and some small text to say what the relation is. The idea is that different tool vendors or project teams might want to customize the arrow style, and the UML standard should not prevent this.

The unfortunate consequence is that people simply use the undifferentiated arrows for all relations, which makes the drawings hard to read. The reader must study the small text to detect which relations are intended, and later on there will be no simple visual clues to help him remember the relations. This, combined with the absence of other drawing conventions, makes many use case diagrams truly incomprehensible.

For that reason, take the trouble to set up different arrow styles for the three relations:

- ◆ 包含 (includes) 關係：請用預設的開放式箭頭，因為它是最常用的一種箭頭。

Includes: Use the default, open-headed arrow, as it should be the most frequently used one.

- ◆ 一般化 (generalizes) 關係：在 UML 中，標準的一般化關係箭頭是用三角中空箭頭代表的，請用這種箭頭。

Generalizes: The standard generalizes arrow in UML is the triangle-headed arrow. Use that.

- ◆ **擴充 (extends) 關係**：請用一種跟包含關係、一般化關係完全不同的箭頭形狀。我用掛勾符號把擴充使用案例勾在基本使用案例上。讀者發現這種箭頭很容易、也馬上可以理解，不會跟其他 UML 符號產生混淆，而且它也具有把擴充使用案例「勾在」基本使用案例上的隱喻。不論你用哪一種箭頭，請讓擴充關係跟圖中的其他關係看起來不同。

Extends: Create a different shape entirely. I use a hook from the extending to the base use case. Readers find it immediately recognizable, it doesn't conflict with any of the other UML symbols, and it brings its own metaphor of an extending use case having its hooks in the base use case. Whatever you use, make the extends connector stand out from the other ones in the diagram.

正確的擴充關係用法

最常寫出擴充使用案例的情況是：使用者想在許多地方中斷基本使用案例、執行非同步服務（最初的討論請看 10.2 擴充使用案例中的何時該用擴充使用案例小節）。通常，這些非同步服務是由其他開發團隊負責開發的，而且我們通常會在建構套裝軟體的套件時這麼做，如圖 A.3 所示。

其他常見的情況則是在替鎖定不變的需求文件新增功能時發生。在漸增式、分不同時期開發的專案中，你可能會在每個版本發行之後鎖定需求，再用新增功能去擴充這個鎖定不變的使用案例。

The most common occasion for creating extension use cases (originally discussed in the subsection When to Use Extension Use Cases on page 116) is when there are many asynchronous services the user might activate to interrupt the base use case. Often, they are developed by different teams. These occasions come up in the construction of shrink-wrapped software packages, as illustrated in Figure A.3. The other common occasion is when you are writing additions to a locked requirements document. In an incrementally staged project, you might lock the requirements after each delivery and then extend a locked use case with one that adds function.

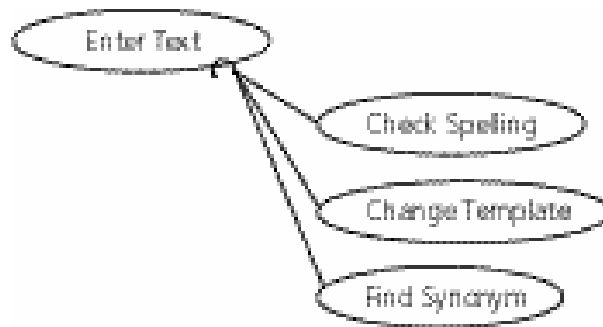


圖 A.3 從基本使用案例中擴充出來、中斷它的三個使用

案例

擴充點

我們之所以發明擴充關係，第一個原因是因為不想修改之前系統的需求文件。在最初發展出使用案例的電話系統中，由於這種行業經常需要新增一些非同步服務，所以擴充關係是很實用的。它讓新開發團隊可先鎖定不變的需求文件，再用很安全的方式開發軟體。在基本使用案例中適合之處，我們可以把新的、非同步服務放到需求當中，卻不用修改原本系統需求文件中的任一行文字。

↑ 英 237

The reason that extends was invented in the first place was the practice of never touching the requirements file of a previous system. In the original telephony systems where use cases were developed, the business often added asynchronous services, and so the extends relation was practical. The new team could build on the safely locked requirements document, adding the requirements for a new, asynchronous service in the base use case wherever it was appropriate, without touching a line of the original system requirements.

然而，在行為中參照其他使用案例是有問題的。如果不用行號的話，我們該如何說明擴充行為是發生在哪一點呢？如果有用行號的話，當我們編輯基本使用案例、行號改變之後，又會發生什麼事呢？

其實，行號代表行標籤。這樣的話，我們就不一定要用數值或循序的行號。加行號是為增加可讀性，讓擴充情況有參考點。不過，由於行號通常是循序的數字，所以隨著時間過去，它會不斷改變。

However, referencing behavior in another use case is problematic. If no line numbers are used, how should we refer to the point at which the extension behavior picks up? And if line numbers are used, what happens if the base use case gets edited and the

line numbers change?

Recall that the line numbers are really line labels. As such, they don't have to be numeric or sequential. They are there for readability and so that the extension conditions have a reference point. Usually, however, they are sequential numbers, which means that they will change over time.

爲了解決上述議題，我們引進擴充點（extension point）的概念。所謂的擴充點就是基本使用案例中公開、看得見的標籤，我們用這樣的暱稱來標示使用案例的行為中某個地方（從技術層面來看，一個擴充點可代表一組位置，不過我們暫時先不討論這一點）。

公開、看得見的擴充點雖然解決了一些問題，卻也帶來一個新問題。基本使用案例寫作人員必須了解使用案例中哪些地方是可擴充的。不論什麼時候，只要有人想在一個新地方擴充基本使用案例，那麼他就必須去修改基本使用案例。回想一下，擴充關係的原始目的不就是要避免修改基本使用案例嗎？

Extension points were introduced to fix these issues. An extension point is a publicly visible label in the base use case that identifies a moment in the use case's behavior by nickname (technically, it can refer to set of places, but let's leave that aside for the moment).

Publicly visible extension points introduce a new problem. The writers of a base use case are charged with knowing where it can get extended. They must go back and modify it whenever someone thinks up a new place to extend it. Recall that the original purpose of extends was to avoid having to modify the base use case.

我們必須解決上面這個問題。我發現：公開宣告擴充點所帶來的好處比不宣告還差。所以我自己是不用暱稱的，比較喜歡用文字來說明基本使用案例中哪個地方是擴充使用案例可擴充的，就像下面 ATM 範例中所秀出來的一樣。

如果你有用擴充點的話，請不要在圖中秀出這些擴充點，因為它們佔掉大部分的橢圓形空間、掠奪使用者目光，模糊掉更重要的目標名稱（請參見圖 A.2）。同時，也請不要在圖中把擴充點參照到的地方寫出來，這樣只會把圖弄得更亂。

譯註：當我們在畫 UML 的圖時，通常會根據不同需要，選擇性地秀出模型元素（例如代表類別的方塊）中的某個部分（例如類別中的方法）。

You will have to deal with one of the above problems. I find publicly declared extension points more trouble than they are worth. I prefer describing textually where in the base use case the extending use case picks up, ignoring nicknames, as in the ATM example below.

If you do use extension points, don't show them on the diagram. They take up most of the space in the ellipse, dominating the reader's view and obscuring the much more important goal name (see Figure A.2). The behavior they refer to does not show up on the diagram. They cause yet more clutter.

關於擴充點，還有一件事要講。擴充點名稱所參照到、可擴充的地方，在基本使

用案例中可以不只代表一個地方，我們可依照自己需要，用一個擴充點名稱代表基本使用案例中所有需要新增擴充使用案例行爲的地方。例如，當我們對 ATM 新增一個擴充使用案例*用其他銀行的 ATM*時，應該就會希望採用這種做法。這個擴充使用案例可以這麼寫：

在接受、執行交易動作之前，系統要先取得顧客的允許，以便對額外服務收費。

...

完成顧客所要求的交易動作之後，系統會對顧客帳號收取額外服務費。
當然，我們可以只寫這麼多。

↑ 英 238

譯註：在上述例子中，擴充使用案例*用其他銀行的 ATM* 會在交易動作的前後新增行爲，所以基本使用案例*使用 ATM* 中被擴充的地方有兩個，我們可用一個擴充點代表這兩個地方。

There is one more thing about extension points. An extension point name is permitted to call out not just one place in the base use case where the extending use cases needs to add behavior but as many as you wish. You would want this for, say, an ATM when adding the extension use case Use ATM of Another Bank. The extending use case needs to say,

Before accepting to perform the transaction, the system gets permission from the customer to charge the additional service fee.

...

After completing the requested transaction, the system charges the customer's account the additional service fee.

Of course, you could just say that.

A.4 UML 中的一般化關係

某個使用案例可以把另一個比較一般性的使用案例*特殊化*(而一般性使用案例則把特殊使用案例*一般化*)。其中(特殊化)子代應該是(一般化)父代的「類似品種」。講得更確切些，UML 1.3 版中說明：「使用案例間的一般化關係隱含子代使用案例中會包含父代使用案例中所具有的所有屬性、一連串行爲與擴充點，也保有父代使用案例的所有關係」。

A use case may specialize a more general one (the general use case generalizes the specific one). The (specializing) child should be of a "similar species" to that of the (general) parent. More exactly, UML 1.3 says, "A generalization relationship between use cases implies that the child use case contains all the attributes, sequences of behavior, and extension points defined in the parent use case, and participates in all the relationships of the parent use case."

正確的一般化關係用法

有一個字可幫我們檢驗出正確使用一般化關係的時機：有**通稱**(generic)存在時。這時候，我們可能會寫出像「某種 XX」的句子。當你說「使用者會做**某種**動作時」，請特別注意，這時候你做的事可能就代表**一般化**(generalizes)關係的存在。下面是**使用 ATM** 使用案例中的一個情節片段。

A good test word is generic, as in "some kind of." Be alert for when you say, "The user does some kind of this action." When you do, you have a candidate for generalizes.

Here is a fragment of the Use the ATM use case.

1. 顧客放入 ATM 卡並輸入 PIN 號碼。
2. ATM 證實客戶帳號與 PIN 號碼。
3. 顧客選擇做某種交易動作：
 - 提款
 - 存款
 - 轉帳
 - 查詢餘額

顧客可不斷做交易動作，直到他選擇離開為止。

4. ATM 歸還卡片。

顧客在動作步驟 3 中會做什麼動作呢？我們把這些動作「通稱」為「交易動作」。在上面例子中，顧客可做**四種**交易動作。「通稱」或「某種 XX」提示我們有一般性或一般化的目標：**做某種交易動作**存在。在簡單的文字版使用案例中，我們不會特別注意到自己正在用使用案例間的一般化關係；我們只是簡單列出使用者可做、並持續會做的作業或交易動作種類而已。然而，在 UML 中，它意味著我們需要畫一條代表一般化關係的箭頭。

What is it the customer does in step 3? The generic answer is "a transaction." There are four kinds of transactions the customer can do. "Generic" and "kinds of" tip us off to the presence of the generic or generalized goal, Do a transaction. In the plain-text version, we don't notice that we are using the generalizes relation between use cases; we simply list the kinds of operations or transactions the user can do and keep going. In UML, though, this is the signal to drag out the generalization arrow.

事實上，我們有兩種做法可選。第一種是忽略代表各種可能動作的一般化業務，只**包含**特定作業就好，如圖 A.4(a)所示。第二種是產生一般性的**做某種 ATM 交易動作**使用案例，再把特定作業變成它的特殊化使用案例，如圖 A.4(b)所示。你愛用哪一種都可以。不過，如果我們只是想寫出散文般的使用案例說明文字，就不用產生一般化使用案例。因為這時候一般性的目標中幾乎不會有任何文字，所以沒有必要另外產生另一頁空的使用案例。然而，如果我們有畫圖的話，由於

沒有辦法直接用圖表達接下來要做的某種交易動作，所以必須寫出一般化的目標。

↑ 英 239

Actually, we have two choices. We can ignore the whole generalizes business and just include the specific operations, as shown in Figure A.4(a). Or we can create a general use case for Do One ATM Transaction and show the specific operations as specializations of it, as in Figure A.4(b).

Use whichever you prefer. Working in prose, I don't create generalized use cases.

There is rarely any text to put into the generic goal, so there is no need to create a new use case page for it. Graphically, however, there is no way to express does one of the following transactions, so you have to find and name the generalizing goal.

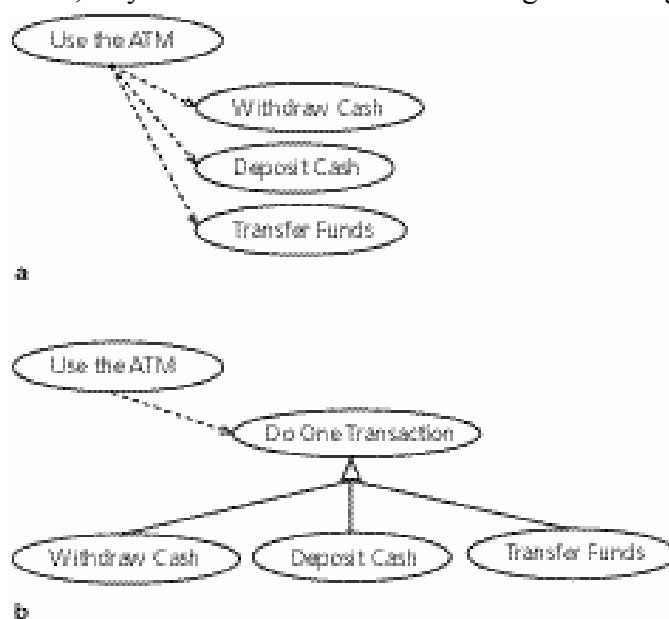


圖 A.4 畫出一般化關係。

這裡先產生某個一般性動作，再把整組被包含使用案例變成它的特殊化使用案例。

寫作指引 16：把一般性的目標畫得高一點

在圖中，請務必把一般性的目標畫得高一點，而且是由下而上畫出三角中空箭頭，而不要像包含關係或擴充關係一樣從側面畫，如圖 A.4 所示。

Always draw the general goal higher on the diagram, and point the triangular arrowhead up into the bottom, not the sides. Figure A.4 illustrates this.

使用一般化關係時，可能會發生的錯誤

當你同時用到參與者的特殊化跟使用案例的特殊化時，請特別小心。有一種該避免的畫法是：*讓特殊化參與者去使用特殊化使用案例*。爲了說明這一點，請參見圖 A.5。在圖中，我們試圖表達一個很自然的想法：銷售櫃員可結束任何一筆生意，不過另一種特殊的銷售櫃員 — 資深代理商 — 才可結束超過某定額度的生意。然而，圖中所表達的卻是正好相反的意思。

Watch out when combining the specialization of actors with the specialization of use cases. The idiom to avoid is that of a specialized actor using a specialized use case. To illustrate, Figure A.5 is trying to express the fairly normal idea that a Sales Clerk can close any deal but that it takes a special kind of sales clerk, a Senior Agent, to close a deal above a certain limit. However, it actually expresses the opposite.

請回想一下 4.2 主要參與者，我們曾提過：特殊化參與者可執行一般性參與者所能做的任何使用案例，而銷售櫃員就是一般化的資深代理商。對許多人來說，這一點好像不是可以很直覺想通的，不過它卻是官方用法、沒有錯。

另一種特殊化關係則似乎相當自然，結束一筆大生意是結束正常生意的特殊情況。然而，UML 中的規則是：*只要有提到一般性使用案例的地方，都可用特殊化使用案例取代它*。因此，圖 A.5 的意思是說：一般性的銷售櫃員可結束一筆大生意。

↑ 英 240

譯註：看參與者間的一般化關係時，我們會把焦點放在參與者跟使用案例的關係上，認爲特殊化參與者具有一般化參與者的所有關聯（association）關係，也就是說，前者可使用跟後者有關的所有使用案例。另一方面，對於使用案例間的一般化關係，由於使用案例是「被使用的」，所以我們會關注它的「介面」，既然介面相同，再加上用到一般化使用案例的所有地方都可用特殊化使用案例替代它，所以除了介面一樣之外，子代還要確保父代原本預期可得到的結果。它符合 *Liskov 替代原則* (Liskov substitution principle)，這種關係比子型態化 (subtyping) 更加嚴謹，後者只表示子代擁有跟父代一樣的介面可用而已。

Recall from Section 4.2, The Primary Actor, that the specialized actor can perform every use case the general actor can. Thus, the Sales Clerk is a generalized Senior Agent. To many people, this seems counterintuitive, but it is official and correct.

The other specialization seems quite natural, that closing a big deal is a special case of closing an ordinary deal. However, the UML rule is A specialized use case can be substituted wherever a general use case is mentioned. Therefore, the drawing says that an ordinary sales clerk can close a big deal!

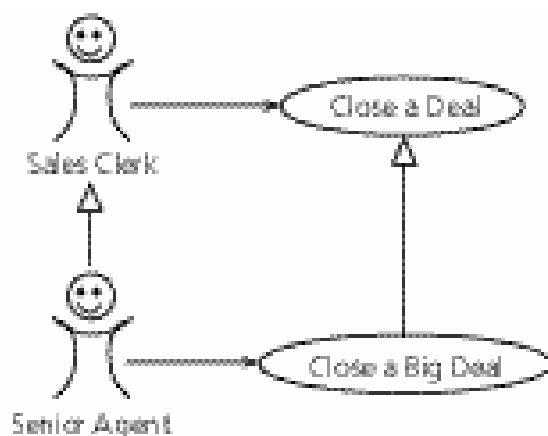


圖 A.5 使用一般化關係時，可能會發生的錯誤 — 結束一筆大生意

圖 A.6 才是正確的畫法。當你在看這張圖時，可能會問：「結束一筆小生意」是把「結束一筆基本的生意」特殊化，還是擴充它？因為寫文字型使用案例時不會產生這種困惑，這樣的疑慮只是損失時間而已，所以我把這個問題留給有興趣的讀者思考。

一般來說，一般化關係的問題是專業社群對它所代表的子型態與特殊化行為意義還不是很清楚，換言之，它所隱含的特性與選項為何。因為使用案例是行為描述，所以我們不知道行為描述特殊化之後的標準意義為何。

如果你真的要用一般化關係的話，我的建議是不要在一般化使用案例中寫出任何東西，就像上面的做某種交易動作一樣。然後在特殊化使用案例中再寫出所有行為，這樣一來，我們就不必擔心之前所描述到的那種陷阱。

↑ 英 241

譯註：如果「結束一筆小生意」只是在某些條件（擴充點）下比「結束一筆基本的生意」多出一些動作步驟的話，那麼應該就可認為前者去擴充後者，「結束一筆大生意」也是同樣情況。不過，圖 A.6 中應該是在「結束一筆基本的生意」中有一些寫得比較「一般性的」動作步驟，而在「結束一筆小生意」或「結束一筆大生意」中則會寫出特定的做法。

The corrected drawing is shown in Figure A.6. You might look at the drawing and ask: Does closing a small deal really specialize closing a basic deal, or does it extend it? Since working with text use cases will not put anyone in this sort of puzzling and economically wasteful quandary, I leave that question as an exercise to the interested reader.

In general, the problem with the generalizes relation is that the professional community has not yet reached an understanding of what it means to subtype and

specialize behavior, that is, what properties and options are implied. Since use cases are descriptions of behavior, there can be no standard understanding of what it means to specialize them.

If you do use the generalizes relation, my suggestion is to make the generalized use case empty, as in Do One Transaction above. Then the specializing use case will supply all the behavior, and you will have to worry only about the one trap just described.

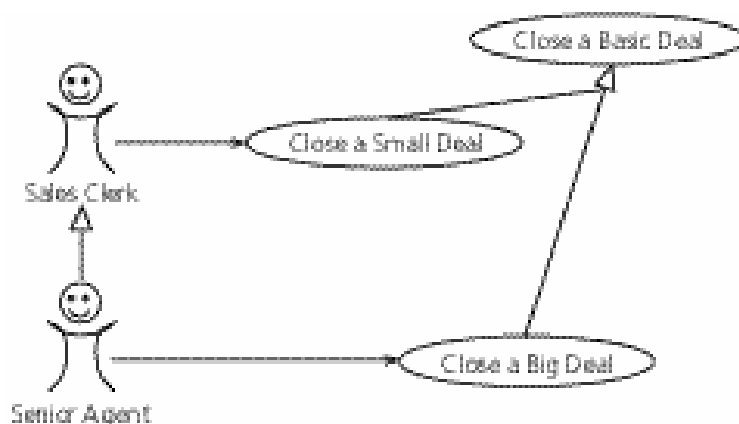


圖 A.6 結束一筆大生意的正確畫法

A.5 下層使用案例 vs. 子使用案例

在 UML 1.3 版規格書的附錄中，作者提到兩種大家很少聽見的使用案例間關係：下層使用案例（subordinate use case）與相對的上層使用案例（superordinate use case），它們都沒有相對應的圖示，也沒有用物件限制語言（OCL）明確規定它們，只有一些簡單的說明而已。

這些關係的目的是：讓我們秀出系統中元件的使用案例是如何合作，以達成比較大系統的使用案例。讓我們感到迷惑的是，元件本身都還沒有出來，它們的使用案例不就要在虛無空間中存活。這種做法好像是在畫一些莫名的合作圖或特殊功能分解，稍後再拿它們去解釋一些現存、合理的合作圖。根據 UML 規格書：

In the extended text section of the UML 1.3 specification, the authors describe two little-known relations between use cases that have no drawing counterpart and are not specified in the object constraint language but simply written into the explanatory text. These are the subordinate use case and its inverse, the superordinate use case.

The intent of these relations is to let you show how the use cases of a system's components work together to deliver the use case of the larger system. In an odd turn, the components themselves are not shown; their use cases just sit in empty space, on their own. It is as though you were to draw an anonymous collaboration diagram, a

special sort of functional decomposition, that you are later supposed to explain with a proper collaboration diagram. According to the UML specification,

詳細說明某個模型元素的使用案例，接下來會修正成更細緻、一組比較小的使用案例，這些小使用案例中會詳述被包含在之前模型元素中的某個小模型元素所提供的服務。...然而，請注意，作為容器的元素其內部結構並沒有在使用案例中寫出來，因為使用案例中只會說明元素所提供的功能性。某個上層使用案例的一些下層使用案例會合作完成上層使用案例中所需做的工作。下層使用案例間的合作可用一些合作情形來說明，而且可秀在合作圖中。

A use case specifying one model element is then refined into a set of smaller use cases, each specifying a service of a model element contained in the first one. . . .

Note, though, that the structure of the container element is not revealed by the use cases, since they only specify the functionality offered by the elements. The subordinate use cases of a specific superordinate use case cooperate to perform the superordinate one. Their cooperation is specified by collaborations and may be presented in collaboration diagrams.

在使用案例規格書的附錄中寫出這些罕見關係的目的不是很清楚，而且我也不打算解釋它。之所以提到它們是因為我在本書中用到「子使用案例」這個術語，某些人可能會因此問我：「Cockburn 的子使用案例跟 UML 中的下層使用案例有什麼關聯嗎？」。

我用「子使用案例」這個術語代表目標等級比較低的目標。一般來說，目標等級比較高的使用案例會呼叫（包含）子使用案例。過去，我會把目標等級比較高或低的使用案例分別稱為「上層使用案例」或「下層使用案例」。不過，因為 UML 1.3 版已經用這些字，所以我只好改變自己的字彙。就我自己的經驗來說，大家不會對「呼叫別人的使用案例」跟「子使用案例（譯註：被呼叫的使用案例）」感到奇怪。就算是新的使用案例寫作人員或讀者也能很清楚了解它們。

The purpose of introducing these peculiar relations in the explanatory text of the use case specification is unclear, and I don't propose to explain them. I only bring up the matter because I use the term "sub use case" in this book, and someone will get around to asking, "What is the relation between Cockburn's sub use case and the UML subordinate use case?"

I use the term sub use case to refer to a goal at a lower goal level. In general, the higher-level use case will call (include) the sub use case. I used to say "subordinate" and "superordinate" for higher- and lower-level use cases, but since UML 1.3 has taken those words, I have shifted vocabulary. My experience is that people do not find anything odd about the terms "calling use case" and "sub use case." They are clear even to the novice writer and reader.

A.6 畫使用案例圖

如果你已定好並遵從一些簡單畫圖慣例的話，將可發現到自己跟讀者間的溝通是很容易的。請不要把讀者置於畫滿一層層箭頭、像補鼠網的圖中，卻希望他們能理解你想表達的東西。請看針對不同使用案例間關係所寫的寫作指引 13 到 16，它們對你會有些幫助。下面這兩個關於畫圖的寫作指引也對你有幫助。

↑ 英 242

You will find that the use case diagrams will communicate more easily to your readers if you set up and follow a few simple diagramming conventions. Please don't hand your readers a rat's nest of arrows and then expect them to trace out your meaning. Guidelines 13 through 16, for the different use case relations, will help. Two more drawing guidelines can help as well.

寫作指引 17：不要在情境圖中畫出目標等級

比使用者目標等級更低的目標

在主要的情境圖中，請不要秀出任何目標等級比使用者目標等級更低的使用案例。畢竟，這個圖的目的是作為設計系統的情境與使用案例目錄。如果你想用圖來分解使用案例的話，請把分解後的結果另外放在其他頁中。

On the main context diagram, do not show any use cases lower than user-goal level. After all, the purpose of the diagram is to provide context and a table of contents for the system being designed. If you decompose use cases in diagram form, put the decompositions on separate pages.

寫作指引 18：把支援性參與者放在圖的右邊

我發現把所有主要參與者放在代表系統的方塊左邊、剩下的支援性參與者（次要參與者）則放在右邊會很有幫助。這樣做可減少我們對主要參與者 v.s. 支援性參與者的困擾。某些人則從未在圖中畫出支援性參與者。這樣一來，他們就可把主要參與者任意放在圖的左右邊了。

I find it helpful to place all the primary actors on the left side of the system box, leaving the right side for the supporting (secondary) actors. This reduces confusion about primary versus secondary actors. Some people never draw supporting actors on their diagrams. This allows them to place primary actors on both sides.

A.7 用以文字為基礎的使用案例代替以圖形為主的_UC

如果你花太多時間研讀這些使用案例圖與其關係，並且感到很困擾的話，你就是把精力花在錯的地方了。在散文式使用案例中，使用案例間的關係是很直覺的，而且你將無法理解其他人為何始終忙於處理這些死結。

If you spend much time studying and worrying about the graphics and the relations, you are expending energy in the wrong place. Put it instead into writing easy-to-read prose. In prose, the relations between use cases are straightforward, and you won't understand why other people are getting tied up in knots about them.

「用以文字為基礎的使用案例代替以圖形為主的使用案例」是許多使用案例專家的一致看法。寫出下面的事例雖然有點自誇，不過我希望大家因此而能了解這個建議的重要性。本人特別感謝 IBM European Object Technology Practice 的 Bruce Anderson，他在 OOPSLA'98 中一場使用案例小組討論會上所發表的意見。在討論會上，有一系列問題是關於包含關係跟擴充關係間的差異，以及情節與橢圓形的數目爆炸的困擾。Bruce 表示他的團隊從未發生情節爆炸情形，而且也不會對包含關係跟擴充關係感到困擾。有人因此問他：為何其他人都關切「情節爆炸跟如何使用擴充關係」，可是他卻不會。Bruce 的答案是：「我只是照著 Alistair 所說的去做」，也就是：把時間多花在寫出清晰的說明文字上、遠離擴充關係，不要太擔心圖。

This is a view shared by many use case experts. It is somewhat self-serving to relate the following event, but I wish to emphasize the seriousness of the suggestion. My thanks to Bruce Anderson of IBM's European Object Technology Practice for the comment he made during a panel on use cases at OOPSLA '98, during which a series of questions surfaced around the difference between includes and extends and the trouble with the exploding number of scenarios and ellipses. Bruce stated that his groups don't run into scenario explosion, and they don't get confused. A questioner asked why everyone else was concerned about "scenario explosion and how to use extends" but he wasn't. Bruce's answer was "I just do what Alistair said to do," which is to spend time writing clear text, staying away from extends, and not worrying about diagrams.

寫得出好的、以文字為基礎的使用案例的人不會跟那些胡亂把心思放在 UML 中棒形人偶、橢圓形與箭頭的人一樣，遭遇那些問題。當你寫出展開後的敘事情節時，這些使用案例間的關係就會很自然；不過，當你老是把心思擺在使用案例間的關係上時，反而變成問題所在。如果有更多的顧問可同時對文字與 UML 都有

經驗，那麼其中一定有更多會同意這一點。

↑ 英 243

譯註：根據譯者個人理解，把焦點擺在說明文字上之後，就不容易遭遇那些問題是因爲：(1).只要我們稍加調整文字，就可把同樣內容變成任意一種使用案例間的關係。(2).說明文字中所隱藏的使用案例間關係不是很明顯，所以當我們選擇一種寫法來寫它時，其實是不會注意到是否有其他寫法存在，自然就不會煩惱該用哪一種使用案例間關係。等到我們寫完之後，再根據說明文字畫出使用案例圖就很自然了。

People who write good text-based use cases simply do not run into the problems encountered by people who fiddle with the stick figures, ellipses, and arrows of UML. The relations come naturally when you write an unfolding story; they become an issue only if you dwell on them. As more consultants gain experience in both text and UML, the more they agree on this.

附錄 B 部分習題解答

第 3 章

習題 3.1

我們有可能是正在描述自己的鄰近區域或整組以電子化方式連接在一起的銀行業。如果所描述的範圍比較小的話，可能是正在描述銀行建築物與照明系統。也有可能正在設計新的銀行電腦系統與 ATM，或者只是 ATM 而已。此外，我們也有可能是正在討論新的按鍵面板設計方式，或者新的 Enter 鍵設計方式。事實上，我們沒有辦法知道這個敘事情節片段中所討論的是哪一種系統。

習題 3.2

跟習題 3.1 一樣，由於我們無法判別使用者故事片段中所討論的是什麼系統，所以下面畫出各種可能的設計範圍。

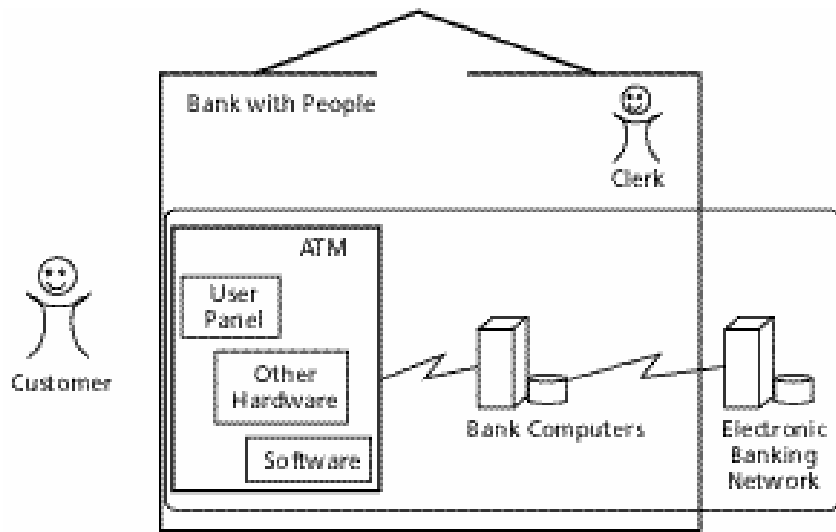


圖 B.1 針對 ATM 所畫出來的各種可能設計範圍

第 4 章

習題 4.2

請回想一下使用案例欄位的合格／不合格測驗。參與者必須要能執行具有行為價值的條件子句，而主要參與者則會有目標、呼叫系統承諾提供的服務。

ATM	SuD。
顧客	主要參與者與關係人。
ATM 卡	因為它沒有足夠的行為能力，所以不是參與者（它只是「無生命、鐵製、儲存檔案用的」卡片而已；具有嵌入式晶片的「智慧卡」比較有可能是參與者）。ATM 卡只是封裝資料用的一種裝置，我們把它當作快速輸入固定內容的客戶端部分。
銀行	就我們的目的而言，它不是參與者。它是包含 ATM 的系統。
ATM 的前方面板	就我們的目的而言，它不是參與者。它是 SuD 的元件之一。
銀行擁有者	關係人，它可能不是主要參與者。
服務人員	主要參與者。
印表機	就我們的目的而言，它不是參與者。它是 SuD 的元件之一。
銀行主電腦系統	次要參與者。如果你認為它在某個情況下會啟動跟 ATM 間的對話，那麼它就有可能是主要參與者。
銀行出納員	視工作分配而定。誰負責清空或補充 ATM 中的現金呢？如果答案是由「現金補充者」或「服務職員」負責的話，我們可能就不會寫出以銀行出納員為主要參與者的使用案例。如果答案是由「銀行出納員」負責的話，那麼他就是主要參與者。
搶銀行的強盜	視設計範圍跟你的想像而定。之前我並無法替搶銀行的強盜想出像樣的使用案例，認為它只是顧客使用案例中的一個擴充情況而已，後來有人建議「搶銀行！」這個使用案例，它可為我們帶來移動偵測器的概念。所以，結果要看如何寫出目標而定，我們可讓強盜擁

有這個使用案例（當然，這個目標永遠不會成功！），或者它只是顧客使用案例中的擴充情況而已。

習題 4.3

答案要看你所選擇的系統設計範圍為何而定（請參見圖 B.1）。

ATM	就我們的目的而言，它不是參與者。它是 SuD 的元件之一。
顧客	它依然還是主要參與者與關係人。
ATM 卡	它不是參與者，放棄理由跟習題 4.2 一樣。
銀行	請看圖 B.1。如果你選的系統設計範圍是「包含人的銀行」，那麼它就是你的 SuD。如果你所選的系統設計範圍是「電子化銀行網路」的話，它可能就是參與者（要看你是不是能從電子化網路銀行中找到它能呼叫的服務）。
ATM 的前方面板	就我們的目的而言，它不是參與者。它是 SuD 的元件之一。
銀行擁有者	要看你所選的系統設計範圍、你想出來的服務目標而定。結果可能是銀行的元件之一（因此，它不是主要參與者），或者是銀行的主要參與者。不過，它可能不會是電子化銀行系統的主要參與者。
服務人員	如果他是外聘的服務人員，那麼它就是主要參與者；如果他是銀行雇員，而且你所選的 SuD 是銀行，那麼他就是 SuD 的元件之一。
印表機	就我們的目的而言，它不是參與者。它是 SuD 的元件之一。
銀行主電腦系統	不論你是選銀行或電子化銀行系統當作 SuD，它都是 SuD 的元件之一。
銀行出納員	如果你所選的 SuD 是銀行，那麼他就是 SuD 的元件之一；如果你所選的 SuD 是電子化銀行系統，那麼他就是主要參與者。
搶銀行的強盜	答案跟習題 4.2 一樣。

第 5 章

習題 5.1

目標摘要等級（白色）：帶某人出外用餐☹（對 ATM 來說，這個目標有點做作）。

目標摘要等級（白色）：使用 ATM ^P。

使用者目標等級（藍色）：從 ATM 中提款 ^{ATM}。

子功能目標等級（靛藍色）：輸入 PIN 號碼 ^{ATM}。

子功能目標等級（黑色）：找尋 Enter 鍵 ^{ATM}。

習題 5.2

參與者	目標	等級
服務人員	把 ATM 設定就緒	目標摘要等級
	執行 ATM 的自我測試	使用者目標等級
銀行職員	重新補充現金	使用者目標等級
	重新填滿 ATM 中的（非現金）物品	使用者目標等級
顧客	使用 ATM	目標摘要等級
	提款	使用者目標等級
	存款	使用者目標等級
	轉帳	使用者目標等級
	查詢存款餘額	使用者目標等級

第 6 章

習題 6.1

想找到最小事後保證的最簡單方式是問：「什麼東西會讓關係人不高興呢？」關係人有可能是顧客、銀行或銀行監督機構。

如果顧客沒有拿到現金，他們會不高興，不過這不是我們所預期的最小事後保證。先讓我們假設他們沒有拿到錢。在這個情況下，如果這筆交易有被記錄下來，而且他們的帳號中有扣掉這筆錢，他們才會不高興。事實上，當他們被扣掉的錢超過他們所領到的錢時，他們都會不高興。此外，他們也希望所有交易動作在系統中都會有紀錄，這樣一來，他們才能保護自己不會被騙。

另一方面，如果顧客領到的錢比他被扣掉的錢還多，這時候，銀行會不高興。它也希望有一份系統紀錄可保護自己，或許是一種特殊紀錄，裡面說明發生災難性錯誤情況時，交易動作進行到什麼程度，讓我們可從其中挑出任何錯誤情況。至於監督機構則希望看到所有指引都有被遵守，所以最好所有交易動作都有系統紀錄。

最後，我們得到的最小事後保證是：被扣的金額等於所送出去的金額，而且系統還會做細部紀錄，說明災難性失敗情況發生時，交易動作進行到什麼程度。此外，每個交易動作都有被系統記錄下來。

習題 6.4

提款的成功事後保證是：被扣的金額是 ATM 所送出去的金額（而不是顧客所要求的金額 — 原因請看失敗條件！）、ATM 卡有被歸還、機器回到最初等待使用者進行交易的狀態，而且交易動作有被記錄下來。

第 7 章

習題 7.1

下面是以描寫使用者介面細節的方式所寫出來的從 *ATM 提款* 使用案例中的主要成功情節。如果把像這樣的使用案例給別人看，相信讀者都會不太高興。用意圖方式所寫出來的使用案例，請參見習題 7.2。

1. 顧客拿 ATM 卡在讀卡機上刷過去。
2. ATM 從卡上讀出銀行 ID、帳號。
3. ATM 詢問顧客，接下來要用西班牙語或英語來顯示畫面。
4. 顧客選擇英語。
5. ATM 要求顧客輸入 PIN 號碼，並按下 Enter 鍵。
6. 顧客輸入 PIN 號碼，然後按下 Enter 鍵。
7. ATM 秀出顧客可執行的活動清單。
8. 顧客選擇「提款」。
9. ATM 要求顧客以 \$5 的倍數輸入要提多少錢，並按下 Enter 鍵。
10. 顧客以 \$5 的倍數輸入金額，然後按下 Enter 鍵。
11. ATM 通知顧客帳號的銀行主系統、告訴它提款金額。
12. 銀行主系統接受這個提款交易動作，然後告訴 ATM 帳戶的新餘額。
13. ATM 送出現金。
14. ATM 詢問顧客是否要列印收據。
15. 顧客回答要印。

16. ATM 送出有印帳戶新餘額的收據。
17. ATM 把這筆交易動作記錄下來。

習題 7.2

下面是以比較有效率方式所寫出來的快速提款版本，裡面只會寫出參與者的意圖，而沒有寫出使用者介面細節。

1. 顧客拿 ATM 卡在讀卡機上刷過去。
2. ATM 從卡上讀出銀行 ID、帳號、加密過的 PIN 號碼，並且跟主銀行系統證實銀行 ID 與帳號是正確的。
3. 顧客輸入 PIN 號碼。ATM 證實輸入的 PIN 號碼跟從卡片讀出來、加密過的 PIN 號碼是一樣的。
4. 顧客選擇快速提款與提款金額，金額必須是\$5 的倍數。
5. ATM 通知顧客帳號的銀行主系統、告訴它提款金額，然後收到確認訊息與新餘額。
6. ATM 送出現金、卡片並列印新餘額的收據。
7. ATM 把這筆交易動作記錄下來。

習題 7.4

這段使用案例中有三種錯誤存在。第一種錯誤是：除了使用案例名稱與簡介之外，裡面所提到的許多東西都跟登入無關，反而跟訂單處理系統有關。所以事實上，它是一個風箏等級的目標摘要等級使用案例。前六個動作步驟都是跟登入有關的事，不過它們的目標等級跟其他動作步驟完全不同，所以應該把它們獨立開來。一旦我們這麼做之後，就會發現登入後的使用者從未做出登出系統動作！

使用者還沒有選擇離開之前，會重複執行下列功能、分支決策結束與迴圈結束都是程式設計師寫程式結構的寫法，對於審查使用案例的人來說是沒有意義的。另外，*條件子句*把整個使用案例弄得很亂。還有，這些動作步驟都是以描寫使用者介面設計方式所寫的。上述幾點都應該改進。

至於**當...時，這個使用案例就會開始與使用案例結束**則是其他老師所建議的寫作慣例。它們沒有什麼特別不對的地方，只不過有裝飾作用而已，我個人覺得不需要寫出來，因為大部分人都會假設使用案例會從第一個動作步驟開始執行，而在主要成功情節的最後一個動作步驟結束。

另一個值得注意的寫法是：**使用者...，就「使用」下訂單**。句子中的「使用」相當於 UML 中的**包含關係**（之前的正式名稱是**使用關係**！）。我發現這種寫法會讓使用案例看起來有點不舒服，所以自己比較喜歡這樣寫成**使用者...，就下訂單**。你可根據自己專案團隊所定、用來表示參考其他使用案例的寫作慣例來寫。最後，我們發現原來的使用案例需要分開變成兩個使用案例：風箏等級的使用案

例使用訂單處理系統，以及子功能目標等級的使用案例登入。你可自行寫出登入使用案例。請注意，我們會把連到其他使用案例的地方加上底線。

使用案例 38 使用訂單處理系統

主要成功情節：

1. 使用者登入系統。
2. 系統秀出可選擇的功能。使用者選擇其中一種功能執行它：
 - 下訂單
 - 取消訂單
 - 查詢訂單狀態
 - 送外型錄
 - 登錄顧客抱怨情形
 - 產生銷售報表
3. 使用者會不斷執行功能，直到他選擇離開為止。
4. 當使用者選擇離開時，系統會把他登出。

第 8 章

習題 8.1

下面是使用 ATM 時可能發生的失敗情況。一般來說，我班上的學生都可列出比這個長兩倍的失敗情況清單。請注意，所有失敗情況都要是系統可偵測到、處理的。你的清單呢？

- 讀卡機壞了或 ATM 卡上有刮痕。
- 沒有提供通儲銀行的 ATM 卡。
- PIN 號碼輸入錯誤。
- 顧客沒有及時輸入 PIN 號碼。
- ATM 當機。
- 主機當機或網路斷線。
- 帳戶餘額不足。
- 顧客沒有即時輸入金額。
- 所輸入金額不是 \$5 的倍數。
- 要求提款金額超過 ATM 每次可提款金額。
- 進行交易時，網路斷線或主機當機。
- ATM 中的現金不足。
- 發錢時卡住了。

收據紙用完或卡紙。
顧客沒有從提錢口拿錢。

習題 8.5

使用案例 39 在網路上買股票

主要參與者：購買者

設計範圍：個人化的顧問／理財套件（PAF）

目標等級：使用者目標等級

關係人與利益：

購買者 — 希望買股票，並且自動把它們加到 PAF 的投資組合中。

股票經紀商 — 希望完整的購買資訊。

事先條件：使用者已經有開啓的 PAF。

最小事後保證：要有足夠的登入資訊，當 PAF 偵測到錯誤情況發生時，會告知使用者提供一些細節（譯註：根據 *UML 與樣式徹底研究第二版【Applying UML and Patterns 2nd】* 一書的建議，我們這裡會用宣告式、被動式、過去式的時態（被...）寫出事後保證，強調我們是在宣告狀態的變化而不是設計如何達成這些變化）。

成功事後保證：被遠端網站告知這筆購買已經成功；歷史紀錄與使用者的投資組合都會被更新。

主要成功情節：

1. 購買者在網路上選擇想買的股票。
2. PAF 從使用者得到他想用的網站名稱（E*Trade、Schwab 等）。
3. PAF 開啓跟這個網站之間的網路連線，保持控制權。
4. 購買者瀏覽網站買股票。
5. PAF 攔截網站回應並更新購買者的投資組合。
6. PAF 排列新的投資組合、秀給使用者看。

擴充情節：

- 2a. 購買者想要進入 PAF 沒有支援的網站：
 - 2a1. 系統給購買者一個新建議，使用者可選擇要不要取消這個使用案例。
- 3a. 在建立網路連線時，發生任何的網站失敗情形：
 - 3a1. 系統回報失敗情形，並且給購買者一些建議，看看是否要回到上一步。
 - 3a2. 購買者可回到這個使用案例的上一步或再連線一次。
- 4a. 在購買交易動作過程中，電腦當機或被關機：
 - 4a1. （我們這時候能做些什麼？）（譯註：其實這裡可寫一些重開機時的應對措施）。

- 4b. 網站沒有告知交易是否成功，不過這筆交易動作會被延遲處理：
- 4b1. PAF 把延遲處理登錄到歷史紀錄中，並且設定計時器，到時候再跟購買者詢問交易結果。
- 5a. 網站沒有傳回交易的必要資訊：
- 5a1. PAF 在歷史紀錄中登錄缺乏資訊，要求購買者更新有問題的購買交易動作。
- 5b. 更新投資組合動作時，磁碟毀損或滿了：
- 5b1. 重新啓動 PAF 時，會偵測到發生不一致狀況的系統紀錄，並要求使用者更新有問題的購買情形。

第 11 章

習題 11.1

使用案例 40 執行清潔火星塞服務

事先條件：車子在修車庫、引擎發動中。

最小事後保證：顧客被告知引擎有更大的問題在；車子沒有修。

成功事後保證：引擎運作順暢。

主要成功情節：

1. 打開車蓋、在防護板上塗上保護塗料。
2. 把火星塞拔起來。
3. 清潔火星塞。
4. 清潔火星塞上的點火口並調整點火距離。
5. 測試並證實火星塞可用。
6. 裝上火星塞。
7. 把點火線跟火星塞連接起來。
8. 檢查並證實引擎可正常運作。
9. 清理維修工具與裝備。
10. 清潔防護板上的保護塗料與車上的任何油污。

擴充情節：

- 4a. 火星塞爆裂或磨損了：更換新的火星塞。
- 8a. 引擎還是無法正常運作：
 - 8a1. 概略診斷引擎 (UC 23)。
 - 8a2. 告知顧客引擎有更大的問題存在 (UC 41)。

附錄 C 字彙表

主要術語

參與者：具有行爲（行爲中可能有條件子句）的某個東西。它可能代表機械系統、電腦系統、人、組織或這些東西的組合。

外部參與者 (external actor) 代表討論中系統之外的參與者。

關係人 (stakeholder) 也是外部參與者，不過系統會保障它的利益，而且系統必須做某些特定的動作來滿足它的利益。不同使用案例的關係人可以不同。

主要參與者 (primary actor) 代表會要求系統達成某個目標的關係人。一般來說（非必要），主要參與者會啟動跟系統間的互動情形。主要參與者可由其他中介者啟動互動情形或由事件自動觸發互動情形。

支援性參與者 (supporting actor) 或 **次要參與者 (secondary actor)** 代表 SuD 對它有目標的其他系統。

幕後參與者 (offstage actor) 或 **第三級參與者 (tertiary actor)** 代表非主要參與者的關係人。

內部參與者 (internal actor) 代表討論中系統（SuD）或 SuD 的子系統，或者代表 SuD 中的主動元件。

互動情形：代表一段訊息、一連串互動情形，或者是一整組互動序列。

情節：代表某些特定條件下發生、由動作與互動情形所構成的一連串序列，我們不會在裡面寫出條件子句或分支情形。

具體情節 (concrete scenario) 代表所有細節（相關參與者名稱與資料值）都會具體寫出來的情節。這時候，我們會用過去式來描述敘事情節，並寫出其中所有相關細節。

在需求的寫作過程中，針對參與者與資料值，有時候我們會用一些像「顧客」或「地址」的代名詞術語寫出情節，這樣的情節稱爲 **一般性情節 (general scenario)**。有時候，我們必須把它跟具體情節做一個區隔。

系統使用情形簡述 (usage narrative or narrative) 代表一段具體情節，裡面寫出各種參與者的動機與意圖。我們拿它作爲閱讀或寫出使用案例的暖身活動。

走完某個使用案例的 **路徑 (path or course)**，它是一般性情節的同義字。

主要成功情節 (main success scenario) 代表從觸發事件開始到結束、完整寫出來的情節，裡面包含目標達成情形，以及在後面發生的任何系統記錄情形。縱然它可能不是唯一成功路徑，不過它是典型、可供示範的一個成功情節。

替代路徑 (alternate course) 代表任何會擴充主要成功情節的其他情節或情節片段。

動作步驟 (action step) 代表寫情節時的基本單位。典型的動作步驟是一個句子，裡面通常會描述某個參與者的行為。

擴充情節：在其他情節中、某個特殊條件下開始執行的情節片段。

擴充情況 (extension condition) 代表發生不同行為的環境。

擴充使用案例 (extension use case) 是在某個特定條件下，會中斷其他使用案例的使用案例。我們把被中斷掉的使用案例稱為**基本使用案例 (base use case)**。

擴充點 (extension point) 是一個標籤或暱稱，它代表基本使用案例中的行為會被擴充使用案例中斷掉的地方。事實上，我們可用擴充點來代表基本使用案例中的一組位置，這樣一來，擴充使用案例就可針對整組條件，蒐集所有會中斷基本使用案例的相關擴充行為。

子使用案例 (sub use case) 代表會被某個情節中的動作步驟所呼叫到的使用案例。在 UML 中，我們會說：呼叫的使用案例中**包含**了子使用案例的行為。

使用案例：使用案例中會表達系統的關係人之間的合約行為部分，裡面描述各種條件下的系統行為與互動情形，以回應關係人之一（**主要參與者**）的請求，而且會秀出主要參與者的目標是如何達成或失敗的。使用案例中蒐集了跟主要參與者的目標有關的情節。

使用案例的分類方式

根據焦點分類：把焦點放在企業或新系統上。

企業使用案例 (business use case) 代表這樣的使用案例中會強調企業業務，而非電腦系統的操作情形。我們可能用企業使用案例寫出任何目標等級的使用案例，不過它的設計範圍只能是企業或組織。

系統使用案例 (system use case) 代表這樣的使用案例中會強調電腦或機械系統的操作情形，而非企業業務。我們可能用系統使用案例寫出任何目標等級、任何設計範圍的使用案例，其中包括企業設計範圍。寫出設計範圍為企業的系統使用案例是為了要強調企業行為下 SuD 的用處為何。

根據正式程度分類：要花多少精力，或者所使用的精確度與正式性有多少。

使用案例簡介 (use case brief) 是用一個段落來說明的使用案例摘要。

非正式使用案例 (casual use case) 是用簡單、散文式的段落寫成。它有可能會漏掉跟使用案例相關的一些專案資訊。跟正式使用案例比起來，它的描述也比較不嚴謹。

正式使用案例 (fully dressed use case) 是根據完整範本所寫成的，裡面會寫出參與者、設計範圍、目標等級、驅動條件、事先條件，以及其他範本中

表頭方面的資訊，此外還會加上專案說明資訊。

根據目標等級分類：使用案例的目標等級高低。

目標摘要等級使用案例 (summary-level use case) 中會包含多個使用者目標執行期間，完成這種目標等級的使用案例可能要花幾星期、幾月，甚至是幾年。它的子使用案例可以是任何目標等級的使用案例。我們用白雲 (☁) 或風箏 (🪁) 圖形來標示它。白雲等級使用案例的動作步驟可能是白雲或風箏等級。風箏等級使用案例的動作步驟則是使用者目標等級。

使用者目標等級使用案例 (user-goal use case) 會滿足對主要參與者來說具有價值的特定或立即目標。一般來說，這樣的使用案例會由主要參與者在 2 到 20 分鐘內執行完畢 (如果主要參與者是電腦的話，時間會更短)，之後系統還可執行其他事情。它的動作步驟可能屬於使用者目標等級或更低的目標等級。我們用海平面 (🌊) 圖示來標示它。

子功能目標等級使用案例 (subfunction use case) 會滿足「使用者目標等級使用案例或其他子功能」中的一部分目標；它的動作步驟是目標等級更低的子功能。我們用魚 (🐟) 或蚌 (🐚) 圖形來標示它。

根據設計範圍分類：使用案例的 SuD 有多大。

企業設計範圍 (enterprise scope) 代表 SuD 是組織或企業。我們會在使用案例範本中的設計範圍欄位寫出組織或企業名稱。根據它是黑箱式或白箱式的使用案例，我們會用灰色建築物 (🏢) 或白色建築物 (🏠) 圖形來標示它。

系統設計範圍 (system scope) 代表 SuD 是機械/硬體/軟體系統或應用程式。我們會在使用案例範本中的設計範圍欄位寫出系統名稱。根據它是黑箱式或白箱式的使用案例，我們會用灰色箱子 (📦) 或白色箱子 (📦) 圖形來標示它。

子系統設計範圍 (subsystem scope) 代表 SuD 是應用程式中的一部分，它可能是子系統或框架。我們會在使用案例範本中的設計範圍欄位寫出子系統名稱。我們會用螺絲 (🔩) 圖形來標示它。

根據可見性分類：在使用案例中是否看得見個體。

黑箱式使用案例 (black-box use case) 裡面不會提到 SuD 中的任何元件。它通常用在系統需求文件上。

白箱式使用案例 (white-box use case) 裡面會提到 SuD 中的元件行爲。它通常用在建立企業模型上。

跟使用案例相關的一些圖

合作圖：在 UML 中，這種圖用跟循序圖不同的形式秀出相同資訊。我們會把參與者散放在圖的各處，而互動情形則是參與者間、編號過的箭頭 (譯註：箭頭上會有參與者間的往來訊息)。時間先後順序只能用箭頭編號來表現。

循序圖：在 UML 中，這種圖會把參與者秀在最上方、各自佔據一欄空間，而互動情形會秀在欄與欄間的箭頭上，時間先後順序則是順著頁面往下。它是以圖形方式秀出某個情節的好方法（譯註：不論是合作圖或循序圖，每張圖中最好只秀出一個情節）。

使用案例圖：在 UML 中，這種圖可秀出外部參與者、系統邊界。使用案例會被畫成橢圓形，而且會用箭頭從參與者連到橢圓形或從某個橢圓形連到另一個橢圓形。它主要用來當作情境圖與使用案例目錄。

附錄 D 參考資料

參考書籍

- Beck, K., *極致軟體製程 (Extreme Programming Explained)*. Reading, MA: Addison-Wesley, 2000.
- Cockburn, Alistair. *Surviving Object-Oriented Projects*. Reading, MA: Addison-Wesley, 1998.
- Cockburn, Alistair. *Software Development as a Cooperative Game*. Boston: Addison-Wesley (due 2001).
- Constantine, Larry, and Lucy Lockwood. *Software for Use*. Reading, MA: Addison-Wesley, 1999.
- Fowler, Martin. *UML精華第二版 (UML Distilled)*. Reading, MA: Addison-Wesley, 1999.
- Hammer, Michael, and James Champy. *改造企業 (Reengineering the Corporation)*, Reprint Edition. New York: HarperBusiness, 1994.
- Hohmann, Luke. *GUIs with Glue* (in preparation as of July 2000).
- Roberson, Suzanne, and James Robertson. *Mastering the Requirements Process*. Reading, MA: Addison-Wesley, 1999.
- Wirfs-Brock, Rebecca., Wilkerson, Brian, and Wiener, Lauren. *Designing Object-Oriented Software*. Upper Saddle River, NJ: Prentice-Hall, 1990.

參考文章

- Beck, Kent, and Ward Cunningham. "A laboratory for Object-Oriented Thinking", *ACM SIGPLAN* 24(10):1-7, 1989.
- Cockburn, Alistair. "VW-Staging," at <http://Alistair.Cockburn.us/papers/vwstage.htm>
- Cockburn, Alistair. "An Open Letter to Newcomers to OO," at <http://members.aol.com/humansandt/papers/oonewcomers.htm>
- Cockburn, Alistair. "CRC Cards," at <http://members.aol.com/humansandt/papers/crc.htm>
- Cunningham, Ward. CRC Cards," at <http://c2.com/cgi/wiki?CrcCards>
- Kraus, Andy, and Michael Dillon. "Use Case Blue," *Object Magazine*, SIGS Publications, May 1996.
- Lilly, Susan. "How to Avoid Use Case Pitfalls," *Software Development* 8(1):40-44, 2000.
- McBreen, Peter. "Test cases from use cases," at <http://www.mcbreen.ab.ca/papers/TestsFromUseCases.html>

有用的網路資源

網路中包含了大量資訊。下面可作為你找尋這些有用資訊的起點：

<http://www.usecases.org>

<http://Alistair.Cockburn.us>

<http://www.foruse.com>

<http://www.pols.co.uk/usecasezone/>

英文索引

- ! (user-goal use cases 使用者目標等級使用案例), 3-4, 6-7, 9-11, 62-64
- * extensions 擴充情節中用來表示任何時候會發生的動作步驟, 103
- + (summary use cases 目標摘要等級使用案例), 3, 7, 62-67, 142, 144
- (subfunctions 子功能目標等級), 62-63, 66-67, 69, 142
- : extensions 擴充情節中的擴充條件會以分號結尾, 103

A

Aas, Torfinn, 6

Accuracy of use cases 使用案例的各種精確度, 17

Action steps 動作步驟, 90-98. See also Scenarios 請參見情節

- bird's eye view for 用鳥瞰觀點來寫動作步驟(Guideline 寫作指引 3), 91, 217
- do until condition 重複執行動作步驟 x-y 直到...為止(Guideline 寫作指引 10), 96-97
- exercises 相關習題, 98, 249-250
- extensions 擴充情節中的動作步驟, 99-100
- forward movement of 每個動作步驟對目標達成都要有所進展(Guideline 寫作指引 4), 91-92
- grammar (simple) for 用簡單句寫出動作步驟(Guideline 寫作指引 1), 90
- intentions of actors 寫出參與者的意圖，而非動作(Guideline 寫作指引 5), 92-93
- interface detail description 在動作步驟中寫出使用者介面詳細描述, 92
- numbering 替動作步驟編號, 97, 218
- reasonable set of 在動作步驟中放入「合理的」一組動作(Guideline 寫作指引 6), 93-95
- repeating steps 重複執行動作步驟 x-y 直到...為止, 96-97
- scenarios 用動作步驟寫出情節, 88
- sentence form for 寫出動作步驟時只用一種句型 (Reminder 寫作提示 3), 206-207
- systems interaction 系統間的互動情形：使用者要系統 A 跟系統 B 做某件事 (Guideline 寫作指引 9), 96
- timing 選擇性提到動作步驟的執行時間(Guideline 寫作指引 8), 95-96
- validation versus checking 用「證實」的語調來寫，而不要用「檢查某個東西是否有成立」的語調來寫(Guideline 寫作指引 7), 95

"Who has the ball?" 球在誰的手上(Guideline 寫作指引 2, Reminder 寫作提示 5), 90-91, 207

Actor-goal lists 參與者 — 目標清單

- use case diagrams versus 使用案例圖 v.s.參與者 — 目標清單, 218
- scope from 可以由參與者 — 目標清單定出功能範圍, 36-37, 51

Actor profile table 參與者描述表, 58

Actors 參與者, 54-60. See also Primary actors; Stakeholders; Supporting actors; System under discussion 請參見主要參與者、關係人、支援性參與者、討論中系統

- aliases of 參與者的別名, 58
- design and primary actors 設計時主要參與者的重要性, 56-57
- exercises 相關習題, 60, 246-247
- goals conceptual model, xix 參與者與目標概念性模型, 23-29
- internal actors and white-box use cases 內部參與者與白箱式使用案例, 59-60
- offstage (tertiary, silent) actors 幕後 (第三級) 參與者, 30, 53-54
- precision 使用案例的各種精確度, 17
- roles 由參與者扮演角色(Reminder 寫作提示 23), 57-58, 226
- system delivery and primary actors 準備交付系統時主要參與者的重要性, 57
- system under discussion (SuD) 討論中系統也是一種參與者, 59
- triggers 使用案例的驅動者, 54-55
- ultimate primary actors 使用案例的最終主要參與者, 54-55

Unified Modeling Language (UML) 統一模型語言中所提供的參與者間特殊化關係, 58

- use case production and primary actors 剛開始產生使用案例時主要參與者的重要性, 55-56
- use case writing and primary actors 寫使用案例時主要參與者的重要性, 56-57
- white-box use cases and internal actors 白箱式使用案例與內部參與者, 59-60

Adding value with use cases 什麼時候使用案例會更有價值, 15-16

Adolph, Steve, 11-12, 133

Agreement on use cases for completion 認同使用案例中所寫出來的東西, 142

Aliases of actors 參與者的別名, 58

Alternate flows (extensions) 在 RUP 中, 擴充情節稱為替代流程, 123

Alternative paths 替代路徑, 217

Anderson, Bruce, 243

Arrow shapes in UML 對 UML 中的各種使用案例關係, 請用不同的箭頭形狀 (Guideline 寫作指引 15), 236-237

Asterisk (*) for extensions 擴充情節中用來表示任何時候會發生的動作步驟, 103

Atlantic Systems Guild Atlantic 系統指引, 13

B

Bear, Kerry, 70

Beck, Kent, 167, 187, 223-224

Behavior 行爲部分. See Contract for behavior。請參見合約的行爲部分

Bird's eye view 用鳥瞰觀點來寫動作步驟(Guideline 寫作指引 3), 91, 217

Black-box requirements 黑箱式需求, 217

Black-box use cases (grey) 黑箱式使用案例(用灰色圖示來標示), 4-7, 9-11, 40-41

Black/indigo, underwater fish/clam graphic, minus sign (subfunctions) 黑色/深靛藍色、水中魚/蚌圖示、減號(子功能目標等級), 3, 7, 62-63, 66-67, 69, 142

Blue, sea-level waves graphic, exclamation mark (user-goal use cases) 藍色、海平面波浪圖示、驚嘆號(使用者目標等級使用案例), 3-4, 6-7, 9-11, 62-64

Body of scenarios 情節的主體, 89

Bolt graphic (component use cases) 螺絲圖示(元件使用案例), 41, 46-49

Bouzide, Paul, 38

Box graphic, grey/white (system use cases) 箱子圖示、灰色/白色(系統使用案例), 3-7, 9-11, 41, 157-159, 216

Brainstorming 腦力激盪方式

Extensions 用腦力激盪法寫出擴充情節, 101-104, 110

use cases for 用腦力激盪法寫出使用案例中的內容, 12(新需求), 16(失敗情節)

Bramble, Paul, 128

Branch-and-join process for project planning 專案規劃時, 採用個人與團體並重的使用案例寫作過程, 180-183

Building graphic, grey/white (business use cases) 建築物圖示、灰色/白色(企業使用案例), 3, 7, 41

Business 企業

priority, scope 寫出功能範圍時可能會列出的欄位: 企業優先順序、設計範圍, 36

process to technology 從(中間的)企業流程開始往技術層面思考, 155-157

rules discovered by extensions 寫擴充情節時所發現到的企業規則, 100

setting and formats 企業環境對使用案例格式的影響, 129

system use cases versus 公司設計範圍跟系統設計範圍之比較 (Reminder 寫作提示 13), 216

use cases (building graphic, grey/white) 企業使用案例(建築物圖示、灰色/白色), 3, 7, 41

Business process modeling 建立企業流程模型, 153-160
business process to technology 從（中間的）企業流程開始往技術層面思考, 155-157
core business, working from 從核心業務開始思考, 154-155
designing versus modeling 進行設計 v.s. 建立模型, 153-157
external primary actors 外部主要參與者, 154
linking business and system use cases 把企業使用案例跟系統使用案例連接起來, 157-159
modeling versus designing 建立模型 v.s. 進行設計, 153-157
prior to use cases 寫出使用案例前先建立企業模型, 218
services 建立企業流程模型後，將可知道人企業所提供的服務, 154
stakeholders 建立企業流程模型後，將可知道跟組織行為有關的關係人, 154
standard 建立企業流程模型時用的使用案例範本標準, 132, 134
system use cases, linking to business 把企業使用案例跟系統使用案例連接起來, 157-159
technology to business process 直接從技術往回看企業流程, 157
triggers 建立企業流程模型後，將可知道組織必須回應的驅動事件, 154
usage experts 系統使用情形專家, 156-157
use case examples 跟企業有關的使用案例範例, 153

C

(Use) Case diagrams versus actor-goal lists 使用案例圖 v.s. 參與者 — 目標清單, 218
CASE tools 電腦輔助軟體工程工具, 127, 227, 230
Casual use cases 非正式格式的使用案例, 7-9, 97, 120, 218
Central Bank of Norway 挪威央行, 6
Chrysler Comprehensive Compensation Chrysler 公司的綜合津貼專案, 223
Clam graphic, indigo/black, minus sign (subfunctions) 蚌圖示、靛藍色／黑色、減號（子功能目標等級）, 3, 7, 62-63, 66-67, 69, 142
Cloud/kite graphic, white, plus sign (summary use cases) 白雲／風箏圖示、白色、加號（目標摘要等級使用案例）, 3, 7, 62-67, 142, 144
Clusters of use cases 把使用案例分群, 143-144
Collaboration diagrams (UML) versus white-box (use) cases UML 中的合作圖 v.s. 白箱式使用案例, 218
Colaizzi, John, 102, 145
Collecting scenarios 把一些情節集合在一起，以形成使用案例, 27-29

use cases from large groups 大型團體中的使用案例蒐集方式, 184-186

Colon (:) for extensions 擴充情節中的擴充條件會以分號結尾, 103

Completeness and formats 完整性對使用案例格式的影響, 131

Completion of use cases 怎樣才算寫完使用案例, 141-142. See also Project planning
請參見專案規劃

Complexity and formats 複雜度對使用案例格式的影響, 130-131

Component use cases (bolt graphic) 元件使用案例 (螺絲圖示), 41, 46-49

Compound interactions 互動情形可以是合成的, 25-27

Conditions 情況或條件

- Extensions 擴充情節中的擴充情況, 99-106
- failure conditions (fourth work step) 失敗情況 (依照不同精確度寫出使用案例時的第四個工作步驟), 16-17, 222
- preconditions 使用案例的事先條件(Reminder 寫作提示 10), 2, 81-83, 211
- scenarios 情節的開始執行條件, 88

Conflict and formats 顧客不同需求間衝突對使用案例格式的影響, 131

Consistency and formats 一致性對使用案例格式的影響, 130

Constantine, Larry, 58, 92, 122, 163, 177

Content and purpose misalignment 寫作目的跟寫作內容不一致, 193

Context diagrams 作為情境圖的使用案例格式 (譯註: 最上層使用案例圖), 128, 227-228

Contract for behavior 合約的行為部分, 23-33. See also Action steps; Goal levels; Scenarios; Reminders 請參見動作步驟、目標等級、情節、寫作提示

- actors and goals conceptual model, xix 參與者與目標概念性模型, 23-29
- compound interactions 互動情形可以是合成的, 25-27
- ever-unfolding story 可不斷展開的敘事情節(Reminder 寫作提示 12), 26, 62, 215
- failure scenario 失敗情節, 31
- goal failures and responses 目標失敗與回應方式, 25
- graphical model 圖形模型, 31-33, 229
- interaction between two actors 兩個參與者間所發生的互動情形, 31
- interactions, compound 互動情形可以是合成的, 25-27
- internal state change 系統內部狀態變動, 31
- main success scenarios (third work step) 主要成功情節 (依照不同精確度寫出使用案例時的第四個工作步驟), 3, 17, 28, 87-89, 222
- offstage actors 幕後 (第三級) 參與者, 30, 53-54
- partial ordering 部份有順序性的動作序列, 26
- primary actors 主要參與者, 23, 27, 30-31
- scenario collection 把一些情節集合在一起, 以形成使用案例, 27-29

scenarios 代表合成互動情形的情節, 25
sequences of interactions 構成互動情形的一連串動作步驟, 25-27
sets of possible sequences 未來可能會發生的一組動作序列, 26-27
stakeholders and interests conceptual model 關係人與利益概念性模型, 29-31
striped trousers image 條紋褲, 27-29
subgoals 爲了達成使用案例目標（合約的行爲部分），系統必須有系統地達到一些子目標, 23-24
supporting actors 主要參與者要支援性參與者達成的子目標，是爲了滿足合約的行爲部分, 23-24
system under discussion (SuD)討論中系統, 24-25, 29
Unified Modeling Language (UML) 用 UML 中的使用案例寫出合約的行爲部分, 31
validation to protect stakeholders 情節中所寫出來的證實性動作（以保障關係人）, 31
Conversations, formats 採用對話概念的使用案例格式, 122
Coppolo, Dawn, 70
Core business, working from 從核心業務開始思考, 154-155
Core values and variations 使用案例格式的核心價值與替代性價值(Reminder 寫作提示 14), 216-219
Coverage and formats 涵蓋範圍對使用案例格式的影響, 130
Create, Retrieve, Update, Delete (CRUD) use cases 新增、讀取、更新或刪除 OO 這類型小使用案例, 145-150
Cross-linking from use cases to missing requirements 從使用案例連到其他需求, 164-165
Cultures and formats 文化衝突對使用案例格式的影響, 129
Curran, Eileen, 70

D

Data 資料

field details and checks (seventh work steps) 資料欄位細節與資料欄位檢查（依照不同精確度寫出使用案例時的第七個工作步驟）, 223
requirement precision 資料需求精確度, 162-164
and technology variations 技術與資料變異情形, 111-112
Delivery and scenarios 每次實作出完整的情節, 170
Design 設計
modeling versus 建立模型 v.s. 進行設計, 153-157

primary actors and 設計時主要參與者的重要性, 56-57
project planning 專案規劃時，從使用案例對應到設計工作清單, 171-174
scenarios and 利用情節來進行設計工作, 177
use cases for 專案規劃時，從使用案例到設計, 174-177
Design scope 設計範圍. See Scope 請參見專案範圍
Detailed functional requirements standard 用來描寫詳細功能需求的使用案例格式標準, 132, 137
Development 開發
 complexity, scope 寫出功能範圍時可能會增列的欄位：開發複雜度, 36
 priority, scope 寫出功能範圍時可能會增列的欄位：開發優先順序, 37
 team for scaling up 爲了擴大規模處理大量的使用案例，根據開發團隊替使用案例分群, 144
Diagram style 用圖形呈現出來的使用案例寫作風格, 127
Dive-and-surface approach 浮潛式方案, 12
Do until condition 重複執行動作步驟 x-y 直到...爲止(Guideline 寫作指引 10), 96-97
Documenting requirements from use cases 使用案例中所記錄下來的需求, 12. See also Requirements 請參見需求
Domain concepts from use cases 使用案例可點出領域概念, 177
DOORS, 230
Drawing use case diagrams, UML 畫出 UML 中的使用案例圖, 242-243

E

Elementary business process 基本企業流程（譯註：相當於使用者目標）, 68
Ellipses and stick figures, UML UML 中的橢圓形與簡單棒形人偶圖示, 233-234
Emphasizing extensions 強調擴充情節的方式, 103
Empower IT, 145
End condition, scenarios 情節的結束情況, 88
Endings (two) of use cases 使用案例的兩種結尾(Reminder 寫作提示 8), 209-210
Energy management 管理自己的精力, 16-17, 217, 221-223
Enterprise-to-system scope 從企業設計範圍到系統設計範圍, 41-42
Essential use cases 本質性使用案例, 122
Evans, Eric, 70
Ever-unfolding story 可不斷展開的敘事情節(Reminder 寫作提示 12), 26, 62, 215
Exclamation mark (!) user-goal use cases 用驚嘆號（！）來代表使用者目標使用案例, 3-4, 6-7, 9-11, 62-64

Experience and formats 經驗對使用案例格式的影響, 130

Extend relations, UML UML 中的擴充關係, 235-238

Extension points, UML UML 中的擴充點, 237-238

Extensions 擴充情節, 99-110. See also Scenarios 請參見情節

- action steps 擴充情節中的動作步驟, 99-100
- asterisk (*) for 擴充情節中用來表示任何時候會發生的動作步驟, 103
- brainstorming 用腦力激盪法寫出擴充情節, 101-104, 110
- business rules discovered 寫擴充情節時所發現到的企業規則, 100
- colon (:) for 擴充情節中的擴充條件會以分號結尾, 103
- conditions 擴充情節中的擴充情況, 99-106
- defined 定義, 3
- drawing lower in UML 畫 UML 圖時, 把擴充使用案例的位置畫得低一點 (Guideline 寫作指引 14), 236
- emphasizing 強調擴充情節的方式, 103
- exercises 相關習題, 110, 251-252
- failures within failures 失敗情況中的失敗情況, 109
- goal delivery 在擴充情節中所傳達目標的等級, 100
- handling 擴充情節中擴充情況的處理方式, 106-110
- indenting condition handling 把擴充情況的處理方式做適當的縮排(Guideline 寫作指引 12), 108
- linking 用連接符號勾子把基本使用案例跟擴充使用案例連接起來, 114-117
- merging conditions 寫擴充情節時, 把等級比較低的失敗情況合併起來, 105-106
- Rational Unified Process (RUP)Rational 統一(開發)流程(RUP)中的擴充情節範本, 108
- rationalizing 把擴充情況清單合理化, 104-105
- rollup failures 寫擴充情節時, 把等級比較低的失敗情況合併起來, 105-106
- system detection of condition 描述擴充情況時, 請說出可偵測到的東西 (Guideline 寫作指引 11), 102-103
- use case creation from 把擴充情節獨立成新的使用案例, 109-110

External primary actors 外部主要參與者, 154

eXtreme Programming (XP)終極(開發)流程, 187, 223-224

F

Failure 失敗情況. See also Extensions 請參見擴充情節

- conditions (fourth work step) 失敗情況(依照不同精確度寫出使用案例時的

第四個工作步驟), 16-17, 222

handling 處理失敗情況(Reminder 寫作提示 21), 17, 225

scenario 情節中的失敗情況, 31

Feature lists from use cases 規劃專案時, 從使用案例對應到設計工作清單, 171-174

Field details and field checks 資料欄位明細與資料欄位檢查, 163-164

Field lists 資料欄位清單, 163

Finding right goal levels 寫出合適的目標等級(Reminder 寫作提示 6), 68-69, 208

Fireman's Fund Insurance, 70-79

FirePond Corporation, 158-159, 194, 205

Fish/clam graphic, indigo/black, minus sign (subfunctions) 魚/蚌圖示、靛藍色/黑色、減號(子功能目標等級), 3, 7, 62-63, 66-67, 69, 142

Ford, Paul, 38

Form of use cases 使用案例的形式(譯註:基本上使用案例是由文字所構成的), 1

Formality and formats 規範性對使用案例格式的影響, 130

Formats 使用案例格式, 119-138

- alternate flows (extensions)在 RUP 中, 擴充情節稱為替代流程, 123
- business process modeling standard 建立企業流程模型時採用的使用案例格式標準, 132, 134
- business setting and 企業環境對使用案例格式的影響, 129
- CASE tools CASE 工具所支援的使用案例格式, 127, 227, 230
- Casual 非正式格式的使用案例, 7-9, 97, 120, 218
- completeness and 完整性對使用案例格式的影響, 131
- complexity and 複雜度對使用案例格式的影響, 130-131
- conflict and 顧客不同需求間的衝突對使用案例格式的影響, 131
- consistency and 一致性對使用案例格式的影響, 130
- context diagrams 作為情境圖的使用案例格式(譯註:最上層使用案例圖), 128, 227-228
- conversations 採用對話概念的使用案例格式, 122
- coverage and 涵蓋範圍對使用案例格式的影響, 130
- cultures and 文化衝突對使用案例格式的影響, 129
- detailed functional requirements standard 用來描寫詳細功能需求的使用案例格式標準, 132, 137
- essential use cases 寫出本質性使用案例用的使用案例格式, 122
- exercises 相關習題, 138, 252
- experience and 經驗對使用案例格式的影響, 130
- forces affecting writing styles 影響使用案例寫作風格的各種力量, 128-132
- formality and 規範性對使用案例格式的影響, 130

fully dressed 正式格式的使用案例, 4-11, 97, 119-120, 218

goals versus tasks 目標 v.s. 工作內容對使用案例格式的影響, 131

graphical notations 使用案例的圖形表示法(Reminder 寫作提示 24), 127-128, 227-228

if-statement style 採用條件子句的使用案例寫作風格, 126, 138, 218

Occam style 採用 Occam 程式語言的使用案例寫作風格, 126-127

one-column tables 使用案例的單欄表格格式, 121

Rational Unified Process (RUP) Rational 統一(開發)流程中的使用案例格式, 123-126

requirements elicitation standard 引導需求用的使用案例格式標準, 132-133

resources and 資源對使用案例格式的影響, 131

short, high-pressure project standard 短期、有高度時間壓力專案適用的使用案例格式標準, 132, 136

sizing requirements standard 評估系統大小規模用的使用案例格式標準, 132, 135

social interaction and 社會互動情形對使用案例格式的影響, 129

stakeholder needs and 關係人需要對使用案例格式的影響, 129

standards for 使用案例的寫作標準, 132-137

(one-column) tables 使用案例的單欄表格格式, 121-122

tasks versus goals 目標 v.s. 工作內容對使用案例格式的影響, 131

two-column tables 使用案例的兩欄表格格式, 122

understanding level and 理解程度對使用案例格式的影響, 129

Unified Modeling Language (UML) 統一模型語言與使用案例格式, 128

Use Case 24 (Fully Dressed Use Case Template) (使用案例的正式格式範本), 119-120

Use Case 25 (Actually Login, Casual Version) (實際的登入動作、非正式格式版本), 120, 135, 218

Use Case 26 (Register for Courses) (選課), 124-126

Use Case 27 (Elicitation Template-Oble a New Biscum) (引導需求用的範本 — OO 一個新的 OO), 133, 250

Use Case 28 (Business Process Template-Symp a Carstromming) (企業流程用的範本 — OO 一個 OO), 134, 251-252

Use Case 29 (Sizing Template-Burple the Tramling)(評估系統大小規模用的範本 — OO 這個 OO), 135, 252

Use Case 30 (High-Pressure Template: Kree a Ranfath) (高壓力專案用的範本: OO 一個 OO), 136

Use Case 31 (Use Case Name-Nathorize a Permion)(使用案例名稱 — OO 一個 OO), 137

Forward movement of action steps 每個動作步驟對目標達成都要有所進展 (Guideline 寫作指引 4), 91-92
Fowler, Martin, 236
Fully dressed use cases 正式格式的使用案例, 4-11, 97, 119-120, 218
Functional decomposition of use cases 使用案例的功能分解, 176
Functional scope 功能範圍, 36-38

G

Generalizes relations, UML UML 中的一般化關係, 236, 239-241
Goal-based core value 以目標為基礎的使用案例核心價值, 216
Goal levels, 61-79 目標等級

- actors conceptual model, xix, 參與者與目標概念性模型 23-29
- delivery, extensions 在擴充情節中所傳達目標的等級, 100
- drawing higher in UML 畫 UML 圖時, 把一般性的目標畫得高一點 (Guideline 寫作指引 16), 240
- elementary business process 基本企業流程 (譯註: 相當於使用者目標), 68
- ever-unfolding story 可不斷展開的敘事情節 (Reminder 寫作提示 12), 26, 62, 215
- exercises 相關習題, 79, 247-248
- failures and responses 失敗情況與系統的回應方式, 25
- finding right 寫出合適的目標等級 (Reminder 寫作提示 6), 68-69, 208
- graphical icons for 代表目標等級的圖示, 67-68
- length of use cases 使用案例的長度問題 (Reminder 寫作提示 20), 69, 224
- low, mistake, 192-193
- outermost scope use cases, 設計範圍最外層的使用案例 49-51, 65-66, 215
- precision 使用案例的精確度, 17
- raising and lowering 提升或降低目標等級, 69, 91-92, 192-193
- scenarios 情節中所傳達的目標等級, 88
- second work step 依照不同精確度寫出使用案例時的第二個工作步驟, 221-222
- subfunctions (underwater fish/clam graphic, indigo/black, minus sign) 子功能目標等級 (水中魚/蚌圖示、靛藍色/黑色、減號), 62-63, 66-67, 69, 142
- summary (strategic) level (cloud/kite graphic, white, plus sign) 目標摘要 (策略) 等級 (白雲/風箏圖示、白色、加號), 3, 7, 62-67, 142, 144
- tasks format versus 目標 v.s. 工作內容對使用案例格式的影響, 131
- Use Case 18 (Operate an Insurance Policy) (處理保單), 65, 67, 153

Use Case 19 (Handle a Claim, Business)(企業、處理索賠), 59, 70-71, 153, 159
Use Case 20 (Evaluate Work Comp Claim) (評估職業傷害賠償的索賠), 65, 71-72, 153, 209
Use Case 21 (Handle a Claim, Systems)(系統、處理索賠), 65, 73-74, 156-157
Use Case 22 (Register a Loss) (登錄損失情形), 74-78, 89, 109, 127, 209
Use Case 23 (Find a Whatever, Problem Statement) (找尋想要的東西), 66, 78-79, 150
user-goal level (sea-level waves graphic, blue, exclamation mark)使用者目標等級 (海平面波浪圖示、藍色、驚嘆號), 62-64
Grammar (simple) for action steps 用簡單句寫出動作步驟 (Guideline 寫作指引 1), 90
Graphical icons/model 圖示／圖形模型. See also Unified Modeling Language 請參見統一模型語言
 contract for behavior 合約的行為部分, 31-33, 229
 for goal levels 表示目標等級的圖示, 67-68
 notations 使用案例的圖形表示法(Reminder 寫作提示 24), 127-128, 227-228
 scope of 用圖形模型畫出專案範圍, 45, 51
 for scope 用圖示來標示設計範圍, 40-41
Graphical user interfaces (GUIs), keeping out 不要寫出 GUI (Reminder 寫作提示 7), 209, 219
Greenberg, Marc, 70
Grey (black-box use cases)黑色 (黑箱式使用案例), 4-7, 9-11, 40-41
Grey/white, box graphic (system use cases)黑色／白色、箱子圖示(系統使用案例), 3-7, 9-11, 41, 157-159, 216
Grey/white, building graphic (business use cases) 黑色／白色、建築物圖示 (企業使用案例), 3, 7, 41
Guarantees for stakeholders 需要對關係人做出事後保證(Reminder 寫作提示 9), 2, 83-85, 210-211, 248
Guidelines 寫作指引
 arrow shapes in UML 對 UML 中的各種使用案例關係, 請用不同的箭頭形狀 (Guideline 寫作指引 15), 236-237
 bird's eye view 用鳥瞰觀點來寫動作步驟 (Guideline 寫作指引 3), 91, 217
 detection of condition 描述擴充情況時, 請說出可偵測到的東西(Guideline 寫作指引 11), 102-103
 do until condition 重複執行動作步驟 x-y 直到...為止 (Guideline 寫作指引 10), 96-97
 extension drawing lower in UML 畫 UML 圖時, 把擴充使用案例的位置畫得低一點 (Guideline 寫作指引 14), 236

forward movement of action steps 每個動作步驟對目標達成都要有所進展 (Guideline 寫作指引 4), 91-92

goals drawing higher in UML 畫 UML 圖時，把一般性的目標畫得高一點 (Guideline 寫作指引 16), 240

grammar (simple) for action steps 用簡單句寫出動作步驟(Guideline 寫作指引 1), 90

higher-level goals in UML 畫 UML 圖時，把等級比較高的目標畫得高一點 (Guideline 寫作指引 13), 235

indenting condition handling 把擴充情況的處理方式做適當的縮排(Guideline 寫作指引 12), 108

intensions of actors 寫出參與者的意圖，而非動作(Guideline 寫作指引 5), 92-93

reasonable set of actions steps 在動作步驟中放入「合理的」一組動作 (Guideline 寫作指引 6), 93-95

supporting actors on right in UML 畫 UML 圖時，把支援性參與者放在圖的右邊(Guideline 寫作指引 18), 243

systems interaction 系統間的互動情形：使用者要系統 A 跟系統 B 做某件事 (Guideline 寫作指引 9), 96

timing 選擇性提到動作步驟的執行時間(Guideline 寫作指引 8), 95-96

user goals in context diagram 不要在情境圖中畫出目標等級比使用者目標等級更低的目標(Guideline 寫作指引 17), 243

validation versus checking 用「證實」的語調來寫，而不要用「檢查某個東西是否有成立」的語調來寫(Guideline 寫作指引 7), 95

"Who has the ball?" 球在誰的手上(Guideline 寫作指引 2, Reminder 寫作提示 5), 90-91, 207

GUIs, keeping out 不要寫出 GUI (Reminder 寫作提示 7), 209, 219

H

Hammer, Michael, 154

Handling extensions 擴充情節中擴充情況的處理方式, 106-110

Heymer, Volker, 108

High-precision view of system's functions 系統功能的高精確度觀點(譯註：寫出使用案例的內容), 180, 182-183

High-pressure project standard 高壓力專案用的使用案例範本標準, 132, 136

Higher-level goals in UML 畫 UML 圖時，把等級比較高的目標畫得高一點 (Guideline 寫作指引 13), 235

Hoare, Tony, 126-127
Hohmann, Luke, 163, 177
Holistic Diversity pattern 全方位團隊樣式, 224
"Hub-and-Spoke" requirements model 需求的「輪軸與輪輻」模型, 15, 164
Hunt, Andy, 227
Hupp, Brent, 70

I

IBM, xix, 180, 243
If-statement style format 採用條件子句的使用案例寫作風格, 126, 138, 218
In/out list for scope 可以用專案範圍內／外清單定出專案範圍, 35-36, 51
Includes relations 包含關係

- sub use cases 「包含」某些子使用案例(Reminder 寫作提示 4), 207
- Unified Modeling Language (UML) UML 中的包含關係, 234-236

Indenting condition handling 把擴充情況的處理方式做適當的縮排(Guideline 寫作指引 12), 108
Indigo/black, underwater fish/clam graphic, minus sign (subfunctions) 靛藍色／黑色、水中魚／蚌圖示、減號(子功能目標等級), 3, 7, 62-63, 66-67, 69, 142
Information nicknames 資訊暱稱, 162
Intensions of actors 寫出參與者的意圖, 而非動作(Guideline 寫作指引 5), 92-93
Interaction between two actors 兩個參與者間的互動情形, 31
Interactions, compound 互動情形可以是合成的, 25-27
Interface detail description 在動作步驟中寫出使用者介面詳細描述, 92
Internal actors and white-box cases 內部參與者與白箱式使用案例, 59-60
Internal state change 系統內部狀態變動, 31
Italics for linking use cases 呼叫某個子使用案例時, 用把被呼叫到的子使用案例名稱變成斜體字, 113
Ivey, Paula, 70

J

Jacobson, Ivar, 93, 227
Jewell, Nancy, 70
Job titles 專案剛開始與快結束時, 工作職稱是很重要的(Reminder 寫作提示 22), 225-226

K

Kite/cloud graphic, white, plus sign (summary use cases) 風箏／白雲圖示、白色、加號（目標摘要等級使用案例）, 3, 7, 62-67, 142, 144

Kraus, Andy, 180, 184-186

L

Lazar, Nicole, 70

Length of use cases 使用案例的長度問題 (Reminder 寫作提示 20), 69, 224

Level of view 不同觀點所需要的使用案例目標等級, 2, 7

Lilly, Susan, 116, 145, 216

Linking use cases 使用案例間的連結關係, 113-117

- business and system use cases 把企業使用案例跟系統使用案例連接起來, 157-159

- exercises 相關習題, 117

- extension use cases 用連接符號勾子把基本使用案例跟擴充使用案例連接起來, 114-117

- italics for 呼叫某個子使用案例時，用把被呼叫到的子使用案例名稱變成斜體字, 113

- sub use cases 動作步驟中的子使用案例, 113

- underlining for 呼叫某個子使用案例時，用把被呼叫到的子使用案例名稱加上底線, 3, 113

- Unified Modeling Language (UML) for 用 UML 畫出基本使用案例跟擴充使用案例間的連結關係, 114-115

Lockwood, Lucy, 58, 92, 122, 163, 177

Lotus Notes for tool 用 Lotus Notes 作為記錄使用案例的工具, 229

Low-precision 低精確度

- representation of use cases 使用案例的低精確度呈現方式, 143

- view of system's functions 系統功能的低精確度觀點（譯註：寫出使用案例名稱）, 180-182

M

Magdaleno, Trisha, 70

Main success scenarios (third work step) 主要成功情節 (依照不同精確度寫出使用案例時的第三個工作步驟), 3, 17, 28, 87-89, 222

Many computers to one application 從包含多個電腦的設計範圍到單一應用程式的設計範圍, 43-45

Many use cases 處理大量的使用案例, 143-144

Margel, Dale, 46

Maxwell, Allen, 145

McBreen, Pete, 178-180, 211

Merging conditions, extensions 寫擴充情節時, 把等級比較低的失敗情況合併起來, 105-106

Minimal guarantees, 83-85, 248

Minus sign (-) subfunctions 減號 (子功能目標等級), 62-63, 66-67, 69, 142

Missing requirements 被漏掉的需求, 161-165

- cross-linking from use cases to 從使用案例連到其他需求, 164-165

- data requirement precision 資料需求精確度, 162-164

- field details and field checks 資料欄位細節與資料欄位檢查, 163-164

- field lists 資料欄位清單, 163

- "Hub-and-Spoke" requirements model 需求的「輪軸與輪輻」模型, 15, 164

- information nicknames 資訊暱稱, 162

- precision in data requirements 資料需求精確度, 162-164

Mistakes, costs of 了解犯錯成本(Reminder 寫作提示 19), 223-224

Mistakes fixed 常見的使用案例寫作錯誤, 189-202

- content and purpose misalignment 寫作目的跟寫作內容不一致, 193

- goal levels, very low 所描述的目標等級太低, 192-193

- primary actors (none) 沒描述到主要參與者的行為, 190-191

- purpose and content misalignment 寫作目的跟寫作內容不一致, 193

- system (none) 沒描述到系統的行為, 189-190

- Use Case 36 (Research a Solution-Before)(找出解決方案 — 改善前的版本), 122, 194-199, 205

- Use Case 37 (Research Possible Solutions-After)(找出可行的解決方案 — 改善後的版本), 199-202

- user interface details, too many 描述太多使用者介面細節, 191-192, 194-202

Modeling versus designing 進行設計 v.s. 建立模型, 153-157

MSS. See Main success scenerios 請參見主要成功情節

N

Navigation Technologies, 38

Numbering actions steps 替動作步驟編號, 97, 218

Nuts and bolts use cases 描述具體細節的使用案例, 41, 46-49

O

Object-oriented design from use cases 直接根據使用案例進行物件導向設計, 176-177

Occam language 採用 Occam 程式語言的使用案例寫作風格, 126-127

Offstage actors 幕後（第三級）參與者, 30, 53-54

One-column table format 使用案例的單欄表格格式, 121

Outermost scope use cases 設計範圍最外層的使用案例, 49-51, 65-66, 215

P

Packages, UML 用 UML 中的套件替使用案例分群, 143

Paragraphs versus numbered steps 直接寫出段落 v.s. 替動作步驟編號, 97, 218

Parameterized use cases 可參數化使用案例, 150-151

Partial ordering 部份有順序性的動作序列, 26

Pass/fail tests 針對每個使用案例所做的欄位合格／不合格測驗(Reminder 寫作提示 10), 211-213

Passini, Susan, 70

Planning from use cases 用使用案例標題與使用案例簡介來規劃專案(Reminder 寫作提示 26), 167-170, 230

Plus sign (+) summary use cases 加號（目標摘要等級使用案例）, 3, 7, 62-67, 142, 144

Pratt, Pamela, 70

Precision 精確度

- data requirements 資料需求的精確度, 162-164

- system's functions 系統功能的精確度, 180-183

- use cases 使用案例的精確度, 16-17

- user interface (UI) design 不同精確度的使用者介面（UI）設計, 178

Preconditions 使用案例的事先條件(Reminder 寫作提示 10), 2, 81-83, 211

Primary actors 主要參與者, 54-58. See also Stakeholders; System under discussion 請參見關係人、討論中系統

- actor profile table 參與者描述表, 58
- aliases of 參與者的別名, 58
- contract for behavior 合約的行為部分, 23, 27, 30-31
- defined 定義, 1-2, 54
- design and 設計時主要參與者的重要性, 56-57
- first work step 依照不同精確度寫出使用案例時的第一個工作步驟, 221-222
- none mistake 沒描述到主要參與者的行為, 190-191
- roles versus 由參與者扮演角色(Reminder 寫作提示 23), 57-58, 226
- scaling up using 爲了擴大規模處理大量的使用案例, 根據主要參與者替使用案例分群, 144
- system delivery and 準備交付系統時主要參與者的重要性, 57
- triggers and 扮演使用案例驅動者的主要參與者, 54-55
- ultimate primary actors 使用案例的最終主要參與者, 54-55
- Unified Modeling Language (UML) 統一模型語言中所提供的參與者特殊化關係, 58
- use case production and 剛開始產生使用案例時主要參與者的重要性, 55-56
- use case writing and 寫使用案例時主要參與者的重要性, 56-57
- user goals for completion 有寫出主要參與者的使用者目標才算寫完使用案例, 141

Profile table, actor 參與者描述表, 58

Project-linking software, use cases as 把使用案例當成連接專案用的結構, 14-15

Project planning 專案規劃, 167-186

- branch-and-join process for 專案規劃時, 採用個人與團體並重的使用案例寫作過程, 180-183
- collecting use cases from large groups 大型團體中的使用案例蒐集方式, 184-186
- "completeness," 漸增式開發方式的「完成」概念 175
- design documents and 專案規劃時, 從使用案例對應到設計文件, 171-174
- design from use cases 專案規劃時, 從使用案例到設計, 174-177
- domain concepts from use cases 使用案例可點出領域概念, 177
- feature lists from use cases 專案規劃時, 從使用案例對應到系統特性清單, 171-174
- functional decomposition of use cases 專案規劃時, 以使用案例進行功能分解, 176
- high-precision view of system's functions 系統功能的高精確度觀點 (譯註:

寫出使用案例的內容), 180, 182-183

low-precision view of system's functions 系統功能的低精確度觀點(譯註: 寫出使用案例名稱), 180-182

object-oriented design from use cases 直接根據使用案例進行物件導向設計, 176-177

precision of user interface (UI) design 不同精確度的使用者介面(UI)設計, 178

releases and use cases 橫跨不同發行版本的使用案例, 169-170, 230

Responsibility-Driven Design 責任驅動式設計方式, 177

scenarios and design 利用情節來進行設計工作, 177

scenarios (complete) delivery 每次實作出完整的情節, 170

task lists from use cases 專案規劃時, 從使用案例對應到設計工作清單, 171-174

test cases from 從使用案例而來的測試案例, 178-180

time required per use case 每個使用案例所需花費的寫作時間, 184

Use Case 34 (Capture Trade-In) (記錄舊換新活動), 172-173

Use Case 35 (Order Goods, Generate Invoice, Testing Example) (下貨品訂單、產生發票【測試範例】), 178-179

use cases for 用使用案例標題與使用案例簡介來規劃專案(Reminder 寫作提示 26), 167-170, 230

user interface design from use cases 不同精確度的使用者介面(UI)設計, 177-178

writing 專案規劃時, 實際可行的使用案例寫作過程, 180-186

Prose essay, use cases as 使用案例是一種散文(Reminder 寫作提示 1), 205

Purpose(s)寫作目的

- and content misalignment 寫作目的跟寫作內容不一致, 193
- of uses cases 使用案例可以有多種寫作目的, 217
- and writing style 使用案例的寫作目的與寫作風格, 7-11

Q

Quality questions 跟整組使用案例有關的品質問題(Reminder 寫作提示 15), 11, 219

R

Raising and lowering goal levels 提升或降低目標等級, 69, 91-92, 192-193

Raison d'etre (elementary business process 基本企業流程【譯註：相當於使用者目標】), 68

Rational Software Corporation, 123

Rational Unified Process Rational 統一（開發）流程(RUP)

- extensions RUP 中的擴充情節格式, 108
- formats RUP 中的使用案例格式, 123-126

Rationalizing extensions 把擴充情況清單合理化, 104-105

Readability of use cases 把使用案例寫的很容易讀(Reminder 寫作提示 2), 205-206, 217

Reasonable set of actions steps 在動作步驟中放入「合理的」一組動作(Guideline 寫作指引 6), 93-95

Recovery (fifth work step)復原用動作步驟（依照不同精確度寫出使用案例時的第五個工作步驟）, 222-223

Reference to another use case (underlined), 3

Relational databases 用關聯式資料庫作為記錄使用案例的工具, 229

Releases and use cases 橫跨不同發行版本的使用案例, 169-170, 230

Releases for scaling up 為了擴大規模處理大量的使用案例，根據發行版本替使用案例分群, 144

Reminders, 205-230

- actor-goal lists versus (use) case diagrams 參與者 — 目標清單 v.s.使用案例圖, 218
- actors play roles 由參與者扮演角色(Reminder 寫作提示 23), 57-58, 226
- alternative paths 把替代路徑放在主要成功情節後面, 217
- bird's eye view 用鳥瞰觀點來寫動作步驟(Guideline 寫作指引 3), 91, 217
- black-box requirements 寫出黑箱式需求, 217
- business process modeling, prior to use cases 寫出使用案例前先建立企業模型, 218
- business versus system use cases 公司設計範圍跟系統設計範圍之比較 (Reminder 寫作提示 13), 216
- (use) case diagrams versus actor-goal lists 使用案例圖 v.s.參與者 — 目標清單, 218
- CASE tools 用來寫出使用案例的 CASE 工具, 127, 227, 230
- casual versus fully dressed 非正式格式 v.s. 正式格式, 218
- collaboration diagrams (UML) versus white-box (use) cases 合作圖 (UML) v.s. 白箱式使用案例, 218
- core values and variations 使用案例格式的核心價值與替代性價值(Reminder 寫作提示 14), 216-219
- data field details and checks (seventh work step) 資料欄位細節與資料欄位檢

查（依照不同精確度寫出使用案例時的第七個工作步驟）, 223

data fields (sixth work step)資料欄位（依照不同精確度寫出使用案例時的第六個工作步驟）, 223

endings (two) of use cases 使用案例的兩種結尾(Reminder 寫作提示 8), 209-210

energy management 管理自己的精力, 16-17, 217, 221-223

ever-unfolding story 可不斷展開的敘事情節 (Reminder 寫作提示 12), 26, 62, 215

failure conditions (fourth work step)失敗情況（依照不同精確度寫出使用案例時的第四個工作步驟）, 16-17, 222

failure handling 處理失敗情況(Reminder 寫作提示 21), 17, 225

fully dressed versus casual 正式格式 v.s.非正式格式, 218

goal-based core value 以目標為基礎的使用案例核心價值, 216

goal level, getting right 寫出合適的目標等級(Reminder 寫作提示 6), 68-69, 208

goals (second work step)目標（依照不同精確度寫出使用案例時的第二個工作步驟）, 221-222

graphical notations 使用案例的圖形表示法 (Reminder 寫作提示 24), 127-128, 227-228

guarantees for stakeholders 需要對關係人做出事後保證(Reminder 寫作提示 9), 2, 83-85, 210-211, 248

GUIs, keeping out 不要寫出 GUI (Reminder 寫作提示 7), 209, 219

Holistic Diversity pattern 全方位團隊樣式, 224

if-statement style 採用條件子句的使用案例寫作風格, 126, 138, 218

includes relation, sub use cases 「包含」某些子使用案例(Reminder 寫作提示 4), 207

job titles 專案剛開始與快結束時，工作職稱是很重要的(Reminder 寫作提示 22), 225-226

length of use cases 使用案例的長度問題 (Reminder 寫作提示 20), 69, 224

Lotus Notes for tool 用 Lotus Notes 作為記錄使用案例的工具, 229

main success scenarios (third work step)主要成功情節（依照不同精確度寫出使用案例時的第三個工作步驟）, 3, 17, 28, 87-89, 222

mistakes, costs of 了解犯錯成本(Reminder 寫作提示 19), 223-224

numbered steps versus paragraphs 直接寫出段落 v.s. 替動作步驟編號, 97, 218

paragraphs versus numbered steps 直接寫出段落 v.s. 替動作步驟編號, 97, 218

pass/fail tests 針對每個使用案例所做的欄位合格／不合格測驗(Reminder 寫作提示 10), 211-213

preconditions 使用案例的事先條件(Reminder 寫作提示 10), 2, 81-83,

211

primary actors (first work step) 主要參與者 (依照不同精確度寫出使用案例時的第一個工作步驟), 221-222

project planning, 167-170, 230 用使用案例標題與使用案例簡介來規劃專案 (Reminder 寫作提示 26)

prose essay, use cases as 使用案例是一種散文 (Reminder 寫作提示 1), 205

purposes (several) of uses cases 使用案例可以有多種寫作目的, 217

quality questions 跟整組使用案例有關的品質問題 (Reminder 寫作提示 15), 11, 219

readability of use cases 把使用案例寫的很容易讀 (Reminder 寫作提示 2), 205-206, 217

recovery (fifth work step) 復原用的動作步驟 (依照不同精確度寫出使用案例時的第五個工作步驟), 222-223

relational databases for 用關聯式資料庫作為記錄使用案例的工具, 229

releases and use cases 橫跨不同發行版本的使用案例, 169-170, 230

requirements and use cases 使用案例只是需求文件中的第三章 (Reminder 寫作提示 16), 13-15, 19, 221

requirements management tools 需求管理工具, 230

roles and actors 由參與者扮演角色 (Reminder 寫作提示 23), 57-58, 226

sentence form (one) 寫出動作步驟時只用一種句型 (Reminder 寫作提示 3), 206-207

sequence diagrams as use case text 用循序圖代替使用案例的說明文字, 219

stakeholders need guarantees 需要對關係人做出事後保證 (Reminder 寫作提示 9), 2, 83-85, 210-211, 248

sub use cases, includes relation 「包含」某些子使用案例 (Reminder 寫作提示 4), 207

system versus business use cases 公司設計範圍跟系統設計範圍之比較 (Reminder 寫作提示 13), 216

tool debate 工具之爭 (Reminder 寫作提示 25), 229-230

12-step recipe 使用案例的 12 步寫作訣竅 (Reminder 寫作提示 18), 223

white-box cases versus collaboration diagrams (UML) 白箱式使用案例 v.s. 合作圖 (UML), 218

"Who has the ball?" 球在誰的手上 (Guideline 寫作指引 2, Reminder 寫作提示 5), 90-91, 207

word processors with hyperlinks for tool 用具有超連結的文字處理器作為記錄使用案例的工具, 229

work breadth first 寫使用案例時先求廣度 (Reminder 寫作提示 17), 221-223

Repeating actions steps 重複執行動作步驟 x-y 直到...為止, 96-97

Requirements. See also Missing requirements; Use cases
discovery, 11-12, 133
elicitation standard 引導需求用的使用案例格式標準, 132-133
management tools 需求管理工具, 230
sizing requirements standard 評估系統大小規模用的使用案例格式標準, 132, 135
use cases and 使用案例只是需求文件中的第三章(Reminder 寫作提示 16), 13-15, 19, 221

RequisitePro, 230

Resources and formats 資源對使用案例格式的影響, 131

Responsibility-Driven Design 責任驅動式設計方式, 177

Robertson, James, 13

Robertson, Suzanne, 13

Roles and actors 由參與者扮演角色(Reminder 寫作提示 23), 57-58, 226

Rollup failures, extensions 寫擴充情節時，把等級比較低的失敗情況合併起來, 105-106

Roshi, 211

RUP. See Rational Unified Process 請參見 Rational 統一（開發）流程

S

Sampson, Steve, 70

Sawyer, Jim, 8

Scaling up 擴大規模處理大量的使用案例, 143-144

Scenarios 情節. See also Action steps; Extensions 請參見動作步驟、擴充情節

- body 情節的主體, 89
- collection 把一些情節集合在一起，以形成使用案例, 27-29
- condition for 情節的開始執行條件, 88
- contract for behavior 合約的行爲部分, 25
- defined 定義, 1, 3
- delivery and 每次實作出完整的情節, 170
- design and 利用情節來進行設計工作, 177
- end condition 情節的結束情況, 88
- extensions 情節中可能會發生的一組擴充情節，我們會把它們寫成情節片段, 88
- goal to achieve 情節達成的目標, 88
- main success scenarios 主要成功情節跟所有的擴充情節一樣，都會有環繞在

情節周圍的相似結構, 3, 17, 28, 87-89, 222

Schick Tanz, Jill, 70

Scope (design scope) 專案範圍 (設計範圍), 35-52 (譯註: 專案範圍包括功能範圍與設計範圍兩者)

actor-goal list for 可以用參與者 — 目標清單定出功能範圍, 36-37, 51

business priority 寫出功能範圍時可能會增列的欄位: 企業優先順序, 36

defined 定義, 2

development complexity 寫出功能範圍時可能會增列的欄位: 開發複雜度, 36

development priority 寫出功能範圍時可能會增列的欄位: 開發優先順序, 37

enterprise-to-system scope 從企業設計範圍到系統設計範圍, 41-42

exercises 相關習題, 51-52, 245

functions scope 功能範圍, 36-38

graphical icons for 用圖示來標示設計範圍, 40-41

graphical model 用圖形模型秀出專案範圍, 45, 51

in/out list for 可以用專案範圍內/外清單定出專案範圍, 35-36, 51

many computers to one application 從包含多個電腦的設計範圍到單一應用程式的設計範圍, 43-45

nuts and bolts use cases 描述具體細節的使用案例, 41, 46-49

outermost scope use cases 設計範圍最外層的使用案例, 49-51, 65-66, 215

triggers 寫出功能範圍時可能會增列的欄位: 驅動者, 36

Unified Modeling Language (UML) 寫使用案例或畫使用案例圖時, 秀出專案範圍 (譯註: 包括功能範圍與設計範圍兩者), 44-45

use case briefs for 可以用使用案例簡介定出功能範圍, 37-38, 187

Use Case 6 (Add New Service, Enterprise) (企業的) 新增服務申請, 42, 50

Use Case 7 (Add New Service, Acura) (Acura 的) 新增服務申請, 42

Use Case 8 (Enter and Update Requests, Joint System) (聯合兩個系統的) 輸入與更新服務申請, 43, 59

Use Case 9 (Add New Service into Acura) (Acura 內的) 新增服務申請, 44

Use Case 10 (Note New Service Requests in BSSO) (BSSO 內的) 通知新增服務申請, 44

Use Case 11 (Update Service Request in BSSO) (BSSO 內的) 更新服務申請, 44

Use Case 12 (Note Updated Request in Acura) (Acura 內的) 通知更新服務申請, 44

Use Case 13 (Serialize Access to a Resource) 對資源做可序列化的存取動作, 46-47, 112

Use Case 14 (Apply a Lock Conversion Policy) 應用資源鎖定轉換政策, 47-48

Use Case 15 (Apply an Access Compatibility Policy) 應用存取權相容性政策, 48

Use Case 16 (Apply an Access Selection Policy) 應用存取權選擇政策, 48-49

Use Case 17 (Make Service Client Wait for Resource Access) 讓服務客戶端等待資源存取權, 49

vision statements 可以用專案願景聲明定出專案範圍, 51

Scott, Dave, 194-202

Sea-level waves graphic, blue, exclamation mark (user-goal use cases) 海平面波浪圖示、藍色、驚嘆號(使用者目標等級使用案例), 3-4, 6-7, 9-11, 62-64

Secondary actors 次要參與者. See Supporting actors 請參見支援性參與者

Sentence form for action steps 寫出動作步驟時只用一種句型 (Reminder 寫作提示 3), 206-207

Sequence diagrams as use case text 用循序圖代替使用案例的說明文字, 219

Sequences of interactions 構成互動情形的一連串動作步驟, 25-27

Services, business process modeling 建立企業流程模型後, 將可知道企業所提供的服務, 154

Sets of possible sequences 未來可能會發生的一組動作序列, 26-27

Short, high-pressure project standard 短期、有高度時間壓力專案適用的使用案例格式標準, 132, 136

Silent (offstage) actors 沉默(幕後)參與者, 30, 53-54

Situations for use cases 不同使用案例寫作風格的適用情況, 7-11

Sizing requirements standard 評估系統大小規模用的使用案例格式標準, 132, 135

Social interaction and formats 社會互動情形對使用案例格式的影響, 129

Software Futures CCH, 108

S.R.A., 116, 145

Stakeholders 關係人. See also Actors 請參見參與者

- business process modeling 建立企業流程模型後, 將可知道跟組織行為有關的關係人, 154
- defined 定義, 1-2, 53-54
- interests conceptual model 關係人利益概念性模型, 29-31
- need guarantees 需要對關係人做出事後保證(Reminder 寫作提示 9), 2, 83-85, 210-211, 248
- needs and formats 關係人需要對使用案例格式的影響, 129

Standards 使用案例的寫作標準, 11, 132-137

Statements, vision 可以用專案願景聲明定出專案範圍, 51

Steps 步驟. See Action steps 請參見動作步驟

Strategic use cases (cloud/kite graphic, white, plus sign) 策略等級使用案例(白雲/風箏圖示、白色、加號), 3, 7, 62-67, 142, 144 (譯註: 作者已經把策略等級使用

案例改稱為目標等級使用案例)

Striped trousers image, contract for behavior 條紋褲、合約的行為部分, 27-29

Subforms of use cases 使用案例的次要寫作型態, 7-11

Subfunctions (underwater fish/clam graphic, indigo/black, minus sign) 子功能目標等級 (水中魚/蚌圖示、靛藍色/黑色、減號), 3, 7, 62-63, 66-67, 69, 142

Subgoals, contract for behavior 為了達成使用案例目標 (合約的行為部分), 系統必須有系統地達到一些子目標, 23-24

Subject area for scaling up 為了擴大規模處理大量的使用案例, 根據主題範圍替使用案例分群, 144

Subordinate versus sub use cases, UML 下層使用案例 vs. 子使用案例, 242

Sub use cases 子使用案例;

- includes relation 「包含」某些子使用案例(Reminder 寫作提示 4), 207
- linking 動作步驟中的子使用案例, 113

Success guarantees 成功事後保證, 84-85, 248

SuD. See System under discussion 請參見討論中系統

"Sufficient" use cases 我們通常只有時間寫出「夠用」的使用案例, 5

Summary use cases (cloud/kite graphic, white, plus sign) 目標摘要等級使用案例 (白雲/風箏圖示、白色、加號), 3, 7, 62-67, 142, 144

Supporting (secondary) actors 支援性 (次要) 參與者

- contract for behavior 主要參與者要支援性參與者達成的子目標, 是為滿足合約的行為部分, 23-24
- defined 定義, 59
- on right in UML 畫 UML 圖時, 把支援性參與者放在圖的右邊(Guideline 寫作指引 18), 243

Swift, Jonathan, 24

System under discussion (SuD) 討論中系統

- actors 討論中系統也是一種參與者, 59
- contract for behavior 合約的行為部分, 24-25, 29
- defined 定義, 2
- delivery and primary actors 準備交付系統時主要參與者的重要性, 57
- detection of condition 描述擴充情況時, 請說出可偵測到的東西(Guideline 寫作指引 11), 102-103
- none mistake 沒描述到系統的行為, 189-190

System use cases (box graphic, grey/white) 系統使用案例(箱子圖示、灰色/白色), 3-7, 9-11, 41, 157-159, 216

System versus business use cases 公司設計範圍跟系統設計範圍之比較 (Reminder 寫作提示 13), 216

Systems interaction 系統間的互動情形: 使用者要系統 A 跟系統 B 做某件事

(Guideline 寫作指引 9), 96

T

- Table format 使用案例的表格格式, 121-122
- Task lists from use cases 專案規劃時，從使用案例對應到設計工作清單, 171-174
- Tasks versus goals format 目標 v.s. 工作內容對使用案例格式的影響, 131
- Technology and data variations 技術與資料變異情形, 111-112
- Technology to business process 直接從技術往回看企業流程, 157
- Template for use cases 使用案例的範本, 7
- Tertiary (offstage) actors 幕後（第三級）參與者, 30, 53-54
- Test cases from project planning 從使用案例而來的測試案例, 178-180
- Text-based use cases versus UML 以文字為基礎的使用案例 v.s. 用 UML 畫出使用案例圖, 243
- Thomas, Dave, 227
- Thomsett, Rob, 35
- Time required per use case 每個使用案例所需花費的寫作時間, 184
- Timing 選擇性提到動作步驟的執行時間(Guideline 寫作指引 8), 95-96
- Tool debate 工具之爭 (Reminder 寫作提示 25), 229-230
- Tools, CASE 電腦輔助軟體工程工具, 127, 227, 230
- Triggers 驅動者
 - actors and 驅動使用案例的參與者, 54-55
 - business process modeling 建立企業流程模型後，將可知道組織必須回應的觸發事件, 154
 - completion and 有寫出所有觸發情況才算寫完使用案例, 141-142
 - defined 定義, 84-85
 - scope and 寫出功能範圍時可能會增列的欄位：驅動者, 36
- 12-step recipe 使用案例的 12 步寫作訣竅(Reminder 寫作提示 18), 223
- Two-column table format 使用案例的兩欄表格格式, 122

U

- UI. See User interface 請參見使用者介面
- Ultimate primary actors 使用案例的最終主要參與者, 54-55
- UML. See Unified Modeling Language 請參見統一模型語言
- Underlining for linking use cases 呼叫某個子使用案例時，用把被呼叫到的子使用

案例名稱加上底線, 3, 113

Understanding level and formats 理解程度對使用案例格式的影響, 129

Underwater fish/clam graphic, indigo/black, minus sign (subfunctions) 水中魚／蚌圖示、靛藍色／黑色、減號（子功能目標等級）, 3, 7, 62-63, 66-67, 69, 142

Unified Modeling Language 統一模型語言(UML), 233-243

- actors 統一模型語言中所提供的參與者間特殊化關係, 58
- arrow shapes, using different 對 UML 中的各種使用案例關係, 請用不同的箭頭形狀(Guideline 寫作指引 15), 236-237
- collaboration diagrams versus white-box (use) cases UML 中的合作圖 v.s. 白箱式使用案例, 218
- contract for behavior 用 UML 中的使用案例寫出合約的行為部分, 31
- drawing use case diagrams 畫出 UML 中的使用案例圖, 242-243
- ellipses and stick figures UML 中的橢圓形與簡單棒形人偶圖示, 233-234
- extend relations UML 中的擴充關係, 235-238
- extension points UML 中的擴充點, 237-238
- extension use cases 用 UML 畫出基本使用案例跟擴充使用案例間的連結關係, 114-115
- extension use cases, drawing lower 畫 UML 圖時, 把擴充使用案例的位置畫得低一點 (Guideline 寫作指引 14), 236
- formats 使用案例格式, 128
- general goals, drawing higher 畫 UML 圖時, 把一般性的目標畫得高一點 (Guideline 寫作指引 16), 240
- generalizes relations UML 中的一般化關係, 236, 239-241
- higher-level goals 畫 UML 圖時, 把等級比較高的目標畫得高一點 (Guideline 寫作指引 13), 235
- includes relations UML 中的包含關係, 234-236
- linking use cases 用 UML 畫出基本使用案例跟擴充使用案例間的連結關係, 114-115
- packages 用 UML 中的套件替使用案例分群, 143
- scope(UML)寫使用案例或畫使用案例圖時, 秀出專案範圍（譯註：包括功能範圍與設計範圍兩者）, 44-45
- subordinate versus sub use cases 下層使用案例 vs. 子使用案例, 242
- supporting actors on right 畫 UML 圖時, 把支援性參與者放在圖的右邊 (Guideline 寫作指引 18), 243
- text-based use cases versus 以文字為基礎的使用案例 v.s. 用 UML 畫出使用案例圖, 243
- user goals in context diagram 不要在情境圖中畫出目標等級比使用者目標等級更低的目標(Guideline 寫作指引 17), 243

Usage experts 系統使用情形專家, 156-157

Usage narratives 寫出使用案例之前, 先以系統使用情形簡述暖身, 17-19

Use case briefs for scope 可以用使用案例簡介定出功能範圍, 37-38, 187

Use cases 使用案例, 1-19. See also Action steps; Actors; Formats; Linking use cases; Project planning; Requirements; Scope (design scope)請參見動作步驟、參與者、使用案例格式、使用案例間的連結關係、專案規劃、需求、專案範圍(設計範圍)

- accuracy 使用案例的各種精確度, 17
- adding value with 什麼時候使用案例會更有價值, 15-16
- black-box use cases (grey) 黑箱式使用案例(黑色), 4-7, 9-11, 40-41
- brainstorming using 用使用案例來進行腦力激盪以寫出新需求, 12
- brainstorming with 用腦力激盪法寫出使用案例中的失敗情況, 16
- business use cases (building graphic, grey/white) 企業使用案例(建築物圖示、灰色/白色), 3, 7, 41
- casual use cases 非正式格式的使用案例, 7-9, 97, 120, 218
- completion 怎樣才算寫完使用案例, 141-142
- component use cases (bolt graphic) 元件使用案例(螺絲圖示), 41, 46-49
- Create, Retrieve, Update, Delete (CRUD) use cases 新增、讀取、更新或刪除OO(CRUD)這類型小使用案例, 145-150
- defined 定義, 1-3
- dive-and-surface approach using 寫使用案例時, 採用浮潛式方案, 12
- documenting requirements using 用使用案例來記錄需求, 12
- energy management 管理自己的精力, 16-17, 217, 221-223
- eXtreme Programming (XP) 終極(開發)流程, 187, 223-224
- failure conditions 使用案例中的失敗情況, 16-17
- failure handling 處理失敗情況 (Reminder 寫作提示 21), 17, 225
- form of 使用案例的寫作型態, 1
- fully dressed use cases 正式格式的使用案例, 4-11, 97, 119-120, 218
- guarantees 需要對關係人做出事後保證(Reminder 寫作提示 9), 2, 83-85, 210-211, 248
- "Hub-and-Spoke" requirements model 需求的「輪軸與輪輻」模型, 15, 164
- length of 使用案例的長度問題 (Reminder 寫作提示 20), 69, 224
- level of view 不同觀點所需要的使用案例目標等級, 2, 7
- main success scenarios (MSS) 主要成功情節跟所有的擴充情節一樣, 都會有環繞在情節周圍的相似結構, 3, 17, 28, 87-89, 222
- parameterized use cases 可參數化使用案例, 150-151
- preconditions 使用案例的事先條件(Reminder 寫作提示 10), 2, 81-83, 211
- project-linking software as 把使用案例當成連接專案用的結構, 14-15
- purpose and writing style 使用案例的寫作目的與寫作風格, 7-11

- quality questions 跟整組使用案例有關的品質問題(Reminder 寫作提示 15), 11, 219
 - reference to another use case (underlined) 呼叫 (參考) 某個子使用案例時, 用把被呼叫到的子使用案例名稱加上底線, 3, 113
 - scaling up 擴大規模處理大量的使用案例, 143-144
 - situations for 不同使用案例寫作風格的適用情況, 7-11
 - standards 使用案例的寫作標準, 11, 132-137
 - subforms of 使用案例的次要寫作型態, 7-11
 - subfunctions (underwater fish/clam graphic, indigo/black, minus sign) 子功能目標等級使用案例 (水中魚/蚌圖示、靛藍色/黑色、減號), 3, 7, 62-63, 66-67, 69, 142
 - "sufficient" use cases 我們通常只有時間寫出「夠用」的使用案例, 5
 - summary (cloud/kite graphic, white, plus sign) 目標摘要等級使用案例 (白雲/風箏圖示、白色、加號), 3, 7, 62-67, 142, 144
 - system use cases (box graphic, grey/white) 系統使用案例 (箱子圖示、灰色/白色), 3-7, 9-11, 41, 157-159, 216
 - techniques 使用案例的寫作技巧, 11
 - technology and data variations 技術與資料變異情形, 111-112
 - template for 各種使用案例範本, 7
 - usage narratives 寫出使用案例之前, 先以系統使用情形簡述暖身, 17-19
 - user-goal level (sea-level waves graphic, blue, exclamation mark) 使用者目標等級使用案例 (海平面波浪圖示、藍色、驚嘆號), 3-4, 6-7, 9-11
 - user goals stated with 使用案例名稱指出系統會支援的使用者目標, 15-16
 - white-box use cases 白箱式使用案例, 7, 40-41, 59-60, 218
- User goals 使用者目標
- context diagram in UML 不要在情境圖中畫出目標等級比使用者目標等級更低的目標(Guideline 寫作指引 17), 243
 - use cases for stating 使用案例名稱指出系統會支援的使用者目標, 15-16
 - use cases (sea-level waves graphic, blue, exclamation mark) 使用者目標等級使用案例 (海平面波浪圖示、藍色、驚嘆號), 3-4, 6-7, 9-11, 62-64
- User interface (UI) 使用者介面
- design from use cases 不同精確度的使用者介面 (UI) 設計, 177-178
 - details, too many 描述太多使用者介面細節, 191-192, 194-202
- User stories 使用者故事, 187

V

Validation 證實

checking versus 用「證實」的語調來寫，而不要用「檢查某個東西是否有成立」的語調來寫(Guideline 寫作指引 7), 95

stakeholders' protection 保障關係人的證實性動作, 31

Vision statements for scope 可以用專案願景聲明定出專案範圍, 51

View of system's functions 系統功能的觀點, 180-183

"VW-Staging" (Cockburn), 142, 170

W

Walters, Russell, 158-160, 194-202, 205

Waves graphic, blue, exclamation mark (user-goal use cases) 海平面波浪圖示、藍色、驚嘆號(使用者目標等級使用案例), 3-4, 6-7, 9-11, 62-64

White, cloud/kite graphic, plus sign (summary use cases) 白色、白雲/風箏圖示、加號(目標摘要等級使用案例), 3, 7, 62-67, 142, 144

White-box use cases 白箱式使用案例, 7, 40-41, 59-60, 218

White/grey, box graphic (system use cases) 灰色/白色、箱子圖示(系統使用案例), 3-7, 9-11, 41, 157-159, 216

White/grey, building graphic (business use cases) 灰色/白色、建築物圖示(企業使用案例), 3, 7, 41

"Who has the ball?" 球在誰的手上(Guideline 寫作指引 2, Reminder 寫作提示 5), 90-91, 207

Williams, Alan, 116-117

Wirfs-Brock, Rebecca, 122, 235

Word processors with hyperlinks for tool 用具有超連結的文字處理器作為記錄使用案例的工具, 229

Work breadth first 寫使用案例時先求廣度(Reminder 寫作提示 17), 221-223

Writing 使用案例寫作. See also Use cases 請參見使用案例

project plans 專案規劃時，實際可行的使用案例寫作過程, 180-186

styles, forces affecting 影響使用案例寫作風格的各種力量, 128-132

X

XP (eXtreme Programming) 終極（開發）流程, 187, 223-224

Y

Young, Steve, 38