推薦序

— *Martin Fowler*

敏捷軟體開發興起的有利成果之一，就是把龐大需求（requirements）分解成較小區塊（chunks）的理念，這些區塊——使用者故事（user stories）——讓專案開發的進展更具能見度。當產品透過一個故事接著一個故事被建造時，每個故事的實作完全被整合到軟體產品中，每個人都能夠看見產品的成長。藉由對使用者有意義的故事，開發者能夠決定接下來要建造哪些故事，進而充分掌握專案。這種更清晰的能見度有助於鼓勵使用者更深入地參與專案——不再只是等待一年或更長的時間，才能夠看到開發團隊端出什麼美味佳餚。

然而，這樣的需求分解具有一些負面效果，其中之一就是很容易丟失軟體系統應該做什麼的整體圖像（big picture，或整體概觀），最後會得到一堆與整體專案不契合的零碎片段，或者建造出一個對使用者無實質幫助的系統——因為迷失在細節裡，因而錯過需求的本質。

故事對照（story mapping）是彌補上述缺點的技術，為一堆故事提供它們經常錯失的整體圖像。

真的，就是這樣——這本書就是在闡述這句話，而這句話蘊涵著許多價值與承諾。整體圖像有助於讓我們跟使用者有效溝通，避免每個人建造不必要的功能，並且為前後連貫的使用者經驗提供方向。當我跟 Thought Works 的同事討論到他們如何發展使用者故事時，故事對照往往是他們採用的核心技術。他們通常是從 Jeff 主持的研習會中學到這項技術，因為他是發展這項技術並且能夠妥善傳達相關理念的人。這本書讓更多人能夠直接從源頭瞭解這項技術。

不過，這本書不只針對名片或線上履歷中具有「商業分析師」頭銜的那類人。或許，在採取敏捷方法的多年經驗中，最讓我失望的是，許多程

式人員只是將使用者故事視為分析師對他們的單向溝通，但事實上，打從根本起，使用者故事應該促進更充分的**對話**（*conversation*）。如果你想要實際構思支援某個活動的有效軟體，就必須將建造軟體的人們視為理解其功能的重要來源，因為最瞭解軟體能夠做什麼的人就是程式人員。程式人員必須理解他們的使用者試圖獲得什麼，並且協同建造捕捉使用者需求的故事。瞭解故事對照的程式人員可以更清楚地看到更寬廣的使用者上下文，並且能夠參與軟體的設計——產生更棒的成果。

在 Kent Beck（「故事」觀念的創造者）發展他的軟體開發理念時，他極力呼籲，充分溝通是有效團隊的關鍵價值，故事是在開發者與使用者之間進行溝通的建構區塊，而故事對照組織並架構起這些建構區塊，因而強化這個溝通過程——那是軟體開發最關鍵的部分。

— Martin Fowler
June 18, 2014

推薦序

— *Alan Cooper*

在 Mary Shelley 的著名科幻小說《**科學怪人**》中，瘋狂的 Frankenstein 醫生利用墳場屍塊創造出恐怖的人形怪物，並且運用當時最先進的電擊技術賦予它生命。當然，我們知道這實際上是不可能的，你無法隨便將身體各個部分縫合起來，繼而創造出美妙的生命。

然而，這是軟體開發者一直試圖要做的事情，他們為軟體添加良好功能，一次一個，然後不解為何只有少數使用者垂青他們的產品，問題的核心在於，開發者使用自己的建構方法（construction method）作為設計工具（design tool），然而，兩者其實是不可互換的。

程式人員一次一個功能地**建造**軟體是完全合理的，多年經驗告訴我們，那確實是相當好的策略，然而，經過多年實證，當它被用來設計數位產品的行為與範疇時，一次一個功能的做法恐怕會產生程式版的科學怪人。

雖然密切關聯，但設計軟體行為與建造軟體之實務其實是大相逕庭的，而且，通常由不同的人運用不同的技能來達成。互動設計師花在觀察使用者與對照行為模式的大量心力會把多數程式人員都搞瘋，相反地，在演算法上揮汗荷鋤地辛勤耕耘對多數互動設計師來說實在太寂寞了。

然而，當設計與開發這兩種實務協同合作，發揮綜效時，就可能創造出具有生命力的產品。團隊合作賦予這個怪物美妙的生命，並且讓人們深深喜愛它。

雖然協同合作的想法既不新穎，亦不特別深刻，但知易行難，想要有效達成實屬不易。開發者工作的方式——他們的步調、用語和節奏——十分不同於互動設計師。

兩個領域的從業人員都很強大，有能力，且訓練有術，但是他們有一個共通的弱點：很難以編程術語表達設計問題，同樣地，也很難以設計術語表達開發問題，兩個姐妹領域缺乏共通的語言，其間的連結正是 Jeff Patton 擅長的領域。

Jeff 的故事對照方法對開發者很合理，對設計者也一樣。故事對照是數位時代的羅塞塔石碑（Rosetta Stone）[譯註]。

儘管與聲明相反，敏捷開發並不是非常有用的設計工具（design tool），它是思考對設計友善之開發的方式，那是很好的事情，然而，單靠它本身並不會讓你得到使用者喜愛的產品。另一方面，有非常多次的經驗，我們看到良好的設計（且說明文件完備）被交給開發者（不管是否為敏捷式），卻在實作過程中扼殺掉該設計的本質精髓。

Patton 的故事對照方法是跨越這個裂隙的橋樑。互動設計全然關乎發掘使用者的真正需求，並且像說故事般地將它敘述出來。軟體開發全然關乎把那些故事分解成細小的功能區塊，並加以實作及整合。然而，在這個複雜的過程中，故事的本質精髓非常容易「走鐘」，功能確實被實作出來，但科學怪人也死在手術檯上了。

透過對照使用者故事，設計保留了它的敘事結構（narrative structure），但還能被解構來進行有效的實作。設計者的故事（係使用者的故事的正式版）在整個開發期間維持完整無缺。

傳統大企業已經證實，二、三百人的團隊幾乎不可能建造出人們喜歡的產品，同時，新創公司社群亦已證實，四、五個人的團隊能夠打造出人們鍾愛的小產品，但這些小產品終究還是會變大，並且失去它們的火花。我們面臨的挑戰是建造人們喜愛的大軟體，大軟體服務廣大人群，進行具有商業利益的複雜工作，說實在，真的很難讓這樣的軟體用起來有趣，學起來容易。

打造非科學怪人式的大型軟體的唯一方式，就是學習如何整合軟體設計與軟體開發這兩個領域，沒有人比 Jeff Patton 更瞭解這件事。

— Alan Cooper
June 17, 2014

[譯註] 請參考 *http://zh.wikipedia.org/wiki/* 羅塞塔石碑。

推薦序

— *Marty Cagan*

我極其有幸能跟許多擁有全球最佳技術的產品團隊共事，這些人打造你鍾愛並且每天使用的產品。這些團隊實際上正在改變這個世界。

我也被引介並且幫助做得不是那麼好的公司。新創公司在燒完錢之前全力爭取市場關注，較大型的公司奮力重現早期的創新文化，團隊無法持續為公司增添價值，領導者對於需要曠時費日才能將想法付諸實現深感挫折，而工程師對產品負責人大感光火。

我所覺察的是，最佳產品公司與其他公司在建造技術性產品上存在著重大差異，我不是指微小的差別，而是指從領導者行為到團隊授權水準的一切；甚至，團隊合作的模式；組織如何考量資金、人力及產品生產；組織文化；產品、設計和工程部門如何協同合作，為客戶找到有效的解決方案。

這本書被命名為《**使用者故事對照**》，但很快你就會發現，它所討論的內容遠超過這項強大卻簡單的技術，這本書直指核心，探討團隊如何協同合作，充分溝通，最後構思出要打造什麼好東西。

許多讀者從來沒有機會仔細觀察強大的產品團隊如何運作，你的經驗可能侷限於你的公司，或者以前工作過的地方，因此，我想要在這裡讓你感受一下最佳團隊與其他團隊有何不同。

感謝 Ben Horowitz 的好文章〈*Good Product Manager, Bad Product Manager*〉，下面簡單說明一下強大產品團隊與差勁產品團隊之間的重要差異：

> 好團隊懷抱著傳教士般的熱情，努力擘劃他們所追求的產品遠景。壞團隊只是一群討生活的傭兵。

好團隊從各種來源汲取靈感並且發想產品——從他們的 KPI 計分卡，觀察客戶的痛苦根源，分析客戶使用產品時產生的數據，並且不斷設法運用新技術解決實際的問題。壞團隊只從銷售額與客戶口中蒐集需求。

好團隊瞭解關鍵的利害關係人是誰，洞察這些利害關係人受到的限制，並且致力於發展務實的解決方案，不僅對使用者與客戶有效，並且謹守企業受到的限制。壞團隊僅從利害關係人身上蒐集需求。

好團隊擅長諸多技術，迅速試驗產品想法，決定哪些發想真正值得發展。壞團隊只是開會討論，產生排好優先順序的路線圖（roadmap）。

好團隊喜歡跟全公司的聰明意見領袖進行腦力激盪與討論。壞團隊在有外部意見介入時感覺被冒犯。

好團隊讓產品、設計，與工程人員並肩作戰，並且就功能性、使用者經驗，以及必要技術交換意見。壞團隊各擁山頭，要求其他人透過文件與會議的形式，請求他們的服務與協助。

好團隊為了創新不斷嘗試新想法，但仍嚴謹保護企業收益與品牌形象。壞團隊總是被動地進行試驗。

好團隊堅持掌握創造成功產品必定要有的技能，例如，卓越的互動設計（interaction design）。壞團隊甚至不曉得什麼是互動設計師。

好團隊確保工程師們每天都有時間試驗探索原型（discovery prototype），以便貢獻新想法，讓產品更卓越。壞團隊僅於衝刺規劃（sprint planning）期間讓工程師們觀看原型，以便進行評估。

好團隊每週與終端使用者和客戶直接聯繫，更確切地瞭解客戶，並且釐清客戶對最新想法的反應。壞團隊視客戶如無物。

好團隊瞭解有許多特別喜歡的想法最終都無法對客戶發揮實效，甚至那些能夠發揮實效的想法都需要經過不斷修正才能夠提供客戶想要的成果。壞團隊只建造路線圖上的東西，故步自封，侷限於交付日期與品質確保的最低要求。

好團隊理解迭代（iteration）的速度是創新的關鍵，並且瞭解這個速度源自於正確的技術，而非強制的勞動。壞團隊抱怨同事不努力，致使他們進度遲緩。

好團隊在評估需求並確認找到對客戶及企業實際有效的可行方案之後，會做出具有高度完整性的承諾。壞團隊抱怨自己身處於銷售導向的公司。

好團隊量測他們的工作成果，以便即刻瞭解產品的使用狀況，並且根據量測資料進行修正。壞團隊認為分析與報告「可有可無」。

好團隊持續整合及釋出新版本，瞭解連續不斷的較小釋出能為客戶提供更穩定的解決方案。壞團隊在痛苦的整合階段結束時一併進行測試，然後一次釋出所有的成果。

好團隊把焦點聚集在參考客戶（reference customers）身上。壞團隊把精力浪費在競爭者身上。

好團隊在他們對企業 KPI 產生重大貢獻時立即慶祝。壞團隊在最終釋出產品時才慶祝。

我瞭解你可能在想這一切跟故事對照有什麼關係，我想，好好讀下去，你將感到驚訝不已，而這正是我為什麼變成故事對照之愛好者的原因。

我所見過真正夠格，並且能夠實際幫助產品團隊提升到符合公司所需之層次的敏捷開發專家實在寥寥可數，Jeff Patton 正是其中之一。根據我的觀察，Jeff 與諸多團隊合作過，切身參與產品發掘（product discovery）的工作，並且完成許多棘手的任務。我把他介紹給一些公司，因為他能夠發揮實質效用。每個團隊都喜歡他，因為他不僅學識淵博，而且為人謙遜。

在過去，產品經理獨自蒐集並且整理需求，設計師忙亂地為產品增添無謂的光彩，工程師窩在地下室裡埋首編程，不食人間煙火，這種現象在最佳團隊中早已不復存在。現在，就將這些惡習從你的團隊中驅除吧。

— Marty Cagan
June 18, 2014

前言

> 在裡頭生活，在裡頭游泳，在裡頭歡笑，在裡頭相愛／清除床單上令人困窘的污漬，沒錯／招待來訪的親人，把三明治化作一場盛宴。
>
> — Tom Waits，〈Step Right Up〉

這本書應該是個小玩意兒…一本小冊子，真的。

我開始撰寫這本書，說明我稱之為**故事對照**（*story mapping*）的簡單實務。我跟許多同儕建造了簡單的**故事地圖**（*story map*），幫助我們與其他人協同合作，並且想像產品的使用經驗。

故事對照讓我們聚焦在使用者及其經驗上，促成更好的對話，最後建造出更棒的產品。

建造故事地圖是非常容易的。與其他人一起工作時，我會述說產品的故事，由左至右使用便利貼，寫下使用者在故事裡會採取的每個大步驟，接著，我們會回頭談談每個步驟的細節，並利用便利貼寫下每個步驟的細節，然後將它們沿垂直方向放在每個步驟下面，最後得到一個簡單的

格狀結構，從左到右地描述故事，並且由上而下地拆解成多個細節，既有趣又迅速。對敏捷開發專案來說，這些細節清楚地描述了故事的待處理項目。

撰寫關於這個主題的書籍能有多複雜？

然而，事實證明，簡單的事情也可能變得相當複雜。我花了許多篇幅，說明你為什麼會想要建立故事地圖，建造故事地圖時會發生什麼事，以及故事地圖的各種運用方式。這項工作的實踐確實超出我的想像。

如果你正在使用敏捷開發流程，你可能會以使用者故事填寫待處理項目（backlog）。我原本以為，因為使用者故事是很平常的實務，在這本書中描寫它們對我來說應該是浪費時間，但我錯了。在 Kent Beck 提出這個詞彙之後，十多年來，使用者故事變得比以前任何時候都更普遍，更受歡迎——但也更嚴重地被誤解及誤用，這讓我很難過，更且，它抹煞了我們從故事對照所得到的一切好處。

因此，在這本書裡，我想要盡可能匡正一些錯誤觀念——關於使用者故事本身，以及它們在敏捷與精實軟體開發（Agile and Lean software development）中被使用的方式。因此，我在前面引用 Tom Waits 的歌詞，將這件事比喻為「把三明治化作一場盛宴」。

為什麼是我？

我喜歡製造東西，動機是從中獲得快樂：我建立軟體，看見人們使用它並且受益於它。我是一個篤信勉強不會有幸福的方法論者，我發現我必須瞭解流程與方法如何運作，才能夠將它們掌握得更好。在經歷二十餘載的軟體開發生涯之後，我現在才在學習如何教導他人，告訴別人我的經驗，而且，我明白我所教授的內容仍是一個變動的目標（moving target），我的認知每個禮拜都在改變，詮釋它的最佳方式也幾乎以相同的速度在變動，這一切讓我在近幾年來遲遲不敢動筆撰寫這本書。

然而，是時候了。

使用者故事與故事地圖是很棒的觀念，許多人從中受益，讓日子過得更輕鬆，讓建造的產品更美好，然而，在某些人過得更好的同時，卻有更多人對這些觀念比以前都還要掙扎。我想要幫忙終止這個困局。

這本書是我可以提供的一點協助，如果它能夠做出貢獻，甚至只是對一些人，都會讓我感到歡欣鼓舞。我會好好慶祝。

如果你對使用者故事仍感不解，這本書正是為你而寫的

因為許多組織皆已採用敏捷與精實流程（Agile and Lean processes），以及它們所伴隨的使用者故事，你可能掉進一或多個因為誤解故事而造成的陷阱，就像這樣：

- 因為使用者故事讓你聚焦在建造小東西上，很容易**看不見整體圖像**（*big picture*），最後經常得到一種「科學怪人產品」，在當中，每個產品使用者都會清楚地看到它是由一些不相配的零件拼裝而成的。
- 當你建造有點規模的產品時，一個小東西接著一個小東西地建造，會讓人們**搞不清楚何時將結束，或者你究竟要交付什麼產品**。如果你是建造者，同樣也會感到很疑惑。
- 因為使用者故事關乎對話，**人們利用這個觀念避免寫下任何事情**，接著，他們忘記自己在對話過程中討論及同意什麼。
- 因為好的使用者故事應該有驗收標準（acceptance criteria），我們聚焦於撰寫驗收標準，但對於需要建造什麼還是沒有共同的瞭解，結果，**團隊未能在時限內完成他們計畫的工作**。
- 因為好的使用者故事應該從使用者的觀點來撰寫，而且有很多部分是使用者看不到的，團隊成員會辯稱，「**我們的產品沒有使用者，所以使用者故事在這裡不會發揮效用**。」

如果你已經落入任何一個陷阱，我會試著在第一時間就釐清造成那些陷阱的誤解。你將學習如何思考整體圖像，如何以綜觀全局與鉅細靡遺的方式進行計畫與估計，如何針對使用者試圖達成的目標進行建設性的對話，以及好軟體需要做什麼才能夠幫助使用者。

誰應該讀這本書？

當然，你應該讀這本書，尤其是如果你已經買了。我個人認為你的投資是明智的。如果你只是借閱這本書，現在就應該自己訂一本，並且在收到新書時，把借閱的那本給還了。

無論如何，這本書值得擔任特定角色的從業人員好好閱讀，並且一定能夠從中獲益：

- **商用產品公司的產品經理與使用者經驗設計師**應該閱讀這本書，以消除在思考整體產品與操作體驗之間的鴻溝，以及在思考戰略計畫與待處理項目之間的裂隙。如果你一直卡在你所想像的遠景與團隊能夠建造的細節之間，故事地圖絕對有幫助。如果你一直在費盡心思幫助其他人想像產品的操作體驗，故事地圖確實能夠幫上忙。如果你一直在奮力釐清如何整合良好的操作體驗與產品設計實務，這本書必然可以幫助你一臂之力。假如你一直在設法整合精實創業風格（Lean Startup–style）的實驗，這本書無疑是你的絕佳幫手。
- ***IT* 產業的產品負責人、商業分析師與專案經理**應該閱讀這本書，以幫助內部使用者、利害關係人與開發者跨越彼此之間的隔閡。如果你一直在掙扎如何讓公司的眾多利害關係人達成共識，故事地圖一定能夠幫上忙。如果你一直在努力幫助開發者看見整體圖像，故事地圖也必然能夠發揮效用。
- **肩負著提升個人與團隊之責任的敏捷與精實流程指導員**應該閱讀這本書，並且，在閱讀過程中，好好想想你的組織對使用者故事有多少誤解。運用這本書描述的使用者故事、簡單練習及實務經驗，幫助你的團隊提升到更高的層次。
- **所有其他人**。使用敏捷流程時，我們經常指望產品負責人與商業分析師掌握大多數與使用者故事有關的工作。然而，有效運用使用者故事需要每個人都具備一定的基本功，當人們不瞭解相關基礎知識時，你會聽到一些抱怨：「使用者故事寫得不好」、「它們太龐大」，或者「缺乏足夠細節」…等，這本書會有幫助，但不是以你認為的方式。你和所有其他人都將學到，使用者故事不是為了寫出更清楚的需求（requirements），而是為了組織及進行更好的對話（conversations）。這本書會讓你明白應該進行何種對話，以便獲得需要的資訊。

我希望你屬於以上描述的一或多個群組，如果不是的話，就將這本書送給需要的人吧。

如果是的話，讓我們開始探索吧。

本書慣用體裁

我料想這不是你唯一讀過的軟體開發書籍，所以不應該有什麼東西令你驚訝不已。

每一章裡的大小標題可以引導你探索該主題

利用它們找到方向，或者即刻略過你不感興趣的題材。

關鍵重點會以這種格式呈現，請想像我比其他文字更大聲地朗誦這類文字。

如果你正在快速瀏覽這些關鍵重點，假如你喜歡它們，或者感覺上不是那麼簡單直白，請閱讀前後段落，應該就會更清楚些。

附帶說明區塊（sidebar）被用來描述：

- **有趣但非關鍵性的觀念**。這些應該是好玩的花絮，至少我希望如此。
- **特定實務的要訣**。你應該能夠利用這些要訣，幫忙展開特定實務操作。
- **其他人貢獻的故事和範例**。你應該從這些材料中汲取一些好觀念，並且試著應用在你的組織中。

這本書被組織成幾個部分，你可以一次閱讀一個部分，或者運用幾個特定部分，幫助你找出解決當前所面臨之挑戰的具體想法。

本書組織

前陣子，我新買了一台彩色雷射印表機，打開包裝，印表機上面有一本小冊子，上頭用紅色字母寫著 "Read This First"（請先讀我），我心裡想，「我**真的**應該先讀這本小冊子嗎？」因為我通常不那麼做，但是，還好我有先讀，因為裡頭好多地方都有塑膠保護件，確保印表機在運送期間安全無虞，而且，如果在插上電源之前，沒有先移除它們，可能會損傷印表機。

這個故事聽起來好像有點離題，實則不然。

這本書包含「請先讀我」的章節，介紹我將在整本書裡使用的兩個關鍵概念與相關詞彙，在進入正題以前，希望你將那些概念深植腦海，如果你在瞭解它們之前，逕自探索使用者故事對照，我就不敢保證你的安全。

使用者故事對照的整體概廓

第 1 ～ 4 章將以高階觀點介紹故事對照，如果你已經運用使用者故事一段時間，並且嘗試過故事地圖，這個部分應該會提供足以讓你即刻出發的知識。

第 5 章提供一個很棒的練習，幫助你學習用來建立良好故事地圖的核心觀念。跟你的團隊一起試試，每個參加者都能夠獲益良多，而且我保證，他們為你的產品所建立的故事地圖稍後必能產生更好的成果。

徹底瞭解使用者故事

第 6 ～ 12 章說明使用者故事的箇中奧妙，實際運作，以及如何將它們運用在敏捷與精實專案中。故事地圖裡有許多能夠用來驅動日常開發的小故事，即使你是一位敏捷開發老手，相信我，你一定會學到一些關於使用者故事的新知識；如果你對使用者故事不熟悉，我保證，你必然會學到足夠的知識，讓辦公室裡自認為對敏捷開發瞭若指掌的人們感到驚訝不已。

更好的待處理項目

第 13 ～ 15 章深入使用者故事的生命週期，我將探討幫助你運用使用者故事與故事地圖的具體實務，從機會（opportunities）開始，一直到識別出待處理項目（backlog，內含描述可行產品（viable product）的各個使用者故事）的發掘工作（discovery）。你將瞭解故事地圖與諸多其他實務如何幫助你穩當地走過產品開發的每一個步驟。

更好的建造

第 16 ～ 18 章進一步深入，策略性地運用使用者故事，一個迭代接著一個迭代（iteration）或者一個衝刺接著一個衝刺（sprint）。你將學習如何準備使用者故事，在建造它們時，全心關注，確實進行，並且從即將轉變成有效軟體的每個使用者故事中學習一些東西。

我發現，許多軟體開發書籍的最後幾章都是廢話，通常可以忽略，可惜，這本書沒有那種章節，你必須通讀全書，我只能安慰你，每一章保證都能夠讓你學到一些可應用在工作上的有用知識。

讓我們開始吧。

請先讀我

本書沒有簡介。

是的，你沒看錯。你現在可能在想，「Jeff 的書為何沒有簡介？他是忘了嗎？經過這些年他的腦袋開始退化了？還是被狗狗啃掉了？」

不，我沒忘記要撰寫簡介，我也還沒開始老化，至少我不這麼認為，而且，我的狗狗也沒吃掉它（雖然我女兒的天竺鼠有點可疑），純粹是因為，我一直認為許多作者花太多時間說服我應該讀他們的書，那些書籍的簡介裡充斥著這類話語，大多數書籍直到第 3 章才開始端上真材實料，我通常跳過簡介，而且我相信，絕對不只我這麼做。

事實上，這本書就從這裡開始。

你不可以略過這個章節，因為它真的是最重要的部分。事實上，如果你只從這本書學到兩個重點，我也會感到欣慰，而且，那兩個重點就在這個章節裡：

- 採用使用者故事的目標不是為了寫出更好的使用者故事。
- 產品開發的目標不是為了製造產品。

請容我解釋。

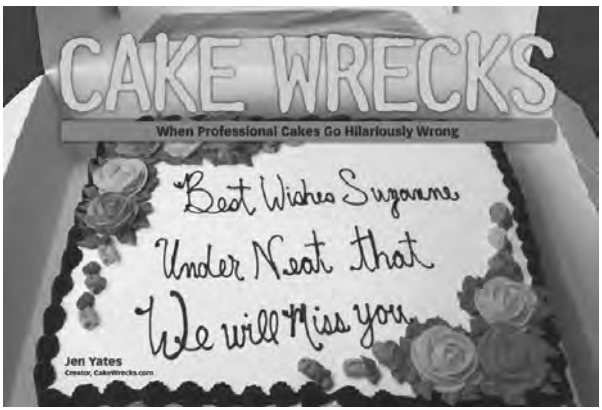
電話遊戲

我相信，你還記得小時候玩的「電話遊戲」（telephone game），在當中，你以耳語的方式告訴某人某事，他再低聲將這個訊息傳達給另一個人，

直到最後一個人說出完全被弄擰的資訊，每個人就笑得不可開支。到今天，我們家還圍著餐桌玩這個遊戲呢。各位家長請注意：這個遊戲是吸引孩子們的好活動，避免他們對大人的晚餐對話感到百般無聊。

在成人的世界裡，我們持續沉浸在這個遊戲裡——只是沒有彼此竊竊私語，我們撰寫冗長的文件，建立看起來非常正式的簡報，將這些文件和簡報交給某人，他再從中擷取出完全不同於我們意圖傳達的訊息，並且使用這些文件創造出更多文件，再交給其他人，然而，跟小孩子玩遊戲不同，最後我們都笑不出來。

當人們閱讀書寫型式的「指示」（instruction）時，會以不同的方式詮釋它，如果你覺得這有點不可思議（畢竟白紙黑字！），那就來看幾個「指示」完全被搞錯的例子。



這是 Jen Yates 所著的《*Cake Wrecks*》（Andrews McMeel Publishing）一書的封面[譯註]（感謝 Jen 與 John Yates 提供），這本書源自於 Jen 所建立的有趣網站，*cakewrecks.com*（蛋糕殘骸）。假如你沒有至少一小時的時間可浪費，請勿隨意參訪該網站。該網站展示一些裝飾奇特、趣味橫生的蛋糕照片——這些蛋糕迥異於傳統，並且包含許多令人莞爾的趣事。現在，在該網站與書籍中一再出現的主題之一是遭到曲解的「需求」（requirements），不過，她當然不稱它們為**需求**，因為那是一種書呆子

[譯註] 在這張照片中，"Under Neat that" 是 "Underneath that"（在那下面）的誤植，亦即，客人要蛋糕店在第一行下面寫下 "We will miss you"，但蛋糕師傅卻如實地將 "Underneath that" 也寫出來，而且還拼錯。不過，拼錯也好，否則，"Underneath that, We will miss you!" 不知是否會讓人引發更多不當的聯想。

用語，她管它們叫作**按字面意義解讀**（*literals*），因為讀者們會直白地理解被寫下的蛋糕題字。在看這些照片時，我能夠想像某個店員正在傾聽並且記下客人需要的蛋糕題字，接著把它交給另一個負責裝飾蛋糕的師傅。

客人：你好，我要訂蛋糕。

店員：好的，你想要在上面寫什麼？

客人：可以在上面寫下紫色（purple）的 "So long, Alicia"（再見，艾麗西亞）嗎？

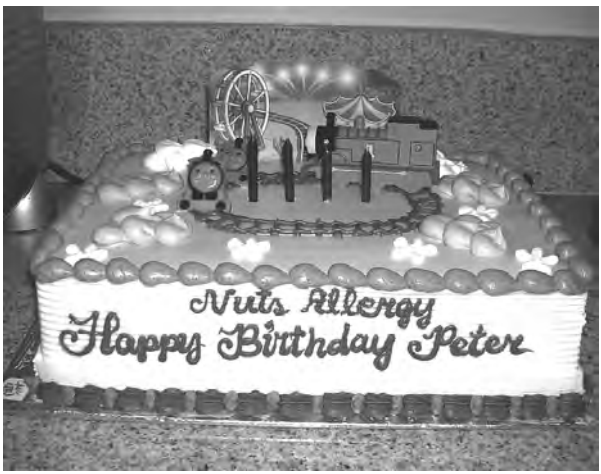
店員：當然可以。

客人：周圍再加一些星星（stars）好嗎？

店員：沒問題，已經記下來了，馬上交給我們的蛋糕裝飾師傅，明天早上就可以拿了。

這是最後的結果：

下面是另一個好笑的例子[譯註]。在軟體開發中，我們稱這些為**非功能性需求**（*nonfunctional requirements*）：



這些都是很有趣的例子，而且，浪費 20 美元在蛋糕上，實在無傷大雅，我們多能一笑置之，但有時候，其中的利害關係可不僅如此。

你可能聽過價值 1.25 億美元的 NASA 火星氣候探測者號（NASA Mars Climate Orbiter）在 1999 年發生的事故[1]，嗯，或許你沒聽過，不過，關鍵如下，NASA 的專案經常被淹沒在大量的需求與說明文件中。然而，儘管檔案櫃裡滿是需求與說明文件，探測者號還是墜毀，因為 NASA 使用公制單位進行量度，而 Lockheed Martin 的工程團隊針對火箭推進器的導航指揮系統使用的卻是英制單位。雖然沒有人確切知道探測者號最終到哪兒去，但某些人認為它已經越過火星，在某處愉快地繞著太陽運行。

諷刺的是，我們把相關資訊寫下來，企圖更清楚地傳達並且避免誤解，卻往往事與願違。

分享文件不是分享共識。

1 有許多文章試圖描述火星氣候探測者號出了什麼問題，其中一篇為：*http://www.cnn.com/TECH/space/9909/30/mars.metric.02/*。

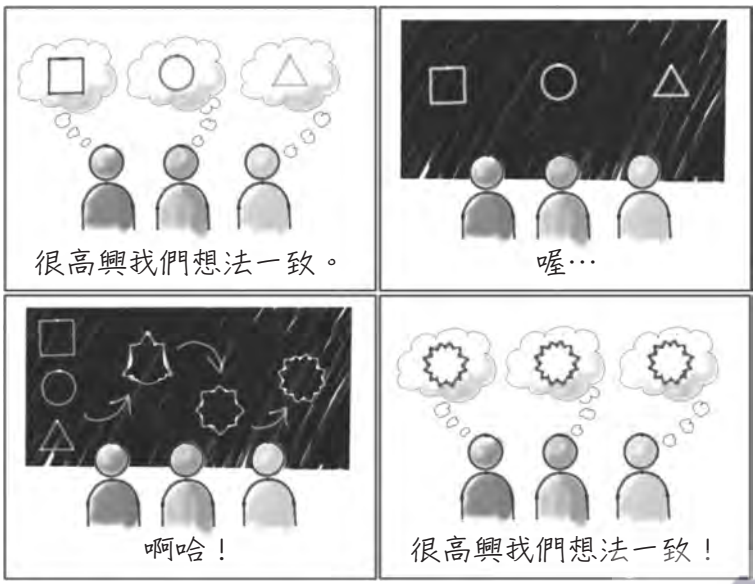
[譯註] 在這張照片中，"Nuts Allergy" 意指「對堅果過敏」。

稍停一下，把這句話記下來，寫在便利貼上，並且放進你的口袋。可以考慮將它刺在身體某處，以便在早上準備開始工作時看到它。當你讀到這句話時，它會提醒你我現在正在述說的故事。

共識（*shared understanding*，**或共同的理解**）是指我們瞭解彼此在想什麼及為什麼。很明顯地，在蛋糕裝飾師傅與以書面形式提供題字資訊的人之間並沒有共識。另外，在 NASA，某個重要人物與負責制導系統（guidance system）的其他人之間也缺乏共識。我相信，如果你已經從事軟體開發一段時間，必然對這種情況印象深刻：兩個人相信他們對於要為軟體增加什麼功能意見一致，但稍後卻發現彼此的想像存在著極大的落差。

建立共識是具破壞性的單純化？

我的前同事，Luke Barrett，最早以漫畫描繪這個問題，我問他最初在哪裡看到這個想法，但他不記得了，因此，這份榮耀只能歸於某個地方的某個人。有好幾年的時間，我看到 Luke 以投影片的方式逐步說明這個四格漫畫，但我當時只是覺得有趣，而沒有特別重視，顯然，我是一個呆瓜，我花了好幾年的時間才瞭解，這個漫畫描繪出在軟體開發中運用使用者故事的最大關鍵。



觀念是，如果我的心裡有某個想法並以書面形式描述它，在你閱讀那份文件時，你想的跟我講的很可能不一樣。我們甚至會詢問每個人，「你們都同意這裡所寫的東西嗎？」，大家可能都說，「是的！是的！」。

然而，如果我們聚在一起交談，你可以告訴我想法，我能夠詢問你問題，如果可以透過畫圖，或者使用索引卡或便利貼組織想法，具體呈現我們的思維，溝通就會更順暢。如果允許彼此花時間用文字與圖像來解釋想法，我們就能夠順利地建立共識。不過，這通常發生在我們發現彼此對事情具有不同的理解時，真是糟糕，但至少我們現在明白這一點。

這無關於某人是對或錯，而是我們全都看到不同且重要的面向。透過結合及精煉不同的想法，我們最終能夠得到涵蓋所有最佳想法的共同理解。這就是將我們的想法具象化為何如此重要的原因。我們可以重新繪製草圖或者四處移動便利貼，最酷的是，我們真的在四處移動我們的想法，真的在逐步演進我們的共識，那是單靠文字幾乎不可能達成的任務。

在離開這個對話時，我們可能還是在處理相同的功能或強化，然而，我們現在真正達成共識，我們感覺認知一致，並且深信大家正一同向前行。那就是我們勉力設法獲得的結果。很遺憾地，那是無形的東西，你看不到也摸不著「共識」，但是你能夠感受它。

別再試圖撰寫完美文件

很多人相信有某種理想方式可用來準備文件——當人們閱讀文件並且得到不同理解時，不是閱讀者的錯誤，就是撰寫者的失敗。其實，以上皆非。

答案就是停止做傻事。

別再試圖撰寫完美的文件。

繼續撰寫資訊，任何資訊，然後透過文字與圖像，利用有效溝通來建立共識。

運用使用者故事的真正目標是達成共識。

敏捷開發的使用者故事得名於它們應該如何被使用，而不是你們寫下什麼資訊。如果你們在開發過程中運用使用者故事，但沒有利用文字與圖像一起進行對話，那麼，你們的做法是有問題的。

如果你閱讀這本書的目標是為了學習如何撰寫更好的使用者故事，你顯然搞錯方向了。

好文件就像度假照片

如果我讓你看我的度假照片之一，你可能看到我們家的小朋友在沙灘上，並且禮貌性地說，「哇，真可愛」，然而，當我觀看我的度假照片時，我會回憶起夏威夷某個特別的海灘，我們得開著四輪傳動的車子，花一個多小時，在荒野小路上留下極深的車轍，接著再花半小時走在熔岩地形上，才能到達那個沙灘。我記得孩子們在碎碎念，抱怨沒有什麼海灘值得這麼大費周章，連我自己都這麼想。但事實證明，一切辛勞全都值得，我們在天堂般的無人海灘上，度過了幸福的一天，那正是我們花了那麼多工夫設法到達那裡的原因。海龜現身沙岸，悠閒地曬著太陽，為那個美麗的日子增添幾許慵懶。

當然，單看這張照片，你不會瞭解這一切，因為你並未在場，而我記得每一個細節，因為我身歷其境。

不論好壞，這就是文件資料實際的運作模式。

如果你參與大量關於要建造什麼軟體的討論，然後建立文件釐清它，你可能與另一個參與者分享它，你們兩人可能都同意它是不錯的，但記住，你們的共識中有許多細節並未被描述在文件裡，另一個未親身參與討論的讀者不會從中得到跟你們一樣的認知，即使他說已經充分瞭解，你也不要相信。大家聚在一起，使用該文件述說蘊涵於其中的故事，就像我使用度假照片向你描繪我的故事那樣。

文件幫助記憶

我聽過人們開玩笑地說，「採取敏捷流程是因為我們已經停止撰寫文件」，明白人都知道這只是一句玩笑話，因為故事驅動的流程需要許多文件才能夠運作，但那些文件看起來完全不像傳統的需求文件（requirements document）。

我們需要對話、畫草圖與撰寫文字，並且使用便利貼或索引卡，我們必須將這些文件帶進對話裡，使用螢光筆標示它，利用註解說明它，整個過程充滿互動與能量。如果你們只是坐在會議桌旁，由某個人將你們所說的話鍵入故事管理系統中，你們可能錯失真正的核心精神。

當你在述說故事時，任何東西大都能夠被用來作為溝通的工具，而且，隨著我們講述這些故事，撰寫大量註解及描繪許多圖片，我們必須將它們保存下來。我們隨身攜帶並且檢視它們，為它們拍下照片，並且重新輸入成為更多文件。

但記住，最重要的事情不是什麼被寫下來——而是當我們閱讀它時會記得什麼，就像度假照片那樣。

對話、畫草圖、撰寫文字、使用便利貼和索引卡，然後將你們的成果拍成照片，甚至，將你們透過白板進行溝通的過程全程錄影。你們會非常深刻地記住許多細節，那是單單文件絕不可能辦到的。

為了幫助記憶，
將你們的對話用照片或影片記錄下來。

談談對的事

許多人相信他們的工作是收集與傳達需求，實則不然。

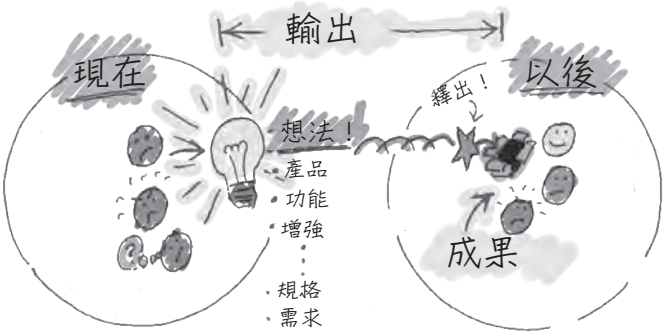
事實上，你的工作是改變世界。

是的，我這麼說是為了吸引你的注意，而且，沒錯，我知道這聽起來蠻誇張的，那是因為這句話通常與世界和平、消弭貧窮，或甚至讓政治人物意見一致（遙不可及的目標）相關聯，但我是認真的，你為產品解決方案所提出的每一個絕妙想法皆以某種程度改變這個世界，影響使用該產品的人們。事實上，如果沒有這樣的話，你就失敗了。

現在和以後

有一種改變世界的簡單模型是我個人愛用並且總是謹記於心的，當你在進行故事對話及建立共識時，你也必須將它深植於腦海裡。

我將該模型繪製如下：



該模型從檢視這個世界的現況開始，當你這麼做時，你會發現人們不爽、困惑、火大或沮喪，但天地何其大，因此，我們把焦點聚集在我們所製作之軟體的使用者身上，或者我們希望將使用該軟體的人們。當你檢視他們在做什麼時——以及他們使用的工具和他們是怎麼做的——你會產生一些想法，這些想法可能是針對：

- 你能夠建造的全新產品
- 增添到既有產品的功能
- 已建造之產品的增強（enhancement）

在某個時點，你必須將你的想法細節傳達給其他人，你可能開始撰寫一些設計（design）和規格（specification），如果你打算將這些東西交給其他人，你可能會將所有這些細節稱作你的**需求**（*requirements*），但切記，需求只是「能夠幫助人們的想法」的別稱。

鑒於那些需求，我們經歷最終促成產品交付的某個流程，建立實際運作的軟體，**並在以後對世界產生實質的影響**。我們希望原本不開心、困惑、火大或沮喪的人們在我們的軟體面世時能夠變得輕鬆愉快。他們現

在不開心是因為缺乏可用的軟體，或者既有的軟體根本不適用。在使用你所建造的軟體、網站、行動 app 或任何東西之後，他們會以不同的、更好的方式處理事情——這就是讓他們變開心的關鍵。

事實上，你無法總是讓每個人都滿意。你的母親早該告訴你這個道理。有些人會比其他人更滿意你的產品，有些人可能還是不開心，無論你多麼努力、或是你的產品多麼讓人感到驚艷。

軟體不是重點

在想法（idea）與交付物（delivery）之間的一切被稱作**輸出**（*output*），那是我們建立的東西。敏捷軟體開發會刻意量測輸出的**速度**（*velocity*），並且試圖增加它們的輸出速率。因為人們正在建造軟體，他們當然會在意建造物的成本與完成速度，理應如此。

雖屬必要，但輸出不是真正的重點；我們真正想要的不是輸出，輸出只是附帶的產物，我們要的是**成果**（*outcome*），成果是事情的結局，很難掌握，因為直到最後才能夠被量測，而且，我們不根據交付的功能數量、或是人們現在能夠做什麼來量測成果，我們量測的是你建造的東西對人們達成目標的方式產生什麼影響，而最重要的是，你是否讓他們的生活變得更加美好[2]。

就是這樣，你改變了世界。

你已經為世界做出貢獻，改變人們達成目標的方式，而且，當人們運用你的產品時，世界也因為它們而改變。

切記，你的目標不只是打造新產品或新功能，當你針對這個產品或功能進行對話時，你會談到它是針對誰、這些人目前在做什麼，以及事情之後會如何因為它而改變。之後的正向改變才是他們想要它的真正原因。

良好的故事對話關係到人與為什麼，
而不只是故事內容。

2 Robert Fabricant 在〈Behavior Is Our Medium〉（*http://vimeo.com/3730382*）的談話中對**輸出**（*output*）與**成果**（*outcome*）這兩個詞彙做了清楚的區分，在那之前，我對這兩個術語感到混淆，其他人也一樣，很高興，Robert 的頭腦相當清楚。

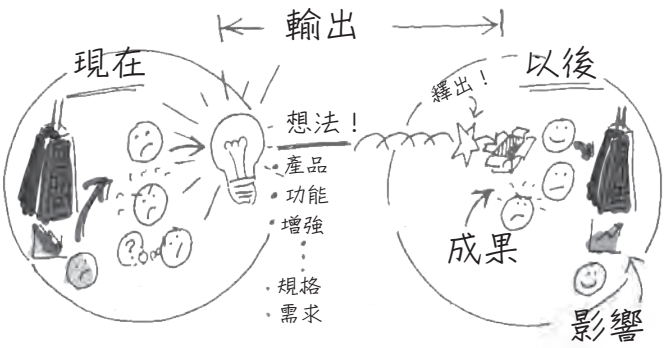
好吧，不只關乎人

我跟其他人一樣關心人們，但是說實話，那不止關乎讓人們感到開心。如果公司付你跟其他人薪水，你們必須聚焦在最終能夠幫公司賺更多錢、保護或擴展市場、或讓公司高效營運的事情上，因為，若是公司不健全，你們就不會有資源（或工作），而能夠幫助任何人。

因此，我必須稍微修正這個模型。它實際上從檢視你的組織內部開始，在那裡，你甚至會找到更多不快樂的人，而且，這通常是因為企業的運作不符合他們的期望。為了修正這個問題，他們可能會想要聚焦在特定客戶或使用者身上，並且建立或改善他們正在使用的軟體產品。事實上，其中的牽連是相當深遠的，因為：

你的公司無法得到它想要的東西，
除非客戶與使用者得到他們想要的東西。

透過選擇要聚焦的人們，要解決的問題，以及能夠轉換成有效軟體的想法，這個流程繼續走，並且，從那裡開始——假如客戶購買，使用者使用，而且人們開心——贊助這項開發的企業最終會看到它所追尋的利益，那會完全反映在增加的營收、較低的營運成本、更愉快的客戶，或擴大的市占率上。這會讓公司裡的許多人感到開心，也會讓你覺得快樂，因為你幫助公司保持健全，同時在過程中讓人們生活得更好，營造出雙贏的局面。



那是較長期的東西，發生在良好成果出現之後，我稱之為**影響**（*impact*）。成果通常是交付之後立刻能夠觀察的東西，影響則需要比較長的時間才能釐清。

建造較少東西

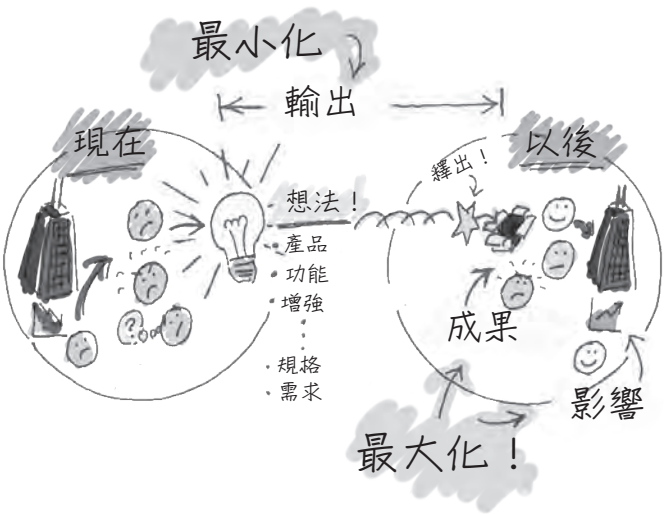
關於軟體世界，有一個令人相當不舒服的事實，我懷疑它在諸多其他地方也存在。不過，我瞭解軟體，我明白：

我們要建造的東西總是超過我們擁有的時間或資源——屢試不爽。

軟體開發的常見誤解之一就是我們試圖更迅速地得到更多輸出，因為，假如有太多事情要做，更快速地進行會有幫助，對吧？然而，如果你把遊戲規則弄清楚，就會瞭解，你的任務不是要建造**較多東西**，而是要建造**較少東西**。

讓輸出最小化，讓成果與影響最大化。

到頭來，你的任務是讓輸出**最小化**，並且讓成果與影響**最大化**。訣竅在於，你必須密切注意你試圖為他們解決問題的人們，包括選擇購買你的軟體來解決其組織問題的人（**選擇者**），以及使用它的人（**使用者**）。有時候，他們是同一批人，有時不是。



你的企業有許多能夠聚焦的潛在使用者與客戶，你的企業策略應該指導你要聚焦在誰身上，才能夠得到你想要的影響。我敢保證，沒有任何企業具有足夠的資源讓每個人都滿意——那是不可能的事情。

別誤會，更迅速地建造更多軟體總是好主意，然而，那永遠解決不了問題。

關於需求

在我的軟體生涯裡，幾乎整整前十年，我為傳產零售商打造軟體，我僥倖避開**需求**（*requirements*）這個詞彙——至少，不太用。對我當時所做的事情來說，那並不是一個重要的術語。我擁有許多不同的客戶，全都對於什麼能夠幫助他們抱持著具體的想法，我也知道我所效命的公司必須靠販售我的產品來賺錢。事實上，我花了很多時間參加商展，幫助公司把它的產品販售給各種客戶，最後，我明白，在將我與團隊開發的產品交付出去之後，我必須繼續跟客戶合作，所以，我勤勉工作，盡力為他們爭取最大利益。那意味著，我其實無法提供每個人他們想要的東西，因為每個人想要的都不一樣，且公司與團隊的時間和資源有限，因此我必須努力釐清讓各方人馬均能滿意的最低限度，這聽起來或許有點讓人挫折，但其實這部分還蠻有趣的。

隨著公司成長，我們增加更多的傳統軟體人力，某一天，另一個團隊的領導者跑來跟我說，「Jeff，我需要你們針對正在進行的產品做這些改變。」

我說，「好的，沒問題，告訴我這些改變是針對誰，並且為他們解決什麼問題。」

她回答，「這些是需求。」

我說，「我知道，但請說明一下它們是針對誰，這些人如何利用這些改變，以及這些改變如何影響他們的工作。」

她看著我，就好像我是白癡一樣，並且十分肯定地再告訴我一次，「這些是**需求**。」

就在那個時候，我明白**需求**這個字實際上的意思就是**閉嘴**。

對許多人來說，那正是需求的功用，它們阻止我們針對人們以及要解決的問題進行對話。事實上，即使你只針對一小部分需求進行建造，仍然可以讓人們感到非常愉快[3]。

記住：說到底，你的任務不是實現需求，而是改變世界。

總結

如果你沒能從這本書獲得其他利益，請至少記住這些事情：

- 使用者故事不是需求的書面形式；利用文字與圖像，透過協同合作述說故事是建立共識的有效機制。
- 使用者故事不是需求；而是為我們的組織、客戶、和使用者解決問題的相關討論，產生關於要建造什麼的共識。
- 你的任務不是更迅速地建造更多軟體：而是最大化從你選擇建造的東西中得到的成果與影響。

使用者故事意在提供完全不同的機制，讓我們思考協同建立軟體時所面臨的挑戰——以及許多其他相關事宜。如果你們能夠有效率地協同合作，建立真正可以解決問題的產品，你們將征服世界，或至少在市場上佔有一席之地。

當你閱讀這本書時，希望你回歸到使用者故事的基礎。我期盼你跟其他人協同合作，述說關於使用者與客戶以及你們如何幫助他們的故事，我希望你繪圖並且建立大型的便利貼模型，我期望你感受到整個團隊全心參與並且充滿創造性，我冀望你感覺到自己正在創造新氣象，因為當你正確運用使用者故事時，必定會有一番作為，而且，一切也會變得更有趣。

現在，讓我們來談談關於使用者故事以及故事對照的箇中奧妙。

3 因為我非常同意這個觀點，所以在此重申 Kent Beck 於《*Extreme Programming Explained*》（Addison-Wesley）中對需求一詞被誤用所提出的警告。