第三章

敏捷原則

如果你問人們需要什麼，他們會回答說一匹更快的馬。
——亨利·福特[1]

沒有一帖秘方就能夠每次都打造出完美軟體這回事。敏捷團隊知道這點。相反地，他們有的是概念和基本原則，可以幫助引導團隊做出正確決定並避免犯錯（或者是去處理那些在無可避免的情況之下發生的問題）。

我們已經在前面的章節中看過敏捷宣言的 4 個價值了。除了那些價值之外，尚有 12 條原則是所有敏捷實踐者應該運用在專案團隊上的。當那 17 位敏捷宣言的原始簽署者待在美國猶他州的雪鳥飯店時，他們很快就想出了 4 個價值。但是要發想出 12 條附加的原則卻花了很長的時間。敏捷宣言的簽署者 Alistair Cockburn 回憶道：[2]

> 我們一群 17 人很快就同意那些價值的挑選。在那次會議剩餘的時間裡，我們都在發展下一階段的陳述說明。最後由那 4 個價值構成了現在看到的這 12 條原則。
>
> 這些陳述說明應該隨著人們對於這些句子的感受而演化。如果在本書付梓時這些陳述說明還沒有過時，我一定會非常驚訝。關於這 12 條原則的最新內容，請見敏捷大會（*http://www.agilealliance.org/*）。

1 雖然有些人質疑亨利·福特是否真的說過這些話，但是幾乎所有人都同意他會喜歡這樣的講法。

2 Alistair Cockburn,《*Agile Software Development: The Cooperative Game*》第二版（Boston: Addison Wesley, 2006）。

Alistair 是對的，事實上，網站上關於這些原則的描述已經與他書上的稍有不同了。描述文字或許會持續演變，但是敏捷概念和原則卻是不變的。

在本章中，你將學到敏捷軟體的這 12 條原則：它們的內容是什麼、為什麼你需要它們，以及它們對你的專案有何影響。透過實際的範例，你將會看到這些原則如何運用在真實的專案上。為了方便學習，我們將這些原則分成 4 類（交付、溝通、執行、改善），因為個別分類裡面的原則和敏捷中心思想是一致的。不過雖然加以分類會比較容易學習，但是每個原則都有各自代表的意義。

敏捷軟體的 12 條原則

1. 我們最優先的任務，是透過及早並持續地交付有價值的軟體來滿足客戶需求。
2. 竭誠歡迎改變需求，甚至已處開發後期亦然。敏捷流程掌控變更，以維護客戶的競爭優勢。
3. 經常交付可用的軟體，頻率可以從數週到數個月，以較短時間間隔為佳。
4. 面對面的溝通是傳遞資訊給開發團隊及團隊成員之間效率最高且效果最佳的方法。
5. 業務人員與開發人員必須在專案全程中天天一起工作。
6. 以積極的個人來建構專案，給予他們所需的環境與支援，並信任他們可以完成工作。
7. 可用的軟體是最主要的進度量測方法。
8. 敏捷流程提倡可持續的開發。贊助者、開發人員及用戶應當能不斷地維持穩定的步調。
9. 持續追求卓越的技術與優良的設計，以強化敏捷性。
10. 精簡（或最大化未完成工作量之技藝）是不可或缺的。
11. 最佳的架構、需求與設計皆來自於能自我組織的團隊。
12. 團隊定期自省如何更有效率，並據之適當地調整與修正自己的行為。[3]

3 來源：*agilemanifesto.org/iso/zhcht/principles.html*（截至 2015 年 4 月時的版本。）

客戶永遠是對的……真的嗎？

翻回本章最前面，重讀那段引言。亨利·福特究竟想說什麼呢？他想表達的是，給予人們真正想要的東西，而不是他們要求的東西。客戶有其需求，假使你正在建構可以符合他們需求的軟體，那麼你必須理解他們的需求——無論你是否有辦法跟客戶一起溝通討論這些需求。你要如何跟在專案初期尚無法明確說出需求的客戶共事呢？你要如何知道他要的其實是汽車，而不是更快的馬兒呢？

這就是那 12 條原則背後的動機：讓團隊建構出用戶實際需要的軟體。這些原則根基的概念是：我們建構專案，以交付**價值**。不過「價值」這個字眼有點微妙，因為每個人在軟體中所看到的價值都不同：不同的人對於軟體有不同的需求。

你手邊可能有一堆關於這個看法的例子。如果你是透過手持式電子書閱讀器在閱讀本書的話，那麼你便是正在利用一套能夠將電子書呈現在閱讀器上的軟體。試著思考一下所有關鍵關係人（對於電子書閱讀器有所需求的人士）對於這套電子書閱讀軟體的可能想法：

- 身為讀者，你希望軟體能夠讓電子書容易閱讀。你在乎的是能夠前後翻頁、畫線或註記心得、搜尋，還要記住你上次看到哪一頁了。
- 身為作者，我們非常關心我們撰寫的每字每句都要能夠正確顯示，條列式的地方都有被縮排成讀者容易閱讀的形式，讓讀者可以很容易地在本文中找到這些條列式的內容，提昇讀者的閱讀體驗，讓讀者可以享受學習的樂趣。
- 我們 O'Reilly 的編輯關心的是能夠很方便地將書籍送到讀者手中，然後如果讀者喜歡這本書的話，也可以很容易地留下正面評論或者購買 O'Reilly 出版的其他書籍。
- 書店希望讀者能夠很容易地瀏覽和購買其他正在打折的書籍，而且快速簡便地下載到讀者的閱讀器上。

你或許可以想到更多的關鍵關係人，以及他們所關心的事項。他們所關心的每件事都代表著一項他們希望被賦予的價值。

市面上出現的早期第一批電子書閱讀器顯然並不符合上述所有需求。花了好長的一段時間，那些閱讀器上執行的軟體才演化成今日的模樣。而且毫無疑問地，隨著開發團隊不斷發現交付新價值的方法，這類型的軟體只會變得越來越好。

電子書閱讀器提供的價值顯而易見，因為這是事後諸葛。要在專案初期就看出價值通常難多了。讓我們很快地做個思考實驗來驗證看看吧。試著思考這個問題：有多少個閱讀器是用瀑布式流程開發出來的呢？

照我說的做

想像你們團隊正在建構早期第一批電子書閱讀器。硬體團隊交付了一台原型裝置，有一個可以用來載入電子書的 USB 插槽，還有一個用於互動的小型鍵盤。你們團隊的責任是寫一軟體，將電子書顯示出來讓用戶閱讀。

不幸的是，你們公司長久以來都使用一種很沒有效率的方式在建構軟體，叫做「預先最大需求」（Big Requirements Up Front）的瀑布式流程。所以專案經理要做的第一件事，就是把所有找得到的人都抓進來一起開一個大型的會議。整個團隊接下來有好幾週的時間都必須耗在會議室裡，每次與會的對象可能不一樣，有時候是你們公司的資深經理，有時候是友好出版商（想出版一些電子書，以便在你們的閱讀器上面顯示）的代表，有時候是線上零售商的資深銷售人員（他們也想販賣電子書），以及許許多多你們專案經理可以找來一起開會的關鍵關係人。

在經過數日密集的會議和激烈的討論之後，你們公司的商業分析師於是比較能夠從不同關鍵關係人的意見和需求中，拼湊出整份完整的大規格。這是一件大工程，但是現在總算做出一份大家都認為不錯的規格了。融合了各式各樣的用戶功能，造就這套前所未見最先進的手持式閱讀器軟體。功能包括幫出版商擷取行銷統計數據，還有一個完整的方便買書的網路店面，甚至有個創新功能是讓作者在寫書的同時能夠預覽和編輯內容，結合整個出版流程。這的確是一套革命性的軟體。當你坐下來請團隊其他人進行評估之後，得出時程大約 15 個月。似乎有點太長，但是每個人都很興奮，而且你也自信團隊能夠順利交付。

時間快轉一年半。電子書閱讀器團隊努力工作，好幾個晚上和週末都在加班中度過，壓力大到令好幾位成員的婚姻岌岌可危。付出了巨大努力，專案總算完成，你們按照計劃準時交付軟體，幾乎就是時程上訂定的那天。（沒錯，看起來不太可能！不過為了簡化這則例子，請先不要質疑，試著想像這次真的準時交付了。）規格中的每項需求都被實作出來、經過測試，並且完成驗收。團隊感到非常光榮，看過的每位關鍵關係人也都同意這就是他們想要的。

這件產品於是就在市場上……一敗塗地。沒有人想買這套閱讀器，也沒有一位關鍵關係人感到開心。究竟是發生了什麼事呢？

原因在於，最終的軟體已不是一年半前人們需要的那個了。在這個專案剛開始的時候，產業制定了一個新的關於電子書的業界標準格式，然而因為此格式並未列在規格當中，所以並不支援。沒有網路零售業者希望出版非標準化格式的電子書。就算你們團隊建構了很棒的網路店面，也遠不及零售業者目前正在使用的店面，所以不再吸引任何人。你們團隊傾力特地為作者打造的預覽功能也比不上競爭對手能夠直接用電子郵件將 MS Word 文件寄送給讀者然後顯示的支援。

簡直糟糕透了！你們團隊開始時的規格對於所有客戶（無論是公司內或公司外的）而言確實存在許多價值。但是現在你們團隊一年半前同意建構的軟體卻是一文不值。其中有些變化可能可以在專案初期就發現，但是許多變化在初期確實不是那麼明顯。在專案期間，團隊必須在許多關鍵點上快速改變方向，以便將變化融入專案之中。「預先最大需求」瀑布式方法彈性不足，無法讓團隊回應這些變化。

那麼，我們要如何找到一個比較好的專案執行方法，能夠讓關鍵關係人和客戶都滿意，而且依然能夠交付可用的軟體呢？

交付專案

敏捷團隊知道，他們最重要的工作就是將可用的軟體交付給客戶。你已經在第 2 章看到過他們是怎麼辦到的：透過像團隊般一起共事、與客戶合作，以及回應變化。但是這些概念要如何轉化成團隊的日常工作呢？

藉由時常交付價值、將變化視為對專案而言是件好事、時常交付軟體，這樣團隊和客戶便可以一邊調整、一邊合作。軟體可能與團隊原先設計的不同，但是**這反而是件好事**——因為團隊最終打造的才是客戶最需要的軟體。

第 1 條原則：我們最優先的任務，是透過及早並持續地交付有價值的軟體來滿足客戶需求。

這條原則涵蓋了 3 個不同的重要概念：及早釋出軟體、持續交付價值，以及滿足客戶。要真正理解這條原則的核心概念，就必須知道這 3 件事如何同時發生。

專案團隊是在真實世界中運作的，而真實世界中總是世事難料。即便團隊盡善盡美地收集和撰寫了需求清單，也總還是會遺漏些什麼，因為要完整且完美地擷取系統的所有需求，是一件不可能的任務。

但是並不是說團隊不應該嘗試看看，也不是說敏捷方法論是基於一些溝通和撰寫需求的優良實踐。事實上，直到客戶把可用的軟體拿到手上為止，他們很難完全想像心目中的軟體真正的**用途**是什麼。

假使客戶在看過可用的軟體之後，只能夠給予真實的、帶有資訊的回饋，那麼獲得回饋的最好辦法就是透過**及早交付**（**Early Delivery**）：儘快將第 1 個可用版本釋出給客戶。即便交付的東西只有一個功能可用，依然是雙贏。團隊之所以贏，是因為客戶現在可以開始給予帶有資訊的回饋，幫助他們讓專案走在正確的方向。客戶也贏，則是因為他們今天終於可以拿到一套可用的軟體了，在這之前他們只能夠憑空想像。因為軟體是可用的，而且因為客戶可以實際用來做一些符合需求的事情，團隊因而交付了真實的價值。或許只是很小的價值，但總是強過完全沒有交付價值——尤其如果讓用戶等太久、遲遲沒有拿到可用的軟體，那麼收集到的意見只會更加不一致。

及早交付的缺點就是，交付到客戶手上的第 1 個版本距離完整版本還非常遙遠。有些用戶和關鍵關係人會「**真的**」很難適應；有些用戶習慣儘早看到軟體，但是有些用戶則非如此。當拿到手中的軟體並非完美時，大部分的人都會相當困擾。事實上，在許多公司（尤其是已經花了好幾年時間與軟體團隊交手的大型公司）裡面，軟體團隊必須透過書面來與軟體的關鍵關係人協商。一旦軟體團隊與交付對象之間沒有穩固的夥伴關係，如果軟體團隊交付了不完整的軟體，那麼用戶和關鍵關係人將會非常嚴厲地批判這套軟體，甚至因為看不到某項他們想要的功能而大發雷霆。

敏捷的核心價值可以解答這個問題：與客戶合作重於合約協商。因為缺乏彈性的官僚體系阻擋了改變，導致團隊被固定規格綁架，將無法讓軟體隨著時間而演化。這樣的團隊必須導入全新的變更管理流程，需要與客戶有一套全新的合約協商機制。另一方面，落實與客戶合作的團隊則有機會順應所有的變化。這就是所謂的**持續交付**（**Continuous Delivery**）。

這就是為什麼敏捷方法論大多採行迭代的原因。敏捷團隊透過選擇能夠交付最大價值的功能和需求，完成專案的迭代規劃。團隊得知哪些功能最有價值的唯一之道，就是與客戶合作，並且採行其前次迭代的回饋。這麼做可以讓團隊滿足客戶，因為團隊在短期內及早展示了價值，而在長期則交付了價值一如預期的最終產品。

第 2 條原則：竭誠歡迎改變需求，甚至已處開發後期亦然。敏捷流程掌控變更，以維護客戶的競爭優勢。

許多成功的敏捷實踐者在第一次碰到這條原則時都發生一些麻煩。竭誠歡迎改變說起來很容易，但是就專案面而言，當團隊遇到一個必須花費相當大功夫的變更時，往往會引起情緒上的反彈——尤其當開發人員知道無論這項變更需要花費多少時間，他的老闆都不會修訂截止日期時。這有點強人所難，尤其如果公司文化是對延遲非常責備的話，更是難上加難。但其實這麼做是非常值得的，因為竭誠歡迎改變需求是敏捷工具箱中威力最強大的工具之一。

為什麼專案變更會引起情緒上的反彈呢？理解這點，將是這條原則的關鍵。回想你上次專案做到一半卻發現必須改變正在建構的東西時的情景。感覺如何？直到那一刻之前，你都還是覺得專案非常順利。你可能做了許多決策：如何架構工作內容、打算建構什麼東西，以及承諾客戶將會交付什麼軟體。現在有個專案之外的人告訴你說，計劃和成品有些地方弄錯了——而且是「**你**」弄錯了。

要接受別人告訴你自己弄錯了是一件非常困難的事，尤其是你已經做完了那個人交代的事項。大部分的軟體工程師都深以自己的**工藝**而自豪：我們希望交付的是我們敢站在後面背書，而且也滿足用戶需求的產品。然而專案變更卻會危及他們的自信，因為專案變更正在質疑你走過的路以及你做過的假設。

很常發生一種情況，就是某人在明確交代完某件事之後，又告訴你要改變。假使這個人要叫你建構某個功能，然後你就真的去做了，而且正做到一半，那麼這個人後來又回過頭來說：「嗯，其實我是在想，我們可以做個完全不一樣的功能嗎？」將是一件多麼令人沮喪的事。所有的努力全都白費了。現在你必須回過頭去，把已經做好的東西再拿出來修改一下。很難「**不**」提防發生這種事！如果你又剛好必須負責解讀客戶的心思，那麼情況將會更糟，截止日期有可能無止境地延長。

幾乎所有專業的開發人員都遭遇過上述的情況，至少一次。按照這樣的局勢發展，我們又要如何敞開心胸竭誠歡迎改變需求呢？

竭誠歡迎改變需求的第 1 步，就是嘗試從客戶的觀點來看待事情。這通常不容易做到，但是非常有啟發性。你認為將你引導到錯誤方向，難道是客戶故意的嗎？當客戶意識到幾個月前告訴過你的事情是錯誤的，你因而浪費了好幾個月在做白工，這時候你從他的腦袋中看見了什麼？

客戶跑來找你，承認他是錯的（你為這個錯誤付出了好多代價），並且要求變更。這件事並不容易做到。無怪乎客戶經常等了好久一段時間，直到要來找團隊碰面時才提出需求變更！這真是教人為難呀，他們也曉得自己帶來了壞消息。你的截止日期將會往後延宕，客戶自己的也是。如果客戶有需求，而且如果他們公司花了錢要建構出符合那些需求的軟體，那麼要是無法符合客戶的需求，專案就無法交付價值。這全是客戶的錯，因為他沒有在專案剛開始時就把事情講對。

換言之，兩個人被要求完成不可能的任務。你被要求讀懂客戶的心思。客戶被要求預測未來。當你仔細看待這個情況，就會開始變得非常樂意接受竭誠歡迎變更了。另一方面，如果你想要抗拒專案變更，並且繼續跟著原訂計劃走，也很容易——只要確定招募的團隊成員個個心靈相通、洞察未來。

竭誠歡迎改變到底是什麼意思？解釋如下：

- 要做出改變的時候，沒有人會覺得「麻煩」。我們承認（我們的老闆也承認）自己是容易犯錯的平凡人，公司最好允許我們經常犯錯並且改正，而不是期望我們第一次就把事情做到完美。
- 我們都在同一艘船上。團隊裡的每一個人（包括你正在合作的客戶本身）都有各自的需求，以及針對那些需求的變更。假使需求是錯誤的，你和客戶雙方面都有錯，所以不需要針對變更的內容相互指責。
- 我們不會坐視變更被延誤到最後。是的，犯錯是件尷尬的事。但是因為我們都知道只要我們盡力且儘早修正，傷害將可以降到最低。
- 我們不再將改變視為犯了錯。我們總是以當時手邊的資訊盡力做出最大的成效，唯有犯錯才能讓一切清楚，因為我們一路走來所做的決策讓我們可以體認到必須立刻做出改變。
- 我們從變更中學到許多。這是團隊成長以及打造優質軟體的最有效方法。

第 3 條原則：經常交付可用的軟體，頻率可以從數週到數個月，以較短時間間隔為佳。

或許你會覺得竭誠歡迎改變需求的概念很有趣，可能對於專案有所助益。但是你也可能覺得那是個可怕的想法。這是很常有的反應。軟體團隊中的許多成員（尤其是傳統的專案經理）在第一次聽到竭誠歡迎變更的想法時，都覺得很感冒。這些專案經理們每天都在應付變更，然而敏捷對於變更的態度卻迥異於他們過往的認知。敏捷實踐者將對於專案變更的傳統態度稱為**一個口令一個動作**（**Command-and-Control**）。

「一個口令一個動作」是從軍事術語借用而來的。在我們 2010 年出版的《團隊之美》中，我們訪問了 Northrop Grumman 公司的首席工程師 Neil Siegel，他是這樣定義這個字的：

> Andrew 問：我對軍事系統不熟，什麼是一個口令一個動作的系統呢？
>
> Neil 答：這是軍事指揮官的資訊系統。可以讓他們互相溝通，幫助他們掌握狀況：大夥兒正在哪裡，他們的狀況如何。指揮官利用這套系統來了解全局。戰場通常都是小範圍——指揮官可以站在山頭用雙筒望遠鏡觀看整個過程。但是雙筒望遠鏡後來變得越來越大，你無法再像拿破崙般站在山頭俯瞰一切。於是改用其他技術來「觀看」整個戰場。靠的就是「一個口令一個動作」這套系統。

一個口令一個動作的專案管理，就跟軍隊沒有兩樣：

- 「口令」就像專案經理指派任務給團隊成員。團隊可能不是直接向專案經理回報，但是專案經理可以掌控要指派的任務。專案經理拆分工作、規劃時程，然後指派活動和資源給團隊成員。
- 「動作」就像專案經理在做變更管理。每個專案都可能遭遇類似的變化：工作做起來比預期的還久、成員生病或離職了、硬體還沒做出來，以及其他種種無法預期的事情接連發生。專案經理不時監控這些變化，然後在每個變化發生時藉由重新評估、更新時程和文件、重新指派團隊任務以及管理關鍵關係人的期待來掌控專案，所有人都牽涉其中。

傳統的專案經理對於竭誠歡迎改變的第一印象都不好，正是因為他們認為同樣的問題也會發生在敏捷專案身上，團隊必須要能夠回應變化。單純只是接受變化並且竭誠歡迎，顯然只會為專案帶來混亂。假使敏捷團隊不是使用一個口令一個動作的方式，他們要如何在跟上所有變化的同時，又依然能夠應付專案團隊必須面對的日常議題呢？

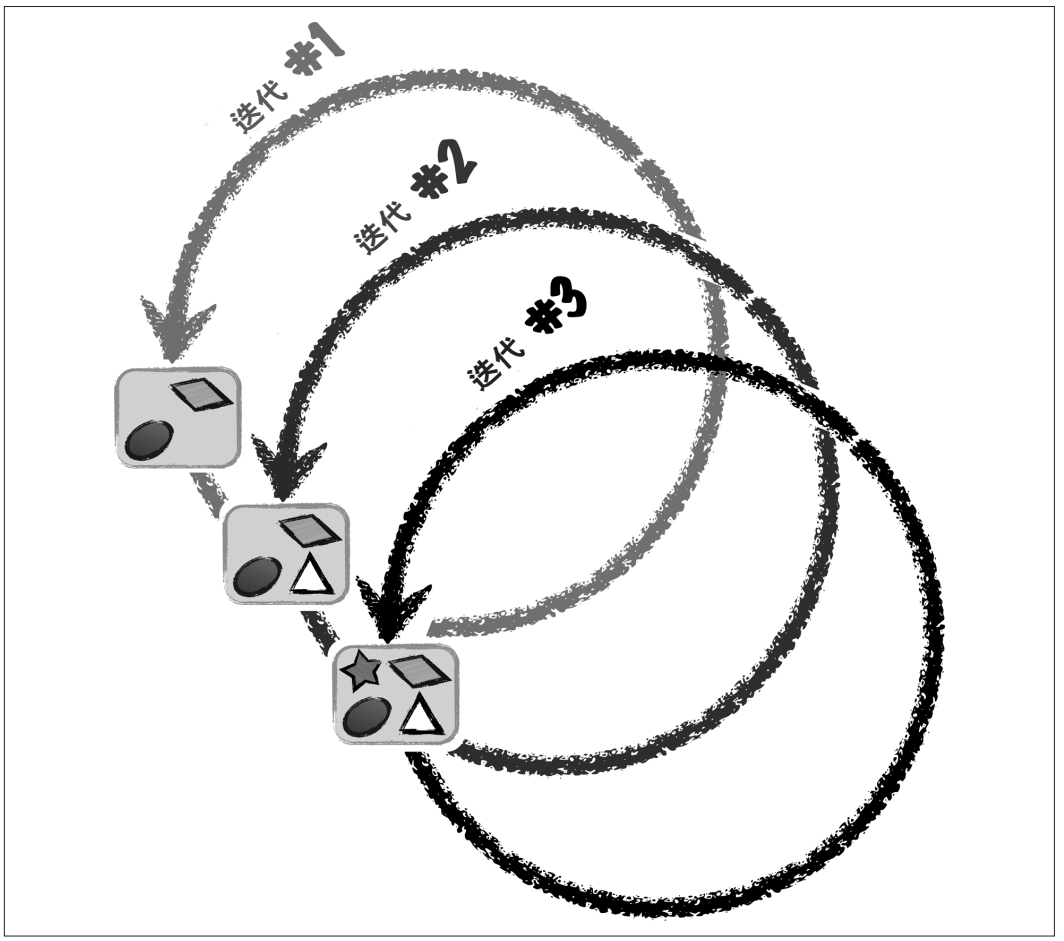

圖 3-1　團隊使用迭代時常交付可用的軟體，每次發佈都會附上新的功能。

竭誠歡迎改變而又不會導致混亂的關鍵，在於時常交付可用的軟體。團隊利用迭代將專案拆分成定期的截止日期。在每個迭代期間，團隊都會交付可用的軟體。而在每個迭代結束時，團隊會向客戶展示他們的建構成果，並回顧吸取到的經驗。接著他們開始規劃新的階段，發想下一次的迭代要建構哪些功能。可預估的時程和固定的查核點幫助團隊及早捕捉變化，讓所有人身處在一個不會互相責備的環境，每個人都可以盡情地討論每個變更，並將發想出來的策略運用在專案上。

從這個地方開始，敏捷對於傳統一個口令一個動作的專案經理而言，變得非常具有吸引力。一個口令一個動作型的專案經理想要掌控截止日期。有時間限制的（Timeboxed）迭代可以給予專案經理那樣的控制權。這同時也解決了專案經理最頭痛的難題之一：對付專案非常後期才湧入的變更。傳統專案經理最艱難的工作之一，就是監控變更。每日審查和迭代回顧讓專案經理可以獲取團隊成員所有的注意力，以便在任何改變成為更嚴重的專案問題之前，能夠更早關注這些變化的發展。

專案經理的角色開始偏離一個口令一個動作，不再只是給予每日作戰計劃然後不斷地調整團隊的進度。相反地，專案經理現在經常與團隊為伍，以確保每位成員都了解全局，並且朝著相同目標前進。當團隊的工作是以交付可用軟體的短期迭代為週期時，比較容易實現這點。每個人都有具體的目標，也比較知道到底要交付什麼——而且所有人都將體認到不只負責建構自己的部分而已，也會知道在迭代結束時整個團隊將會交付什麼東西。

比較好的電子書閱讀器專案的交付方式

上述這幾條原則要如何幫助我們那個漏洞百出的電子書閱讀器專案呢？回想一下專案團隊所面臨到的那些問題：他們的產品之所以失敗，是因為缺少了競爭對手所擁有的某些重要功能（支援電子書的產業標準格式、允許用戶將文件透過電子郵件寄送到裝置上），而有些功能則已經不再需要了（網路商店）。

讓我們再次執行這個專案，但是這一次我們讓專案經理與關鍵關係人和團隊一起合作，設定了幾個長度為 1 個月的 Sprint。這次的專案變得相當不一樣：

- 在第 3 次 Sprint 之後，有位開發人員回報即將通過一份關於電子書的產業標準新格式。團隊決定實作一個程式庫，以便在第 4 次 Sprint 時支援這種格式，同時也會在第 5 次迭代時在閱讀器的使用者介面中完成這項支援。
- 經歷 10 個月的開發，他們建構了可以載入到原型機上的可用軟體，並將此早期測試版提供給用戶。專案經理與這些用戶討論，發現他們真的很希望有個方式能夠將 Microsoft Word 文件和報紙文章放在閱讀器上。團隊在下次 Sprint 期間花了一部分的時間來完成電子郵件功能的整合，於是用戶們便可以透過電子郵件將文章寄送到他們的裝置上了。
- 在專案執行了 1 年之後，關鍵關係人讓團隊知道他們不再需要網路商店，畢竟現在的零售商全都使用標準化的電子書格式了。幸運的是，這項功能一直都是放在待處理項目比較後面的位置，而且其他的 Sprint 都用來處理更重要的功能了，所以在網路商店這部分尚未動工太多。

由於團隊都會在每次 Sprint 之後交付可用的軟體，因此待處理項目中那些他們能夠及早交付的功能都已被完成！出版社夥伴已經把書準備好了，因為他們的資深經理都有拿到早期的軟體版本，也都使用過原型機了。這使得他們全程參與，讓他們有機會確認他們能夠儘快在第一版產品就緒時也把書準備好。

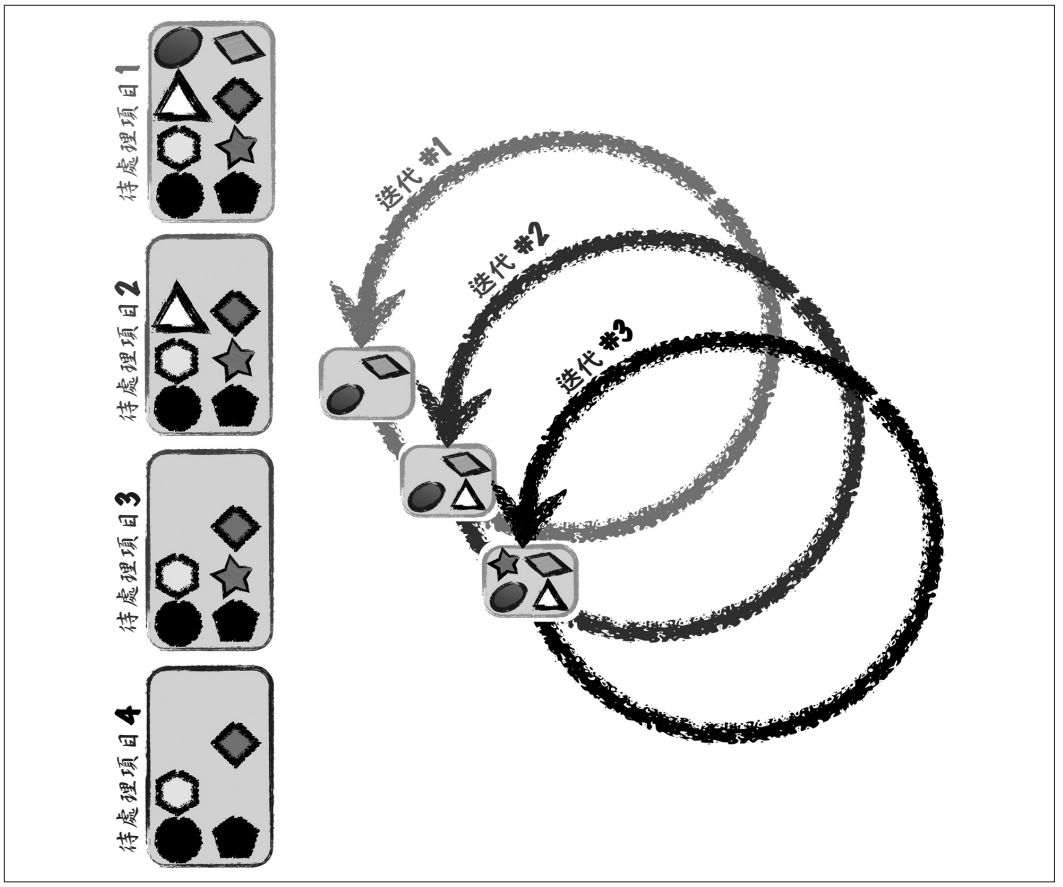

圖 3-2　在每次迭代開始時，團隊都會從待處理項目中挑選要建構的功能。

透過持續發佈、竭誠歡迎改變，以及在每次迭代之後交付可用的軟體，電子書閱讀器專案因而更能夠交付成功的產品。相較之下，在沒有效率的瀑布式流程中，團隊只有在需求確認之後的簽約儀式上與客戶碰過一次面，而敏捷團隊則是持續與客戶保持密切聯繫。後者的方式讓團隊能夠回應變化與建構較好的產品。

但是對於我們的電子書閱讀器團隊而言，事情並沒有那麼美好——甚至還差得遠了。他們使用迭代來交付可用的軟體，但是他們又妥協於文件的規範。每個人都很開心不必再打造那些根本不會拿來販賣的軟體了。但是每次當團隊發現一個需要對專案進行的良性變更時，一半的團隊成員都必須回過頭去更新規格，以確保計劃在最新狀態、成員也都在正軌上。他們花費在更新文件的心力，似乎和撰寫程式碼的一樣多。

團隊成員討論多時，想辦法要減少這些文件的維護工作。他們討論文件的「正確的詳細程度」（Right Level of Detail）。但是每次當他們試著拿掉某個東西的時候，總會有人鏗鏘有力地指出要是沒有寫下那個特殊功能、需求、設定或測試案例的話，其他人可能會無法理解。假設那個功能沒有被正確實作，他們就會指責這點。所以看起來似乎文件的每個片段都是不可或缺的，因為如果不這麼做的話，團隊就會陷入建構錯誤軟體的兩難。

有辦法降低團隊的負擔，而又不會傷害專案嗎？專案的文件真的有「正確」程度這回事嗎？

重點提要

- 敏捷宣言中的**敏捷開發的 *12* 條原則**提供敏捷實踐者一些採取實踐和方法論時的方向和指引。
- 敏捷團隊滿足客戶的方式是**及早在專案期間獲得回饋**，並且**持續交付軟體**以便持續獲得最新的回饋。（第 1 條原則）
- 敏捷團隊將專案變更視為對專案的正面及健康發展，所以他們**擁抱變化**。（第 2 條原則）
- 敏捷團隊透過使用有時間限制的迭代，因此能夠**時常交付可用的軟體**，並且不斷調整專案，進而為客戶交付最大價值。（第 3 條原則）