“身處優秀的 Agile 團隊能感受一種魔法。問題、人際衝突、與政治都在強調與專注於產出、學習、再產出的過程中消失。但要建構這樣的團隊並不容易。現在簡單多了，深入淺出 Agile 開發一書是我們很多人等待的巨著。這本書的 Scrum、Extreme Programming、Kanban 等實務建議可以幫助所有人建構非常棒的 Agile 團隊”。

— Mike Cohn，《Succeeding with Agile》、《Agile Estimating and Planning》、《User Stories Applied》等書的作者

“如果曾經在軟體團隊工作過，你會覺得《**深入淺出 Agile**》一書的案例研究精準且具洞察性。我真希望以前就有人告訴我他的建議，但無論在軟體產業中待過多久，你還是會發現與學習到檢視舊問題的新方式。我很驚訝的發現除了 Pair Programming 之外 XP 還有很多可以學習的地方 —— 我一定會讓我的團隊從這些實務建議中獲益”。

—Adam Reeve，RedOwl Analytics 的架構設計長

“《**深入淺出 Agile**》的寫作方式很容易消化吸收，一讀就停不下來。由於寫作方式讓人互動而非只是提供事實，我感覺到對 Agile 的原則與實踐更好的理解” 。

— Patrick Cannon，Dell 的資深程式經理

“《**深入淺出 Agile**》完整的解釋非常具挑戰性的 Agile 開發概念，能夠讓你通過 PMI-ACP 考試並以這些概念與練習、真實案例加強學習。這本書獨特的風格使得初學者與專家都有收穫。感謝 Andrew 與 Jennifer 寫出另一本傑出的深入淺出系列好書！”。

— John Steenis， PMP、CSM、CSPO

“《**深入淺出 Agile**》非常棒！我喜歡作者以有趣、容易閱讀與理解的風格說明重要主題。為 Andrew 與 Jennifer 的成就致敬！”。

— Mark Andrew Bond，某高等教育機構的網管專案經理

“任何規模的軟體開發團隊的成員，無論是否採用 Agile，都應該讀這一本書。完全或部分採用 Agile 的團隊能更好的認識要如何改善流程。沒有採用 Agile 的團隊可以學到如何開始進行 Agile。對我來說，這本書很好的反映出我過去 20 年的成功與失敗專案”。

— Dan Faltyn，BlueMatrix 的安全主管

“Agile 在過去是業界中的熱門字，許多人將它掛在嘴上而並未真正的認識這種實踐的原則與價值。《**深入淺出 Agile**》一書闡述此一主題並提供非常好的 Agile 指引”。

— Philip Cheung，軟體開發者

對本書的讚譽

“我是個專案經理，去年以《**Head First PMP**》取得 PMP。《**深入淺出 Agile**》有相同的概念…視覺化思考與學習認識這些概念。市面上高品質的書很稀少，這本書更是珍貴且能用於準備 PMI-ACP 考試。模擬考題與第 7 章（包含領域審核）非常適合考試準備。內容很棒，我第一次就過了！”。

— Kelly D. Marce，某金融機構的專案經理

“《**深入淺出 Agile**》一書顯示 Agile 不只是單一的方法論，還是多種思考開發生命週期的路徑與方式。這本書能指引你找出最適合你的團隊的方法並認識如何持續的進行改善。像 Kanban 這樣的方法，透過視覺化工作流程和限制進行中的工作來建立牽引式系統，可以特別的強化團隊”。

— Nick Lai，Uber Technologies 的資深工程經理

“還有什麼比 Agile 更好的方式來處理工作。丟掉過去無聊的教科書並拾起《**深入淺出 Agile**》開發來獲取更好的學習體驗”。

— Tess Thompson，MS、PMP、PMI-ACP、CSP、Saint Mary’s University 的教授

“Agile 對學習傳統工作方式的 PM 是個挑戰。這本書的作者特別注意容易閱讀並直接展示出通過 PMI-ACP 考試的資訊以及採用 Agile 的實務上更為重要的資訊”。

— Dave Prior，Scrum 認證講師、PMP、PMI-ACP

“《**深入淺出 Agile**》完整的討論 Agile 的實務問題與解決方案。作者解釋了不同類型的 Agile，提供大量 Agile 資源給讀者”。

— Keith Conant，某支付公司資深軟體工程師

“《**深入淺出 Agile**》一書是非常好的資源，它不只教授 Agile 的實務與方法，更重要的是思考方式的轉變。這本書的內容基於 Agile 宣言所述的價值與原則。強烈建議這本書給想要學習 Agile 的人”。

— Mike MacIsaac，Scrum Master、MBA、PMP、CSM

“我是 IBM 的 Agile 認證講師，我幫助組織中的人成為 Agile 開發者。我發現《**深入淺出 Agile**》一書是非常好的書，為準備 PMI-ACP 考試的人提供很豐富的學習資源”。

— Renato Barbieri，PMP、PMI-ACP、經理、講師、IBM 的 Kepner-Tregoe 計劃的領導人

“Andrew 與 Jenny 在《**Head First C#**》一書中展示出非常棒的 C# 教學。它以非常容易閱讀的獨特方式討論大量的細節。如果你受不了傳統的 C# 書本，你一定會喜歡這一本”。

— Jay Hilyard，軟體開發者、《C# 3.0 Cookbook》的作者之一

“閱讀《**Head First C#**》是個很棒的體驗。我還沒看過其他教的這麼好的書…絕對會向想要學習 C# 的人推薦這本書”。

— Krishna Pala，MCP

“《**Head First Web Design**》真正解開網頁設計流程的迷霧，任何網頁程式設計師都應該讀讀看。對於沒有上過網頁設計課程的網頁開發者，《**Head First Web Design**》闡述了許多業界的理論與最佳實踐”。

— Ashley Doughty，資深網頁開發者

“建構網站絕對不只是寫程式而已。《**Head First Web Design**》展示製作良好體驗給使用者的必要知識。又是一本深入淺出的好書！”。

— Sarah Collings，使用者體驗軟體工程師

“《**Head First Networking**》解釋許多甚至是資深專家也難以理解的網路概念使人們毫無困難的吸收。幹得好”。

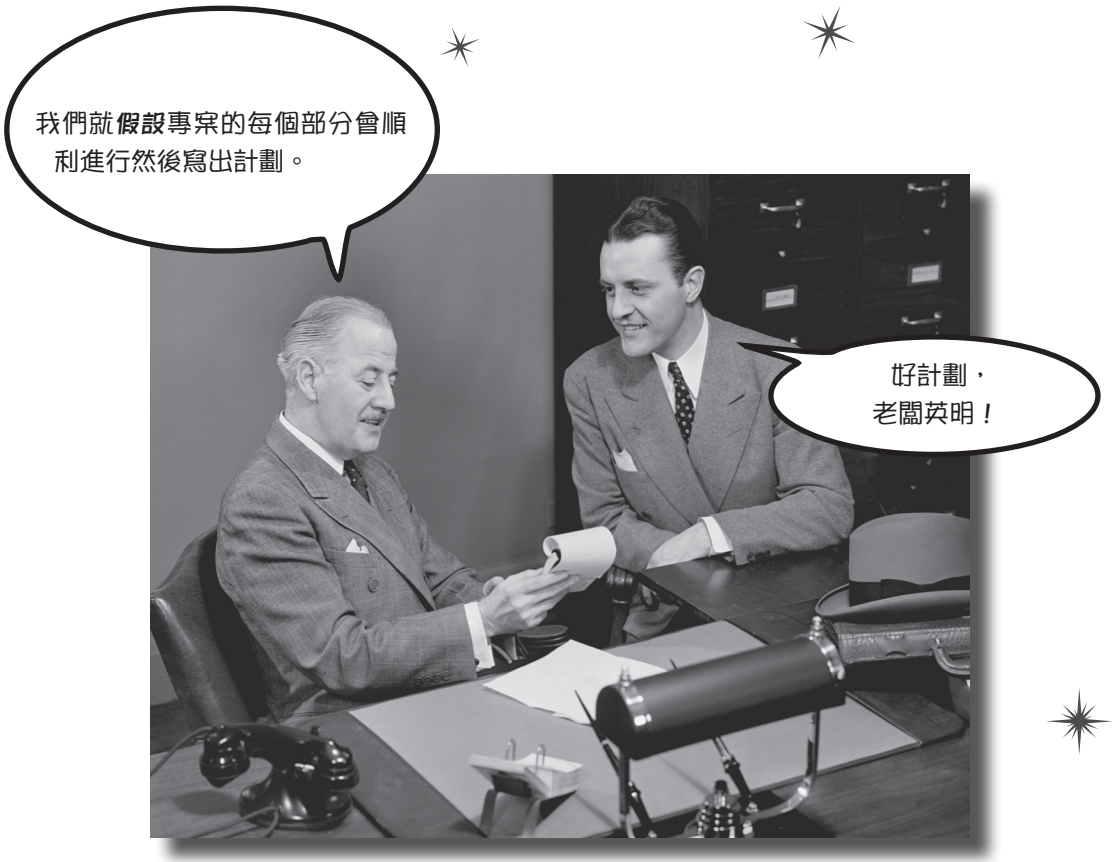
— Jonathan Moore，Forerunner Design 的老闆

“許多資訊技術書籍經常漏失大觀。《**Head First Networking**》專注於真實世界，以精細的內容提供經驗知識給 IT 新人。結合實務問題與說明使這本書成為非常好的學習工具”。

— Rohn Wood，University of Montana 的資深分析師

1 Agile 是什麼？

原則與實踐



現在是敏捷的大好時機！業界終於發現解決困擾著數代軟體開發者問題的真正可行方法。敏捷開發不僅產生很好的**結果**，也讓開發團隊工作進行的更順利。但如果敏捷方法這麼好，為什麼不是每個人都採用它？答案是對一個團隊有用的敏捷方法可能導致另一個團隊的嚴重問題，因為不同之處在於團隊的**心態**。準備好改變你看待專案的方式！

新功能聽起來不錯…

這是 Kate，她是矽谷一家新創公司的專案經理。她的公司設計影音串流服務與網路廣播使用的軟體，此軟體即時分析受眾並據此產生推薦節目給使用者。現在 Kate 的團隊有機會產出真正對公司有幫助的東西。

…但事情不一定如預期一樣

Kate 與團隊的溝通沒有達到她的理想。她要怎麼告訴 Ben ？

Kate：如果新功能可以放在下一版，我們都會領到很多獎金。

Mike：嗯，聽起來不錯。

Kate：太好了！所以我們說定了？

Mike：等一下！我說**聽起來**不錯，但是辦不到。

Kate：蛤？ Mike 你別耍我。

Mike：聽著，如果四個月前開始設計資料分析服務的時候就告訴我們要這些功能，事情會很簡單。現在我們得修改很多程式…我不想說太多細節。

Kate：好，我也不想聽細節。

Mike：所以…我們說完了？我的團隊還有很多工作。

Agile 是救星！

Kate 正在研究 Agile，她覺得這有可能幫助她讓這些功能加入下一版。Agile 在軟體團隊中越來越受歡迎，因為常聽到“採用了 Agile”的人表示效果卓著。他們的軟體更好了，給他們與他們的使用者產生非常大的改變。不僅如此，除了 Agile 團隊很有效率外，工作起來也更愉快！壓力更小，工作環境也更舒適。

為什麼 Agile 變得如此受歡迎？有很多原因：

- ★ 團隊採用Agile後，他們發現比較容易趕上進度。
- ★ 他們還發現軟體的錯誤變少了。
- ★ 程式碼更容易維護 ——修改、擴充、變更程式不再頭痛。
- ★ 使用者更高興，所以大家的日子更輕鬆。
- ★ 最棒的是，Agile團隊有效率，團隊成員的生活變得更好，因為他們可以早點下班並很少加班（這對很多開發者來說是最重要的！）。

daily standup 是個好的開始

最常見的 Agile 實踐之一是稱為 **daily standup** 的每日例行會議，團隊成員報告目前工作與遭遇到的挑戰。此會議讓每個人都站著以縮短時間。很多團隊因為採用 daily standup 而讓專案獲得成功，這通常是 Agile 的第一個步驟。

在 daily standup 會議中每個人都站著，如此能保持會議簡短、輕鬆、且有重點。

但這傢伙有在注意聽別人說話嗎？

Kate 嘗試進行 daily standup

Kate 萬萬沒想到 Mike 的團隊並非每個人都對這種做法感興趣。事實上，其中一個開發者對增加的會議很不開心，覺得每天要報告進度是一種侮辱。

Kate 覺得 Mike 的團隊不講理，但或許他們有道理。你怎麼看？

動動腦

發生了什麼事？是 Mike 不講理嗎？是 Kate 要求太多嗎？為什麼這個簡單的做法會引發衝突？

不同團隊成員有不同的態度

Kate 從一開始採用 Agile 就遇到問題 ——這不是個案。

事實上很多團隊的 Agile 沒有達到預期中的效果。你知道有一半以上的軟體公司嘗試過 Agile 嗎？除了成功案例 —— 很多！ —— 也有很多團隊嘗試 Agile 但結果並非特別好。事實上，他們感覺到上當了！ Agile 掛保證會成功，但嘗試採用 Agile 的專案似乎都動不起來。

這就是發生在 Kate 身上的事情。她獨自規劃，接著想要團隊報告進度，所以開始把團隊拉進會議室。這麼做真的會有效果嗎？擔心人們不會跟著她的計劃走，所以她專注於聽取每個人的進度報告。另一方面，Mike 與他的團隊希望會議盡可能快一點結束，以便回到“真正”的工作上。

在 Kate 的低效率會議上，其他人都在等著輪流報告，輪到時就短短的說一下。她還是得到了一些有用的資訊，但代價是衝突與無聊 ——且沒有人是合作的。

兩個人對相同做法的看法非常不同。如果他們都不覺得有什麼意義，則這種做法非常沒效率。

軟體專案就是這個樣子，是吧？
教科書案例在真實世界中並不可行。
你一點辦法都沒有，是吧？

不！正確的心態能讓方法更有效。

讓我們認清一件事：Kate 舉行會議的方式跟其他 daily standup 一樣。雖然沒效率，但這種會議方式**還是有成果**。Kate 會發現計劃的問題，而 Mike 的團隊也會有好處，因為影響到他們的問題會提早發現。整個會議並沒有佔用每天太多時間，所以還是值得繼續。

但成功的 Agile 團隊與行禮如儀的團隊間有個重大差別。差別的關鍵在於團隊在專案中的**心態**。信不信由你，每個人的參加態度會讓它更有效率！

每個團隊成員參加 daily standup 等活動的態度會對其效率有重大的影響。就算大家都不認同，就算很無聊，但因為有足夠的效果，會議還是值得進行。

好的心態讓此實踐運作的更好

如果 Kate 與 Mike 有不同的心態呢？如果每個團隊成員以完全不同的態度參加 daily standup 呢？

舉例來說，如果 Kate 讓團隊一**起參與**專案規劃呢？如此會讓她真正的傾聽每個開發者的聲音。若 Kate 改變對會議的想法，她會停止找出他們如何偏離她的計劃以便改正。她的會議目的變成：認識團隊成員一起制定的計劃，而她的任務是幫助團隊有效率的進行工作。

這是非常不同的規劃方式，從未出現在 Kate 所受的專案管理訓練課程中。她一直被教育她的工作就是拿出專案計劃與獨裁團隊。她有評量團隊依循計劃的工具，以及強制施行改變的嚴格程序。

沒有事情對她是完全不同的。她發現讓 daily standup 可行的唯一方式是**努力與團隊合作**，以讓團隊一起找出進行專案的最佳方式。此時 daily standup 變成整個團隊合作確保每個人做出可靠的決策的方式，而專案也會上軌道。

以前 Kate 在太晚發現計劃改變而使團隊無法有效率的應變時會很沮喪。

舉行 daily standup 後，團隊與她每日合作找出改變，因此可以提前應對。這樣做有效率的多！

所以 daily standup 表示你會**傾聽我跟團隊**的意見並真正改變專案進行的方式？

若 Mike 覺得此會議不只是進度報告，還有**認識專案的狀況**，並合作找出讓每個人工作的更順暢的方式呢？如此會讓他覺得 daily standup 很重要。

好的開發者不只對他自己的程式，還對整個專案的方向有想法。daily standup 變成他確保專案可行有效的方式 —— 而 Mike 知道長期下來會對團隊更有幫助，因為程式設計之外的事情也會順利的進行。他知道在會議中把問題攤開時，***每個人都會傾聽***，而專案會因此運作的更好。

事情在 Mike（與其餘團隊成員）發現 daily standup 可幫助他們規劃次日工作時會更好 ——團隊中的每個人都參與了規劃過程。

這很合理！專案規劃在每個人都參與時做得更好。但我認為只有會議中的每個人都投入時才會如此。

動動腦

要如何改變團隊或個人的心態？你能從你的專案中舉出某人 —— 或許是你自己改變心態的例子嗎？

所以 Agile 是什麼？

Agile 是一組**方法**（**method**）與**方法論**（**methodology**），幫助軟體團隊更好的處理與簡化所遇到的特定問題，使它們能夠更直接的解決。

這些方法與方法論處理包括專案管理、軟體與架構設計、流程等所有傳統軟體工程的問題。每個方法與方法論由**實踐**（**practice**）組成，它是最佳化的流程以便更容易採行。

我花了這麼多時間做出計劃，但團隊似乎根本不鳥它。我可以靠 daily standup 確保他們**完全**聽話。

心態與方法論

Agile 也是**心態**，這對沒有接觸過 Agile 的人來說是個新鮮事。每個團隊成員對這些實踐的態度大幅影響這些實踐的效果。Agile 心態專注於幫助人們分享資訊，如此能讓他們更容易做出重要的決定（而不是只聽老闆或專案經理的決定）。重點是開放讓整個團隊參與規劃、設計、與流程改善。為幫助每個人進入有效率的心態，每個 Agile 方法論各有一組讓團隊成員作為指引的**價值**（**value**）。

如果我們*一起合作*規劃專案，則我們可以靠 daily standup 在過程中確保方向正確。

動動腦

如果有個團隊成員在 daily standup 中離席且並沒有傾聽他人的意見要怎麼辦？

削尖你的鉛筆

Kate、Ben、與 Mike 在 daily standup 提出了幾個問題。我們相對列出了幾個你經常會看到 Agile 團隊使用的不同實踐的名稱。不用擔心你還沒有見過其中幾項 —— 接下來你會深入學習。我們對每個實踐加上簡短的說明以幫助你理解。試試看你能否將每個問題與可能有幫助的實踐對上。

"我們在糾纏不清的程式碼中浪費很多時間找 *bug*！"

"OK，我們已經討論完使用者故事，接下來讓我們研究一下以計劃轉這幾週的工作。"

"每個釋出版本總是反覆遇到相同類型的問題。"

"影音串流使用者提出幾個功能需求。客户說新功能沒有真正的解決他的問題。"

"我以為我們已經說好在禮拜五更新音樂資料庫程式碼。你們現在才告訴我還要三個禮拜？"

回顧（retrospective）是一種會議，每個人在會議中討論專案最近一部分的狀況並討論從中學到的教訓。

使用者故事（user story）是一種表達使用者特定需求的方式，通常是寫在便條紙或索引卡上的幾個句子。

任務版（task board）是一種 Agile 計劃工具，將使用者故事貼在版上並根據其狀態以欄分類。

燃盡圖（burndown chart）是個每日更新的線圖，記錄專案所剩工作，在工作完成時"燃盡"。

開發者持續的**重構**（refactoring）程式碼或改善程式碼結構而不改變其行為來改正程式問題。

如果你還不知道這些實踐也沒關係。接下來幾章會深入討論。

Scrum 是 Agile 最常見的方式

團隊有很多種進行 Agile 的方式，Agile 團隊使用很多種方法與方法論。但多年來的調查研究發現最常見的 Agile 方式是 **Scrum**，它是一種專注於專案管理與產品開發的軟體開發架構。

團隊採用 Scrum 時，每個專案都依循相同的基本模式。Scrum 專案有三種主要角色：**產品負責人**（**Product Owner**）（例如 Ben）一起維護**產品待辦項目**（**Product Backlog**）；**Scrum 大師**（**Scrum Master**）指導團隊克服障礙；以及**開發團隊成員**（**Development Team member**）（團隊中的其他人）。專案根據 Scrum 模式分成**衝刺段**（**sprint**），或等長的週期（通常是兩週或 30 天）。在一個衝刺段的開始，團隊進行**衝刺段規劃**（**sprint planning**）來決定衝刺過程中要完成哪些產品待辦項目列出的功能。這稱為**衝刺待辦項目**（**Sprint Backlog**），而團隊根據項目建構所有功能。團隊每天召開稱為 **Daily Scrum** 的會議。在衝刺段的最後於**衝刺段審核**（**sprint review**）中展示可用軟體給產品負責人與主管，而團隊召開**回顧會議**以找出所學到的教訓。

我們會在第 3 與第 4 章深入討論 Scrum，它們不只會教你如何幫助團隊建構更好的軟體與執行更成功的專案，還會探索所有 Agile 團隊共同的重要概念與想法。

XP 與 Lean/Kanban

雖然 Scrum 是最常見的 Agile 方法論，但許多團隊採用其他方式。另一個常見的方法論是 **XP**，此常與 Scrum 結合的方法論專注於軟體開發與程式設計。有些 Agile 團隊採用 **Lean** 與 **Kanban**，這種心態提供工具給你以認識你現在建構軟體的方式以及幫助你日後進入更好的狀態的方法。你會在第 5 與第 6 章學到 XP 與 Lean/Kanban。

一下子冒出太多新名詞嗎？

我們在新名詞首次出現時以**粗體字**突顯。這一頁出現很多新名詞 —— 如果有幾個沒見過也沒關係！在相關背景中見到新東西之後再深入學習，能幫助你的大腦記住。這是深入淺出系列以對大腦友善的神經科學，讓你輕鬆記憶和學習！

削尖你的鉛筆 解答

團隊可以透過回顧專案並討論什麼是對的與什麼可以改善來避免犯重複的錯誤。

“我們在糾纏不清的程式碼中浪費很多時間找 *bug*！”

任務版是讓所有團隊成員看到相同專案大局的好方法。

“*OK*，我們已經討論完使用者故事，接下來讓我們研究一下以計劃轉這幾週的工作。”

“每個釋出版本總是反覆遇到相同類型的問題。”

“影音串流使用者提出幾個功能需求。客户說新功能沒有真正的解決他的問題。”

團隊中每個人都知道使用者要什麼的時候會更好的建構出使用者需要的軟體。

“我以為我們已經說好在禮拜五更新音樂資料庫程式碼。你們現在才告訴我還要三個禮拜？”

回顧（retrospective）是一種會議，每個人在會議中討論專案最近一部分的狀況並討論從中學到的教訓。

使用者故事（user story）是一種表達使用者特定需求的方式，通常是寫在便條紙或索引卡上的幾個句子。

任務版（task board）是一種 Agile 計劃工具，將使用者故事貼在版上並根據其狀態以欄分類。

燃盡圖（burndown chart）是個每日更新的線圖，記錄專案所剩工作，在工作完成時“燃盡”。

這是個 *XP* 實踐。有些專案經理在第一次發現 *Agile* 實踐不只是規劃與執行專案還專注於程式碼時感到驚奇。

開發者持續的**重構**（refactoring）程式碼或改善程式碼結構而不改變其行為來改正程式問題。

問：聽起來 Scrum、XP、Lean/Kanban 是非常不同的東西，怎麼可能都是 Agile ？

答：Scrum、XP 與 Lean/Kanban 專注於不同領域。Scrum 主要專注於專案管理：要完成什麼工作，並確保與使用者和主管的要求一致。XP 專注於軟體開發：建構設計良好且容易維護的高品質程式碼。Lean/Kanban 結合 Lean 心態與 Kanban 方法，團隊以它專注於持續改善建構軟體的方式。

換句話說，Scrum、XP、Lean/Kanban 專注於軟體工程的三種不同領域：專案管理、設計與架構、以及程序改善。因此它們有不同的做法是合理的 ——這就是不同之處。

下一章會討論它們的共同點：能幫助團隊採用 Agile 心態的共同價值與原則。

問：這不是新瓶裝舊酒嗎？例如 Scrum 的衝刺段只是里程碑與專案階段，不是嗎？

答：第一次接觸到 Scrum 等方法論時，很容易會覺得內容與你已經知道的東西類似 —— 這是件好事！如果你曾經參加過團隊，應該會覺得很多 Agile 實踐很熟悉。你的團隊打造某些東西，而你與團隊成員應該會好好的做出你（目前）不想再修改的東西。

但人們很容易掉入 Agile 看似熟悉的部分與之前就知道的東西相同的陷阱中。舉例來說，Scrum 衝刺段與**專案階段並不同**。傳統專案管理中的階段或里程碑與 Scrum 的衝刺段有很多不同處。

舉例來說，在傳統的專案計劃中，所有階段都在專案開始時規劃好；在 Scrum 中，只有下一個衝刺段會做細節規劃。這種差別對習慣傳統專案管理的人來說很奇怪。

你接下來會學到 Scrum 的規劃如何進行以及它跟你已經習慣的方式有何不同。此時要保持開放的態度 ——注意到你覺得“這與我已經知道的一樣”的時刻。

重點提示

- 許多想要採用 Agile 的團隊從 **daily standup** 開始做起，大家站著開會以保持簡短。
- Agile 是一組**方法與方法論**，但也是**心態**，或團隊成員共同的態度。
- daily standup 在團隊成員具有正確的**心態**時更有效率 —— 每個人傾聽他人並合作確保專案在正軌上。
- 每個 Agile 方法論有一組**價值**來幫助團隊進入最有效率的心態。
- 團隊成員依循共同原則與共同的價值時會讓所採取的方式**更有效率**。
- **Scrum** 是專注於專案管理與產品開發的一種架構，是最常見的 Agile 方式。
- 在 Scrum 專案中，專案根據 Scrum 模式分成**衝刺段**（**sprint**），或等長的週期（通常是 30 天）。
- 每個衝刺段從決定要進行什麼工作的**衝刺規劃**開始。
- 在衝刺段期間團隊每天召開稱為 **daily scrum** 的短會。
- 衝刺段結束時，團隊與主管召開**衝刺段審核**並展示軟體。
- 團隊召開**回顧會議**以檢視衝刺段的進行並一起討論如何改進來完成衝刺段。

照過來！

不要立即排斥！

許多人 ——特別是硬派開發者 ——一聽到心態、價值、原則等詞彙就嗤之以鼻。對習慣關起門來不與人交談的程式設計師更是如此。如果你開始這麼想，試著給這些想法一個考慮的機會。畢竟有很多好軟體是這麼開發的，因此它一定有些優點…不是嗎？

削尖你的鉛筆

下列哪些情境應用了實踐，哪些情境應用了原則？沒有看過這些實踐也沒關係，試著從題目中找出正確的答案（這是應考時的技巧！）。

1. Kate 知道與團隊溝通專案重要資訊最有效率的方式是**面對面的交談**。

☐ 原則　　☐ 實踐

2. Mike 與他的團隊知道使用者可能會改變主意，而這些改變會損害程式，因此他們使用**漸進設計**來確保程式碼在之後容易修改。

☐ 原則　　☐ 實踐

3. Ben 使用**人物**建構典型使用者的模型，因為他知道團隊越能認識使用者則越能建構良好的軟體。

☐ 原則　　☐ 實踐

4. Mike 總是確保他的團隊做出可以展示給 Kate 與 Ben 看的東西，因為他知道**開發中的軟體**是展示團隊進度最好的方式。

☐ 原則　　☐ 實踐

5. Kate 想要改善團隊建構軟體的方式，所以她召集所有人一起對流程做出**合作改進與進化實驗**並使用數據證明這些改變讓事情變得更好。

☐ 原則　　☐ 實踐

6. Mike 與他的團隊以建構在過程中容易修改的程式來**擁抱改變**。

☐ 原則　　☐ 實踐

答案見第 20 頁

Kate：這個專案比以前進行的更好。這都是因為每天的一場小會議！

Mike：呃，我不這麼認為。

Kate：哎呦，Mike ！不要這麼悲觀。

Mike：真的，不開玩笑。你不會真的以為你是第一個嘗試以增加開會來解決問題的人吧？

Kate：這個…嗯…

Mike：結果還不錯，我就老實跟你說，你開始 daily standup 時，團隊中幾乎每個人都不高興。

Kate：真的？

Mike：是的。你不記得前一兩週我們都在低頭看手機嗎？

Kate：沒錯。我猜是沒什麼用。其實我真的有考慮取消。

Mike：然後有個程式設計師提出了一個嚴重的架構問題。大家因為他很強而傾聽他的意見，而大家也很尊重他的意見。

Kate：對，當時我們必須做出重大改變，而我決定砍掉兩個功能。

Mike：就是！那非常重要。我們遇到這種問題時通常必須工作到很晚。例如我們發現分析演算法有嚴重的瑕疵時。

Kate：對啊，那很糟糕。我經常發現我們承諾了辦不到的事情。這次我們提前發現問題，我和 Ben 合作管理使用者的期待，並讓你們有時間做出新方案。

Mike：我們絕對會在出現這種問題時提出來討論。

Kate：等一下 —— 蝦密？這種問題很常發生？

Mike：拜託，我就沒有遇過開始進行程式設計後從不出包的專案。這就是真正的軟體專案，年輕人。

看起來 Kate 領悟到實際上的軟體專案比紙上計劃更混亂與複雜。以前她只需要寫好計劃然後逼團隊跟上…有問題都是他們的錯。

另一方面，此專案進行的比以前都好。她必須更努力的處理問題，但結果更好！