推薦序

所謂的*左移*意思是把事務往時間軸的左邊移動，近年來這種超前部署的概念頻繁出現，我們也聽聞左移的重要性，以及它在軟體設計、安全性與測試等面向上的影響，測試越早進行，就越有可能在臭蟲發生的當下就立刻解決它，因此降低了未來需要大幅修正或重構的可能性，並意味著成本與複雜度的降低，而如果是性能測試，我們也可以從性能走勢圖中看出性能瓶頸並加以改善，而無須擔心真實上線後的表現，藉由測試，我們可以從中發掘出架構中的低性能因子，或者是某些造成性能損失的人為失誤。

然而測試左移也就意味著測試當下的軟體還是未完成、施工中的，因此我們需要更強的問題解決能力，為此付出的精力有可能高於於持續測試（continuous testing）所要付出的成本，特別是當自動化測試大量部署之後，一般來說，除了探索測試會以人工進行，大部分的測試最好都盡可能自動化。

實務上要完善一套系統的確有太多的測試必須進行，就如同此書的標題，全棧測試帶給我們的是覆蓋整個系統的測試議題總覽，我們關注系統的性能、UI、API 規約（contract）、E2E（端對端）功能、單元測試、可用性測試等各個面向，但問題是許多人不知道該如何著手進行如此大規模的測試，這也是本書的價值所在，市面上也有許多以測試或敏捷測試為主題的書籍，它們也同樣倡導測試左移，而 Gayathri 的書則更深入探討現代化應用框架下的測試觀點，本書描述了各種層面的測試議題，以及各自適用的測試原則與策略。

本書包含了一系列的動手練習，從中具體展示測試的實際施作方式，儘管練習中的工具可能會隨著時代而淘汰，但這樣的練習依然是相當有價值的，它讓我們知道該如何用適當的工具建構正確的測試類型，讓讀者感受測試具體運行的方式，並且工具提供了對測試進行試驗的可能性，因此儘管工具會迭代更換，但從中學習到的測試策略實踐經驗，才是真正的價值所在。

本書所談及的測試議題相當廣泛，包括靜態測試、資料測試策略、探索測試等等，特別是當代軟體的複雜度越來越高，探索性測試就更顯重要，此外還有一章專門探討安全測試，讓我們知道系統上有哪些可能存在的弱點，可用性測試也是獨立的一個章節，探討的是如何讓系統更簡單易用的方法，特別是針對那些身心障礙人士而言。

對於各種層面的測試，我們都尋求找出其中可能會失誤之處，並且藉由強化測試策略來完善那些失誤，完整的測試策略有助於我們建構出更強健的資訊系統，在這本書中，Gayathri 憑借她豐富的個人經驗，告訴我們該如何將測試落實在各種不同的系統之中，讓軟體從業人員能藉此發展出專業又適切的測試策略，來完善軟體的品質。

—*Rebecca Parsons* 博士
Thoughtworks CTO、《建立演進式系統架構》共同作者

前言

如果您身處軟體產業，不論角色為何，不太可能完全沒碰到一點測試，因為測試是成就一套軟體的必經之路，它融入軟體開發週期的每個階段，隨著數位時代的降臨，人們的生活中充斥大量的應用，或許是網頁或許是手機，如此多元場景下，多維度測試也顯得更為重要。

當我們在檢視軟體測試的種種時，也可以看到它在這幾十年來間的演進，不斷的有新的實踐法則、新的方法、新的框架、新的工具出現，像是手動測試演化成手動探索測試並成為當代測試的基本方法之一，以及自動化測試與 CI/CD 的整合又為測試帶來更大的價值，自動化測試的範圍不僅針對功能性，更涵蓋了跨功能的面向，包括性能、安全性、可靠性等，這些都是當代資訊系統所重視的，也是作為一個高品質資訊系統所必須的，這些也是當代軟體產業把全棧測試視為一門專業的原因，我相信讀者之所以閱讀本書也是為了能提高自身軟體的品質而來，我首先為我們共同的目標致上敬意，並且歡迎您的加入。

為何我寫下此書

首先我想表示的是，在我之外有許多的測試專家也有能力寫出這樣一本專門談測試的書籍，只是他們可能沒有足夠的時間，或者他們並不打算成為一名作者，不論原因為何，因為他們的謙讓才給了我這樣的機會，我也很榮幸能成為此書的作者（但我更希望在我還是新手時就有人寫一本這樣的書，就可以讓我省下大量的精力去查閱、搜尋、測試一大堆的工具和方法，窮盡數年才換來這一身功力。）

在多年的顧問生涯中，我觀察到那些有規劃完善測試策略的團隊大多是成功的，反之缺乏測試策略的團隊大多是失敗的，例如某個團隊只做了 UI E2E 測試，而最終疲於奔命在應付維運工作上，或者是只做了手動測試，最終上線後跑出一大堆當初沒測到的問題，又或者是只做了功能性測試，忽視了非功能性的問題，最終這些團隊都因為不完善的測試策略導致軟體品質低落，對內士氣降低，對外缺乏競爭力，在當代的軟體產業中，還會看到這類只注重單一面向的測試規劃令人感到不可思議。近年來測試已經被視為一項專門的學問，對於上述的現象，我認為是源自市場上缺乏真正懂測試的專家，或許是因為大廠間的人才戰爭導致優秀的人才都被它們納為己用，但我認為擁有正確的知識不只是大廠的專利，更應該散播到每一個地方。

雖然市面的測試工具都已經有各自的教學，但那些教學彼此獨立缺乏連貫性，令人難以整合運用，而對於一些比較專門的測試領域，特別是安全與可用性測試方面，也欠缺較為入門的教材，而此書出版的目的就是作為一本完整的測試資源手冊，讓網頁或手機應用的測試新手都能藉此提昇自身的測試技能，往上升級成「高級新手」。

如果您對「高級新手」這個稱號感到好奇，其實這是來自德雷福斯模型（Dreyfus model），這是一個技能學習階段理論的分類模型，它將人們獲取技能的程度分為五個等級：新手、高級新手、勝任者、專家，而本書的目標是透過傳授十種主要的測試技能與實務範例帶領讀者通過前面兩個等級，而對於第三級勝任者而言，得需要更多更廣的實務經驗才有可能到達，我相信這本書會盡可能帶領讀者前往這樣的目標。

誰適合閱讀此書？

本書設定的讀者為測試新手以及想要拓展既有知識的測試專家，也包括任何有沾到測試工作的工程師，例如開發工程師、DevOps 工程師等，他們也都可以透過本書豐富他們對測試的理解，不論角色為何，讀者必須對程式有基本的認識，特別是 Java，本書大部分的練習都是以 Java 撰寫，也有少部分的 JavaScript，如果您是軟體界的新手，那建議先去了解軟體開發流程，不論是敏捷或瀑布式開發流程皆可，有了基本的開發流程知識後再來閱讀本書。

本書導讀

在最開始我們會介紹到所謂的全棧測試，以及說明十種用於確保軟體品質的測試技能，有了基本認識後，後面我們為這十種技能開設獨立的章節講解其中的細節，這些章節都具有相同的構成要素：

- 與該章節主題相關的背景知識會統整於「組成元素」一節，如果您對該章主題不太熟悉，可以透過此部分來建構對其的基本理解，包括該技能的 what、why、與 where，讓您了解該技能是什麼、為什麼、以及用在哪裡。
- 接著會談到該測試技能的應用策略，包括該在哪些場景下應用此測試的議題。
- 後續是實務練習，帶領讀者一步步的以各種工具實踐該測試。
- 此外部分章節還有額外的其他工具或方法的補充，裡面會談到與演練相關的其他工具，或者是能為演練帶來額外效益的補充工具，讓讀者能藉此更加全面掌握該章所要傳達之技能。
- 最後一部分則是我本人的觀點，分享本人過往的經驗與觀察，以及本章要點，回顧此章所談到的重要議題。

經歷完這十章測試技能後，我們會談到測試的第一性原則以及個人所需的軟技能，最後是專為走在潮流前端讀者準備的額外章節，介紹新興應用所需的測試技術，例如 AI/ML、區塊鏈、IoT、AR/VR 等，讓身處前沿應用產業的讀者，也可以從中學習到該領域的測試技術。

本書編排慣例

以下是本書的編排慣例：

斜體字（*Italic*）

用於表示新詞彙、URL、email 信箱、檔名、副檔名等。中文以楷體表示。

定寬字（`Constant width`）

用於表示程式原始碼，以及在段落內表示程式中的變數、函式、資料庫、資料型態、環境變數、陳述式、關鍵字等與程式相關的元素。

定寬粗體字（`Constant width bold`）

用於表示命令或應該由用戶輸入的文字。

定寬斜體字（*`Constant width italic`*）

用於表示應該被用戶輸入值所取代的字串，或者表示會根據前後文所變化的字串。

第一章

全棧測試簡介

在今日的社會，數位化已是企業不可或缺也是規模增長的一部分，許多企業走在前方引領潮流，而有些則還處於數位轉型的前端。

數位化是企業走向全球化的關鍵要素之一，它能為企業帶來更高的觸及率以及更高的營收，不論是醫療、零售、旅遊、學術、社群、銀行、娛樂，幾乎任何產業都把數位化戰略視為擴大用戶和利潤成長的關鍵之一。

在數位化與現代化的道路上，創新是背後驅動的力量，幾十年來，只有持續創新的企業才能為自己帶來蓬勃的發展，Netflix 是其中典型的例子，Netflix 開始於 1990 年代，最初只是一間線上 DVD 出租店，它們在 2007 年大膽轉向線上影音串流，自此線上影音業務逐漸超越原本的 DVD 出租業務，後來他們也開始自行投資製作一系列 *Netflix Originals* 原創節目，到 2021 年底，Netflix 已是全球最大的線上影音業者（*https://oreil.ly/AyHBL*），在全球擁有超過兩億以上的訂閱用戶。

隨著推動企業創新的需求增長，技術也在持續不斷的向前演進，排隊買票只為了看一場電影、開車出遠門只為了買一樣東西、手寫清單四處找尋只為了採買生活雜貨，這些場景今日都已不復見，科技帶給我們過往不曾有過的便利生活，想要看片在客廳就有影音串流、想要試衣服有虛擬試衣間、想要採購生活雜貨有定期配送服務、想要喝咖啡有聽得懂語音指令的咖啡機，不止這些，還有更多的一切都源自於科技與創新。

隨著科技的快速發展，產品策略也必須更加多元才能滿足不同的用戶需求並藉以維持企業自身的競爭力，這絕對不是建個形象網站就能完事的了，必須有更長遠的眼光，看看 Uber 或 Lyft，它們的叫車服務深入每個管道，可以從網站叫，也可以從 Android 或 iOS 上的應用叫，還可以用 WhatsApp 叫（*https://oreil.ly/1ijA9*），像這樣盡可能覆蓋所有通路的產品策略才有可能讓他們更大、更強，直到超越對手。

創新與多元化讓企業得以接觸到大量用戶，隨之而來的挑戰是如何更進一步擴大規模、營收、與用戶，以 Amazon 為例，已經是產業巨頭的它透過交叉銷售策略更進一步壯大自己，最初的 Amazon 只專注於圖書銷售，而後它的銷售品項一路擴展，從生鮮食品到電子產品、服飾、珠寶等等無所不包，幾乎進軍了每個人日常生活的所有領域。

但為什麼我們要在一本以軟體測試為主題的書談這些呢？因為當前的軟體產業之所以創新的目的，就是為了滿足前述的商業需求，以新技術實現新的產品概念，將其實踐，最後擴展到全球的每個角落，在這樣的過程中，顯然軟體開發團隊是站在最前沿的角色，特別是還要達到所謂的**高品質**時。在現今競爭激烈的商業環境中，軟體品質已是絲毫不可妥協的，一旦放棄對品質的追求，就相當於參加一場註定會輸的競賽，從過往的案例中屢屢可以證明這一點，例如印度兩大電商巨頭 Snapdeal 和 Flipkart 在 2014 年都為了當時的銷售旺季籌劃多時，備足了糧草，但遺憾的是 Flipkart 卻承受不住「Big Billion Day」湧入的流量而頻頻當機（*https://oreil.ly/C20pD*），不僅損失了許多顧客，更反向壯大了對手，另一個典型的例子是 Yahoo!，儘管它是市場的先行者，但因為忽視搜尋品質的重要性（*https://oreil.ly/CiYDd*），又不重視資安，導致在 2013 年發生有史以來最大規模的資料外洩（*https://oreil.ly/CP5ma*），超過三十億筆的用戶資料外流。以此為鑑，這些案例再再證明了軟體品質的重要性。

類似的案例在全球不斷上演，不論您的產品構想有多厲害，只要軟體品質有問題，唯一的結局只有走向衰敗，而顧客也只會毫不留情的投入對手懷抱。有些時候，面對上市時程的壓力，企業難免對軟體品質有所妥協，但必須謹記，這些妥協最後都會變成隱藏的負債，最好在對手發現以前解決掉這些負債，因此我們可以認定，優良的軟體品質是企業得以長期發展的基石，而優良的軟體來自軟體開發與測試的緊密結合，也來自仔細審視整個架構內的每個細節。為了讓讀者能走入這條軟體測試之道，後續我們將介紹何謂全棧測試，以及在網頁或手機上又是如何做到全棧測試的。

以全棧測試塑造品質

一開始我們先來談談什麼是**軟體品質**，過去我們認為軟體只要沒有蟲就是有品質，但如今大概沒有多少人還是這麼想的了。如果去問用戶什麼是品質，他們多半會說要簡單、要好用、要美觀、要保護隱私、要快、要不斷線；而如果去問企業什麼叫品質，您會聽到要能帶來報酬、要有即時分析、要不停機、不要被供應商綁死、要能擴展、要注重資安、要合規、要一大堆東西，以上種種概括而言都是今日所謂品質的一部分，任何一部分的缺失都意味著品質面的缺陷，這也是為何測試如此重要的原因。

雖然前面提到要達到所謂高品質的要求那麼多又那麼雜，但所幸我們有一系列的工具和方法來幫助我們完成這些任務，這些工具就像是品質路上的橋樑，透過它們我們才得以踏入品質化境，然而對我們而言，更重要的是掌握在開發或測試上運用這些工具的技能，唯有如此才能真入涅槃。本書旨在帶領讀者建立一系列的工具橋樑，以及傳授能成就高品質軟體的那些心法與技能。

概括而言，測試指的是確保應用的行為如所預期的手段，一個成功的測試，需要同時關注宏觀與微觀兩大層面，這與應用本身的粒度密切相關，微觀方面包括測試每個類別內的方法、測試欄位的輸入值、測試紀錄輸出的正確性、測試錯誤碼的正確性等等。而宏觀方面，包括單一功能測試、多功能整合測試、E2E 流程測試等，但測試並不僅於此，還有更多非功能的測試面向：安全性、性能、可用性、易用性等等，這些都是成就一個高品質軟體所必須的要素，以上種種不同規模、不同面向的測試，我們將其概括稱為**全棧測試**！參見圖 1-1，全棧測試涵蓋了影響軟體品質的各個層面（如資料庫層、服務層、UI 層）的測試，以及橫跨整體應用面的測試。

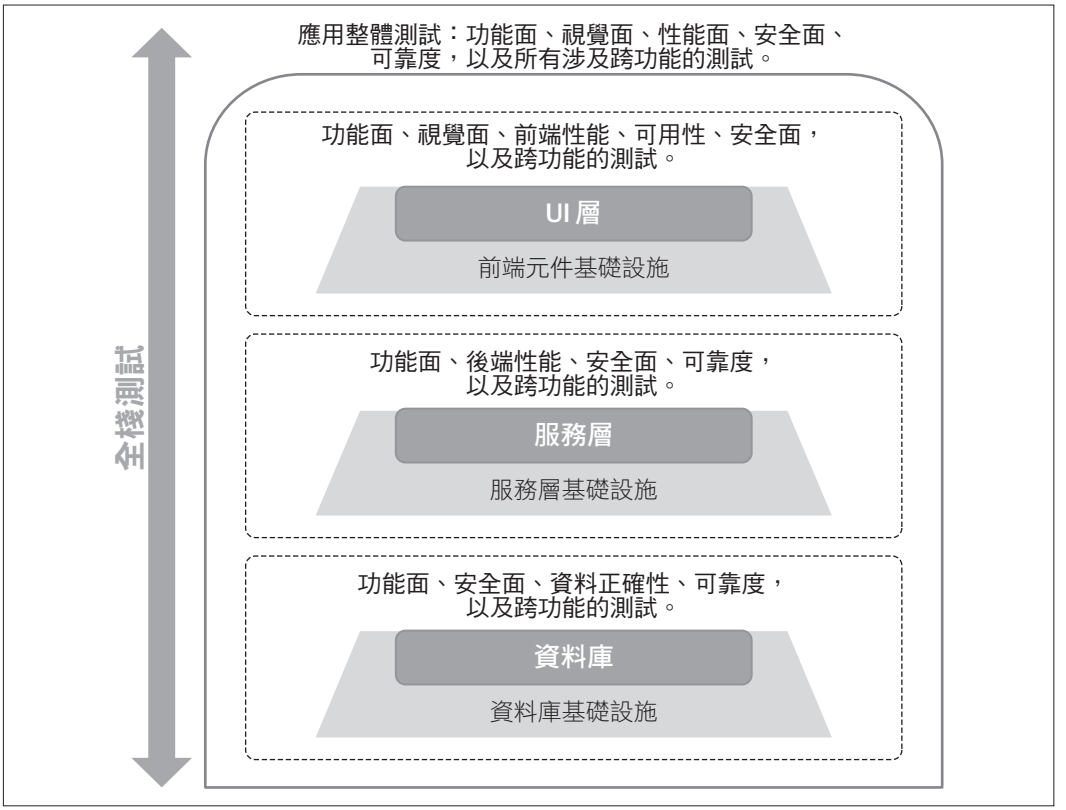

圖 1-1　全棧測試示意圖

在全棧測試的概念下，開發與測試應該是互相依賴的，就好比鐵路兩側的軌道，我們必須同時沿著它們前進才能抵達品質車站，否則脫軌將在所難免。以一個電商系統為例，裡面某個計算總金額的程式，我們既要考量到計算的正確性也要考慮到它的安全性，如果錯失其一，那就好比軌道上的缺口，如果還無意識地在不穩固的基礎上開發，最後的品質必然也是低劣的。想要把測試意識深入人心，必須先擺脫測試是開發後的事的傳統觀念，全棧測試下的測試必須與開發一同進行，並且要成為每個交付週期的一部分，讓團隊能快速取得每個週期的回饋，把測試從傳統的開發後移往開發中，這樣的測試概念我們稱之為**左移測試**（*shift-left testing*），左移測試帶給我們更有品質的產出，它也是本書遵循的關鍵原則。

左移測試

如果我們畫出傳統的軟體開發流程，依序會是**需求分析**、**設計**、**開發**，最後才是**測試**，如圖 1-2。而左移測試則提倡把測試擺到最前面，這帶來的是更高品質的作品。

圖 1-2　左移測試

讓我們用蓋房子來比喻，試問把房子全部蓋完才做品質驗收是合理的嗎？如果最後才發現尺寸錯了、承重性不足該怎麼辦？左移測試就是為了避免上述的窘境發生，最好在規劃之初就導入品質測試，並且隨著開發的腳步一路檢查，這樣才可確保最終產品的優良。

伴隨開發過程的持續性品質檢查意味著這是一連串重複迭代的過程，隨著開發進度的推進，檢查的項目也逐個累積，這樣的模式也更有利於引進迭代中的小變動，回到房子的比喻，就好比在蓋完每一道牆後就立即實施檢查，如此就算有什麼問題也可以馬上修正，像這種廣泛的測試工作需要仰賴高度的自動化測試與 CI/CD 機制，程式碼變動後由 CI 發起自動化測試，讓測試腳本週期性的對每次的變動跑一輪測試，如此來確保應用是

有被持續測試的，並且相較於耗時耗力的人工測試，自動化測試在各方面也都有成本上的優勢。

回到軟體開發層面，我們試著把左移測試展開到開發流程中，以敏捷開發為例，在開發週期內的各個階段分別施作不同的品質檢查，整理如圖 1-3。

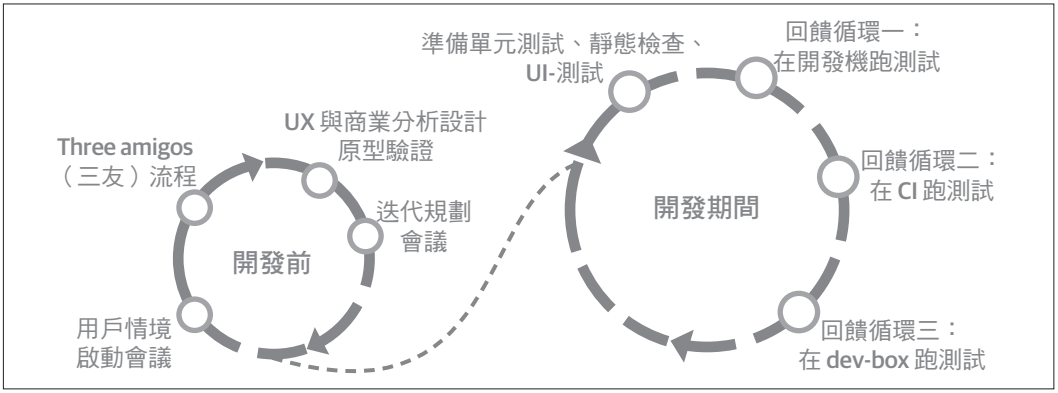

圖 1-3　測試左移概念下的一系列測試項目

圖 1-3 從左邊開始是一系列著手開發前的檢查項目：

- 在分析階段有個被稱為 *three amigos*（三友）流程（*https://oreil.ly/WFABh*）的步驟，此三友乃商業代表、開發、測試員，由此三角色負責彙整一個需求的不同面向，他們給出各自角色上對該需求的洞察，例如某些場景下的極端使用案例、某些沒被注意到的商業要求等等，這是左移測試的第一步，用以確保一個功能需求是經過彼此驗證的。
- 與此同時，商業代表還會與 UX 設計師一同合作對 UI 設計做出初步驗證。
- 當前述兩個測試確認後，就開始迭代規劃會議（*iteration planning meeting*，*IPM*），用於在每次迭代 / 衝刺（sprint）前確認情境（user story）和細節，這個場合讓開發團隊內的每位成員，都有再一次確認所有細節的機會。
- 在每次迭代之間，會有新的情境啟動會議（*story kickoff*），此步驟用於討論用戶的需求情境與極端使用案例，但沒有 three amigos 那麼正式，經過此階段，我們可以認定情境已是經過確認的了。

在開發期間，則有下面這些檢查項目，我們可以用這些檢查來得到品質上的回饋：

- 開發者根據使用情境撰寫單元測試，並將測試納入 CI 流程中，此外，程式碼的靜態檢查與分析工具也應納入 CI，讓我們可以取得每次 CI 跑完測試的回饋。
- UI 測試方面，有些人會在開發時就邊寫測試，並納入 CI 流程中，而有些人則會到開發後才寫 UI 測試，這兩種都是常見的做法。
- 在每次提交程式碼前，我們應該在自己的機台上跑一次測試，這是我們得到的第一輪回饋。
- 在提交程式碼後，CI 也會自動在測試環境跑測試（包括單元測試、服務測試、UI 測試等等），這是我們得到的第二輪回饋。
- 第三輪回饋來自 *dev-box* 測試，做法是讓商業代表和測試員在開發者的機台上實際操作實行人工探索性測試，用於快速驗證該次新開發的功能是否如預期。

經過這幾道嚴格的考驗，開發團隊馬上就能取得超過半數以上的品質回饋，反之如果走傳統老路，那就只能在開發後再用人力慢慢的檢測，前述的開發流程也就是測試左移的實踐，讓測試者在開發的各個階段就能取得不同方面的品質結果，而不用等到最後一刻才能驗證功能的正確與否。

因此我們可以說，透過在開發機和 CI 上一輪又一輪的測試，左移測試不僅能防止產品缺陷發生，還有早期發現、早期治療之效，對測試員而言，他們也有更多的機會去檢視軟體各方面的品質問題，確保交付出高品質的產品。

極限編程（Extreme Programming，XP）是一種融合了左移測試的敏捷開發框架，如果您想親身了解極限編程，在此推薦 Kent Beck 的《Extreme Programming Explained》（Addison-Wesley Professional 出版）

把測試左移的概念並不僅限於功能性測試，也可以應用在安全測試、性能測試等其他測試，以安全測試為例，可以用 Talisman 這類工具，在每次提交前自動掃描程式碼是否有不該出現的密碼、密文，關於這方面的實務演練，在後面的章節我們還會進一步討論。

總體而言，在開發流程中提早納入品質檢測，從原型設計一路驗證到產品需求，開發團隊成員各有其角色，這整個過程具體體現了「品質是團隊責任」（Quality is the team's responsibility）的信念，因此我們也可以說，想交付高品質軟體，全體成員必須得有相關的品質學能！

十項全棧測試技能

過往談到測試，我們多半會將其粗略的分為手動測試、自動測試兩種型式，但隨著時代演進，這種分法就顯得過於籠統，面對網頁和手機應用當道，現在的人們必須學習更新、更多元的測試技能才能滿足當代多元應用的品質要求。圖 1-4 展示了當代所需的十項全棧測試技能。

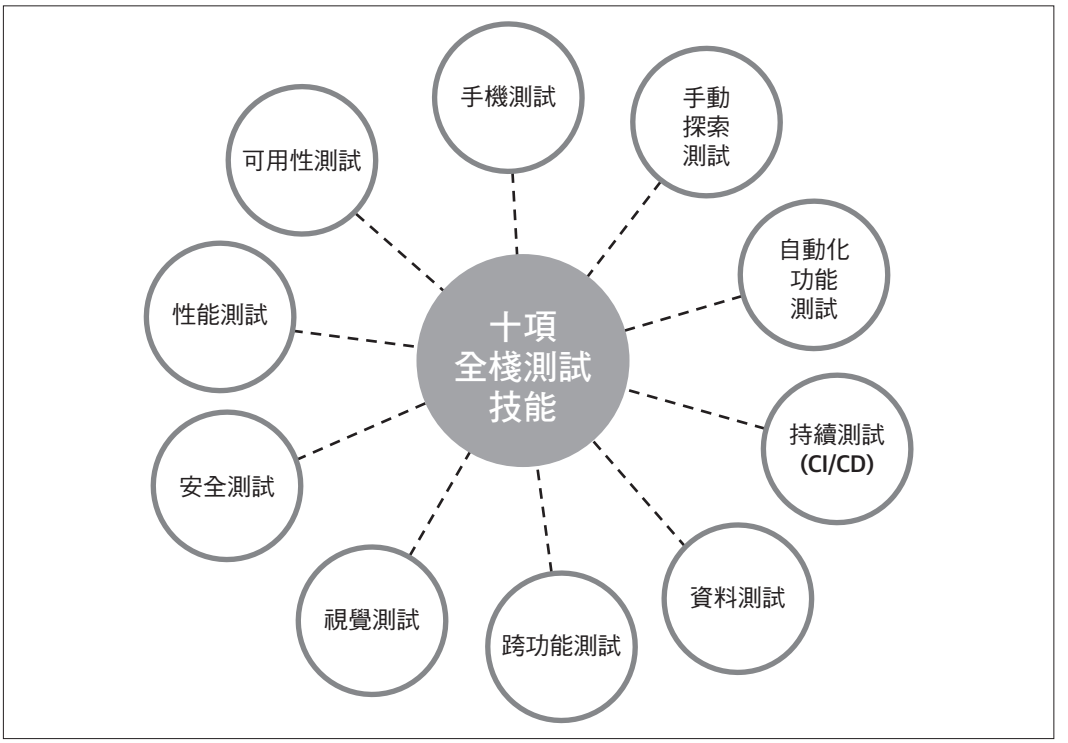

圖 1-4　符合當代網頁、手機應用品質需求的十項全棧測試技能

讓我們看看它們分別是什麼，以及為什麼我們應該學習這些技能：

手動探索測試

首先必須澄清，手動探索測試並非手動測試，手動測試純粹就是拿需求清單一項項檢查，沒有分析的思維，相較之下，探索測試講求的是依照需求的真實情境，在測試環境下進行測試，並從中觀察應用的行為是否如預期。測試者需要具備邏輯和分析思維，對於企圖打造零臭蟲的應用的我們而言，手動探索測試是最為重要的測試，在第 2 章我們會進一步探討各種探索測試的方法。

自動化功能測試

前面提過，自動化功能測試是左移測試的重點所在，自動化測試能大幅節省投入人工測試的精力，尤其是當應用的功能越來越多時自動化更顯效益。簡而言之，所謂的自動化測試就是用程式測程式，測試過程中無須人為介入，自動化測試通常必須借助一些工具來協助我們針對應用的不同層面建構相對的測試程式，然而也必須注意到，自動化測試有一些必須迴避的反模式，我們在第 3 章會進一步探討自動化測試的方方面面。

持續測試

相較於大版次發佈模式，持續交付（continuous delivery，CD）指的是小步伐、快速迭代的交付模式，在商業層面，透過持續交付讓我們可以更早更快獲得收益，並且可以快速驗證產品發展策略在市場的反應，為了達成持續交付，也就必然得透過持續測試來確保每一次交付的品質，而最好的做法就是把測試融入 CI/CD 管線（pipeline）流程中，如此既確保了品質也簡化了測試流程。對於持續測試，需要考慮的是該在哪個階段進行何種測試，這影響到測試與 CI/CD 的整合方式，也影響到我們會何時取得測試報告，關於這些議題，我們會在第 4 章做進一步討論。

資料測試

您可能聽說過「資料就是金錢」或者「資料是新時代的黑金（Data is the new Oil）」，這類說法凸顯了當今資料正確性的重要。如果把用戶資料搞丟或搞錯，無疑地我們將徹底失去他們的信任，在資料測試方面，我們必須了解各種不同的資料儲存與處理機制（如資料庫、快取、事件串流等），並為它們設計適當的測試案例。在第 5 章，我們會討論到資料測試的相關議題，包括當資料在系統元件間傳遞時應當採取的測試方式。

視覺測試

應用外觀與風格的優劣決定了品牌價值的高低，尤其是作為百萬級以上超大型 B2C 企業來說尤為如此，低下的視覺呈現對品牌的傷害也是百萬級的，因此我們有必要驗證應用的視覺讓用戶感受是和諧、愉悅的。在視覺測試方面，需要對 UI 及網頁或手機的人機互動有所理解，視覺測試也是可以自動化的，當然它與功能測試所使用的自動化工具必然有所差異，關於視覺測試化的工具，以及它與功能測試之間的差異，我們會在第 6 章進一步探討。

安全測試

在現今社會，安全漏洞的消息總是三不五時地傳出，連 Facebook 或 Twitter 這樣的巨頭都無法完全避免這類攻擊，只要發生問題，不論對個人或企業，都要付出巨大的代價，代價包括個資的洩漏、法律上的處罰、品牌商譽的損害等等，如今的產業界，安全業務已經是具有一定規模的利基市場，但許多開發團隊的做法還是只在開發的最後階段才進行滲透測試。隨著外部資安風險逐年升高，而具有資安專業的人員又普遍缺乏的情況下，開發團隊自身最好也有基本的安全思維，以及落實基本的安全測試，在第 7 章，我們會談到如何以駭客的角度檢視應用的安全性，以及介紹自動化安全測試工具。

性能測試

對企業來說，即使是微小的性能下降也可能導致巨大的財務和商譽損失，比如我們前面提過的 Flipkart 就是很典型的案例。在性能測試方面，我們會針對應用的不同層面設定幾種關鍵指標（KPI）並加以測試，性能測試同樣可以自動化進行並整合進 CI 管線（pipeline）流程中，納入成為持續測試的一部分。我們會在第 8 章深入討論如何將性能測試左移，以及與之相關的測試工具。

可用性測試

網頁或手機已經是每個人日常生活的一部分，為殘障人士設想應用的可用性不只是為了滿足法規要求，更是基於普世道德價值觀。想要進行可用性測試，必須先了解相關的法律規定或標準，再利用手動或自動化測試機制來驗證應用是否有符合那些標準，在第 9 章我們除了談應用可用性測試方法外，也會談到可用性如何為企業帶來正面效益的議題。

跨功能測試

不論是來自用戶或開發商本身，都有太多的品質面向需要被滿足，包括可用性、擴展性、維護性、觀測性等等，這些品質需求都不僅止於功能面上的追求，這類需求我們統稱為**跨功能需求**（*cross-functional requirements*，*CFRs*）。對大部分人而言，他們的眼光大多只注意到功能面有沒有被滿足，但跨功能面的需求對軟體品質也有著重大影響，這類需求一旦形成缺陷，同樣會喪失用戶的滿意度，也容易導致團隊的失敗，因此跨功能測試也應該是必要的測試之一，在第 10 章我們會進一步探討各種形式的跨功能測試的方法與工具。

業界也有人把跨功能需求也被稱為**非功能需求**（*non-functional requirement*，NFR），在第 10 章我們也會談到這兩個詞彙之間的微妙差異。

手機測試

在 2021 年，Google Play 和 Apple App Store 這兩大手機應用商店上架的應用總數有多少呢？答案是嚇死人的 570 萬（*https://oreil.ly/L47MG*），這樣強勢的增長當然是拜手機設備的普及化所賜，數據也呈現相同的趨勢，根據網站分析公司 Global State 公佈的數據，在 2016 年來自手機的使用量就超過了桌機（*https://oreil.ly/mL3YF*），基於以上資料，手機應用與網頁應用對不同設備之間的相容性就尤為重要了。

前面我們已經談過好幾種不同的測試，這些測試在手機端也同樣需要被重視，但不僅如此，還有一些手機端獨有的測試，並且在手機方面的測試也必須從手機的角度去思考與規劃，因此我們針對手機測試開立了一個獨立的章節，在第 11 章，我們將會討論到手機測試與常規測試之間的差異。

以上談到的這十種測試技能，將能使我們覆蓋網頁、手機應用的全方面品質需求，如同我們前面提過的，身處開發團隊的每個人應該都要具備一部分的品質思維與測試技能，而這本書將帶領您一步一步的從實務演練中學習到這些技能。

本章要點

以下為本章要點：

- 當今的軟體品質要求不僅只是功能正常，還包括功能以外的面向（安全、性能、視覺等等），只要某個面向有所缺陷，那就意味著整體品質的下降。
- 全棧測試是面向一款應用的所有品質層面的測試概念，並藉此確保產出高品質的軟體。
- 全棧測試的目標就是交付高品質軟體，為了滿足這樣的目標，我們應該將測試左移，讓測試從需求分析階段就一同展開，並持續與整個開發週期一同進行。
- 全棧測試體現了「品質是團隊責任」的信念，為此我們要求每個人都必須具有品質觀念，並且在各自的角色上為自己的品質把關，並且團隊同仁必須學習相關的測試技能，並加以精進。
- 籠統的自動測試、手動測試二分法已經不足以表示當前多元的測試形式，在本章我們介紹了十種不同的測試，他們都是滿足當代對軟體高品質需求的根基，而在後續的章節中，我們將會一一深入探討它們各自特性。