持續測試權威指南

2018

如何在持續測試之旅中取得成功

第 1 章

隨著業務步伐不斷加快，公司開始體認到，若要維持競爭力，開發和發行軟體的過程需要改變。發行速度大幅加快。客戶無法再接受 6 到 18 個月的發現和修正週期所造成的延遲。一切都需要加速發展，而且需要加速準備就緒，並達到最佳狀態。

目前發生一場巨大的改變，測試責任現在往左側移動，更接近軟體開發生命週期的開端，然後一路到各個階段。這完全打破了開發人員和測試人員兩分面各有職責的既有區隔。測試變得更加迫切，因為未能妥善執行測試的後果變得更明顯。

這是一種技術變革，如此的變革需要文化變革。隨著測試在 SDLC 發展和擴展，公司及其 IT 部門必須設法改變軟體開發和測試的整個文化。

也許最簡單的是：我們必須停止將測試視為一種事件。這並非只有在某個特定點才做的事情。這反而應該是每個人都必須完成的事情，甚至在開發之前就進行。產品從出廠送到客戶手中時，瑕疵和問題必須消弭於無形，不可陷入捕捉/漏掉的循環。

開發營運旨在打破開發和營運之間的隔閡，確保藉由敏捷的方法建構的軟體能夠在使用者準備就緒後立即快速部署，而且不會犧牲品質。但是，即使存在開發營運，公司仍然會在品質和速度之間進行權衡。

在現今的測試文化中，63% 的軟體開發延遲發生在整個生命週期的測試 QA 過程中，而且 70% 的測試仍然是手動進行。不過，這已經不再實用。您不能只進行手動測試，另外仍需要考慮其他問題：

56% 的關鍵相依性不復存在

50% 的時間用於尋找測試資料

64% 的瑕疵在需求階段發生

缺陷 (尤其是會接觸到公眾的缺陷) 對於資料、業務流程，客戶採用、留客率和品牌造成損害。使用過時的瀑布開發法或甚至敏捷的過程無法以速度和品質來改變軟體。

我們需要更全面、更動態且更流暢的過程才能確保軟體在開發和部署時達到既定品質，而且從一開始就偵測到錯誤，而不必等到生命週期結束時才進行驗證。這就是為什麼我們需要踏上持續測試之旅原因。

什麼是持續測試之旅?

持續測試是對於 SDLC 中的每個活動進行的測試，以便意外行為發生時立即發現並予以修正。

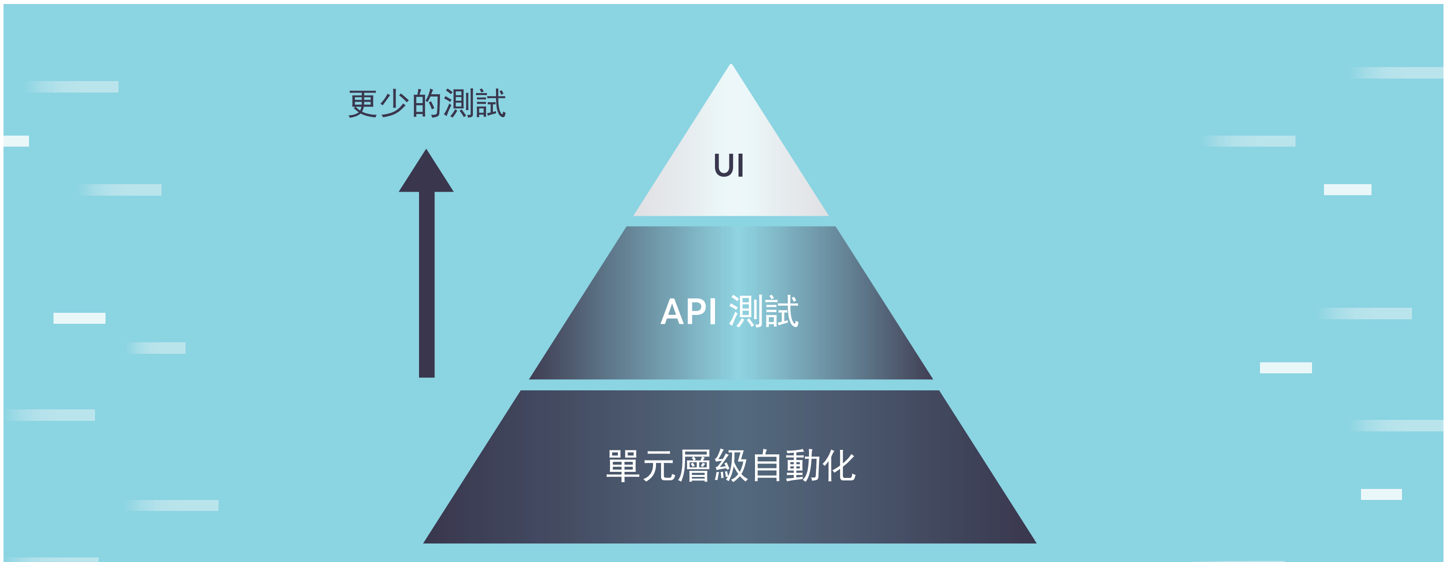
持續測試是對於應用生命週期之中從需求到生產的每個活動進行持續的基礎測試，以確保達成如預期的商業價值。

IT 的生活向來是困難的挑戰。這個部門充滿了專業人士和專職工程師，致力於開發和建構符合規格的產品，不過這些規格可能不一定符合一般使用者的期望和限制。軟體的角色已經改變。這些產品必須能夠在各種平台上運作，而且必須提供不斷推陳出新的服務。

我們在測試、開發和建構軟體的人員之間築起無法踰越的高牆時，品質就會受影響。IT 不應再被視為後端辦公室的工作。這個部門必須在決策桌上佔有一席之地。

當不切實際的最後期限迫在眼前，並且將各種功能硬塞進沒有足夠的度量可以衡量品質的發布中時，品質也會受到影響。

一直以來，測試都是在獨立環境中進行，因此穩定度相當高。測試人員和業務人員會問：「我的測試有多好？就功能覆蓋範圍而言，我具有什麼樣的覆蓋率？除了我的程式碼覆蓋範圍之外，我的功能覆蓋範圍如何？我的驗收條件是什麼？我是否真的進行有效的測試？」許多測試人員都不知道。



測試自動化金字塔

為了確保速度，測試人員需要在單元層級進行大量測試，在 API 層級進行較少的測試，並在 UI 進行更少的測試。最終，測試應該變為並行或同時進行 (同時測試功能、效能和安全性)，然後提前進行，以確保從生命週期開始持續進行測試。

一般人往往將時間花在無價值的活動上，測試不正確的產品或不正確的組合。一個由八人組成的團隊可能會說「我們很優秀」，但是 40% 的工作一般必須經過重新測試和修正，這表示一個八人團隊正在做四個人就能完成的工作，而且進度往往落後幾個月。這些人需要執行價值流程分析，以確保藉由正確的方式進行測試。價值流程分析使用精簡管理意識形態分析現今流程，並尋求更良好、更有效率的流程。

大多數組織使用的工具都是「舊型」工具。這些工具適用於瀑布開發法，但不適用於使測試提前進行。這些工具對於以敏捷的速度進行的測試造成阻礙。對於持續整合環境，這些工具無法提供正確的自動化，因此開發人員拒絕使用這些工具。需要一套新的工具，不會妨礙速度和品質；這些工具可以在過程中確保品質，而不是像過去一樣「單純」測試應用程式。

但是，不僅測試自動化必須改變，需求定義過程也必須改進。需求必須明確、完整且可測試。

目前 **64%** 的瑕疵出現在需求階段。

需求通常以概略的形式提出，這對於具備領域主題專業知識的開發人員和測試人員來說已經足夠，這些人員能夠進行解讀並補足需求中的空白。但是，這會導致團隊成員以不同的方式解釋相同的事情，在編寫程式碼之前造成生命週期出現瑕疵。另一方面，需求可以藉由相當詳細的方式指定，篇幅長達數十頁之多。在這種情況下，需求需要很長時間才能指定，而且時間愈久愈難維護。更不用說開發人員和測試人員發現很難掌握如此多的資訊匯集而成的龐大需求。因此，必須使用對於開發營運的敏捷團隊適合而言更好的方法來指定需求。

品質保證也必須以不同方式處理。這個過程必須從資深領導階層開始，資深領導階層必須始終關注品質，而不僅止於關注交付。這是因為交付過程如今關乎使用者/客戶的整體體驗，而不再只是單純部署或發行軟體的行為。這為交付正確的需求提供脈絡，有助於開發人員和測試人員瞭解所建構的內容。如今，隨著每個月的發行，正確有效的回饋迴圈相當重要。QA 不可再被視為次等角色或屈居次等地位，因為這個部門到目前為止也扮演同樣重要的服務角色。

若要實現這項變革，必須進行文化和組織變革。這是人為問題，並不是技術問題，而且對於成功的持續測試之旅相當重要。隨著任何技術或過程產生變化，例如持續測試或提前進行，有些人可以因應變化，而其他人則無法因應。這種變化也對於卓越中心 (CoE) 構成威脅，因為測試人員質疑如何將「測試」交由未經訓練的人進行。

這就是為什麼「卓越中心」Center of Excellence (CoE) 必須轉變為「能力中心」Center of Enablement (CoE) 的原因。現在必須承擔 SDLC 測試責任的人必須熟悉正確的流程，並擁有更容易進行測試的工具，而且具備最多技能和經驗的測試人員必須啟用這類工具。這需要從上到下以及從下到上的變更管理支援。

隨著測試提前進行，效能和安全性也必須同樣提前展現。現在不應再將這兩者視為「無功能」，必須將這兩者視為等同於功能測試。目前在進行生產的部署之前，並沒有足夠的安全性測試，現在該是將安全視為重要角色的時候，將專門的安全性測試做法納入持續測試過程中。

應用程式投入生產後，可能會發現在實際使用者流量下無法正常運作。更多的措施似乎可做為充分的快速解決方案，但這只能在雲端的基礎架構中實現。因此，需要確保在生產之前測試應用程式效能，由開發人員一直提前到單元層級進行，然後，隨著程式碼的單元開始形成實現案例、功能和架構的更大組件，逐漸擴大到效能測試範圍，始終運用先前建構而成的較小效能測試。公司需要運用適當的工具和流程來確保這種情況發生。

從上而下和從下而上的文化需求變革措施與卓越的領導力相結合。整體而言，這一切需要持續的測試文化。管理人員和 IT 經理必須與測試人員和開發人員溝通，並專注於共同制定計劃。這些人員可透過參與整項計劃而成為解決方案的一部份。讓這些人員有從失敗中學習的機會。

第 3 章　構成持續測試的項目

在持續測試中，每個人都會撰寫程式碼並進行測試。該團隊單獨和共同負責工程作業並提供高品質的軟體。這突顯出從瀑布開發法到 Scrum 的轉變，藉由持續測試，將敏捷和開發營運靈活融合為企業敏捷度。

集體責任藉由回顧反省來檢討和調整過程及其結果。這將重新界定完成的定義。建構持續測試思維能夠為開發人員提供快速回饋，確保開發人員的程式碼能夠順利運作，以便快速送交發行管道。這表示需要建構或尋找資料用於生產前和生產中進行的所有功能和非功能測試。

構成持續測試的三個關鍵項目是：

人員

為開發人員提供測試功能。開發人員和敏捷團隊需要瞭解測試是他們的責任之一。

程序

一切自動化。

科技

提供開發人員將使用的測試解決方案這表示開放原始碼在雲端中「做為程式碼」呈現。

組織必須確保持續測試的定義廣為瞭解。這仍然是相對較新的做法，和其他許多軟體開發領域一樣，定義和做法可能會有所不同。

持續測試是對於 SDLC 中的每個活動進行的測試，以便意外行為發生時立即發現並予以修正。

這可以想像成自動化的軟體裝配線，其中包括每個步驟的測試，從自動進行的需求收集開始，然後從第一次程式碼簽入一直到生產。這包括測試的所有層面，一直到生產中的整合和效能測試與監控。持續測試應該是以應用程式為中心而且有效率且有成果的測試機制，涵蓋 CI、CD 和生產的端點到端點範圍。其目標是盡可能在最短的交付週期內，以最高的品質和最佳的客戶體驗為目標改變生產方式。

第 4 章　持續測試的原則

透過持續測試來加速和改進程式碼發行程序需要一系列規則來確保品質與時俱進。為了確保應用程式的品質，團隊必須在整個應用程式管道中，以執行小規模品質檢查的形式加入更多新型的測試做法。這將能夠不斷測試小部份程式碼。

測試自動化並不一定是必須先解決的問題。原因是，即使在 API 測試和 GUI 測試自動進行並整合到持續整合代理程式之後，這些測試仍具有相依性，在執行之前必須滿足測試資料、介面和環境的需求。您必須：

- 消除環境限制，並且讓開發人員和測試人員有餘裕完成作業，即使是手動作業都能完成。將您未擁有、未控制或不想測試的所有介面虛擬化。
- 運用短暫的測試環境。
- 自動進行測試資料提供和管理，以手動或自動的方式隨需進行測試。
- 自動進行測試，以便使用適當的測試資料針對您的虛擬化介面進行。
- 讓您的管道協調引擎設定為不會對上述步驟進行手動干預。
- 接下來，專注於採取補強原則。

持續測試的 11 個原則

1. **虛擬化環境**
   持續測試表示更頻繁進行測試。這表示您將更頻繁進行觸及多個環境。如果這些環境一直不存在，這會構成問題而造成瓶頸。某些環境可透過 API 存取，其他環境則可透過各種傳訊介面存取。這些環境可以採用最新架構進行建構，而其他環境則是單一的舊型傳統用戶端/伺服器或大型機系統。那麼，您如何協調不一定保持其環境正常運作以供您測試的多個環境擁有者進行測試？將這些環境虛擬化能夠測試程式碼，完全不需要顧慮不想要變更的區域 (亦即其他系統和環境) 。透過虛擬化隨需使用這些環境，可以從開發生命週期中消除這些約束，因為這些環境始終可供使用。您可以控制這些環境，根據需要進行調整。

2. **測試資料管理**
   對於正面和負面情境，您必須擁有正確的資料和適當的資料多樣性。您不會在生產中發現所有多樣性，這一點相當重要。因此，綜合資料產生是實現持續測試，完全確信資料中的任何個人識別資訊不會面臨風險。此外，您無法從生產中擷取資料、調整資料，並以應用程式管道所需的速度提供資料。

3. **測試自動化**
   在完全協調的應用程式管道中，由於與應用程式程式碼無關，因此現有的指令碼會失敗。所以，沒有人會相信結果。您需要可靠的測試自動化來進行持續測試。

4. **管道協調**

   骨幹。一切都與此息息相關。這必須與您的自動化套件整合。您必須瞭解測試的作業方式、測試結果的解讀方式，以及測試的調整方式。如此一來，即可透過在管道中整合持續測試屬性的方式設定持續測試。確保過程完全透明，而且每個人都可以完全瞭解管道中正在運作的內容。這是一套自動化工作流程工具，可以進行所有自動化測試，並且與透過管道進行的程式碼部署活動完全整合。在任何開發營運採用計劃中，如果沒有標準化和自動化管道的可靠性和速度，則期望進行持續測試相當不切現實。

5. **API 測試**

   這將與測試金字塔保持一致。組織應盡可能在單元和 API 層級進行測試，並儘量減少對於 UI 測試的依賴。若要進行持續測試，您必須正確接受測試金字塔的概念，加強單元和 API 測試，並減少對 UI 測試的依賴，尤其是對於業務邏輯測試。

6. **效能/負載測試**

   即使應用程式功能良好，您也必須從根本改變您對效能測試的看法。您提前進行測試時，會在生命週期的早期測試較少的應用程式和基礎架構組件，這表示測試範圍較小。但是這些的數量也相當大。使所有敏捷團隊中的每個人都可以存取該功能。這表示所有開發人員和測試人員都必須能夠建立效能測試並自行執行，完全不需要向任何人發送任何請求。

7. **安全性測試**

   開發人員需要明確知道對於程式碼狀態的期望，以便允許或拒絕發行。他們必須獲得適當的訓練和工具來衡量他們的程式碼的安全性。開發人員的教育必須持續進行，因為應用程式安全領域一直不斷在發展。應用程式管道必須設定為自動執行靜態分析、動態分析和軟體組合分析，以便發現和突顯任何安全問題，而且開發人員必須藉由這些問題而有更加發展的機會。安全性測試需要在生命週期的每一個步驟都進行。

8. **測試導向開發和行為導向開發的驗收**

   採用個別使用者案例的驗收條件，並建構測試以確保滿足這些條件。因此測試完全集中於衝刺計劃，並確保開發人員開發企業期望的內容。隨著時間推移，團隊將定義更詳細的驗收條件，因此測試也必須更全面。

9. **自動產生測試**

   手動設計和編寫手動測試或自動測試指令碼的活動是一個瓶頸。在衝刺計劃開始時，對於手動和自動測試一般會產生新的驗收層級測試，因此，開發人員開始簽入並合併最早完成的程式碼時，我們就可以開始進行自動測試。如此一來，開發人員就能夠在程式碼簽入於來源控制時立即獲得有關程式碼品質的回饋。

10. **需求工程**

    您必須將所有 SDLC 相關人員納入到持續測試之旅中。這表示能夠透過種更好的方式來傳達和共同滿足需求。愈早將需求工程納入您的持續測試計劃中，持續測試之旅就愈順利，因為團隊成員從工作一開始時達成共識。

11. **回饋迴圈**

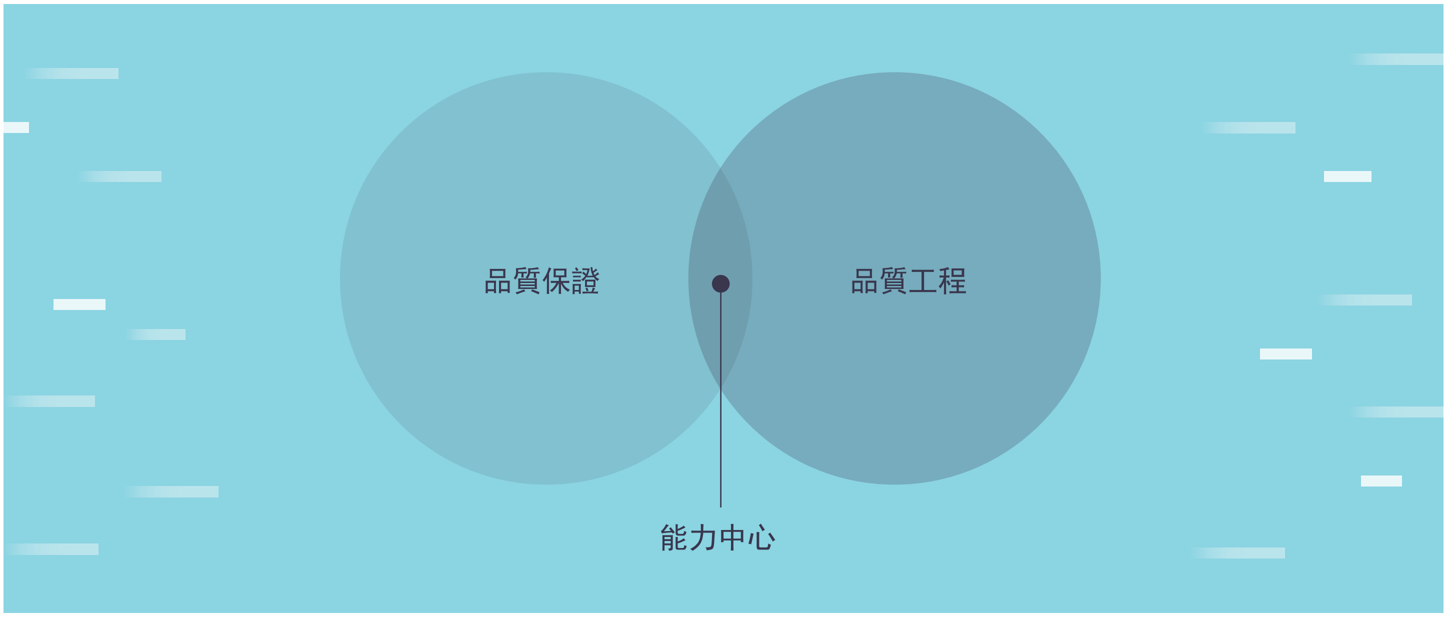
    這些對於成功的持續測試相當重要。即時儀表板。必須自動化，整個團隊必須擁有存取權限。整個 SDLC 的回饋迴圈，而不僅是生產，應該做為有助於您進行持續測試轉換的指引。

向持續測試文化邁進是實現開發營運的關鍵組成部份，但這表示要致力於新的知識和做法領域。這需要一些人逐漸養成習慣，而且有科學和做法知識的紮實根基。這種改進計劃之所以緊迫，是因為零售和 B2B 行業的客戶對於不佳的體驗極難以容忍，而且會更快速、更明顯影響公司的成功。

其中一些新做法可能自然會與現有文化背道而馳，但這就是變革和創新的本質。團隊需要這些新做法的介紹，並獲得訓練或協助管理過渡期。

例如，在編寫功能的程式碼時，應該在程式碼中嵌入發行說明，以便任何人都可以參閱。就像在專案管理領域一樣，建構足以讓其他人視需要接手的計劃。

品質保證 (QA) 應該發展為品質工程的能力中心。在其中，團隊可以在軟體加入如何測試、部署、監控和甚至自我修正的知識，以便應用程式團隊可以在 Scrum 中運用這些知識，完全不需要依賴外部員工。



應將所有內容視為程式碼，以便可以透過 CI 以可擴展的標準方式控制程式碼。這需要團隊思考如何測試程式碼，以確保如果要將程式碼整合到原始程式碼存放庫中，這將執行這應該執行的操作，而且不會對應用程式的其他部份造成負面影響。

參與的人員、開發人員和測試人員必須成為這種新文化的一部份。應該有開發營運組織負責僱用或重新訓練人員，讓人員成為具備許多能力的工程師。這可能包括矩陣式組織，讓應用程式開發團隊視需要僱用或運用所需的技術人員。對於人的變革管理和對於持續測試的實體轉變同樣重要。

從平台的角度來看，雲端是極為看好的成功因素。僅使用內部部署系統和舊型系統無法實現持續測試之旅。

第 6 章 我在持續測試之旅的哪個階段？

在持續測試之旅的早期階段，領導者和相關人員必須停下來問自己：「我在部署時間表的什麼階段？」首先要瞭解持續測試的必要性，然後對其進行充分定義，這一點相當重要。在準備啟動時，必須詢問團隊目前的進展。

此時需要謹記兩個關鍵點

- 始終將客戶定位在變革過程及其最終持續測試活動的中心
- 確定如何妥善衡量品質。

此時需要採用新技術，此外，識別和消除完成度有其障礙存在。

採用的技術

1. **改善測試人員與每個開發人員之間的關係**
   保持小規模的團隊，並鼓勵團隊之間的協作，而且使報告易於存取和線上共用。

2. **使自動化成為優先事項**
   不是去除所有手動測試任務，而是採取「自動化優先」的思維。重點必須放在反覆執行的區域。在本質上，這就像先裝設水管，以便讓水流過，完全不需要擔心或顧慮漏水。對應 SDLC 並確定自動化機會。

3. **變小**
   將工作切分為較小的部份，在撰寫程式碼和設計之後更容易測試。如此一來，即可更容易自動進行測試，而且還更容易部署。

4. **進行鉅細靡遺的追蹤**
   使用度量並設定通過/失敗標準。持續測試可立即識別做法是否有效，所以務必確保您可以輕鬆設定持續測試。

5. **獲得正確的持續測試工具**
   找到有助於您持續開發、測試和分析的工具。選擇最佳工具，以便輕鬆融入工作環境。選擇有社群支援的工具，每個人都分享各自的挑戰、解決方案和有趣的使用案例。

6. **開發顯示結果的系統**
   深入探究結果，因為唯有如此才能得知程式碼是否有成效，並得知哪些地方需要補強。定義您的 KPI 和驗收條件，並使這些可量化。建構儀表板以追蹤 KPI，包括基準線和後續變更。

向持續測試過渡的過程可能顯得繁重，最好從小專案開始，待確實成功之後，再處理其他小專案。這是一個反覆進行的過程，在這個過程中獲得小成果，並持續累積動能和知識。

失敗必將發生，但是，失敗發生時，團隊和領導者應該準備因應小失敗並向前邁進，花時間學習經驗。

推動組織責任制的資深領導階層將引發實質的不同。個別傳播者只能接觸 20% 至 30% 的人口。對於組織的其餘部份，則需要由領導者設定期望目標並堅持到底。

持續測試過渡有其障礙，必須逐一克服。這些可能包括無法存取和使用開放原始碼。工作文化 - 不僅是人，也包括做法 - 可能尚未充分準備。可能沒有足夠的時間開始實施、訓練和過渡到持續測試的過程。這可能與效率不彰的領導和舊型應用程式的存在相關聯。

此階段的最終目標是開發和部署持續測試成熟度模型。這表示在態度和能力方面建立正確的思維。在單元和 API 層級建構測試自動化，以進行功能、效能和安全性測試，並將 UI 層級的測試自動化降至最低，因為指令碼在 UI 變更時相當脆弱。加入回饋迴圈，其中包括生產前環境中做為管道一部份的效能和使用者監控工具。全遙測有助於解決瑕疵問題。

所有這些功能必須完全整合和協作，完全不能孤立。公司必須能夠接受和瞭解開放原始碼，因為這能夠讓一般人加速進行實驗和找出失敗的原因，完全不需要顧慮成本問題。公司也應確保測試資料可供充分取得，因為這是對於完成而言最重要的障礙之一。

第 7 章　度量

每個專案都需要度量來確定進度和品質，並確認商業價值。這與持續測試毫無差別。事實上，持續測試可稱為持續驗證，因為您經常會驗證您的應用程式或服務是否達到 (或即將達到) 預期的商業價值。一些整體基準包括客戶採用率、參與度和營收。團隊應該尋求瞭解和量化一般使用者的情緒，並將其加入到對於開發人員和業務的連續回饋迴圈中。

您也必須能夠追蹤有多少問題發生，也就是生產前和生產中有多少瑕疵，並按照您可以部署的速度進行條協調。隨著開發人員編寫愈來愈好的程式碼，應該可以預期瑕疵的數量隨著持續的發行次數而減少，或甚至隨這增加的發行次數而減少。

採用程式碼覆蓋範圍和所需的涵蓋範圍 (功能涵蓋範圍)，並結合這兩者追蹤整體功能涵蓋範圍。

確保瞭解品質對您的組織表示的意義並設定基準線。如果您不知道您從哪裡開始，則無法得知您是否改進。使用確立生產品質的 KPI。

瞭解單元測試和功能性測試所需的覆蓋範圍以及效能和安全性。這不應再被視為功能性和非功能性測試。

衡量生產的整體時間。持續測試的關鍵是縮短發行更高品質的版本所需的時間，以展現商業價值。專注於增加發行次數，同時降低生產瑕疵率，並透過自動化縮短產品上市時間。

透過界定與完成相對應的定義，促使一般人的思維超越通過/失敗的範疇。必須追蹤瑕疵，以便使用分析來進行改進。

您可以透過在較低的測試環境中進行測試來衡量持續測試成功，由於測試規模較小，因此可以快速完成測試。認知測試可以跟上開發的步伐，您可以隨時發送程式碼，而不僅僅是在衝刺計劃結束時。

在評估和衡量品質與數量時，請考慮下列因素：

- 上市時間
- 價值流程分析
- 人工測試執行的時間和成本
- 迴歸測試

透過使測試工具普及並將測試人員轉變為工程師，開始向開發人員轉移。

對於最有價值的持續測試度量，每個領導者都可以而且應該有自己的看法。需要追蹤的理想度量包括：

- 處理時間
- 發行
- 生產事件
- 商業價值

您從瀑布開發法轉向敏捷做法時，採用的過程需要測試效率，理想情況是從自動化和提前進行所達到的效率。隨著您在持續測試之旅中的發行時間愈來愈短，不僅可達到更多的自動化和更多的提前進行，而且關乎您如何進行測試。

過程中需要與持續測試有關的能力，這不是您在 CI 和 CD 之間所做的事情，而是能夠透過一系列的工具實現 CI 和 CD 的能力，這些工具支援這些進行測試的不同方式和看待測試的不同方式。

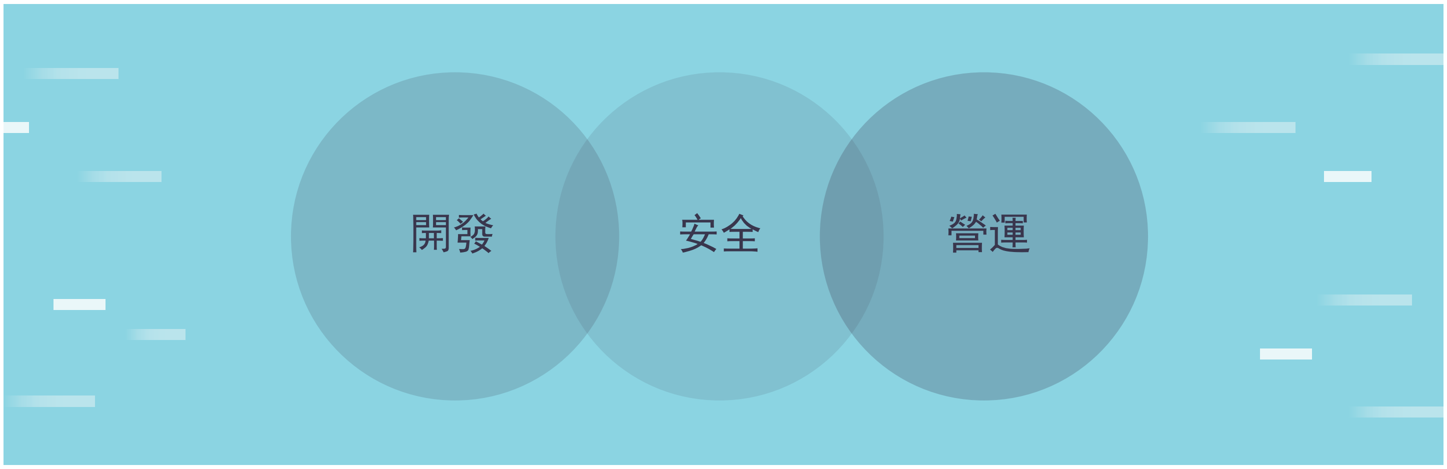
您需要有助於規劃、測試計劃和測試自動化的工具，然後透過 CI 和 CD 實施工具鏈實施這些工具。

在制定持續測試過渡的策略時，相當重要的是要考慮安全性、前端和中心。安全專家會相當正確指出，安全性一直被視為「終端」問題，同時是與開發人員相對立的行為。儘管開發人員希望將產品推向市場，但安全人員總是說「不要那麼快」。

安全性應該在一開始就安排在持續測試流程中。在識別和確認價值鏈的所有方面 (從開始到客戶) 時，這是一個主要關注點。回饋迴圈將有助於確認產品是否達到等級，並使團隊能夠在出現安全相關問題時找出問題並加以解決。

開發人員經常在不知不覺中造成安全性漏洞。持續測試安全性不僅是為了監控這些漏洞；這也涉及找出漏洞可以提供哪些訓練，使開發人員能夠更確實瞭解並注意程式碼的安全性，而不要一開始就出現這些漏洞。安全性必須成為合作夥伴、值得信賴的顧問和專家，為開發人員提供可在下游測量的訓練和工具。

安全性測試需要在每個步驟展開，因此開發人員需要進行安全性掃描，以便發現正在發生的問題。目前，安全性和測試都處在開發人員的對立面。這並不是安全性和品質之間的衝突，而是需要更多的合作來共同解決這個問題。



必須明確說明建構充足安全性的需求。安全性是每個人的工作，但我們需要確定何時完成。開發團隊必須接受訓練而能夠瞭解，並具備衡量需求的工具。在我們的管道中應該使用這些工具進行測試，以便我們不斷自我評估。和任何良好的開發營運程序一樣，安全性掃描能夠測量目前狀態的回饋，而且隨著時間讓團隊藉由訓練而提升。除了更安全的結果之外，透過教導開發人員第一次就正確編寫程式碼，也能夠大幅提高速度。

安全性的責任也有賴於公司高層的作為。從高層開始，問題和解決方案一直向下擴及組織基層。如果您只是從基層開始，可能會吸引少數幾個團隊，但要在整個公司中真正獲得採用，則需要獲得高層支持。

最後，安全團隊本身必須調整其思維，確保其成員正確看待本身是 IT 不可分割的一部份。

最後，在不斷變化和加快速度的年代，持續測試計劃必須滿足未來需求。為了滿足未來需求，您必須制定適合各方面發展的策略。這涵蓋從撰寫程式碼到測試，包括功能、效能和安全性。這也包括從測試資料產生到服務虛擬化的環境管理。

必須保持明確的路線，確保用於持續測試的工具採用相同的模式。這種做法必須與公司的業務策略角度保持一致，而且 IT 需要成為該業務策略的一部份，而不僅是扮演支援角色。

公司需要專門的資源來完成這項工作。這不應該完全交由第三方進行，但也不應該排除第三方。可以使用全球系統整合商 (GSI) 和全球服務供應商 (GSP) 來達成目標，但是公司應該有自己的人員參與這個過程。累積您自己的專業知識基礎，並且檢查並鞏固您與第三方的關係，因為第三方可能現在或未來有解決方案可以整合到管道中。

指派並組織專責團隊建構支援應用程式管道的平台，並訂定與持續測試相符的核心度量。這表示從擁有一些人工測試人員和少數工程師的組織發展，成為更有建構能力的組織。

持續測試之旅表示黑箱測試的數量將開始減少，而白箱測試將變得更加突出和相關，測試人員將需要能夠做到這一點。這需要將 CoE 從卓越中心轉變為能力中心，而且將進一步需要改變團隊和測試人員的構成。

大數據對於測試相當重要，尤其是在針對所有類型的測試進行程式碼覆蓋範圍的細化方面。關於迴歸套件的覆蓋範圍，市場呈現爆炸性成長，尤其對於發展科學方法來瞭解迴歸測試是否達到適當的程式碼覆蓋範圍。

公司必須做好變革的準備，並願意接受變革。一些關鍵的里程碑其中包括：

AI 扮演更重要的角色

- 自動進行自動化，立即自動產生測試將取代無法再維護的測試迴歸套件。
- 需要對使用者進行即時觀察以及後續測試，以驗證觀察結果。
- 記錄和重現已成為過去式。
- 未來也將持續加強關注改進需求，並發展能夠分析這些需求和察覺缺失的新方法。

執行領導團隊必須體認到周圍世界的變化速度。行動技術和雲端應用程式正在發展階段，速度是其生命線。一切都在朝著緊密的端點到端點客戶體驗發展。雲端、開發營運、敏捷、大數據，所有這些的終極目標是緊密的客戶體驗。您必須決定如何將這些應用於測試。這是兩個世界交會之處。

使用者/消費者必須認知企業所期望的價值。這是持續測試成為推動創新的奧妙之處。

物聯網透過各種方式連接裝置。目前，持續測試對於瀏覽器和 API 來說極具挑戰性，但是，對於實體和行動裝置以及物聯網而言，能夠進行持續測試的整合中心將成為變革的驅動因素。只要有所欠缺，任何人都會開始察覺。

在預測分析等方面，您必須始終提前三個步驟考慮。

頂尖的 QA 和 QE 人員將迅速發展。這些人員的技能水準會提高，持續測試所需的技能組合與目前相比將大為不同。公司必須先具備行動力並成為全通路。

最靈活的公司不僅透過採用持續測試文化贏得市佔率，而且有能力吸引最優秀的人才來加入公司。

總結

改採持續測試是一個旅程，就像過去的敏捷和開發營運一樣，未來也仍將如此。公司及其 IT 團隊將會遇到他們必須預測到並接受的挫折。這就是為什麼計劃在持續測試之旅的每一個步驟都不可或缺的原因。

持續測試既是一種文化，也是一種技術。藉由「失敗並向前邁進」方法，整個團隊可以從過去完全隔閡的獨立環境中記取教訓。所有偉大的發展和個人勝利都必然奠基在知識與經驗相結合的基礎上。

雖然持續測試有技術做為工具，但是終究以人為核心。一旦人員變得穩定且有信心，就會產生必要的能量、眼光和絕佳的作業能力。

透過這些成就，動能會持續產生，組織將因此而充分提升，在這個新的全球經濟中無往不利。

CS200-383435_0818