導論

你覺得五年後的自己是什麼樣子？這個經典的面試問題相當於「你長大後想做什麼？」——這個問題有一些符合社會期望的答案，而且時間跨度足夠長，讓你不需要全情投入。[1] 但如果你是一名資深軟體工程師，希望在職業生涯中持續成長，那麼這個問題就變得非常現實。[2] 你認為自己會前往何處？

兩條道路

你可能會發現自己站在一個岔路口（圖 P-1），你的前方有兩條不同的道路。在某一條路上，你會成為一名經理，需要管理下屬。而在另一條路上，你將成為一位沒有下屬的技術領袖，這個角色通常被稱為 *Staff* 工程師（*staff Engineer*）。如果你真的能看到這兩條路未來五年的發展，你會發現它們有很多共同點：它們通向許多相同的地方，當你走得越遠，你就越需要許多相同的技能。但在一開始，它們看似截然不同。

1 話雖如此，對於這個面試問題，你不應該回答「一位動物園管理員，同時也是一位太空人」。成人的生活是很有局限性的。

2 為了簡潔起見，我在本書中會一直提到「軟體工程師」；但是，如果你是一位系統工程師、資料科學家或任何其他技術從業者，我想你也會認為這本書令人心有戚戚焉。歡迎所有人閱讀！

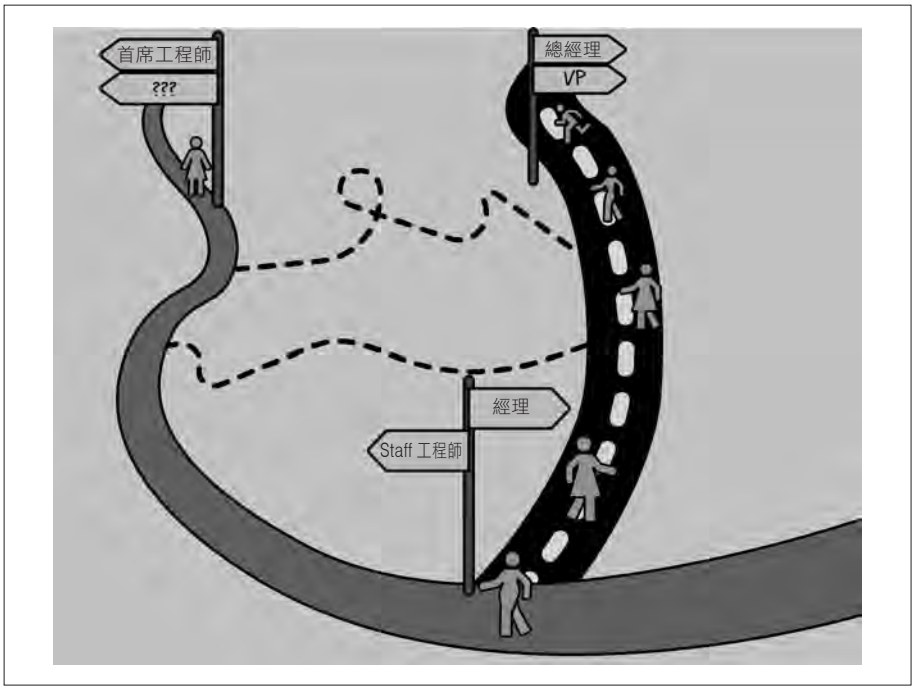

圖 P-1 岔路

經理人之路是一條清晰明確，許多人已經走過的路。對於任何能夠清晰溝通，在危機中保持冷靜，並幫助同事將工作做到更好的人來說，成為經理是一種常見的，也許還是一種預設的職業選擇。你很有可能認識一些選擇這條路的人。你可能曾有過經理或主管，對於他們做得對或做錯的地方，你可能也有自己的看法。管理學是一門被廣泛研究的學科。*升職*和*領導*這兩個詞通常被等同於是「成為某人的老闆」，而機場書店裡充滿了關於如何做好這項工作的建議。所以，如果你選擇踏上管理之路，這不會是一條容易的路，但你至少會對你的工作有大致的概念。

Staff 工程師之路就不那麼明確了。雖然現在許多公司允許工程師在不管理直接下屬的情況下持續提高工作資歷，但這種「技術職涯」仍然很混亂，而且路上缺乏明確的道路標誌。考慮走這條路的工程師可能從來沒有和 Staff 工程師一起共事過，或者在這個角色中看到的皆是非常狹隘的個性特質，以至於這個職位看起來像是高不可攀的魔法師。（其實不然，一切都是可以學習的。）不同公司對這項工作的期望各不相同，即

使在同一個公司裡，僱用或提拔 Staff 工程師的標準也可能相當模稜兩可，並不總是切實可行。

通常情況下，即使你接下這個角色，具體工作內容也不會變得更清晰明確。在過去的幾年裡，我曾與許多公司的 Staff 工程師聊天，他們都不太清楚組織對他們的期望是什麼，而工程經理也不知道如何與他們的 Staff 工程師下屬和同事一起共事。[3] 如果你的工作沒有被定義，要如何知道你做得好不好？或者你是否有達到基本要求？

即使定義了明確的期望，但實現這些期望的道路可能仍不明確。作為一個新上任的 Staff 工程師，你可能聽說過眾人期待你成為一個技術上的領導者，做出良好的商業決策，並在沒有實際權力的情況下產生影響力。但究竟要怎麼做呢？你該從哪裡開始？

Staff 工程師的特質

我懂這種感覺。投身這個行業的 20 年裡，我一直堅持走 Staff 工程師之路，現在我是一名資深首席工程師（senior pricipal engineer），在職涯階梯上的職級相當於我公司的資深經理（senior director）。雖然我曾多次考慮要不要走經理之路，但我始終認為「技術軌道」的工作真正地為我帶來活力，讓我早上會想起床去上班。我希望有時間鑽研新技術，深入瞭解架構，並學習新的技術領域。不論是什麼領域，你投入的時間越多，你的能力就能變得越強，而我一直想在技術方面不斷提升。[4]

不過，在我職業生涯的早期，我一直在努力理解這條路。身為一名中階工程師，我很納悶為什麼還有高於「資深」（senior）的級別 —— 那些人整天要做些什麼？當時的我當然看不出從我所在的位置通往這些職級角色的路徑。後來，成為一名新手 Staff 工程師之後，我發現了一些不

3 這個狀況正在發生改變。Will Larson（*https://staffeng.com*）、LeadDev（*https://leaddev.com/staffplus-new-york*），和其他一些人一直致力於鋪平道路，做出令人驚豔的工作成果。我將在本書中提供相關資源的超連結。

4 我保留以後改變主意的權利。

為人知的期望和缺失的技能，我不知道該如何描述這些東西，更不知道該如何行動。多年下來，我從許多專案和經驗中學習，不論是成功和失敗的經驗，並且從其他公司的優秀同事和同行那裡獲益良多。現在，這份工作對我來說已經可以理解，但我希望當時就能知道我現在體會到的東西。

如果你已經走上了 Staff 工程師的道路，或者正在考慮這樣做，歡迎你！這本書是為你準備的。如果你和 Staff 工程師一起共事，或者正在管理 Staff 工程師，並且想更加瞭解這個新興的職位角色，這本書也有很多內容適合你閱讀。在接下來的九章中，關於如何成為一名優秀的 Staff 工程師，我將與你分享我所學習到的經驗。現在，我要先警告你，我不會對每一個主題下硬性規定，也不會回答每一個問題：這個角色天生自帶大量的模糊性，而許多問題的答案往往是「這取決於事情的脈絡」。但我會告訴你如何駕馭這種模糊性，瞭解什麼是至關重要的，並與你合作的其他領導者保持協同一致。

我認為 Staff 工程師的角色由以下三大支柱構成：大局觀、專案執行，以及提升與你共事的工程師的水準。

大局觀

大局觀意味著能夠退後一步，以更廣闊的視野來看待問題。這意味著超越眼前的細節，理解你所處的環境脈絡。它還意味著超越當前時間的思考，無論是啟動為期一年的專案、建立易於退役的軟體，或是預測你的公司在三年內的需求。[5]

專案執行

到了 Staff 這個職級，你接手的專案會變得更加混亂、更加模糊。這些專案會涉及更多的人，需要用更多的政治資本、影響力或文化變革來取得成功。

---

5 在本書中，在談到雇主時，我將使用「公司」一詞，但當然你可以服務於非營利組織、政府機構、學術機構或其他類型的組織。歡迎將「公司」換成任何符合你情況的詞語。

提高標準

資歷越高，就越有責任提升和你處在相同技術軌道內的工程師的標準和技能，無論對象是你團隊裡的組員、組織內的同事，還是整個公司或行業的工程師。這種責任包括透過教學和指導的刻意影響力，以及成為榜樣而帶來的意外影響力。

我們可以把這三者看作是成就你的影響力的支柱，如圖 P-2 所示。

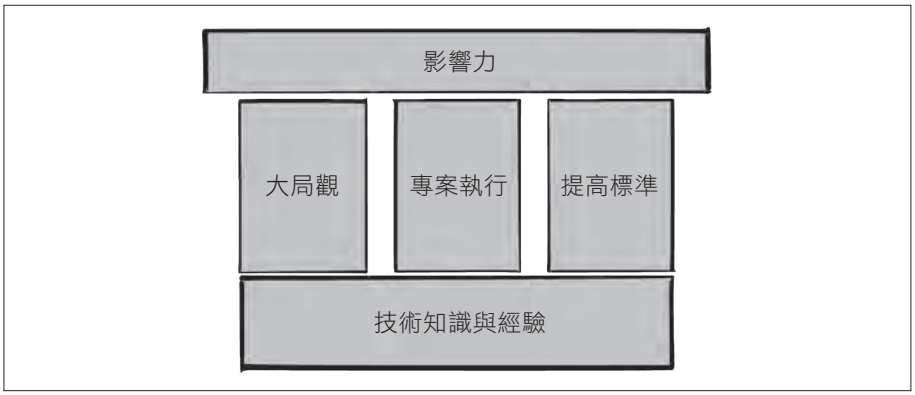

圖 P-2 Staff 工程師角色的三大支柱

你會發現，這些支柱建立在技術知識和經驗的堅實基礎之上，這個基礎至關重要。你的大局觀包括理解哪些事情是可能的，同時具備良好的判斷力。在執行專案時，你的解決方案需要能真正解決問題。當你作為一位榜樣時，你的審查意見應該使程式碼和設計變得更好，而你的意見需要經過深思熟慮 —— 你必須是正確的！技術能力是每個 Staff 工程師角色不可或缺的重要基礎，你必須不斷地鍛鍊技術。

但僅有技術知識仍遠遠不夠。在這個職級上的成功和成長，意謂你要做的事情不能僅僅依靠技術能力，你必須付出更多。如果你想成為善於為大局思考的人，執行更大的專案，並幫助你周圍的人提升水準，你需要「人性化」的技能，比如：

- 溝通與領導力
- 駕馭複雜局面
- 正確看待你的工作
- 輔導、贊助與授權
- 確定問題的框架，讓其他人接手處理
- 無論你覺得自己是不是領導者，都要如領袖般採取行動。[6]

你可以將這些技能想像成哥德式教堂上的飛簷（如圖 P-3）：它們不能取代真正的牆壁或是你的技術判斷，但它們讓建築師能夠打造更高聳、更宏偉、更令人敬畏的偉大建築。

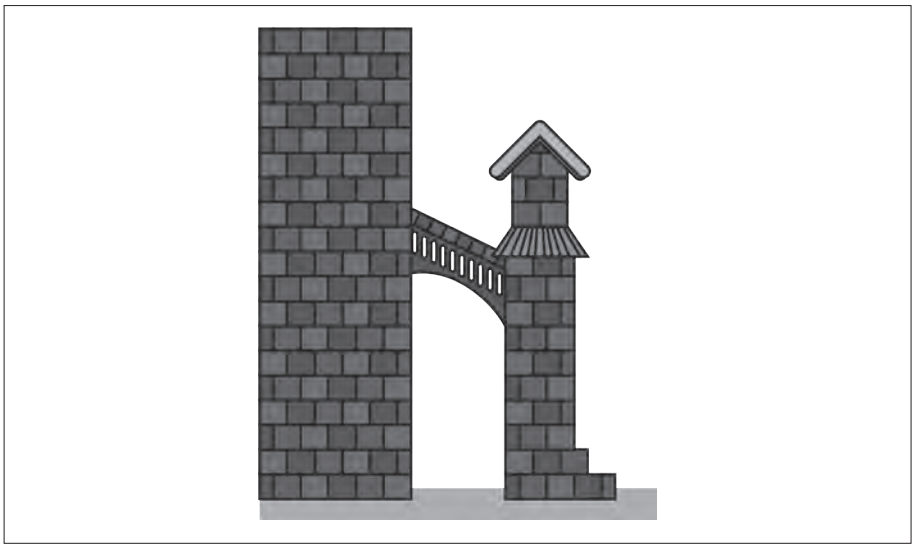

圖 P-3 領導力就像飛簷一樣，幫助我們為大型建築提供更好的支撐力。

三個支柱都各有一套必要的技能，而你應對每一個支柱的天份資質都有所不同。我們之中的一些人在領導和贊助大型專案時可能相當得心應

6 還有很多。請看 Camille Fournier 的文章：〈An Incomplete List of Skills Senior Engineers Need, Beyond Coding〉（*https://oreil.ly/gGe2T*）。

手，但在兩個策略方向之間做出抉擇時卻容易感到退卻。另一些人可能在理解公司和行業的發展方向上有很強的直覺，但在管理突發事件時卻容易失去對整體局面的控制。還有一些人的能力是提升與他們合作的每個人的技能，但卻很難針對某個技術決策促成共識。好消息是，這些所有的技能都是可以學習的，你可以成為三個支柱的專家。

本書分為三大部分。

第一部分：大局觀

在第一部分中，我們將探討如何以涉及全局的戰略眼光來看待你的工作。第 1 章向你提出關於你的角色的大問題。組織對你的期望是什麼？Staff 工程師是做什麼的？在第 2 章中，我們將眼光放遠，試著獲得一些觀點，我們將在環境脈絡中審視你的工作、導航你的組織，並揭曉你的真正目標。最後，在第 3 章，我們將會打造一個技術願景或策略，豐富我們的大局觀。

第二部分：專案執行

第二部分更偏向戰術實踐，我們將焦點轉向領導專案和解決問題的實際情況。第 4 章將探討如何選擇工作內容，我將分享如何決定要把時間投入在哪些地方的技巧、如何管理你的精力，以及如何妥善「花掉」你的信譽和社會資本，同時不減損它們的價值。在第 5 章中，我將討論如何領導跨團隊和組織的專案：為他們的成功做準備、做出正確的決定，並保持資訊順暢流通。第 6 章將討論如何駕馭你在前進道路上遇到的障礙、慶祝專案大功告成，以及當專案被喊停時該如何回顧檢討（儘管如此，還是要慶祝！）。

第三部分：提高標準

第三部分的主題是關於如何提升你的組織。第 7 章將會示範優秀工程師的行為、如何大聲學習以及如何建立心理安全的文化，以提高每個人

的水平。我們將研究如何在意外事件或技術分歧中成為「房間裡的成年人」。第 8 章是關於以更有目的性的方式來提升同事的技能，例如教學和輔導、設計審查、程式碼審查和促成文化變革。最後，第 9 章將探討如何提升**自己**：如何保持成長，以及如何為自己的職業生涯打算。在你目前的角色之後，你的下一站會去哪裡？我會在此討論一些選項。

在我們進入正題之前，有一個警告你必須知道：這是一本關於保持在「技術軌道」的**書**。它不是一本教你技術的書。正如我所說，你需要有堅實的技術基礎才能成為一名 Staff 工程師。這本書並不能幫助你獲得這些技術基礎。技術能力端看各個特定領域，如果你拿起了這本書，那麼我會假設你已經擁有 —— 或者正在學習你所需要的任何專業技能，以便成為你所在領域中最資深的工程師之一。無論「技術」對你來說是指程式碼、架構、UX 設計、資料模型、生產營運、漏洞分析，還是其他任何技術，幾乎每個領域都有大量書籍、網站和課程可以為你提供幫助。

如果你是一個技術能力至上的人，那麼你不太可能在這裡找到你想要的東西。弔詭的是，你也可能是能從這本書中得到最多收穫的人。無論你的技術知識有多深奧，你都會發現，當你能夠說服其他人採納你的想法，提升你周圍的工程師的能力水準，並且游刃有餘地透過使每個人都慢下來的組織僵局時，這些工作就不會那麼令人厭煩了。這些技能並不容易學習，但我保證它們都是人人都能夠學會的，我將在本書中盡我所能為你指點迷津。

你想成為一名 Staff 工程師嗎？不追求更高階的工程職位是可以的。你也可以中途改道去當經理（或者在兩者之間來回！），或留在資深工程師的位子上，做你喜歡的工作。但是，如果你想幫助你的組織實現目標，並且繼續培養技術實力，同時使你周圍的工程師在他們的工作上做得更好，那麼請你一定要繼續閱讀。

1

你如何描述你的工作？

Staff 工程師軌道（Staff engineer track），或者是「技術軌道」（technical track），對很多公司來說是一種很新的概念。各家公司組織對於最高職級的工程師應該具有什麼樣的屬性，以及這些工程師應該做什麼樣的工作，有著各自不同的觀點。儘管大多數人都同意，就如 Silvia Botros 所寫的那樣（*https://oreil.ly/xwgRn*），技術軌道的頂端絕不僅是「更資深的資深工程師」，然而我們對於 Staff 工程師這個職位本身的理解並不一致。因此，在這一章的開頭，我們將從探究這個角色的存在意義開始：為什麼組織會需要非常資深的工程師留在公司裡？有了基本認識後，我們將進一步解析這個角色：關於這個職位的技術要求、領導力要求，以及自主工作的意義。

Staff 工程師的角色有許多不同的形式，而且有許多有效且可行的方法來做好這份工作。不過，有些形式更適合於某些特定情境，而且不是所有的組織都需要各形各色的 Staff+ 工程師。因此，我想要先聊一聊該如何描述一個 Staff 工程師的角色：這個角色的職責範圍、深度、回報結構、主要焦點以及其他屬性。你可以利用這些角色描述來更加精確地瞭解你要如何執行工作，成長為什麼樣的角色，或者需要僱用什麼樣的人才。最後，由於不同的公司對 Staff+ 工程師應該負責的工作內容有各不相同的看法，我們會努力使你的理解與你所在組織中其他關鍵人物的理解相符。

讓我們先搞懂這份工作到底在做些什麼。

Staff 工程師究竟是何方神聖？

假如成為管理者是唯一的職涯發展路徑（就像圖 1-1 中左側所描述的公司），許多工程師將面臨一道嚴峻且困難的選擇題：究竟該留在工程崗位上，繼續打磨技術，還是轉換跑道到管理職，讓他們的職業生涯持續成長？

值得慶幸的是，許多公司現在提供「技術」或「個人貢獻者」軌道，為工程師提供更多樣的職業發展選項，而不只有管理職。圖 1-1 中右邊的職涯發展階梯就是一個例子。

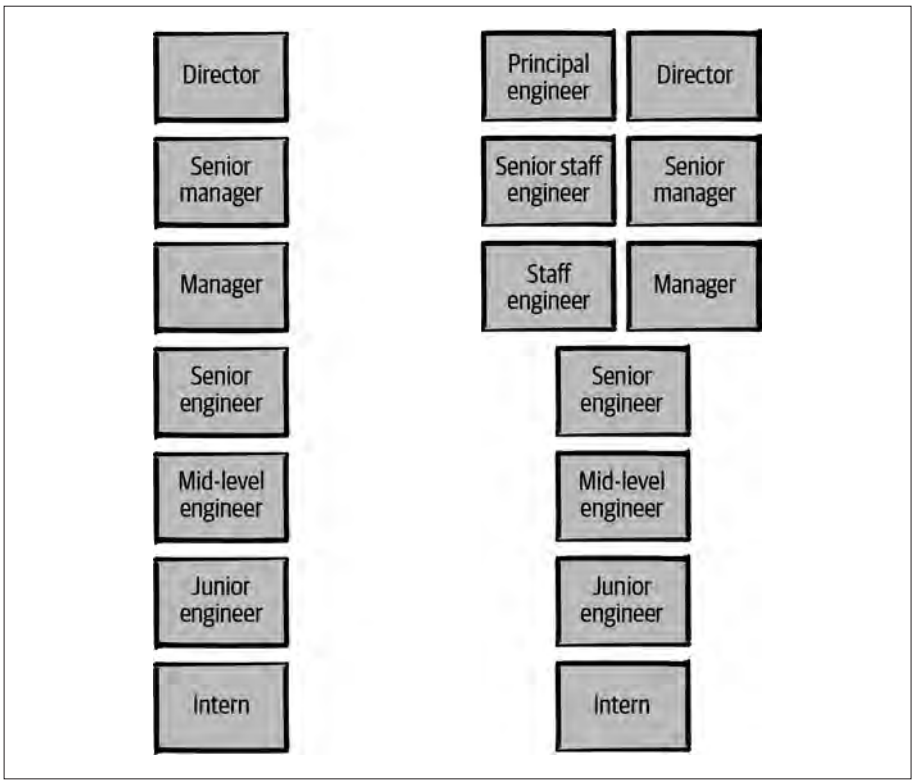

圖 1-1 兩種職涯階梯範例：右者提供多元發展選項

由於各家公司的職位層級存在不少差異，以至於有一個叫做 *level.fyi* 的網站，專門比對各個公司技術軌道的職等級別。[1] 在職涯階梯上，每家公司提供的職等數量可能有所不同，你甚至會發現不同公司採用不一樣的職級次序。[2] 很多時候，Senior（資深）一詞經常被使用。Marco Rogers，一位在兩家公司建立了職涯階梯的工程總監，他將 Senior 職等形容為職涯階梯的「錨（anchor）」（*https://oreil.ly/MpwsJ*）。他認為「Senior 以下的職級是為了讓人們增加自主權；Senior 以上的職級則是為了擴大影響力和責任。」

Senior 職等有時被看作是「終身（tenure）」級別：你不需要再往上走了。[3] 但如果你繼續向上升職，你將會進入「技術領袖」級別。Senior 職級的再上一級通常被稱為「Staff 工程師」，這也是我在本書中使用的名稱。

在圖 1-1 的雙軌制職涯階梯中，資深工程師可以選擇培養技能，爭取晉升為經理或 Staff 工程師的角色。一旦他們獲得升遷機會，日後從 Staff 工程師轉換跑道到管理職的角色，或是從經理回到技術職，將被視為一種水平的橫向移動，而不是更進一步的升職。Senior Staff Engineer（資深主任工程師）的職等與 Senior Manager（資深經理）相同，Principal Engineer（首席工程師）則相當於總監，以此類推；在一些公司的職業階梯中，可能還存在比這些職等更高的職稱。（為了代表所有高於 Senior 的角色，我將使用 *Staff+* 一詞，這是 Will Larson 在他的《*Staff Engineer*》一書中創造的表達方式。）

---

1 我還推薦造訪 progression.fyi，這個網站整理了各家科技公司公布的職級資訊。

2 我聽說有一家公司按工作資歷使用 Senior、Staff 和 Pricipal 這三個職等，但被另一家職等次序為 Senior、Principal 和 Staff 的公司收購。結果一團混亂。收購方公司將所有的 Staff 改為 Principal，將所有的 Principal 改為 Staff，然而沒有人為此感到高興。Staff 工程師和 Principal 工程師都認為這種變化是一種降等。頭銜真的很重要！

3 我喜歡我的朋友 Tiarnán de Burca 對資深工程師的定義：在這個級別上，某個人可以停止前進，並在餘下的職業生涯中繼續保持他們目前的生產力、能力和產出水準，如果他們離開組織，仍會被視為是「遺憾的人才流失」。

關於頭銜

我偶爾會聽到有人堅持工作頭銜和職等不應該被看重（或者根本不重要）。提出這種主張的人往往會試圖合理化，說他們的公司是一個秉持平等主義、任人唯賢的公司，對階級制度所隱含的危險保持警惕。他們這麼說：「我們是一個自下而上的文化，所有的想法都受到尊重。」這個目標確實令人敬佩：即便你身處職業生涯的早期，也不表示你的想法或點子就該被視而不見。

但是頭銜確實很重要。Medium 工程團隊寫了一篇部落格文章（*https://oreil.ly/oUkHe*），闡述了職稱存在之必要的三大原因：「幫助人們理解他們有所進展、將權限賦予那些可能不會自動得到的人、並向外界傳達（對這個職級）預期的能力水準。」

雖然第一個原因是屬於個人的內在原因，這也許不是驅動所有人的動機，但其他兩個原因則描述了頭銜對其他人的影響。無論一家公司是否宣稱自己採用扁平的、平等的組織結構，總會有一些人對不同職等的人做出不同的應對與態度，至少我們大多數人都會意識到地位高低差異。正如科羅拉多州立大學創業學系的實踐教授 Kipp Krukowski 博士在他 2017 年的論文〈*The Effects of Employee Job Titles on Respect Granted by Customers*（職稱對於來自客戶的尊重之影響）〉（*https://oreil.ly/zD3kp*）中所說：「職位頭銜就像一種符號，公司用這些符號向公司內外的人們表明員工的品質水準」。

我們無時無刻不在對人做出隱性判斷和假設。除非我們投入大量的時間和精力來察覺自己的隱性偏見，否則這些假設極有可能受到刻板印象的影響。舉例來說，2015 年的一項調查（*https://oreil.ly/snmmY*）發現，在接受調查的 557 名從事 STEM（科學、科技、工程與數學等領域）的非裔與拉丁裔職業女性中，約有一半人曾被誤認為是清潔工或行政人員。

當一個軟體工程師參加一場陌生會議，類似的隱性偏見就會發揮作用。白人和亞裔男性軟體工程師往往被認為是更資深、更「懂技術」和更擅長寫程式碼的人，不管他們是昨天剛畢業還是已經做了幾十年的工作。相較之下，女性，尤其是有色人種的女性，

則被下意識地認為資歷較淺、較不夠格。她們必須在會議上更努力地工作，才能被人認為真的有能力。

正如那篇 Medium 文章所說，職稱頭銜將權限賦予給那些可能不會自動得到的人，並傳達了（組織）預期的能力水準。透過錨定期望值，爭取到某個頭銜能夠節省時間和精力，否則人們將不得不一次又一次地證明他們自己。這讓他們能在一週內重新獲得幾個小時的時間。

你現在擁有的頭銜也會影響到你接下來的工作。像我們行業的許多人一樣，我每天都會收到 LinkedIn 上招募人員傳來的訊息。在我的人生中，我只收到過三次冷郵件（cold email），邀請我去面試一個比我現有的職位更資深的職位。所有其他的郵件都是建議我去做一個和我現在的級別完全相同的職位，或者一個更低級的職位。

所以，這就是工作職稱在職涯發展階梯上的樣子。不過，讓我們來看看**為什麼**還會出現「技術領袖」這類級別。我在導論中談到了技術軌道的三大支柱：大局觀、專案執行和提升水準。為什麼我們需要**工程師**具備這些技能？究竟為什麼需要 Staff 工程師？

為什麼工程師要有大局觀？

所有的工程組織都在不斷做出決策：選定技術、決定要打造什麼東西、投入於一個系統或是淘汰它。其中一些決策有著明確的所有者和可預測的後果。其他的決策則是奠定一切基礎的架構選擇，會影響到其他所有的系統，沒有人能夠信誓旦旦地說清楚它們將會如何發展。

良好的決策需要**脈絡**。有經驗的工程師知道，大多數技術選擇的答案是「這要看情況」。了解某個特定技術的優缺點還不夠 —— 你還必須知道使用情境的細節。你想做出什麼東西？你有多少時間、金錢和耐心？你的風險承受能力是什麼？業務需要什麼？這些就是決策的脈絡。

蒐集脈絡資訊需要時間和心力。個別團隊傾向為己方爭取最佳利益；單個團隊中的工程師可能會專注於實現團隊目標。往往那些看似屬於一個團隊的決策，其影響卻遠遠超出了這個團隊的範圍。**區域極大值**（*local maximum*），也就是單個群體的最佳決策，如果從更廣泛的角度來看，也許並不會是最佳決策。

圖 1-2 展示了一個例子：一個團隊要在 A 和 B 兩個軟體中做出選擇。這兩個軟體都擁有必要的功能，但 A 軟體明顯更容易設置，它就是能發揮作用。B 軟體則更難一些：它需要幾個衝刺週期的啟動時間，而沒有人願意等那麼久。

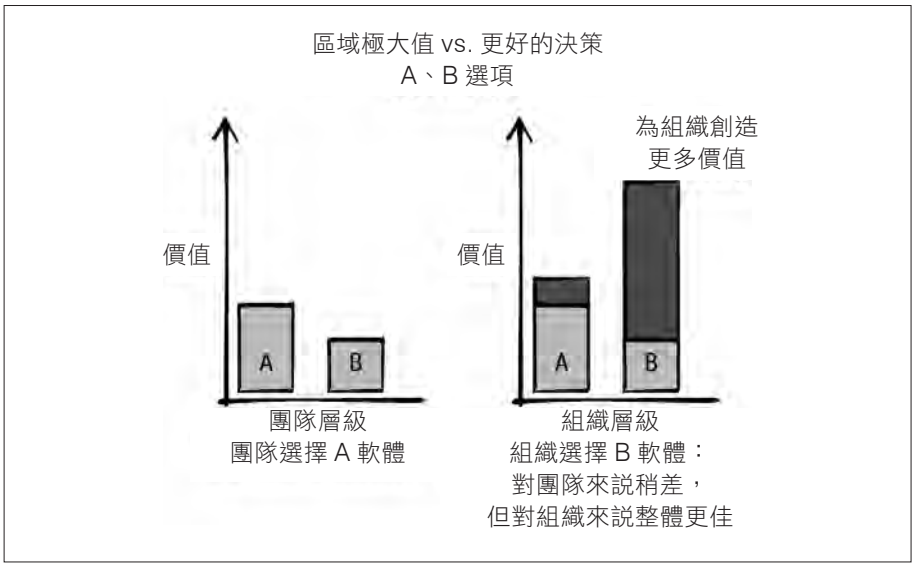

圖 1-2 區域極大值 vs. 更好的決策

從這個團隊的角度來看，選擇 A 軟體是個明智的決定。何必要選擇其他東西呢？然而其他團隊寧願他們選擇 B 軟體。因為事實證明，A 軟體會給法律和安全團隊帶來持續性工作，它的認證需求意味著 IT 和平台團隊將不得不永遠把它作為一個特殊案例。選擇 A 軟體，即該團隊的區域極大值，此團隊在不知不覺中選擇了一個對公司整體來說得投入更多時間的解決方案。B 軟體對團隊來說只是稍微差一點，但整體上要好得

多。額外的兩個衝刺週期將在一個工作季度內得到回報，但這一事實只有當團隊中有人以更具大局觀的視角來觀察時才得以顯現。

為了避免產生區域極大值，團隊需要決策者（或至少是決策影響者）從**局外人的角度**看見更大格局 —— 同時考量多個團隊的目標，並選擇一條對整個組織或整個企業最有利的道路。第 2 章會介紹如何將視野縮放並且看清大局。

與看到**現在的**大局同等重要的是，預測你的決策在未來會如何發展。一年後，你會對什麼事情感到後悔？三年後，你會希望你現在就做哪些事情？為了朝著同一個方向前進，各個小組需要在技術策略上達成一致，比如要投資哪些技術、對哪些平台進行標準化等等。這些巨大決定的最終結果可能難以捉摸，而且往往是有爭議性的，所以做出決策的關鍵要點在於能夠分享脈絡資訊，並幫助其他人確實理解。第 3 章的主題是作為一個團體如何選定方向。

因此，如果你想做出廣泛的、具有前瞻性的決定，你需要能夠看見大局的人。但是，難道這個人不能是經理嗎？難道不能由首席技術長（CTO）來搞懂所有的「業務事項」，將其轉化為技術目標，並將這些重要目標告訴下面的人嗎？

在一些團隊中，他們確實可以。對於一個小團隊來說，經理往往可以擔任最有經驗的技術專家，負責重大決策和技術指導。在一個小公司裡，首席技術長可以深入參與每個決定的細節。這些公司可能不需要 Staff 工程師。但是，管理權限可能會掩蓋技術判斷：即使有更好的解決方案，下屬可能會覺得與經理爭論技術決測是不合適的。而管理本身就是一項需要全身心投入的全職工作。一個致力於成為優秀人事經理的人，很難分配較多時間去瞭解技術發展的最新情況，而任何身處複雜工作情境的管理者，會較難滿足其下屬需求。短期來說，這可能還過得去：一些團隊不需要很多的關心就能繼續在邁向成功的道路上前進。但是，當團隊的需求和技術策略的需求之間出現矛盾時，經理就必須選擇要把重點放在哪裡。要麼是團隊成員，要麼是技術方向會被忽視。

這也是許多組織為技術領袖和人事領袖開闢各自獨立的職涯路徑的原因之一。如果公司裡有不少工程師，如果每一個決定都需要在首席技術長或資深經理的辦公桌上敲定，那麼工作效率的低落無庸置疑，更不用說這同時削弱了人們的工作自主性。假如讓有經驗的工程師有時間深入研究並梳理脈絡，並且賦予權限，讓他們設定正確的技術方向，你會得到更好的結果和設計。

這並不表示工程師得獨自設定技術方向。經理，作為負責為技術計畫分配人力的人，也需要參與重大的技術決策。我將在本章後面談到如何保持工程師和經理之間的一致性，並在第 3 章談及策略時再次探討。

那「架構師」呢？

在一些公司，「架構師」是職涯階梯中技術軌道上的一個職等。在其他公司，架構師是抽象的系統設計師，他們有自己的職涯路徑，與實作系統的工程師不同。在本書中，我打算將軟體設計和架構視為 Staff 工程師角色的一部分，但請注意，這在我們的行業中並非舉世皆然的普遍事實。

為什麼工程師需要領導跨團隊專案？

在一個理想的世界裡，一個組織中的各個團隊應該像拼圖一樣環環相扣，工作範圍涵蓋任何正在進行的專案的方方面面。然而，在這個理想的世界裡，每個人都在為一個嶄新的綠地專案工作，沒有任何約束或遺留系統需要解決，而且每個團隊都能全身心致力於這個專案。團隊的界限很明確，沒有爭議。事實上，我們一開始就採用了 Thoughtworks 技術顧問所稱的「逆康威模式（Inverse Conway Maneuver）」（*https://oreil.ly/HdKyK*）：一組團隊完全對應了所需架構的所有組成部分。這個烏托邦專案的困難部分之所以困難，只因為它們涉及了深入的、迷人的

研究和發明，而且它們的所有者對於這些問題的技術挑戰躍躍欲試，殷切渴望著輝煌的職業榮耀。

我想為這個專案工作，你難道不想嗎？很遺憾，現實世界與此有些不同。幾乎可以肯定的是，參與任何跨團隊專案的團隊在這個新專案醞釀之前就已經存在了，並且團隊正在忙於其他的事情，甚至可能是他們認為更重要的事情。他們會在專案的中途發現未曾預期的依賴關係。他們的團隊邊界有過度的重疊和差距，並且蔓延到整個架構之中。而專案中陰暗和困難的部分並不是迷人的演算法研究問題：它們涉及到對遺留程式碼的研究，與那些不想改變任何東西的繁忙團隊交涉談判，以及猜測多年前離開的工程師們的當時意圖[4]。如果你仔細看一下設計文件，你可能會發現它推遲了那些最需要調整的關鍵決定，或者只有一筆帶過。

這才是一個更符合現實的專案情形。無論再怎麼小心翼翼地將團隊分攤到一個巨大的專案上，有些責任最終還是不屬於任何人，而另一些責任則被兩個團隊所共同擁有。資訊不能流動，或者在溝通交流中被擾亂，最後造成衝突。每個團隊都做出了優秀而符合區域極大值的決定，結果導致軟體專案陷入困境。

保持專案進展的一個方法是讓某人對整個專案擁有全部的所有權，而不是這個專案的任何個別部分。甚至在專案啟動之前，這個人就可以清楚界定專案範圍，並建立一個提案。當專案開始後，他們很可能是大方向系統設計的作者或共同作者，也是專案的主要聯絡人。他們秉持高工程標準，以經驗為據預測風險，並提出切中要點的困難問題。他們還會花時間非正式地指導或輔導，或是為專案的各個部分的負責人樹立一個好的榜樣。當專案陷入困境時，他們有足夠的視角來追蹤原因並進行疏通（第 6 章將著墨更多）。在專案之外，他們要描述正在發生的事情和原因，向公司其他部門推銷願景，並解釋這項工作將會實現願景，以及新專案如何影響到每個人。

---

4 他們在想什麼？這真的是他們想做的嗎？當然，未來的團隊也會對我們提出同樣的問題。

技術專案經理（technical program managers，TPM）難道不能做好這種建立共識和溝通的工作嗎？確實，在 TPM 的職責上與此有所重疊。不過，總的來說，TPM 負責的是交付，而不是設計，也不是工程品質。TPM 負責確保專案按時完成，而 Staff 工程師則負責確保專案以高工程標準完成。Staff 工程師負責確保開發出來的系統是強大的，並且與公司的技術環境相輔相成。他們謹慎看待技術債，對任何會成為這些系統未來維護者的陷阱的東西都保持警惕。TPM 通常不太會撰寫技術設計或為測試或程式碼審查制定專案標準，也沒有人期待他們要對一個遺留系統的種種細節進行深入研究後，再來決定哪些團隊需要與之整合。當 Staff 工程師和 TPM 在一個大專案中愉快合作，他們可以成為一個夢幻團隊。

為什麼工程師應該發揮良好影響力？

軟體很重要。我們建立的軟體系統可以影響人們的幸福感和收入。維基百科的軟體錯誤清單（*https://oreil.ly/eNIXO*）非常值得一讀，甚至帶有警世意味。我們已經從飛機失事（*https://oreil.ly/iJgF2*）、救護車系統故障（*https://oreil.ly/s9GQf*）和醫療設備故障（*https://oreil.ly/fr7Dj*）中了解軟體 bug 或是軟體故障足以致命，假如你認為未來不會再出現更多、更嚴重的軟體相關悲劇，那可就太天真了。[5] 我們必須認真看待軟體。

即使風險較低，我們仍然在製造軟體，這是有原因的。除了一些研發方面的例外，工程組織的存在通常並不只是為了創造更多的技術。他們是為了解決一個實際的商業問題，或者創造一些人們願意使用的東西。他們希望以某種可接受的品質、對資源的有效利用和最少的混亂來實現這一目標。

---

5 Hillel Wayne 的文章〈*We Are Not Special*〉（*https://oreil.ly/WK0TK*）指出，很多過去必須仔細校正調整物理設備的工程解決方案，現在都用「勉強拼湊起來的軟體」來代替。到目前為止，我們很少有來自軟體的重大致命事故，這始終令我感到很驚訝。我不希望只靠純粹的運氣。

當然，品質、效率和秩序遠遠無法保證，特別是涉及到趕死線的時候。當「做得正確」意味著速度變慢時，急於交付的團隊可能會跳過測試、偷工減料，或對程式碼進行走過場式的橡皮圖章審查。而且，打造優秀的軟體並不容易也不直觀。團隊需要的是那些已經打磨好技能的資深人士，經驗告訴他們什麼是成功的、什麼是失敗的，而且他們會承擔起創造可用軟體的責任。

我們從每個專案中學習，但每個人只能反思有限的經驗。這意味著，我們需要從**彼此的**錯誤和成功中學習。經驗不足的團隊成員可能從未見過好的軟體如何被製作出來，或者可能認為生產程式碼是軟體工程中唯一重要的技能。經驗豐富的工程師可以透過指導程式碼和設計審查，提供架構的最佳實踐，以及打造各種工具，使大家的工作更快、更安全，因而發揮巨大的影響力。

Staff 工程師是人們的榜樣。經理負責在他們的團隊中建立文化，強制執行好的行為，並確保標準得到滿足。相對地，工程規範是由專案中最受尊敬的工程師的行為而設定的。不管專案標準洋洋灑灑寫了些什麼，假如最資深的工程師不寫測試，那麼你永遠無法說服其他人去寫。這些規範超越了技術影響，它們也隱含了文化意味。當資深的人大聲讚美別人的付出，互相尊重，並提出問題來釐清狀況時，其他人也會更容易效而仿之。當菜鳥或新手工程師把某個人當作他們想「成為」的那種工程師來尊重時，這是一種強大的激勵因素，讓人想要起而效法。（第 7 章將探討如何成為榜樣來提升你的組織水準。）

也許現在你已經被說服，工程師應該具有大局觀、專案執行、發揮良好影響力的能力，但問題是：在資深工程師的程式碼工作量之上，還要做好這些事情是不太可能的。你在寫技術策略、審查專案設計或制定標準的任何時候，你都不是在寫程式、架構新的系統，或做很多用來評估軟體工程師績效的工作。如果一個公司最資深的工程師整天都在寫程式，那麼他們的優異技能可以為程式庫帶來好處，但公司將與只有他們才能勝任的工作失之交臂。這種技術領導力必須放上做好這份工作的職責描述中。這不是對工作的干擾：**它就是工作本身**。