推薦序

我想成為一名軟體架構師的渴望，源自於我對軟體設計技術面上的興趣。我非常喜歡討論如何能善用科技來解決問題，以及如何建造出高度模組化、架構完善且易於使用的碼庫（codebase）。

雖然如此，但沒有人會告訴你這些技術面向只是整個架構拼圖中的一部分而已。它並不只是技術與設計軟體，它還涉及了如何在一個特定的組織情境下，設計軟體並解決問題，以及瞭解週遭所發生的事，而能在必要的時刻，成功地看到問題並發揮影響力。這是關鍵所在。因此，架構師要意識到，他們必須在其直屬團隊環境的內外部，在不同的層級中，與扮演各種角色的人員溝通，進而產生影響。

然而，從企業的角度來看，在訓練軟體開發人員轉型成軟體架構師角色的工作方面，我們所做的相對較貧乏，更不用說為軟體架構師提供什麼協助了。特別是在非技術面向上，尤其明顯。很快地瞄一下書店書架上所擺的書就知道，大部分都是有關軟體架構、架構風格、架構模式、DevOps、自動化、企業架構、精實、敏捷等等的書。你只能找到相對少量之關於人與溝通的書，能涵蓋上述這些主題的書，更是少之又少。

軟體架構師全方面提升指南從更廣泛的角度來探討架構師的角色，填補了這個空缺。它將教你如何避免陷入傳統的、有點功能失調的「業務對 IT」心態、如何去找到能運用並影響到組織視野的關鍵點、如何作有效的決策、如何與供應商打交道，以及如何跟組織上下層級中的所有同仁溝通。對一個想要成功扮演架構師角色的人來說，這些都是基本的要求。

參考延伸閱讀可看到更多本書中介紹之實務技巧與技術的補充，但壞消息是，許多相關故事聽來，你會覺得這些事情再熟悉不過了！雖然 Gregor 的故事與設有一個傳統 IT 部門之中大型公司裡的許多工作者更為相關，但其中大部分的過程，也適用於站在新浪潮上的「數位公司」。看到這些狀況同樣出現在這類的組織裡，也令我感到訝異！

整體而言，這是一本適合架構師或有志成為架構師之人閱讀的一本絕妙的書，勝過其他討論相同主題的書。透過本書，你能快速地盤點自己架構工具箱中現有的工具組。我非常推薦本書給想要有所作為的架構師或 CTO 之類的人來閱讀。無論你是正在尋找擴展自身技能的方法、想要瞭解架構到底是什麼，或是已經被賦予提高組織生產力與成效任務的人，都可以在本書中找到能提升自身能力的方法。

— *Simon Brown*,
Software Architecture for Developers 作者

推薦序

我還記得第一次被指派要在一個 IT 部門中組織一個架構團隊的事。當時我還不知它意味著什麼，但覺得這件事聽來就很酷，有信心能把它搞清楚。不過，這個自信只維持了大概 5 分鐘，因為有個團隊成員問我，我們要扮演的是技術架構師還是企業架構師，我才知道自己根本不瞭解二者間的差別！

二十年後，我有幸成為一個全球性組織的首席架構師，雖然還沒能完美地掌握到架構師的工作精髓，但我瞭解到，面對這種模糊不清的狀況而還能泰然處之，就是一位好架構師最重要的特質之一—會問一些尷尬的問題，像我的團隊成員那樣，則是另一個重要特質！

本書透過生動描繪在目前資訊技術革命進程下之架構師生活與使命的方式，幫助你瞭解成為一名架構師會是什麼樣的一種情況。團隊跟我的時間都花在搭乘架構升降梯：從組織的這邊跑到另一邊、對接、解釋、質疑，並在資訊不完整的情況下，試著做出好的決策。這部升降梯成天載著我們在從程式碼到業務策略的樓層間，來來回回地上下移動著。

架構在企業科技領域裡已經被斷斷續續地關注了一段時間，架構師有時也常會被指責成「沒啥建樹」的人。我相信架構師體現出二件非常重要且求之未必可得的事：他們行之有理，他們明智決策。不論架構師是幫其組織瞭解一個愈來愈難掌握的領域，或者是找出其必須要做的決策並設法以合理的方式，在正確的時間點做出這些決策，這都是架構師應盡的責任。此外，如同本書所說明的，若你沒辦法做出有意義的決策（第 6 章），讓決策明確，並協助同仁理解這些決策，你就不能算是在做架構。

當然，這並不是輕易就能駕馭的技巧。人類瞭解複雜事物的能力有限，在資訊有限的情況下，很難做出好的決策。架構師可以透過採用合適的技術與多年經驗累積所成的思考方式，來協助自己與其所屬的公司。他們可以把學習曲線變成是斜坡而不是斷崖，讓理念更能被理解，透過市場的語言來做出更好的決策（第 18 章），同時也將一些好的作法推廣給企業（第 9 章）。

架構無法一直持續受到關注的其中一個原因是，組織需要架構師做的事已經變了。在我的職涯中有一些時刻，組織認為我應該去定義他們的現況與未來，並找出連結二者的通道。這是一種可被理解的信念：想要瞭解我們目前所處的狀況，我們要何去何從，以及我們怎麼到那裡去。這似乎很合理。但這種信念也是基於一種靜態的世界觀而來的，在這種觀念下，所有的變化都由穩定的狀態所衍生。

在當今世界中，運行任何組織的技術必須是動態的，而且組織也必須能夠去調整技術以順應經濟的速度（第 35 章）。現今架構師的任務是為組織的速度與活力創造條件：同時滿足速度調整與提升服務品質的設計目標（以及協助同仁瞭解這些目標並沒有衝突；第 40 章）。若你仍舊認為你的任務就是用數年的規劃來定義未來的架構，則可以先看看本書的第五部。

用架構升降梯這種形象來說明是貼切的，因為它是一種貫穿組織中心的一種持續性行動。升降梯也是一種轉型技術：這個發明造就了摩天大樓，永遠地改變了我們天際線的景觀。如果你想成為一名架構師，那麼你應該要能習於這種經常會變動與轉型的生活。如果你充滿好奇心，常常追尋問題的答案，渴望與人交流，也想參與決策，架構師也許是你想扮演的角色。你還是不會知道架構師需要做的所有事情，但本書可以協助你去把這些問題的答案找出來。

—*David Knott* 博士，
HSBC 首席架構師

關於本書

當數位經濟改變了傳統企業的遊戲規則時，架構師的角色也有了根本上的變化。架構師不再只專注在技術實作面，他們必須連結組織中設定商業策略的高層，與能實現科技的技術部門。唯有連結起這兩個部分，IT 的角色才能從成本中心轉變成數位競爭力的優勢。從組織裡的這個樓層走到另一個樓層是沒辦法搭起這種連結的。現代的架構師會採取快速路徑來跳過現有結構：**架構師升降梯（Architect Elevator）**。

本書能協助（啟發）架構師擁抱對架構師有意義的新觀點，並讓他們能搭著架構師升降梯，在許多樓層之間穿梭，讓組織與技術的發展齊頭並進，推動持續性的變革。

首席架構師的日子：並不是孤獨地待在上頭

我們對 IT 領導者與首席架構師有許多期待：他們必須在 IT 仍被視為是成本中心的組織中活動，其在這類的組織中，所做的事代表跟「改變（change）」相反的「執行（run）」，而中階管理者就成了既不用瞭解商業策略也不用清楚底層技術，只要照章辦事就行的人。一直以來，架構師被期待著要能掌握最新科技、管理供應商、將漂亮的行

話，轉換成有用的策略，然後還要能招募頂尖的人才。然後，這並不令人意外，資深的軟體與 IT 架構師就成了世界上最搶手的 IT 專業人才之一。

然而，期望如此之高，要如何才能成為一位成功的首席架構師呢？而在成為架構師之後，你要怎麼跟上環境的變化呢？成為首席 IT 架構師之後，我期待的並不是找到任何神奇的答案，反而是在找一本書，能讓我不用一直重複現有解決方法的書。

我參加過許多 CIO/CTO 的活動，雖然有用，但這些活動主要都聚焦在高階發展方向上，而不是關心如何在某一技術層次上，實際地完成一項任務。在沒辦法找到一本合適參考書的情況下，我決定將自己超過 20 年從事軟體工程師、顧問、新創公司共同創辦人與首席架構師工作的經驗集結起來，寫成一本我自己的書。

我會學到什麼？

本書依照支援大型 IT 轉型的架構師職涯，將內容安排成幾個主要部分。這個職涯的發展過程，將從靠近 IT 引擎室的地方開始，然後慢慢往組織的層峰靠過去：

第一部，架構師

在企業情境下，瞭解一位架構師所應具備的素質。

第二部，架構

將架構的價值主張重新定義為變革驅動器。

第三部，溝通

向各利害關係人有效傳達技術課題。

第四部，組織

運用架構心態去瞭解組織結構與系統。

第五部，轉型

在組織中持續推動變革。

第六部，結語：架構 *IT* 轉型

扮演變革代言人的角色。

邀請你按照從技術性到組織性主題的順序，從頭到尾把本書看完。當然，你也可以從最感興趣的章節開始，細讀本書。運用所附的延伸參考資料來交叉對照，照你的順序來閱讀。畢竟，網際網路就是這樣，我想，也許也能用這樣的方式來閱讀本書。

這不是一本技術性的書。這是一本討論如何提高架構師水平，使其能在大型組織中，有效運用自身技術能力的書。本書不會教你如何架設 Hadoop 叢集，或運用 Docker 與 Kubernetes 來做好容器編管（container orchestration）工作。相對地，本書會教你如何解析大型架構；如何確保你的架構有利於商業策略；如何運用供應商的專業知識；以及如何在關鍵的決策上，與上級管理層溝通。

它確實有用嗎？

如果你是在找科學證據，能讓一個科技組織轉型的可重現「方法」，你可能要失望了（不過，若你找到證據，請讓我知道）。本書的結構並不那麼制式，你要的可能只是一些能讓你成功的建議，但卻不得不被迫讀一些看似無關緊要的閒人軼事，你可會有點惱火。不過，這些軼事就是架構師在日常工作上會碰到的。你不能照其他人所做出的決策依樣畫葫蘆，但卻可以從他們的經驗中學習，而做出更好的決策。

本書以我在 IT 界 20 多年的工作經驗為藍本。我擔任過新創公司的共同創辦人（很有趣，但錢不多）、系統整合師（提高稅務稽核效率）、顧問（弄了很多 PowerPoint）、作者（蒐集並記錄許多洞察）、網際網路工程師（創建未來）、一家大型跨國企業的首席架構師（棘手，但報酬豐厚），以及 CTO 顧問（洞察跟分享）。我認為 IT 轉型是帶有個人色彩的，因為架構在本質上某種程度帶有個人的色彩。望著一座知名的建築物，你大老遠就可以認出設計它的建築師來。看來像白色盒子：Richard Meier；看來都是彎彎扭扭的曲線：Frank Gehry；看來像編織交錯出來的線條：Zaha Hadid。雖然沒有這麼戲劇化，但每一位（首席）IT 架構師著重的重點與風格，都會展現在其作品之中。

本書所集結的這些洞察，反映了我的觀點，用「雞塊」式的寫法可便於理解，也可被更加廣泛地運用。側欄裡所寫的則是在傳統與數位公司中所習得的經驗。

架構師是很忙碌的一群人。因此，我試著將我的洞察包裝成容易吸收的故事，希望它們讀起來會有趣一些。我期盼你在這條路上，能體會到一種混合了「不是只有我遇到這種問題」跟「那是一種看待事物之新角度」的感覺。

關於架構與轉型還有許多未能擺進本書裡頭的東西，因此你可以在參考資料裡，看到我建議閱讀的其他相關書籍與文章，它們都有助於你繼續更深入地探究任何特定的主題。

講個故事給我聽

我選擇用一些故事來架構出本書，因為我們身處於一個複雜的世界中，講故事是一種很棒的教學方式。有研究指出，比起純粹的事實列舉來，人們更容易記住故事。也有證據顯示，聽故事能觸發大腦其他的部分，幫助我們理解與記憶。亞里斯多德早就知道一場好的演講，其成功的因素不只包括**邏輯**（***logos***），即事實與結構，還有**品格**

（**ethos**），令人信賴的特質，以及**情感**（**pathos**），情緒，而這些都能被一個好的故事所引發。

要讓組織轉型，你並不需要去解數學方程式。你需要讓人改變，這就是你為什麼要能講出一個好故事，畫出一幅具有說明力之願景圖的原因。你可以從運用本書中容易引人注意的標語（「殭屍要吃你的腦了！」）開始，然後再接著講你自己的故事。你曾看過人們在看電影時，即使他們知道故事是虛構的，這些表演都是假的，但卻也哭了又笑了嗎？這就是講故事所發揮出來的力量。

編排慣例

本書呈現許多強調傳統與數位公司之對比的真實故事。底下的圖示代表著相對應之角色的故事：

「經理」圖示代表的是描繪傳統 IT 組織之思考與工作方式的例子。

「數位原生代」圖示代表的是描繪現代「數位」組織運作方式的例子。

這個圖示代表一般的註記或評論。

這個圖示代表警示或提醒。

第一部

架構師

在 IT 企業中，架構師的工作既令人感到興奮也具有相當的挑戰性。不少經理人與技術人員認為他們只是一群領著過高的薪水，成天活在自己象牙塔裡頭的人。他們脫離現實，只知道用投影片與一面牆那麼大的海報，把自己的想法強加在其他人身上。而且，他們還經常會追求一些八竿子打不著的理想，延宕了專案的時程。

往好處看，因為傳統企業正尋求 IT 轉型以與數位顛覆者競爭，IT 架構師已成為某些最吃香的 IT 專業人士之一。但諷刺的是，不少擁有世界級軟體與系統架構的知名軟體公司，卻連個架構師都沒有。

所以，除了印在名片上的頭銜之外，是什麼可讓一個人成為一名架構師？

架構師不是什麼

有時候，說明某件事物不是什麼，比給它一個明確的定義，要來得容易。就架構師而言，過份誇張的期待，甚至可能把架構師描繪成左手能解決效能問題，右手能轉變整個企業文化的人物。這樣的期待就會誤將架構師當成是下列幾種角色，而明顯地偏離了架構師的職責：

資深開發者

開發者的下一個職涯發展目標（與薪資等級），通常就被設定成是成為架構師。不過，成為一名架構師與一名明星工程師是二條不同的職涯道路，二者並沒有哪一條路比較好的問題。架構師必須涉獵的層面比較廣，涵蓋組織性與策略性的面向，工程師則傾向專注於可行軟體的交付上。成熟的 IT 組織應該要瞭解這種情形，提供平行的職涯發展道路，讓工作人員適性選擇。

救火隊

因為架構師對現行系統有著廣泛地瞭解，許多經理人都希望架構師能處理問題並應付各種突發狀況。架構師不應該忽略產品問題，因為這些問題就是寶貴的回饋，反映出了可能的架構缺陷。若一位架構師要忙著做一堆滅火工作的話，就沒有時間去思考架構，如此架構就無法推行。

專案經理

架構師必須能兼顧許多不同但相關的事務。他們的判斷需要考慮（並會影響）到專案時程、人員配置與其所需的技能。因此，較高階的管理人經常會依賴架構師提供專案資訊，特別是專案經理人忙於填寫狀態報告範本的時候（第 30 章）。對架構師而言這是一件不容易處理的事，雖然它是一件有價值的工作，但卻不是架構師的主要任務。

科學家

架構師需要有敏銳的智力，也必須能以模型與系統的方式思考（第 10 章），但他們所下的決定卻會衝擊到實際的業務專案。因此，許多組織會將首席架構師與首席科學家的角色拆分開來。個人傾向用首席工程師來強調架構師的產出不是只有論文。最後，雖然科學家可將事情弄得看起來很複雜且不容易瞭解，讓自己的論文得以發表，但架構師的工作恰好相反：要讓複雜的事變得容易瞭解（第 8 章）。

架構師的許多樣貌

架構師在不同抽象層次的環境中工作。就像現實生活中的建築物會有城市規劃師與建築、景觀與內部架構師那樣，IT 架構師會有許多特化後的樣貌：你會看到網路架構師、安全架構師、軟體架構師、解決方案架構師、企業架構師與更多其他不同類型的架構師。像在真實世界中那樣，各種架構師都同樣重要。舉例來說，住在規劃不良，到處都是交通阻塞，公共設施欠缺的都市裡，雖房子的架構完善，但這跟住在運行順暢城市中架構不良的房子裡一樣，令人感到沮喪。IT 界也是如此——你的優美設計與完善模組化的應用，若搞錯重點或只是重複現有應用所做的事，有做跟沒做是一樣的。同樣地，若應用無法與企業網路結合，沒有多少使用者會覺得好。因此，哪一類架構師比較重要並不是重點；重要的是要能讓各類的架構師一起合作，一起把事做好。

架構師處理非需求

通常我們會認為開發者要處理功能性的需求，而架構師要處理的是非功能性的需求，通常是那些常被稱為“某某性”的：可擴展性（scalability）、可維護性（maintainability）、可取得性（availability）、可交互運作性（interoperability）等等。但實際的情況並不這麼單純。我更常看到架構師要處理非需求（*nonrequirements*）。這個詞並不是指那些不需要做的事，而是那些不是到處都會講的需求。包括情境（context）、隱性假設（tacit assumptions）、潛藏的相依性（hidden dependencies）以及其他從沒被講出來的東西。把這些隱性的需求找出來，並明確地將它們呈現出來，這是架構師最有價值的貢獻之一。再次強調，這些工作可能由企業架構師到軟體架構師的任何層次中開展——這就是那個重要的連結。

衡量一名架構師的價值

架構師的價值並不是那麼容易可以說清楚的。通常我會這麼向別人說，若一套 IT 系統經過許多年之後仍能承受得住高的變動率，則這個專案團隊裡頭一定有名好的架構師。現今，要用幾年才來衡量一名架構師的價值，似乎有點不切實際。其實，我們可以從幾個面向（dimensions）來觀察架構師所產生的價值：

架構師會“把點連起來”

通常，一套 IT 架構中的每一個獨立元素，都是經過深思熟慮而出且能運作良好的，不過所有這些完善系統總體的產出，還是無法滿足業務所需。架構師會檢視這些盒子，確保它們之間的相互依賴性是清楚的。

架構師懂得權衡利弊得失

系統設計與發展涉及到許多決策。大部分有意義的決策並不會只有好的影響，也會有不好的影響。架構師會權衡利弊得失，依據目標與原則取得平衡。

架構師注重的不只是產品

有太多的 IT 決策是為產品選擇（product selection，第 16 章）所驅動的。架構師不只要看著產品名稱與功能列表，還要考慮許多事情，以萃取出決策的選項並權衡其輕重。

架構師要提出策略

IT 的目標是要支援商業策略。架構師透過將商業需求轉化成技術的驅動力來搭建出二者的連結。

架構師與複雜度奮戰

IT 是複雜的。架構師會平順地減少複雜度。比方說，架構師會以架構審查委員會的形式進行治理與開創（第 32 章）。其中也包含了讓系統“退休（retiring）”（電影銀翼殺手中的用語），除非你想在殭屍之間掙扎著過活（第 12 章）。

架構師能交付

對架構師而言，腳踏實地面對現實，並從真實的專案實作中取得回饋，是很重要的。否則一切在掌控之中就還是錯覺。（第 27 章）。

由此可見，在漂亮的架構圖背後，架構師還是做了很多事情！

架構師如同變革的代言人

現今，成功的架構師已不僅僅是 IT 專家，他們也扮演著重要的變革代言人。因此，架構師必須擁有一套特殊技能而不僅是科技。

本書這部分的章節會讓你為這個角色做好準備，這些內容會教你如何去：

第 1 章，架構師升降梯

搭著架構師升降梯，跨越組織的各個層面。

第 2 章，明星架構師

像電影明星那樣，扮演具不同人格特質的角色。

第 3 章，活在一階導數中的架構師

活在一階導數中。

第 4 章，企業架構師或企業中的架構師？

連結業務與 IT。

第 5 章，三腳架構師

不只擁有技術，那只是架構師的一隻腳。

第 6 章，作決策

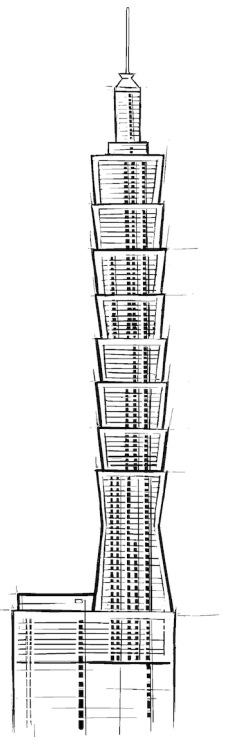
在面對不確性時，練就好的決策紀律。

第 7 章，質疑每件事

質疑每件事，直搗問題核心。

架構師升降梯

往返於頂樓與引擎室之間



搭升降梯才能上高樓

架構師扮演著連結與轉譯的關鍵角色，特別是在一個各部門講著不同語言、有不同的觀點，甚至是往不同目標努力的大型組織中，架構師尤為重要。許多管理層級只會透過電話往上或向下溝通，無疑加劇了

公司層級間的隔閡[1]。最糟的情況是掌握資訊或專業的人沒辦法被賦予做決策的權限，而有權做決策的人卻無法掌握相關資訊。這對企業的 IT 部門而言，並不是好事，特別是科技已成為大多數企業重要驅動力之一的現今。

架構師升降梯

架構師能填補大型企業裡的這個重要間隙：他們密切地與負責專案的技術人員合作與溝通，也能將技術議題傳達給上層的管理圈，而不漏失訊息的本質（第 2 章）。從相反的方向來看也一樣，他們瞭解公司的業務策略，也能將之轉化成技術決策並提供有力的支持。

若將組織的層級想像成是建物中的樓層，架構師就能運用我稱之為架構師升降梯的無形結構：他們搭著升降梯上下，在大型企業的決策辦公室與實際打造軟體的引擎室間來回穿梭。在各層級間這樣的直接連結，在快速 IT 演化與數位衝擊（digital disruption）的時代中，更是比以往任何時候都要來得重要。

把這個類比延伸一下，放到大型船艦的情境下來看，若艦橋的指揮官發現有障礙物，油輪需要轉向迴避的話，他們會讓引擎逆轉，把船舵轉向右舷。但若當時的引擎正以最高速前進，則將發生一場大災難。這也就是為什麼即便是一艘老式的蒸氣船，也會有一條用來複頌船長命令之通訊管線的原因。在大型企業中，架構師扮演的就是這樣的角色！

某些機構的樓層特別多

回到建物的隱喻上，架構師要搭升降梯過去的樓層數量取決於組織的類型。扁平型組織也許完全不需要升降梯——爬段樓梯就夠了。這也意味著在這類組織中，架構師的這種連結上下樓層的角色，其重要性

1 在這種電話遊戲中，小朋友們圍成一圈，一個接著一個地傳遞訊息。當訊息傳回到原發訊人時，他會發現，訊息幾乎在過程中完全走了樣。

也許小一些：若管理是能敏銳地在細節的需求層次上覺查到技術上的現實，而技術人員也能直接參與高層管理工作的話，則少一點「企業」架構師是有必要的。我們可以說，這種數位公司是開在平房裡頭的，因此不需要升降梯。

在升降梯隱喻中的架構師價值，不應該由他們能上到多「高」來衡量，而應該由他們能涉足於多少樓層而定。

不過，駐紮在大型組織中的典型 IT 團隊，似乎有許多樓層蓋在他們的頭上。他們在摩天大樓裡頭工作。樓是如此之高，以致於單一個架構師升降梯可能無法涉及於所有樓層的事務中。在這種情況之下，若一位技術架構師與一位企業架構師互相搭配，並各自負責自己在「那一半」建物中的責任區，還是可行的。在這種情境下，架構師的價值不應該由他們能上到多「高」來衡量，而應該由他們能涉足於多少樓層而定。在大型的組織中，位居頂樓中的那些人可能會犯只看到並評價上半建物中之架構師的錯誤。相對地，許多開發者或技術架構師會將這類「企業」架構師視為是比較沒有用處的，因為他們不寫程式。某些情況下，這可能是真的——這類架構師通常喜歡待在建物上層，他們不太會再熱衷於搭升降梯下來。不過，一位願意下到建物下半層來與技術架構師們分享策略看法的「企業」架構師，就能產生顯著的價值。

不是一條單行道

你總是會遇到一些搭升降梯但一到了樓頂就不會再下來的人。他們太享受頂樓上的好景緻，從而覺得他們不用再努力工作，到髒髒的引擎室來。通常，你只要聽到像「我以前是搞技術」這樣的一句話，就能認出他們。而我也會忍不住要這樣子反駁「我以前是個經理」（這是真的），或「你為什麼不再做了？現在不擅長搞技術了？」若你要更老練些（或哲學一點），可引用 Fritz Lang 拍的電影大都會（*Metropolis*）中的情節，在人們瞭解到「頭跟手需要協調」之前，頂樓與機房間的隔閣，幾乎讓整個城市完全被摧毀。無論如何，升降梯是用來搭著上下跑的。地下室被水淹時，待在頂樓裡吃魚子醬並無法轉變企業 IT。

搭乘升降梯在組織裡上下跑，對架構師而言，也是一種重要的機制。如此可以取得對決策的各種看法，也能瞭解其在實行階段的各種可能結果。時間長的專案實踐循環無法產生一種好的學習循環（*learning loop*，第 36 章），可能還會導致一種「架構師的夢，開發者的夢魘」狀況。在其中，架構師能達成其抽象的理想，但其實作則不切實際。只讓架構師享受往上看的樂趣，總是會變成可怕的只講權威不講責任的反模式[2]。這個模式唯有在架構師必須去面對，或至少觀察到，其決策之結果時，才能打破。要能這樣做，他們必須持續地搭著升降梯上下跑才行。

高速升降梯

在過去，IT 的決策離商業策略很遠：IT 非常「普通（vanilla）」，且其主要的參數，或重要效能指標（key performance indicator，KPI）卻不便宜。因此，跟新資訊比起來，搭不搭這升降梯，並不那麼重要。但現今，即使就「傳統」企業而言，企業目標與選用技術間的連結已更為直接。舉例而言，想把上市時間（time-to-market）弄得再快些的這種期望，其所形成的競爭壓力已轉化成對彈性雲計算的需求，也就是說，應用需要能橫向擴展，所以需要將之設計成無狀態（stateless）的形式。客戶管道上的目標內容，需要分析性的模型，這需要透過 Hadoop 叢集（cluster），去攪動大量的數據才能調整，而這反而又傾向於運用本地端硬碟式的存儲，超過共享網路式的存儲。用一、二句話來說，企業所需要的，已轉化成應用或基建（infrastructure）的設計，而這就強調了架構師得要去搭這升降梯。而且，他們愈來愈需要搭高速的升降梯，以跟上企業與 IT 交織融合的步伐。

在傳統的 IT 公司裡頭，建物中的低樓層可能擠滿了外部顧問（第三十八章），這可讓企業架構師不用動手去處理事情。不過，這樣一來，因為只著重在效率上，而忽略了速度經濟（economies of speed，

2 “Authority Without Responsibility,” Wikiwikiweb, 2004, *https://oreil.ly/WhXg-*.

第 35 章）。在技術快速演化的時代中，這樣子的配置，成效必定不佳。以往習於待這類環境中的架構師，必須要擴展其角色，由供應商技術的純消費者，轉化成能主動定義技術的角色。要這樣做，他們需要發展自己的 *IT 世界觀*（第 16 章）

乘客群像

若你正如一位成功的架構師，搭著升降梯上上下下，你也許會在升降梯裡遇到一些人。也許你會，比方說，遇到一些業務或非技術的人，這些人已更深地瞭解到 IT 就業務而言是很重要的。善待這些人，把他們帶上，帶他們到處看看。

加入他們的對話——這能讓你更瞭解業務需求目標。也許，他們甚至能帶你到從未去過的更高樓層。

你也許也會遇上從頂樓搭電梯下來，只想要用一些行話來推銷自己想法的人。我們不管這些人叫架構師。只搭電梯但不出去的人，通常被稱為*梯弟*（*lift boys*）。他們從頂樓的忽視中獲益，所追求的是一種不用與實際技術接觸的「技術」職涯。透過讓他們對引擎室所發生的事感興趣，你也許能夠改變其中的一些人。若你沒成功，則大家最好就維持電梯裡的那種心照不宣的沈默，看看每一塊天花板磚的細節，以避免眼神的接觸。把你的「升降梯高論（elevator pitch）」留給與資深高管共乘一部升降梯時發表，而不用說給僅是傳訊息的人聽。

搭升降梯的危險

你可能會認為雇主會很感謝架構師搭著升降梯上上下下。畢竟他們為著企業的 IT 轉型，使其在數位世界中更有競爭力而提供了顯著的價值。令人訝異地，這樣的架構師也會遭遇到阻礙。實際上，頂樓與引擎室間可能已相當習慣於斷開彼此的連結：公司領導者會覺得數位化轉型進行得很好，而引擎室的人卻享受著在沒有許多監督下，嘗試新技術的自由。頂樓與引擎室這樣地斷開連結，就像一艘巡航艦，全速往冰山撞去：等到公司領導人瞭解到底發生了什麼事時，為時已晚。

我曾經被引擎室人員批評，說我強推公司的流程，違背了開發者意向。同一時間，也被公司高層責難，說我只為了樂趣而嘗試新的解決方案。諷刺的是，這似乎代表著我找到了一個好的平衡點。

某些人可能會喜歡這種像比薩斜塔（Leaning Tower of Pisa）的組織，其地基與頂樓並不在同一條垂直線上。在這種建物中搭乘電梯，勢必更富挑戰性。在踏入這樣的環境時，升降梯架構師必須作好面對二邊的抗拒。沒人說作一位推動變革者很容易，特別是當系統抗拒改變的時候（第 10 章）。

在這些情況下，最好的策略是開始小心地連結各個樓層，等待好的時機以分享資訊。比方說，你可以開始向管理層傳達在引擎室工作的那些人，做得多麼好。當你能取得更詳細的技術資訊時，可以讓他們有更高的可見度與識別度。

其他見到你搭升降梯覺得很不是滋味的居民，可在中間樓層找到：看你呼嘯而過，連接層峰與機房，讓他們覺得自己被略過。因此組織中對你的工作，會產生一種「沙漏」型的欣賞：管理層峰將你視為能引領轉型的關鍵，而在引擎室的人也樂意有一位能夠實際瞭解與欣賞他們工作的人。但在中間樓層的人，則將你視為是其生計的威脅，包括其子女的教育費與其在山裡頭的渡假小屋。這是件微妙的事。有些人甚至會主動地擋你的道：要停在每個樓層一一作出解釋——也就是調控（第 30 章）——讓搭升降梯沒辦法比走樓梯快。

最後，因為搭升降梯的人少，擅長做一件事通常會讓其他人說你不會做其他事。比方說，能為管理層做有意義且具啟發性報告的架構師，通常不會被認為是一位優秀的技術師，即使這就是他們的報告有意義的原因。所以，每隔一段時間，你要讓上層的人知道，你也是能待在引擎室的。

弄平建物

與其不辭辛勞地乘著升降梯上上下下，為何不把那些不需要的樓層打掉？畢竟，你任職的數位公司正試著以少更多樓層在競爭著。不幸地，你不能直接把某些樓層直接打掉。而且，把這些樓層打掉，你留下的只會是殘磚破瓦，而不是一棟較低的建築。中間樓層的人通常是組織與 IT 視野中關鍵知識的持有者，特別是，若其中還有大型的黑市（第二十九章）存在的話。因此，短期內，組織沒辦法在沒有他們的情況下運作。

一點一點慢慢將建物打平看來可以是一種長期策略，但這會耗太多時間，因為這需要在公司的文化上，作出根本的變革。這也會改變或消除待在中層樓層之人所扮演的角色，他們會激烈地抵抗。這並不是一場架構師能打贏的戰爭。不過，一位架構師可以開始把事情弄得不那麼緊張一些；比方說，設法讓頂樓的人對引擎室傳來的資訊感興趣，或者提供更快的回饋循環。