# 未來的系統架構

## 將系統架構的重點放在針對現有服務資源進行重新設定以強化企業能力，而不是單純的重新建置整個系統。

文／John McDowall 譯／潘得龍

不論我們願不願意承認，資訊系統架構方法是持續演化的。從過去幾十年以來，系統架構的建置方法，就有如在蓋建築物一樣：首先決定需要什麼樣的系統，然後確認需要那些重要的子系統以及這些子系統互動的方式。接著，持續以這樣的方式拆解，直到所規劃的系統足以讓開發團隊做出這些個別的子系統，然後將這些子系統整合以建立出理想中的系統。不過這樣的方式最近幾年來已經有些改變，而且變化的比率逐漸增加。

### 網路的影響

隨著網路系統從1990年代普及之後，系統架構開始有了改變。主從式架構的出現成為當時的主要設計類型，網路能力成為架構中必要的元素。系統不再是放在一個諾大房間中的一組巨大機器，然後使用終端機操作。取而代之的，是許多能夠散佈於任意地點的一般個人電腦。

未來受到網路影響系統架構改變，則是系統的整合。我們很容易就發現，讓操作者在不同的系統中輸入相同的資料，是非常浪費時間，同時也很容易出錯的。所以我們開始嘗試做系統整合的事情，讓資料能夠共享。這樣的趨勢，就產生了「以服務為導向」(SOA)的架構。SOA的基本觀念，是讓提供服務的供應者，在網路上以可用服務的方式提供資源，網路上的任何應用程式都可以直接提出要求而獲得服務。所謂的「服務」就是針對想要提供的能力定義出完整的介面。SOA讓系統走入動態程式碼的時代，出現能夠根據新的業務需求自動調整，而不需要修改程式碼的應用程式。

### 服務的普遍應用

跟許多吸引眾人目光的新科技一樣，SOA一開始並沒有能夠表現出符合大家期待的能力。也與許多其它熱門科技一樣，當經過十多年之後，熱潮退燒，SOA的真正效益才慢慢地開展出來。如我們現在所見的，許多的功能都以服務的方式呈現，並且也普遍被許多企業採用。例如，Facebook和Goolge所提供的驗證服務。如果你開啟了一個網站，然後想要在先驗證使用者允許使用網站上的所有功能，你並不需要自己建置驗證子系統－只要使用這些這類公司所提供的服務。相同的情況，評論留言區、社群媒體整合、使用者統計，以及其它許多的功能也都能以服務的方式提供。整個雲端運算的革命，事實上是一種將運算硬體轉移到隨選服務。

雖然目前的面貌不是當初設想的方式，SOA革命確實是明確發生的。大部分的企業目前在整合上的努力，讓系統介面能夠公開使用。這通常是指「應用程式介面」(API)的基本哲學理念。API基本哲學的最有名的事件，或許是「the Steve Yegge rant」這段文字中的描述，其中作者責怪Google沒有採用Amazon以API為優先設計的哲學。這篇怒氣衝衝的言論，基本上是強烈期待所有的系統功能都應該藉由

API的方式在網路上顯示出來，一方面可以有助於，另外一方面則是能夠減少企業功能重複開發（以及付費）的數量。

## API如何影響系統的架構

到目前為止，以API為優先作法的主要效果，已經讓開發人員習慣於他們所設計API規格製作文件並公開化。不過對於Amazon以API為優先作法的主要信念，是建立於能夠減少因為在多個系統重複開發相同功能的程式，所帶來的成本降低。因為大部分的企業都要好幾年才會更新一次系統，但隨著時間的推移，這些影響將會讓人感覺到，特別是當以API為優先作法與建置前重複使用的作法結合，會需要系統開發人員重複使用在企業中已經存在的功能然還才開發新的。

當越來越多的系統讓他們的功能可以經由API分享出來，而開發團隊也被要求必須在開發之前確認有沒有可重複使用的子系統，這將會出現一個現象，就是原本想要建立的新系統，會被以現有功能整合的新功能的方式重新設計。由於各種不同的目的，在整個系統之間重複的數量是令人驚訝的。大部分的系統需要一種儲存與讀取資料的方法，又大部分的系統需要驗證與使用者授權的方法，同樣的大部分系統需要有能力顯示文字與繪製圖形。這些列出來的功能或許都能夠從企業已經存在的資源中，不斷地被重複使用。對於早期的系統開發人員，開發人員會需要自行設計屬於自己系統中的上列功能，成為系統中的一個迷你功能系統。但隨著這些基本的功能都可以採用服務的方式提供，系統設計者的工作就從必須設計整個系統，演化成在系統生態系統中設計增加邊際能力。

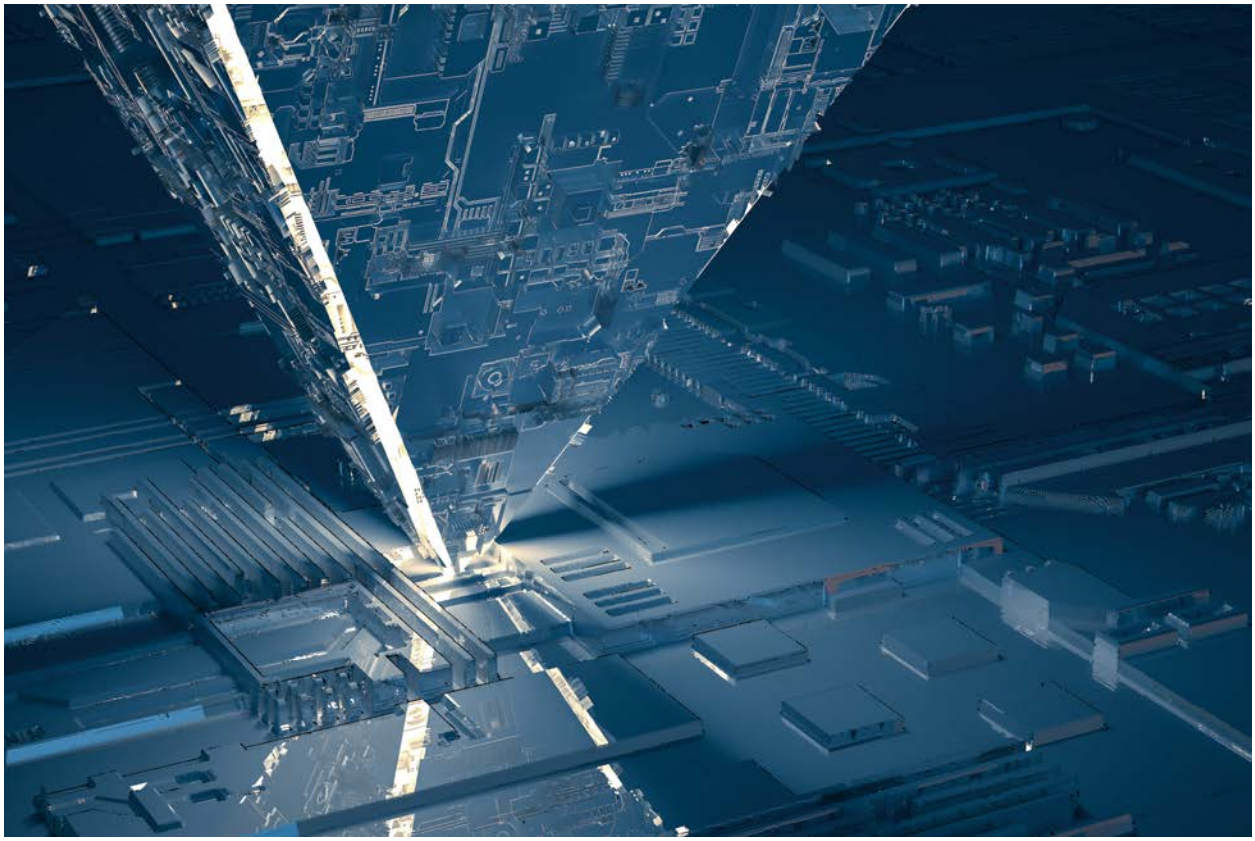
## 趨向以功能為中心的架構

我們目前都必須面對企業生態系統，所需要各類的功能快速擴張，並且都逐漸以服務的方式設計，尤其在雲端的環境更是如此。雲端供應商之間也以提供更多的功能作為與對手之間競爭籌碼，並且也已經可能藉由整合新的服務與一些現有軟體或是描述性與語言，而開發出一些基本的系統。如此一來，開發人員就只需要幾個星期的時間建立出許多不用自己設計功能系統，這在以往可能需要好幾個月的時間。雲端供應商所提供的基本系統，可以透過加入所需要使用的新服務而快速提昇功能，如果還沒有服務可以使用，則可以安裝額外的模組補充。想要在實際建置之前，就嘗試找出系統的所有細節是沒有意義的，我們需要新的方式思考系統架構與設計。

大部分的情況下，在一個企業中，已經存在有許多服務可以使用，建立新的系統應該要先清楚定義出所需的系統需要那使用些功能，然後與目前已經存在的所有服務功能清單進行比對。這樣的作法可以顯示出在企業內部已經存在多少可用的理想系統，並且也清楚有多少功能是需要另外開發的。現有可以使用的功能與需要再定義能力的功能，這兩者之間的差異，正是企業目前的能力與想要達到能力之間的差異。當我們走向未來，系統架構的主要任務就會從設計整個系統轉移成確定目前能力差異，然後設計縮小差異的最佳方法。

**當我們走向未來，系統架構的主要任務就會從設計整個系統轉移成確定目前能力差異，然後設計縮小差異的最佳方法。**

我們目前還不是很清楚，這種以功能為中心的架構是否會比較容易進行。我們能夠掌握整個企業中所有可用的服務，事實上是有很嚴厲的限制。任何誠實的網路管理員，都會承認他們無法掌握在他們所管理的網路上，所有可用的完整服務清單。他們可能知道有什麼機器連接到網路上，這些機器運行什麼軟體，以及這些機器開啟那些通訊埠與通訊協定。不過這些資訊只能告訴我們這類內容網路層級的觀點，而無法披露這類項目的實際使用方式。例如，在網路上使用著8443通訊埠，並且接受HTTP連線，有可能表示只是單純提供簡單的網頁資料內容，但也可能實際上



藉由該介面提供許多REST服務。

當然目前有方法可以顯示出這類內容，不過大部分都是採用手動的方式。例如，維護一份企業中所有可以使用的服務清單百科，不過這就必須要求開發人員在每次建置新的或是異動服務的時候，就需要同步更新這份清單。當然採用自動化的方式，表示能夠以即時識別並且建立服務介面的目錄，這樣是更有效率，不過這已經是另外一個主題了。

## 例外情況

某些地方還是需要使用傳統的系統架構處理，因為這些環境需要這樣的系統才能運作。當在營運環境中將功能公開於整個企業會產生問題的時候，就需要採用傳統的方式，也就是整體設計新的系統。例如，航空飛行控制系統就真的無法以呼叫由地面所提供的服務或是相關的功能，因為這會影響飛行安全。類似的情況，衛星系統以及其它類型的內嵌型軟體，都需要在系統本身提供重要的功能，而無法以呼叫外部服務取代。

傳統的系統架構並不會整個消失，不過那的確已經是舊的方式，而我們現在應該要開始思考如何實際上改善系統架構的效率，以便更能支援目前企業快速演化的情況。