低碼程式與公民開發者

文／葉宏謨

當組織中的成員有很多是公民開發者的時候，組織自然會數位轉型，變成資料驅動型組織。

人人都應該會寫程式

現在，不會寫字的人稱為文盲；未來，不會寫程式的人就是文盲。

低碼（Low-Code）是一種快速設計與開發應用程式的方法，公民開發者（Citizen Developer）是開發應用程式給自己或同事使用的企業員工。

企業有需要寫那麼多程式嗎？現在的人每天寫 Line，有需要寫那麼多 Line 嗎？

因為隨時會有一些想法，想跟他人分享，就有了寫 Line 的動機。

並非所有分析角度在 ERP 系統中都有事先寫好的程式可用，使用者等資訊部門寫好程式可能已經錯過時機，因此每個人都必須有能力寫程式從 ERP 系統取得資料、分析資料並送上 Line 或其他通訊工具，來和同仁們分享、溝通，才能取得共識。在 ERP 系統中寫程式是填空題不是作文題，所以非資訊部門員工也能寫程式，這些會寫程式的非資訊部門員工就是「公民開發者」（Citizen Developer）。

企業的組織架構越來越扁平化，作業面的工作越來越自動化，需要人處理的主要是管理面和決策面的活動。企業中與管理、決策相關的同仁越來越像伙伴或合夥人（Partner）關係。同仁們會想從企業資訊系統中找到資料並與他人分享，因此會經常使用 ERP 系統

（或其他企業資訊系統）查詢資料，甚至必須自己寫程式製作 ERP 系統沒有的報表。

ERP 系統是企業正式列管的資訊系統，由資訊專業人員負責，有一定的資安及開發、維護標準程序。公民開發者寫的程式是為了滿足自己在工作上「一時」的需求，不會影響到 ERP 系統，也不需遵循標準程序，只要能快速產生自己需要的資料就可以。就好像籃球迷經常觀賞 NBA 球賽，但偶爾也會自己下場打打籃球一樣。

客製化的老問題

有相當多的學術或實務論文探討為何 ERP 系統導入專案的失敗率那麼高，結論一致認為最主要的原因是 — 客製化，所以都強調必須做到「最小客製化」（Minimal Customization），導入 ERP 系統才會成功。

為何客製化會造成系統導入的失敗？因為溝通問題。使用者提出需求後，系統分析師（SA 或 ERP 顧問）要去了解或挖掘（elicit）使用者「真正的」需求，使用者往往說不清楚真正的需求，或者說了以後隔幾天又變了。系統分析師和使用者談過之後，又要和系統設計師（SD）溝通使用者需求，系統設計師才能開出規格，接著程式設計師才能開始寫程式。從使用者到 SA 需求被打了八折，從 SA 到 SD 又打八折，從 SD 到程式設計師又是八折，等到交付程式時，使用者說這不是他要的。因為花了太長的時間，使用者一直看不到 ERP 系統導入的成效，再而衰、三而竭，終於累了、放棄了。

解決客製化最直接的方法是「客製 DIY」，就是讓使用者自己客製程式。使用者自己寫程式不會有溝通的問題，不會打折，速度又快。只要會閱讀資料目錄（Data Catalog，例如 SOA-ERP 的服務文件），會複製貼上，就能寫程式給自己用。利用 SOA-ERP 的如同積木般的服務元件，所謂客製化就只剩改變介面（UI）和流程（BP），就像組積木那樣，做出自己要的東西，不需要去改變積木。寫程式是填空題不是作文題，客製一個應用程式需要的工夫，現成的服務元件（包含資料庫和商業邏輯）就佔了 90% 以上，UI 和 BP 只佔不到 10%，而且是「填空題」，所以客製是低碼、是有可能 DIY 的。

資料目錄

問題是企業應用系統那麼龐大，想要取得所需資料如同大海撈針，這時就需要資料目錄（Data Catalog）。資料目錄在 SOA-ERP 稱為服務文件（Service Doc）。開發者可利用關鍵字搜尋服務文件，找到所需的服務元件。

另一個尋找資料服務元件的方法是執行 SOA-ERP 的標準介面，NEO，在目錄樹搜尋關鍵字找到相關的功能及服務元件。SOA-ERP 只有服務元件沒有 UI，NEO 是對應到 SOA-ERP 服務元件的 UI，服務元件有多少欄位，NEO 就有多少欄位。SOA-ERP 系統為了滿足不同行業、不同規模企業的需求，欄位很多、流程很細，而使用者只關心自己的需求，沒必要承受不必要的複雜度。故若使用者要直接使用 NEO，則應利用它的無碼（No-Code）客製工具加以簡化，只留下必要的 UI 欄位和 BP 步驟，越簡單越好。使用者也可以使用任何前端工具，針對自己的需求客製最簡單的 UI 和 BP，呼叫 SOA-ERP 服務元件。

SOA-ERP 資料目錄就像一本很厚的書，雖然有上述方法讓使用者找到需要的服務元件，但最快的方法還是打個電話或 Line 直接詢問專家，也就是資訊單位同仁或外部顧問，馬上得到答案。而且，想做一件事可用的服務元件可能不只一個，專家們會告訴你應

該使用什麼服務元件最有效。

SOA-ERP 有各種程式範例模板。從資料目錄找到所需服務元件之後，複製該服務元件的關鍵字，貼入範例模板中即可完成程式，這種作法稱為低碼（Low-Code）程式設計，也就是僅需寫（填）很少的程式碼，使用者只要接受數小時訓練就能開始自己寫程式。

雖然有資料目錄和低碼程式設計工具，很難、也沒必要要求每個人都自己寫程式，因為時間寶貴，會寫不一定要寫，除非是舉手之勞，否則還是應該交給資訊部同仁代勞。資訊部同仁可以隨時幫其他同仁以低碼方式「快速」寫出程式。有瑕疵、不美觀無所謂，只要輸出的資料正確就好，重點是快速。資訊部同仁再也不會被責怪、被追著跑，反而經常會被讚賞、感謝。寫的人雖然是資訊部同仁，也是公民開發者。

公民開發者的應用程式不是企業正式列管維護的程式，是給自己用的，也可以分享給他人。企業只要做好資安，公民開發者不需要一板一眼、按照嚴格的軟體工程 SOP 客製低碼程式。SOA-ERP 有嚴謹的權限管理，一萬多個服務元件中的每一個服務元件都可設權限，就算使用者很會寫程式，程式使用到的相關服務元件若沒有被授權，還是無法執行程式。所以，讓公民開發者自由開發程式，不會有資安的問題。

學生也可以是公民開發者

多年來我在輔仁大學和台灣大學開設兩門企業應用程式設計的課，「行動式雲端服務應用設計」和「響應式雲端服務系統開發」，教同學寫 APP 和 RWD 企業應用程式。學生不限科系，有很多文學院、商學院、法學院的學生來修課，他們大部分都沒寫過程式。一個學期下來，有的學生開發了出勤管理系統，有的開發了網路購物系統，有的開發了銷售採購庫存系統，也有開發二手書交換系統，等等。這些學生開發的系統除了作業表單和流程外，也都有數據分析的功能。這些學生並非資訊背景，為什麼他們做得出來，答案是積木，也就是 SOA-ERP 服務元件。他們學會了基本程式指令後，接下來的工作就是組裝積木，而組裝積木的工作就是複製資料目錄的文字貼到現成的範例程式。學生就是公民開發者，他們所寫的程式就是低碼程式。

在我和麻省理工學院（MIT）教授合著的《數據長與數據驅動型組織》書的第七章、第八章、附錄 A、和附錄 B，分別介紹了 APP 和 RWD 企業應用程式的開發範例。

企業應用程式課程所使用的系統之所以能成為公民開發者的低碼工具的原因是背後有一萬多個SOA-ERP 服務元件，但大型企業所使用的 ERP 系統可能不是 SOA-ERP，而是 SAP、Oracle、Baan、Dynamics 等，不論什麼 ERP 系統都可以叫用 SOA-ERP 的服務元件匯入並同步相關資料至 SOA-ERP，接下來公民開發者就能隨心所欲開發自己需要的應用程式。

修過我的企業應用程式課程的學生畢業後到企業任職，可能會負責業務、法務、財務、會計、採購、製造等工作，他們可以自己寫程式滿足自己在工作上的資訊需求，他們就是公民開發者。

老少配的三贏局面

學生畢業後到企業任職自然會累積實務經驗，可以開發自己需要的應用程式。但，在學的學生沒有工作經驗，寫出來的程式不一定實用。怎麼辦？老少配。

台灣有很多退休的企業人士，經驗豐富、身體健康。我們如果成立一個顧問平台，號召這些資深人

員，讓他們和中小企業使用者溝通，了解他們的痛點，提出改善建議，解決中小企業的問題，退休顧問可以開創事業的第二春。解決企業問題不能沒有資訊系統，顧問們在輔導中小企業的時候，可以把會寫企業應用程式的學生帶在旁邊，由學生把顧問的改善建議寫成程式，對中小企業的幫助很大。因為參與了實務工作，解決了實際問題，對學生的幫助也很大。學生和已經退休的顧問應該都沒太大的經濟壓力，所以收費也相對低，中小企業負擔得起。這是一個三贏的局面。中小企業不用安裝任何軟硬體，由顧問平台直接提供作業面、管理面、決策面所需資料的商業模式，我稱之為「資料服務」。

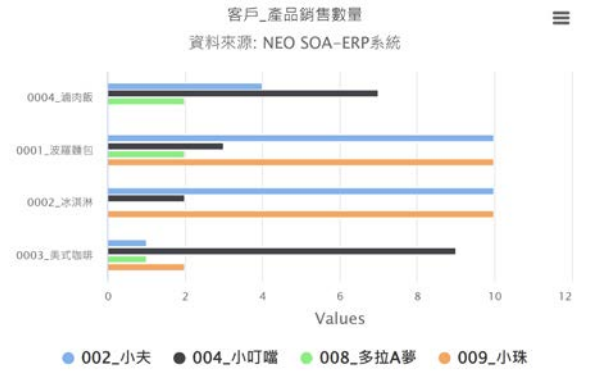
低碼程式範例 — 銷售分析

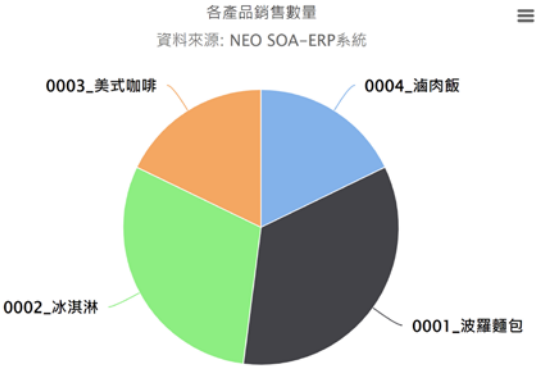
公民開發者開發程式時用不到資料庫，所以也不必懂資料庫。開發程式就如同填空題，而不是作文題。想知道什麼資料，想看什麼報表，打開服務文件的網頁，複製貼上填空題，程式就完成了。以下是一位公民開發者為了想知道某幾個客戶買了哪些產品而寫的程式範例（為了隱私，資料是假的，但程式是真的）。

先下查詢條件：

按「查詢」，出現統計表：

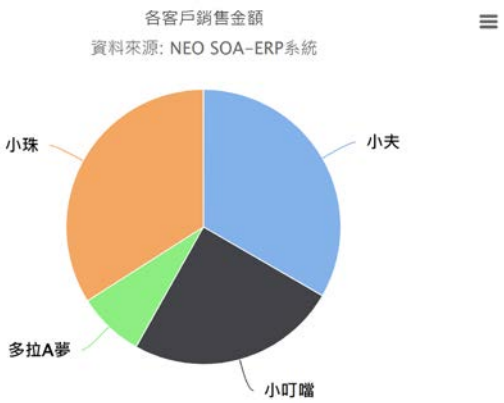
按「看圖」，出現統計圖：





低碼程式範例 — 出納作業

某公司的眾多業務員每次回公司就會拿一疊發票來請款，會計人員必須將發票資料一張一張輸入 ERP 系統。ERP 系統的發票維護程式因為要顧及銷項



進項、正向負向、含稅不含稅、是否要沖帳、沖哪一種帳、…等狀況，故有數十個欄位。除了建立發票資料，為了會計作帳，還要開立傳票底稿單，每一張發票都要建立3個分錄（借費用、借進項稅額、貸零用金），會計人員非常辛苦。公民開發者了解會計人員的痛點後，發現業務員的請款發票一定是進項，不可能需要沖帳，且大部份都有 QR 碼；而作帳所需會計科目不外乎伙食費、交通費等少數幾個，稍加訓練業務員必定知道如何選擇。於是開發了「掃描條碼新增進項發票」程式給業務員使用。這個程式是響應式（RWD），手機拿直所有欄位會自動變成垂直佈置。

業務員拿到發票後用自己的手機掃描發票上的 QR 碼，再按「取」，所有資料自動出現在畫面上。若發票沒有 QR 碼，例如手寫發票或車票，則在畫面輸入相關資料。接著業務員選擇會計科目代號，再按「增」自動產生傳票底稿單號：

業務員再按「確定新增」，就會在 ERP 系統中新增發票以及傳票底稿單和其分錄，不但建立發票也完成了會計傳票的工作。自動產生的傳票內容如下：

SOA-ERP 會通知會計人員審核發票及傳票。會計人員不用再輸入資料，只要檢查內容再按一個鈕就完成發票及會計傳票作業，並直接撥款至員工帳戶。業務員還沒回到公司，他所墊付的費用就已經進入他的銀行帳戶。

因為公民開發者資料治理更加重要

公民開發者寫的程式不是公司正式列管維護的程式，是使用者 DIY 隨心所欲寫的程式，故企業的「資料治理」就很重要了。就像治理國家一樣，訂定法令規章，提供基礎設施，接下來就讓人民自由發揮了。企業必須訂定程式 DIY 的規範，包括角色、部門或個人存取資料的權限。企業也必須提供安全的基礎設施，包括程式樣式、範例模板、資料元件（例如 SOA-ERP 的服務元件）、及資料元件的文件（即資料目錄）。每一個服務元件都需要權限管理，使用者再怎麼會寫程式也看不到不該看的資料。

利用組裝積木式的低碼程式開發方法，讓企業中的公民開發者 DIY 寫程式給自己或同事使用，將會是未來的常態。

當組織內部有很多公民開發者的時候，自然會轉型為資料驅動型組織。