共同開發平台世界中之協同合作　— 第一部分：協同開發工具

級別：入門

Gary Pollice，伍斯特理工學院實務教授

2007 年 4 月 15 日

本文來自 Rational Edge：軟體開發領域的協同合作涵蓋各式各樣的團隊活動，小至同一地點的小型專案團隊，大至跨越多個時區、涉及數百名開發人員（包括委外）的跨國專案，都需要協商合作。本文說明現行的協同開發狀況，並探討有哪些因素可能妨礙協同合作最佳化。

全新的 *Rational Edge 讀者論壇*

本文最後提供的鏈結，可前往 *Rational Edge* 讀者專屬電子報。您可針對本文或網站中的任何主題，發表意見。

軟體開發人員及技術人員總是希望有新發現，許多人都致力預測產業的未來趨勢。要成功開發軟體，協同開發及其支援工具是一大關鍵，這個月我將針對這個領域重點探討。

何謂協同合作與協同開發？

軟體開發人員需要在團隊中合作，通常是 5 - 10 人的團隊，共同開發軟體產品或大型產品的元件。規模更大或更小的專案團隊當然也有，但絕大多數都在 5 - 10 人。[1]

在探討協同開發爲何如此重要之前，需要先說明何謂「協同開發」。我們與別人共事時，都會共用各種不同的資訊分工合作。在小團隊中，溝通途徑相對有限，團隊規模擴大時，溝通途徑也大幅增加。不過，協同合作不只是溝通而已，更需要不同實體（個人與群體）之間的互動，彼此朝相同的目標努力，協作時共享知識與成品，相互幫助，以期達成目標。

軟體專案有數種協同合作方式，我們會與直屬團隊的成員共享流程，讓其他團隊瞭解我們的工作方式，也將他人的工作方式融入我們的工作模式；開發流程則會共享程式碼、設計、測試，以及其他衍生的構件。

協同合作能夠達到增效作用，協同合作順利時，會有更好的表現。如果能改善個

人及團隊之間的互動，就能減少工作障礙。就像汽車的運作，如果所有零件能夠順利配合運作，汽車的效能就會大幅提升。

廿一世紀的發展

過去十年來，商業環境已從相互競爭轉爲協同合作。軟體產業發展了元件型系統，以及服務型（如 Web 服務）系統，這些元件與服務並不是由單一的同質團隊所建立，而是來自全球各地的開發人員，原本的設計目的也各不相同。如果發現哪個元件或服務適用我們使用時，經常需要與元件或服務建立者合作，以瞭解其特性甚至加以調整，以符合我們的需求。許多元件是來自開放源碼的提供者，這時候我們通常會僱用該提供者或熟悉軟體的人員，負責調整軟體，整合至我們的系統。

正如 Thomas Friedman 在其著作*《世界是平的：廿一世紀簡史》*所言，世界已經出現重大的轉變。全球各地的成功企業開始利用資源的協同合作，以最有效的方式取得企業所需的服務。隨著協同合作日漸普及，相關的支援工具也出現在市場上，在未來的軟體產業中，協同開發將成爲關鍵要素。我們可以看看有哪些可能的變革。

全新的開發方法與實務

目前的軟體開發流程不斷出現新發展。本專欄上個月曾經探討「敏捷的軟體開發流程」（Agile processes），但這並不是唯一的新方法。軟體開發實務的各個層面都出現新方法，包括專爲協同開發而設計的方法。

假以時日，協同開發流程將會成爲獨立的領域，形成其特質與實務。由於面對面溝通就是「敏捷」的基本原理，所以「敏捷」目前主要針對能夠面對面接觸的團體。[2] 但有許多「敏捷」社群也在努力開發擴充各種流程，使無法定期碰面的團隊成員，也可利用「敏捷」互相合作。我相信除了「敏捷」以外，還有其他有效解決問題的方法；

更正式、定義更清楚的流程，便符合跨地協同合作的需求，例如，由 Watts Humphrey 開發的「團隊軟體流程」（Team Software Process，簡稱 TSP），就是爲了解決跨地協同合作的問題。[3] TSP 的基礎是軟體工程學院的「能力成熟度模式」（Capability Maturity Model，簡稱 CMM）與「能力成熟度整合模式」（Capability Maturity Model Integration，簡稱 CMMI）。許多大型組織都採用這些模式，有效管理大規模專案。

不少軟體流程設計的權威人士，已經開始注意跨地、跨時區的協同合作，Ivar Jacobson 就是一個例子，他致力尋找有效合併實務的方法，並在 "Enough of Processes: Let's do Practices" 中詳細介紹。[4] 分散的團隊要有效協同合作，必須採取不同的實務，Grady Booch 已多次談到協同開發，[5] 他與 Jacobson 都是軟體開發界的先驅，正如業界其他人士，也都認同支援分散團隊的重要性。

目前的實務會有什麼變革？其中一個焦點是同步處理分散團隊的活動與構件的能力，目前使用的協同開發環境便是有力的例子。Rational 工具如 ClearCase、ClearQuest 與 RequisitePro 都提供這類功能，讓團隊共享不同類型的資產。部分的協同開發社群如 VA Software 的 SourceForge，就爲開放源碼的開發者建立協作環境。VA Software 建立了協同開發流程，提供最佳實務方便使用者將 SourceForge 納入組織現有流程，也建立完整流程，專供目前尚無定義流程的組織使用。[6] 然而，只有 SourceForge 的企業版才提供這些功能，在 SourceForge.net 的社群一般不會提供。

協同開發工具

支援協同開發的工具領域亟需創新的構想。如果瓦斯特理工大學的學生跑來告訴我，他們對建置軟體開發工具有興趣，我會鼓勵他們著眼於協同開發工具，以及爲跨地區、時區團隊而設計的技術。

協同開發工具之所以吸引人，是因爲其中既有短期商機，也有長期商機。在短期方面，我們必須調整現有工具，以支援分散式協同合作，目前已有許多工具能夠支援多個使用者。多年以來，需求管理工具、原始碼管理與版本控制軟體，都爲多個使用者提供不錯的支援。如果建立專案的工具無法支援多個使用者，則主要透過版本控制軟體共享專案構件。我們通常會將文件、計劃、設計與其他構件儲存在團隊檔案庫中，讓團隊各成員能夠一致使用。中央檔案庫不但可以控制團隊成員的資料存取權，還能統一專案。分散式檔案庫已日漸增加，只不過仍遠不如中央儲存庫那麼普遍。

目前有些工具號稱爲協同開發環境 (CDE)，SourceForge 與 CollabNet 就是其中兩種熱門工具，工具包含的「協同開發」，是指團隊用來溝通與共享構件（包括程式碼）的中央檔案庫。這類工具都會使用資料庫來儲存構件與版本控制工具（通常是 CVS 或 Subversion），以便管理原始碼。也可以延伸使用外部工具如 IBM Rational ClearCase 與其他工具，方便建立及管理團隊的構件。IBM 已發展出名爲 Jazz 的全新開發平台方案，並在 2006 年的 Rational Software Developer Conference 發佈，預計近期就會上市，這項方案將大幅提升協作支援的水準。Jazz 是以 Eclipse 架構爲基礎。

瓦斯特理工大學採用 SourceForge 企業版已將近三年，目前代管了大約 1,000 名使用者與 300 多個專案的資訊。「代管」一詞正足以說明現行各種 CDE 所提供的功能。工具會爲每個專案建立工作區，以此「代管」團隊所建立與使用的資源，同時提供其他功能，協助團隊更有效地協同合作。由於我對 SourceForge 最爲熟悉，我將介紹當中爲分散式協同開發所提供的額外功能。我以各種「標籤」名稱（如圖 1 所示）指出工具的各個部分，但並不依出現的順序討論。

圖 1： SourceForge Enterprise 專案標籤

「**Discussions**」是討論主題，可在支援分散社群的黑板系統或網站找到。近來 SourceForge 納入了 Wiki 功能，我的專案更常使用此功能，而非討論區。雖然 Wiki 提供簡單的資訊架構，但容許討論結構不斷擴張，不受既定的框架侷限，支援團隊成員進行更多樣化的討論。

SourceForge 支援團隊任務管理，以及追蹤缺失與其他構件。SourceForge 的「**Tasks**」標籤讓團隊建立結構化的工作指派記錄。實作時可建立任務階層，以不同的方式對映至團隊。SourceForge 大部分的視圖都可過濾檢視的資訊，選取每頁的項目數量。圖 2 顯示過濾後的任務，當中只顯示還未開始的任務。目前的過濾機制仍相當簡單，隨著產品演進，應可繼續改良。

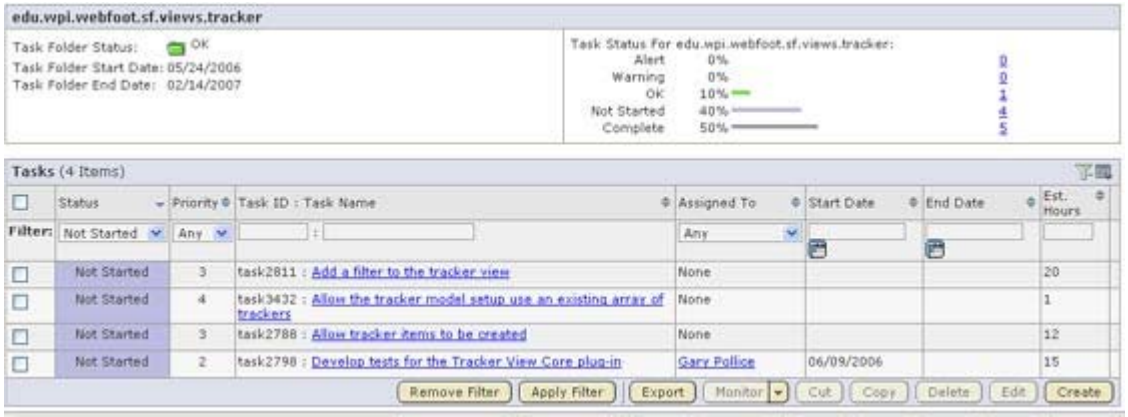

圖 2： 過濾 Task 視圖
按一下放大

「Documents」標籤專供一般性儲存使用，可以儲存專案文件，或團隊可能參考、產生的各類檔案。「Documents」的審查概念是一項很好的支援功能。開始審查文件時，選取必要與選用的審查者，然後通知他們，審查完成以後您將會收到通知。過程其實並不簡單，但一般常視之爲理所當然，或以特定方法執行。其他的

CDE 與協同合作工具如 Microsoft SharePoint，也提供這項功能。

SourceForge 不以特定的構件類型追蹤缺失，而是提供一般的構件追蹤器。每個專案皆可設計、使用與追蹤任何數目的構件類型，這個 SourceForge 的功能非常有彈性，可以設計缺失報告、使用者案例與其他供專案使用的構件。至於「**File Releases**」標籤，則對產生軟體的專案非常重要。使用標籤可以包裝產品，針對不同平台設定多種配置，並給予標籤，讓客戶藉此功能存取。若是開放源碼的專案，標籤功能可透過立即可用式表單，輕鬆取得軟體。任何 CDE 都必須維護存取安全。SourceForge 與其他 CDE 都非常重視此課題，爲專案管理者提供多種方式，自訂使用者專用權與存取權。如果少了這項功能，CDE 就無法幫助協助組織管理其智慧財產權。

目前還沒有萬靈仙丹

使用 CDE 當然有幫助，但仍有幾方面妨礙團隊的協同合作。我個人曾遭遇兩大問題：

- 不同用途需要使用不同的工具，因而經常切換環境定義。
- 不同的工具有重複的功能，因此產生混淆。

我們逐一檢討這些問題。例如我在自己的程式設計環境 (IDE) 工作，必須查看最新的 API 或 UML 模組圖，而該模組以 Web 服務整合，我需要的資訊很可能位於使用中的原始碼管理系統，也就是我的 IDE 所連結的系統。這時候，我可能需要登出，然後在 IDE 尋找一個可以檢視構件的視圖。Eclipse 與其他類似的 IDE 早就已經支援這種操作，然而，如果需要的構件是位於其他類型的檔案庫，例如 SourceForge 專案的「**Documents**」標籤，該怎麼辦？在這種情況下，我必須離開 IDE，進入 SourceForge 尋找文件，將文件下載至機器檢視，而且可能需要第三個工具來檢視。理想的解決方案是在 SourceForge 中尋找文件，然後在 IDE 開啓。

另外，如果不同的工具有重複的功能，會發生更嚴重的問題。我向學生示範不同的軟體開發工具時，就曾發生以下狀況：所有的學生都使用 Eclipse 與 SourceForge，我們也有一些 Rally 的授權，那是可以支援「敏捷軟體開發」的 Web 型專案管理工作區。我希望在課堂上使用這三種產品。Eclipse 是瓦斯特理工大學選擇使用的 IDE，SourceForge 擁有前述的功能，但 Rally 在支援需求管理、使用者案例與使用案例規格方面有絕佳的表現。存取各產品必須切換環境定義，SourceForge 與 Rally 又需要不同的登入 ID，而我們就在切換時發現衝突。三種產品各有稱爲 Task 的構件。

在 Eclipse 中，Task 只不過是含名稱的標籤，以及附加在部分 IDE 元素的有限狀態，該元素可能是程式碼或檔案的區段。而在 SourceForge 中，Task 則是定義團隊成員執行的工作，其預先定義的格式無法變更。Rally 的 Task 概念與 SourceForge 相當類似，但 Task 項目的綱目則有所不同。所以，我們必須想辦法解決。Eclipse 的 Task 概念局限大太，不適合我們使用，我們並不想使用 Task 爲團隊成員安排工作。然而，我們在 SourceForge 儲存了許多資源，如果因此選擇 SourceForge 的 Task，我們便無法利用 Rally 的 Task，以及爲使用者案例與其他需求提供的部分鏈結。我們可以手動複製所有的 Task，但也不是明智之舉。最後，爲了減少開機次數，我們決定放棄 Rally 工具不用。我覺得很可惜，因爲同學們本來可以使用 Rally，學習到許多需求規格。

我們可以更深入探討這個問題，例如有一個專案涉及分散的開發團隊，而每個團隊都使用不同的工具組合，這種情況並不是不可能的，現在委外與跨企業協同合作都非常普遍。假設團隊 A 使用 Eclipse 做爲 IDE，並採用 IBM Rational ClearQuest 來追蹤缺失；團隊 B 使用 SourceForge 與 Eclipse；而團隊 C 則使用 NetBeans 做爲 IDE，又採用 Rally 執行專案管理與專案工作區。這些團隊該如何協同合作？他們只有兩種選擇：同意所有團隊採用單一環境，或是耗費精力複製構件，並保持所有環境同步。然而，兩種方案都不可行。

協同開發工具的未來

這個複雜的可能性，正點出協同開發工具的美好前景。我們如果要全球協同合作，便必須能夠融合開發文化與工具，這就需要一個極爲彈性的環境，讓各種工具得以有效搭配運作。要整合文字編輯器這類工具的個人偏好設定，其實並不困難，甚至根本不需要改變。相對的，如果要處理程式碼庫，有些程式碼庫可能儲存於 ClearCase 檔案庫，有些則儲存在 Subversion 檔案庫，所以需要使用橋接工具。

要成功橋接，需要進行兩件事：

1. 開啓個別工具；「開啓」是指工具必須擁有定義良好的 API，可供其他工具加以整合或將其延伸。工具不一定都要開放源碼，但必須是開放式介面。
2. 協同合作工具必須讓分散的開發團隊，輕易取到所需概念的摘要，同時還能解決衝突，並且在各團隊的環境中，以構件重現概念。

我前面提到的許多工具，都符合第一個條件。Eclipse 與 NetBeans 都是開放源

碼專案的 IDE，兩者已發佈 API，可用於建置延伸。[7] SourceForge 與 Rally 也有用於整合的 API。API 的品質參差，主要原因是產品仍未成熟，不過未來應可逐漸改善。

比較困難的問題是滿足第二項條件。我們需要一種工具，即使專案構件異質性很高，仍然能夠將專案構件同步。這些問題解決以後，便可大幅改善協同開發環境。

協作工具的前景非常樂觀。IDE 將會結合通訊技術，視訊與音訊也可加入不同的註解，使需求更精確，在不同的文化更易理解，這是令人振奮的發展。

在不久的將來還有特別令我振奮的事情，我前面說過，IBM 新開發的平台方案 Jazz 即將推出。雖然我不是非常熟悉，但我看過 IBM 麻省劍橋實驗室的早期研究報告，其中探討了部分 Jazz 的基礎原理。而在最近舉行的 Eclipse 會議中，也有幾場有關 Jazz 的簡報。雖然我沒有參加這些會議，但我知道 Jazz 是以 Eclipse 平台爲建置基礎，針對改善團隊之間的協同合作，提供了一組整合工具。以 Eclipse 爲建置基礎，可以解決前面的一些問題。我熱切期待取得及使用 Jazz，看看瓦斯特理工大學的研究團隊可以如何運用。

註解

[1] Grady Booch 曾在數年前的 Rational 使用者會議中提及。

[2] 請參閱 http://www.agilemanifesto.org/principles.html。

[3] 請參閱 http://www.sei.cmu.edu/str/descriptions/psp_body.html。其中表示「開發 TSP 是爲了更廣泛的專案應用，包括大規模的多個團隊、分散不同區域的團隊，以及功能團隊 [McAndrews 00]。」

[4] 文章分爲兩部分，刊載於 *Dr. Dobb's Journal* 2007 年 4 月號及 5 月號。

[5] 請參閱 http://www.alphaworks.ibm.com/contentnr/cdeintro?open, http://www.ddj.com/dept/architect/196900222, and his paper with Alan Brown, http://www.booch.com/architecture/blog/artifacts/CDE.pdf。

[6] 請參閱 http://www.vasoftware.com/sourceforge/process_controls.php。

[7] 我不確定 CollabNet 有沒有這類 API，但我相信應該是有的。

立即討論本文！

親愛的讀者：我們特別建立新討論區，專供討論 *Rational Edge 文章*，您可以就本文、本期其他文章，或前期文章，發表您的感想與意見。您可以閱讀全球同儕的看法，也可以發表新的討論主題，或參與現有的討論主題。按一下這裡開始。

作者簡介

Gary Pollice 是麻省伍斯特理工學院的實務教授。他教授軟體工程、設計、測試，以及其他的電腦科學課程，同時也指導學生專案。在跨入教育界之前的 35 年中，他曾開發過各類軟體，從企業應用程式到編譯器和工具，涵蓋類型眾多。他在業界的最後一份工作是在 IBM Rational Software，人稱「RUP 倔老頭」，他也是 Rational Suite 團隊的創設成員。他是 2004 年 Addison-Wesley 出版的 *《小型團隊軟體發展：以 RUP 爲中心的方法》*一書主要作者。Gary Pollice 擁有數學學士及電腦科學碩士學位。