開源社群：

廖永賦

2018-07-02

摘要

緒論

自由與開源軟體（“Free and open-source software,” 2018）不同於版權軟體（如微軟的 Office、Google 的 Gmail 等），是開放原始碼[1]的軟體。開放原始碼意味使用者能自由的使用軟體，例如，根據自己的喜好修改軟體以符合使用的需求或偏好，並且也能向軟體開發者提供修改軟體的建議或程式碼。開源軟體因而能吸收來自網路上眾多「駭客」[2]的貢獻，增加其修改軟體缺陷的速度與能力（Raymond, 1999），而產生許多優秀的開源軟體。手機作業系統 Android、辦公室軟體 LibreOffice、統計語言 R、影音播放器 VLC 等，是幾個知名的開源軟體。

開源社群則是圍繞著開源軟體開發而產生的社群，由一群對軟體開發有熱情的人組成。他們所開發的軟體以開放原始碼的形式釋出，並且接受來自網路上使用者的回饋以及修正。開源社群最大、也最引人注目的特色在於其參與者並無法直接從開發軟體中獲利（因爲開源軟體開放任何人能夠使用），但卻有非常多的人從事開源軟體的開發，且社群正穩定地成長與擴大。這在許多研究者眼中顯得相當不可思議，因此引發了許多研究，企圖了解開源社群的運作。以下，本文將先回顧開源社群形成的脈絡；接著再以個人動機以及禮物交換的邏輯探討開源社群的運作；最後探討代表資本主義的商業公司與開源社群互動、結合的現象。

歷史背景

開源社群的出現與電腦資訊科技息息相關。1960-1970 年代的電腦作業系統與網際網路的開發，多由大學以及私人公司資助的實驗室（如貝爾實驗室）進行。當時的研究人員時常分享彼此的原始碼，以非正式的方式授權不同機構的人們互相使用程式碼。因此，開源社群最早的形式是一群互相分享原始碼的研究人員，亦即，開源社群起源於學術界。如今，開源社群的一些特色也顯示其源自學術界的特性，例如，強調「研究[3]可複製」的重要性（Peng, 2011）、軟體開發時的同儕審查機制[4]（Bergquist Magnus & Ljungberg Jan, 2008）等。

隨著資訊科技的逐漸發展，1980 年美國將軟體納入智慧財產的保障（“Software copyright,” 2018），私人公司開始封閉其所擁有軟體之原始碼。Richard Stallman 對於此現象感到不滿，他認爲自由地使用、修改軟體是每個人的權利，而封閉原始碼阻止了人們修改程式碼，是不道德的（**???**）。因此，Stallman 推動了自由軟體運動（Free Software Movement）並成立了自由軟體基金會（Free Software Foundation），旨在推廣自由軟體開發。其中一個最能代表 Stallman 理念的例子是 GPL 條款：「使用 GPL 授權的軟體，容許其使用者自由地修改、複製、發布軟體的衍生版本（即需開放原始碼）。修改過的軟體，必須使用 GPL 作爲授權條款」。透過 GPL 條款，一個軟體可以一直維持開源的狀況，爲開源軟體的發展提供了一項保障。

---

[1] 軟體的原始碼是由高階的程式語言所撰寫，如常見的 C++ 語言、Java 等，這些語言是人類能夠「看懂」的語言。一個軟體要能在電腦上運行，需要將高階的程式語言轉換成低階的機械碼。機械碼是電腦「看的懂」的語言（全部由 0 或 1 組成），但人類很難讀懂的語言，且機械碼也無法被轉換回高階語言。因此，私有的版權軟體爲保護自己的智慧財產，通常僅以機械碼的形式將軟體發布給使用者。開源軟體則會公開其原始碼，讓使用者能夠修改該軟體。

[2] 駭客（hacker）並非大眾媒體上所指「破解電腦以竊取資料」的電腦高手，其正確的詞彙應爲 ‘cracker’（Raymond, 2004）。駭客在開源社群或更廣泛的駭客社群中，是帶有正面意義與認同的詞彙。

[3] 這裡的「研究」在強調軟體開發的開源社群中變成「程式碼」，亦即，任何人皆能夠在自己的電腦上透過該程式碼得到相同的結果。

[4] 軟體開發專案的擁有者能決定是否採用其他人對於軟體的修改。

1990 年代，隨著網際網路的普及，開源社群的發展也更加蓬勃。同時，商業公司與開源社群的互動開始變得更加密切，這也反映在開源軟體運動（Open-Software Movement）的出現。不同於自由軟體運動的理念，開源軟體運動認爲開源軟體不應該排拒閉源軟體，例如，其認爲軟體的授權條款不應該限制軟體各組成部份應使用何種授權條款。GPL 條款即不符合開源軟體運動的精神，因爲其要求修改過之軟體必須以 GPL 條款釋出。開源軟體運動的出現，提倡使用較寬鬆的授權條款，使得開源軟體專案可以使用私人閉源軟體作爲其部分。

開源社群的運作

開源社群的出現與長期下來的穩定成長，造成許多學者的關注，包含對於社群成員之內在動機的探討 [（Lakhani & Wolf, 2003; Lerner & Tirole, 2001; Lerner Josh & Tirole Jean, 2003）、對於社群內運作之探討（Bergquist Magnus & Ljungberg Jan, 2008; Zeitlyn, 2003）以及對於私人公司與開源社群互動之探討（Andersen-Gott, Ghinea, & Bygstad, 2012; Dave Elder-Vass, 2015）。本文以個人動機切入，企圖透過個體了解開源社群的文化與核心概念，以嘗試理解禮物交換的邏輯如何運作於開源社群之內。

在進入的詳細的討論前，先了解開源軟體專案的運作模式有很大的幫助。一個典型的開源軟體專案通常會將軟體的原始碼託管在 GitHub 上，讓所有人透過網路即能檢視該軟體如何被寫成。軟體專案通常具有一位或少數幾位管理者，通常也是該軟體專案的創建者。軟體的絕大部份由這些人撰寫，並且在軟體開發成熟後轉換爲軟體的維護者，整合來自各方所提出對於修改軟體的建議。一個好的軟體會吸引許多使用者，隨著使用者的增加，改進該軟體會變得更加容易，因爲使用者會向軟體管理者回報使用上遇到的程式漏洞（bug）。少數具備優秀程式能力的使用者更會自己修改軟體，並將修改過後的程式碼提交[5]給管理者，其中的更少數可能因爲經常提交修改而被邀請加入軟體管理者的行列。

個人動機

禮物交換

資本主義

參考資料

Andersen-Gott, M., Ghinea, G., & Bygstad, B. (2012). Why do commercial companies contribute to open source software? *International Journal of Information Management*, *32*(2), 106–117. https://doi.org/10.1016/j.ijinfomgt.2011.10.003

Bergquist Magnus, & Ljungberg Jan. (2008). The power of gifts: Organizing social relationships in open source communities. *Information Systems Journal*, *11*(4), 305–320. https://doi.org/10.1046/j.1365-2575.2001.00111.x

Dave Elder-Vass. (2015). The Moral Economy of Digital Gifts. *The International Journal of Social Quality*,

[5]這在版本控制系統 Git 以及應運而生的 GitHub 出現後變得非常容易進行。軟體管理者可以清楚地看到軟體修改前後的差異，並決定是否將此變更整合進原本的軟體，或是需要更進一步地修改。

5(1), 35–50. https://doi.org/10.3167/IJSQ.2015.050103

Free and open-source software. (2018). *Wikipedia*.

Lakhani, K. R., & Wolf, R. G. (2003). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. *MIT Sloan Working Paper No. 4425-03*.

Lerner, J., & Tirole, J. (2001). The open source movement: Key research questions. *15th Annual Congress of the European Economic Association*, *45*(4), 819–826. https://doi.org/10.1016/S0014-2921(01)00124-6

Lerner Josh, & Tirole Jean. (2003). Some Simple Economics of Open Source. *The Journal of Industrial Economics*, *50*(2), 197–234. https://doi.org/10.1111/1467-6451.00174

Peng, R. D. (2011). Reproducible Research in Computational Science. *Science*, *334*(6060), 1226. https://doi.org/10.1126/science.1213847

Raymond, E. (1999). The cathedral and the bazaar. *Knowledge, Technology & Policy*, *12*(3), 23–49. https://doi.org/10.1007/s12130-999-1026-0

Raymond, E. (2004). The Jargon File: Hacker. http://www.catb.org/jargon/html/H/hacker.html.

Software copyright. (2018). *Wikipedia*.

Zeitlyn, D. (2003). Gift economies in the development of open source software: Anthropological reflections. *Open Source Software Development*, *32*(7), 1287–1291. https://doi.org/10.1016/S0048-7333(03)00053-2