TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GTVT
KHOA CÔNG NGHỆ THÔNG TIN

THS. PHAN NHƯ MINH
(Bộ môn truyền thông và mạng máy tính)

BÀI GIẢNG
PHẦN MỀM MÃ NGUỒN MỞ
DÙNG CHO SINH VIÊN KHOA CÔNG NGHỆ THÔNG TIN

LƯU HÀNH NỘI BỘ
Hà nội 2022

Mục lục

1 Giới thiệu

> "Nếu bạn có một quả táo và tôi có một quả táo và chúng ta trao đổi táo cho nhau, thì bạn và tôi sẽ vẫn mỗi người có một quả táo. Nhưng nếu bạn có một ý tưởng và tôi có một ý tưởng và chúng ta trao đổi cho nhau những ý tưởng đó, thì mỗi người chúng ta sẽ có 2 ý tưởng".
>
> Bernard Shaw.

Phần mềm tự do (PMTD) là gì? Nó là gì và những ảnh hưởng của một giấy phép chương trình tự do là gì? PMTD được phát triển như thế nào? Các dự án PMTD được cấp vốn như thế nào và các mô hình kinh doanh nào có liên quan tới chúng mà chúng ta đang trải nghiệm? Cái gì thôi thúc các lập trình viên, đặc biệt là những tình nguyện viên, để tham gia vào trong các dự án PMTD? Các lập trình viên này thích cái gì? Các dự án của họ được điều phối như thế nào, và các phần mềm mà họ sản xuất ra giống thứ gì? Ngắn gọn, bức tranh toàn cảnh của PMTD là gì? Đây là dạng các câu hỏi mà chúng ta sẽ cố gắng trả lời trong tài liệu này. Mặc dù vì PMTD đang gia tăng sự hiện diện của nó trên các phương tiện thông tin đại chúng và trong các tranh cãi giữa những người chuyên nghiệp về công nghệ thông tin (IT), và dù ngay cả các công dân nói chung đang bắt đầu nói về nó, thì điều này đối với hầu hết mọi người vẫn còn lạ lẫm chưa rõ. Và ngay cả những người mà đã quen với nó vẫn thường nhận thức được chỉ về một trong những tính năng của nó, và hầu hết bỏ qua về những thứ khác. Để bắt đầu, trong chương này chúng tôi sẽ trình bày những khía cạnh đặc thù của PMTD, tập trung chủ yếu vào việc giải thích nền tảng cơ bản của nó cho những người tiếp cận chủ đề này lần đầu tiên, và nhấn mạnh tới tầm quan trọng của nó. Như một phần của nền tảng cơ bản, chúng tôi sẽ phản ánh về định nghĩa của khái niệm này (để biết chúng ta đang nói về cái gì) và về những hậu quả chính của việc sử dụng (và chỉ là sự tồn tại của) PMTD.

1.1 Khái niệm về sự tự do của phần mềm

Kể từ đầu những năm 70 chúng ta đã bắt đầu quen với thực tế là bất kỳ ai thương mại hóa một chương trình có thể áp đặt (và đang áp đặt) những điều kiện theo đó chương trình có thể được sử dụng. Việc cho một bên thứ 3 mượn có thể sẽ bị cấm, ví dụ vậy. Dù thực tế là các phần mềm là khái niệm của công nghệ hầu như mềm dẻo và có thể thích nghi được mà chúng ta có, thì nó có thể áp đặt sự ngăn cấm (và nó thường bị cấm) áp dụng nó cho những nhu cầu cụ thể nào đó, hoặc sửa các lỗi của nó, mà không có sự đồng ý rõ ràng của nhà sản xuất, người thường giữ quyền riêng cho những khả năng này. Nhưng điều này chỉ là một trong những khả năng mà pháp luật hiện hành đưa ra: PMTD, mặt khác, đưa ra những sự tự do mà các phần mềm sở hữu độc quyền từ chối.

Phần mềm sở hữu độc quyền (PMSHĐQ)

Trong tài liệu này chúng tôi sẽ sử dụng khái niệm PMSHĐQ để chỉ tới bất kỳ chương trình nào mà nó không được coi là PMTD theo định nghĩa mà chúng tôi sẽ đưa ra sau đây.

1.1.1 Định nghĩa

Vì thế, khái niệm PMTD, như được hiểu bởi Richard Stallman trong định nghĩa của ông (FSF, "Định nghĩa PMTD" http://www.gnu.org/philosophy/free-sw.html [120]), tham chiếu tới các quyền tự do được đảm bảo cho người nhận nó, được nhắc tới 4 quyền sau:

1. Tự do chạy chương trình ở bất cứ đâu, vì bất kỳ mục đích gì và vĩnh viễn.
2. Tự do nghiên cứu cách mà nó làm việc và để áp dụng nó cho các nhu cầu của chúng ta. Điều này cần tới sự truy cập vào mã nguồn.
3. Tự do phân phối lại các bản sao, sao cho chúng ta có thể giúp được những bạn bè và hàng xóm của chúng ta.
4. Tự do cải tiến chương trình và đưa những cải tiến đó ra cho công chúng. Điều này cũng cần tới mã nguồn.

Cơ chế mà đảm bảo cho những quyền tự do này, theo pháp luật hiện hành, là sự phân phối theo một giấy phép đặc biệt như chúng ta sẽ thấy sau (trong chương 3). Thông qua giấy phép này, tác giả trao các quyền cho người nhận chương trình để thi hành các quyền tự do này, cũng như bổ sung thêm bất kỳ sự hạn chế nào mà tác giả có thể mong muốn áp dụng (như để công nhận các tác giả ban đầu trong trường hợp của một sự phân phối lại). Để giấy phép được xem là tự do, những hạn chế này phải không làm mất tác dụng các quyền tự do đã được nêu ở trên.

Sự không rõ nghĩa của khái niệm tự do

Khái niệm PMTD trong tiếng Anh là "Free" ("Tự do"), ám chỉ 'sự tự do', nhưng khái niệm này có thể còn có nghĩa là 'miễn phí' hoặc 'biếu không', mà nó gây ra nhiều sự hiểu lầm. Chính vì thế mà vì sao trong một số trường hợp tiếng Anh mượn các từ tiếng Tây Ban Nha hoặc Pháp và ám chỉ tới PMTD, đối nghịch lại với *phần mềm biếu không*.

Vì thế, những định nghĩa của PMTD không tham chiếu tới thực tế rằng nó có thể bị hiểu là miễn phí: PMTD và phần mềm biếu không là 2 thứ rất khác nhau. Tuy nhiên, để nói về điều này, chúng tôi cũng phải giải thích rằng vì quyền tự do thứ 3, nên bất kỳ ai cũng có thể phân phối lại một chương trình mà không cần phải hỏi sự cho phép hoặc sự thưởng công bằng tài chính, mà chúng làm cho không thể, trên thực tế, để đạt được những lợi nhuận lớn chỉ bằng cách phân phối PMTD: bất kỳ ai mà đã có được PMTD đều có thể phân phối lại nó với một giá thấp hơn, hoặc là hoàn toàn tự do.

Lưu ý

Bất chấp thực tế là bất kỳ ai cũng có thể thương mại hóa một chương trình cho trước ở bất kỳ giá nào, và việc về mặt lý thuyết thì điều này có nghĩa là giá thành của việc phân phối lại có xu hướng trở thành giá cho việc sao chép chương trình, vẫn có các mô hình doanh nghiệp dựa chính xác trên việc bán PMTD, vì có nhiều hoàn cảnh trong đó khách hàng sẽ được chuẩn bị để trả tiền để có được những lợi ích cụ thể khác nào đó, ví dụ như một sự đảm bảo, dù là một sự đảm bảo tưởng tượng, cho các phần mềm được mua hoặc một giá trị gia tăng được lựa chọn, việc nâng cấp và tổ chức của một tập hợp các chương trình.

Từ quan điểm thực tế, một vài văn bản xác định chính xác hơn những điều kiện gì một giấy phép phải đáp ứng để được xem là một giấy phép PMTD.

Trong số đó, chúng tôi có thể nhấn mạnh tới tầm quan trọng lịch sử của chúng, định nghĩa PMTD của FSF (http://www.gnu.org/philosophy/free-sw.html) 120], mà Debian chỉ dẫn cho việc quyết định liệu một chương trình có là tự do hay không (http://www.debian.org/social_contract.html#guidelines) [104] và định nghĩa khái niệm nguồn mở bởi tổ chức Sáng kiến Nguồn Mở (http://www.opensource.org/docs/definition_plain.html) [215], mà nó rất tương tự như những thứ nêu ở trên.

Lưu ý

Ví dụ, các chỉ dẫn của Debian đi vào chi tiết của việc cho phép tác giả yêu cầu rằng các mã nguồn được phân phối sẽ không được sửa trực tiếp, mà thay vào đó là bản gốc được đi kèm với các bản vá riêng rẽ và rằng các chương trình nhị phân được tạo ra phải với các tên khác so với bản gốc. Họ cũng yêu cầu rằng các giấy phép không được làm xấu đi các chương trình khác được phân phối theo cùng cách này.

1.1.2 Các khái niệm có liên quan

Khái niệm *Phần mềm nguồn mở* (PMNM), được khởi xướng bởi Eric Raymond và tổ chức Sáng kiến Nguồn Mở (OSI) là tương đương với khái niệm PMTD. Về mặt triết học mà nói, thì khái niệm này là rất khác vì nó nhấn mạnh tới tính sẵn sàng của mã nguồn và không nhấn mạnh về sự tự do, nhưng định nghĩa này về mặt thực tế là y hệt như của Debian ("Định nghĩa nguồn mở", 1998 http://www.opensource.org/docs/definition_plain.html) [183]. Khái niệm này về mặt chính trị là vô trùng hơn và nhấn mạnh khía cạnh kỹ thuật, mà nó có thể đưa ra những lợi ích về kỹ thuật, như các mô hình phát triển và kinh doanh được cải thiện, an ninh tốt hơn, vân vân. Bị chỉ trích mạnh mẽ bởi Richard Stallman (“Vì sao PMTD là tốt hơn nguồn mở”) [204] và FSF (http://www.fsf.org) [27], nó đã cộng hưởng tốt hơn nhiều với văn hóa thương mại và với các chiến lược của các công ty mà bằng một cách nào đó khác hỗ trợ mô hình này.

Những khái niệm có liên quan theo cách nào đó tới PMTD là như sau:

Freeware Phần mềm quảng bá	Đây là các chương trình biếu không. chúng thường chỉ được phân phối ở dạng nhị phân, và có thể lấy được miễn phí. Đôi khi nó có khả năng có được phép để phân phối lại, và đôi khi không, nghĩa là sau đó nó chỉ có thể có được từ site “chính thức” được duy trì cho mục đích đó. Nó thường được sử dụng để quảng bá cho các chương trình (thường với chức năng hoàn chỉnh hơn) hoặc dịch vụ khác. Các ví dụ của dạng chương trình này bao gồm Skype, Google Earth hoặc Microsoft Messenger.
Shareware Phần mềm chia sẻ	Đây không phải là phần mềm biếu không, mà là một phương thức phân phối khi mà các chương trình có thể được sao chép một cách tự do, thường là không có mã nguồn, nhưng không được sử dụng một cách liên tục mà không phải trả tiền cho chúng. Yêu cầu trả tiền có thể có động lực hoặc chỉ là sự lôi cuốn tới đạo đức của người sử dụng. Hơn nữa, những khái niệm pháp lý của giấy phép này có thể được sử dụng chống lại người vi phạm.

Charityware, careware Phần mềm từ thiện	Đây thường là phần mềm chia sẻ mà nó đòi hỏi trả tiền để được chỉ tới một tổ chức từ thiện được bảo trợ. Trong nhiều trường hợp, thay vì yêu cầu trả tiền, một đóng góp tự nguyện có thể sẽ được yêu cầu. Một số PMTD, như Vim, yêu cầu những đóng góp tự nguyện theo cách này (Brian Molenaar, "Ngữ cảnh của phần mềm từ thiện là gì?") [173].
Public domain Miền công cộng	Ở đây, tác giả hoàn toàn không thừa nhận tất cả các quyền của mình vì lợi ích của miền cộng đồng, và điều này cần phải được tuyên bố chắc chắn trong chương trình vì nếu không, chương trình sẽ bị cho là sở hữu độc quyền và không có gì có thể được làm với nó. Trong trường hợp này, nếu mã nguồn bổ sung được cung cấp, thì chương trình cũng là tự do.
Copyleft Ngược của từ bản quyền (Copyright)	Đây là một trường hợp đặc biệt của PMTD nơi mà giấy phép đòi hỏi bất kỳ sửa đổi được phân phối nào cũng phải là tự do.
Proprietary, locked-in, non-free Sở hữu độc quyền, bị khóa trói, không tự do	Những khái niệm này được sử dụng để tham chiếu tới các phần mềm mà chúng không phải là tự do và cũng không phải là nguồn mở.

1.2 Động lực

Như chúng ta đã thấy, có 2 dòng động lực lớn cho sự phát triển PMTD, mà cũng đưa ra 2 cái tên được biết tới như:

- Động lực về đạo đức, được bênh vực bởi FSF (http://www.fsf.org) [27], mà nó đã kế thừa được văn hóa của các *cao thủ (hacker)* và hỗ trợ sử dụng khái niệm *tự do*, viện lý rằng phần mềm là tri thức mà phải được chia sẻ không có trở ngại gì, rằng việc dấu nó là phản xã hội và nói thêm rằng khả năng để sửa đổi các chương trình là một dạng tự do diễn đạt. Bạn có thể nghiên cứu điều này sâu hơn bằng cách đọc (*PMTD, xã hội tự do. Những tiểu luận chọn lọc của Richard M. Stallman*) [211] hoặc phân tích của Pekka Himanen (*Đạo đức và tinh thần cao thủ của kỷ nguyên thông tin*. Random House, 2001) [144].
- Động cơ thực dụng, được bảo vệ bởi tổ chức OSI (http://www.opensource.org) [54] mà nó ủng hộ sử dụng khái niệm *nguồn mở*, và viện lý trường hợp này là về những ưu thế về kỹ thuật và tài chính mà chúng tôi sẽ thảo luận trong phần tiếp sau.

Bên cạnh 2 động lực chính này, mọi người làm việc về PMTD có thể làm thế vì nhiều lý do khác, bao gồm cả vì thú vui (Linus Torvalds và David Diamond, Texere, 2001) [217] hoặc vì tiền, đa phần với *các mô hình kinh doanh* bền vững. Chương 4 nghiên cứu những động lực này một cách chi tiết trên cơ sở các phân tích khách quan.

1.3 Hệ quả của sự tự do của phần mềm

PMTD đưa ra nhiều ưu điểm và, trong một số ít các nhược điểm, thì nhiều cái đã được cường điệu hóa (hoặc được sáng tạo ra) bởi các đối thủ cạnh tranh sở hữu độc quyền. Những nhược điểm dễ thấy nhất là về tài chính, vì như chúng ta đã thấy là không thể kiếm nhiều tiền từ sự phân phối nó, mà có thể và có xu hướng sẽ được làm bởi ai đó khác chứ không phải tác giả. Điều này giải thích vì sao các mô hình kinh doanh và các cơ chế tài chính khác là cần thiết, mà chúng ta sẽ xem trong chương 5. Những nhược điểm khác, như thiếu sự hỗ trợ hoặc chất lượng kém, sẽ liên quan tới việc cung cấp tài chính nhưng cũng trong nhiều trường hợp là không đúng, vì ngay cả các phần mềm không có dạng cung cấp tài chính cũng có xu hướng đưa ra các mức hỗ trợ tốt nhờ các nhóm thảo luận của người sử dụng và các lập trình viên, và thường chất lượng là rất cao.

Ghi tạc trong tâm trí những cân nhắc về tài chính, chúng ta phải lưu tâm rằng mô hình giá thành của PMTD rất khác so với mô hình giá thành của PMSHĐQ, vì một số lượng lớn của nó phát triển bên ngoài nền kinh tế tiền tệ chính thống, và thường sử dụng các cơ chế trao đổi/đổi chác: "Tôi đưa cho anh một chương trình mà anh quan tâm, và anh thích nghi nó vào kiến trúc của anh và tiến hành các cải tiến mà anh cần". Chương 7 thảo luận về các cơ chế kỹ thuật phần mềm đúng đắn để làm cho hầu hết các tài nguyên còn chưa được khai thác của con người với những tính năng đặc biệt của riêng họ, trong khi chương 8 nghiên cứu các công cụ được sử dụng để làm cho sự hợp tác này có hiệu quả. Hơn nữa, một phần lớn giá thành được giảm thiểu bởi thực tế rằng nó là tự do, vì các chương trình mới không cần bắt đầu từ đầu, vì chúng có thể sử dụng lại ngay các phần mềm đã có sẵn. Sự phân phối cũng có giá thành thấp hơn nhiều, vì nó được phân phối thông qua Internet và với việc quảng cáo tự do thông qua các nhóm thảo luận công cộng được thiết kế cho mục đích này.

Một kết luận nữa về các quyền tự do là chất lượng có được từ sự hợp tác tự nguyện của những người đóng góp hoặc phát hiện và thông báo các lỗi trong các môi trường hoặc tình huống mà chúng là không thể tưởng tượng được đối với người lập trình phát triển ban đầu. Hơn nữa, nếu một chương trình không đưa ra chất lượng đủ, thì sự cạnh tranh có thể nắm lấy nó và cải tiến nó trên cơ sở những gì đã có. Đây là cách mà *sự hợp tác và cạnh tranh*, 2 cơ chế hùng mạnh, kết hợp được để sản xuất ra chất lượng tốt hơn.

Bây giờ hãy xem xét những hệ quả có lợi đối với người nhận.

1.3.1 Đối với người sử dụng đầu cuối

Người sử dụng đầu cuối, bất kể là cá nhân hay công ty, có thể thấy sự cạnh tranh thực sự trong thị trường với một xu thế độc quyền. Để chính xác, không cần thiết phải phụ thuộc vào sự hỗ trợ của nhà sản xuất phần mềm, vì có thể có một số công ty, ngay cả là nhỏ với mã nguồn và tri thức mà chúng cho phép họ kinh doanh trong khi vẫn giữ các chương trình cụ thể nào đó là tự do.

Việc cố gắng tìm kiếm chất lượng của một sản phẩm không còn dựa quá nhiều vào *tính đáng tin cậy* của nhà sản xuất như trong chỉ dẫn được đưa ra bởi sự chấp nhận và tính sẵn sàng của cộng đồng đối với mã nguồn. Hơn nữa, chúng ta có thể quên về các hộp đen, mà phải được tin cậy "vì chúng tôi nói thế", và các chiến lược của các nhà sản xuất mà họ có thể quyết định một cách đơn phương liệu có bỏ

qua hay duy trì một sản phẩm cụ thể nào đó hay không.

Việc đánh giá các sản phẩm trước khi chúng được áp dụng được thực hiện dễ dàng hơn nhiều hiện nay, vì tất cả những gì chúng ta phải làm là cài đặt các sản phẩm thay thế trong môi trường thực tế của chúng ta và thử nghiệm chúng, trong khi đối với các PMSHĐQ thì chúng ta phải dựa vào những báo cáo hoặc cố gắng thương thảo bên ngoài với các nhà cung cấp mà không phải lúc nào cũng có thể. Vì sự tự do để sửa đổi chương trình vì sự sử dụng của riêng mình, nên người sử dụng có khả năng tùy biến nó hoặc thích nghi nó vào những yêu cầu sửa các lỗi của riêng mình nếu có. Quá trình sửa lỗi được thấy bởi những người sử dụng các PMSHĐQ thường là đặc biệt gian khổ, nếu không nói là không thể, vì nếu chúng ta định làm cho các lỗi được sửa, thì sự sửa đúng thường sẽ phải được kết hợp trong phiên bản tiếp sau, mà có thể mất nhiều năm để được tung ra, và hơn nữa chúng ta sẽ phải mua chúng một lần nữa. Mặt khác, với PMTD, chúng ta có thể tiến hành sửa lỗi hoặc tự mình sửa lỗi, nếu chúng ta có khả năng, hoặc nếu không thì cũng có thể đưa nhờ làm dịch vụ bên ngoài. Chúng ta còn có thể, trực tiếp hoặc bằng việc ký hợp đồng các dịch vụ bên ngoài, tích hợp chương trình này với các chương trình khác hoặc kiểm thử chất lượng của nó (ví dụ như các vấn đề về an ninh). Mở rộng hơn nữa, sự kiểm soát được đi qua từ nhà cung cấp cho tới người sử dụng, chứ không chỉ nằm ở phía nhà cung cấp như thường thấy đối với các PMSHĐQ.

1.3.2 Đối với nền hành chính nhà nước

Nền hành chính nhà nước là một người sử dụng lớn với những đặc tính đặc thù, vì nó có một bổn phận đặc biệt hướng tới các công dân của mình, liệu có hay không để đưa ra các dịch vụ có thể truy cập được, trung lập trong quan hệ đối với các nhà sản xuất, hay để đảm bảo tính toàn vẹn, tính có ích, tính riêng tư và an ninh các dữ liệu của họ về lâu dài. Tất cả những thứ trên là bổn phận đối với nền hành chính nhà nước sẽ phải tôn trọng hơn đối với các chuẩn nếu đem so với các công ty tư nhân và duy trì các dữ liệu trong các định dạng mở và để xử lý các dữ liệu với các phần mềm mà chúng phụ thuộc thường vào các chiến lược của các công ty nước ngoài, được chứng thực là an ninh bởi một sự kiểm tra nội bộ. Sự áp dụng đối với các chuẩn là một tính năng đáng lưu ý của PMTD mà PMSHĐQ không quan tâm đúng mức y hệt như vậy, vì nó thường nóng lòng để tạo ra các thị trường bị giam cầm.

Hơn nữa, nền hành chính phục vụ như một dạng trình diễn và chỉ dẫn đối với nền công nghiệp, có nghĩa là nó có một ảnh hưởng to lớn, mà phải được định hướng vào việc đan kết các cơ cấu công nghệ để tạo ra được sự thịnh vượng của quốc gia. Sự thịnh vượng này có thể được tạo ra bởi việc khuyến khích sự phát triển của các công ty chuyên về việc phát triển PMTD mới cho nền hành chính, hoặc duy trì, áp dụng hoặc kiểm tra các phần mềm hiện đang tồn tại. Trong chương 6, chúng ta sẽ xem xét vấn đề này sâu hơn.

1.3.3 Đối với lập trình viên

Đối với lập trình viên và người sản xuất phần mềm, sự tự do đáng kể sẽ thay đổi các qui định của cuộc chơi. Nó làm cho dễ dàng hơn để tiếp tục cạnh tranh trong khi vẫn là nhỏ và để có được công nghệ hiện đại. Nó cho phép chúng ta tận dụng được ưu thế công việc của những người khác, cạnh tranh ngay cả

với sản phẩm khác bằng việc sửa đổi mã nguồn của riêng mình, dù đối thủ cạnh tranh sao chụp được có thể sau đó cũng tận dụng được ưu thế về mã nguồn của chúng ta (nếu nó là copyleft). Nếu dự án được quản lý tốt, thì có khả năng có được sự hợp tác tự do của một số lượng người lớn, hơn nữa, để có được sự truy cập tới một hệ thống phân phối gần như hoàn toàn tự do và toàn cầu. Dẫu rằng, vấn đề còn lại là làm thế nào để có được những tài nguyên về tài chính, nếu phần mềm không phải là sản phẩm để bán vì tiền hoa hồng. Chương 5 làm việc với khía cạnh này.

1.3.4 Đối với nhà tích hợp

Đối với các nhà tích hợp, PMTD là thiên đường. Nó có nghĩa là không còn các hộp đen mà cần phải khớp được cùng nhau, thường sử dụng tới kỹ thuật nghịch đảo. Những góc cạnh thô ráp có thể được làm trơn và các phần của các chương trình có thể được tích hợp để có được sản phẩm tích hợp theo yêu cầu, vì có một kho khổng lồ các PMTD từ đó có các phần có thể được trích ra.

1.3.5 Đối với các nhà cung cấp và duy trì dịch vụ

Việc có mã nguồn làm thay đổi mọi thứ và đặt chúng ta vào vị trí y hệt như nhà sản xuất. Nếu vị trí này là không y hệt, thì là vì chúng ta thiếu một tri thức sâu về chương trình mà chỉ các lập trình viên mới có, mà nó có nghĩa là được khuyến cáo cho các nhà cung cấp sự duy trì để tham gia vào các dự án mà chúng đòi hỏi sự duy trì. Giá trị gia tăng của các dịch vụ được đánh giá cao hơn nhiều vì giá thành của chương trình là thấp. Hiện nay thì đây là việc kinh doanh rõ ràng nhất đối với PMTD và là một nơi có sự cạnh tranh nhất có thể.

1.4 Tóm lược

Chương đầu tiên này đã phục vụ như là một cuộc gặp gỡ mào đầu với thế giới của PMTD. Khái niệm này được xác định bởi Richard Stallman dựa trên 4 quyền tự do (tự do chạy, tự do nghiên cứu, tự do phân phối lại và tự do cải tiến), 2 trong số đó đòi hỏi sự truy cập tới mã nguồn. Tính có thể truy cập được này và những ưu thế của nó đã tạo động lực cho quan điểm ít đạo đức hơn và thực dụng hơn, được bảo vệ bởi OSI, mà đã đưa ra khái niệm khác: *PMNM*. Chúng ta cũng lưu ý những khái niệm khác tương tự có liên quan hoặc đối nghịch để làm rõ các khái niệm khác nhau này. Cuối cùng, chúng ta đã thảo luận các hệ quả của PMTD đối với các bên chủ chốt có liên quan.

2 Một chút về lịch sử

> "Khi tôi đã bắt đầu làm việc tại Phòng thí nghiệm Trí tuệ Nhân tạo của MIT vào năm 1971, thì tôi đã trở thành một phần của một cộng đồng chia sẻ phần mềm mà nó đã tồn tại nhiều năm. Việc chia sẻ phần mềm từng không bị hạn chế đối với cộng đồng cụ thể của chúng tôi; nó cũng lâu đời như những chiếc máy tính vậy, hệt như việc chia sẻ các công thức cũng lâu đời như việc nấu ăn. Nhưng chúng tôi đã thực hiện nó nhiều hơn so với hầu hết mọi người. […] Chúng tôi đã không gọi các phần mềm của chúng tôi là *PMTD*, vì khái niệm đó đã chưa từng tồn tại; mà đó là những gì đã từng thế. Bất kỳ khi nào mọi người từ trường đại học khác hoặc từ một công ty khác muốn đưa vào sử dụng một chương trình, thì chúng tôi vui vẻ để cho họ làm. Nếu bạn từng thấy ai đó sử dụng một chương trình không quen biết và thú vị, thì bạn luôn có thể yêu cầu cho xem mã nguồn, sao cho bạn có thể đọc được nó, thay đổi được nó, hoặc tháo tung các phần của nó ra để làm một chương trình mới".
>
> Richard Stallman, "Dự án GNU" (ban đầu được xuất bản trong cuốn sách *Nguồn Mở*) [208].

Dù tất cả lịch sử có liên quan tới IT là vắn tắt cần thiết, thì lịch sử của PMTD vẫn là dài nhất. Trên thực tế, chúng ta có thể nói rằng lúc ban đầu hầu hết tất cả phần mềm được phát triển đã thỏa mãn định nghĩa của PMTD, ngay cả dù khái niệm này đã còn chưa tồn tại. Sau này tình thế đã thay đổi hoàn toàn, và PMSHĐQ đã áp đảo sân chơi, hầu như là độc chiếm, một thời gian khá dài. Chính trong thời gian này các quỹ đã được sắp đặt cho PMTD như chúng ta biết nó ngày nay, và khi mà từng chút một các chương trình tự do đã bắt đầu xuất hiện. Cùng với thời gian, những thứ này bắt đầu tăng trưởng thành một xu thế mà đã tiến bộ và chín muồi tới ngày hôm nay, khi mà PMTD là một khả năng đáng để xem xét trong gần như tất cả các lĩnh vực.

Lịch sử này là không được biết tới một cách rộng rãi, mà đối với nhiều người chuyên nghiệp về IT thì PMSHĐQ là phần mềm "ở tình trạng tự nhiên của nó". Tuy nhiên, tình hình là ngược lại và những mầm mống của sự thay đổi có thể được thấy trong thập kỷ đầu tiên của thế kỷ 21 đã được gieo trong những năm đầu 1980.

Thư mục tham khảo

Không có nhiều câu chuyện chi tiết về PMTD, và những câu chuyện mà chúng có, thường là những tài liệu hạn chế đối với chủ đề chính của chúng. Trong nhiều trường hợp, các độc giả có quan tâm có thể mở rộng tri thức của họ về những gì chúng tôi đã mô tả trong chương này bằng việc đọc "Sáng kiến Nguồn Mở. Lịch sử của OSI" [146] (http://www.opensource.org/docs/history.php), mà nó nhấn mạnh ảnh hưởng của PMTD lên cộng đồng doanh nghiệp vào những năm 1998 và 1999; "Một câu chuyện ngắn về phong trào PMTD/PMNM" [190], của Chris Rasch, mà nó bao trùm lịch sử của PMTD cho tới năm 2000, hoặc "Nguồn gốc và tương lai của PMNM" (1999) [177], của Nathan Newman, mà nó tập trung vào một sự mở rộng rộng lớn trong sự khuyến khích không trực tiếp của chính phủ Mỹ về PMTD hoặc các hệ thống tương tự trong các thập nhiên 1970 và 1980.

2.1 Phần mềm tự do trước phần mềm tự do

PMTD như một khái nhiệm đã không xuất hiện cho tới đầu những năm 1980.

Tuy nhiên, lịch sử của nó có thể được dõi ngược lại vài năm trước.

2.1.1 Và lúc ban đầu nó từng là tự do

Trong những năm 60, toàn cảnh IT đã bị áp đảo bởi các máy tính lớn, chủ yếu được cài đặt trong các công ty và cơ quan chính phủ. IBM từng là nhà sản xuất hàng đầu, con đường phía trước đối với sự cạnh tranh của hãng. Trong thời kỳ này, khi mua một máy tính (phần cứng), thì phần mềm được bổ sung vào. Cho tới khi hợp đồng duy trì được thanh toán, thì việc truy cập được trao cho catalog phần mềm của nhà sản xuất. Hơn nữa, ý tưởng của các chương trình là thứ gì đó "tách biệt" khỏi một quan điểm thương mại từng là khác thường.

Trong giai đoạn này, phần mềm thường được phân phối cùng với mã nguồn của nó (trong nhiều trường hợp chỉ như là mã nguồn), và nói chung, không có bất kỳ hạn chế thực tế nào.

Các nhóm người sử dụng như SHARE (những người sử dụng các hệ thống của IBM) hoặc DECUS (những người sử dụng của DEC) đã tham gia vào những trao đổi này, và ở mức độ nào đó, đã tổ chức ra chúng. Khu vực "các thuật toán" của tạp chí Communication of the ACM từng là một ví dụ tốt khác về một diễn đàn trao đổi. Chúng tôi có thể nói rằng trong những năm đầu đó của IT, phần mềm từng là tự do, ít nhất theo nghĩa là những ai mà đã có được sự truy cập tới nó thì có thể thường có được sự truy cập tới mã nguồn, và đã được sử dụng cho việc chia sẻ nó, sửa đổi nó và cũng chia sẻ những sửa đổi đó.

Vào ngày 30/09/1969, IBM đã công bố rằng từ năm 1970, hãng có thể bán một phần phần mềm của hãng một cách riêng rẽ (Burton Grad, 2002) [131]. Điều này có nghĩa là các khách hàng của hãng có thể không còn có được các chương trình cần thiết được đưa vào trong giá thành của phần cứng nữa. Phần mềm đã bắt đầu sẽ được cảm nhận như thứ gì đó với một giá trị thực bên trong, và hậu quả là, sự truy cập hạn chế một cách cực kỳ kỹ lưỡng tới các chương trình và khả năng của người sử dụng để chia sẻ, sửa đổi hoặc nghiên cứu phần mềm đã bị hạn chế càng nhiều càng tốt (cả về mặt kỹ thuật và pháp lý) ngày càng trở nên phổ biến. Nói một cách khác, tình hình đã thay đổi tới mức mà nó tiếp tục là tình trạng trong thế giới phần mềm lúc bắt đầu thế kỷ 21.

Thư mục tham khảo

Các độc giả có quan tâm trong việc tìm hiểu về giai đoạn chuyển đổi này, có thể đọc, ví dụ như "ICP Directory đã bắt đầu như thế nào" [226] (1998), trong đó Larry Welke thảo luận cách mà một trong những catalog phần mềm không có liên quan tới một nhà sản xuất đã được sinh ra thế nào, và làm thế nào mà trong quá trình này nó đã được phát hiện ra rằng các công ty có thể được chuẩn bị để trả tiền cho các chương trình không được làm ra bởi các nhà sản xuất máy tính của chúng.

Vào giữa những năm 1970 thì PMSHĐQ đã hoàn toàn là phổ biến trong lĩnh vực IT. Điều này có nghĩa là một sự thay đổi văn hóa khổng lồ giữa những người chuyên nghiệp mà đã làm việc với phần mềm và là điểm khởi đầu cho một sự thịnh vượng của một số lượng lớn các công ty chuyên về việc kinh doanh mới này. Nó có thể vẫn còn là hầu hết một thập kỷ trước khi những gì mà chúng ta biết như PMTD đã bắt đầu xuất hiện theo cách có tổ chức và như một phản ứng đối với tình trạng này.

2.1.2 Những năm 70 và đầu những năm 80

Ngay cả khi xu thế áp đảo này đã khai thác được mô hình PMSHĐQ, thì cũng đã có những sáng kiến mà chúng đã chỉ ra một số đặc tính của những gì có thể sau đó được xem là PMTD. Trên thực tế, một số trong số họ đã sản xuất PMTD như chúng ta có thể xác định được nó ngày hôm nay. Tất nhiên, chúng ta có thể nhắc tới SPICE, TeX và Unix, mà là trường hợp phức tạp hơn nhiều.

SPICE (Chương trình Mô phỏng với việc Nhấn mạnh Mạng Tích hợp) là một chương trình được phát triển bởi Đại học Berkeley, California để mô phỏng các đặc tính điện tử của một mạng tích hợp. Nó đã được phát triển và đặt trong miền công cộng bởi tác giả của nó, Donald O. Pederson, năm 1973. SPICE ban đầu từng là một công cụ đào tạo, và đã nhanh chóng lan truyền tới các trường đại học trên toàn thế giới. Ở đó nó đã được sử dụng bởi các sinh viên của những gì mà sau này đã là một nguyên lý mới nổi lên: thiết kế mạng tích hợp. Vì nó nằm trong miền công cộng, nên SPICE có thể được phân phối lại, sửa đổi, nghiên cứu. Nó còn có thể được áp dụng cho những yêu cầu cụ thể nào đó, và phiên bản đó có thể được bán như một sản phẩm sở hữu độc quyền (mà nó là những gì mà một số lượng lớn các công ty đã làm trong một thời gian qua lịch sử của họ). Với những đặc tính này, SPICE đã có tất cả các tấm thẻ để trở thành chuẩn công nghiệp với các phiên bản khác nhau của nó. Và quả thực, đó là những gì đã xảy ra. Đây có thể đã là chương trình đầu tiên có những đặc tính của PMTD mà trong một giai đoạn nào đó đã chiếm lĩnh được một thị trường, phần mềm về trình mô phỏng các mạng tích hợp, và không còn nghi ngờ gì là đã có khả năng để làm được chính xác như vậy nhờ vào những đặc tính đó (bổ sung thêm vào những phẩm chất kỹ thuật không thể chối cãi được của nó).

Thư mục tham khảo

Thông tin nhiều hơn về lịch sử của SPICE có thể tra cứu trong "Cuộc sống của SPICE", được trình bày trong cuộc Gặp gỡ Công nghệ và Mạng Lưỡng cực, Minneapolis, MN, Mỹ, vào tháng 09/1996 [175].

Bạn có thể thấy trang web của SPICE tại http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/.

Donald Knuth đã bắt đầu phát triển TeX trong một kỳ nghỉ của năm 1978. TeX là một hệ thống mẫu trình bày kiểu in điện tử thường được sử dụng để tạo ra các tài liệu chất lượng cao. Từ ban đầu, Knuth đã sử dụng một giấy phép mà ngày nay có thể được xem là một giấy phép PMTD. Khi hệ thống này được xem là đủ ổn định, vào năm 1985, ông đã duy trì giấy phép đó. Khi đó, TeX từng nằm trong những hệ thống nổi tiếng nhất và rộng lớn nhất mà chúng có thể được xem là PMTD.

Thư mục tham khảo

Bạn có thể thấy một số cột mốc trong lịch sử của TeX bằng việc tra cứu trực tuyến tại http://www.math.utah.edu/software/plot79/tex/history.html [39]. Chi tiết hơn, bài viết trên Wikipedia cũng cực kỳ hữu dụng, http://www.wikipedia.org/wiki/TeX [233].

2.1.3 Sự phát triển ban đầu của Unix

Unix, một trong những hệ điều hành có thể khả chuyển được đầu tiên, ban đầu đã được tạo ra bởi Thompson và Ritchie (cùng những người khác) từ Bell Labs của AT&T. Nó đã tiếp tục phát triển kể từ khi ra đời khoảng năm 1972, giúp tạo ra những biến thể bất tận được bán (theo nghĩa đen) bởi hàng chục công ty.

Vào những năm 1973 và 1974, Unix đã tới nhiều trường đại học và trung tâm nghiên cứu trên khắp thế giới, với một giấy phép mà đã cho phép sử dụng nó cho các mục đích hàn lâm. Dù đã có những hạn chế nhất định mà chúng đã ngăn trở sự phân phối tự do của nó, thì trong số các tổ chức mà đã đặt ra một giấy phép mà việc hoạt động là rất giống với những gì sau này được coi là có trong nhiều cộng đồng PMTD.

Những người mà đã truy cập vào mã nguồn của Unix đã làm việc với một hệ thống mà họ có thể nghiên cứu, cải tiến và mở rộng. Một cộng đồng các lập trình viên đã nổi lên xung quanh nó, mà sớm bị hút vào CSRG của Đại học California, tại Berkeley. Cộng đồng này đã phát triển văn hóa riêng của mình, mà như chúng ta thấy sau này, đã là rất quan trọng trong lịch sử của PMTD. Unix, ở một mức độ nào đó, từng là một sự thí điểm sớm cho những gì mà chúng ta có thể thấy với GNU và Linux vài năm sau đó. Nó đã được đưa vào trong một cộng đồng nhỏ hơn nhiều, và giấy phép của AT&T đã là cần thiết, nhưng trong tất cả các khía cạnh, sự phát triển của nó là rất tương tự (trong một thế giới liên kết ít hơn nhiều).

Các phương pháp phát triển cố hữu của PMTD

Trong *Các công dân mạng (Netizens). Trong lịch sử và ảnh hưởng của Usenet và Internet* (IEEE Computer Society Press, 1997 [139], page 139) chúng ta có thể đọc một ít dòng mà chúng có thể tham chiếu tới nhiều dự án PMTD: "Việc đóng góp cho giá trị của Unix trong sự phát triển ban đầu của nó, sự thực là việc mã nguồn đã được mở và sẵn sàng. Nó có thể được kiểm tra, cải thiện và tùy biến".

Trang 142 của cùng thứ này nói rằng: "Những người tiên phong như Henry Spencer đồng ý về tầm quan trọng của việc phải có mã nguồn để có thể xác định và sửa các lỗi được phát hiện. […] Ngay cả trong những năm cuối 1970 và đầu những năm 1980, thực tế là mỗi site Unix đều đã có các nguồn hoàn chỉnh".

Văn bản của Marc Rochkind "Phỏng vấn với Dick Haight" còn chi tiết hơn (*Unix Review*, tháng 05/1986) [198]: "đó là một trong những thứ tuyệt vời về Unix trong những ngày đầu: mọi người thực sự được chia sẻ cho nhau mọi thứ. […] Không chỉ chúng tôi đã học được nhiều trong những ngày xưa ấy từ việc chia sẻ tư liệu, mà chúng tôi còn không bao giờ có lo lắng về mọi thứ thực sự đã làm việc như thế nào vì chúng tôi luôn có thể đi đọc mã nguồn".

Qua thời gian, Unix cũng trở thành một ví dụ sớm về các vấn đề mà có thể nảy sinh từ các hệ thống sở hữu độc quyền mà ban đầu được coi là "đã có một số tính năng của PMTD". Về cuối những năm 1970 và đặc biệt trong thập kỷ 1980, AT&T đã thay đổi chính sách của mình và sự truy cập tới các phiên bản mới của Unix trở nên khó khăn và đắt đỏ. Triết lý của những năm đầu mà đã làm cho Unix quá là phổ biến trong các lập trình viên, đã thay đổi hoàn toàn ở một mức độ mà trong năm 1991 AT&T đã còn cố gắng kiện Đại học Berkeley vì việc xuất bản mã nguồn Unix BSD mà CSRG của Berkeley đã tạo ra. Nhưng đây là một câu chuyện khác mà chúng tôi sẽ kể sau.

2.2 Sự bắt đầu: BSD, GNU

Tất cả mọi trường hợp được bàn luận trong phần trên từng hoặc là những sáng kiến riêng lẻ hoặc đã không tuân thủ nghiêm ngặt với những yêu cầu của PMTD. Điều này đã không xảy ra cho tới đầu những năm 1980 mà các dự án có tổ chức và có ý thức đầu tiên để tạo ra những hệ thống chứa PMTD đã xuất hiện. Trong giai đoạn này, những nền tảng về đạo đức, pháp lý và ngay cả tài chính của các dự án này đã bắt đầu được thiết lập (có lẽ còn quan trọng hơn), với chúng đang được phát triển và hoàn thiện cho tới tận ngày hôm nay. Và vì hiện tượng mới này đã cần tới một cái tên, điều đã xảy ra khi khái niệm PMTD lần đầu tiên sinh ra.

2.2.1 Richard Stallman, GNU, FSF: phong trào PMTD ra đời

Đầu năm 1984, Richard Stallman, người khi đó được thuê bởi Phòng thí nghiệm AI của MIT, đã rời bỏ công việc của ông để bắt đầu làm việc về dự án GNU. Stallman tự coi bản thân là một cao thủ máy tính mà sung sướng chia sẻ những mối quan tâm về công nghệ và mã nguồn của ông. Ông đã không thích cái cách mà ông từ chối ký các hợp đồng độc quyền và không chia sẻ đã làm cho ông trở thành một kẻ bị ruồng bỏ trong thế giới riêng của ông, và cách sử dụng PMSHĐQ trong môi trường của ông đã làm cho ông bất lực đối mặt với tình trạng mà có thể dễ dàng được giải quyết trước đó.

Ý tưởng của ông khi ông rời bỏ MIT là để xây dựng một hệ điều hành hoàn chỉnh, để sử dụng chung, mà hoàn toàn tự do ("Dự án GNU", DiBona et al.) [208]. Hệ thống này (và dự án mà có thể có trách nhiệm trong việc tạo ra nó đã trở thành đúng đắn) đã được gọi là GNU ("GNU không phải là Unix", một từ viết tắt đệ qui). Dù ngay từ đầu dự án GNU đã đưa các phần mềm vào trong hệ thống của mình mà nó đã sẵn sàng (như TeX hoặc, sau này, hệ X Window), thì đã còn có nhiều thứ phải xây dựng. Richard Stallman đã bắt đầu bằng việc viết một trình biên dịch C (GCC) và một trình soạn thảo (Emacs), cả 2 đều vẫn còn được sử dụng ngày nay (và rất phổ biến).

Ngay từ đầu dự án GNU, Richard Stallman đã quan tâm về sự tự do mà những người sử dụng các phần mềm này có thể có. Ông đã muốn không chỉ những người đã nhận được các chương trình trực tiếp từ dự án GNU tiếp tục tận hưởng y hệt những quyền (sửa đổi, phân phối lại, …), mà cả những người mà nhận được nó sau bất kỳ số lượng phân phối lại nào và sửa đổi (có thể) nào. Vì lý do này ông đã phác thảo ra giấy phép GPL, có lẽ là giấy phép PMTD đầu tiên được thiết kế đặc biệt để đảm bảo rằng một chương trình có thể là tự do theo cách này.

Richard Stallman đã gọi cơ chế chung mà các giấy phép dạng GPL này sử dụng để đạt được những đảm bảo này, copyleft, mà nó tiếp tục sẽ trở thành cái tên của một họ lớn các giấy phép của PMTD (FSF, Giấy phép Công cộng Chung GNU, phiên bản 2, tháng 06/1991) [118].

Richard Stallman cũng đã sáng lập ra FSF để có được các đầu tư vốn, mà ông sử dụng để phát triển và bảo vệ PMTD, và đã thiết lập các nguyên tắc đạo đức của ông với "Tuyên ngôn GNU" (FSF, 1985) [117] và "Vì sao phần mềm phải không có chủ sở hữu" (Richard Stallman, 1998) [207].

Từ quan điểm kỹ thuật, dự án GNU được nhận thức như một nỗ lực có cấu trúc cao với các mục tiêu rất

rõ ràng. Phương pháp thường được dựa vào các nhóm người khá nhỏ (thường là những người tự nguyện) phát triển một trong những công cụ mà nó có thể sau đó phù hợp tuyệt vời trong một trò chơi ghép hình hoàn chỉnh (hệ GNU). Tính được phân thành các module của Unix, trong đó dự án này đã được truyền cảm hứng, trùng hợp hoàn toàn với ý tưởng đó. Phương pháp làm việc này thường mặc nhiên sử dụng Internet, nhưng vì vào lúc đó nó còn chưa được thâm nhập vào một cách tuyệt đối, nên FSF cũng còn bán được các băng từ trên đó nó ghi lại các ứng dụng, mà có nghĩa là nó đã có thể là một trong những tổ chức đầu tiên có được sự cân đối tài chính (dù theo một cách khá hạn chế) từ việc tạo ra PMTD. Vào đầu những năm 90, khoảng 6 năm sau khi dự án này được hình thành, GNU đã rất gần đạt tới việc có một hệ thống hoàn chỉnh tương tự như Unix. Tuy nhiên, tại thời điểm đó nó còn chưa sản xuất ra được một trong những phần chủ chốt: nhân của hệ điều hành (còn được biết tới như là nhân kernel), một phần của hệ điều hành mà nó điều khiển các phần cứng, chiết tách nó, và cho phép các ứng dụng chia sẻ tài nguyên, và cơ bản là chạy được).

Tuy nhiên, phần mềm GNU đã là rất phổ biến trong những người sử dụng một số biến thể khác nhau của Unix, vào lúc đó hầu hết các hệ điều hành được sử dụng phổ biến trong các doanh nghiệp. Hơn nữa, dự án GNU đã trở thành khá nổi tiếng trong số những người chuyên nghiệp về IT, và đặc biệt là trong số những người làm việc tại các trường đại học. Trong giai đoạn đó, các sản phẩm của nó đã có được một uy tín xứng đáng cho tính ổn định và chất lượng tốt.

2.2.2 CSRG của Berkeley

Từ năm 1973, Nhóm Nghiên cứu Khoa học Máy tính CSRG (Computer Science Research Group) của Đại học California tại Berkeley đã là một trong những trung tâm nơi mà hầu hết các phát triển liên quan tới Unix đã được làm ra, đặc biệt giữa những năm 1979-1980. Không chỉ những ứng dụng được đưa ra và những ứng dụng mới khác được xây dựng để chạy trên Unix, mà còn cả những cải tiến quan trọng đã được làm cho nhân và nhiều chức năng đã được bổ sung. Ví dụ, trong những năm 80, một vài hợp đồng của DARPA (của Bộ Quốc phòng Mỹ) đã cấp tiền cho việc triển khai cài đặt TCP/IP mà cho tới hôm nay đã được xem là tham chiếu cho các giao thức mà chúng tạo ra công việc của Internet (trong quá trình, việc liên kết sự phát triển của Internet và sự mở rộng của các máy trạm Unix). Nhiều công ty đã sử dụng các phát triển của CSRG như những nền tảng cho các phiên bản Unix của họ để tạo ra những hệ thống nổi tiếng khi đó, như SunOS (Sun Microsystems) hoặc Ultrix (Digital Equipment). Điều này giải thích tại sao Berkeley đã trở thành một trong 2 nguồn Unix cơ bản, cùng với một nơi "chính hiệu" nữa là AT&T.

Để sử dụng được tất cả các mã nguồn mà CSRG đã tạo ra (và mã nguồn của những người cộng tác của cộng đồng Unix này mà trong một chừng mực nào đó họ đã phối hợp), cần thiết phải có giấy phép Unix của AT&T, mà nó ngày càng trở nên khó khăn (và đắt giá) để có được, đặc biệt nếu sự truy cập tới mã nguồn của hệ thống được yêu cầu. Một phần theo một dự định để vượt qua được vấn đề này, vào tháng 06/1989 CSRG đã tung ra phần Unix có liên quan tới TCP/IP (triển khai cài đặt của các giao thức trong nhân và các tiện ích), mà nó đã không bao gồm các mã nguồn của AT&T. Nó đã được gọi là *Phiên bản Mạng 1* (Net-1). Giấy phép mà nó được đưa ra là *giấy phép nổi tiếng BSD*, giấy phép này loại bỏ một số vấn đề với các mệnh đề của nó về các bổn phận quảng cáo, đã luôn được xem là một ví dụ về một giấy phép PMTD của những người thiểu số (mà nó bổ sung việc cho phép phân phối lại, cũng cho phép

kết hợp vào các sản phẩm của sở hữu độc quyền). Hơn nữa CSRG đã thử nghiệm một mô hình tài chính mới (mà FSF đã thí điểm thành công); nó đã bán các đầu băng với phát tán của mình với giá mỗi băng 1,000 USD. Dù thực tế là bất kỳ ai tới lượt mình cũng có thể phân phối lại được nội dung của các cuốn băng cho hàng ngàn tổ chức để có được tiền để tiếp tục phát triển.

Đã chứng kiến sự thành công của phát tán Net-1, Keith Bostic đã đề xuất để viết lại tất cả các mã nguồn mà vẫn còn giữ được từ Unix gốc của AT&T. Dù có sự nghi ngờ của một số thành viên của CSRG, thì ông vẫn đã tuyên bố công khai yêu cầu trợ giúp để hoàn thành nhiệm vụ này, và từng chút một các tiện ích (được viết lại trên cơ sở của các đặc tả kỹ thuật) đã trở nên tích hợp được vào trong hệ thống của Berkeley. Trong khi đó, quá trình y như vậy đã được thực hiện với nhân, theo đúng cái cách mà hầu hết các mã nguồn đã được sản xuất bởi Berkeley hoặc các cộng tác viên tình nguyện đã được viết một cách độc lập. Vào tháng 06/1991, sau khi có được phép từ cơ quan quản lý là Đại học Berkeley thì *Phiên bản Mạng 2* (Net-2) đã được tung ra, với hầu như tất cả mã nguồn của nhân và tất cả các tiện ích của một hệ điều hành Unix hoàn chỉnh.

Bộ này một lần nữa đã được phân phối theo giấy phép BSD và hàng ngàn các đầu băng đã được bán với giá mỗi băng 1,000 USD. Chỉ 6 tháng sau khi tung ra Net-2, Bill Jolitz đã viết mã nguồn mà đã bị mất đối với nhân để hoạt động trên kiến trúc i386, tung ra 386BSD, mà nó đã được phân phối qua Internet. Trên cơ sở của mã nguồn này sau đó đã nổi lên, như một biên dịch của các bản vá mà đã được đóng góp qua Net để cải tiến 386BSD; sau này FreeBSD đã xuất hiện, như một nỗ lực để tập trung vào hỗ trợ kiến trúc i386; một vài năm sau đó dự án OpenBSD đã được hình thành, với một sự nhấn mạnh vào an ninh. Và cũng đã có một phiên bản sở hữu độc quyền dựa trên Net-2 (dù nó chắc chắn là bản gốc, vì nó đã chào cho các khách hàng của nó tất cả các mã nguồn như một phần của phát tán cơ bản này), mà đã được thực hiện một cách độc lập bởi một công ty tới nay đã chết BSDI (Berkeley Software Design Inc.).

Phần vì như một phản ứng đối với phát tán được làm ra bởi BSDI, bộ phận phụ của AT&T mà đã giữ các quyền của giấy phép Unix này, các Phòng thí nghiệm Hệ thống Unix USL (Unix System Laboratories), đã cố kiện BSDI đầu tiên và sau đó là Đại học California. Việc tố cáo là việc BSDI đã phân phối sở hữu trí tuệ của AT&T mà không có phép. Sau một loạt vận động pháp lý (mà bao gồm cả một vụ kiện ngược của Đại học California chống lại USL), thì Novell đã mua các quyền Unix từ USL, và vào tháng 01/1994 đã đạt được một vụ dàn xếp ngoài tòa án với Đại học California. Như là kết quả của sự dàn xếp này, CSRG đã phân phối phiên bản 4.4BSD-Lite, mà nó đã sớm được sử dụng bởi tất cả các dự án của họ *BSD. Ngay sau đó (sau khi tung ra phiên bản 4.4BSD-Lite Phiên bản 2), thì CSRG đã biến mất. Tại thời điểm đó, một số đã sợ rằng nó có thể sẽ là sự kết thúc của các hệ thống *BSD, nhưng thời gian đã chỉ ra rằng chúng vẫn còn sống và biến theo một dạng quản lý mới mà đặc thù hơn đối với các dự án PMTD. Ngay cả trong thập kỷ đầu tiên của năm 2000 các dự án được quản lý bởi họ *BSD là những dự án lâu đời nhất và vững chắc nhất trong thế giới PMTD.

Thư mục tham khảo

Lịch sử của Unix BSD là bức tranh minh họa một cách khác thường về phát triển phần mềm trong thời kỳ những năm 1970 và 1980. Bất kỳ ai quan tâm tới nó có thể thưởng thức đọc "Hai mươi năm Berkeley Unix" (Marshall Kirk McKusick, 1999) [170], mà nó đi theo sự tiến hóa từ đầu băng mà Bob Fabry đã lấy cho Berkeley với ý tưởng làm ra một trong những phiên bản đầu tiên các mã nguồn của Thompson và Ritchie hoạt động trên một PDP-11 (được mua cùng bởi các khoa toán, thống kê và thông tin), qua tới

các vụ kiện được đệ trình bởi AT&T và những phiên bản mã nguồn mới nhất mà chúng đã tạo ra họ các hệ điều hành *BSD.

2.2.3 Sự khởi đầu của Internet

Hầu hết từ sự tạo ra của nó trong thập kỷ những năm 1970, Internet đã từng gắn chặt với PMTD. Một mặt, ngay từ đầu, cộng đồng các lập trình viên mà họ xây dựng Internet đã có vài nguyên tắc rõ ràng mà chúng sau này đã trở thành kinh điển trong thế giới của PMTD; ví dụ, tầm quan trọng của người sử dụng có khả năng giúp sửa các lỗi hoặc chia sẻ mã nguồn. Tầm quan trọng của BSD Unix trong sự phát triển của nó (bằng việc cung cấp trong những năm 1980 hầu hết sự triển khai cài đặt phổ biến của các giao thức TCP/IP) làm cho nó dễ dàng truyền đi nhiều tập quán và cách thức của việc làm ra những thứ từ một cộng đồng - các lập trình viên đã tập trung xung quanh CSRG - đối với cộng đồng khác - các lập trình viên mà đã từng xây dựng những gì khi đó là NSFNet và có thể sau này đã trở thành Internet - và ngược lại. Nhiều ứng dụng cơ bản cho sự phát triển Internet, như Sendmail (máy chủ thư điện tử) hoặc BIND (triển khai cài đặt của các dịch vụ tên miền) đã là tự do và, ở một mức độ nào đó, là kết quả của sự hợp tác giữa 2 cộng đồng này.

Cuối cùng, về cuối những năm 80 và trong thập niên 90, cộng đồng PMTD từng là một trong những cộng đồng đầu tiên khai thác sâu những khả năng được đưa ra bởi Internet cho việc hợp tác của các nhóm phân tán về mặt địa lý. Ở một mức độ nào đó, thì sự khai thác này đã tạo ra sự tồn tại có thể của cộng đồng BSD, FSF hoặc sự phát triển của GNU/Linux.

Một trong những khía cạnh thú vị nhất của sự phát triển Internet, từ quan điểm của PMTD, là sự quản lý mở hoàn toàn các tài liệu và qui định của nó. Dù nó có thể được xem là bình thường ngày hôm nay (vì nó là quen rồi, ví dụ, trong IETF hoặc nhóm World Wide Web), lúc này, tính sẵn sàng tự do của tất cả các đặc tả kỹ thuật của nó, và các tài liệu thiết kế bao gồm các chuẩn mà xác định các giao thức, là thứ gì đó cách mạng và cơ bản đối với sự phát triển của nó. Trong những năm 90. *Trong lịch sử và ảnh hưởng của Usenet và Internet* [139] (trang 106) chúng ta có thể đọc:

> "Quá trình mở này đã xảy ra và dẫn tới sự thay đổi về thông tin. Sự phát triển kỹ thuật chỉ thành công khi thông tin được phép chảy một cách tự do và dễ dàng giữa các bên liên quan. Việc khuyến khích sự tham gia là nguyên tắc chính đã tạo ra sự phát triển của Net có thể được".

Chúng ta có thể thấy vì sao đoạn này có thể hầu như được hỗ trợ một cách chắc chắn bởi bất kỳ lập trình viên nào tham chiếu tới dự án PMTD trong đó anh ta có liên quan.

Trong một trích dẫn khác, về "Sự tiến hóa của chuyển mạch gói" [195] (trang 267) chúng ta có thể đọc:

> "Vì ARPANET từng là một dự án công cộng kết nối nhiều trường đại học và viện nghiên cứu chính, nên sự triển khai cài đặt và các chi tiết về tốc độ thực thi đã được công bố một cách rộng rãi".

Rõ ràng, đây là những gì có xu hướng xảy ra với các dự án PMTD, nơi mà tất cả các thông tin có liên quan tới một dự án (và không chỉ tới sự triển khai cài đặt của nó) thường là công khai.

Trong bối cảnh này, và trước khi Internet, vâng trong những năm 90, trở thành một việc kinh doanh hoàn toàn, thì cộng đồng những người sử dụng và mối quan hệ của nó với các lập trình viên đã là mang

tính sống còn. Trong giai đoạn này nhiều tổ chức đã học được để tin tưởng không chỉ một nhà cung cấp đơn nhất các dịch vụ giao tiếp dữ liệu, mà còn là một sự kết hợp phức tạp của các công ty dịch vụ, các nhà sản xuất thiết bị, các lập trình viên chuyên nghiệp, và các tình nguyện viên, … Những triển khai cài đặt tốt nhất của nhiều chương trình đã không phải là những thứ mà tới với hệ điều hành được mua cùng với phần cứng, mà là những triển khai cài đặt tự do mà có thể nhanh chóng thay thế chúng. Những phát triển đổi mới sáng tạo nhất đã không xuất phát từ các kế hoạch nghiên cứu của các công ty lớn mà là sản phẩm của các sinh viên hoặc những người chuyên nghiệp mà họ đã thử nghiệm các ý tưởng và đã thu thập những ý kiến phản hồi cho họ bởi hàng loạt những người sử dụng các chương trình tự do của họ.

Như chúng tôi đã nhắc tới ở trên, Internet cũng đã đưa ra cho PMTD các công cụ nền tảng cho sự hợp tác từ xa. Thư điện tử, nhóm tin, các dịch vụ truyền tệp FTP nặc danh (mà từng là những nơi lưu trữ khổng lồ đầu tiên của PMTD) và, cuối cùng, các hệ thống phát triển tích hợp dựa trên web đã từng là cơ bản (và không thể thiếu được) cho sự phát triển của cộng đồng PMTD như chúng ta biết nó hôm nay, và đặc biệt, cho việc hoạt động của đa số lớn các dự án PMTD. Ngay từ đầu, các dự án như là GNU hoặc BSD đã tạo ra sự sử dụng đông đảo và mạnh mẽ của tất cả các cơ chế này, việc phát triển, cùng lúc như họ sử dụng chúng, các công cụ và hệ thống mới mà tới lượt chúng đã cải thiện cho Internet.

Thư mục tham khảo

Các độc giả có quan tâm trong sự tiến bộ của Internet, được viết bởi vài người giữ vai trò chủ đạo chính của nó, có thể tự xem "Lịch sử ngắn gọn về Internet" (được xuất bản bởi ACM, 1997) [166].

2.2.4 Các dự án khác

Trong những năm 1980 nhiều dự án PMTD quan trọng khác đã thấy được ánh sáng ban ngày. Chúng tôi nhấn mạnh về tầm quan trọng và sự thích hợp trong tương lai của chúng, X Window (hệ thống cửa sổ cho các hệ điều hành dạng Unix), được phát triển tại MIT, một trong những ví dụ đầu tiên về việc đầu tư tài chính phạm vi rộng cho một dự án tự do được cấp tiền bởi một nhóm các doanh nghiệp. Cũng đáng để nhắc tới là Ghostscript, một hệ thống quản lý tài liệu PostScript được phát triển bởi một công ty được gọi là Aladdin Software, mà nó đã là một trong những trường hợp đầu tiên về việc tìm kiếm một mô hình kinh doanh dựa trên việc sản xuất PMTD.

Về cuối những năm 1980, đã còn có một chòm sao toàn là những dự án PMTD nhỏ (và không nhỏ lắm) được thực hiện. Tất cả chúng, cùng với những dự án lớn mà chúng tôi đã nhắc tới cho tới nay, đã thiết lập nên những cơ sở của các hệ thống hoàn toàn tự do đầu tiên, mà chúng đã xuất hiện trong đầu những năm 1990.

2.3 Mọi thứ đều theo cách của nó

Khoảng những năm 1990, hầu hết các thành phần của một hệ thống hoàn chỉnh đã sẵn sàng như là PMTD. Một mặt, dự án GNU và các phát tán BSD đã hoàn chỉnh hầu hết các ứng dụng mà chúng tạo ra một hệ điều hành. Mặt khác, các dự án như X Window hoặc bản thân GNU đã xây dựng từ những

môi trường cửa sổ cho các trình biên dịch, mà chúng thường đã là trong số những thứ tốt nhất trong loại hạng của chúng (ví dụ, nhiều quản trị viên của các hệ thống SunOS hoặc Ultrix có thể thay thế các ứng dụng sở hữu độc quyền các hệ thống của họ cho các phiên bản tự do của GNU hoặc BSD cho những người sử dụng của họ). Để có một hệ thống hoàn toàn được xây dựng chỉ với PMTD, chỉ có một thành phần còn thiếu: nhân kernel. 2 nỗ lực độc lập và riêng rẽ này đã tới để điền đầy khoảng trống: 386BSD và Linux.

2.3.1 Yêu cầu về một nhân kernel

Về cuối những năm 1980 và bắt đầu những năm 1990, dự án GNU đã có một loạt cơ sở tiện ích và công cụ có khả năng tạo ra một hệ điều hành hoàn chỉnh. Ngay cả khi đó, nhiều ứng dụng tự do, bao gồm cả trường hợp đặc biệt thú vị của X Window, đã là tốt nhất trong lĩnh vực của chúng (các tiện ích của Unix, các trình biên dịch...). Tuy nhiên, để hoàn tất trò chơi xếp hình thì một mảnh sống còn vẫn còn thiếu: nhân của hệ điều hành. Dự án GNU đã tìm kiếm mảnh còn thiếu đó bằng một dự án được biết tới là Hurd, mà nó định xây dựng một nhân có sử dụng các công nghệ tiên tiến.

2.3.2 Họ *BSD

Thực tế cùng một lúc, cộng đồng BSD cũng đã trên đường hướng tới một nhân tự do. Phát tán Net-2 đã chỉ thiếu có 6 tệp để hoàn tất nó (phần còn lại đã được xây dựng bởi CSRG hoặc các cộng tác viên của nó). Vào đầu năm 1992, Bill Jolitz đã hoàn tất những tệp này và đã phân phối 386BSD, một hệ thống mà nó đã hoạt động trên kiến trúc i386 và đúng lúc có để nâng tầm cho các dự án NetBSD, FreeBSD và OpenBSD. Sự tiến bộ trong các tháng tiếp sau là nhanh chóng, và vào cuối năm đó nó đã đủ ổn định để được sử dụng trong các môi trường sản xuất không mang tính sống còn, mà bao gồm, ví dụ, một môi trường cửa sổ nhờ vào dự án XFree (mà nó đã cung cấp X Window cho kiến trúc i386) hoặc một trình biên dịch chất lượng tuyệt vời, GCC. Dù vẫn còn những thành phần đã sử dụng các giấy phép khác (như những giấy phép từ các dự án GNU, mà chúng đã sử dụng GPL), hầu hết hệ thống này đã được phân phối theo giấy phép BSD.

Thư mục tham khảo

Một vài câu chuyện của giai đoạn này minh họa cho khả năng của mô hình phát triển PMTD. Có trường hợp nổi tiếng của Linus Torvalds, người đã phát triển Linux trong khi còn là một sinh viên năm thứ 2 của Đại học Helsinki. Nhưng đây không chỉ là trường hợp duy nhất một sinh viên mà đã tạo ra con đường của mình nhờ vào những phát triển tự do. Ví dụ, Thomas Roel đã chuyển X11R4 (một phiên bản của hệ thống X Window) sang một máy tính cá nhân PC dựa trên một vi xử lý 386. Sự phát triển này đã dẫn anh ta tới làm việc tại Dell, và sau này đã trở thành người sáng lập của các dự án X386 và Xfree, mà đã là những nền tảng cho việc nhanh chóng trao cho GNU/Linux và họ *BSD một môi trường cửa sổ. Bạn có thể đọc nhiều hơn về câu chuyện về XFree và vai trò của Roel trong “Lịch sử của xFree86” (Tạp chí Linux, tháng 12/1991) [135].

Rồi tới vụ kiện từ USL, mà nó đã làm cho nhiều người sử dụng tiềm năng sợ những vụ kiện chống lại họ trong trường hợp nếu Đại học California bị thua kiện hoặc đơn giản, rằng dự án đi tới chỗ bế tắc. Có lẽ điều này đã là một lý do vì sao sau này, nền tảng cài đặt của GNU/Linux tuyệt vời hơn nhiều so với

tất cả họ *BSD cộng lại. Nhưng chúng ta không thể biết điều này một cách chắc chắn.

2.3.3 Sự ra đời của GNU/Linux

Tháng 07/1991 Linus Tovalds (một sinh viên Phần Lan 21 tuổi) đã đưa ra thông điệp đầu tiên nhắc tới dự án của anh ta (khi đó) để xây dựng một hệ điều hành tự do tương tự như Minix. Vào tháng 09 anh ta đã đưa ra phiên bản đầu tiên nhất (0.01), và sau mỗi ít tuần các phiên bản mới đã xuất hiện. Vào tháng 03/1994 phiên bản 1.0 đã xuất hiện, lần đầu tiên một phiên bản được gọi là ổn định, dù nhân mà Linus đã xây dựng đã có thể sử dụng được từ vài tháng trước. Trong giai đoạn này, đúng là hàng trăm lập trình viên đã chuyển sang Linux, tích hợp tất cả các phần mềm của GNU xung quanh nó, cũng như XFree và nhiều chương trình tự do khác nữa. Không giống như họ *BSD, nhân Linux và một số lượng lớn các thành phần được tích hợp xung quanh nó đã được phân phối bằng giấy phép GPL.

Thư mục tham khảo

Câu chuyện về Linux có lẽ là một trong những câu chuyện thú vị nhất (và nổi tiếng) trên thế giới của PMTD. Bạn có thể thấy nhiều đường liên kết tới các thông tin về nó từ những trang đánh dấu 10 năm kỷ niệm tuyên bố của nó, dù có lẽ một trong những câu chuyện thú vị nhất là "Lịch sử của Linux", của Ragib Hasan [138]. Vì tò mò, bạn có thể xem dòng trong đó Linus Torvalds đã công bố rằng ông đã bắt đầu tạo ra những gì sau này trở thành Linux (trên nhóm thảo luận comp.os.minix) tại http://groups.google.com/groups?th=d161e94858c4c0b9. Ở đó ông giải thích cách mà ông đã từng làm việc trên nhân của mình từ tháng 04 và cách mà ông đã đưa một số công cụ của dự án GNU lên nó (đặc biệt nhắc tới Bash và GCC).

Về nhiều phát triển đã nổi lên xung quanh Linux, một trong những câu chuyện thú vị nhất là khái niệm *phát tán*[1]. Những phát tán đầu tiên đã sớm xuất hiện, trong năm 1992 (MCC Interim Linux, của Đại học Manchester; TAMU, của Texas A&M, và nổi tiếng nhất, SLS, mà sau này làm nổi tiếng cho Slackware, mà vẫn còn được phân phối trong thập niên đầu tiên của năm 2000), gây ra sự cạnh tranh trong thế giới các hệ điều hành đóng gói xung quanh Linux. Mỗi phát tán cố gắng đưa ra một GNU/Linux sẵn sàng để sử dụng, và bắt đầu từ nền tảng của cùng các phần mềm phải cạnh tranh được bằng việc tạo ra những cải tiến được cho là quan trọng bởi nền tảng người sử dụng của họ. Hơn nữa để cung cấp các gói sẵn sàng sử dụng được đã được biên dịch trước, thì các phát tán cũng có xu hướng đưa ra các công cụ của riêng họ cho việc quản lý sự lựa chọn, cài đặt, thay thế và bỏ cài đặt các gói này, bổ sung vào việc cài đặt ban đầu lên máy tính, cùng với sự quản lý và quản trị hệ điều hành.

Qua thời gian, các phát tán đã kế tiếp nhau như những phát tán khác nhau đã trở thành nổi tiếng nhất. Trong số tất cả chúng, chúng tôi có thể nhấn mạnh những phát tán sau:

1. Debian, được phát triển bởi một cộng đồng những người sử dụng tự nguyện.
2. Red Hat Linux, mà lần đầu tiên được phát triển nội bộ bên trong công ty Red Hat, nhưng sau đó đã áp dụng một mô hình dựa trên cộng đồng, làm nổi cho Fedora Core.
3. SuSE, mà làm nổi cho OpenSuSE, sau một sự tiến bộ tương tự như của Red Hat.

1 Khái niệm này được giải thích chi tiết trong bài viết tương ứng trong Wikipedia, www.wikipedia.org/wiki/Linux_distribution

4. Mandriva, (hậu duệ của Mandrake Linux và Conectiva).
5. Ubuntu, bắt nguồn từ Debian và được phát triển trên cơ sở của Debian bởi hãng Canonical.

2.4 Thời gian chín muồi

Nửa những năm đầu thập kỷ 2000, GNU/Linux, OpenOffice.org hoặc Firefox đã được trình diễn trên các phương tiện rất thường xuyên. Đa số áp đảo các công ty sử dụng PMTD cho ít nhất một số qui trình IT của họ. Khó mà thành một sinh viên IT mà không sử dụng một số lượng lớn các PMTD. PMTD không còn là một lời chú ở cuối trang trong lịch sử IT và đã trở thành thứ gì đó rất quan trọng cho khu vực này. Các công ty IT, các công ty trong khu vực thứ cấp (những công ty mà sử dụng phần mềm một cách tích cực, ngay cả dù hoạt động hàng đầu của họ là khác) và các nền hành chính nhà nước đang bắt đầu coi nó là thứ gì đó mang tính chiến lược.

Và dù chậm nhưng chắc chắn nó đang đi tới những người sử dụng gia đình. Ở nghĩa rộng lớn hơn, chúng ta đang đi vào một giai đoạn chín muồi.

Và trong sâu thẳm của tất cả những thứ này, một câu hỏi quan trọng bắt đầu nảy sinh, mà nó đúc kết theo một cách mà những gì đang xảy ra: "liệu chúng ta có đang đối mặt với một mô hình mới của nền công nghiệp phần mềm hay không?". Có lẽ, nó có thể xảy ra việc PMTD trở nên không còn là một xu hướng thoảng qua để được nhớ tới một cách vấn vương một ngày nào đó. Mà nó có thể cũng sẽ là một mô hình mới (và điều này hình như đang gia tăng) mà tồn tại ở đây, và có lẽ sẽ thay đổi một trong những nền công nghiệp non trẻ nhất nhưng cũng gây ảnh hưởng nhất của thời đại chúng ta.

2.4.1 Kết thúc những năm 90

Vào giữa những năm 1990, PMTD đã đưa ra những môi trường hoàn chỉnh (các phát tán của GNU/Linux, các hệ thống *BSD...) mà chúng đã hỗ trợ công việc hàng ngày của nhiều người, đặc biệt là các lập trình viên phần mềm. Vẫn còn nhiều việc còn chưa được giải quyết xong (chủ yếu là để có các giao diện đồ họa cho người sử dụng tốt hơn tại thời điểm khi mà Windows 95 đã được coi là chuẩn), nhưng đã có vài ngàn người trên thế giới đã sử dụng hoàn toàn các PMTD cho công việc hàng ngày của họ. Các dự án mới đã được công bố theo sự kế tục liên tục và PMTD đã cập bến được trên con đường dài hướng tới nhận thức nói chung của các công ty, phương tiện truyền thông và của công chúng.

Giai đoạn này cũng liên quan tới việc Internet cất cánh như một mạng cho mọi người, trong nhiều trường hợp đã được dẫn dắt bởi bàn tay của các chương trình tự do (đặc biệt trong nền tảng của nó). Sự ập tới của mạng vào trong các ngôi nhà của hàng triệu người sử dụng đầu cuối đã tăng cường cho tình trạng này, ít nhất về phương diện các máy chủ: hầu hết các máy chủ web (HTTP) đã luôn là tự do (đầu tiên là máy chủ NCSA, tiếp theo là Apache). Có lẽ khởi đầu con đường cho PMTD cho tới phiên bản đầy đủ trong công chúng được mô tả tốt nhất trong bài viết nổi tiếng của Eric Raymond, "Nhà thờ lớn và cái chợ" (Eric S. Raymond, 2001) [192]. Dù nhiều thứ trong đó được mô tả đã là nổi tiếng trong cộng đồng các lập trình viên PMTD, thì việc đặt nó vào trong bài viết và phân phối nó một cách tích

cực đã làm cho nó trở thành một công cụ gây ảnh hưởng cho việc khuyến khích khái niệm PMTD như một cơ chế phát triển thay thế cho một cơ chế được sử dụng bởi nền công nghiệp phần mềm truyền thống.

Bài viết quan trọng khác trong thời kỳ này là "Việc thiết lập cửa hàng. Kinh doanh PMNM" [141] của Frank Hecker, mà lần đầu tiên đã mô tả các mô hình kinh doanh tiềm năng cho PMTD, và nó đã được viết để gây ảnh hưởng tới quyết định đưa ra mã nguồn của Netscape Navigator.

Trong khi bài viết của Raymond từng là một công cụ tuyệt vời cho việc khuyến khích một số đặc tính cơ bản của PMTD, thì sự tung ra mã nguồn của Netscape Navigator từng là trường hợp đầu tiên trong đó một công ty khá lớn, trong một lĩnh vực rất đổi mới sáng tạo (sau đó nền công nghiệp web mới xuất hiện) đã quyết định đưa một trong những sản phẩm của mình ra thành PMTD. Vào lúc đó, Netscape Navigator đã thua cuộc chiến trình duyệt chống lại sản phẩm của Microsoft (Internet Explorer), một phần vì các chiến thuật của Microsoft kết hợp nó với hệ điều hành của hãng. Nhiều người tin tưởng rằng Netscape đã chỉ làm mỗi thứ mà nó có thể đã hoàn thành: cố gắng thay đổi những luật lệ để có khả năng cạnh tranh với một người khổng lồ. Và từ sự thay đổi này trong các luật lệ (cố gắng để cạnh tranh bằng một mô hình PMTD) mà dự án Mozilla đã được sinh ra. Dự án này, mà từng có những vấn đề của riêng nó, đã dẫn tới vài năm sau đó trở thành trình duyệt mà, dù nó còn chưa bao phủ được thị phần khổng lồ mà Netscape đã từng có trong những năm tháng của mình, thì dường như về mặt kỹ thuật ít nhất nó cũng tốt như của các đối thủ cạnh tranh sở hữu độc quyền.

Trong mọi trường hợp, bất chấp sự thành công sau này của nó, thì tuyên bố của Netscape rằng hãng có thể đưa ra mã nguồn của trình duyệt của hãng đã có một ảnh hưởng to lớn lên nền công nghiệp phần mềm. Nhiều công ty đã bắt đầu coi PMTD đáng để xem xét. Các thị trường tài chính cũng đã bắt đầu chú ý tới PMTD. Trong tình trạng của vụ nổ dotcom, nhiều công ty PMTD đã trở thành các mục tiêu cho các nhà đầu tư. Có lẽ trường hợp nổi tiếng nhất là việc Red Hat, một trong những công ty đầu tiên nhận thức được rằng việc bán các đĩa CD với các hệ điều hành GNU/Linux sẵn sàng để sử dụng có thể trở thành một mô hình kinh doanh tiềm năng.

Red Hat đã bắt đầu phân phối Red Hat Linux của hãng, với sự nhấn mạnh khổng lồ (ít nhất cho những gì là chung tại thời điểm đó) vào sự dễ dàng sử dụng và dễ dàng duy tu bảo trì cho mọi người mà không cần một nền tảng IT đặc biệt nào. Qua thời gian nó đã đa dạng hóa, giữ được trong quỹ đạo của PMTD, và vào tháng 09/1998 hãng đã công bố rằng Intel và Netscape đã đầu tư vào hãng. "Nếu nó tốt cho Intel và Netscape, thì nó phải tốt cho chúng tôi", là những gì nhiều nhà đầu tư đã phải suy nghĩ sau đó. Khi Red Hat đã ra công chúng vào mùa hè năm 1999, thì IPO đã được thuê bao hoàn toàn và giá trị của mỗi cổ phiếu sớm gia tăng một cách ngoạn mục. Đây là lần đầu tiên mà một công ty đã có được việc cấp tài chính từ thị trường chứng khoán với một mô hình dựa vào PMTD. Nhưng đây không phải là trường hợp duy nhất: sau đó, những hãng khác như VA Linux hay Andover.net (mà sau này đã bị mua bởi VA Linux) đã làm y như vậy.

Lưu ý

Red Hat cung cấp một danh sách các cột mốc của hãng tại http://fedora.redhat.com/about/history/.

Trong giai đoạn này, nhiều công ty cũng đã được sinh ra với các mô hình kinh doanh dựa trên PMTD. Dù không ra công chúng hoặc đạt được đỉnh thị trường khủng khiếp như vậy, thì họ cũng rất quan trọng

cho sự phát triển của PMTD. Ví dụ, nhiều công ty đã xuất hiện mà đã bắt đầu phân phối các phiên bản GNU/Linux của họ, như SuSE (Đức), Conectiva (Brazil) hoặc Mandrake (Pháp), mà nó có thể sau này tham gia vào Conectiva để tạo ra Mandriva. Những công ty khác đã đưa ra các dịch vụ cho các công ty mà họ muốn duy trì hoặc áp dụng các sản phẩm tự do: LinuxCare (Mỹ), Alcove (Pháp), ID Pro (Đức), và nhiều hãng khác.

Trong khi đó, những người khổng lồ của khu vực này đã bắt đầu định vị trí cho bản thân họ trong mối quan hệ với PMTD. Một số công ty, như IBM, đã kết hợp nó một cách trực tiếp vào trong chiến lược của hãng. Những hãng khác, như Sun Microsystems, đã có một mối quan hệ kỳ cục với nó, lúc thì hỗ trợ nó, lúc thì lại khác, khi chống lại nó. Hầu hết các công ty (như Apple, Oracle, HP, SGI, …) đã khai thác mô hình của PMTD với các chiến lược khác nhau, từ việc giải phóng có chọn lọc các phần mềm tới việc đưa ra một cách trực tiếp các sản phẩm của họ cho GNU/Linux. Giữa 2 thái cực đó đã có nhiều con đường hành động khác, như sử dụng tăng cường ít nhiều các PMTD trong các sản phẩm của họ (như trường hợp với Mac OS X) hoặc khai thác các mô hình kinh doanh dựa trên việc duy tu bảo trì các sản phẩm PMTD.

Từ quan điểm kỹ thuật, hầu hết sự kiện đáng nhớ của giai đoạn này có lẽ là sự xuất hiện của 2 dự án có tham vọng được thiết kế để đưa PMTD tới môi trường *máy tính để bàn* cho những người sử dụng IT không có kinh nghiệm: KDE và GNOME. Đơn giản mà nói, mục tiêu cuối cùng không phải là phải sử dụng dòng lệnh để tương tác với GNU/Linux hoặc *BSD hoặc với các chương trình trong các môi trường đó.

KDE đã được công bố vào tháng 10/1996. Việc sử dụng các thư viện đồ họa Qt (lúc đó là một sản phẩm sở hữu độc quyền của hãng Trolltech, nhưng là miễn phí để sử dụng trên GNU/Linux[2]), mà sự xây dựng đã bắt đầu một tập hợp các ứng dụng máy tính để bàn mà chúng có thể làm việc được theo một cách tích hợp và có một giao diện thống nhất. Vào tháng 07/1998 phiên bản 1.0 của Môi trường Để bàn K (K Desktop Environment) đã được tung ra, và đã sớm được tiếp tục bởi các phiên bản mới ngày càng hoàn chỉnh hơn và chín muồi hơn. Các phát tán GNU/Linux sớm kết hợp KDE vào như một màn hình giao diện cho những người sử dụng của họ (hoặc ít nhất như một môi trường máy tính để bàn mà người sử dụng có thể chọn).

Hầu như là một phản ứng đối với sự phụ thuộc của KDE vào thư viện sở hữu độc quyền của Qt, vào tháng 08/1997 dự án GNOME đã được công bố (Miguel de Icaza, "Lịch sử của dự án GNOME") [101], với các mục đích và đặc tính tương tự đối với KDE, nhưng tuyên bố mục tiêu chắc chắn về tất cả các thành phần của nó là PMTD. Vào tháng 02/1999, GNOME 1.0 đã được tung ra, mà nó cũng được cải thiện và ổn định theo thời gian. Vào thời điểm đó, hầu hết các phát tán của các hệ điều hành tự do (và nhiều hệ điều hành sở hữu độc quyền có nguồn gốc Unix) đã đưa ra màn hình GNOME hoặc KDE như một lựa chọn, và các ứng dụng của cả 2 môi trường.

Trong khi đó, các dự án PMTD chính đang được thực hiện vẫn giữ được sự tốt lành với các dự án mới nổi lên hầu như mỗi ngày. Trong một loạt các thị trường thích hợp, PMTD đã được thấy là giải pháp tốt nhất (được thừa nhận hầu như khắp thế giới). Ví dụ, kể từ khi xuất hiện vào tháng 04/1995, Apache đã

2 Sau này, Qt đã bắt đầu được phân phối theo giấy phép tự do QPL (Qt Public License), không tương thích với GPL, mà nó tạo ra một số vấn đề, vì hầu hết của KDE được phân phối theo GPL. Đúng lúc, Trolltech cuối cùng đã quyết định phân phối Qt theo giấy phép GPL, kết thúc được các vấn đề này.

duy trì được thị phần lớn nhất cho các máy chủ web; Xfree86, dự án tự do mà nó phát triển X Window, cho tới nay vẫn là phiên bản nổi tiếng nhất của X Window (và vì thế, là hệ thống cửa sổ mở rộng nhất cho các hệ điều hành dạng Unix); GCC được thừa nhận như là trình biên dịch C khả chuyển được tốt nhất và là một thứ chất lượng cao nhất; GNAT, hệ thống biên soạn cho Ada 95, đã giành được phần tốt nhất của thị trường cho các trình biên dịch Ada chỉ trong ít năm, và hơn thế nữa.

Trong năm 1998, OSI đã được thành lập, nó đã quyết định áp dụng khái niệm PMNM như một thương hiệu cho việc giới thiệu PMTD vào thế giới các doanh nghiệp, trong khi tránh sự tối nghĩa về khái niệm *tự do* (mà có thể còn có nghĩa là cả tự do sử dụng và miễn phí). Quyết định này đã làm bùng lên một trong những tranh luận gay gắt nhất trong thế giới PMTD (mà nó vẫn tiếp tục cho tới ngày nay), khi mà FSF và những người khác đã cho rằng nó phù hợp hơn nhiều để nói về PMTD (Richard Stallman, "Vì sao *PMTD* là tốt hơn *nguồn mở*", 1998) [206]. Trong mọi trường hợp, OSI đã tiến hành một chiến dịch quảng bá tuyệt vời cho thương hiệu mới của mình, mà đã được áp dụng bằng nhiều cách thức được ưa chuộng hơn để nói về PMTD, đặc biệt cho thế giới nói tiếng Anh. Để định nghĩa *PMNM*, OSI đã sử dụng một định nghĩa có xuất xứ từ định nghĩa được sử dụng của dự án Debian để định nghĩa PMTD ("Những chỉ dẫn về PMTD của Debian "), http://www.debian.org/social_contract.html#guidelines) [104], mà cùng thời gian đó phản ánh một cách khá gần gũi ý tưởng của FSF về nó (Định nghĩa PMTD, http://www.gnu.org/philosophy/free-sw.html) [120], nghĩa là từ quan điểm thực tế hầu hết các chương trình được coi là PMTD có thể cũng được coi là PMNM và ngược lại. Tuy nhiên, các cộng đồng PMTD và PMNM (hoặc ít nhất những người mà xác định với chúng) có thể là rất khác nhau.

2.4.2 Thập niên 2000

Vào những năm đầu thập niên 2000, PMTD đã là một đối thủ cạnh tranh nghiêm túc trong phân khu các máy chủ và đã bắt đầu sẵn sàng cho máy tính để bàn. Các hệ thống như GNOME, KDE, OpenOffice.org và Mozilla Firefox có thể được sử dụng bởi những người sử dụng ở nhà và đủ cho các nhu cầu của nhiều công ty, ít nhất những nơi mà các ứng dụng được quan tâm. Các hệ thống tự do (và đặc biệt là các hệ thống dựa trên Linux) là dễ dàng cài đặt, và sự phức tạp về duy tu bảo trì và cập nhật chúng so sánh được với những thứ đó của các hệ thống sở hữu độc quyền khác.

Ngay bây giờ, mỗi công ty trong nền công nghiệp phần mềm đều có một chiến lược về PMTD. Hầu hết các tập đoàn đa quốc gia hàng đầu (IBM, HP, Sun, Novell, Apple, Oracle...) kết hợp PMTD ở một mức độ ít nhiều nào đó. Ở một cực độ chúng ta có thể thấy các công ty như Oracle, mà nó phản ứng đơn giản bằng việc đưa các sản phẩm của hãng lên GNU/Linux. Ở một cực khác, chúng ta có thể thấy IBM, mà hãng có hầu hết các chiến lược quyết định và đã thực hiện các chiến dịch công khai lớn nhất về GNU/Linux. Trong số những tập đoàn hàng đầu trong thị trường IT, chỉ có Microsoft đã định vị trí cho mình chống lại PMTD và đặc biệt phần mềm được phân phối theo giấy phép GPL một cách rõ ràng.

Về bản thân thế giới PMTD, bất chấp những tranh cãi mà chúng thỉnh thoảng khuấy động cộng đồng này, thì sự tăng trưởng của nó là khổng lồ. Mỗi ngày có nhiều thêm các lập trình viên, các dự án PMTD tích cực hơn, nhiều người sử dụng hơn, … Với mỗi ngày trôi qua PMTD lại đang chuyển khỏi những con đường phụ và trở thành một lực lượng sẽ phải được tính tới.

Theo ánh sáng của điều này, những nguyên lý mới đang nổi lên mà chúng đặc biệt nghiên cứu PMTD,

như thiết kế kỹ thuật của PMTD. Dựa trên những nghiên cứu, từng chút một chúng ta đang bắt đầu hiểu PMTD hoạt động như thế nào trong hàng loạt các khía cạnh của nó: các mô hình phát triển, các động lực của các lập trình viên, …

Những năm này chúng ta cũng bắt đầu thấy những ảnh hưởng đầu tiên của *việc đưa thuê ngoài* mà sự phát triển của PMTD cho phép: các quốc gia đã xem xét tới "sự ngoại vi" là việc tham gia tích cực trong thế giới của PMTD. Ví dụ, số lượng các lập trình viên người Mexico và Tây Ban Nha (cả 2 quốc gia này với truyền thống hạn chế về công nghiệp phần mềm) trong các dự án như GNOME là đáng kể (Lancashire, "Mã nguồn, văn hóa và tiền: chủ nghĩa vị tha phai màu của sự phát triển nguồn mở", 2001) [164].

Và vai trò của Brazil còn thú vị hơn, với vô số các lập trình viên và chuyên gia trong các công nghệ của PMTD của quốc gia này, và việc ủng hộ mang tính quyết định từ các cơ quan hành chính nhà nước. gnuLinEx là một trường hợp mà đáng có sự chú ý đặc biệt, như một ví dụ về việc làm thế nào mà một khu vực với rất ít truyền thống về phát triển phần mềm có thể cố gắng thay đổi tình hình thông qua một chiến lược quyết liệt về gắn chặt vào các PMTD.

Từ viễn cảnh ra quyết định khi nói về việc triển khai các giải pháp phần mềm, chúng tôi có thể nhấn mạnh thực tế rằng có những thị trường nhất định nào đó (như các dịch vụ Internet hoặc các ứng dụng văn phòng) trong đó PMTD là một sự lựa chọn tự nhiên mà không thể không thấy được khi nghiên cứu dạng hệ thống nào để sử dụng.

Ở mặt tiêu cực, những năm này đã cho thấy làm thế nào môi trường pháp lý trong đó PMTD hoạt động lại đang thay đổi nhanh chóng toàn thế giới. Một mặt, các bằng sáng chế về phần mềm đang ngày càng gia tăng áp dụng trong nhiều quốc gia hơn nữa. Mặt khác, các luật mới về bản quyền làm cho khó khăn hoặc không thể phát triển các ứng dụng tự do trong một số hoàn cảnh, mà tình trạng nổi tiếng nhất là những trình xem DVD (do thuật toán mã hóa CSS mà công nghệ này sử dụng).

gnuLinEx

Vào đầu năm 2002 Chính quyền vùng Extremadura đã tuyên bố công khai dự án gnuLinEx. Ý tưởng là đơn giản: khuyến khích sự tạo ra một phát tán dựa trên GNU/Linux với mục tiêu cơ bản để sử dụng nó trong hàng ngàn máy tính sẽ được cài đặt trong các trường công khắp vùng này. Extremadura, nằm ở phần phía tây của Tây Ban Nha, giáp với Bồ Đào Nha, có khoảng 1 triệu dân và chưa bao giờ nổi tiếng về những sáng kiến công nghệ của mình. Trên thực tế, vùng này thực tế đã không có nền công nghiệp phần mềm.

Trong ngữ cảnh đó, gnuLinEx đã tạo ra một sự đóng góp rất thú vị cho bức tranh toàn cảnh về PMTD trên một phạm vi toàn cầu. Ngoài việc chỉ là một phát tán mới của GNU/Linux dựa trên Debian (mà vẫn là một câu chuyện đùa đáng giá), và ngoài ảnh hưởng khổng lồ của nó lên các phương tiện thông tin đại chúng (đây là lần đầu tiên Extremadura được đưa lên trang bìa của tờ Bưu điện Washington và một trong những sản phẩm PMTD đầu tiên đã làm được điều đó), những gì khác thường là sự ủng hộ mạnh mẽ của một nền hành chính nhà nước đối với PMTD. Chính quyền vùng Extremadura đã quyết định thử nghiệm một mô hình khác nơi mà các phần mềm giáo dục đã được quan tâm, và sau đó mở rộng mô hình này cho tất cả các phần mềm trong phạm vi ảnh hưởng của mình. Điều này đã làm cho nó

trở thành chính quyền nhà nước đầu tiên của một quốc gia phát triển đã quyết định áp dụng tiếp cận này. Nhiều sự quan tâm đã được tạo ra xung quanh sáng kiến của chính quyền vùng này, cả trong và ngoài Extremadura: có những viện hàn lâm mà dạy IT sử dụng gnuLinEx; những cuốn sách đã được viết để hỗ trợ việc dạy học này; những máy tính đang được bán với gnuLinEx được cài đặt sẵn. Nói chung, họ đang cố gắng tạo ra một nhà máy giáo dục và kinh doanh xung quanh kinh nghiệm này để hỗ trợ cho nó. Và kinh nghiệm này đã được xuất khẩu. Vào đầu thế kỷ 21 này, một vài cộng đồng tự trị ở Tây Ban Nha đã ủng hộ PMTD trong giáo dục (bằng cách này hay cách khác), và nói chung, tầm quan trọng đối với các chính quyền được thừa nhận một cách rộng rãi.

Knoppix

Vào cuối những năm 90, có những phát tán GNU/Linux có thể dễ dàng cài đặt được, nhưng Knoppix, phiên bản đầu tiên của nó đã xuất hiện vào năm 2002, có lẽ đã cho phép ý tưởng này đạt tới sự bày tỏ đầy đủ của nó. Đây là một CD mà nó khởi động được trên hầu hết các máy tính cá nhân, chuyển nó (mà không cần phải định dạng đĩa cứng, vì nó có thể được sử dụng "sống") vào một máy GNU/Linux hoạt động đầy đủ hoàn toàn, với một lựa chọn các công cụ thường dùng nhất. Knoppix kết hợp sự dò tìm tốt phần cứng một cách tự động với một sự lựa chọn tốt các chương trình và việc hoạt động "sống" được. Ví dụ, nó cho phép một kinh nghiệm nhanh chóng và trực tiếp của những gì có nghĩa để làm việc với GNU/Linux. Và nó làm gia tăng cho toàn bộ một họ các phát tán y hệt dạng này, đặc trưng cho những yêu cầu đặc thù của người sử dụng.

OpenOffice.org

Vào năm 1999, Sun Microsystems đã mua một công ty Đức có tên là Stardivision, sản phẩm ngôi sao của hãng này là StarOffice, một bộ các ứng dụng văn phòng tương tự về tính năng của bộ các công cụ của Microsoft Office. Một năm sau, Sun đã phân phối hầu hết các mã nguồn của StarOffice theo một giấy phép tự do (GPL) tạo ra dự án OpenOffice.org. Dự án này đã cho ra phiên bản 1.0 của OpenOffice.org vào tháng 05/2002. OpenOffice.org đã trở thành một bộ các ứng dụng văn phòng chất lượng với một chức năng tương tự với bất kỳ sản phẩm văn phòng nào khác, và, quan trọng hơn cả, nó tương tác được rất tốt với các định dạng dữ liệu của Microsoft Office. Những tính năng này đã làm cho nó trở thành ứng dụng PMTD tham chiếu trong thế giới các bộ phần mềm văn phòng.

Tầm quan trọng của OpenOffice.org, từ quan điểm của việc mở rộng PMTD cho số lượng lớn người sử dụng, là khổng lồ. Cuối cùng nó có khả năng thay đổi, hầu như không có vấn đề gì, từ các môi trường sở hữu độc quyền thông dụng với các bộ văn phòng (không nghi ngờ ứng dụng ngôi sao này trong thế giới các doanh nghiệp) tới toàn bộ các môi trường tự do (như GNU/Linux cộng với GNOME và/hoặc KDE cộng với OpenOffice.org). Hơn nữa, sự chuyển dịch có thể được thực hiện rất trơn tru: vì OpenOffice.org cũng làm việc được trên Microsoft Windows nên không cần thiết phải thay đổi các hệ điều hành để trải nghiệm sâu sắc với việc sử dụng PMTD.

Mozilla, Firefox và phần còn lại

Thực tế là kể từ khi có sự xuất hiện của nó vào năm 1994 cho tới 1996, Netscape Navigator đã từng là người dẫn đầu không thể thách thức trong các trình duyệt web, với thị phần lớn hơn 80%. Tình thế này đã bắt đầu thay đổi khi Microsoft đưa Internet Explorer vào trong Windows 95, làm cho Netscape Navigator dần dần mất thị phần. Vào đầu năm 1998 Netscape đã công bố rằng hãng sẽ phân phối một phần lớn mã nguồn của trình duyệt này như là PMTD, mà nó đã làm trong tháng 03 cùng năm đó, khởi động dự án Mozilla. Trong một thời gian dự án này đã bị bao phủ bởi sự không chắc chắn, và ngay cả sự bi quan (ví dụ, khi người đứng đầu của nó, Jamie Zawinski, đã rời bỏ nó), vì trong một thời gian đã không có sản phẩm nào được tung ra.

Vào tháng 01/2000, dự án đã tung ra Mozilla M13, mà nó đã được coi là phiên bản khá ổn định đầu tiên. Vào tháng 05/2002 phiên bản 1.0 cuối cùng đã được tung ra, phiên bản ổn định chính thức đầu tiên, 4 năm sau khi mã nguồn của Netscape Navigator đã được tung ra.

Cuối cùng thì Mozilla đã trở thành một thực tế, dù có lẽ quá muộn, nếu chúng ta nhớ trong đầu thị phần mà Internet Explorer đã có vào năm 2002 hoặc 2003 (khi nó đã từng là người dẫn đầu không thể tranh cãi bỏ Mozilla và những trình duyệt khác vào một vị thế hoàn toàn cách biệt). Nhưng bất chấp điều đó, Mozilla và những trình duyệt khác đã cho ra kết quả; không chỉ kết quả được mong đợi (trình duyệt của Mozilla), mà còn những thứ phụ thêm khác, như Firefox làm ví dụ, một trình duyệt khác dựa trên cùng máy HTML y hệt, mà nó đã trở thành sản phẩm chính, và kể từ khi nó xuất hiện vào năm 2005 thì từng tí một nó đã làm xói mòn thị phần của các trình duyệt khác. Dự án Mozilla đã giúp lấp một khoảng trống lớn trong thế giới của PMTD. Trước khi trình duyệt Konqueror xuất hiện (trình duyệt của dự án KDE), đã không có bất kỳ trình duyệt tự do nào với giao diện đồ họa. Khi Mozilla ra đời, một số lượng khổng lồ các dự án dựa trên nó đã nổi lên mà đã sản sinh ra một số lượng lớn các trình duyệt. Cùng lúc, sự kết hợp của Mozilla Firefox và OpenOffice.org cho phép PMTD được sử dụng cho hầu hết các tác vụ chung, ngay cả trong một môi trường của Microsoft Windows (chúng cả 2 làm việc không chỉ trong GNU/Linux, *BSD và các hệ thống dạng Unix, mà còn trên cả Windows). Lần đầu tiên trong lịch sử của PMTD, nó đã thực hiện được việc chuyển đổi từ PMSHĐQ sang PMTD trong các môi trường văn phòng bằng một tác vụ đơn giản: chúng ta có thể bắt đầu bằng việc sử dụng 2 ứng dụng này trên Windows, mà không cần thay đổi các hệ điều hành (cho những ai sử dụng nó thông thường), và qua thời gian sẽ hạn chế phần duy nhất không tự do và chuyển sang GNU/Linux hoặc FreeBSD.

Thư mục tham khảo

Trong “Netscape Navigator”, của Brian Wilson, [234], chúng ta có thể tra cứu một danh sách chi tiết về những phiên bản chính của Netscape Navigator và Mozilla, và những đặc tính chính của chúng.

Vụ kiện của SCO

Vào đầu năm 2003, tập đoàn SCO (trước đó là Caldera Systems và Caldera International) đã đệ trình một vụ kiện pháp lý chống lại IBM vì đã cho là vi phạm các quyền sở hữu trí tuệ của hãng. Dù vụ kiện là phức tạp, nó đã tập trung vào việc tố cáo rằng IBM đã đóng góp cho nhân Linux bằng mã nguồn của SCO. Vào tháng 05/2007, vấn đề này vẫn còn chưa được giải quyết xong và còn trở thành phức tạp hơn nữa bởi các vụ kiện pháp lý tiếp sau (IBM và Red Hat chống lại SCO, SCO chống lại AutoZone và

DaimlerChrysler, 2 người sử dụng IT lớn) và bằng các chiến dịch của SCO đe dọa truy nã các công ty lớn mà họ đã sử dụng Linux, …

Dù người chiến thắng cuộc chiến pháp lý khổng lồ này vẫn còn chưa nổi lên, thì vụ kiện này đã nhấn mạnh những khía cạnh pháp lý chắc chắn có liên quan tới PMTD. Đặc biệt, nhiều công ty đã coi những vấn đề mà họ có thể phải đối mặt nếu họ sử dụng Linux và các chương trình tự do khác, và sự đảm bảo rằng bằng cách làm như thế họ sẽ không vi phạm các quyền sở hữu công nghiệp hoặc quyền sở hữu trí tuệ của bên thứ ba. Theo một vài cách thức, thì vụ kiện này và những vụ kiện khác (như những vụ liên quan tới tính hợp lệ của các giấy phép GPL mà đã được giải quyết tại Đức vào năm 2005) cũng có thể được hiểu như một tín hiệu của sự chín muồi của PMTD. Nó đã dừng là một người lạ lẫm đối với thế giới của các doanh nghiệp để trở thành một phần của nhiều hoạt động của thế giới này (bao gồm những hoạt động liên quan tới các chiến lược pháp lý).

Ubuntu, Canonical, Fedora và Red Hat

Dù Canonical (công ty mà làm và phân phối Ubuntu) có thể được xem là một kẻ mới tới gần đây trong việc kinh doanh các phát tán GNU/Linux, thì những hoạt động của hãng xứng đáng với sự chú ý của chúng ta. Trong một thời gian khá ngắn, Ubuntu đã thiết lập được cho bản thân hãng như một trong những phát tán nổi tiếng nhất và được sử dụng rộng rãi nhất, với một uy tín về chất lượng tốt, và cực kỳ dễ dàng để cài đặt và sử dụng. Ubuntu cũng nổi tiếng về sự chú ý tuyệt vời hơn của hãng cho việc đưa vào những PMTD cơ bản so với hầu hết các phát tán được sản sinh ra bởi các công ty.

Tuy nhiên, đặc tính cơ bản của Ubuntu (và chiến lược của Canonical) đã từng dựa trên Debian, một phát tán được tạo ra và duy trì bởi những tình nguyện viên. Trên thực tế, Ubuntu không phải là trường hợp đầu tiên của một phát tán dựa trên Debian (một trường hợp nổi tiếng khác là gnuLinEx), mà có lẽ nó là một phát tán đã nhận được sự đầu tư nhiều nhất. Ví dụ, Canonical đã thuê một số lượng lớn các chuyên gia của Debian (nhiều người trong số họ tham gia trong dự án này) và đã theo đuổi một chiến lược tìm kiếm sự hợp tác với dự án của những tình nguyện viên. Ở một mức độ nào đó, thì Canonical đã cố gắng lấp đi những gì mà nó coi là còn thiếu từ Debian để giành được sự chấp nhận từ người sử dụng thông thường.

Về phần mình thì Red Hat đã đi theo một lối khác để giành chiến thắng trong một tình thế tương tự. Bắt đầu từ một phát tán được sản xuất hoàn toàn với những tài nguyên của riêng hãng, nó đã quyết định hợp tác với Fedora, một nhóm các tình nguyện viên mà họ đã từng làm việc với các phát tán dựa trên Red Hat, để tạo ra Fedora Core, phát tán "cộng đồng" của nó. Red Hat duy trì phiên bản của hãng cho các công ty, nhưng sự hợp tác này với các tình nguyện viên, cuối cùng, rất giống với sự hợp tác mà đã tạo ra Ubuntu.

Có lẽ tất cả những phong trào này không gì hơn là sản phẩm của sự cạnh tranh khốc liệt diễn ra trong thị trường các phát tán GNU/Linux và của một trong những xu thế đáng chú ý hơn: sự hợp tác của các công ty với các tình nguyện viên (*với cộng đồng*) để sản xuất PMTD.

Các phát tán được tùy biến

Vì Linux đã lên sân chơi, một số lượng lớn các nhóm và công ty đã tạo ra những phát tán của riêng họ dựa trên nó. Nhưng trong những năm này, hiện tượng này đã gắn vào với nhiều tổ chức và công ty mà họ muốn tùy biến các phiên bản cho những yêu cầu của riêng họ. Sự tùy biến là có khả năng mở rộng vì quá trình này đã trở nên rẻ hơn và có sự sẵn sàng rộng rãi về tri thức kỹ thuật để làm thế, ngay cả việc làm cho điều này trở thành một thị trường phù hợp cho một số công ty nào đó.

Có lẽ một trong những trường hợp nổi tiếng nhất về các phát tán được tùy biến là một phát tán cho các cộng đồng tự trị của Tây Ban Nha. Chính quyền vùng Extremadura với gnuLinEx của mình đã làm bùng phát một xu thế mà nhiều cộng đồng tự trị khác kể từ đó đã đi theo. Quá trình này là quá phổ biến đến nỗi một vài trong số họ thường xuyên triệu tập các nhà thầu để tạo ra và duy trì các phiên bản mới cho các phát tán của họ.

Sự tạo ra các phát tán được tùy biến cho thấy rõ một xu thế mà thế giới PMTD đã từng bàn luận từ lâu: việc áp dụng các chương trình cho những nhu cầu cụ thể của người sử dụng mà không phải là những nhà sản xuất gốc thì cần thiết phải tiến hành thực hiện sự thích nghi.

Thư mục tham khảo

Một số phát tán GNU/Linux nổi tiếng nhất tại các cộng đồng tự trị của Tây Ban Nha gồm:

- gnuLinEx: http://linex.org (Extremadura)
- Guadalinex: http://guadalinex.org (Andalucía)
- Lliurex: http://lliurex.net (Comunidad Valenciana)
- Augustux: http://www.zaralinux.org/proy/augustux/ (Aragón)
- MAX: http://www.educa.madrid.org/web/madrid_linux/ (Madrid)
- MoLinux: http://molinux.info (Castilla-La Mancha)

Những hợp tác giữa công ty với công ty và giữa công ty với những tình nguyện viên

Từ thực tế khởi đầu của PMTD, đã có những công ty hợp tác với các tình nguyện viên trong việc phát triển các ứng dụng. Tuy nhiên, trong những năm này dường như là khi mà chúng ta đạt được độ chín muồi thì sẽ có sự tăng trưởng về số lượng các công ty sử dụng PMTD như một phần chiến lược của họ để hợp tác với các công ty khác, khi họ thấy nó thú vị. 2 trong số các trường hợp điển hình, được tổ chức đặc biệt với mục tiêu này, là ObjectWeb (một liên minh được hình thành tại Pháp mà nó theo thời gian rõ ràng đã trở thành quốc tế) và Morfeo (tại Tây Ban Nha). Trong cả 2 trường hợp, một nhóm các công ty đã đồng ý phát triển một tập hợp các hệ thống tự do mà họ quan tâm, và đã quyết định phân phối nó như là các PMTD.

Trong các trường hợp khác, các công ty đã tìm một cách tích cực để hợp tác trong các dự án tự do được khuyến khích bởi các tình nguyện viên, hoặc đã cố gắng làm cho các tình nguyện viên hợp tác với các dự án tự do của riêng họ. Quỹ GNOME hoặc Ubuntu đã được nhắc tới ở trên đối với Debian là những

ví dụ của kịch bản đầu tiên này. Sun và OpenOffice.org và OpenSolaris, hoặc Red Hat với Fedora Core, là những ví dụ thứ 2.

Việc mở rộng tới những phạm vi khác

PMTD đã chứng minh rằng trong lĩnh vực sản xuất các chương trình có một cách khác để tiến hành. Trong thực tế, chúng ta đã thấy cách mà việc trao sự tự do để phân phối, sửa đổi và sử dụng có thể đạt được tính bền vững, hoặc thông qua công việc của các tình nguyện viên, hoặc thông qua thế hệ các doanh nghiệp mà nó cho phép các công ty sống sót được.

Thời gian qua đi, ý tưởng y hệt này đang được truyền sang những lĩnh vực khác của công việc tri thức. Các giấy phép sáng tạo chung Creative Commons đã làm cho nó có thể đối với các lĩnh vực tự do như văn học, âm nhạc hoặc video. Wikipedia đang chứng minh rằng một lĩnh vực đặc biệt như sản xuất các từ điển bách khoa toàn thư có thể chuyển qua một con đường rất thú vị. Và sẽ có nhiều hơn và nhiều hơn nữa các tác giả văn học, các băng nhóm âm nhạc và ngay cả những nhà sản xuất phim có quan tâm trong các mô hình sản xuất và phân phối tự do.

Trong tất cả các lĩnh vực này sẽ vẫn còn một con đường dài phải đi, và trong hầu hết tất cả chúng thì kinh nghiệm thực tế còn chưa được chứng minh đầy đủ rằng sự sáng tạo bền vững là có thể với các mô hình tự do. Nhưng nó không thể bị phủ nhận mà những thí điểm với nó đang đạt tới một điểm sôi.

Phần mềm tự do như là một đối tượng nghiên cứu

Dù một số công việc, như "Nhà thờ lớn và cái chợ" nổi tiếng đã làm sạch con đường cho việc nghiên cứu của PMTD như vậy, thì cho tới năm 2001 và những năm tiếp sau cộng đồng hàn lâm đã bắt đầu coi PMTD như thứ gì đó đáng nghiên cứu. Qua thời gian, tính sẵn sàng to lớn các dữ liệu (hầu như mọi thứ trên thế giới này về PMTD là công cộng và sẵn sàng từ những kho lưu trữ thông tin công khai) và những đổi mới sáng tạo mà PMTD đã chứng minh đã thu hút sự chú ý của nhiều nhóm. Những năm giữa thập niên của năm 2000 đã có một vài hội nghị tập trung đặc biệt vào PMTD, các tạp chí hàng đầu sản sinh ra những bài viết về nó, và các cơ quan đầu tư nghiên cứu đang mở ra những con đường đặc biệt hướng tới nó.

2.5 Tương lai: Một tiến trình đầy trở ngại?

Tất nhiên, khó đoán trước được tương lai. Và điều đó chắc chắn không phải là mục tiêu của chúng tôi. Vì thế, thay vì cố giải thích những gì tương lai của PMTD có thể, chúng tôi sẽ cố gắng chỉ ra những vấn đề mà nó sẽ phải đối mặt một cách nhìn thấy trước được (và quả thực đã từng đối mặt lâu dài). Thế giới PMTD có khả năng vượt qua được những trở ngại này ra sao thì không nghi ngờ gì là sẽ xác định được tình trạng của nó trong thời gian vài năm.

- FUD (*sự sợ hãi, không chắc chắn, nghi ngờ*). Đây là một kỹ thuật khá phổ biến trong thế giới công nghệ thông tin, được sử dụng bởi các đối thủ cạnh tranh của PMTD để làm mất lòng tin

vào PMTD, ít nhiều bằng sự biện hộ và các mức độ thành công khác nhau. Nói chung, PMTD đã khá miễn dịch đối với các kỹ thuật này, có lẽ nhờ vào sự phức tạp và các cách thức khác nhau của việc thấm vào các công ty.

- Sự biến mất. Nhiều công ty đang thử nghiệm những giới hạn của PMTD như một mô hình, và đặc biệt đang cố gắng đưa ra cho các khách hàng của họ các mô hình mà chúng thể hiện được một vài đặc tính tương tự như đối với PMTD. Vấn đề chính mà có thể tự nó thể hiện bằng dạng mô hình này là việc nó tạo ra sự lúng túng trong các khách hàng và các lập trình viên, những người cần đọc tài liệu in chi tiết để nhận thức được những gì mà họ đang được chào không hề có những ưu thế mà các PMTD chào cho họ. Mô hình nổi tiếng nhất của dạng này là chương trình Mã nguồn Chia sẻ (Shared Source) của Microsoft.
- Thiếu tri thức. Trong nhiều trường hợp, những người sử dụng chuyển sang PMTD đơn giản vì họ nghĩ rằng nó là miễn phí; hoặc vì họ nghĩ rằng nó là "hợp thời trang". Nếu họ không nhìn sâu hơn vào bên trong nó, và nghiên cứu bằng một số lượng chắc chắn nào đó một cách chi tiết những ưu điểm mà PMTD có thể đưa ra như một mô hình, thì họ có rủi ro không tận dụng được ưu thế đầy đủ của chúng. Trong nhiều trường hợp, những giả thiết ban đầu trong thế giới PMTD là quá khác với những giả thiết theo lối truyền thống trong thế giới của PMSHĐQ mà một phân tích tối thiểu được yêu cầu để hiểu rằng những gì trong một trường hợp thường là trong trường hợp khác có thể là không thể được, và ngược lại. Vì thế sự thiếu hụt tri thức chỉ có thể tạo ra sự không thỏa mãn và mất các cơ hội cho bất kỳ ai hoặc tổ chức nào tiếp cận PMTD.
- Những cản trở pháp lý. Đây chắc chắn là vấn đề chính mà PMTD sẽ gặp phải trong những năm sắp tới. Dù môi trường pháp lý trong đó PMTD được phát triển trong những năm 80 và nửa đầu những năm 90 đã không phải là lý tưởng, thì ít nhất nó cũng đã để lại đủ không gian cho nó phát triển một cách tự do. Kể từ đó, sự mở rộng phạm vi của việc cấp bằng sáng chế cho phần mềm (mà đã xảy ra trong nhiều nước phát triển) và pháp luật mời về bản quyền (hạn chế sự tự do của những lập trình viên để tạo ra) đang ngày càng tạo ra những cản trở lớn hơn cho sự thâm nhập của PMTD vào trong những lĩnh vực của các ứng dụng quan trọng.

2.6 Tóm lược

Chương này trình bày lịch sử của PMTD. Những năm 60 từng là một giai đoạn bị áp đảo bởi các máy tính lớn của IBM mà trong đó phần mềm đã được phân phối cùng với phần cứng, và thường là với cả mã nguồn. Trong những năm 70, phần mềm đã bắt đầu được bán một cách riêng rẽ, và sớm có những phân phối sở hữu độc quyền, mà chúng đã không bao gồm mã nguồn và đã không trao quyền để sửa đổi hoặc phân phối lại, đã trở thành hầu như là lựa chọn duy nhất.

Trong thập kỷ 1970, công việc đã bắt đầu trong việc phát triển hệ điều hành Unix tại Bell Labs của AT&T, sau này tạo ra Unix BSD. Sự tiến hóa của nó, song song với sự ra đời của Internet, đã phục vụ như một nơi thí điểm cho những cách thức mới về phát triển trong hợp tác, mà sau này đã trở thành phổ biến trong thế giới của PMTD.

Trong năm 1984, Richard Stallman đã bắt đầu làm việc trong dự án GNU, sáng lập FSF, viết giấy phép

GPL, và nói chung, thiết lập ra những quỹ của PMTD như chúng ta biết bây giờ.

Trong những năm 90 Internet đã chín muồi, đưa ra cho các cộng đồng PMTD những kênh mới cho sự giao tiếp và phân phối. Trong năm 1991, Linus Torvalds đã bắt đầu phát triển một nhân tự do (Linux) mà nó đã giúp hoàn tất hệ điều hành GNU, mà nó hầu như đã có sẵn tất cả các phần để trở thành một hệ điều hành hoàn chỉnh tương tự như Unix: Trình biên dịch ngôn ngữ C (GCC), trình soạn thảo (Emacs), hệ thống cửa sổ (X Window), …

Đây là cách mà các hệ điều hành GNU/Linux đã được sinh ra, rẽ nhánh thành nhiều phát tán, như Red Hat GNU/Linux và Debian GNU/Linux. Về cuối những năm 90, những hệ thống này đã được hoàn chỉnh với 2 môi trường giao diện màn hình: KDE và GNOME.

Trong thập niên của những năm 2000, PMTD đã dẫn đầu trong một vài khu vực (như là các máy chủ web, được áp đảo bởi Apache) và các công cụ mới đã xuất hiện bao trùm một số lượng lớn các yêu cầu về IT.

Xem thêm

Những độc giả có quan tâm sẽ thấy trong Phụ lục B một danh sách của một số ngày tháng thích hợp nhất trong lịch sử của PMTD.

3 Các khía cạnh pháp lý

> "Các giấy phép cho hầu hết các phần mềm được thiết kế để lấy đi sự tự do của bạn để chia sẻ và thay đổi nó".
>
> Giấy phép Công cộng Chung GNU, phiên bản 2 (GPLv2).

Chương này xem xét những khía cạnh pháp lý chính có liên quan tới PMTD. Để đặt chúng vào trong ngữ cảnh này, chúng tôi bắt đầu với một sự giới thiệu nhỏ về những khái niệm cơ bản nhất của các quyền sở hữu trí tuệ và sở hữu công nghiệp, trước khi đưa ra định nghĩa chi tiết của *PMTD, PMNM* và các khái niệm liên quan khác. Chúng tôi cũng xem xét một số chi tiết trong các giấy phép PMTD phổ biến nhất và ảnh hưởng của chúng lên các mô hình kinh doanh (chủ đề được đề cập tới một cách chi tiết hơn trong chương 5) và các mô hình phát triển.

3.1 Giới thiệu ngắn gọn về sở hữu trí tuệ

Khái niệm *sở hữu trí tuệ* có một loạt ý nghĩa theo ngữ cảnh của nó và ai sử dụng nó. Ngày nay nó thường được sử dụng trong nhiều lĩnh vực để tham chiếu tới một loạt các quyền ưu tiên được trao cho những hàng hóa vô hình bằng những giá trị kinh tế. Nó bao gồm các khái niệm như là *bản quyền* và tương tự, mà nó bảo vệ khỏi việc sao chép các tác phẩm văn học hoặc nghệ thuật, các chương trình máy tính, các biên dịch dữ liệu, các thiết kế công nghiệp, … mà không được phép; các thương hiệu, mà nó bảo vệ cho các biểu tượng; các chỉ dẫn địa lý, mà chúng bảo vệ các danh hiệu gốc; các bí mật thương mại, mà chúng bảo vệ việc che dấu thông tin, và các bằng sáng chế, mà chúng thừa nhận các nhà độc quyền một cách tạm thời đối với những phát minh sáng chế trong sự trao đổi đối với việc tiết lộ chúng. Tuy nhiên, trong nhiều truyền thống pháp lý, bao gồm cả truyền thống của Tây Ban Nha và Bồ Đào Nha, một sự khác biệt được tạo ra giữa *sở hữu trí tuệ*, mà nó tham chiếu một cách riêng biệt cho bản quyền, và *sở hữu công nghiệp*, mà nó bao trùm các khái niệm khác.

Trong mọi trường hợp, pháp luật áp dụng được cho tất cả các khía cạnh này là một trong những khía cạnh được phối hợp một cách thực tế tốt nhất trên thế giới. Một mặt, Tổ chức Sở hữu Quốc tế Toàn cầu WIPO (Worldwide International Property Organisation) bao trùm cả 2 dạng sở hữu theo tất cả các khía cạnh của chúng. Mặt khác, hiệp định về các khía cạnh liên quan của quyền sở hữu trí tuệ TRIPS (Trade-Related aspects of Intellectual Property rights) thiết lập nên những mức độ tối thiểu nào đó về bảo vệ và bắt buộc tất cả các quốc gia thành viên của Tổ chức Thương mại Thế giới WTO (World Trade Organisation) phải phát triển chúng trong những khung thời gian cụ thể nào đó, theo mức phát triển của quốc gia[3] đó.

Điều 27 của Tuyên ngôn về Quyền con người nhận thức rằng mỗi người có quyền bảo vệ những kết quả về quyền lợi đạo đức và vật chất từ bất kỳ việc sản xuất khoa học, văn học hoặc nghệ thuật nào của kết quả mà anh ta là tác giả. Tuy nhiên, trong nhiều trường hợp (và thường là trong trường hợp của phần mềm), quyền này được chuyển vào trong thực tế tới các công ty mà họ sử dụng những người sáng tạo hoặc những công ty phân phối hoặc bán những sáng tạo của họ. Tuy nhiên, sở hữu trí tuệ được

3 Hiệp định TRIPS đã được ký dưới sức ép từ các quốc gia công nghiệp hóa (đặc biệt là Mỹ và Nhật Bản).

chứng minh là đúng không chỉ về phương diện đạo đức, mà còn vì những lý do thực tiễn, để phù hợp với quyền khác: quyền công khai đối với lợi ích từ sự sáng tạo, khuyến khích nó thông qua những khích lệ và việc bảo vệ đầu tư trong sáng tạo, nghiên cứu và phát triển.

Để hài hòa 2 quyền này, sở hữu trí tuệ là tạm thời và sẽ hết hạn một khi nó đã hoàn thành chức năng khuyến khích của nó.

Nhưng sự hết hạn không chỉ là tính năng phân biệt giữa sở hữu trí tuệ và sở hữu thông thường. Ngày nay, các sản phẩm của nó có thể được sao chép dễ dàng và rẻ tiền, mà không có bất kỳ sự mất mát nào về chất lượng. Việc sao chép không gây thiệt hại cho bên mà đã hưởng lợi từ những gì được sao chép, không giống như tên trộm, mà tước đoạt đi của người chủ sở hữu gốc ban đầu. Việc sao chép có thể gây thiệt hại cho người chủ sở hữu, bằng việc tước đoạt từ anh ta lợi tức tiềm tàng từ việc bán hàng.

Việc kiểm soát việc sao chép những thứ vô hình còn phức tạp hơn nhiều so với việc kiểm soát tên trộm các sở hữu hữu hình và có thể dẫn chúng ta tới một hoàn cảnh của một trạng thái cảnh sát, phải kiểm soát tất cả các bản sao các thông tin, và mất an ninh về pháp lý, khi mà tiềm năng cho sự vi phạm ngẫu nhiên không cố ý các quyền gia tăng. Hơn nữa tính sáng tạo là gia tăng dần: luôn tạo ra các bản sao của thứ gì đó, và việc phân biệt giữa một sự bắt chước yếu kém và sự truyền cảm hứng là một thứ huyền ảo khó thấy.

Để nghiên cứu điều này sâu sắc hơn, các phần sau đi qua một số chủng loại sở hữu trí tuệ. Trong mọi trường hợp, chúng tôi có thể đã đưa ra rằng PMTD đề xuất một điểm mới về sự cân bằng trong lĩnh vực này, viện lý cho những lợi ích của việc sao chép và đổi mới sáng tạo dần dần đối nghịch lại với sự kiểm soát độc nhất một tác phẩm bởi tác giả của nó.

3.1.1 Bản quyền

Bản quyền bảo vệ sự diễn đạt của nội dung, chứ không phải bản thân nội dung. Bản quyền đã được phát triển để bù đắp cho các tác giả các cuốn sách hoặc tác phẩm nghệ thuật. Các công việc được bảo vệ có thể diễn đạt các ý tưởng, tri thức hoặc các phương pháp mà chúng được sử dụng một cách tự do, nhưng bị cấm tái sinh chúng mà không có sự cho phép một phần hoặc toàn phần, có hoặc không có những sửa đổi. Việc bảo vệ này là rất đơn giản, vì nó tự động có hiệu lực với một phạm vi hầu như vạn năng chỉ khi công việc này được xuất bản/tung ra. Hiện tại, nó đã được mở rộng cho các chương trình máy tính và (trong một số lĩnh vực về địa lý) đối với những biên soạn dữ liệu.

Luật về Sở hữu Trí tuệ (LPI) tại Tây Ban Nha, và các luật tương tự tại các nước khác, được phát triển trên cơ sở của Công ước Berne năm 1886 cho việc bảo vệ các tác phẩm văn học và nghệ thuật, điều chỉnh lại cho bản quyền. Các quyền này được chia thành các quyền về đạo đức và sở hữu trí tuệ. Cái đầu đảm bảo cho sự kiểm soát của tác giả đối với việc phân phối tác phẩm của anh ta, dưới tên hoặc bút danh của anh ta, sự thừa nhận nguồn tác giả, tôn trọng tính toàn vẹn của tác phẩm và quyền sửa đổi và rút nó lại. Cái sau trao cho tác giả quyền khai thác tác phẩm về mặt kinh tế và có thể được nhượng lại toàn phần hoặc một phần, độc quyền hoặc không, đối với bên thứ 3. Các quyền về đạo đức là suốt đời hoặc không hạn định, trong khi các quyền về trí tuệ có một thời hạn khá dài (70 năm sau cái chết của tác giả, trong trường hợp của một người cụ thể theo luật của Tây Ban Nha).

Việc nhượng lại các quyền này được thiết lập bằng một dạng hợp đồng được biết tới như một giấy phép. Trong trường hợp của các chương trình sở hữu độc quyền, thường được phân phối thông qua các giấy phép sử dụng "không dành riêng", được hiểu như được chấp nhận một cách tự động bằng việc mở hoặc cài đặt sản phẩm. Vì thế không cần thiết phải ký hợp đồng, vì trong trường hợp người nhận không chấp nhận nó, thì các quyền mặc định theo luật được điều chỉnh một cách tự động, nghĩa là không. Các giấy phép không thể hạn chế một số quyền được trao bởi pháp luật hiện hành, như là quyền để thực hiện các bản sao cá nhân của tác phẩm nghệ thuật hoặc âm nhạc, mà nó cho phép một bản sao của việc ghi lại để được trao cho một người bạn như một món quà, nhưng quyền này không áp dụng được cho các chương trình. Theo LPI năm 1996 (Luật Tây Ban Nha về Sở hữu Trí tuệ. Sắc lệnh Lập pháp của nhà Vua 1/1996, ngày 12/04) [77], được sửa đổi năm 2006 (Luật về Sở hữu Trí tuệ. Luật 23/2006, ngày 07/07) [79], về phương diện các chương trình thì luôn có thể tạo một bản sao lưu, các chương trình có thể được nghiên cứu để được làm cho tương hợp được và chúng có thể được sửa cho đúng và áp dụng cho các nhu cầu của chúng ta (mà là khó, vì thường thì chúng ta không có sự truy cập được vào mã nguồn). Các quyền này có thể không bị hạn chế thông qua các giấy phép, dù các luật đang được xem xét lại, theo một xu hướng hình như không thể dừng lại được nhằm hạn chế các quyền của người sử dụng. Những biên soạn có tổ chức các tác phẩm hoặc dữ liệu của bên thứ 3 cũng sẽ tuân thủ theo bản quyền, dù theo các điều khoản khác với một khung thời gian ngắn hơn.

Các công nghệ thông tin mới, và đặc biệt là web, đã làm biến chuyển một các sâu sắc việc bảo vệ bản quyền, khi mà những trình diễn nội dung là dễ dàng hơn nhiều để sao chép so với bản thân nội dung. Và trong trường hợp của các chương trình và một số tác phẩm nghệ thuật (âm nhạc, hình ảnh, phim, và ngay cả văn học) thì chúng "làm việc" một cách tự động trên máy tính mà không cần bất kỳ nỗ lực nào của con người mà có thể đánh giá được. Tuy nhiên, những thiết kế hoặc phát minh sáng chế cần phải được xây dựng và có thể đưa vào sản xuất. Khả năng tạo ra của cải mà không có chi phí này đã dẫn tới một tỷ lệ lớn công chúng, đặc biệt ở các nước nghèo, nhân bản các chương trình mà không trả tiền cho giấy phép, không có nhận thức của công chúng về điều này đang là một "hành động độc hại" (không giống như trong trường hợp ăn cắp tài sản vật lý, ví dụ thế). Trong khi đó, các nhà sản xuất chương trình, hoặc cá thể hoặc trong liên minh (thông qua BSA - Liên minh Phần mềm Doanh nghiệp, ví dụ), gây áp lực khổng lồ để thuyết phục các chính phủ mua các giấy phép để tránh tình trạng được cho là sự ăn cướp.

Lưu ý

Từ ăn cướp đã trở nên được chấp nhận chung như một từ đồng nghĩa cho 'sự vi phạm bất kỳ dạng sở hữu trí tuệ nào, đặc biệt trong trường hợp sao chép bất hợp pháp các chương trình, âm nhạc và phim ảnh'. Khái niệm này dường như bị cường điệu hóa và trong từ điển của Học Viện Ngôn ngữ Hoàng Gia Tây Ban Nha thì nó xuất hiện với nghĩa đó trong nghĩa có tính tượng trưng, vì từ gốc tham chiếu tới 'sự cướp bóc bằng bạo lực liên quan ở biển'. Điều này giải thích vì sao Richard Stallman khuyến cáo tránh nó ("Một vài sự khó hiểu hoặc các từ và nhóm từ nặng nề đáng tránh", 2003) [212].

Chính xác để bảo vệ bản quyền của các nội dung với các giấy phép sở hữu độc quyền, cái gọi là các hệ thống quản lý các quyền số DRM (Digital Rights Management), được thiết kế để kiểm soát sự truy cập và sử dụng các dữ liệu ở định dạng số hoặc để hạn chế sử dụng nó trong các thiết bị cụ thể nào đó. Việc sử dụng các hệ thống DRM đã bị kịch liệt chỉ trích trong nhiều lĩnh vực, vì chúng bảo vệ bản quyền bằng việc áp đặt các hạn chế vượt quá giới hạn, mà nó giải thích vì sao một số tổ chức, như FSF,

khuyến cáo việc dịch từ đồng nghĩa này như là sự quản lý những hạn chế số, trong một nỗ lực để tránh sử dụng các quyền của từ này, vì nó coi rằng có một sự tước đoạt quá đáng các quyền của người sử dụng vì lợi ích của việc thỏa mãn những đòi hỏi của bản quyền.

3.1.2 Bí mật thương mại

Một tài nguyên khác mà các công ty sử dụng để kiếm lợi nhuận từ những đầu tư của họ là bí mật thương mại, được bảo vệ bởi các luật về sở hữu công nghiệp, miễn là các công ty thực hiện các biện pháp đủ để dấu các thông tin mà họ không muốn tiết lộ. Trong trường hợp các sản phẩm hóa học và dược học mà chúng đòi hỏi sự phê chuẩn của chính phủ, thì chính phủ cam kết không tiết lộ các dữ liệu đã được đệ trình mà chính phủ không buộc phải đưa ra công khai.

Một trong những ứng dụng nổi tiếng nhất về bí mật thương mại là nền công nghiệp phần mềm sở hữu độc quyền, mà nó thường bán các chương trình được biên dịch mà không có sự truy cập được vào mã nguồn, để ngăn cản các chương trình dẫn xuất từ chương trình đang được phát triển.

Nhìn thấy ngay dường như là sự bảo vệ bí mật thương mại là ngang trái, vì nó có thể tước đoạt của xã hội những tri thức hữu dụng một cách vô hạn định. Ở một mức độ nào đó, một số luật pháp cũng hiểu theo cách này, và cho phép kỹ thuật nghịch đảo để ngăn cấm hoạt động này tại nhiều quốc gia, và trong những quốc gia khác chỉ được làm cho có khả năng trên cơ sở của tính tương thích.

Việc có hay không chuyện bí mật thương mại là một sự tước đoạt, trong nhiều trường hợp còn tốt hơn một bằng sáng chế vì nó đưa ra một ưu thế cạnh tranh đối với người đặt một sản phẩm vào thị trường trong khi sự cạnh tranh đó cố gắng bắt chước nó thông qua kỹ thuật nghịch đảo.

Sản phẩm càng tinh vi phức tạp bao nhiêu thì nó sẽ càng gây tổn thất cho sự cạnh tranh để tái sinh nó bấy nhiêu, trong khi nếu nó là tầm thường, thì nó sẽ được sao chép một cách nhanh chóng. Sự bắt chước với những cải tiến từng là nền tảng trong sự phát triển của các siêu cường ngày nay (Mỹ và Nhật Bản) và là rất quan trọng cho sự độc lập về tài chính của các quốc gia đang phát triển.

3.1.3 Các bằng sáng chế và các mô hình tiện ích

Phương án khác của bí mật thương mại là bằng sáng chế. Để trao đổi cho một sự độc quyền 17 – 25 năm và một giá thành tài chính cụ thể, thì một *phát minh* được hé lộ một cách công khai sao cho nó có thể dễ dàng được tái sinh. Nó có mục đích để khuyến khích nghiên cứu riêng tư cá nhân, không mất chi phí đối với người đóng thuế và không thua thiệt về hậu quả. Người giữ bằng sáng chế có thể quyết định liệu có hay không việc cho phép những người khác sử dụng nó và cái giá phải trả cho giấy phép.

Theo học thuyết chính thống thì hệ thống bằng sáng chế khuyến khích đổi mới sáng tạo, nhưng ngày càng có nhiều tiếng nói mà bản thân họ nghe thấy được với quan điểm rằng nó cản trở đổi mới sáng tạo, hoặc vì hệ thống này được triển khai một cách kém cỏi hoặc vì tự bản thân họ coi đó là sự ngang trái sai lầm (François-René Rideau, "Các bằng sáng chế là một sự vô lý ngớ ngẩn về kinh tế", 2000) [194].

Những gì được xem xét khi một sự đổi mới sáng tạo đã thay đổi theo thời gian, và có sức ép khổng lồ để mở rộng phạm vi của hệ thống, để đưa vào những thuật toán, những chương trình, những mô hình kinh doanh, những vật chất tự nhiên, những gen và hình thái của cuộc sống, bao gồm cả thực và động vật. TRIPS đòi hỏi hệ thống các bằng sáng chế phải không phân biệt đối xử đối với bất kỳ lĩnh vực tri thức nào. Sức ép của Tổ chức Sở hữu Trí tuệ Thế giới (WIPO) có mục tiêu để hạn chế nhu cầu đối với một đổi mới sáng tạo để có được một ứng dụng công nghiệp và cũng để giảm các chuẩn về đổi mới sáng tạo được yêu cầu đối với một bằng sáng chế. Mỹ là quốc gia hàng đầu với những yêu cầu tối thiểu về tính có thể được cấp bằng sáng chế, và cũng là nước tham chiến nhiều nhất đối với các quốc gia khác để áp dụng các chuẩn của mình, mà quên rằng Mỹ đã từ chối chấp nhận các bằng sáng chế của nước ngoài khi nước này từng là một quốc gia còn chưa phát triển đầy đủ.

Sau khi giành được một bằng sáng chế, các quyền của người sở hữu là độc lập với chất lượng của phát minh và nỗ lực được đầu tư vào để giành được nó. Đưa ra giá thành cho việc duy trì một bằng sáng chế, và giá thành vụ kiện pháp lý, chỉ các công ty lớn là có khả năng để duy trì và tiếp tục duy trì một hồ sơ lớn về các bằng sáng chế, mà chúng đặt các công ty lớn này vào vị thế để đấu tranh chống lại bất kỳ sự cạnh tranh nào. Đưa ra sự dễ dàng để có được các bằng sáng chế trong các giải pháp tầm thường hoặc được áp dụng một cách rộng rãi, họ có thể vì thế độc quyền một lĩnh vực rộng rãi các hoạt động kinh tế.

Với các bằng sáng chế, nhiều hoạt động, đặc biệt việc lập trình, đã trở nên cực kỳ rủi ro, vì nó rất dễ rằng trong việc phát triển một chương trình phức tạp sẽ có một sự vi phạm tình cờ ngẫu nhiên một số bằng sáng chế. Khi 2 hoặc nhiều hơn các công ty tiến hành nghiên cứu để giải quyết một vấn đề, thì có xác xuất cao có thể là họ sẽ đạt được một giải pháp tương tự hầu như ở cùng một thời điểm, nhưng chỉ một trong số họ (thường là công ty với tài nguyên nhiều nhất) sẽ có được bằng sáng chế cho phát minh của họ, ngăn cản những người khác có được bất kỳ cơ hội nào bù đắp lại đầu tư của họ.

Bất kỳ sự phát triển công nghệ phức tạp nào cũng trở thành cơn ác mộng nếu để giải quyết từng phần mà bạn trước tiên cần thấy liệu giải pháp được tìm ra đã có bằng sáng chế hay là chưa (hoặc đang treo bằng sáng chế), để có được giấy phép hoặc tìm ra được một giải pháp thay thế. Vấn đề này đặc biệt khắc nghiệt với PMTD, nơi mà sự vi phạm các bằng sáng chế về thuật toán là hiển nhiên từ việc kiểm tra mã nguồn một cách đơn giản. Dù tại châu Âu vẫn còn là bất hợp pháp đối với việc cấp bằng sáng chế cho một thuật toán, thì nó sẽ trở nên có thể trong tương lai gần, có lẽ vào thời gian mà độc giả đọc các dòng này.

3.1.4 Thương hiệu và logo được đăng ký

Thương hiệu và logo là tên và biểu tượng mà chúng đại diện cho một chất lượng được thiết lập (hoặc một sự đầu tư lớn một cách công khai). Chúng không thật quan trọng trong thế giới của PMTD, có lẽ vì việc đăng ký chúng có một chi phí nào đó. Vì thế, chỉ một ít cái tên quan trọng như Nguồn Mở (bởi Quỹ Nguồn Mở – OSI), Debian (bởi Quỹ Phần mềm theo Lợi ích Công chúng), GNOME (bởi Quỹ GNOME), GNU (bởi FSF) hoặc OpenOffice.org (bởi Sun Microsystems), được đăng ký, và chỉ tại một số ít các quốc gia. Tuy nhiên, việc không đăng ký tên đã gây ra những vấn đề. Ví dụ, tại Mỹ (1996) và Hàn Quốc (1997) mọi người đã đăng ký tên Linux và đã yêu cầu chi tiền để sử dụng nó. Việc giải quyết những tranh cãi này dẫn tới những chi phí pháp lý và nhu cầu phải chứng minh việc sử dụng tên này

trước ngày đăng ký.

3.2 Các giấy phép của PMTD

Về phương diện pháp lý mà nói, thì tình trạng của các chương trình tự do trong mối tương quan với các PMSHĐQ là rất không khác nhau: chúng cả 2 đều được phân phối theo một giấy phép. Sự khác biệt nằm ở những gì giấy phép này cho phép. Trong trường hợp các giấy phép của chương trình tự do, mà nó đặc biệt không hạn chế việc sử dụng, phân phối lại và sửa đổi chúng, thì những gì có thể được áp đặt là những điều kiện cần phải đáp ứng được một cách chính xác trong trường hợp muốn phân phối lại chương trình. Ví dụ, có thể yêu cầu sự chú ý chỉ ra nguồn tác giả hoặc đưa mã nguồn vào nếu muốn phân phối lại chương trình sẵn sàng để chạy.

Dù về cơ bản *PMTD* và *PMSHĐQ* khác nhau về giấy phép theo đó các tác giả xuất bản các chương trình của họ, thì điều quan trọng để nhấn mạnh là sự khác biệt này được phản ánh trong những điều kiện hoàn toàn khác nhau về sử dụng và phân phối lại. Như chúng ta đã thấy trong vài năm gần đây, điều này không chỉ tạo ra những phương pháp phát triển hoàn toàn khác nhau, mà còn, trên thực tế, đối nghịch nhau (theo nhiều khía cạnh) đối với việc hiểu về IT.

Các luật về sở hữu trí tuệ đảm bảo rằng trong sự thiếu vắng quyền hạn chắc chắn thì gần như không có gì có thể thực hiện được với một tác phẩm (trong trường hợp của chúng ta, là một chương trình) nhận được hoặc mua được. Chỉ có tác giả (hoặc người nắm giữ các quyền của tác phẩm) có thể trao cho chúng ta quyền đó. Trong mọi trường hợp, quyền sở hữu về tác phẩm không thay đổi bởi việc trao một giấy phép, vì điều này không dẫn tới sự chuyển giao quyền tác giả, mà chỉ là quyền sử dụng, và trong một số trường hợp (bắt buộc với PMTD), sự phân phối và sửa đổi. Các giấy phép của PMTD là khác với các giấy phép của PMSHĐQ chính xác là ở chỗ thay vì việc hạn chế một cách cẩn thận những gì được phép, thì nó tiến hành sự cho phép một cách rõ ràng chắc chắn. Khi mọi người nhận được một chương trình tự do thì họ có thể phân phối lại nó hoặc không, nhưng nếu họ phân phối lại nó, thì họ chỉ có thể làm thế được vì giấy phép cho phép điều này. Nhưng để làm vậy thì phải xem tới giấy phép.

Quả thực, giấy phép chứa các qui định sử dụng mà người sử dụng, nhà phân phối, nhà tích hợp và tất cả các bên tham gia khác có liên quan trong thế giới IT phải xem tới.

Để hiểu đầy đủ tất cả những đầu vào đầu ra của pháp luật được đề cập tới trong chương này (và những điều không có câu hỏi mà rất quan trọng để hiểu được bản chất tự nhiên của PMTD) chúng ta cũng phải nhận thức được rằng mỗi phiên bản mới của một chương trình được coi là một tác phẩm. Tác giả, một lần nữa, có toàn quyền để làm những gì anh ta muốn với nó, ngay cả để phân phối nó theo những điều khoản và điều kiện hoàn toàn khác (nói cách khác, với một giấy phép hoàn toàn khác so với giấy phép trước đó). Bằng cách đó nếu độc giả là tác giả duy nhất của một chương trình, anh ta có thể xuất bản một phiên bản theo một giấy phép của PMTD và, nếu anh ta muốn, giấy phép sau đó theo một giấy phép sở hữu độc quyền.

Trong trường hợp có nhiều tác giả hơn và phiên bản mới có chứa mã nguồn mà họ đã làm ra, nếu mã nguồn đó sẽ được xuất bản theo những điều kiện khác, thì tất cả họ sẽ phải phê chuẩn sự thay đổi này trong giấy phép.

Một vấn đề còn khá mở là giấy phép mà áp dụng cho những đóng góp từ bên ngoài. Thường thì giả thiết là ai đó mà đóng góp cho một dự án chấp nhận một cách *de facto* rằng sự đóng góp của anh ta hoặc chị ta điều chỉnh theo những điều kiện được chỉ định bởi giấy phép của nó, dù những nền tảng pháp lý cho điều này là kém. Sáng kiến của FSF yêu cầu bằng phương tiện của một bức thư (vật lý) cho sự nhượng lại tất cả *bản quyền* từ bất kỳ ai mà đóng góp hơn 10 dòng mã lệnh cho một dự án phụ của GNU là một biểu thị rõ ràng rằng trong thế giới của PMTD có những chính sách khắc nghiệt hơn về những đóng góp này.

Dựa trên những điều ở trên, trong phần còn lại của chương này chúng ta sẽ tập trung vào việc phân tích các giấy phép khác nhau. Để đặt phân tích này vào đúng ngữ cảnh, chúng ta phải nhớ rằng từ nay trở đi, khi chúng ta nói rằng một giấy phép là một giấy phép của PMTD, những gì chúng ta hàm ý là việc nó làm thỏa mãn những định nghĩa của *PMTD* đã được trình bày trong phần 1.1.1.

3.2.1 Các loại giấy phép

Có nhiều loại giấy phép tự do, dù vì lý do thực tế hầu hết các dự án sử dụng một nhóm nhỏ khoảng 4 hoặc 5 giấy phép. Một mặt, nhiều dự án không muốn hoặc không có tài nguyên để thiết kế giấy phép của riêng họ; mặt khác, hầu hết những người sử dụng yêu thích một giấy phép nổi tiếng hơn là phải đọc và phân tích các giấy phép hoàn chỉnh.

Thư mục tham khảo

Có một sự biên tập và thảo luận về các giấy phép được cho là không tự do hoặc tự do nhưng không tương thích với GPL từ quan điểm của FSF trong FSF, "Các giấy phép Tự do" [121]. Sự khác biệt về triết lý theo quan điểm của OSI được thể hiện trong danh sách của nó (Sáng kiến Nguồn Mở, "Các giấy phép Nguồn Mở") [181]. Chúng ta có thể thấy sự không nhất quán trong một số giấy phép, như Giấy phép Nguồn Công cộng Apple phiên bản 1.2 (APSL v1.2), mà FSF coi là không tự do vì sự bắt buộc phải xuất bản tất cả những thay đổi (ngay cả nếu chúng là riêng tư), để lưu ý Apple về sự phân phối lại, hoặc khả năng hủy bỏ đơn phương. Dù vậy, sức ép của sự phân loại này đã buộc Apple xuất bản phiên bản 1.0 của hãng vào tháng 08/2003, mà FSF sau đó đã coi là tự do.

Có thể phân chia các giấy phép của PMTD thành 2 họ lớn. Họ đầu bao gồm các giấy phép mà không đặt ra những điều kiện đặc biệt lên việc *phân phối lại lần 2* (nói cách khác là chỉ chỉ định rằng phần mềm có thể được phân phối lại hoặc sửa đổi, nhưng điều đó không đặt ra những điều kiện đặc biệt để làm như vậy, mà cho phép, ví dụ như, ai đó nhận được chương trình rồi sau đó phân phối lại nó như một PMSHĐQ): chúng là những gì mà chúng ta sẽ tham khảo như là các *giấy phép dễ dãi*. Họ thứ 2, mà chúng ta sẽ gọi là các *giấy phép mạnh* (hoặc các giấy phép copyleft), bao gồm những giấy phép mà chúng ở dạng của GNU GPL, áp đặt những điều kiện trong trường hợp muốn phân phối lại phần mềm, có mục đích để đảm bảo tuân thủ với những điều kiện của giấy phép sau lần *phân phối lại lần đầu*. Trong khi nhóm đầu nhấn mạnh tới sự tự do của người nhận chương trình để làm hầu như mọi thứ anh hoặc chị ta muốn với nó (về những điều kiện cho sự phân phối lại trong tương lai), thì nhóm 2 lại nhấn mạnh tới sự tự do của bất kỳ ai mà có thể có tiềm năng nhận, rồi một ngày nào đó, một tác phẩm dẫn xuất từ chương trình đó xuất hiện, thì buộc những sửa đổi và phân phối lại đến sau phải tôn trọng những điều khoản của giấy phép gốc ban đầu. Sự khác biệt giữa 2 loại giấy phép này từng là (và vẫn còn là) một vấn đề gây tranh cãi trong cộng đồng PMTD. Trong mọi trường hợp, chúng ta nên nhớ rằng

chúng tất cả đều là những giấy phép tự do.

Lưu ý

Khái niệm *copyleft* khi áp dụng cho một giấy phép, được sử dụng chủ yếu bởi FSF để tham chiếu tới những giấy phép của riêng Quỹ này, có ngụ ý tương tự như những giấy phép được tham chiếu tới như là những *giấy phép mạnh* được sử dụng trong văn bản này.

3.2.2 Những giấy phép dễ dãi

Những giấy phép dễ dãi, cũng còn đôi khi được biết như là những giấy phép *hào phóng* hoặc *tối thiểu*, hầu như không áp đặt bất kỳ điều kiện nào lên người nhận phần mềm, và vâng, cho phép sử dụng, phân phối lại và sửa đổi. Từ quan điểm cụ thể nào đó, tiếp cận này có thể được nhìn nhận như một sự đảm bảo sự tự do tối đa cho người nhận chương trình. Nhưng từ quan điểm khác, nó cũng có thể được hiểu như sự cẩu thả tối đa xét về việc đảm bảo rằng một khi ai đó nhận được chương trình, thì người đó đảm bảo những quyền tự do y hệt khi phân phối lại chương trình đó. Trong thực tế, những giấy phép này thường cho phép phần mềm mà tác giả của nó phân phối theo một giấy phép dễ dãi để được phân phối lại bằng một giấy phép sở hữu độc quyền.

Trong số những giấy phép này, thì giấy phép BSD là nổi tiếng nhất, ở một chừng mực nào đó thì những giấy phép dễ dãi thường sẽ được tham chiếu tới như các giấy phép *loại BSD*. Giấy phép BSD (Berkeley Software Distribution) bắt nguồn từ xuất bản phẩm của các phiên bản khác nhau của Unix của Đại học Berkeley, California tại Mỹ. Bổn phận duy nhất mà nó đưa ra là để công nhận các tác giả, trong khi nó cho phép phân phối lại ở cả các định dạng nhị phân và mã nguồn, mà không bắt ép theo bất kỳ cách nào trong mọi trường hợp. Nó cũng trao quyền để thực hiện bất kỳ sự thay đổi nào và được tích hợp vào trong các chương trình khác mà hầu như không có bất kỳ hạn chế nào.

Lưu ý

Một trong những hệ quả trong thực tế của các giấy phép loại BSD từng là để truyền bá các chuẩn, vì các lập trình viên thấy không có cản trở cho việc làm cho các chương trình tương thích được với một triển khai cài đặt tham chiếu theo dạng giấy phép này. Trong thực tế, đây là một trong những lý do cho sự khác thường và lan tỏa nhanh chóng của các giao thức Internet và giao diện lập trình dựa trên các khe cắm (socket), vì hầu hết các lập trình viên thương mại đã dẫn xuất triển khai cài đặt của họ từ một giao diện lập trình của Đại học Berkeley.

Các giấy phép dễ dãi khá phổ biến, và có hẳn một họ với những đặc tính tương tự như đối với BSD: X Window, Tcl/Tk, Apache, … Về mặt lịch sử, những giấy phép này đã xuất hiện vì các phần mềm tương ứng đã được phát triển bởi các trường đại học bằng các dự án nghiên cứu được tài trợ bởi Chính phủ Mỹ. Các trường đại học đã không bán các chương trình này, với giả thiết rằng chúng đã được trả tiền bởi Chính phủ, và vì thế bởi người trả thuế, mà nó có nghĩa rằng bất kỳ công ty hoặc cá nhân nào cũng có thể sử dụng những phần mềm này mà hầu như không có bất kỳ hạn chế nào.

Như đã được nhắc tới, trên cơ sở của một chương trình được phân phối theo một giấy phép dễ dãi thì một giấy phép khác có thể sẽ được tạo ra (trong thực tế, một phiên bản mới), mà nó có thể sẽ là sở hữu độc quyền. Những chỉ trích về các giấy phép BSD cho thấy một sự nguy hiểm trong tính năng này, vì

nó không đảm bảo sự tự do của các phiên bản của chương trình trong tương lai. Những người ủng hộ nó, ngược lại, coi nó là sự thể hiện tối đa sự tự do và viện lý rằng, cuối cùng, bạn có thể làm (hầu như) mọi thứ mà bạn muốn với phần mềm này.

Hầu hết các giấy phép dễ dãi theo từng từ một là bản sao của một giấy phép gốc của Berkeley, sửa đổi đi chỉ những gì tham chiếu tới quyền tác giả. Trong một số trường hợp, như giấy phép của dự án Apache, nó đưa vào một mệnh đề bổ sung, như việc cấm đưa ra cùng một cái tên như bản gốc đối với các phiên bản được phân phối lại. Tất cả những giấy phép này thường đưa vào, như BSD, sự cấm sử dụng tên của đúng người giữ bản quyền nhằm khuyến khích những sản phẩm dẫn xuất.

Cùng lúc, tất cả các giấy phép này, dù là loại BSD hay không, cũng đưa vào một hạn chế của sự đảm bảo mà nó thực sự là một sự khước từ, cần thiết để tránh những khiếu nại pháp lý cho những đảm bảo tuyệt đối. Dù sự khước từ này trong PMTD từng bị chỉ trích một cách rộng rãi, thì nó cũng là một thực tế chung với cả PMSHĐQ, mà nó thường chỉ đảm bảo rằng phương cách này là đúng và rằng chương trình chạy được trong nghi hoặc.

Phác thảo tóm lược về giấy phép BSD

Copyright © [của *người chủ sở hữu*]. All rights reserved.

Việc phân phối lại và sử dụng ở các dạng mã nguồn và nhị phân, có hoặc không có sự sửa đổi, là được phép miễn là những điều kiện sau đây được đáp ứng:

1. Việc phân phối lại mã nguồn phải tái sinh lưu ý về bản quyền, và liệt kê những điều kiện và sự khước từ này.
2. Việc phân phối lại ở dạng nhị phân phải tái sinh lưu ý về bản quyền và liệt kê những điều kiện và sự khước từ này trong tài liệu được đưa ra.
3. Tên của chủ sở hữu và tên của những người đóng góp có thể không được sử dụng để xác nhận các sản phẩm dẫn xuất từ phần mềm này nếu không có quyền.

Chương trình này được cung cấp "như nó có", và bất kỳ sự cho phép nào dù rõ ràng hay tiềm ẩn, về tính có thể thương mại được hoặc phù hợp được cho một mục đích cụ thể nào đó sẽ bị khước từ. Không trong sự kiện nào, *người chủ sở hữu* có bổn phận đối với bất kỳ thiệt hại nào được gây ra bởi việc sử dụng của nó (bao gồm cả việc mất dữ liệu, mất lợi nhuận hoặc gián đoạn việc kinh doanh).

Tiếp theo chúng ta mô tả ngắn gọn một ít giấy phép dễ dãi:

- Giấy phép X Window, phiên bản 11 (X11) (http://www.x.org/Downloads_terms.html) [73]. Đây là giấy phép được sử dụng để phân phối hệ thống X Window, hệ thống các cửa sổ được sử dụng rộng rãi nhất trong thế giới các môi trường của Unix, và cũng của GNU/Linux. Nó rất tương tự như giấy phép BSD, mà nó cho phép phân phối lại, sử dụng và sửa đổi mà thực tế không có bất kỳ hạn chế nào. Đôi khi nó được gọi là giấy phép MIT (với một sự thiếu chính xác nguy hiểm, vì MIT đã sử dụng các dạng giấy phép khác). Các công việc dẫn xuất từ hệ thống X Window, như XFree86 cũng được phân phối theo giấy phép này.

- Zope Public License 2.0 (http://www.zope.org/Resources/ZPL) [76]. Giấy phép này (thường được tham chiếu như là ZPL) được sử dụng cho sự phân phối Zope (một máy chủ ứng dụng) và các sản phẩm liên quan khác. Nó tương tự như BSD, với tính năng kỳ lạ rằng nó chỉ cốt để cấm sử dụng các thương hiệu được đăng ký bởi tập đoàn Zope.
- Giấy phép Apache

  Đây là giấy phép mà theo đó hầu hết các chương trình được sản xuất bởi dự án Apache được phân phối. Nó tương tự như giấy phép BSD. Có một số chương trình tự do mà không được phân phối với một giấy phép cụ thể, hay đúng hơn là tác giả công bố một cách rõ ràng chúng thuộc về *miền công cộng*. Hệ quả chính của tuyên bố này là việc tác giả khước từ mọi quyền đối với chương trình này, mà nó có thể vì thế được sửa đổi, phân phối lại, sử dụng, …, theo bất kỳ cách nào. Theo những điều khoản thực tế, nó rất tương tự như một chương trình theo một giấy phép loại BSD.

3.2.3 Các giấy phép mạnh

GNU General Public Licence (GNU GPL)

Giấy phép Công cộng Chung GNU (FSF, 1991) [118] (được biết tốt hơn bằng cái tên tắt là GPL), nó có trong phụ lục C, cho tới nay là giấy phép phổ biến và nổi tiếng nhất trong thế giới của PMTD. Nó đã được tạo ra bởi FSF (người sáng lập ra dự án GNU), và ban đầu đã được thiết kế để trở thành giấy phép cho tất cả các phần mềm được tạo ra bởi FSF. Tuy nhiên, việc sử dụng nó đã mở rộng xa hơn trở thành giấy phép được sử dụng nhiều nhất (ví dụ, hơn 70% các dự án đã công bố trên Freshmeat được cấp phép theo GPL), ngay cả bởi các dự án hàng đầu trong thế giới của PMTD, như nhân Linux.

Giấy phép GPL là thú vị từ một quan điểm pháp lý vì nó tạo ra sự sử dụng một cách sáng tạo pháp luật về bản quyền: thay vì việc giới hạn các quyền của người sử dụng, nó đảm bảo cho họ. Vì lý do này, hành vi này thường được gọi là copyleft (một cách chơi chữ thay thế cho từ “phải” bằng “trái”). Ai đó có khiếu hài hước đã còn nghĩ ra khẩu hiệu “copyleft, mọi quyền được nghịch đảo”.

Trong những điều khoản cơ bản, giấy phép GPL cho phép phân phối lại ở dạng nhị phân và dạng mã nguồn, dù trong trường hợp sự phân phối lại bằng mã nhị phân thì sự truy cập tới mã nguồn cũng là bắt buộc. Nó còn cho phép những sửa đổi được thực hiện mà không có bất kỳ hạn chế nào. Tuy nhiên, điều này chỉ có thể để phân phối lại mã nguồn được cấp phép theo GPL tích hợp được với mã nguồn khác (ví dụ, việc liên kết mã nguồn) nếu nó có một giấy phép tương thích. Điều này từng được gọi là *hiệu ứng virus* (dù nhiều người coi cái tên này là bất kính) của GPL, vì một khi mã nguồn đã được xuất bản với các điều kiện này thì chúng có thể không bao giờ thay đổi được.

Lưu ý

Một giấy phép là không tương thích với GPL khi nó hạn chế bất kỳ các quyền nào được đảm bảo bởi GPL, hoặc dứt khoát bằng việc phủ nhận bất kỳ mệnh đề nào của nó, hoặc hoàn toàn, bằng việc áp đặt một hạn chế mới. Ví dụ, giấy phép BSD hiện hành là tương thích, nhưng giấy phép Apache là không tương thích vì nó đòi hỏi rằng các tư liệu công khai chỉ để nhắc rằng công việc kết hợp chứa mã nguồn của từng và mỗi người trong số những người nắm giữ các quyền. Điều này không ngụ ý rằng các chương

> trình với cả 2 giấy phép không thể được sử dụng cùng một lúc, hoặc ngay cả được tích hợp với nhau. Nó chỉ có nghĩa là những chương trình được tích hợp không thể được phân phối, vì nó không thể tuân thủ cùng một lúc với những điều kiện phân phối lại của cả 2.

Giấy phép GPL được thiết kế để đảm bảo sự tự do của mã nguồn vĩnh viễn, vì một chương trình được xuất bản và cấp phép theo những điều kiện của nó có thể không bao giờ trở thành sở hữu độc quyền được. Hơn nữa, chương trình đó và những sửa đổi của nó có thể không xuất bản được với một giấy phép khác ngoài GPL. Như đã được nhắc tới, những người ủng hộ các giấy phép loại BSD cho là có một sự hạn chế về tự do trong mệnh đề này, trong khi những người ủng hộ GPL thì tin tưởng rằng nó là cách để đảm bảo rằng phần mềm này sẽ luôn là tự do. Một cách để xem xét nó có thể là coi giấy phép GPL tối đa hóa sự tự do của người sử dụng, trong khi giấy phép BSD tối đa hóa sự tự do của các lập trình viên. Tuy nhiên, chúng ta nên lưu ý rằng trong trường hợp thứ 2 thì chúng ta đang tham chiếu tới các lập trình viên nói chung và không tham chiếu tới các tác giả, vì nhiều tác giả coi giấy phép GPL sẽ là vì quyền lợi của họ nhiều hơn, vì nó bắt các đối thủ cạnh tranh phải xuất bản những sửa đổi của họ (những cải tiến, sửa lỗi cho đúng, …) nếu họ phân phối lại phần mềm của họ, trong khi với giấy phép loại BSD thì điều này không là trường hợp cần thiết.

Như bản chất tự nhiên đối nghịch của giấy phép này so với *bản quyền*, chính là vì triết lý của nó (và là của FSF) là việc phần mềm phải không có chủ sở hữu (Richard Stallman, "Vì sao phần mềm phải không có chủ sở hữu", 1998) [207]. Dù nó là đúng rằng phần mềm được cấp phép theo GPL có một tác giả, là người mà cho phép pháp luật về bản quyền áp dụng cho nó, còn những điều kiện mà theo đó nó được xuất bản ban cho một sự tự nhiên như vậy lên phần mềm mà chúng ta có thể coi quyền sở hữu sẽ truyền sang cho người có quyền sở hữu phần mềm chứ không phải là sang cho người đã tạo ra nó.

Tất nhiên, giấy phép này cũng đưa vào những *khước từ* để bảo vệ các tác giả. Cũng như vậy, để bảo vệ uy tín tốt của các tác giả gốc ban đầu, bất kỳ sự sửa đổi nào của tệp nguồn phải được đưa vào một lưu ý chỉ ra ngày tháng và tác giả của sự sửa đổi đó.

GPL cũng tính tới các bằng sáng chế phần mềm, và yêu cầu rằng nếu mã nguồn có chứa các thuật toán được cấp bằng sáng chế (như chúng ta đã nói tới, thứ gì đó chung và hợp pháp tại Mỹ, nhưng hiện lại không hợp pháp tại châu Âu), thì hoặc là một giấy phép để sử dụng bằng sáng chế không mất tiền phải được trao, hoặc nó không thể được phân phối theo GPL.

Phiên bản mới nhất của giấy phép GPL, phiên bản 2, đã được xuất bản năm 1991 (dù lúc viết phiên bản 3 là trong một giai đoạn chuẩn bị tiến bộ hơn). Đặc biệt ghi nhớ trong đầu các phiên bản trong tương lai, giấy phép này khuyến cáo việc cấp phép theo những điều kiện của phiên bản 2 hoặc bất kỳ phiên bản sau nào khác được xuất bản bởi FSF, mà là những gì mà nhiều tác giả làm. Tuy nhiên, những người khác, bao gồm cả cụ thể Linus Torvalds (người tạo ra Linux), chỉ xuất bản phần mềm của họ theo các điều kiện của phiên bản 2 của GPL, tự tách mình xa khỏi những tiến bộ tiềm tàng trong tương lai của FSF.

Phiên bản 3 của GPL (http://gplv3.fsf.org) [115] dự kiến sẽ cập nhật nó cho kịch bản phần mềm hiện hành, chủ yếu về phương diện của các khía cạnh như là các bằng sáng chế, DRM (quản lý các quyền số) và những hạn chế khác về sự tự do của phần mềm. Ví dụ, phác thảo hiện sẵn có vào thời điểm đang viết (tháng 05/2007) không cho phép một nhà sản xuất phần cứng khóa việc sử dụng các module phần mềm cụ thể nào đó nếu chúng không mang chữ ký số gán vào một tác giả xác định nào đó.

Một ví dụ về thực tế này xảy ra với các đầu ghi video số TiVo, mà chúng cung cấp mã nguồn cho tất cả phần mềm của chúng (được cấp phép với GPLv2) cùng một lúc khi chúng không cho phép những sửa đổi mã nguồn sẽ được sử dụng trong các phần cứng[4] đó.

Giấy phép này cũng không cho phép các phần mềm được chạy một cách ép buộc trong một môi trường được thiết lập sẵn trước, như việc xảy ra khi sử dụng các nhân không được ký sẽ bị cấm phân phối và coi đó là phù hợp vì những lý do an ninh.

Lưu ý

Có một vài điểm trong giấy phép GPLv3 đã tạo ra một mức độ đối lập nào đó. Một trong những nhóm đối lập là nhóm các lập trình viên phát triển nhân Linux (bao gồm cả bản thân Linus Torvalds). Họ cho rằng yêu cầu sử dụng các thành phần phần mềm được ký cho phép những tính năng an ninh cụ thể nào đó sẽ được đưa ra mà nếu khác sẽ là không thể, cùng lúc khi sự ngăn cấm rõ ràng của chúng có thể sẽ mở rộng giấy phép này tới lĩnh vực của phần cứng. Hơn nữa, sự hạn chế này được thiết lập bởi cơ chế của các chữ ký có thể chỉ xảy ra với các nền tảng phần cứng và phần mềm mà chúng được thiết kế theo cách đó, nghĩa là phần mềm có thể được sửa đổi cho việc sử dụng nó trên các phần cứng khác nhau. Về điểm này, FSF đồng ý sử dụng cơ chế chữ ký mà nó khuyến cáo chống lại việc sử dụng các thành phần không được ký vì những lý do an ninh, nhưng tin tưởng việc không cấm những cơ chế chữ ký mà chúng ngăn cản việc sử dụng các thành phần không được ký, có thể dẫn tới các kịch bản nơi mà có thể có những nền tảng phần cứng hoặc phần mềm mà trên đó sẽ chạy những sửa đổi phần mềm, nghĩa là sự tự do của PMTD vì thế có thể trở thành hoàn toàn bị hạn chế ở những nơi mà việc sửa đổi mã nguồn được quan tâm.

The GNU Lesser General Public Licence (GNU LGPL)

Giấy phép Công cộng Chung Nhỏ GNU (FSF, GNU LGPL, phiên bản 2.1, 02/1999) [119], thường được tham chiếu tới từ những chữ cái đầu là LGPL, là một giấy phép khác của FSF. Ban đầu được thiết kế để sử dụng nó với các thư viện (chữ L, ban đầu có nghĩa là *thư viện – library*), gần đây nó đã được sửa để được coi là cô em nhỏ - little sister (lesser) của GPL.

LGPL cho phép các chương trình tự do được sử dụng với các PMSHĐQ. Bản thân chương trình này được phân phối lại dường như nó là theo giấy phép GPL, nhưng sự tích hợp của nó với bất kỳ gói phần mềm nào khác được cho phép mà hầu như không có bất kỳ hạn chế nào. Như chúng ta có thể thấy, ban đầu giấy phép này đã có mục đích hướng vào các thư viện, để khuyến khích việc sử dụng và phát triển chúng mà không gặp phải những vấn đề về tích hợp được ám chỉ bởi GPL. Tuy nhiên, khi nó đã được tung ra thì mục tiêu theo đuổi để làm cho các thư viện tự do phổ biến đã không được bù đắp bởi thế hệ các chương trình tự do, nên FSF đã quyết định thay đổi chữ thư viện thành cô em nhỏ và đã khuyến cáo chống lại việc sử dụng nó, ngoại trừ trong những hoàn cảnh rất đặc biệt và theo từng vụ việc. Ngày nay, nhiều chương trình mà chúng không phải là các thư viện được cấp phép theo những điều khoản của LGPL. Ví dụ, trình duyệt của Mozilla hoặc bộ phần mềm văn phòng OpenOffice.org cũng được cấp phép theo LGPL.

4 Trường hợp này đã gợi ra việc sử dụng từ tivoisation (tivo hóa) đối với các trường hợp tương tự đã xảy ra.

Lưu ý

Như trường hợp với GPL, phiên bản được xuất bản mới nhất của LGPL là phiên bản 2, dù đã có một mẫu template của phiên bản 3 (http://gplv3.fsf.org/pipermail/info-gplv3/2006-July/000008.html) [116]. Phiên bản mới này là ngắn hơn phiên bản trước vì nó tham chiếu tất cả văn bản của nó tới GPLv3 và chỉ nhấn mạnh tới những khác biệt của nó.

Các giấy phép mạnh khác

Các giấy phép mạnh khác đáng để lưu ý gồm:

- Giấy phép con mèo ngủ Sleepycat (www.sleepycat.com/download/oslicense.html) [59].

  Đây là giấy phép theo đó công ty Sleepycat (http://www.sleepycat.com/) [60] đã phân phối các chương trình của hãng (trong đó chúng ta có thể nhắc tới Berkeley DB nổi tiếng). Nó ép buộc một số điều kiện cụ thể bất kỳ khi nào chương trình hoặc công việc dẫn xuất từ chương trình được phân phối lại. Đặc biệt, nó bắt mã nguồn phải được đưa ra, (bao gồm cả những sửa đổi trong trường hợp của một công việc dẫn xuất) và để phân phối lại thì buộc phải đưa vào cùng những điều kiện y hệt cho người nhận. Dù nó ngắn hơn nhiều so với GNU GPL, thì nó là rất giống theo những hiệu lực chính.

- eCos License 2.0 (http://www.gnu.org/licenses/ecos-license.html) [25]. Đây là giấy phép mà theo đó eCos (http://sources.redhat.com/ecos/) [24], một hệ điều hành thời gian thực, được phân phối. Nó là một sửa đổi của GNU GPL mà không coi mã nguồn được liên kết tới các chương trình mà nó bảo vệ, phải tuân theo các điều khoản của GNU GPL nếu được phân phối lại. Từ quan điểm này, những hiệu lực của nó là tương tự như của GNU LGPL.

- Affero General Public License (http://www.affero.org/oagpl.html) [78]. Đây là một sửa đổi thú vị của GNU GPL mà nó coi trường hợp các chương trình đưa ra các dịch vụ thông qua web, hoặc nói chung, thông qua các mạng máy tính. Dạng chương trình này đại diện cho một vấn đề từ quan điểm của các giấy phép mạnh. Vì việc sử dụng chương trình không ngụ ý phải nhận nó thông qua một sự phân phối lại, ngay cả dù nó được cấp phép theo GNU GPL, ví dụ vậy, mà không có việc phân phối lại theo bất kỳ cách thức nào, và vì thế, không có bổn phận, ví dụ vậy, phải phân phối mã nguồn của nó. Affero GPL có một mệnh đề bắt buộc rằng nếu chương trình có một phương tiện cho việc đưa ra mã nguồn của nó thông qua web cho bất kỳ ai sử dụng nó; tính năng này có thể không được vô hiệu hóa.

  Điều này có nghĩa là nếu tác giả gốc ban đầu đưa khả năng này vào trong mã nguồn, thì bất kỳ người sử dụng nào cũng có thể có được nó, và cộng với việc phân phối lại phải tuân thủ các điều kiện của giấy phép này. FSF đang xem xét đưa vào các khoản tương tự vào phiên bản 3 của GNU GPL.

- IBM Public License 1.0 (http://oss.software.ibm.com/developerworks/opensource/license10.html) [40]. Đây là một giấy phép mà nó cho phép một sự phân phối lại nhị phân của các công việc dẫn xuất chỉ nếu (giữa những điều kiện khác) một cơ chế được định sẵn trước cho người nhận chương trình để nhận mã nguồn. Sự phân phối lại mã nguồn phải được thực hiện theo cùng giấy phép này. Giấy phép

này cũng thú vị vì nó bắt bên tham gia phân phối lại chương trình với những sửa đổi phải cấp phép tự động và không được lấy tiền đối với bất kỳ bằng sáng chế nào có ảnh hưởng tới những sửa đổi như vậy và chúng là sở hữu của người phân phối cho bên nhận chương trình.

- Mozilla Public License 1.1 (http://www.mozilla.org/MPL/MPL-1.1.html) [49]. Đây là một ví dụ về giấy phép tự do được thiết kế bởi một công ty. Đây là một sự tiến bộ của giấy phép tự do đầu tiên mà Netscape Navigator đã có, mà nó đã từng là rất quan trọng trong những ngày đó vì nó đã là lần đầu tiên mà một công ty nổi tiếng đã quyết định phân phối một chương trình theo giấy phép tự do của riêng hãng.

3.2.4 Phân phối theo vài giấy phép

Cho tới nay chúng ta đã giả thiết rằng mỗi chương trình được phân phối theo chỉ một giấy phép duy nhất mà nó chỉ định các điều kiện để sử dụng và phân phối. Tuy nhiên, một tác giả có thể phân phối các công việc theo các giấy phép khác nhau. Để hiểu được điều này, chúng ta phải nhớ rằng mỗi xuất bản phẩm là một công việc mới, và những phiên bản khác nhau đó có thể được phân phối với sự khác biệt chỉ trong giấy phép của chúng. Như chúng ta sẽ thấy, hầu hết thời gian điều này hiểu theo sự việc rằng phụ thuộc vào việc người sử dụng muốn gì để làm với phần mềm mà anh ta sẽ phải xem tới những điều khoản của một giấy phép này hoặc giấy phép khác.

Một trong những ví dụ nổi tiếng nhất của giấy phép đôi là một ví dụ đối với thư viện của Qt, trong đó môi trường cho máy tính để bàn KDE được thiết lập. Trolltech, một công ty có trụ sở ở Nauy, đã phân phối Qt bằng một giấy phép sở hữu độc quyền, dù hãng đã khước từ thanh toán cho các chương trình đã không sử dụng nó vì lợi nhuận. Vì lý do này và vì các đặc tính kỹ thuật của nó, đây từng là sự lựa chọn của dự án KDE vào những năm giữa thập kỷ 90. Điều này đã gây ra một sự tranh cãi với FSF vì sau đó KDE đã dừng việc là PMTD một cách hoàn toàn, vì nó đã phụ thuộc vào một thư viện sở hữu độc quyền. Tiếp theo tranh cãi rộng rãi (trong khi GNOME đã xuất hiện như là đối thủ cạnh tranh tự do của KDE trong môi trường máy tính để bàn), thì Trollech đã quyết định sử dụng hệ thống giấy phép đôi cho sản phẩm ngôi sao của hãng: các chương trình theo GPL có thể sử dụng một phiên bản Qt GPL, trong khi nếu dự định mà là sẽ tích hợp nó với các chương trình mà đã có các giấy phép không tương thích với GPL (như các giấy phép sở hữu độc quyền), thì một giấy phép đặc biệt phải được mua từ họ. Giải pháp này đã làm thỏa mãn được cho tất cả các bên, và cho tới nay KDE được cho là PMTD.

Những ví dụ nổi tiếng khác về các giấy phép đôi là StarOffice và OpenOffice.org, hoặc Netscape Communicator và Mozilla. Trong cả 2 trường hợp, sản phẩm đầu tiên là sở hữu độc quyền trong khi sản phẩm thứ 2 lại là một phiên bản tự do (thường theo những điều kiện của một vài giấy phép tự do). Dù các dự án tự do ban đầu được hạn chế thì những phiên bản sở hữu độc quyền là anh chị em của chúng, cùng với thời gian chúng đã đi theo con đường của riêng chúng, nghĩa là ngày nay chúng có một mức độ độc lập cao.

3.2.5 Tài liệu chương trình

Tài liệu đi với một chương trình tạo thành một phần không thể thiếu của nó, khi đưa ra các dẫn giải về

mã nguồn, như được thừa nhận, ví dụ bởi Luật Sở hữu Trí tuệ của Tây Ban Nha. Đưa ra mức tích hợp này, có thể coi là logic mà những tự do y hệt phải được áp dụng cho tài liệu và nó phải tiến bộ theo hệt cách thức như với chương trình: bất kỳ sửa đổi nào được thực hiện trong một chương trình đòi hỏi thay đổi cùng một lúc và luôn được cập nhật trong tài liệu của nó.

Hầu hết tài liệu này có xu hướng được thực hiện như những tệp văn bản không có định dạng, vì mục tiêu là làm cho nó truy cập được một cách vạn năng bằng một môi trường tối thiểu các công cụ, và (trong trường hợp của các chương trình tự do) thường đưa vào một lời giới thiệu nhỏ cho chương trình (tệp README), các chỉ dẫn cài đặt (tệp INSTALL), một vài lịch sử về sự tiến hóa và tương lai của chương trình (tệp CHANGELOG và TODO), các tác giả và bản quyền (tệp AUTHORS và COPYRIGHT hoặc COPYING), cũng như những chỉ dẫn sử dụng. Tất cả những thứ này, bao gồm cả các tác giả và bản quyền, phải là tự do và có thể sửa đổi được khi chương trình tiến hóa. Đối với các tác giả thì chúng ta chỉ cần đưa vào những cái tên và những ủy quyền mà không có việc hạn chế bất kỳ thứ gì, và bản quyền chỉ phải được sửa đổi nếu các điều kiện cho phép.

Những chỉ dẫn sử dụng thường được ghi trong những định dạng phức tạp hơn, vì chúng có xu hướng sẽ là những tài liệu dài hơn và giàu định dạng hơn.

PMTD đòi hỏi rằng tài liệu có thể thay đổi được một cách dễ dàng, mà tới lượt nó bắt tuân theo sự sử dụng của cái gọi là các định dạng minh bạch, với những đặc tả kỹ thuật được biết và có khả năng xử lý được bằng các công cụ tự do, như, bổ sung cho văn bản thuần khiết và sạch sẽ, định dạng của các trang sách chỉ dẫn của Unix, TexInfo, LaTeX hoặc DocBook, mà không có định kiến để cũng có khả năng phân phối kết quả của việc truyền những tài liệu nguồn này vào các định dạng phù hợp hơn cho việc hình dung và in ấn, như HTML, PDF hoặc RTF (thường là các định dạng mờ hơn).

Tuy nhiên, tài liệu chương trình thường được chuẩn bị bởi các bên thứ 3 mà họ không tham gia vào trong sự phát triển. Đôi khi tài liệu là một thứ về bản chất tự nhiên để dạy học, để tạo điều kiện thuận lợi cho sự cài đặt và sử dụng của một chương trình cụ thể (tệp HOWTO); đôi khi nó là tài liệu rộng hơn mà nó bao trùm vài chương trình và sự tích hợp của chúng, mà so sánh được các giải pháp, …, hoặc ở dạng của một sách chỉ dẫn tham chiếu hoặc sách trợ giúp học; đôi khi nó chỉ là một bản biên dịch của những câu hỏi thường gặp và những câu trả lời của chúng (tệp FAQ). Một ví dụng đáng lưu ý là dự án về tài liệu của Linux (http://www.tldp.org) [44]. Trong chủng loại này chúng ta cũng có thể đưa vào những tài liệu kỹ thuật khác, không nhất thiết về các chương trình, ví dụ như những chỉ dẫn cho việc bắc dây cho một mạng cục bộ, làm một cái lò sử dụng năng lượng mặt trời, sửa một động cơ hoặc chọn một nhà cung cấp các công cụ.

Những tài liệu này là nửa đường giữa tài liệu chương trình thuần túy và những bài báo hoặc những cuốn sách thực tiễn và kỹ thuật cao. Không có thành kiến đối với sự tự do để đọc, sao chép, sửa đổi và phân phối lại, tác giả có thể mong muốn đưa ra những ý kiến mà tác giả không muốn bị bóp méo, hoặc ít nhất không muốn bất kỳ sự bóp méo nào được quy cho tác giả; hoặc tác giả có thể muốn giữ những đoạn, như là những thừa nhận; hoặc làm cho nó có sức thuyết phục để sửa đổi những thứ khác, như là đầu đề chẳng hạn. Dù những mối quan tâm này cũng có thể nảy sinh với bản thân phần mềm, thì chúng không nên được trình bày một cách mãnh liệt trong thế giới của tài liệu tự do như trong thế giới của PMTD.

3.3 Tóm lược

Trong chương này chúng ta đã xem những khía cạnh pháp lý mà chúng chi phối hoặc ảnh hưởng tới thế giới của PMTD. Chúng tạo thành một phần của sở hữu công nghiệp hoặc sở hữu trí tuệ mà pháp luật nhận thức được, về nguyên tắc, để khuyến khích tính sáng tạo bằng việc tưởng thưởng cho những người sáng tạo trong một khoảng thời gian nhất định nào đó. Về các dạng khác nhau, cái gọi là *bản quyền* là thứ mà hầu như ảnh hưởng tới PMTD, một khi áp dụng một cách đúng mức thì nó có thể được sử dụng để đảm bảo sự tồn tại của PMTD ở dạng các giấy phép tự do. Chúng ta đã thấy tầm quan trọng của các giấy phép trong thế giới của PMTD. Và chúng ta cũng đã trình bày nhiều hình thái của các giấy phép đang tồn tại, những nền tảng theo đó chúng được xây dựng, những hậu quả, ưu và nhược điểm của chúng. Chắc chắn, chúng ta có thể nói rằng GPL cố gắng để tối đa hóa sự tự do của phần mềm cho người sử dụng, cho dù họ nhận được PMTD trực tiếp từ tác giả của nó hay không, trong khi các giấy phép loại BSD cố gắng tối đa hóa sự tự do của người sửa đổi hoặc người phân phối lại.

Đưa ra những gì mà chúng ta đã thấy trong chương này, chúng ta có thể suy luận rằng, rất quan trọng để quyết định sớm về việc chọn giấy phép nào cho một dự án sẽ phải nhận thức được đầy đủ về những ưu điểm và nhược điểm của nó, vì sự sửa đổi sau này có xu hướng sẽ là cực kỳ khó khăn, đặc biệt nếu số lượng những đóng góp từ bên ngoài là rất lớn.

Để kết luận, chúng ta muốn nhấn mạnh thực tế rằng PMTD và PMSHĐQ khác nhau duy nhất và độc nhất về giấy phép mà theo đó các chương trình được phân phối. Tuy nhiên, trong chương sau, chúng ta sẽ thấy rằng sự khác biệt về pháp lý thuần túy này có hoặc không thể ảnh hưởng tới cái cách mà phần mềm được phát triển, tạo ra một mô hình phát triển mới, mà có thể khác với các phương pháp phát triển "truyền thống" được sử dụng trong nền công nghiệp phần mềm ở một mức độ ít hơn hoặc nhiều hơn, phụ thuộc vào từng trường hợp một.

4 Các lập trình viên và những động lực của họ

> "Là một cao thủ lập trình (hacker) là rất thú vị, nhưng đây là một dạng thú vị mà cần nhiều nỗ lực. Nỗ lực tạo nên những động lực. Các vận động viên thành công tạo động lực của họ từ một dạng thích thú vật lý trong việc để cơ thể của họ thực hiện, trong việc tự thúc đẩy bản thân họ vượt qua được những giới hạn vật lý của riêng họ. Tương tự, để trở thành một cao thủ lập trình bạn phải tạo được một sự rung cảm cơ bản từ việc giải quyết được các vấn đề, mài sắc các kỹ năng của bạn và trau dồi tri thức của bạn".
>
> Eric Steven Raymond, "Làm thế nào để trở thành một cao thủ lập trình".

4.1 Giới thiệu

Phương thức một phần nặc danh và phân tán trong đó PMTD được phát triển đã có ý nghĩa là trong nhiều năm những tài nguyên của con người dựa vào vẫn còn chưa được biết tới một cách rộng rãi. Kết quả của việc thiếu sự hiểu biết này đã trở thành những chuyện được thần thoại hóa, ít nhất ở một mức độ nào đó, thế giới của PMTD và cuộc sống của những người đứng đằng sau nó, ít nhiều dựa trên những định kiến rộng rãi về văn hóa của các cao thủ lập trình và máy tính. Trong những năm vừa qua, cộng đồng các nhà khoa học đã thực hiện được một nỗ lực khổng lồ để biết được tốt hơn về những con người mà họ tham gia vào những dự án PMTD, những động lực, những nền tảng hàn lâm, và những khía cạnh liên quan tiềm tàng khác của họ. Từ quan điểm thực dụng thuần túy, việc biết ai có liên quan trong dạng các dự án này và vì sao, có thể sẽ cực kỳ hữu ích khi nói về việc tạo ra PMTD. Một số nhà khoa học, chủ yếu là các nhà kinh tế học, tâm lý học và xã hội học, đã muốn đi xa hơn và đã thấy được trong cộng đồng này mầm mống của các cộng đồng thực sự trong tương lai với những luật lệ và tôn ti trật tự của riêng họ, trong nhiều trường hợp hoàn toàn khác biệt với những gì chúng ta đã biết trong xã hội "truyền thống". Một trong những điều huyền bí quan trọng nhất phải giải quyết là học những gì đã tạo động lực cho các lập trình viên phần mềm để tham gia vào một cộng đồng tự nhiên này, đưa ra thực tế là những lợi ích về tài chính, ít nhất một cách trực tiếp, trên thực tế là không tồn tại, trong khi một cách gián tiếp thì khó để cân đong đo đếm được.

4.2 Các lập trình viên là những ai?

Phần này có mục đích để cung cấp một miêu tả khái quát mang tính toàn cầu về những người đã bỏ thời gian và công sức của họ ra để tham gia vào các dự án PMTD. Các dữ liệu mà chúng tôi chỉ ra có nguồn gốc chủ yếu từ nghiên cứu khoa học trong vài năm qua, đáng kể nhất nhưng không một chút nào độc chiếm cho việc đưa vào các *PMTD và PMNM*. Khảo sát và nghiên cứu, phần IV: "Khảo sát về các lập trình viên", 2002 [126], và "Ai đang làm nó? Biết thêm về các lập trình viên PMTD", 2001 [197].

Các lập trình viên phần mềm thường là những người trẻ tuổi. Tuổi trung bình là khoảng 27. Sự khác biệt về tuổi là đáng kể, vì nhóm áp đảo là độ tuổi từ 21 tới 24, và hầu hết thường xuất hiện giá trị là 23. Thật thú vị để lưu ý về tuổi tham gia phong trào PMTD nằm trong khoảng 18 và 25 và đặc biệt rõ ràng giữa 21 và 23, trùng khớp với độ tuổi của các trường đại học. Bằng chứng này đối nghịch với việc nói

rằng PMTD chủ yếu là thứ của thiếu niên, dù có một sự liên quan rõ ràng của thiếu niên (khoảng 20% các lập trình viên là dưới 20 tuổi). Để chắc chắn, những gì chúng tôi có thể thấy là việc hầu hết các lập trình viên (60%) là ở độ tuổi 20 của họ, với những người ở độ tuổi dưới 20 và trên 30 là như nhau và chia nhau 40% phần còn lại.

Từ độ tuổi tham gia này chúng ta có thể suy luận ra rằng có sự ảnh hưởng to lớn của các trường đại học lên PMTD. Điều này không ngạc nhiên, biết rằng như chúng ta đã thấy trong chương về lịch sử, trước khi PMTD được biết tới bởi cái tên đó thì nó đã có quan hệ chặt chẽ với nền giáo dục cao hơn. Ngay cả ngày nay, các nhóm người sử dụng là sinh viên và các trường đại học tiếp tục dẫn dắt việc sử dụng và mở rộng PMTD. Vì thế, không ngạc nhiên rằng hơn 70% các lập trình viên có giáo dục đại học. Dữ liệu này còn đáng kể hơn nếu chúng ta nhớ rằng 30% phần còn lại không phải ở độ tuổi đại học vì họ vẫn còn ở trong các trường phổ thông. Ngay cả như vậy, thì cũng có liên quan và được đánh giá không ít hơn so với các lập trình viên mà họ đã không bao giờ có sự truy cập được tới nền giáo dục cao hơn, nhưng là những người nhiệt thành về công nghệ thông tin.

Lập trình viên của PMTD thường là phái nam. Những con số thống kê bởi các cuộc khảo sát khác nhau về sự hiện diện của phái nữ trong cộng đồng này du di khoảng 1% và 3%, tranh đua với khoảng dung sai lỗi của riêng chúng. Cùng lúc, đa số (60%) nói có một bạn khác giới, trong khi số lượng các lập trình viên là trẻ em chỉ là 16%. Đưa ra độ tuổi của các lập trình viên của PMTD, dữ liệu này khá trùng khớp với một ví dụ ngẫu nhiên, nghĩa là nó có thể được coi là "bình thường". Câu chuyện hoang đường về lập trình viên cô đơn mà sự nhiệt thành của người đó đối với công nghệ thông tin chỉ là thứ để khoe mẽ trong cuộc đời của anh ta, như chúng ta có thể thấy, là một ngoại lệ hơn là một qui luật.

4.3 Lập trình viên làm gì?

Nói một cách chuyên nghiệp, thì các lập trình viên PMTD tự mô tả họ như những kỹ sư phần mềm (33%), sinh viên (21%), lập trình viên (11%), các nhà tư vấn (10%), các giáo sư đại học (7%), … Về phạm vi, ở chiều ngược lại, chúng tôi thấy rằng họ có xu hướng không tạo thành một phần của các phòng bán hàng hoặc marketing (khoảng 1%). Thú vị để lưu ý xem có bao nhiêu người trong số họ tự xác định như là các kỹ sư phần mềm hơn là các lập trình viên, gần như là gấp 3 lần, hãy ghi nhớ, như chúng ta sẽ thấy trong chương này về kỹ thuật phần mềm, thì đó là sự ứng dụng các kỹ xảo của kỹ thuật phần mềm kinh điển (và ngay cả một số ứng dụng hiện đại) thực sự không phải là lạ gì trong thế giới của PMTD.

Sự kết nối với các đại học đã được chứng minh, được đưa lên hàng đầu một lần nữa trong phần này. Khoảng 1 trong số 3 lập trình viên là sinh viên hoặc giáo sư đại học, chỉ ra có sự hợp tác khổng lồ với những người chủ yếu tới từ giới công nghiệp phần mềm (phần còn lại là 2/3) và giới hàn lâm.

Cùng lúc, đã có khả năng để xác định một phạm vi rộng những nguyên tắc được pha trộn: 1/5 số các lập trình viên tới từ một lĩnh vực mà không phải là công nghệ thông tin. Điều này, cộng với thực tế là cũng có một số lượng tương tự những lập trình viên không ở trong các trường đại học, phản ánh một sự phong phú về những lợi ích, gốc gác, và chắc chắn, sự kết hợp của những đội phát triển. Rất khó để tìm được một giới công nghiệp hiện đại, nếu có, nơi mà mức độ hỗn hợp không đồng nhất lại rộng lớn như thứ mà chúng ta có thể thấy trong PMTD. Bổ sung thêm vào khoảng 20% các sinh viên, 64% các lập

trình viên hầu hết là những nhân viên được trả tiền, trong khi tỷ lệ các lập trình viên tự được thuê chỉ là 14%. Cuối cùng, chỉ 3% nói là những người không được thuê làm, một con số đáng kể khi mà khảo sát này đã được thực hiện sau khi mà cuộc khủng hoảng dotcom đã bắt đầu.

Lưu ý

Thực tế là mô hình kinh doanh PMTD, không giống như với PMSHĐQ, không thể đạt được thông qua việc bán các giấy phép đã luôn làm nguội các cuộc tranh luận nóng bỏng đối với cách mà các lập trình viên kiếm tiền để nuôi sống họ. Trong các khảo sát mà chúng tôi đưa ra trong chương này, 50% các lập trình viên đã nói đã giành được sự bù đắp về tài chính trực tiếp hoặc gián tiếp cho sự tham gia của họ vào PMTD. Tuy nhiên, nhiều người khác không chắc chắn lắm. Richard Stallman, người sáng lập ra dự án GNU, khi được hỏi một lập trình viên của PMTD phải làm gì để kiếm tiền, thì có xu hướng trả lời rằng anh ta có thể làm việc như một người hầu bàn.

4.4 Phân bố theo địa lý

Việc có được các dữ liệu theo địa lý của các lập trình viên là một vấn đề mà cần có được tiếp cận theo một cách khoa học hơn. Vấn đề này với nghiên cứu được chỉ ra trong chương này cho là vì nó dựa trên các khảo sát trên Internet, nên mở ra cho bất kỳ ai muốn tham gia, sự tham gia đã phụ thuộc ở một mức độ nào đó vào các site mà nó đã được đưa lên, và cách mà theo đó nó đã được công bố. Để chính xác, chúng tôi phải lưu ý rằng những khảo sát này đã không có mục đích sẽ đại diện theo cách này, mà để có được các câu trả lời và/hoặc các ý kiến về số lượng lớn nhất có thể được các lập trình viên của PMTD.

Tuy nhiên, chúng ta có thể mạo hiểm để đưa ra một số tuyên bố về vấn đề này, biết rằng những dữ liệu này không được tin cậy như những dữ liệu trước đó, và vì thế, số lượng lỗi sẽ là nhiều hơn nhiều. Những gì được coi là một thực tế là hầu hết các lập trình viên của PMTD tới từ các quốc gia công nghiệp, và rằng sự hiện diện của các lập trình viên từ cái gọi là các quốc gia thuộc thế giới thứ ba là rất hiếm. Hệ quả là, sẽ không ngạc nhiên rằng bản đồ các lập trình viên của dự án Debian (http://www.debian.org/devel/developers.loc) [187], ví dụ, khớp với ảnh của Trái đất về buổi tối: nơi mà được chiếu sáng - đọc là "nơi mà có một nền văn minh công nghiệp" - đó là nơi mà các lập trình viên của PMTD có xu hướng tập trung về. Điều này thoạt đầu có thể coi là logic, ngược với những cơ hội tiềm tàng mà PMTD có thể đưa ra cho các quốc gia thuộc thế giới thứ ba. Chúng ta có thể thấy một ví dụ rõ ràng trong bảng sau, mà nó chứa các quốc gia chung nhất về gốc gác của các lập trình viên dự án Debian trong 4 năm gần đây nhất. Có một xu hướng đáng lưu ý hướng tới sự li tâm của dự án này, bằng chứng từ thực tế là sự tăng trưởng số lượng các lập trình viên từ Mỹ, quốc gia đóng góp nhiều nhất, lại thấp hơn mức trung bình. Và thực tế là, nói chung, các quốc gia có gấp đôi số lượng các tình nguyện viên từ 1999 đến 2003, và Pháp, quốc gia có sự hiện diện gấp tới 5 lần, là ví dụ rõ rệt nhất theo hướng này. Cho rằng Debian đã thực hiện những bước ban đầu của nó tại châu Mỹ (đặc biệt là Mỹ và Canada), thì chúng ta có thể thấy rằng trong ít năm qua dự án này đã trở thành *của châu Âu* rồi. Chúng ta giả thiết là bước tiếp sau sẽ được tìm kiếm hơn nhiều - sau sự toàn cầu hóa, với sự kết hợp của các quốc gia Nam Mỹ, châu Phi và châu Á (với ngoại lệ đối với Hàn Quốc và Nhật Bản, mà những nước này đã được thể hiện tốt), dù các dữ liệu mà chúng tôi có (2 cộng tác viên từ Hy Lạp, Trung Quốc hoặc Ấn Độ, và một tại Mexico, Thổ Nhĩ Kỳ hoặc Colombia vào tháng 06/2003) sẽ không hứa hẹn nhiều theo nghĩa này.

Trong thế giới của PMTD (và không chỉ trong trường hợp của Debian), có một tranh luận gay gắt về ưu thế của châu Âu và Mỹ. Hầu như tất cả các nghiên cứu đã chỉ ra rằng sự hiện diện của các lập trình viên châu Âu khá là cao hơn so với Bắc Mỹ, một ấn tượng mà nó được làm cho dịu bớt bởi thực tế rằng dân số châu Âu là lớn hơn so với Mỹ. Vì thế, chúng ta đang làm việc với một cuộc chiến tranh về các con số, vì số lượng các lập trình viên tính theo đầu người là cao hơn trong số những người Bắc Mỹ. Nếu chúng ta tính tới số lượng người có truy cập Internet thay vì tổng dân số, thì châu Âu lại đứng đầu một lần nữa.

Về các quốc gia, những khu vực với mức độ cao nhất về sự thâm nhập sâu (theo số lượng các lập trình viên được phân theo dân số) là Bắc Âu (Phần Lan, Thụy Điển, Nauy, Đan Mạch và Iceland) và Trung Âu (Bạch Nga, Đức và Tiệp), tiếp sau là Úc, Canada, New Zealand và Mỹ. Dù tầm quan trọng của nó trong tuyệt đối (nhờ vào dân số lớn của Pháp, Ý và Tây Ban Nha), thì vùng Địa Trung Hải là thấp hơn trung bình.

Bảng 1. Các quốc gia với số lượng lớn nhất các lập trình viên dự án Debian

Quốc gia	01/07/1999	01/07/2000	01/07/2001	01/07/2002	20/06/2003
Mỹ	162	169	256	278	297
Đức	54	58	101	121	136
Anh	34	34	55	63	75
Úc	23	26	41	49	52
Pháp	11	11	24	44	51
Canada	20	22	41	47	49
Tây Ban Nha	10	11	25	31	34
Nhật Bản	15	15	27	33	33
Ý	9	9	22	26	31
Hà Lan	14	14	27	29	29
Thụy Điển	13	13	20	24	27

4.5 Sự cống hiến

Số giờ mà các lập trình viên của PMTD bỏ ra để phát triển PMTD là một con số rất không được rõ. Chúng ta cũng phải chỉ ra đây là một trong những khác biệt chính với phần mềm được tạo ra bởi các công ty, nơi mà các thành viên của đội và thời gian bỏ ra bởi mỗi thành viên trong đội để phát triển được biết rõ. Thời gian mà các lập trình viên cống hiến cho PMTD có thể được tính như một phép đo không trực tiếp về mức độ chuyên nghiệp của họ. Trước khi chỉ ra các dữ liệu có sẵn hiện nay, điều quan trọng để lưu ý là nó được lấy từ những ước tính được đưa ra bởi các lập trình viên trong một loạt

các khảo sát, vì thế bổ sung thêm vào sự không chính xác vốn có của việc thu thập dạng dữ liệu này, chúng ta cần coi sai số có liên quan tới cách mà mỗi lập trình viên tính thế nào là thời gian phát triển. Vì thế, chắc chắn là nhiều lập trình viên sẽ không đưa vào thời gian bỏ ra cho việc đọc các thư điện tử (hoặc có thể họ sẽ) và chỉ đưa ra thời gian mà họ bỏ ra lập trình và sửa lỗi. Vì thế, tất cả các con số mà chúng ta chỉ ra tiếp sau cần được xem xét ở mức độ dè dặt.

Nghiên cứu đã đưa ra ngày tháng chỉ rằng mỗi lập trình viên phần mềm bỏ ra 11 giờ trung bình trong một tuần ("Động lực của các lập trình viên phần mềm trong các dự án nguồn mở: một khảo sát dựa trên Internet về những người đóng góp cho nhân Linux", 2003) [143]. Tuy nhiên, con số này có thể dễ làm cho lầm lẫn, vì có một sự khác nhau về thời gian bỏ ra của các lập trình viên phần mềm. Trong nghiên cứu về *PMTD và PMNM, Khảo sát và nghiên cứu*, phần IV: "Khảo sát và các lập trình viên", 2002 [126], 22.5% những người được khảo sát đã nói rằng sự đóng góp của họ là ít hơn 2 giờ đồng hồ trong một tuần, và con số này đã gia tăng 26.5% đối với những người bỏ ra từ 2 - 5 giờ đồng hồ mỗi tuần; khoảng từ 6 – 10 giờ mỗi tuần là thời gian được bỏ ra của 21%, trong khi 14.1% đã bỏ ra từ 11-20 giờ mỗi tuần; 9.2% đã nói rằng thời gian họ bỏ ra phát triển PMTD là từ 20-40 giờ mỗi tuần và 7.1% hơn 40 giờ mỗi tuần.

Bảng 2. Sự cống hiến theo số giờ trong một tuần

Số giờ trong tuần	<2	Từ 2-5	Từ 5-10	Từ 10-20	Từ 20-40	>40
Số phần trăm	22.5	26.1	21	14.1	9.2	7.1

Lưu ý

Bổ sung thêm vào việc chỉ ra mức chuyên nghiệp của các đội phát triển PMTD, thời gian bỏ ra theo giờ là một tham số hợp lý khi mà nói tới việc ước tính và so sánh giá thành thực hiện với các mô hình phát triển PMSHĐQ trong nền công nghiệp này. Với PMTD, bây giờ, chúng ta chỉ có được những sản phẩm cuối cùng (những dẫn xuất phần mềm mới, việc đồng bộ hóa các mã nguồn mới trong hệ thống các phiên bản...) mà không cho phép chúng ta biết được bao nhiêu thời gian lập trình viên đã bỏ ra để đạt được chúng.

Một phân tích của các con số này nói cho chúng ta rằng khoảng 80% các lập trình viên thực hiện các nhiệm vụ này trong thời gian rỗi của họ, trong khi chỉ 1/5 có thể coi rằng họ bỏ nhiều thời gian vào hoạt động này như một người chuyên nghiệp. Sau này, trong chương về kỹ thuật phần mềm, chúng ta sẽ thấy các dữ liệu này khớp được với những đóng góp của các lập trình viên như thế nào, khi mà họ đều dường như tuân theo luật Pareto (xem phần 7.6).

4.6 Những động lực

Từng có nhiều phỏng đoán về những động lực mà các lập trình viên phải phát triển PMTD, đặc biệt khi nó được thực hiện vào thời gian rỗi (mà, như chúng ta đã thấy, tương ứng với 80% các lập trình viên).

Như trong những phần trước, chúng ta chỉ có các dữ liệu khảo sát, mà nó giải thích vì sao lại quan trọng để nhận thức rằng các câu trả lời đã được đưa ra bởi bản thân các lập trình viên, nghĩa là họ có thể ít nhiều gắn được vào với thực tế. Số phần trăm chỉ ra vượt quá 100% vì đã có một lựa chọn cho những người tham gia chọn một vài câu trả lời.

Trong mọi trường hợp, dường như từ những câu trả lời của họ mà hầu hết muốn học và phát triển những kỹ năng mới (khoảng 80%) và nhiều người làm thế để chia sẻ kiến thức và các kỹ năng (50%) hoặc tham gia vào một dạng cộng tác mới (khoảng 1/3).

Các dữ liệu ban đầu là không ngạc nhiên, đưa ra rằng một người chuyên nghiệp với tri thức nhiều hơn sẽ đòi hỏi cao hơn người có tri thức ít hơn. Tuy nhiên, hoàn toàn không dễ dàng để giải thích các dữ liệu thứ 2, và nó có thể dường như còn đối nghịch với ý kiến của Nikolai Bezroukov trong "Cái nhìn thứ 2 vào nhà thờ lớn và cái chợ" (Tháng 12/1999) [91] mà những người cầm đầu của các dự án PMTD cẩn thận không chia sẻ tất cả các thông tin về quyền sở hữu của họ để ghi danh mãi mãi sức mạnh của họ. Trong khi đó, lựa chọn thứ 3 thường xuyên nhất là không nghi ngờ gì, một sự phản ánh đúng về nhiệt thành của các lập trình viên về cách thức mà PMTD được tạo ra nói chung; khó tìm được một giới công nghiệp nào mà trong đó một nhóm những người tình nguyện được tổ chức lỏng lẻo lại có thể - về mặt công nghệ mà nói - đứng lên chống lại được những người khổng lồ của khu vực này.

Dù lý thuyết "kinh điển" cho việc giải thích vì sao các lập trình viên của PMTD đã bỏ thời gian ra để đóng góp cho dạng các dự án này là uy tín và những lợi ích kinh tế không trực tiếp về trung và dài hạn, nó có thể dường như là các lập trình viên bản thân họ không đồng tình với những câu nói này. Chỉ 5% trong số những người được khảo sát đã trả lời rằng họ phát triển PMTD để kiếm tiền, trong khi số lượng những người đã làm thế để thiết lập một uy tín tăng tới 90%, rất xa với những câu trả lời được đưa ra trong đoạn trước. Trong mọi trường hợp, dường như là việc nghiên cứu những động lực của các lập trình viên để trở thành một phần của cộng đồng PMTD là một nhiệm vụ cơ bản, mà các nhà xã hội học và tâm lý học sẽ phải đối mặt trong tương lai gần.

4.7 Sự lãnh đạo

Uy tín và sự lãnh đạo là 2 đặc tính được sử dụng để giải thích thành công của PMTD, và đặc biệt, mô hình cái chợ, như chúng ta sẽ thấy từ chương về kỹ thuật phần mềm. Như chúng ta đã thấy trong chương khác, về các giấy phép phần mềm, có những khác biệt chắc chắn giữa các giấy phép của PMTD và các giấy phép tương ứng của nó trong lĩnh vực tài liệu. Những khác biệt này bắt nguồn từ cách duy trì sự lãnh đạo và ý kiến được nhấn mạnh hơn của các tác giả trong văn bản hơn là trong các chương trình.

Trong PMTD và *PMNM. Khảo sát và nghiên cứu*, phần IV: "Khảo sát về các lập trình viên" (2002) [126] một câu hỏi đã được đưa vào mà các lập trình viên được hỏi để chỉ ra những người nào từ một danh sách được biết đối với họ, không nhất thiết với tư cách cá nhân.

Kết quả, được đưa ra ở bảng 3, chỉ ra những người có thể được chia thành 3 nhóm khác biệt rõ ràng:

Bảng 3. Mức độ thừa nhận đối với các lập trình viên quan trọng

Lập trình viên	Linus Torvalds	Richard Stallman	Miguel de Icaza	Eric Raymond	Bruce Perens
Được biết bởi	96.50%	93.30%	82.10%	81.10%	57.70%
Lập trình viên	Jamie Zawinski	Mathias Ettrich	Jörg Schilling	Marco Pesenti Gritti	Bryan Andrews
Được biết bởi	35.80%	34.20%	21.50%	5.70%	5.60%
Lập trình viên	Guenter Bartsch	Arpad Gereoffy	Martin Hoffstede	Angelo Roulini	Sal Valliger
Được biết bởi	3.50%	3.30%	2.90%	2.60%	1.20%

- Nhóm đầu tiên là những người với những ý nghĩa lịch sử và triết lý rõ ràng bên trong thế giới của PMTD (dù, như chúng ta đã biết, họ có thể cũng có những kỹ năng kỹ thuật đáng chú ý)

1. Linus Torvalds: Người sáng tạo ra nhân Linux, hệ điều hành tự do được sử dụng nhiều nhất, và đồng tác giả của *Chỉ để vui: câu chuyện về một cuộc cách mạng ngẫu nhiên* [217].
2. Richard Stallman: Nhà tư tưởng và người sáng lập FSF và lập trình viên trong một loạt các dự án GNU. Tác giả của vài bài viết quan trọng về PMTD ("Vì sao *PMTD* là tốt hơn *PMNM*", 1998 [206], "Copyleft: Chủ nghĩa lý tưởng thực dụng", 1998 [205], "Dự án GNU" [208] và "Tuyên ngôn GNU", 1985 [117]).
3. Miguel de Icaza: Đồng sáng lập của dự án GNOME và hãng Ximian, và là lập trình viên trong GNOME và MONO.
4. Eric Raymond: Người sáng lập ra OSI, tác giả của "Nhà thờ lớn và cái chợ" [192] và là lập trình viên chính của fetchmail.
5. Bruce Perens: Cựu lãnh đạo của dự án Debian, người sáng lập (được chuyển đổi) của OSI và lập trình viên của công cụ e-fence.
6. Jamie Zawinsky: Cựu lập trình viên của Mozilla và nổi tiếng vì bức thư năm 1999 trong đó ông đã để dự án Mozilla tranh cãi rằng mô hình mà họ đã đi theo có thể không bao giờ đâm hoa kết trái ("Từ chức và sau cái chết", 1999) [237].
7. Mathias Ettrich: Người sáng lập của KDE và lập trình viên của LyX và những thứ khác.

- Nhóm thứ 2 là các lập trình viên. Khảo sát này đã lấy những cái tên của những lập trình viên hàng đầu của 6 dự án nổi tiếng nhất theo site tải về các PMTD FreshMeat. Chúng ta có thể thấy rằng (với sự ngoại lệ của Linus Torvalds, vì những lý do rõ ràng, và Jörg Schilling) mức độ thừa nhận của những lập trình viên này là nhỏ:

1. Jörg Schilling, người sáng tạo ra cdrecord và những ứng dụng khác.
2. Marco Pesenti Gritti, lập trình viên chính của Galeon.
3. Bryan Andrews, lập trình viên của Apache Toolbox.
4. Guenther Bartsch, người sáng lập ra Xine.
5. Arpad Gereoffy, lập trình viên của MPEGPlayer.

- Nhóm 3 có những cái tên của 3 "người" cuối cùng trong bảng. Những cái tên này đã được sáng tạo ra bởi đội khảo sát để kiểm tra sai số.

Chúng ta có thể đưa ra 2 kết luận từ các kết quả này: đầu tiên là chúng ta có thể coi sai số là nhỏ (ít hơn 3%), và thứ 2 là việc hầu hết các lập trình viên của hầu hết các ứng dụng PMTD phổ biến nhất là nổi tiếng như những người mà những người này không tồn tại. Dữ liệu này làm cho những ai mà khẳng định rằng một trong những lý do cho việc phát triển PMTD là tìm kiếm danh tiếng hãy nghĩ lại 2 lần.

4.8 Tóm lược và các kết luận

Chương này đã cố gắng đưa ra ánh sáng về vấn đề còn chưa được biết một cách rộng rãi về những người đã dành thời gian cho PMTD. Nhìn chung, chúng ta có thể nói rằng một lập trình viên của PMTD là một thanh niên trẻ tuổi với một trình độ đại học (hoặc trên con đường để có được nó). Mối quan hệ giữa thế giới của PMTD và các trường đại học (các sinh viên và giáo sư) là rất gần gũi, dù lập trình viên không có gì để làm với môi trường hàn lâm cho tới khi vượt trội hơn hẳn.

Về số giờ cống hiến, chúng ta đã chỉ ra rằng có một sự cách biệt to lớn, tương tự như đối với những gì là mặc định trong luật Pareto. Những động lực của các lập trình viên, theo ý kiến của riêng họ, là tách xa với tiền bạc và tính ích kỷ, khi mà các nhà kinh tế học và tâm lý học có xu hướng giả thiết như vậy, và hầu hết họ làm để chia sẻ và học hỏi. Cuối cùng, chúng ta đã chỉ ra một bảng với những cá nhân đáng kể nhất trong thế giới của PMTD (bao gồm cả những người khác mà họ đã không phải là nổi tiếng như vậy, như chúng ta đã thấy) và chỉ ra rằng uy tín trong cộng đồng PMTD rộng lớn có xu hướng phụ thuộc vào nhiều thứ nữa hơn, chứ không chỉ vào việc lập trình một ứng dụng PMTD thành công.

5 Kinh tế

"Res publica non dominetur".

“Những thứ của công là không có chủ” (dịch tự do)

Đã xuất hiện trong một quảng cáo của IBM về Linux (2003)

Chương này xem xét một số khía cạnh kinh tế có liên quan tới PMTD. Chúng ta sẽ bắt đầu bằng việc chỉ ra cách mà các dự án PMTD được cấp tài chính (khi chúng quả thực được cấp tài chính, vì trong nhiều trường hợp chúng chỉ dựa vào những nỗ lực và tài nguyên được đóng góp một cách tự nguyện). Tiếp theo, chúng ta sẽ xem xét các mô hình kinh doanh chính được đưa vào thực tế bởi các công ty có liên quan trực tiếp tới PMTD. Chương này kết thúc với một nghiên cứu nhỏ về mối quan hệ giữa PMTD và các nhà độc quyền trong nền công nghiệp phần mềm.

5.1 Cấp vốn cho các dự án PMTD

PMTD được phát triển theo nhiều cách khác nhau và việc sử dụng các cơ chế để giành vốn là hết sức khác nhau theo từng trường hợp. Mỗi dự án tự do có cách riêng của bản thân nó trong việc cấp vốn, từ dự án cấu tạo hoàn toàn từ những lập trình viên tự nguyện và chỉ sử dụng vốn nhượng lại một cách vị tha, cho tới dự án triển khai bởi một công ty mà nó báo giá 100% giá thành của dự án cho một tổ chức có quan tâm trong sự phát triển tương ứng đó.

Trong phần này, chúng ta sẽ tập trung vào các dự án nơi mà có vốn từ bên ngoài và không phải tất cả công việc là tự nguyện. Trong những trường hợp này, thường sẽ có một số dạng dòng vào tiền mặt, từ một nguồn bên ngoài vào dự án, có trách nhiệm về việc cung cấp vốn cho sự phát triển của nó. Cách này, PMTD mà được xây dựng có thể được coi, ở một phạm vi nào đó, sẽ là một sản phẩm của vốn bên ngoài này. Điều này giải thích vì sao nó là chung đối với nguồn bên ngoài đó để quyết định (ít nhất một phần) vốn đó sẽ được chi thế nào và cho cái gì.

Trong một số trường hợp, vốn bên ngoài này cho các dự án tự do có thể được coi như một dạng tài trợ, dù sự tài trợ này không có lý do cho việc không có sự mưu lợi (và thường là không). Trong các phần sau chúng ta sẽ thảo luận về các dạng chung nhất của vốn bên ngoài. Tuy nhiên, trong khi học về chúng, chúng ta phải nhớ rằng chúng chỉ là một số cách thức mà các dự án PMTD có được các nguồn. Nhưng có những cách khác, và trong số đó thì cách quan trọng nhất (như chúng ta đã thấy trong chương 4) là công việc của nhiều lập trình viên tự nguyện.

5.1.1 Vốn nhà nước

Một dạng rất đặc biệt của việc cấp vốn cho các dự án tự do là việc cấp vốn của nhà nước. Cơ quan cấp vốn có thể một cách trực tiếp là chính phủ (địa phương, vùng, quốc gia hoặc ngay cả là siêu quốc gia) hoặc một cơ quan nhà nước (ví dụ, một quỹ). Trong những trường hợp này, việc cấp vốn có xu hướng sẽ tương tự như cho các dự án nghiên cứu và phát triển (R&D) và trên thực tế nó là chung cho việc cấp

vốn tới từ các cơ quan nhà nước mà họ khuyến khích R&D. Thường thì, cơ quan cấp vốn sẽ không tìm cách để hoàn vốn đầu tư (hoặc ít nhất không trực tiếp), dù nó có xu hướng sẽ có những mục tiêu rõ ràng (như việc khuyến khích sự sáng tạo của một công trình công nghiệp hoặc dựa trên nghiên cứu, khuyến khích một công nghệ cụ thể nào đó hoặc dạng ứng dụng nào đó, …).

Trong hầu hết các trường hợp này, sẽ không có việc cấp vốn rõ ràng của các sản phẩm và dịch vụ có liên quan tới PMTD, nhưng điều này thường có xu hướng sẽ là sản phẩm phụ của một hợp đồng với các mục tiêu khác chung hơn. Ví dụ, như một phần của các chương trình nghiên cứu của mình, Ủy ban châu Âu cấp vốn cho các dự án hướng tới việc cải thiện tính cạnh tranh trong một số lĩnh vực cụ thể nào đó. Một số dự án này phải là một phần của các mục tiêu của chúng để sử dụng, cải thiện và tạo ra những PMTD bên trong phạm vi của nghiên cứu (như một công cụ nghiên cứu hoặc một sản phẩm dẫn xuất từ nó).

Những động lực cho dạng cấp vốn này là rất khác nhau, nhưng chúng ta có thể phân biệt như sau:

1. Khoa học: Đây là cách thường thấy nhất trong trường hợp các dự án nghiên cứu được cấp vốn nhà nước. Dù mục tiêu của nó không phải là để tạo ra phần mềm nhưng là để điều tra nghiên cứu một lĩnh vực cụ thể nào đó (dù có liên quan hay không tới công nghệ thông tin), có thể yêu cầu các chương trình sẽ được phát triển như các công cụ cho việc đạt được các mục tiêu của dự án. Thường thì dự án này không có quan tâm trong việc thương mại hóa các công cụ này, hoặc thậm chí có thể được quan tâm một cách tích cực trong các nhóm khác có sử dụng và cải thiện chúng. Trong các trường hợp như vậy, khá là chung để phân phối chúng như là PMTD. Theo cách này, nhóm tiến hành nghiên cứu có một phần vốn dành riêng cho việc sản xuất phần mềm, nên chúng ta có thể nói rằng nó đã được phát triển với việc cấp vốn của nhà nước.

2. Khuyến khích các chuẩn. Việc có một triển khai tham chiếu là một trong những cách thức tốt nhất để khuyến khích một chuẩn. Trong nhiều trường hợp điều này liên quan tới việc có các chương trình mà chúng tạo thành một phần của triển khai được nói tới (hoặc nếu chuẩn này tham chiếu tới lĩnh vực phần mềm, thì sẽ là những triển khai tự bản thân chúng). Đối với triển khai tham chiếu sẽ là hữu dụng trong việc khuyến khích chuẩn, nó cần phải là sẵn sàng, ít nhất để kiểm tra tính tương hợp cho tất cả những ai mong muốn phát triển các sản phẩm mà chúng đăng ký tới chuẩn đó. Và trong mọi trường hợp cũng được tư vấn cho các nhà sản xuất để có khả năng áp dụng triển khai tham chiếu này một cách trực tiếp để sử dụng nó với các sản phẩm của họ nếu muốn.

   Ví dụ, làm thế nào mà các giao thức Internet đã được phát triển, mà chúng bây giờ đã trở thành một chuẩn vạn năng. Trong những trường hợp như vậy, việc đưa ra các triển khai tham chiếu như PMTD có thể đóng góp vô cùng lớn cho việc khuyến khích. Một lần nữa, PMTD ở đây là một sản phẩm phụ, trong trường hợp này là của việc khuyến khích chuẩn. Và thường bên có trách nhiệm cho việc khuyến khích này là một cơ quan nhà nước (dù đôi khi nó có thể là một nhóm các doanh nghiệp tư nhân).

3. Xã hội: PMTD là một công cụ rất thú vị cho việc tạo ra nền tảng cơ sở cho xã hội thông tin. Những tổ chức có quan tâm trong việc sử dụng PMTD để cải thiện sự truy cập vạn năng tới xã hội thông tin có thể cấp vốn cho các dự án có liên quan tới nó (thường với các dự án cho việc phát triển các ứng dụng mới hoặc việc áp dụng những ứng dụng đang tồn tại sẵn rồi).

Lưu ý

Một ví dụ cho việc cấp vốn của nhà nước cho một mục tiêu xã hội ban đầu là trường hợp của gnuLinEx, được đưa ra bởi Chính quyền Vùng Extremadura (Extremadura, Tây Ban Nha) để khuyến khích xã hội thông tin một cách cơ bản trong vấn đề hiểu biết về máy tính. Chính quyền vùng này đã cấp vốn cho sự phát triển của một phát tán dựa trên Debian để đạt được mục tiêu này. Trường hợp tương tự khác là chính phủ Đức cấp vốn cho những phát triển của GnuPG, có mục đích làm cho chúng dễ dàng hơn để sử dụng đối với những người sử dụng không có kinh nghiệm, với ý tưởng về khuyến khích sử dụng thư an ninh của các công dân của mình.

Sự phát triển của GNAT

Một trường hợp đáng lưu ý về việc cấp vốn nhà nước cho một phát triển của PMTD là trường hợp của trình biên dịch GNAT. GNAT, một trình biên dịch Ada, đã được cấp vốn bởi dự án Ada 9X của Bộ Quốc phòng Mỹ, với ý tưởng về việc có một trình biên dịch của phiên bản mới của ngôn ngữ lập trình Ada (sau này trở thành Ada95), mà nó đã cố gắng để cải thiện vào thời điểm đó. Một trong những trường hợp được xác định có liên quan tới các công ty phần mềm áp dụng phiên bản đầu tiên của Ada (Ada 83) từng là khả năng chậm trễ của một trình biên dịch của một ngôn ngữ và giá thành cao của nó khi nó cuối cùng đã được tung ra. Vì thế, họ đã cố gắng ngăn cản thứ y như vậy xảy ra với Ada 95, đảm bảo rằng trình biên dịch này đã hầu như sẵn sàng cùng một lúc với việc tung ra chuẩn mới của ngôn ngữ này.

Để làm như vậy, Ada 9X đã hợp đồng với một dự án với một đội từ Đại học New Yord (NYU), với một giá trị xấp xỉ 1 triệu USD, để triển khai một "triển khai khái niệm" của trình biên dịch Ada 95. Việc sử dụng các vốn này, và tận dụng sự tồn tại của GCC (trình biên dịch C của GNU, trong đó hầu hết các nền tảng đã được sử dụng), đội NYU đã xây dựng thành công trình biên dịch Ada 95 đầu tiên, mà nó đã tung ra theo giấy phép GNU GPL. Trình biên dịch này đã quá thành công nên khi dự án đã kết thúc thì một số những người tạo ra nó đã thành lập một công ty (Ada Core Technologies), mà kể từ đó đã trở thành người dẫn đầu thị trường trong các trình biên dịch và các công cụ trợ giúp cho việc xây dựng các chương trình trong Ada.

Trong dự án này đáng lưu ý sự kết hợp của các yếu tố nghiên cứu (trong thực tế, tri thức tiên tiến của dự án này về việc xây dựng các phần mặt tiền (front ends) và các hệ thống thời gian thực cho các trình biên dịch ngôn ngữ dạng Ada) và sự khuyến khích các chuẩn (mà chúng từng là mục tiêu rõ ràng nhất của cơ quan cấp vốn).

5.1.2 Việc cấp vốn không vì lợi nhuận của tư nhân

Dạng cấp vốn này có nhiều đặc điểm tương tự đối với dạng trên, mà nó thường được tiến hành bởi các quỹ hoặc các tổ chức phi chính phủ. Động lực trực tiếp trong những trường hợp này có xu hướng sẽ sản sinh ra PMTD để sử dụng trong lĩnh vực mà cơ quan cấp vốn coi là đặc biệt phù hợp, nhưng chúng ta cũng có thể thấy động lực gián tiếp của việc đóng góp để giải quyết vấn đề (ví dụ, một quỹ mà khuyến khích nghiên cứu trong một căn bệnh có thể cấp vốn cho việc xây dựng của một chương trình thống kê mà giúp phân tích các nhóm thí nghiệm được sử dụng như một phần của nghiên cứu trong căn bệnh đó).

Nói chung, cả những động lực và những cơ chế cho dạng cấp vốn này là rất giống với những cách cấp vốn của nhà nước, dù về bản chất tự nhiên chúng sẽ luôn nằm trong ngữ cảnh của những mục tiêu của

cơ quan cấp vốn.

Lưu ý

Có lẽ, trường hợp nguyên mẫu của một quỹ mà nó khuyến khích sự phát triển của PMTD là FSF. Từ giữa những năm 1980 thì quỹ này được cống hiến cho việc khuyến khích dự án GNU và cho việc hỗ trợ phát triển PMTD nói chung.

Trường hợp thú vị khác, dù trong một lĩnh vực khá là tách biệt, là Quỹ Tin Sinh Mở (Open Bioinformatics Foundation - quỹ tin học trong sinh học). Mục tiêu của quỹ này bao gồm việc khuyến khích sự phát triển của các chương trình máy tính cơ bản trong nghiên cứu trong bất kỳ nhánh nào của tin sinh. Nói chung, nó khuyến khích điều này bằng việc cấp vốn và đóng góp cho việc xây dựng các chương trình tự do.

5.1.3 Cấp vốn bởi ai đó yêu cầu những cải tiến

Một dạng cấp vốn khác cho sự phát triển của PMTD, mà nó không thật vị tha, diễn ra khi ai đó cần tiến hành những cải tiến cho một sản phẩm tự do. Ví dụ, để sử dụng nội bộ, một công ty có thể cần một chương trình cụ thể nào đó phải có một chức năng cụ thể nào đó hoặc để sửa một ít các lỗi. Trong những trường hợp này, thường đối với công ty yêu cầu ký hợp đồng cho sự phát triển theo yêu cầu này. Sự phát triển này thường là PMTD (hoặc vì giấy phép của chương trình được sửa áp đặt nó, hoặc vì công ty quyết định nó).

Trường hợp của Corel và Wine

Về cuối những năm 1990, Corel đã quyết định chuyển các sản phẩm của hãng sang GNU/Linux. Trong quá trình này hãng đã phát hiện ra rằng một chương trình tự do được thiết kế để tạo điều kiện thuận lợi cho việc chạy các tệp nhị phân cho Windows trong các môi trường của Linux có thể giúp tạo ra sự tiết kiệm đáng kể khi phát triển. Mà để làm được thế, thì nó phải được cải tiến, về cơ bản bằng việc bổ sung thêm sự mô phỏng của một vài chức năng của Windows mà các chương trình của Corel đã sử dụng.

Vì điều này, Corel đã hợp đồng với Macadamian, mà nó đã đóng góp những cải tiến của mình vào dự án Wine. Bằng cách này, cả Corel và Wine đều đã được lợi.

5.1.4 Việc cấp vốn với những lợi ích liên quan

Bằng dạng cấp vốn này, cơ quan cấp vốn có mục tiêu để đạt được lợi nhuận từ các sản phẩm có liên quan tới chương trình mà cơ quan này cấp vốn cho sự phát triển của nó. Thông thường, trong những trường hợp này thì lợi nhuận giành được của cơ quan cấp vốn là không độc chiếm, vì những người khác cũng có thể tham gia vào thị trường này để bán các sản phẩm có liên quan, mà thị phần nó giành được là đủ cho nó không lo lắng quá nhiều về việc chia sẻ chiếc bánh với những người khác, hoặc nó có một ưu thế cạnh tranh rõ ràng.

Một số ví dụ về những sản phẩm liên quan tới một phần mềm cụ thể là như sau:

- Các cuốn sách: Công ty theo yêu cầu bán các sách học, các chỉ dẫn cho người sử dụng, các tư

liệu khóa học, … có liên quan tới chương trình tự do mà nó giúp cấp vốn. Tất nhiên, các công ty khác cũng có thể bán các sách liên quan, nhưng thường thì việc cấp vốn cho dự án sẽ trao cho công ty sự truy cập sớm tới các lập trình viên chủ chốt trước sự cạnh tranh, hoặc đơn giản cung cấp một hình ảnh tốt hướng tới cộng đồng người sử dụng của chương trình này theo yêu cầu.

- Phần cứng: Nếu một công ty cấp vốn để phát triển các hệ thống tự do cho một dạng phần cứng nào đó. Một lần nữa, vì phần mềm được phát triển là tự do, nên các đối thủ cạnh tranh bán cùng một dạng các thiết bị có thể xuất hiện, mà họ sử dụng những phát triển y hệt nhau mà không phải hợp tác trong việc cấp vốn. Ngay cả như vậy, thì công ty theo yêu cầu vẫn có được một vài lợi thế so với các đối thủ cạnh tranh, và một trong số chúng có thể là vị thế của công ty như một nguồn cấp vốn cho dự án nên cho phép công ty sử dụng ảnh hưởng sao cho ưu thế đó được đưa ra cho những phát triển trong đó nó có quan tâm nhất.
- CD với các chương trình: có lẽ, mô hình phổ biến nhất của dạng này là một trong những công ty cấp vốn cho những pháp triển cụ thể nào đó mà họ sau đó áp dụng cho việc phân phối các phần mềm của họ. Ví dụ, việc có một môi trường máy tính để bàn tốt có thể giúp nhiều để bán một CD với một phân phối GNU/Linux cụ thể nào đó, và vì thế, việc cấp vốn cho sự phát triển của nó có thể là một việc kinh doanh tốt cho bên bán các đĩa CD.

Chúng ta cần nhớ trong đầu rằng dưới tiêu đề này thì việc cấp vốn theo yêu cầu sẽ phải được thực hiện với một động lực về lợi nhuận, và vì thế cơ quan cấp vốn phải giành được một lợi nhuận tiềm tàng từ việc cấp vốn. Tuy nhiên, trong thực tế, thường ở đó sẽ có một sự kết hợp về động lực về lợi nhuận và sự vị tha khi một công ty cung cấp vốn cho một dự án tự do sẽ được thực hiện, từ đó nó mong đợi để có lợi nhuận một cách không trực tiếp.

Lưu ý

Trường hợp nổi tiếng về đóng góp vốn cho một dự án, dù khá là gián tiếp, là sự trợ giúp mà nhà xuất bản O'Reilly trao cho sự phát triển của Perl. Về bản chất tự nhiên, điều này không trùng khớp việc O'Reilly cũng là một trong những nhà xuất bản chính của các chủ đề có liên quan tới Perl.Trong mọi trường hợp, rõ ràng là O'Reilly không có sự độc chiếm đối với xuất bản phẩm của các sách dạng này, và rằng các nhà xuất bản khác cạnh tranh trong phân khu thị trường này, với các mức độ thành công khác nhau.

VA Software (ban đầu là VA Research và sau này là VA Linux) đã kết hợp một cách tích cực trong việc phát triển nhân Linux. Thông qua việc này, hãng đã đạt được, cùng với những hãng khác, sự duy trì tiếp tục có đảm bảo, mà nó đã đặc biệt sống còn cho hãng trong quan hệ với các khách hàng của hãng khi việc kinh doanh chính của hãng là bán các thiết bị với GNU/Linux được cài đặt sẵn.

Red Hat đã cấp vốn cho sự phát triển của các thành phần GNOME, cơ bản là giành được một môi trường máy tính để bàn cho phát tán của hãng, mà hãng đã đóng góp để gia tăng việc bán hàng của hãng. Như những trường hợp trước, các nhà sản xuất của các phát tán khác đã có lợi từ sự phát triển này, dù nhiều người trong số họ đã không hợp tác với dự án GNOME ở một mức độ như Red Hat (và khá ít các hãng đã hoàn toàn không hợp tác). Bất chấp thực tế này, Red Hat vẫn hưởng lợi từ sự đóng góp của hãng vào GNOME.

5.1.5 Việc cấp vốn như một sự đầu tư nội bộ

Có những công ty mà phát triển PMTD trực tiếp như một phần của mô hình kinh doanh của họ. Ví dụ, một công ty có thể quyết định bắt đầu một dự án tự do mới trong lĩnh vực nơi mà hãng tin tưởng rằng có những cơ hội kinh doanh, với ý tưởng về việc giành được sau đó sự hoàn vốn đầu tư. Mô hình này có thể được coi là một biến thể của việc cấp vốn trước đó (cấp vốn gián tiếp), và "những lợi ích có liên quan" có thể là những ưu thế mà công ty giành được từ việc sản xuất chương trình tự do. Nhưng vì trường hợp này chính là bản thân sản phẩm tự do mà nó được mong đợi để tạo ra lợi nhuận, nên nó dường như phù hợp để trao cho nó đầu đề riêng của nó. Dạng cấp vốn này làm nảy sinh một loạt các mô hình kinh doanh. Khi chúng ta phân tích chúng (trong phần 5.2) chúng ta cũng sẽ giải thích những ưu điểm mà một công ty thường giành được từ dạng đầu tư này trong một dự án và những phương pháp nào có xu thế được sử dụng để làm cho nó sinh lợi. Nhưng trong mọi trường hợp, chúng ta phải nhớ rằng đôi khi phần mềm theo yêu cầu có thể được phát triển đơn giản để thỏa mãn những nhu cầu của riêng công ty, và rằng chỉ sau đó công ty có thể quyết định được tung nó ra, và có lẽ, để mở một dòng kinh doanh dựa trên nó.

Lưu ý

Digital Creations (bây giờ là tập đoàn Zope) là một trong những trường hợp nổi tiếng nhất về một công ty đã cống hiến cho việc phát triển PMTD với sự mong đợi về việc kiếm được sự hoàn vốn đầu tư. Dự án tự do mà Zope đầu tư nhiều vào là một máy chủ ứng dụng mà nó giành được một số thành công nhất định. Lịch sử của nó với PMTD đã bắt đầu khi mà Digital Creations đang tìm kiếm vốn đầu tư rủi ro để phát triển máy chủ cho các ứng dụng sở hữu độc quyền của mình, khoảng năm 1998. Một trong những nhóm có quan tâm nhất trong việc đầu tư vào họ (Opticality Ventures) đã thiết lập như một điều kiện rằng sản phẩm kết quả phải là tự do, vì nếu không họ đã không biết làm thế nào họ có thể giành được một thị phần đáng kể. Digital Creations đã đồng ý theo tiếp cận này và ít tháng sau đã công bố phiên bản đầu tiên của Zope. Ngày nay, Tập đoàn Zope chuyên tâm trong việc tư vấn, huấn luyện và hỗ trợ cho các hệ thống quản trị nội dung dựa trên Zope, và các sản phẩm khác mà Zope là hòn đá tảng một cách chắc chắn.

Ximian (trước đó là Helix Code) là trường hợp nổi tiếng của sự phát triển các ứng dụng tự do trong môi trường doanh nghiệp. Được liên kết chặt chẽ từ ban đầu với dự án GNOME, Ximian đã sản xuất ra các hệ thống phần mềm như là Evolution (một trình quản lý thông tin cá nhân mà nó đưa vào một chức năng khá giống với Microsoft Outlook), Red Carpet (một hệ thống dễ sử dụng cho việc quản lý các gói trong một hệ điều hành) và MONO (một triển khai của một phần lớn của .NET). Hãng này được thành lập vào tháng 10/1999 và đã cuốn hút được nhiều lập trình viên từ GNOME, những người đã trở thành những thành viên của đội phát triển của nó (trong khi việc tiếp tục trong nhiều trường hợp để hợp tác với dự án GNOME). Ximian đã định vị bản thân hãng như một hãng kỹ thuật chuyên gia trong việc áp dụng GNOME, trong việc xây dựng các ứng dụng dựa trên GNOME, và nói chung, trong việc cung cấp các dịch vụ phát triển dựa trên PMTD, đặc biệt là các công cụ có liên quan tới môi trường máy tính để bàn. Vào tháng 08/2003, Ximian đã được Novell mua.

Hệ thống In Doanh nghiệp của Cisco (CEPS) (http://ceps.sourceforge.net/) [17] là một hệ thống quản lý in ấn cho các tổ chức mà sử dụng rất nhiều máy in. Nó đã được phát triển nội bộ trong Cisco để thỏa mãn các nhu cầu của riêng hãng và đã được đưa ra tự do vào năm 2000 theo giấy phép GNU GPL. Khó biết chắc những lý do mà Cisco làm điều này, nhưng chúng có thể có liên quan tới việc cấp vốn cho những đóng góp từ bên ngoài (các báo cáo lỗi, các trình kiểm soát mới, các bản vá, …). Trong mọi

trường hợp, những gì là rõ ràng là việc vì Cisco đã không có các kế hoạch để thương mại hóa sản phẩm này và thị trường tiềm năng của nó là không thật rõ, hãng đã có quá ít thứ để mất với quyết định này.

5.1.6 Các phương thức cấp vốn khác

Có những phương thức cấp vốn khác mà chúng khó phân loại theo những đầu đề ở trên. Như một ví dụ, chúng ta có thể nhắc tới những thứ sau:

- Sử dụng thị trường để đặt các lập trình viên và các khách hàng liên hệ được với nhau. Ý tưởng chứng minh cho phương thức cấp vốn này là việc, đặc biệt cho những phát triển nhỏ lẻ, khó đối với một khách hàng muốn một sự phát triển đặc biệt nào đó có được mối liên hệ với một lập trình viên có khả năng tiến hành nó một cách có hiệu quả.

  Để cải thiện tình trạng này, các thị trường phát triển của PMTD được thiết lập nơi mà các lập trình viên có thể quảng cáo các kỹ năng của họ và các khách hàng, những phát triển mà họ cần. Một lập trình viên và một khách hàng đạt được thỏa thuận; chúng ta có một tình trạng tương tự như đối với thứ đã được mô tả như là "việc cấp vốn bởi một bên có yêu cầu những cải tiến" (phần 5.1.3).

  SourceXchange

  SourceXchange là một ví dụ về một thị trường mà nó đặt các lập trình viên trong mối quan hệ với các khách hàng tiềm năng. Để quảng cáo cho một dự án, một khách hàng có thể trình bày một yêu cầu đề xuất (RFP) chỉ ra sự phát triển được yêu cầu và những tài nguyên mà nó đã được chuẩn bị để cung cấp cho sự phát triển đó. Các RFP đã được xuất bản trên site này. Khi một lập trình viên đọc một thứ thú vị với anh ta, thì anh ta có thể đưa ra một đề xuất cho nó. Nếu một lập trình viên và một khách hàng đồng ý về những điều khoản của sự phát triển, thì một dự án có thể bắt đầu. Thông thường, mỗi dự án đã được giám sát bởi một người kiểm tra ngang hàng, một người kiểm tra có trách nhiệm đảm bảo rằng lập trình viên này đã biên dịch với những đặc tả kỹ thuật và quả thực những đặc tả kỹ thuật này có ý nghĩa, và việc khuyến cáo về cách triển khai thông qua dự án này, việc đảm bảo những khả năng của người kiểm tra, việc đảm bảo thanh toán trong trường hợp các dự án được hoàn tất và đưa ra các công cụ giám sát (các dịch vụ để nó báo giá cho khách hàng). Dự án đầu tiên đã qua trung gian SourceXchange đã được hoàn tất vào tháng 03/2000, mà chỉ qua một năm sau, vào tháng 04/2001, site này đã đóng.

- Việc cấp vốn cho dự án thông qua bán phiếu nợ (bond). Ý tưởng đằng sau dạng cấp vốn này tương tự như của thị trường phiếu nợ thông thường được tiếp cận bởi các công ty, nhưng được tập trung vào việc phát triển PMTD. Nó có vài biến thể, nhưng một trong những thứ nổi tiếng hoạt động như sau. Khi một lập trình viên (một cá nhân hoặc một công ty) có một ý tưởng cho một chương trình mới, hoặc sự cải tiến cho một chương trình đang tồn tại, anh ta viết nó thành một đặc tả kỹ thuật, với một ước lượng giá thành cho sự phát triển của nó và phát hành các phiếu nợ cho việc xây dựng nó. Giá trị của những phiếu nợ này chỉ được thực hiện nếu dự án cuối cùng được hoàn thành. Khi lập trình viên đã bán đủ số phiếu nợ, thì sự phát triển bắt đầu, được cấp vốn với ghi nợ dựa trên chúng. Khi sự phát triển kết thúc, và một bên thứ 3 độc lập xác nhận rằng quả thực những gì đã được làm là phù hợp với những đặc tả kỹ thuật, thì lập trình viên "thực hiện" các phiếu nợ, thiết lập các khoản nợ, và những gì còn lại là lợi nhuận được làm từ sự phát triển này. Ai có thể có quan tâm trong việc mua các phiếu nợ này? Rõ ràng, những

người sử dụng mà họ muốn chương trình mới hoặc sự cải thiện cho một chương trình đang tồn tại sẽ được thực hiện. Ở một mức độ nào đó, hệ thống phiếu nợ này cho phép các bên có quan tâm thiết lập được các ưu tiên cho lập trình viên (ít nhất trong một phần), thông qua việc mua các phiếu nợ. Điều này cũng có nghĩa rằng giá thành của sự phát triển sẽ không được đảm bảo chỉ bởi một công ty, mà có thể được chia sẻ giữa vài (bao gồm cả các cá nhân) công ty, những người chỉ phải trả tiền một cách bổ sung nếu dự án kết luận thành công khi kết thúc. Một cơ chế tương tự đối với điều này đã được đề xuất một cách chi tiết hơn trong "Giao thức của người trình diễn của Phố Uôn", của Chris Rasch (2001) [191].

Thư mục tham khảo

Hệ thống phiếu nợ được mô tả được dựa trên giao thức của người trình diễn trên đường phố ("giao thức của người trình diễn trên đường phố"), trong: Hội thảo về USENIX về các Thủ tục của Thương mại Điện tử, 1998 [152], và "Giao thức và bản quyền số của người trình diễn trên đường phố", 1999 [153], một cơ chế dựa trên thương mại điện tử được thiết kế để tạo điều kiện thuận lợi cho việc cấp vốn tư nhân của những sáng kiến tự do. Ngắn gọn, bất kỳ ai có quan tâm trong một công việc cụ thể nào đó có thể chính thức hứa hẹn trả một số lượng tiền nào đó nếu công việc này được hoàn thành và được xuất bản như là tự do. Mục đích của nó là để tìm ra một cách mới cho việc cấp vốn cho những công việc tương đối nhỏ mà chúng được làm cho sẵn sàng cho mọi người, nhưng có thể được mở rộng theo nhiều cách (các phiếu nợ cho việc xây dựng PMTD là một trong số chúng). Chúng ta có thể thấy một trường hợp nhỏ của việc đặt ra nguồn gốc của giao thức này trong thực tế, giao thức có lý của người trình diễn trên đường phố (Paul Harrison, 2002, [137]) nơi mà http://www.csse.monash.edu.au/~pfh/circle/funding_results.htmlit được áp dụng để giành được vốn cho phần tài chính của The Circle, một dự án PMTD.

- Hợp tác của các lập trình viên: Trong trường hợp này, các lập trình viên của PMTD, thay vì làm việc riêng rẽ hoặc cho một công ty, tham gia vào một vài dạng hiệp hội (thường tương tự như một hợp tác xã). Theo tất cả những khía cạnh khác, nó hoạt động y hệt cách như một công ty, với một ý nghĩa phụ về cam kết đạo đức của nó đối với PMTD, mà có thể tạo thành một phần của qui chế công ty của nó (dù một công ty thông thường cũng có thể làm điều này). Trong dạng tổ chức này, chúng ta có thể thấy một loạt sự phối kết hợp của những công việc tình nguyện và được trả tiền. Một ví dụ là các lập trình viên tự do.
- Hệ thống quyên góp: Điều này liên quan tới việc làm cho một cơ chế thanh toán cho tác giả của một phần mềm cụ thể nào đó, thông qua trang web mà nó điều tiết dự án. Cách này, người sử dụng có quan tâm trong dự án tiếp tục đưa ra những phiên bản mới có thể hỗ trợ tài chính cho nó bằng việc quyên góp tự nguyện theo cách cấp vốn cho lập trình viên.

5.2 Các mô hình kinh doanh dựa trên PMTD

Bổ sung vào các cơ chế cấp vốn cho các dự án mà chúng ta đã nói, một khía cạnh khác liên quan tới kinh tế mà đáng để nhắc tới là các mô hình kinh doanh. Nói về các cơ chế cấp vốn, chúng ta đã nhắc tới một ít thoáng qua. Ở đây, trong phần này, chúng ta sẽ mô tả chúng theo một cách có phương pháp hơn.

Thông thường, chúng ta có thể nói rằng nhiều mô hình kinh doanh đang được khai thác xung quanh PMTD, một số này kinh điển hơn và một số khác thì sáng tạo hơn. Chúng ta cần tính tới rằng không dễ

dàng để sử dụng những thứ dựa trên việc bán các giấy phép, những mô hình được thấy chung nhất trong nền công nghiệp phần mềm, vì trong thế giới của PMTD cơ chế tài chính này là rất khó khai thác. Tuy nhiên, chúng ta có thể sử dụng những thứ dựa trên các dịch vụ đối với bên thứ 3, với sự thuận lợi là nó có khả năng đưa ra sự hỗ trợ hoàn chỉnh cho một chương trình mà không cần phải là người sản xuất ra nó.

Bán PMTD được thật nhiều cho một bản sao

Trong thế giới của PMTD khó mà lấy tiền được đối với việc sử dụng các giấy phép, nhưng không phải là không thể. Nói chung, không có điều nào trong những định nghĩa của PMTD ngăn cấm một công ty tạo ra một sản phẩm và chỉ phân phối nó cho bất kỳ ai mà họ trả một số tiền cụ thể nào đó. Ví dụ, một nhà sản xuất cụ thể nào đó có thể quyết định phân phối sản phẩm của mình với một giấy phép tự do, nhưng chỉ cho những ai trả 1,000 euro cho một bản sao (như trong thế giới kinh điển của PMSHĐQ).

Tuy nhiên, dù về mặt lý thuyết điều này là có thể, thì trong thực tế khó cho điều này xảy ra. Vì một khi nhà sản xuất đã bán bản sao đầu tiên, thì bất kỳ ai nhận nó có thể có động cơ để cố gắng và kiếm lại sự đầu tư của anh hoặc chị ta bằng việc bán nhiều bản sao hơn với giá thành thấp hơn (thứ gì đó mà không thể bị cấm bởi giấy phép của các chương trình nếu nó là tự do). Trong ví dụ trước, một người có thể cố gắng bán 10 bản sao giá 100 euro mỗi bản, nghĩa là ngoài ra sản phẩm này có thể đưa ra miễn phí (cũng có thể, điều này làm cho nó rất khó đối với nhà sản xuất gốc ban đầu để bán bản sao khác với giá 1,000 euro, vì sản phẩm có thể có được một cách hợp pháp với giá bằng 1/10 giá này). Dễ thấy quá trình này có thể tiếp tục theo kiểu thác nước cho tới khi các bản sao đã được bán với giá gần bằng giá thành của việc sao chép, mà với các công nghệ hiện nay thì thực tế là bằng 0.

Ngay cả như thế, và nhớ trong đầu là cơ chế được mô tả sẽ có nghĩa là thường thì một nhà sản xuất không thể đặt một cái giá (đặc biệt là một giá cao) chỉ dựa vào thực tế về sự phân phối chương trình, dù có những mô hình kinh doanh mà chúng hoàn toàn chỉ làm có thế. Một ví dụ là trường hợp của các phát tán GNU/Linux, mà được bán với một mức giá thấp so với các đối thủ cạnh tranh sở hữu độc quyền, nhưng lớn hơn (và thường là lớn hơn nhiều) giá thành để sao chép (ngay cả khi nó có thể được tải về một cách tự do từ Internet). Tất nhiên, trong những trường hợp này thì các yếu tố khác cũng đi kèm, như hình ảnh thương hiệu hoặc sự tiện lợi cho người tiêu dùng. Nhưng đây không là trường hợp duy nhất. Vì thế, thay vì nói rằng PMTD "không thể bán nhiều cho từng bản sao", chúng ta nên nhớ trong đầu rằng khó mà làm thế được, và có lẽ điều này sẽ tạo ra lợi nhuận ít hơn, nhưng có thể có các mô hình chính xác là dựa trên đó.

Đưa ra những hạn chế này (và cả những ưu thế), trong vài năm gần đây những biến thể trong các mô hình kinh doanh thông thường trong nền công nghiệp phần mềm đang được thử nghiệm, cùng một lúc những giải pháp khác mang tính sáng tạo hơn được tìm ra cho việc khai thác những khả năng mà PMTD đưa ra. Không nghi ngờ gì, trong ít năm sắp tới chúng ta sẽ thấy còn nhiều hơn những thử nghiệm trong lĩnh vực này, và cũng sẽ có được nhiều thông tin hơn về những mô hình nào có thể làm việc và theo những điều kiện hoàn cảnh nào.

Trong phần này chúng ta đưa ra toàn cảnh của các mô hình kinh doanh mà chúng ta thường gặp nhất ngày nay, chia thành các nhóm với mong muốn chỉ ra cho độc giả những gì chúng chia sẻ nói chung và những gì phân biệt được chúng, tập trung vào những thứ dựa trên sự phát triển và các dịch vụ xung quanh một sản phẩm PMTD.

Doanh số, trong trường hợp này, tới trực tiếp từ những hoạt động phát triển và các dịch vụ cho sản

phẩm, nhưng không nhất thiết bao hàm cả sự phát triển sản phẩm mới. Khi sự phát triển này xảy ra, những mô hình này có việc cấp vốn của các sản phẩm PMTD như một sản phẩm phụ, nghĩa là chúng là những mô hình đặc biệt thú vị với một ảnh hưởng rộng lớn một cách tiềm tàng lên thế giới của PMTD nói chung.

Trong mọi trường hợp, và dù ở đây chúng ta đưa ra một sự làm sáng tỏ khá rõ ràng, thì chúng ta phải không được quên rằng hầu hết tất cả các công ty trong thực tế sử dụng những sự kết hợp của các mô hình mà chúng ta mô tả, và với cả những mô hình truyền thống hơn khác.

5.2.1 Hiểu biết tốt hơn

Công ty mà đi theo mô hình kinh doanh này cố gắng kiếm lợi nhuận dựa trên sự hiểu biết của mình về một sản phẩm tự do (hoặc một tập hợp các sản phẩm). Doanh số của nó sẽ tới từ các khách hàng của những sản phẩm mà công ty sẽ bán các dịch vụ có liên quan tới sự hiểu biết đó: sự phát triển dựa trên sản phẩm, sự sửa đổi, áp dụng, cài đặt và tích hợp với các sản phẩm khác. Ưu thế cạnh tranh của công ty sẽ có liên quan chặt chẽ tới sự hiểu biết về sản phẩm: vì thế, công ty sẽ đặc biệt có vị thế tốt nếu công ty là nhà sản xuất hoặc một người tham gia tích cực trong dự án sản xuất sản phẩm phần mềm đó.

Đây là một trong những lý do các công ty mà sử dụng mô hình này có xu hướng sẽ là những thành viên tích cực trong các dự án có liên quan tới phần mềm mà đối với phần mềm đó họ cố gắng bán các dịch vụ: đây là một cách rất hữu hiệu để có được sự hiểu biết về nó, và quan trọng hơn, sự hiểu biết sẽ được thừa nhận. Chắc chắn, việc có khả năng để nói cho một khách hàng rằng các nhân viên của công ty bao gồm một loạt các lập trình viên về dự án mà sản sinh ra các phần mềm, mà, ví dụ, cần phải được thay đổi, có xu hướng để đưa ra được một sự đảm bảo tốt.

Mối quan hệ với các dự án phát triển

Vì thế, các công ty dạng này rất quan tâm trong việc truyền bán hình ảnh của việc có được sự hiểu biết tốt về những sản phẩm tự do chắc chắn nào đó. Một kết quả thú vị của điều này là việc hỗ trợ cho các dự án PMTD (ví dụ, bằng việc tham gia tích cực vào chúng, hoặc cho phép các nhân viên làm như vậy trong thời gian của ngày làm việc) là không vì thế chỉ là thứ gì đó thuần túy là tính thương người.

Ngược lại, có thể một trong những tài sản sinh lợi nhất của công ty, khi mà các khách hàng sẽ đánh giá nó rất tích cực như một dấu hiệu rõ ràng rằng công ty đó là có hiểu biết về sản phẩm theo yêu cầu. Cộng với, bằng cách này nó sẽ có khả năng tuân theo sự phát triển một cách chặt chẽ, cố gắng để làm cho chắc chắn, ví dụ, rằng những cải tiến được yêu cầu bởi các khách hàng sẽ trở thành một phần của sản phẩm được phát triển bởi dự án.

Phân tích điều này từ một quan điểm chung hơn, thì đây là một tình huống mà trong đó các bên tham gia, cả công ty và dự án, đều hưởng lợi từ sự hợp tác. Dự án hưởng lợi từ sự phát triển được thực hiện bởi công ty, hoặc vì một số các lập trình viên của nó được trả tiền (ít nhất bán thời gian) cho công việc của họ trong dự án. Công ty hưởng lợi trong sự hiểu biết về sản phẩm, hình ảnh đối với các khách hàng, và một mức độ ảnh hưởng đối với dự án. Dãy các dịch vụ được cung cấp bởi dạng công ty này có thể rất rộng, nhưng thường là sự phát triển tùy biến, những áp dụng thích nghi hoặc những tích hợp của các sản phẩm mà họ là những chuyên gia, hoặc việc tra cứu các dịch vụ nơi mà họ khuyến cáo cho các khách hàng của họ cách tốt nhất để sử dụng sản phẩm đó theo yêu cầu (đặc biệt nếu đây là một sản

phẩm phức tạp hoặc việc nó hoạt động đúng là sống còn cho khách hàng).

Ví dụ

Những ví dụ về các công ty mà đã sử dụng mô hình kinh doanh này như sau:

- LinuxCare (http://www.linuxcare.com) [45]. Được thành lập năm 1996, nó ban đầu đã cung cấp các dịch vụ tư vấn và hỗ trợ cho GNU/Linux và PMTD tại Mỹ, và nhân viên của nó về cơ bản là những chuyên gia về GNU/Linux. Tuy nhiên, trong năm 2002 mục tiêu của công ty đã thay đổi và kể từ đó công ty đã chú tâm vào việc cung cấp các dịch vụ hầu như độc nhất vô nhị đối với việc GNU/Linux chạy trên các máy ảo trong những máy tính lớn của IBM. Mô hình kinh doanh của công ty cũng đã thay đổi sang "những hiểu biết tốt hơn với những hạn chế", khi mà như một phần nền tảng của các dịch vụ mà công ty đưa ra là ứng dụng không tự do, Levanta.

- Alcôve (http://www.alcove.com) [3]. Được thành lập năm 1997 tại Pháp, nó chủ yếu đưa ra các dịch vụ tư vấn PMTD, tư vấn chiến lược, hỗ trợ và phát triển. Kể từ khi thành lập, Alcôve đã để các lập trình viên một loạt dự án tự do trong các nhân viên, cố gắng đổi lại hình ảnh của công ty về điều này. Công ty cũng đã thử đưa ra hình ảnh, nói chung, về một công ty có liên kết với cộng đồng PMTD, bằng việc hợp tác, ví dụ, với các hiệp hội người sử dụng và đưa ra công khai đối với những hợp tác của công ty với các dự án tự do (ví dụ, thông qua Alcôve - Labs [4]).

5.2.2 Hiểu biết tốt hơn với những hạn chế

Các mô hình này là tương tự như những gì được mô tả trong phần trước, nhưng cố gắng để hạn chế sự cạnh tranh mà họ có thể phải đối mặt. Trong khi trong các mô hình thuần túy dựa vào sự hiểu biết tốt hơn, bất kỳ ai cũng có thể, về nguyên tắc, tham gia vào sự cạnh tranh, vì phần mềm được sử dụng là y hệt như nhau (và tự do), trong trường hợp mà dự định là để tránh tình trạng đó bằng việc đặt ra những trở ngại cho sự cạnh tranh. Những trở ngại này có xu hướng tạo nên từ những bằng sáng chế hoặc những giấy phép sở hữu độc quyền, mà thường ảnh hưởng tới một phần nhỏ (nhưng cơ bản) của sản phẩm được phát triển. Điều này giải thích vì sao nhưng mô hình này có thể được xem như là mô hình pha trộn, theo nghĩa là chúng là nửa đường giữa PMTD và PMSHĐQ.

Trong mọi trường hợp, cộng đồng PMTD phát triển phiên bản riêng của nó, nghĩa là ưu thế cạnh tranh có thể biến mất, hoặc ngay cả quay đầu chống lại công ty theo yêu cầu nếu đối thủ cạnh tranh tự do trở thành chuẩn của thị trường và được yêu cầu bởi các khách hàng của riêng công ty.

Ví dụ

Có nhiều trường hợp sử dụng mô hình kinh doanh này, vì nó thường được coi là ít rủi ro hơn so với mô hình thuần túy của sự hiểu biết. Tuy nhiên, các công ty mà đã sử dụng nó đã tiến hóa theo những cách thức khác nhau. Một vài công ty như sau:

- Caldera (http://www.sco.com) [16]. Lịch sử của Caldera là phức tạp. Ban đầu, nó đã tạo ra phát tán GNU/Linux của riêng hãng, hướng vào các doanh nghiệp: Caldera OpenLinux. Vào năm 2001 nó đã mua phiên bản Unix từ SCO, và vào năm 2002 nó đã thay đổi tên của mình thành SCO Group. Chiến lược kinh doanh của nó đã thay đổi cũng thường xuyên như tên của nó, từ sự hỗ trợ toàn bộ cho GNU/Linux, sang vụ kiện pháp lý của nó chống lại IBM và Red Hat vào năm 2003 và việc bỏ phát tán của riêng nó. Nhưng trong mối liên quan tới việc đặt tên này, việc kinh

doanh của Caldera, ít nhất là cho tới năm 2002, là một mô hình rõ ràng về sự hiểu biết tốt hơn với những hạn chế. Caldera đã cố gắng khai thác sự hiểu biết của mình về nền tảng GNU/Linux, nhưng việc hạn chế sự cạnh tranh mà nó có thể đã đối mặt bằng việc đưa vào PMSHĐQ trong phát tán của nó. Điều này làm khó cho các khách hàng trong việc thay đổi phát tán một khi họ đã áp dụng nó, vì ngay cả dù các phát tán GNU/Linux khác được đưa vào phần tự do của Caldera OpenLinux, thì họ đã không đưa vào phần sở hữu độc quyền được.

- Ximian (http://ximian.com) [74]. Được thành lập năm 1999 dưới tên Helix Code bởi các lập trình viên có liên quan chặt chẽ với dự án GNOME, nó được mua bởi Novell vào tháng 08/2003. Hầu hết các phần mềm mà nó đã phát triển đã là tự do (nói chung một phần của GNOME). Tuy nhiên, trong một lĩnh vực rất đặc thù thì Ximian đã quyết định cấp phép cho một thành phần như là PMSHĐQ: kết nối cho Exchange (Connector to Exchange). Module này cho phép một trong những sản phẩm ngôi sao của nó, Evolution (một trình quản lý thông tin cá nhân mà bao gồm thư điện tử, chương trình nghị sự, lịch, …), để tương tác được với các máy chủ của Microsoft Exchange, mà nó được sử dụng một cách phổ biến trong các tổ chức lớn.
- Điều này giải thích cách mà nó đã cố gắng cạnh tranh với một ưu thế hơn các công ty khác mà đã đưa ra các dịch vụ dựa trên GNOME, có lẽ với các sản phẩm được phát triển bởi chính Ximian mà chúng không thể tương tác được một cách dễ dàng với Exchange. Với sự ngoại lệ của sản phẩm này, mô hình của Ximian đã trở thành một trong những "sự hiểu biết tốt hơn", và cũng đã dựa trên việc có nguồn của một chương trình (như chúng ta sẽ thấy sau này). Trong mọi trường hợp, thành phần này đã được tung ra như là PMTD vào năm 2005.

5.2.3 Nguồn của một sản phẩm PMTD

Mô hình này tương tự như mô hình dựa trên sự hiểu biết tốt hơn nhưng với một sự chuyên môn hóa, nghĩa là công ty sử dụng nó là nhà sản xuất, hầu như toàn bộ, của một sản phẩm tự do. Về cơ bản, ưu thế cạnh tranh gia tăng thông qua việc là các lập trình viên của sản phẩm theo yêu cầu, việc kiểm soát sự tiến hoá của nó và có nó trước khi có sự cạnh tranh. Tất cả những thứ này tạo vị thế cho công ty phát triển rất mạnh đối với các khách hàng mà họ đang tìm kiếm các dịch vụ cho chương trình đó. Hơn nữa, đây là mô hình rất thú vị xét về hình ảnh, vì công ty đã chứng minh sự phát triển của mình một cách tiềm tàng bằng việc tạo ra và duy trì ứng dụng theo yêu cầu, mà nó có thể rất hữu dụng khi nói về việc thuyết phục các khách hàng đối với các khả năng của công ty. Cũng vậy, nó tạo ra một hình ảnh tốt đối với cộng đồng PMTD nói chung, vì nó nhận được một sản phẩm mới từ công ty mà trở thành một phần của miền cộng đồng chung.

Ví dụ

Nhiều sản phẩm tự do đã bắt đầu được phát triển trong một công ty, và rất thường là công ty đã tiếp tục dẫn dắt sự phát triển tiếp sau của nó. Một số ví dụ:

- Ximian: Chúng ta đã nhắc tới cách mà nó một phần đã sử dụng mô hình về sự hiểu biết tốt hơn với những hạn chế. Nhưng nói chung, Ximian đã tuân theo mô hình rõ ràng dựa trên việc vừa là nguồn của các chương trình tự do. Các sản phẩm của hãng, như Evolution hoặc Red Carpet, đã từng được phân phối theo các giấy phép GPL. Tuy nhiên, những sản phẩm khác cũng quan trọng, như Mono, được phân phối chủ yếu theo các giấy phép MIT X11 hoặc LGPL. Trong mọi trường hợp, Ximian đã phát triển các sản phẩm hầu như độc nhất vô nhị từ ban đầu. Hãng đã cố

gắng hoàn vốn đầu tư vào những phát triển này bằng việc giành được các hợp đồng để làm cho chúng tiến hóa theo những cách nào đó, áp dụng chúng cho các nhu cầu của các khách hàng, và đưa ra sự tùy biến và duy trì.

- Zope Corporation (http://www.zope.com) [75]. Trong năm 1995 Digital Creations đã được thành lập, phát triển một sản phẩm sở hữu độc quyền cho quản lý các quảng cáo được phân loại trên web. Vào năm 1997 nó đã nhận được một sự đầu tư vốn từ, trong số những thứ khác, công ty đầu tư rủi ro có tên là Opticality Ventures. Những gì là kỳ lạ về sự đầu tư này (vào lúc đó) là điều kiện mà đã được ép buộc cho việc phân phối sản phẩm tiến hóa như là một PMTD, mà sau này đã trở thành Zope, một trong những trình quản trị nội dung nổi tiếng nhất trên Internet. Kể từ đó, mô hình kinh doanh của công ty là sản xuất Zope và các sản phẩm liên quan, và đưa ra các dịch vụ áp dụng và duy trì cho tất cả các sản phẩm đó.

5.2.4 Nguồn sản phẩm với những hạn chế

Mô hình này là tương tự như mô hình trước, nhưng tính tới việc hạn chế sự cạnh tranh hoặc tối đa hóa doanh số. Trong số những hạn chế chung nhất, chúng ta có thể thấy những điều sau:

Phân phối sở hữu độc quyền một thời gian, rồi tung ra như một PMTD. Có hoặc không có một sự hứa hẹn về một phân phối tự do sau này, mỗi phiên bản mới của sản phẩm được bán như là PMSHĐQ. Sau một khoảng thời gian chắc chắn nào đó (thông thường, khi một phiên bản mới được tung ra, cũng như một PMSHĐQ), thì phiên bản cũ được phân phối với một giấy phép tự do. Bằng cách này, công ty sản xuất giành được doanh số từ các khách hàng có quan tâm trong việc có những phiên bản mới, và cùng một lúc hạn chế được sự cạnh tranh, vì bất kỳ công ty nào muốn cạnh tranh mà có sử dụng sản phẩm đó chỉ có thể làm thế với phiên bản mới (chỉ sẵn sàng khi phiên bản mới sở hữu độc quyền được tung ra, mà nó được cải thiện một cách có hỗ trợ và hoàn chỉnh hơn).

Phân phối một cách hạn chế trong một giai đoạn. Trong trường hợp này, phần mềm là tự do vào thời điểm nó lần đầu được phân phối. Nhưng vì không có gì trong giấy phép tự do ép phải phân phối chương trình cho bất kỳ ai muốn nó (đây là thứ gì đó mà người sở hữu phần mềm có thể hoặc không làm), nhà sản xuất phân phối một thời gian chỉ cho các khách hàng của mình, những người trả tiền cho nó (thường ở dạng của một hợp đồng duy trì). Sau một thời gian, nó phân phối phần mềm cho bất kỳ ai, ví dụ bằng việc đặt phần mềm trong một tệp truy cập công cộng. Bằng cách này, nhà sản xuất giành được doanh số từ các khách hàng của mình, những người thừa nhận tính sẵn sàng ưu tiên này của phần mềm như một giá trị gia tăng. Về cơ bản, mô hình này chỉ làm việc nếu các khách hàng tới lượt mình không làm cho chương trình thành công cộng khi họ nhận được nó. Đối với những dạng khách hàng đặc biệt nào đó, thì điều này có thể không phải là phổ biến. Nói chung, trong các trường hợp các công ty phát triển giành được những lợi ích nhắc tới được, chứ không phải là giá thành bằng 0. Vì sự trì hoãn của sản phẩm để sẵn sàng cho cộng đồng PMTD, nên thực tế là không thể có khả năng đóng góp cho sự phát triển của nó, nghĩa là nhà sản xuất sẽ hưởng lợi rất ít từ những đóng góp từ bên ngoài.

Ví dụ

Một số công ty sử dụng mô hình kinh doanh này như sau:

- artofcode LLC (http://artofcode.com/) [9]. Từ năm 2000, artofcode bán Ghostscript theo 3 phiên

bản (trước đó Alladin Enterprises đã thực hiện điều này với một mô hình tương tự). Phiên bản mới nhất được phân phối là AFPL Ghostscript, theo một giấy phép sở hữu độc quyền (mà nó cho phép sử dụng cả sự phân phối không thương mại). Phiên bản tiếp sau (với khoảng trễ cỡ 1 năm) được phân phối như là GNU Ghostscript, theo giấy phép GNU GPL. Ví dụ, vào mùa hè năm 2003, phiên bản AFPL là 8.11 (được tung ra ngày 16/08), trong khi phiên bản GNU là 7.07 (được phân phối vào ngày 17/05, nhưng là phiên bản AFPL tương đương của nó đã được tung ra vào năm 2002). Cũng vậy, artofcode đưa ra một phiên bản thứ 3, với một giấy phép sở hữu độc quyền mà cho phép tích hợp nó với các sản phẩm không tương thích với GNU GPL (trong trường hợp này nó sử dụng một mô hình giấy phép đôi, mà chúng ta sẽ mô tả sau).

- Ada Core Technologies (http://www.gnat.com/) [2]. Nó được thành lập vào năm 1994 bởi những tác giả của trình biên dịch Ada 95, được phát triển bằng sự cấp vốn một phần từ chính phủ Mỹ, dựa vào GCC, trình biên dịch GNU. Ngay từ đầu thì các sản phẩm của nó đã là PMTD. Nhưng hầu hết chúng được đưa ra lần đầu cho các khách hàng như một phần của một hợp đồng duy trì. Ví dụ, trình biên dịch của nó, mà nó tiếp tục sẽ dựa trên Pro. Ada Core Technologies không đưa ra trình biên dịch này cho công chúng nói chung bằng bất kỳ phương thức nào, và thường thì bạn không thể thấy các phiên bản của nó trên Net. Tuy nhiên, với một sự chậm trễ khác nhau (khoảng 1 năm), Ada Core Technologies đưa ra các phiên bản trình biên dịch của nó cho công chúng, rất tương tự nhưng không có bất kỳ dạng hỗ trợ nào, ở dạng một tệp FTP nặc danh.

5.2.5 Các giấy phép đặc biệt

Theo các mô hình này, công ty sản xuất một sản phẩm mà nó phân phối theo 2 hoặc nhiều hơn các giấy phép. Ít nhất một trong số đó là PMTD, nhưng những cái khác thường là sở hữu độc quyền và cho phép sản phẩm được bán ít nhiều theo cách truyền thống. Thường thì, việc bán này được bổ sung với việc bán các dịch vụ tư vấn và phát triển có liên quan tới sản phẩm. Ví dụ, một công ty có thể phân phối một sản phẩm như là PMTD theo giấy phép GNU GPL, nhưng cũng đưa ra một phiên bản sở hữu độc quyền (cùng một lúc, và không có sự trì hoãn nào giữa 2 phiên bản đó) cho những ai không muốn những điều kiện của GPL, ví dụ, vì họ muốn tích hợp sản phẩm này với một sản phẩm sở hữu độc quyền (mà GPL không cho phép).

Ví dụ

Sleepycat Software (http://www.sleepycat.com/download/oslicense.html) [60]. Công ty này được thành lập năm 1996 và đã tuyên bố rằng hãng đã kiếm lợi nhuận từ đầu (mà chắc chắn là đáng kể trong một công ty liên quan tới phần mềm). Các sản phẩm của nó, bao gồm Berkeley DB (một hệ quản trị dữ liệu rất nổi tiếng vì nó có thể dễ dàng nhúng được vào trong các ứng dụng khác), được phân phối theo một giấy phép mà chỉ định rằng trong trường hợp nhúng với sản phẩm khác, thì nó phải đưa ra mã nguồn của cả 2. Sleepycat đưa ra các dịch vụ tư vấn và phát triển cho các sản phẩm của hãng, nhưng cũng chào chúng theo các giấy phép mà cho phép chúng được nhúng vào mà không phải phân phối mã nguồn. Tất nhiên, làm điều này theo một hợp đồng đặc biệt, và nói chung, theo một chế độ bán PMSHĐQ. Vào năm 2005 thì Oracle đã mua Sleepycat Software.

5.2.6 Bán thương hiệu

Dù có thể có được những sản phẩm rất tương tự với ít tiền hơn nhiều, nhiều khách hàng sẵn sàng trả thêm tiền để mua một *thương hiệu*. Nguyên tắc này được áp dụng bởi các công ty mà họ đầu tư vào việc thiết lập ra một thương hiệu với một hình ảnh tốt và được thừa nhận tốt mà nó cho phép họ sau đó bán các sản phẩm tự do với một khoản lợi nhuận đủ. Trong mọi trường hợp, họ không chỉ bán những sản phẩm đó, mà còn bán các dịch vụ mà các khách hàng cũng sẽ chấp nhận như một giá trị gia tăng.

Những trường hợp nổi tiếng nhất của mô hình kinh doanh này là các công ty mà bán các phát tán GNU/Linux. Các công ty này cố gắng bán thứ gì đó mà nói chung có thể giành được với giá thành thấp hơn nhiều từ Net (hoặc những nguồn khác với hình ảnh thương hiệu thấp hơn nhiều). Vì thế, họ phải làm cho các khách hàng nhận thức được thương hiệu của họ và được chuẩn bị để trả tiền cho giá thành bổ sung. Để làm được như vậy, họ không chỉ đầu tư vào quảng cáo công khai, mà họ cũng đưa ra những ưu điểm có mục tiêu (ví dụ, một phân phối được lắp ráp tốt hoặc một kênh phân phối mà đưa ra sự gần gũi cho khách hàng). Cũng vậy, họ có xu hướng đưa ra một số lượng lớn các dịch vụ xung quanh nó (từ việc huấn luyện cho các chương trình cấp chứng chỉ của bên thứ 3), để làm nhiều nhất cho hình ảnh thương hiệu.

Ví dụ

Red Hat (http://www.redhat.com) [56]. Red Hat Linux đã bắt đầu được phân phối vào năm 1994 (hãng này đã bắt đầu được biết tới như tên hiện này vào năm 1995). Một thời gian dài, Red Hat đã định thiết lập tên của hãng như một phát tán GNU/Linux xuất sắc (dù vào giữa những năm 2000 hãng chia sẻ vị thế này với các công ty khác như OpenSuSE, Ubuntu, và có lẽ cả Debian). Một vài năm Red Hat bán tất cả các dạng dịch vụ có liên quan tới phát tán này, GNU/Linux và các phần mềm nói chung.

5.3 Những phân loại mô hình kinh doanh khác

Các tài liệu của PMTD đưa ra những phân loại khác về các mô hình kinh doanh truyền thống. Như một ví dụ, đây là một ít.

5.3.1 Phân loại theo Hecker

Sự phân loại này được đưa ra trong "Thiết lập cửa hàng: kinh doanh PMNM" (Frank Hecker, 1998) [141] đã được sử dụng nhiều nhất trong sự quảng bá của OSI, và cũng là một trong những người đầu tiên cố gắng phân loại các doanh nghiệp mà đã nổi lên khoảng thời gian này. Tuy nhiên, nó đưa vào một loạt các mô hình mà có ít điều để làm với PMTD (nơi mà PMTD ít nhiều là bạn đường đối với mô hình chính). Trong mọi trường hợp, các mô hình mà nó mô tả là như sau:

- *Người bán sự hỗ trợ* (bán các dịch vụ có liên quan tới sản phẩm). Công ty khuyến khích một sản phẩm PMTD (mà nó đã phát triển hoặc nó tham gia tích cực vào trong đó) và bán các dịch vụ như là tư vấn hoặc áp dụng thích nghi cho những yêu cầu đặc thù nào đó.
- *Người dẫn đầu thua thiệt* (bán các sản phẩm sở hữu độc quyền khác). Trong trường hợp này,

chương trình tự do được sử dụng để quảng bá cách nào đó cho việc bán các sản phẩm sở hữu độc quyền khác có liên quan tới nó.

- *Bán các thiết bị* (bán phần cứng): Việc kinh doanh chính là bán phần cứng và PMTD được coi như một sự bổ sung mà có thể giúp công ty giành được một ưu thế cạnh tranh.
- *Bán các phụ tùng* (bán các phụ tùng): Các sản phẩm liên quan tới PMTD được bán, như là sách, các thiết bị máy tính, …
- *Bán các dịch vụ* (bán các dịch vụ): PMTD phục vụ để tạo ra một dịch vụ (thường có thể truy cập được một cách trực tuyến) từ đó công ty kiếm lợi nhuận.
- *Cấp phép cho thương hiệu* (bán một thương hiệu). Một công ty đăng ký thương hiệu mà nó có ý định liên kết với các chương trình PMTD, có thể nó đã tự phát triển. Sau đó nó kiếm lợi nhuận thông qua việc cấp phép sử dụng các thương hiệu này.
- *Bán nó, rồi giải phóng nó*. Đây là mô hình tương tự với người dẫn đầu thua thiệt, nhưng được thực hiện theo một cách thức tuần tự theo chu kỳ. Đầu tiên một sản phẩm được đưa ra thị trường như là PMTD. Nếu nó khá thành công, thì phiên bản tiếp sau được phân phối như là PMSHĐQ một thời gian, sau đó nó được giải phóng tự do. Khi đó, một phiên bản sở hữu độc quyền đang được phân phối, và cứ tiếp tục như thế.
- *Trao đặc quyền phần mềm*: Một công ty trao đặc quyền sử dụng thương hiệu của mình có liên quan tới một chương trình tự do cụ thể nào đó.

Lưu ý

Độc giả đã quan sát thấy rằng sự phân loại này là khá khác biệt so với sự phân loại mà chúng đã đã đưa ra, nhưng ngay cả như vậy thì một số các chủng loại cũng hầu như hoàn toàn khớp với một số của chúng ta.

5.4 Ảnh hưởng lên vị thế độc quyền

Thị trường phần mềm có xu thế hướng tới sự áp đảo của một sản phẩm trong từng phân khu của nó. Người sử dụng muốn hầu hết những nỗ lực được thực hiện trong việc học cách một chương trình làm việc được, các công ty muốn tuyển mộ những người quen biết với việc sử dụng các phần mềm của họ, và mỗi người muốn các dữ liệu mà họ vận hành sẽ quản lý được bởi các chương trình của các công ty và những người mà họ làm việc với những người đó. Điều này giải thích vì sao những sáng kiến được thiết kế để phá bỏ một tình trạng de facto trong đó một sản phẩm áp đảo một cách rõ ràng thị trường được định sẵn để sản sinh ra thứ y hệt nhiều hơn: nếu nó thành công, sản phẩm mới sẽ tới chiếm chỗ, và trong một khoảng thời gian ngắn chúng ta sẽ có một sản phẩm áp đảo mới. Chỉ những thay đổi về công nghệ mới sản sinh ra, trong một khoảng thời gian ngắn, sự không ổn định đủ để không một ai áp đảo được một cách rõ ràng.

Nhưng thực tế là có một sản phẩm áp đảo không nhất thiết phải dẫn tới việc tạo ra một sự độc quyền kinh doanh. Ví dụ, dầu khí là một sản phẩm mà hầu như áp đảo thị trường nhiên liệu cho các ô tô tư

nhân, nhưng (trong thị trường dầu khí tự do) có nhiều công ty sản xuất và công ty phân phối cùng sản phẩm y hệt nhau này. Trong thực tế, khi chúng ta nói về phần mềm, những gì đang gây lo ngại là những gì xảy ra khi một sản phẩm định áp đảo thị trường vì sản phẩm đó có một nhà cung cấp duy nhất có khả năng. PMTD đưa ra một giải pháp khác cho tình trạng đó: các sản phẩm tự do có thể được khuyến khích bởi một công ty cụ thể nào đó, nhưng công ty đó không kiểm soát được chúng, hoặc ít nhất ở một mức độ nào đó không như PMSHĐQ bắt chúng ta phải quen. Trong thế giới của PMTD, một sản phẩm áp đảo không nhất thiết dẫn tới sự độc quyền của một công ty. Mà ngược lại, bất chấp sản phẩm là áp đảo thị trường, thì nhiều công ty có thể cạnh tranh được trong việc cung cấp nó, cải thiện nó, và áp dụng thích nghi nó cho các nhu cầu của khách hàng và đưa ra các dịch vụ có liên quan tới nó.

5.4.1 Các yếu tố thiên vị cho các sản phẩm áp đảo

Trong phần mềm máy tính, thường thì có một sản phẩm áp đảo trong từng phân khúc thị trường. Và điều này là thông thường với vài lý do, trong số đó chúng ta có thể nhấn mạnh như sau:

- Các định dạng dữ liệu: Trong nhiều trường hợp định dạng dữ liệu được liên kết rất chặt chẽ tới một ứng dụng. Khi một số lượng đủ lớn mọi người sử dụng nó, thì định dạng dữ liệu này trở thành chuẩn de facto, và áp lực để sử dụng nó (và vì thế, ứng dụng đó) là khổng lồ.
- Các chuỗi phân phối: Thường thì, một trong những vấn đề với việc bắt đầu sử dụng một chương trình là việc có được một bản sao của nó. Và điều này thường là khó để tìm các chương trình mà không phải là những chương trình hàng đầu trong thị trường của nó. Các chuỗi phân phối là đắt giá để duy trì, nghĩa là khó cho các đối thủ cạnh tranh thiểu số với tới được cửa hàng máy tính nơi mà người sử dụng đầu cuối có thể mua chúng. Tuy nhiên, đối với sản phẩm áp đảo thì nó là dễ dàng: nơi đầu tiên có được quan tâm trong việc có nó sẽ là bản thân cửa hàng máy tính đó.
- Marketing: Việc marketing "tự do" mà một sản phẩm giành được một khi một tỷ lệ đáng kể người dân sử dụng nó là khổng lồ. "Truyền khẩu" cũng làm việc rất tốt khi chúng ta hỏi và trao đổi thông tin với mọi người mà chúng ta biết. Nhưng trên hết tất cả thì sự ảnh hưởng từ các phương tiện truyền thông là khổng lồ: các tạp chí máy tính sẽ tham chiếu nhiều lần tới một sản phẩm nếu nó dường như là một sản phẩm được sử dụng nhiều nhất; sẽ có các khóa đào tạo xung quanh nó, các cuốn sách mô tả nó, các cuộc phỏng vấn với những người sử dụng, …
- Đầu tư vào việc huấn luyện: Một khi thời gian và tiền bạc đã được bỏ ra vào việc huấn luyện cách mà một công cụ hoạt động, thì sẽ có một động lực cao không phải để thay đổi công cụ đó. Cũng vậy, công cụ đó thường là một công cụ mà đã áp đảo thị trường, vì nó dễ dàng hơn để tìm kiếm mọi người và các tư liệu để giúp dạy cách sử dụng nó.
- Các phần mềm cài đặt sẵn: Việc nhận một máy tính với phần mềm được cài đặt sẵn chắc chắn là một sự khích lệ lớn hướng tới việc sử dụng nó, ngay cả nếu nó phải trả tiền một cách riêng rẽ. Và thông thường, dạng phần mềm mà người bán máy tính được chuẩn bị để cài đặt sẵn sẽ chỉ là những phần mềm được sử dụng nhiều nhất.

5.4.2 Thế giới của PMSHĐQ

Trong thế giới của PMSHĐQ thì sự xuất hiện của một sản phẩm áp đảo trong mọi phân khúc là tương đương với một sự độc quyền về phần của công ty mà sản xuất ra nó. Ví dụ, chúng ta có những tình trạng độc quyền de facto (hoặc hầu như) của một sản phẩm và một công ty trong thị trường cho các hệ điều hành, cho việc xuất bản, cơ sở dữ liệu, thiết kế đồ họa, xử lý văn bản, bảng tính, … trên máy tính để bàn.

Và điều này là như vậy vì công ty theo yêu cầu có sự kiểm soát khổng lồ đối với sản phẩm dẫn đầu, nhiều tới mức mà chỉ họ có thể dẫn dắt sự tiến bộ của nó, các dòng cơ bản cùng với nó sẽ được phát triển, chất lượng của nó... Người sử dụng có rất ít sự kiểm soát, vì họ có rất ít động lực để coi những sản phẩm khác (vì những lý do mà chúng ta đã nhắc tới trong phần trước). Về vấn đề này, có ít sự cạnh tranh có thể làm, ngoại trừ việc cố gắng và thách đố vị thế áp đảo của sản phẩm bằng việc cải tiến các sản phẩm của riêng họ, (cố gắng và kháng cự lại những lý do đó), thường với thành công hạn chế.

Tình trạng này đặt toàn bộ khu vực này vào tay của chiến lược của công ty áp đảo. Tất cả mọi nhân tố đều phụ thuộc vào nó, và ngay cả sự phát triển của công nghệ phần mềm trong lĩnh vực đó cũng sẽ bị dàn xếp vì những cải tiến mà nó thực hiện cho sản phẩm của mình. Nói chung, đây là một tình trạng nơi mà những ảnh hưởng kinh tế tồi tệ nhất của một sự độc quyền nảy sinh, và đặc biệt, thiếu động lực đối với công ty áp đảo chỉnh sửa các sản phẩm cho các nhu cầu (luôn tiến hóa) của các khách hàng của nó, khi chúng trở thành một thị trường bị giam hãm.

5.4.3 Tình trạng với PMTD

Tuy nhiên, trong trường hợp của PMTD thì một sản phẩm áp đảo không tự động biến thành một sự độc quyền của doanh nghiệp. Nếu sản phẩm là tự do, thì bất kỳ công ty nào cũng có thể làm việc được với nó, cải thiện nó, áp dụng nó cho các nhu cầu của khách hàng, và nói chung, giúp nó tiến bộ. Hơn nữa, chính xác nhờ có vị thế áp đảo của nó, sẽ có nhiều công ty có quan tâm làm việc với nó. Nếu nhà sản xuất "gốc ban đầu" (công ty mà ban đầu đã phát triển ra sản phẩm này) mong muốn duy trì được trong kinh doanh, thì nó sẽ phải cạnh tranh với tất cả họ và vì thế sẽ có động lực cao để làm cho sản phẩm của hãng tiến bộ một cách chính xác theo những con đường mà người sử dụng muốn. Tất nhiên, nó sẽ có ưu thế có sự hiểu biết tốt hơn về chương trình, nhưng điều đó không phải là tất cả. Họ sẽ phải cạnh tranh vì mỗi khách hàng.

Vì thế, sự xuất hiện của các sản phẩm áp đảo trong thế giới của PMTD, truyền vào sự cạnh tranh hơn nữa giữa các công ty. Và với nó thì những người sử dụng nắm lại được sự kiểm soát: các công ty trong cạnh tranh không thể làm bất kỳ thứ gì ngoài phải nghe theo họ nếu họ muốn sống sót. Và điều này chính xác là những gì đảm bảo rằng sản phẩm được cải tiến.

Các sản phẩm tự do mà áp đảo trong khu vực của nó

Đã từ lâu, Apache đã là người dẫn đầu trong thị trường các máy chủ web. Nhưng có nhiều công ty đằng sau Apache, từ những công ty rất lớn (như IBM) cho tới những công ty khác nhỏ hơn nhiều. Và tất cả các công ty này không có sự lựa chọn nào khác ngoài việc cải tiến nó hoàn toàn và thường bằng việc đóng góp cho dự án này bằng những cải tiến của họ. Bất chấp thực tế là Apache hầu như là một sự độc

quyền trong nhiều lĩnh vực (ví dụ, nó hầu như là máy chủ web duy nhất được xem xét trên nền tảng GNU/Linux hoặc *BSD), nó không phụ thuộc vào một công ty duy nhất nào cả, mà vào hàng tá các công ty theo nghĩa đen.

Những phát tán của GNU/Linux cũng là trường hợp thú vị. GNU/Linux chắc chắn không phải là một sự độc quyền, mà có lẽ là lựa chọn thứ 2 trong thị trường hệ điều hành. Và điều này đã không đặt ra một tình trạng nơi mà một công ty có sự kiểm soát đối với nó. Ngược lại có hàng chục các phát tán được làm bởi các công ty khác nhau, mà chúng tự do cạnh tranh trong thị trường. Mỗi phát tán trong số chúng sẽ đưa ra những cải tiến, mà những đối thủ cạnh tranh của nó phải áp dụng trong sự rủi ro sẽ bị tụt hậu. Hơn nữa, chúng không thể bị lạc quá xa khỏi những gì là "chuẩn của GNU/Linux", vì điều này có thể sẽ bị từ chối bởi người sử dụng như một "sự đi ra khỏi chuẩn". Tình trạng này sau vài năm tăng trưởng thị phần cho GNU/Linux chỉ ra cho chúng ta hàng chục công ty mà họ cạnh tranh và cho phép hệ thống này tiến hóa. Và một lần nữa, tất cả chúng bám theo việc làm thỏa mãn cho những yêu cầu của người sử dụng. Đây là cách duy nhất họ có thể nắm lại trong thị trường này.

GCC là một sản phẩm áp đảo trong thế giới của các trình biên dịch C và C++ đối với thị trường GNU/Linux. Và vấn đề này không dẫn tới bất kỳ tình trạng độc quyền nào của công ty, ngay cả Cygnus (bây giờ là Red Hat) đã có trách nhiệm một thời gian dài đối với việc điều phối sự phát triển của nó. Có nhiều công ty mà họ thực hiện những cải tiến cho hệ thống này và tất cả bọn họ cạnh tranh với nhau theo sự thích hợp đặc biệt để thỏa mãn được những yêu cầu của người sử dụng của họ. Trên thực tế, khi một công ty hoặc tổ chức đặc biệt nào đó đã thất bại trong nhiệm vụ phối hợp (hoặc một vài người sử dụng đã nhận thức được điều này) thì đã có chỗ cho dự án được rẽ nhánh, với 2 sản phẩm cùng chạy song song trong một thời gian, cho tới khi chúng phải quay trở lại cùng nhau (như bây giờ xảy ra với GCC 3.x),

5.4.4 Các chiến lược cho việc trở thành một sự độc quyền với PMTD

Bất chấp thực tế là thế giới của PMTD là không thân thiện hơn nhiều đối với các độc quyền doanh nghiệp hơn là thế giới của PMSHĐQ, vẫn có những chiến lược mà một công ty có thể sử dụng để tiếp cận một tình trạng áp đảo độc quyền một thị trường. Những thực tế này là thông thường trong nhiều khu vực khác của nền kinh tế và để ngăn cản chúng chúng ta có các cơ quan mà họ điều chỉnh sự cạnh tranh, mà nó giải thích vì sao chúng ta sẽ không đi quá nhiều vào chi tiết về chúng. Tuy nhiên, chúng ta sẽ nhắc tới một điều, về điểm này, là đặc biệt cho thị trường phần mềm, và nó đã được trải nghiệm trong một số tình huống cụ thể nào đó: sự chấp nhận chứng nhận sản phẩm của bên thứ ba.

Khi một công ty mong muốn phân phối một sản phẩm phần mềm (tự do hoặc sở hữu độc quyền) mà hoạt động trong sự kết hợp với những phần mềm khác, thì thường phải "chứng nhận" sản phẩm đó đối với một sự kết hợp cụ thể nào đó. Nhà sản xuất cam kết đưa ra các dịch vụ (cập nhật, hỗ trợ, giải quyết các vấn đề, …) chỉ nếu khách hàng đảm bảo rằng sản phẩm này là đang được sử dụng trong một môi trường được chứng nhận. Ví dụ, nhà sản xuất của một chương trình quản trị cơ sở dữ liệu có thể chứng nhận cho sản phẩm của mình đối với một phát tán GNU/Linux cụ thể nào đó, chứ không phải phát tán khác. Điều này ngụ ý rằng các khách hàng của nó sẽ phải sử dụng phát tán GNU/Linux đó hoặc quên việc có được sự hỗ trợ của nhà sản xuất (mà nếu sản phẩm là sở hữu độc quyền có thể sẽ không thể trong thực tế). Nếu một nhà sản xuất cụ thể nào đó định đạt được một vị thế áp đảo rõ ràng như một sản phẩm được chứng nhận bởi bên thứ 3, thì người sử dụng sẽ không có bất kỳ sự lựa chọn nào khác ngoài

việc sử dụng sản phẩm đó. Nếu theo sự chứng nhận phân khúc đó là quan trọng, thì chúng ta sẽ một lần nữa đối mặt với một tình trạng độc quyền doanh nghiệp.

Lưu ý

Đối với điểm này, trong thị trường đối với các phát tán GNU/Linux chúng ta đang bắt đầu thấy một ít các trường hợp có xu thế hướng tới một sự độc quyền de facto thông qua sự chứng nhận. Ví dụ, có nhiều nhà sản xuất các sản phẩm sở hữu độc quyền mà họ chỉ chứng nhận các sản phẩm trên một phát tán được đưa ra của GNU/Linux (rất thường là Red Hat Linux). Cho tới nay điều này còn chưa gây ra một tình trạng độc quyền cho bất kỳ công ty nào, mà nó có thể nhờ vào thực tế rằng sự chứng nhận là không phù hợp đối với những người sử dụng trong thị trường các phát tán GNU/Linux. Nhưng chỉ tương lai mới nói được liệu ở thời điểm nào đó tình trạng này tiếp cận được một sự độc quyền de facto hay không.

Dù vậy, điều quan trọng phải ghi nhớ trong đầu 2 bình luận có liên quan ở trên. Bình luận đầu là những vị thế độc quyền này sẽ không dễ dàng đạt được, và trong mọi trường hợp sẽ đạt được thông qua các cơ chế "không phải phần mềm" nói chung (không giống như tình trạng sản phẩm độc quyền, mà như chúng ta đã thấy là khá thông thường, đạt được thông qua các cơ chế thuần túy liên quan tới công nghệ thông tin và các mẫu dạng sử dụng của nó).

Bình luận thứ 2 là nếu tất cả các phần mềm được sử dụng là tự do, thì chiến lược đó đã hạn chế được những cơ hội thành công (nếu có). Một nhà sản xuất có thể muốn có nhiều công ty để chứng nhận cho các sản phẩm của nó, nhưng các khách hàng sẽ luôn có khả năng xem xét các công ty khác nhau cho các dịch vụ và sự hỗ trợ khác so với những thứ mà đã chứng nhận cho nó, nếu họ coi đó là thích đáng.

6 PMTD và hành chính nhà nước

> "[...] Đối với phần mềm sẽ được chấp nhận cho Nhà nước, thì nó về mặt kỹ thuật không chỉ cần phải có khả năng thực hiện một nhiệm vụ, mà còn là những điều kiện ký kết hợp đồng của nó phải đáp ứng được một loạt yêu cầu về việc cấp phép, mà không có chúng thì Nhà nước không thể đảm bảo cho các công dân của mình rằng các dữ liệu của họ đang được xử lý một cách phù hợp, với trách nhiệm về tính bảo mật và tính có thể truy cập được theo thời gian, vì chúng là những khía cạnh mang tính sống còn cao đối với trách nhiệm chung của Nhà nước".
>
> Edgar David Villanueva Núñez (thư trả lời cho tổng giám đốc của Microsoft Peru, 2001).

Các cơ quan nhà nước, cả những cơ quan với khả năng làm luật và những cơ quan chuyên tâm đối với việc quản lý điều hành Nhà nước ("các cơ quan hành chính nhà nước"), đóng một vai trò rất quan trọng ở những nơi mà việc áp dụng và khuyến khích các công nghệ được quan tâm. Dù cho tới năm 2000 những cơ quan này về mặt thực tiễn đã không chỉ ra sự quan tâm trong hiện tượng của PMTD (với một số ngoại lệ), thì tình trạng này đã bắt đầu thay đổi từ đó. Một mặt, nhiều cơ quan hành chính nhà nước đã bắt đầu sử dụng PMTD như một phần hạ tầng công nghệ thông tin của họ. Mặt khác, trong vai trò của họ như là những người khuyến khích xã hội thông tin, một số đã bắt đầu khuyến khích trực tiếp hoặc gián tiếp sự phát triển và sử dụng PMTD. Hơn nữa, một số cơ quan làm luật đã bắt đầu chú ý (từng chút một) tới PMTD, đôi khi ưu tiên cho sự phát triển của nó, đôi khi gây cản trở cho nó, và đôi khi chỉ nắm lấy sự hiện diện của nó vào để xem xét.

Trước khi đi sâu vào chi tiết, điều quan trọng phải nhớ rằng từ lâu PMTD đã được phát triển mà không có sự ủng hộ dứt khoát nào (hoặc ngay cả quan tâm) từ các cơ quan nhà nước. Vì lý do này, sự chú ý gần đây mà nó đang diễn ra từ nhiều trong số họ không phải là không có sự đối kháng, lúng túng và những vấn đề. Hơn nữa, trong ít năm vừa qua, những sáng kiến có liên quan tới các chuẩn mở đang giành được xung lượng, thường có kết quả trong những đo đếm (ít nhiều một cách trực tiếp) có liên quan tới PMTD.

Trong chương này chúng ta sẽ cố gắng mô tả hiện trạng và những tính chất riêng biệt của PMTD trong mối liên quan tới khu vực "nhà nước" này.

6.1 Ảnh hưởng lên các cơ quan hành chính nhà nước

Một vài nghiên cứu đã được thực hiện, tập trung vào việc sử dụng PMTD trong các cơ quan hành chính nhà nước (ví dụ, "PMNM cho cơ quan hành chính nhà nước", 2004 [159]; "PMNM trong chính phủ điện tử (CPĐT), phân tích và những khuyến cáo được rút ra bởi một nhóm làm việc theo ban lãnh đạo về công nghệ của Đan Mạch", 2002 [180]; "PMTD/PMNM: các cơ hội của xã hội thông tin cho châu Âu", 1999 [132], và "Trường hợp cho sự khuyến khích của chính phủ đối với PMNM", 1999 [213]. Tiếp theo, chúng ta sẽ thảo luận một số trong số những tài liệu đáng chú ý nhất đó (cả tích cực và tiêu cực).

6.1.1 Những ưu điểm và những ảnh hưởng tích cực

Một số ưu điểm của việc sử dụng PMTD trong cơ quan hành chính nhà nước và những viễn cảnh mới chủ chốt mà nó đưa ra là như sau:

1. Việc phát triển nền công nghiệp bản địa

Một trong những ưu điểm chính của PMTD là khả năng của việc phát triển một nền công nghiệp phần mềm bản địa. Khi chúng ta sử dụng PMSHĐQ, mọi chi phí bỏ ra cho các giấy phép đi trực tiếp vào nhà sản xuất sản phẩm, và sự mua sắm làm tăng cường cho vị thế của nhà sản xuất, mà không nhất thiết là tiêu cực, nhưng lại rất không có hiệu quả đối với vùng miền mà ở đó cơ quan hành chính nhà nước có liên quan khi chúng ta xem xét tới giải pháp thay thế của việc sử dụng PMTD.

Trong trường hợp này, các công ty bản địa sẽ có khả năng cạnh tranh trong việc cung cấp các dịch vụ (và bản thân chương trình) cho cơ quan hành chính nhà nước, theo những điều kiện rất tương tự đối với bất kỳ công ty nào. Hãy nói rằng bằng cách này hay cách khác cơ quan hành chính nhà nước đang xóa bỏ sự bất bình đẳng về sân chơi và làm dễ dàng hơn cho mọi người để cạnh tranh trong sân chơi đó.

Và tất nhiên, rằng "bất kỳ ai" bao gồm cả các công ty bản địa, những người sẽ có cơ hội để khai thác những ưu thế cạnh tranh của họ (sự hiểu biết tốt hơn về những nhu cầu của khách hàng, sự gần về địa lý, …).

2. Sự độc lập với nhà cung cấp và cạnh tranh thị trường

Rõ ràng, bất kỳ tổ chức nào cũng sẽ thích phụ thuộc vào một thị trường cạnh tranh hơn là vào một nhà cung cấp duy nhất có khả năng áp đặt những điều kiện theo đó nó cung cấp sản phẩm của mình. Tuy nhiên, trong thế giới của cơ quan hành chính nhà nước, sự ưu tiên này trở thành một yêu cầu cơ bản, và ngay cả một bổn phận pháp lý trong một số trường hợp. Nói chung, cơ quan hành chính nhà nước không thể chọn để hợp đồng với một nhà cung cấp được đưa ra, mà phải chỉ định những yêu cầu của nó theo một cách sao cho bất kỳ công ty có quan tâm nào mà đáp ứng được những đặc tính nhất định nào đó và đưa ra được sản phẩm hoặc dịch vụ đáp ứng được yêu cầu, đều có thể được lựa chọn cho một hợp đồng.

Một lần nữa, trong trường hợp của PMSHĐQ, từng sản phẩm chỉ có một nhà cung cấp (ngay cả nếu nó sử dụng một số lượng các trung gian). Nếu một sản phẩm cụ thể được chỉ định, thì cơ quan hành chính nhà nước cũng sẽ quyết định nhà cung cấp nào để trao hợp đồng. Và trong nhiều trường hợp hầu như không thể tránh được việc chỉ định một sản phẩm cụ thể, khi chúng ta làm việc với các chương trình máy tính. Những lý do về tính tương thích bên trong cơ quan hoặc việc tiết kiệm trong huấn luyện và quản trị hệ thống, hoặc nhiều hơn nữa, thường làm cho một cơ quan hành chính quyết định sử dụng một sản phẩm nhất định nào đó.

Cách duy nhất thoát khỏi tình trạng này là bằng việc làm cho sản phẩm chỉ định đó thành tự do. Bằng cách này, bất kỳ công ty nào có quan tâm cũng sẽ có khả năng cung cấp nó và cũng bất kỳ dạng dịch vụ nào có liên quan tới nó (chỉ tùy thuộc vào các khả năng và sự hiểu biết của công ty về sản phẩm). Hơn nữa, trong trường hợp của dạng hợp đồng này, thì cơ quan hành chính nhà nước có thể thay đổi nhà cung cấp trong tương lai nếu muốn, và không có bất kỳ vấn đề kỹ thuật nào, vì ngay cả nếu nó thay đổi công ty, thì nó vẫn sẽ sử dụng sản phẩm y hệt.

3. Tính mềm dẻo và sự thích nghi đối với các yêu cầu đặc thù

Dù sự thích nghi đối với những yêu cầu đặc thù là thứ gì đó mà bất kỳ tổ chức sử dụng máy tính nào cũng cần tới, thì những khác biệt của các cơ quan hành chính nhà nước làm cho điều này trở thành một yếu tố rất quan trọng cho việc thâm nhập sâu một cách thành công của một hệ thống phần mềm. Trong trường hợp của PMTD, sự thích nghi được thực hiện dễ dàng hơn nhiều, và quan trọng hơn, có thể dựa vào một thị trường cạnh tranh nếu việc hợp đồng là cần thiết. Khi cơ quan hành chính nhà nước mua một sản phẩm sở hữu độc quyền, việc sửa đổi nó thường liên quan tới việc đạt được một hợp đồng với nhà sản xuất, người mà là bên duy nhất có thể làm được về mặt pháp lý (và thường là cả về mặt kỹ thuật). Theo những hoàn cảnh này, khó mà thương thảo đặc biệt được nếu nhà sản xuất không có quan tâm một cách đặc biệt trong thị trường được đưa ra bởi cơ quan hành chính nhà nước cụ thể đó.

Tuy nhiên, bằng việc sử dụng một sản phẩm tự do, cơ quan hành chính nhà nước có thể sửa đổi nó như mong muốn, nếu nó sử dụng những cá nhân có khả năng, hoặc đưa ra thuê ngoài sự sửa đổi đó. Về nguyên tắc, việc thuê ngoài có khả năng là một số công ty có thể mong đợi sẽ được cạnh tranh với nhau. Một cách tự nhiên, điều này hướng tới việc làm cho giá thành rẻ hơn và cải thiện được chất lượng.

Trường hợp các phát tán GNU/Linux

Trong vài năm qua tại Tây Ban Nha, đã trở thành thông thường đối với một số chính quyền vùng để tạo ra những phát tán GNU/Linux của riêng họ. Xu hướng này đã bắt đầu với GNU/Linux, nhưng bây giờ có nhiều hơn nữa. Dù một số chuyên gia đã chỉ trích sự hiện diện của các phát tán này, thì điều này là ví dụ rõ ràng về tính mềm dẻo mà PMTD cho phép. Bất kỳ cơ quan hành chính nhà nước nào, bằng việc bỏ ra những tài nguyên khá vừa phải, có thể hợp đồng sửa lại một GNU/Linux áp dụng được cho những nhu cầu và ưu tiên của nó, mà thực tế không có bất kỳ hạn chế nào. Ví dụ, nó có thể thay đổi giao diện máy tính để bàn, chọn tập hợp các ứng dụng và ngôn ngữ mặc định, cải tiến việc bản địa hóa các ứng dụng, … Nói cách khác: nếu muốn, máy tính để bàn (và bất kỳ yếu tố phần mềm nào mà làm việc trên máy tính) cũng có thể được thích nghi cho những yêu cầu chính xác nào đó.

Tất nhiên, sự sửa lại cho hợp này sẽ liên quan tới một số chi phí, nhưng kinh nghiệm chỉ ra rằng nó có thể đạt được một cách khá là rẻ, và xu hướng này xuất hiện để chỉ ra rằng nó sẽ ngày một dễ dàng hơn (và rẻ hơn) để tiến hành tùy biến các phát tán.

4. Áp dụng dễ dàng hơn các chuẩn mở

Đưa ra bản chất rất tự nhiên của chúng, các chương trình tự do thường sử dụng các chuẩn mở không sở hữu độc quyền. Trong thực tế, hầu như hoàn toàn bằng định nghĩa, bất kỳ khía cạnh nào của một chương trình tự do mà chúng ta có thể quan tâm để xem xét đều có thể sản xuất lại một cách dễ dàng và, vì thế, không phải là sở hữu độc quyền. Ví dụ, các giao thức được sử dụng bởi một chương trình tự do để tương tác với các chương trình khác có thể được nghiên cứu và sản xuất lại, nghĩa là chúng sẽ không phải là sở hữu độc quyền. Hơn nữa, rất thông thường và theo mối quan tâm của bản thân các dự án, chúng ta sẽ cố gắng sử dụng các chuẩn mở.

Trong mọi trường hợp, bất chấp lý do, một thực tế là các chương trình tự do thường sử dụng các chuẩn không phải là sở hữu độc quyền cho sự trao đổi các dữ liệu. Những ưu điểm của điều này đối với các cơ quan hành chính nhà nước là lợi hơn nhiều so với bất kỳ tổ chức nào khác, vì sự khuyến khích các chuẩn sở hữu độc quyền (ngay cả một cách gián tiếp, bằng việc sử dụng chúng) là có nhiều nỗi lo lắng

hơn nhiều. Và trong ít nhất một khía cạnh, việc sử dụng các chuẩn không phải là sở hữu độc quyền là nền tảng cơ bản, nơi mà sự tương tác với các công dân được quan tâm, vì họ phải không bị ép phải mua bất kỳ sản phẩm nào từ một công ty cụ thể nào để có khả năng tương tác được với cơ quan hành chính nhà nước.

5. Sự xem xét kỹ lưỡng về an ninh của cơ quan hành chính nhà nước

Đối với một cơ quan hành chính nhà nước, việc có khả năng đảm bảo rằng các hệ thống máy tính của nó chỉ làm những gì chúng phải làm là một bổn phận cơ bản, và trong nhiều quốc gia, là một yêu cầu pháp lý. Thường những hệ thống này quản lý các dữ liệu riêng tư, mà các bên thứ 3 có thể quan tâm (ví dụ như các dữ liệu thuế, các hồ sơ tội phạm, các dữ liệu trưng cầu hoặc bầu cử, …). Nếu một ứng dụng sở hữu độc quyền được sử dụng, mà không có mã nguồn có sẵn, khó mà đảm bảo rằng các ứng dụng sẽ xử lý các dữ liệu theo cách mà nó phải làm. Nhưng ngay cả nếu nó cung cấp mã nguồn, thì những khả năng của một cơ quan hành chính nhà nước đảm bảo rằng nó không chứa các mã nguồn lạ sẽ rất bị hạn chế. Chỉ nếu nhiệm vụ này có thể được ủy quyền đều đặn một cách như thường lệ cho các bên thứ 3, và cộng với bên có quan tâm mới có thể soi xét kỹ lưỡng nó được, có thể cơ quan hành chính nhà nước sẽ chắc chắn một cách hợp lý rằng nó phù hợp với những nhiệm vụ cơ bản của mình, hoặc ít nhất tiến hành những biện pháp trong quyền hạn của nó để làm thế.

6. Tính sẵn sàng về lâu dài

Nhiều dữ liệu được xử lý bởi các cơ quan hành chính nhà nước, và các chương trình được sử dụng để tính toán chúng, cần phải sẵn sàng trong hàng thập kỷ và hàng thập kỷ. Rất khó để đảm bảo rằng bất kỳ chương trình sở hữu độc quyền nào sẽ sẵn sàng sau thời gian đó, đặc biệt nếu ý tưởng là để nó làm việc trên nền tảng thông dụng vào thời điểm đó trong tương lai. Ngược lại, có khả năng là nhà sản xuất đó có thể đã đánh mất mối quan tâm vào sản phẩm và đã không còn đưa nó lên những nền tảng mới nữa, hoặc chỉ được chuẩn bị để làm thế với rất nhiều tiền. Một lần nữa, chúng ta cần nhớ rằng chỉ duy nhất nhà sản xuất có thể đưa ra sản phẩm, nghĩa là những thương thảo sẽ khó khăn. Tuy nhiên, trong trường hợp của PMTD, ứng dụng là sẵn sàng, với sự chắc chắn, sao cho bất kỳ ai cũng có thể đưa nó ra và để nó thực hiện chức năng theo các nhu cầu của cơ quan hành chính nhà nước. Nếu điều này không xảy ra cùng một lúc, thì cơ quan hành chính nhà nước có thể luôn tìm kiếm được một số công ty để tiến hành chào tốt nhất cho công việc này. Điều này đảm bảo rằng ứng dụng và dữ liệu mà nó xử lý sẽ sẵn sàng khi cần thiết.

7. Ảnh hưởng vượt ra ngoài sự sử dụng về phần của cơ quan hành chính nhà nước

Nhiều ứng dụng được sử dụng hoặc khuyến khích bởi các cơ quan hành chính nhà nước cũng được sử dụng bởi các khu vực khác của xã hội. Vì lý do này, bất kỳ sự đầu tư nhà nước nào trong sự phát triển của một sản phẩm tự do là có lợi không chỉ cho bản thân cơ quan hành chính nhà nước, mà còn cho tất cả các công dân, những người sẽ có khả năng sử dụng sản phẩm này cho những nhiệm vụ máy tính của họ, có lẽ với những cải tiến được thực hiện bởi cơ quan hành chính nhà nước.

Lưu ý

Một trường hợp rất đặc biệt, nhưng là trường hợp với ảnh hưởng khổng lồ, mà thể hiện việc sử dụng này tốt hơn các tài nguyên của nhà nước là bản địa hóa chương trình (sự thích nghi cho những sử dụng và tùy biến của cộng đồng). Dù khía cạnh có thể thấy được nhất về bản địa hóa là sự dịch chương trình và

tài liệu của nó, thì có những thứ khác mà chúng cũng bị ảnh hưởng bởi nó (từ việc sử dụng đơn vị tiền tệ bản địa cho tới việc trình bày các định dạng ngày tháng và thời gian của cộng đồng theo yêu cầu, tới việc sử dụng các ví dụ trong tài liệu và các cách thức diễn đạt được áp dụng cho các khách hàng bản địa).

Trong mọi trường hợp, rõ ràng nếu một cơ quan hành chính nhà nước sử dụng vốn để bản địa hóa một ứng dụng cụ thể nào đó chỉnh sửa ứng dụng đó cho những nhu cầu của mình, thì hơn thế là những nhu cầu trùng khớp với những nhu cầu của các công dân, nghĩa là nó sẽ tạo ra, không chỉ một chương trình làm thỏa mãn những yêu cầu của riêng nó, mà còn, những nhu cầu mà nó có thể được làm cho sẵn sàng cho bất kỳ công dân nào cũng có khả năng để thực hiện được nó tốt nhất mà không cần chi phí bổ sung nào. Ví dụ, khi một cơ quan hành chính nhà nước cấp vốn để thích nghi một chương trình máy tính cho một ngôn ngữ mà nó được sử dụng trong cộng đồng của mình, thì nó sẽ không chỉ có khả năng sử dụng chương trình đó bên trong các văn phòng của riêng cơ quan, mà cũng đưa nó ra được cho mọi người dân, với mọi thứ mà điều này có liên quan đối với sự phát triển một xã hội thông tin.

Thư mục tham khảo

Các độc giả có quan tâm về những ưu điểm của PMTD trong báo cáo đối với cơ quan hành chính nhà nước, được viết theo ngữ cảnh của Mỹ năm 1999, có thể tra cứu "Trường hợp đối với sự khuyến khích của chính phủ đối với PMNM" (Mitch Stoltz, 1999) [213].

6.1.2 Những khó khăn của việc áp dụng và những vấn đề khác

Tuy nhiên, dù có nhiều ưu điểm cho cơ quan hành chính nhà nước sử dụng PMTD, cũng còn nhiều khó khăn mà cần phải đối mặt khi nói về việc đặt nó vào trong thực tế. Trong số chúng, chúng ta có thể đặc biệt nhắc tới những thứ sau đây:

1. Thiếu sự hiểu biết và cam kết chính trị

   Vấn đề đầu tiên mà PMTD gặp phải đối với việc đưa vào cơ quan hành chính nhà nước là việc các tổ chức khác nhau chia sẻ một cách không nghi ngờ rằng: những người ra quyết định vẫn còn chưa biết tới PMTD.

   May thay, đây là một vấn đề mà dần dần được giải quyết, nhưng trong nhiều khu vực các cơ quan hành chính nhà nước, PMTD vẫn còn bị thừa nhận như thứ gì đó lạ lẫm, nên các quyết định về việc sử dụng nó vẫn còn liên quan tới những rủi ro nhất định nào đó.

   Bổ sung cho điều này, chúng ta có xu hướng đi sang một vấn đề về việc ra quyết định chính trị. Ưu điểm chính của PMTD đối với cơ quan hành chính nhà nước không phải là giá thành (vì giá thành, trong mọi trường hợp, là cao, đặc biệt khi chúng ta nói về một triển khai cho một số lượng lớn các máy tính trạm), nhưng như chúng ta đã nói, những lợi ích là vượt trên tất cả mọi chiến lược. Và vì thế, quyết định rơi vào phạm vi chính trị, hơn là phạm vi kỹ thuật. Không có ý chí chính trị để thay đổi các hệ thống phần mềm và triết lý được cam kết với nó, thì khó có sự tiến bộ trong sự phát triển của PMTD trong các cơ quan hành chính nhà nước.

2. Sự áp dụng kém cỏi các cơ chế hợp đồng

Các cơ chế hợp đồng mà cơ quan hành chính nhà nước sử dụng ngày nay, trải từ các mô hình đấu thầu công khai tới việc chi tiết hóa theo các điều khoản giá thành, về cơ bản được thiết kế cho mua sắm các sản phẩm công nghệ thông tin và mua sắm các dịch vụ có liên quan tới chương trình. Tuy nhiên, khi chúng ta sử dụng PMTD, thường sẽ không có sản phẩm để mà mua, hoặc giá của nó là không đáng kể. Ngược lại, để tận dụng được ưu điểm của các cơ hội được đưa ra bởi PMTD, điều thuận tiện để có khả năng làm hợp đồng các dịch vụ xung quanh nó. Điều này là cần thiết, trước khi PMTD có thể được sử dụng một cách nghiêm túc, phải thiết kế được các cơ chế mà chúng tạo điều kiện thuận lợi cho việc làm hợp đồng trong những trường hợp này.

3. Thiếu chiến lược triển khai

   Thường thì một cơ quan hành chính nhà nước có thể bắt đầu sử dụng PMTD đơn giản vì sự mua sắm có giá thấp hơn. Thông thường trong những trường hợp đối với sản phẩm theo yêu cầu sẽ được kết hợp vào hệ thống máy tính mà không có việc lên kế hoạch tiếp theo, và nói chung, không có một chiến lược toàn cầu cho việc sử dụng và làm thành hầu hết các PMTD. Điều này gây ra hầu hết các lợi ích của nó sẽ bị mất cùng cách thức đó, vì mọi thứ sẽ lắng xuống như đối với việc sử dụng một sản phẩm rẻ hơn, trong khi chúng ta đã nhìn thấy rằng, nói chung, những lợi ích chính là về một dạng khác.

   Nếu bổ sung vào điều này, sự biến chuyển là không được thiết kế một cách phù hợp, thì sử dụng PMTD có thể gây ra giá thành đáng kể, và chúng ta sẽ thấy rằng trong những trường hợp bị cô lập nào đó, nằm ngoài một khung công việc rõ ràng, thì việc sử dụng của PMTD trong cơ quan hành chính nhà nước có thể sẽ không thành công và gây nản lòng.

4. Sự khan hiếm các sản phẩm PMTD trong những phân khúc nhất định

   Sự triển khai của PMTD trong bất kỳ tổ chức nào cũng có thể gặp phải sự thiếu các giải pháp thay thế tự do có chất lượng đối với những dạng nhất định nào đó các ứng dụng. Đối với những trường hợp này, giải pháp là phức tạp: tất cả những thứ chúng ta có thể làm là cố gắng khuyến khích sự xuất hiện của sản phẩm tự do mà chúng ta cần. May thay, các cơ quan hành chính nhà nước là trong một vị thế tốt để nghiên cứu một cách nghiêm túc liệu họ có thể có quan tâm trong việc khuyến khích hoặc ngay cả cấp vốn hoặc cùng cấp vốn cho sự phát triển của sản phẩm đó hay không. Chúng ta nên nhớ rằng các mục tiêu của nó thường bao gồm việc cung cấp cho các công dân của nó với sự truy cập tốt nhất tới xã hội thông tin, ví dụ, hoặc việc khuyến khích nền kinh tế công nghiệp bản địa. Chắc chắn, sự tạo ra nhiều chương trình tự do sẽ có một ảnh hưởng tích cực lên cả 2 mục tiêu, nghĩa là chúng ta phải bổ sung thêm vào sự tính toán chỉ những giá thành/lợi ích trực tiếp, còn những quyết định như vậy sẽ có những lợi ích gián tiếp.

5. Tính tương hợp với các hệ thống đang tồn tại

   Không thông thường cho một sự chuyển đổi đầy đủ sang PMTD được thực hiện với toàn bộ hệ thống cùng một lúc. Vì thế, điều quan trọng là phần chúng ta muốn chuyển đổi phải tiếp tục hoạt động tốt với phần còn lại bằng những phần mềm phải tương hợp được với nhau. Đây là vấn được biết rõ với bất kỳ sự chuyển đổi nào (ngay cả nếu nó là một sản phẩm sở hữu độc quyền), nhưng có thể có một ảnh hưởng đặc biệt trong trường hợp của PMTD. Trong mọi trường hợp,

nó sẽ là thứ gì đó được tính tới khi nghiên cứu sự chuyển đổi. May thay, chúng ta có thể thường áp dụng PMTD mà những nhu cầu phải được cài đặt để tương hợp được một cách thích đáng với các hệ thống khác, mà nếu điều này là cần thiết, thì điểm này sẽ phải được xem xét khi tính toán ngân sách cho giá thành chuyển đổi.

6. Chuyển đổi các dữ liệu

   Đây là một vấn đề chung của bất kỳ sự chuyển đổi nào sang các ứng dụng mới mà sử dụng các định dạng dữ liệu khác, ngay cả nếu chúng là sở hữu độc quyền. Trên thực tế, trong trường hợp của PMTD thì vấn đề này thường được làm dịu bớt, vì thường phải tiến hành một nỗ lực đặc biệt để dự kiến trước được càng nhiều định dạng và chuẩn trao đổi dữ liệu có thể càng tốt. Mà thường thì các dữ liệu phải được chuyển đổi. Và giá thành của việc làm này là cao. Vì thế, trong việc tính toán giá thành của một sự chuyển đổi tiềm năng sang PMTD, yếu tố này cần phải được xem xét một cách thận trọng.

6.2 Hành động của cơ quan hành chính nhà nước trong thế giới PMTD

Các cơ quan hành chính nhà nước gây ảnh hưởng tới thế giới của phần mềm ít nhất theo 3 cách:

- Bằng việc mua các chương trình và dịch vụ có liên quan tới chúng. Các cơ quan hành chính nhà nước, như là những người sử dụng lớn của phần mềm, là những người chơi chính trong thị trường phần mềm.
- Bằng việc khuyến khích các cách thức khác nhau của việc sử dụng (và mua sắm) các chương trình nhất định nào đó của các cá nhân hoặc công ty. Sự khuyến khích này đôi khi đạt được bằng việc đưa ra những khuyến khích về tài chính (giảm thuế, những khuyến khích trực tiếp, …), đôi khi thông qua các thông tin và khuyến cáo, đôi khi bằng cách "đi theo ví dụ của tôi"...
- Bằng việc cấp vốn (trực tiếp hoặc gián tiếp) cho các dự án nghiên cứu phát triển mà chúng thiết kế tương lai của phần mềm.

Trong từng lĩnh vực này thì PMTD có thể đưa ra những ưu điểm nhất định (bổ sung thêm cho những ưu điểm đã được mô tả trong phần trước) về lợi ích cho cả cơ quan hành chính nhà nước và xã hội nói chung.

6.2.1 Làm sao thỏa mãn nhu cầu của các cơ quan hành chính nhà nước?

Các cơ quan hành chính nhà nước là những người tiêu dùng lớn của công nghệ thông tin. Ở những nơi mà phần mềm được quan tâm, họ thường mua các sản phẩm có sẵn cũng như các hệ thống được tùy biến. Từ quan điểm này, họ về cơ bản là các trung tâm mua sắm lớn, tương tự như đối với các công ty lớn, nhưng với những tính năng đặc thù của riêng họ. Ví dụ, trong nhiều lĩnh vực, các quyết định mua sắm của các cơ quan hành chính nhà nước được giả thiết để tính tới không chỉ giá thành đối với các tham số chức năng mà còn cả những thứ khác, như là ảnh hưởng của sự mua sắm lên lợi ích của nền công nghiệp và xã hội hoặc những mối quan tâm về lâu dài, mà chúng cũng có thể là quan trọng. Trong

mọi trường hợp, thường thì hiện nay với các phần mềm có sẵn là sử dụng các sản phẩm sở hữu độc quyền hàng đầu của thị trường. Số lượng tiền của nhà nước bỏ ra bởi các chính quyền thành phố, vùng và quốc gia, và các cơ quan hành chính nhà nước quốc tế (như Liên minh châu Âu) vào việc mua Windows, Office hoặc các giấy phép của các sản phẩm tương tự chắc chắn là đáng kể. Nhưng dần dần các giải pháp tự do đang bắt đầu thâm nhập vào thị trường này. Ngày một gia tăng, các giải pháp dựa trên PMTD đang được xem xét cho các máy chủ, và các sản phẩm như OpenOffice.og, và GNU/Linux với GNOME và KDE đang ngày một được sử dụng gia tăng cho máy tính để bàn.

Những gì có thể giành được từ sự chuyển đổi này sang PMTD? Để minh họa chỉ những thứ này, hãy xem xét kịch bản sau đây. Hãy giả thiết là với một dung sai của những gì được bỏ ra cho 2 hoặc 3 sản phẩm "ngôi sao" sở hữu độc quyền của tất cả các cơ quan hành chính nhà nước của châu Âu (hoặc có thể của những cơ quan hành chính nhà nước của bất kỳ một quốc gia phát triển tầm trung nào) để cải tiến và áp dụng các chương trình tự do có sẵn hiện nay sao cho trong vòng từ 1 tới 2 năm họ có thể sẽ sẵn sàng cho sự sử dụng hàng loạt số đông, ít nhất cho các nhiệm vụ tiêu chuẩn nhất định nào đó (nếu họ còn chưa có). Hãy tưởng tượng, ví dụ, một nỗ lực được phối hợp, trong một phạm vi quốc gia hoặc châu Âu, nơi mà tất cả các cơ quan hành chính nhà nước đã tham gia vào một nhóm có trách nhiệm về việc quản lý các vụ thầu này. Trong một quãng thời gian ngắn có thể có một nền công nghiệp "bản địa" đặc thù trong việc tạo ra những cải tiến và áp dụng. Và các cơ quan hành chính nhà nước có thể chọn giữa 3 hoặc 4 phát tán tự do được sản xuất bởi nền công nghiệp đó. Để khuyến khích sự cạnh tranh, mỗi công ty có thể được đền bù theo số lượng mà các cơ quan hành chính nhà nước chọn sử dụng phát tán của các công ty. Và kết quả toàn bộ của chiến dịch này, vì nó có thể là PMTD, cũng có thể sẵn sàng cho những người sử dụng là các công ty và cá nhân, mà trong nhiều trường hợp có thể có những nhu cầu tương tự như của các cơ quan hành chính nhà nước.

Trong trường hợp của các phần mềm được tùy biến, quá trình thông thường hiện hành liên quan tới việc hợp đồng cho các chương trình cần thiết từ một công ty theo một mô hình sở hữu độc quyền. Bất kỳ sự phát triển nào được thực hiện theo yêu cầu của cơ quan hành chính nhà nước là tài sản của công ty mà phát triển nó. Và thông thường, cơ quan hành chính nhà nước ký hợp đồng bị trói vào nhà cung cấp này trong mọi thứ có liên quan tới những cải tiến, cập nhật và hỗ trợ, theo một vòng không hợp cách mà làm cho sự cạnh tranh khó khăn và làm chậm quá trình hiện đại hóa của các cơ quan hành chính nhà nước. Còn tồi tệ hơn là việc thường chương trình y hệt như vậy được bán đi bán lại cho các cơ quan hành chính nhà nước tương tự, bằng việc áp dụng theo từng trường hợp giá thành nảy sinh ra như cho việc phát triển hoàn toàn từ đầu.

Hãy xem xét một lần nữa cách mọi thứ có thể sẽ khác. Một nhóm các cơ quan hành chính nhà nước cần một dạng phần mềm được tùy biến có thể yêu cầu rằng kết quả giành được sẽ là PMTD. Điều này có thể cho phép các cơ quan hành chính nhà nước khác hưởng lợi từ công việc và trong trung hạn có thể gây sự quan tâm cho họ trong việc hợp tác trong nhóm sao cho những yêu cầu cụ thể của họ có thể được xem xét. Vì các phần mềm được tạo ra có thể là tự do, có thể sẽ không có bổn phận để ký hợp đồng cho những cải tiến và áp dụng đối với cùng nhà cung cấp, nghĩa là sự cạnh tranh có thể vào được thị trường (mà hiện tại hầu như là bị giam cầm). Trong tất cả các tình huống được nêu ở trên, giá thành cuối cùng cho bất kỳ cơ quan hành chính nhà nước nào có liên quan có thể không bao giờ lớn hơn so với nếu áp dụng một mô hình sở hữu độc quyền.

Liệu các kịch bản này có là khoa học viễn tưởng không nhỉ? Như chúng ta sẽ thấy sau đây, có những

sáng kiến rụt rè theo những hướng tương tự cho những thứ được mô tả. Bổ sung vào việc hỗ trợ để tạo ra và duy trì một nền công nghiệp trong khu vực mua sắm của cơ quan hành chính nhà nước, PMTD đưa ra những ưu điểm cụ thể hơn theo miền công cộng. Ví dụ, cách hiệu quả nhất của việc có phần mềm được phát triển trong những ngôn ngữ thiểu số (một mối quan tâm cơ bản của nhiều cơ quan hành chính nhà nước). Nó có thể cũng giúp nhiều hướng tới việc duy trì sự độc lập mang tính chiến lược về lâu dài và việc đảm bảo tính có thể truy cập được các dữ liệu trong sự chăm sóc của các cơ quan hành chính nhà nước về lâu dài. Đối với tất cả các lý do này, các cơ quan nhà nước đang ngày càng quan tâm trong PMTD như những người sử dụng.

Một vài trường hợp có liên quan tới các cơ quan hành chính nhà nước của Đức

Vào tháng 07/2003 phiên bản ổn định đầu tiên của Kolab đã được tung ra, một sản phẩm của dự án Kroupware. Kolab là một hệ thống trợ giúp công nghệ thông tin tự do cho nhóm làm việc (*groupware*) dựa trên KDE. Lý do cho việc nhắc tới dự án này là việc ban đầu nó từng là một vụ thầu của chính phủ của Văn phòng Liên bang Đức về An ninh Thông tin (BSI). Vụ thầu này tìm kiếm một giải pháp mà nó có thể, một mặt, tương hợp được với Windows và Outlook, và mặt khác, với GNU/Linux và KDE. Từ vụ thầu được đệ trình, đề xuất chung của 3 công ty, Erfrakon, Intevation và Klarälvdalens Datakonsult, đã được trao hợp đồng, với đề xuất của họ để cung cấp một giải pháp tự do một phần dựa vào các phần mềm đã được phát triển bởi dự án KDE, được hoàn thiện với những phát triển tự do của riêng hãng, tạo ra Kolab.

Vào tháng 05/2003, Tòa thị chính của Munich (Đức) đã phê chuẩn sự chuyển đổi sang GNU/Linux và các ứng dụng của bộ phần mềm văn phòng tự do cho tất cả các máy tính để bàn, khoảng 14,000 chiếc tổng số. Quyết định làm điều này đã không thuần túy về tài chính: các khía cạnh chiến lược và chất lượng cũng đã được đưa ra xem xét, theo các tác giả. Trong một phân tích tổng hợp mà đã được thực hiện trước khi đưa ra quyết định này, giải pháp mà cuối cùng đã được chọn (GNU/Linux cộng với OpenOffice.og, về cơ bản) đã giành được 6,218 điểm (từ tối đa là 10,000 điểm) đối chọi lại ít hơn 5,000 điểm một chút là giải pháp "truyền thống" dựa trên các phần mềm của Microsoft đã giành được.

Vào tháng 07/2003, Koordinierungs-und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (KBSt), thuộc Bộ Nội vụ Đức, đã đưa ra công khai tài liệu *Leitfaden für die Migration von Basissoftwarekomponenten auf Server- und Arbeitsplatzsystemen* [107] ('Hướng dẫn chuyển đổi cho các thành phần phần mềm cơ bản của các máy tính chủ và máy tính trạm'), mà nó đưa ra cho các cơ quan nhà nước của Đức một tập hợp các chỉ dẫn về cách để chuyển đổi cho các giải pháp dựa trên PMTD. Những chỉ dẫn này được thiết kế cho bên ra quyết định để đánh giá liệu một chuyển đổi sang PMTD có phù hợp không và làm thế nào để triển khai sự chuyển đổi nếu quyết định đó được đưa ra.

6.2.2 Khuyến khích xã hội thông tin

Các cơ quan nhà nước bỏ ra nhiều tài nguyên vào những khích lệ để thúc đẩy chi tiêu cho công nghệ thông tin. Đây là một công cụ tuyệt vời, mà nó có thể giúp các công nghệ mới mở rộng được trong xã hội. Nhưng đây cũng là một công cụ nguy hiểm. Ví dụ, nó có thể không là một ý tưởng rất tốt để khuyến khích sử dụng của xã hội đối với Internet bằng việc khuyến cáo một trình duyệt nhất định nào đó như việc khuyến khích cho vị thế độc quyền de facto của một công ty, vì về lâu dài điều này có thể là tiêu cực cho xã hội mà chúng ta đang cố gắng làm cho có lợi. Một lần nữa, PMTD có thể giúp trong

các tình huống này. Thứ nhất, nó là trung lập đối với các nhà sản xuất, vì không ai có được sự độc chiếm đối với bất kỳ chương trình tự do nào. Nếu một cơ quan hành chính nhà nước muốn khuyến khích sử dụng một họ các chương trình tự do, thì nó có thể mở một vụ thầu, mà bất kỳ công ty nào trong lĩnh vực này cũng có thể tham gia thầu, để quản lý sự phân phối của nó đối với các công dân, sự cải tiến hoặc chức năng mở rộng của nó, … Thứ hai, nó có thể giúp nhiều trong các khía cạnh kinh tế. Ví dụ, trong nhiều trường hợp với cùng một số vốn có thể bỏ ra trong việc mua một số lượng nhất định giấy phép của các chương trình sở hữu độc quyền được đem cho việc mua một bản sao tự do và việc hợp đồng hỗ trợ hoặc áp dụng thích nghi cho nó; hoặc ngay cả trong việc thương thảo với một nhà sản xuất PMSHĐQ về các quyền để chuyển sản phẩm của nó thành PMTD.

Trong một lĩnh vực riêng rẽ, chúng ta có thể tưởng tượng việc dành riêng một phần số lượng được phân bổ cho việc tin học hóa các trường học bằng việc tạo ra một phát tán GNU/Linux được áp dụng cho những yêu cầu đào tạo của các trường tiểu học. Và với phần còn lại của vốn được cấp cho hợp đồng hỗ trợ đối với việc duy trì các phần mềm này trong các trường học, sao cho các phần mềm không chỉ là "để trình diễn" mà mọi người thực sự có trách nhiệm cho việc đảm bảo rằng nó làm việc một cách đúng đắn. Điều này không chỉ bao trùm những yêu cầu giáo dục mà còn tạo ra một thị trường cho các công ty, thường là các công ty bản địa, có khả năng cung cấp các dịch vụ duy trì. Và tất nhiên, nó để lại con đường cho tương lai hoàn toàn mở: phần mềm sẽ không bị cô lập chỉ trong một ít năm nghĩa là chúng ta lại cần phải bắt đầu từ không có gì, mà thay vào là nó có thể được cập nhật từng chút một, năm này qua năm khác, duy trì những lợi ích của chương trình với một sự đầu tư tương tự.

Lưu ý

Các độc giả mà quen với các sáng kiến của nhà nước về PMTD sẽ nhận thức được trường hợp của gnuLinEx trong ví dụ này. Về cuối năm 2001, Chính quyền vùng Extremadura (Tây Ban Nha) đã quyết định sử dụng một phát tán GNU/Linux để máy tính hóa tất cả các trường công trong vùng. Để làm việc này, nó đã cấp vốn cho việc xây dựng gnuLinEx, một phát tán GNU/Linux dựa trên Debian mà đã được công bố vào mùa thu năm 2002, và chắc chắn rằng nó là một yêu cầu trong tất cả các vụ thầu mua thiết bị máy tính cho các trường học. Hơn nữa, nó đã bắt đầu huấn luyện các chương trình cho các giáo viên, tạo các tư liệu giảng dạy và mở rộng kinh nghiệm trong các lĩnh vực khác. Vào giữa năm 2003, nó được coi là kinh nghiệm thành công, khi nó đã mở rộng một cách có tổ chức tới các vùng khác (ví dụ, Andalucia, cũng tại Tây Ban Nha, thông qua dự án Guadalinex).

6.2.3 Thúc đẩy nghiên cứu

PMTD cũng cung cấp những lợi ích đáng chú ý nơi mà các chính sách nghiên cứu phát triển (R&D) được quan tâm. Tiền của nhà nước đang được sử dụng để cấp vốn cho một số lượng lớn các phát triển phần mềm mà xã hội không ngừng hưởng lợi, ngay cả một cách gián tiếp. Thông thường, vốn cấp cho các chương trình R&D của nhà nước, toàn bộ hoặc một phần, dành cho các dự án để tạo ra các phần mềm mà thực sự không lo lắng về các quyền mà công chúng sẽ có đối với chúng. Trong nhiều trường hợp thì các kết quả, không có một kế hoạch thương mại hóa nào thích đáng, được đệ trình một cách đơn giản và để lại để hứng bụi. Trong những trường hợp khác, cũng chính những con người đã cấp vốn cho một chương trình thông qua các khoản thuế sẽ dừng việc cấp tiền cho nó một lần nữa nếu họ muốn sử dụng nó (khi họ cần phải mua các giấy phép để sử dụng).

PMTD đưa ra một sự lựa chọn thú vị, mà các đơn vị chức trách về các chính sách đổi mới sáng tạo trong nhiều cơ quan hành chính nhà nước sẽ thường bắt đầu xem xét với sự thận trọng. Đặc biệt khi nghiên cứu là sự cạnh tranh trước (phổ biến nhất trong trường hợp cấp vốn nhà nước), thì thực tế là các chương trình kết quả là tự do sẽ cho phép toàn bộ nền công nghiệp (và hệ quả là xã hội) hưởng lợi khổng lồ từ tiền của nhà nước được bỏ ra cho R&D trong lĩnh vực phần mềm. Nơi mà một công ty có thể thấy một kết quả mà không thể bán được, thì công ty khác có thể thấy một cơ hội kinh doanh. Bằng cách này, một mặt, các kết quả của các chương trình nghiên cứu được cực đại hóa, và mặt khác, sự cạnh tranh giữa các công ty mong muốn sử dụng các kết quả của một dự án gia tăng, vì tất cả họ sẽ cạnh tranh trên cơ sở của cùng những chương trình là kết quả từ dự án.

Mô hình này là không mới. Ở một mức độ nào đó thì đây là một mô hình mà đã cho phép Internet phát triển. Nếu các cơ quan hành chính nhà nước đòi hỏi rằng các kết quả của nghiên cứu được triển khai với các nguồn vốn của mình được phân phối ở dạng của PMTD, thì nó có thể sẽ không ngạc nhiên đối với các trường hợp tương tự sẽ xuất hiện, ở những mức độ khác nhau. Hoặc đầu ra của nghiên cứu đó sẽ là kém cỏi hoặc vô dụng (trong trường hợp đó cách thức lựa chọn các dự án cấp vốn cần phải xem xét lại), hoặc sự năng động được tạo ra bởi việc để lại chúng sẵn sàng cho bất kỳ công ty nào để có khả năng chuyển chúng thành một sản phẩm có thể cho phép những phát triển đơn giản không thể nào biết trước được.

6.3 Những ví dụ của những sáng kiến về pháp lý

Trong phần tiếp sau chúng ta xem một số sáng kiến về pháp lý đặc thù có liên quan tới sử dụng và khuyến khích PMTD của các cơ quan hành chính nhà nước. Tất nhiên, danh sách chúng ta đưa ra không có ý sẽ là toàn diện, và đã tập trung vào những sáng kiến mà chúng từng là tiên phong theo một số cách thức (ngay cả nếu chúng cuối cùng từng không được phê chuẩn). Những bạn đọc có quan tâm hoàn toàn có thể hoàn chỉnh nó bằng việc tra cứu "GrULIC. Pháp lý về sử dụng PMTD của Nhà nước" [133], mà nó trích dẫn nhiều trường hợp tương tự hơn. Hơn nữa, trong phụ lục (phụ lục D) chúng ta đưa vào những mục đích có tính minh họa toàn bộ văn bản hoặc hầu hết các phần thích ứng của một vài sáng kiến này.

6.3.1 Các dự luật tại Pháp

Vào năm 1999 và 2000 tại Pháp 2 dự luật có liên quan tới PMTD đã được trình bày, mà chúng là những người tiên phong trong một loạt dài các tranh luận pháp lý về vấn đề này:

- Dự luật của 1999-495, được đệ trình bởi Lafitte, Tresgouet và Cabanel, đã sẵn sàng trên máy chủ web của Thượng viện nước Cộng hòa Pháp vào tháng 10/1999. Sau một quá trình tranh luận trên Internet (http://www.senat.fr/consult/loglibre/index.htm) [102], mà nó đã kéo dài 2 tháng, bản phác thảo này đã được sửa đổi. Kết quả là dự luật số 2000-117 (Laffitte, Trégouet và Cabanel, Đề xuất của Luật số 117, Thượng viện của nước Cộng hòa Pháp, 2000) [162], mà nó đã viện lý bắt buộc sử dụng PMTD đối với các cơ quan hành chính nhà nước, dự kiến những ngoại lệ và các cách thức chuyển dịch đối với các trường hợp nơi mà nó không thể đáp ứng về

mặt kỹ thuật, trong một ngữ cảnh chung hơn về việc mở rộng sử dụng Internet và PMTD trong khắp các cơ quan hành chính nhà nước của Pháp.

- Vào năm 2000, các nghị sĩ quốc hội Jean-Yves Le Déaut, Christian Paul và Pierre Cohen đã đệ trình một luật mới mà mục đích của nó là tương tự như dự thảo của Laffitte, Trégouet và Cabanel: để tăng cường sự tự do và an ninh của người sử dụng, bổ sung thêm vào việc cải thiện sự bình quyền trong xã hội thông tin.

Tuy nhiên, không giống như dự luật của Laffitte, Trégouet và Cabanel, dự luật thứ 2 này đã không bắt buộc sử dụng PMTD đối với các cơ quan hành chính nhà nước. dự luật này đã tập trung vào thực tế rằng phần mềm được sử dụng bởi các cơ quan hành chính nhà nước phải có mã nguồn sẵn sàng, nhưng không ép buộc nó phải được phân phối với các giấy phép của PMTD.

Để đạt được những mục tiêu của họ, những người làm luật đã hướng tới việc đảm bảo "quyền cho tính tương thích" của phần mềm, bằng việc đưa ra các cơ chế mà đưa vào thực tế nguyên tắc của tính tương hợp được phản ánh trong Chỉ thị của Ủy ban châu Âu (EC) có liên quan tới bảo vệ pháp luật về các chương trình máy tính (Chỉ thị của Ủy ban 91/250/EEC, ngày 14/05/1991, về bảo vệ pháp lý các chương trình máy tính, 1991) [111].

Cả 2 phác thảo của Pháp đã không được phê chuẩn thành luật, nhưng cả 2 đã phục vụ để truyền cảm hứng cho hầu hết các sáng kiến sau này trên toàn thế giới, mà nó giải thích vì sao chúng đặc biệt thú vị để nghiên cứu.

Dự thảo thứ 2 (được đệ trình bởi Le Déaut, Paul và Cohen) đã theo đuổi tính tương thích và tương hợp của phần mềm, nhấn mạnh tính sẵn sàng của mã nguồn cho các phần mềm được sử dụng bởi các cơ quan hành chính nhà nước. Tuy nhiên, nó đã không yêu cầu các ứng dụng được phát triển phải là PMTD, được hiểu theo nghĩa là phần mềm được phân phối theo các giấy phép mà chúng đảm bảo sự tự do để sửa đổi, sử dụng và phân phối lại chương trình.

Sau này (trong phần D.1 và phần D.2 của phụ lục D) chúng ta xem lại hầu như toàn bộ các bài viết và các bản ghi nhớ có giải thích của cả 2 dự luật này. Các bản ghi nhớ có giải thích này là đặc biệt thú vị, vì chúng nhấn mạnh những vấn đề hiện đang đe dọa các cơ quan hành chính nhà nước về sử dụng phần mềm nói chung.

6.3.2 Dự luật của Brazil

Vào năm 1999, nghị sỹ quốc hội Walter Pinheiro đã đệ trình một dự luật về PMTD cho Phòng Ủy quyền Liên bang Brazil (Proposição pl-2269/1999. Dispõe sobre a utilização de programas abertos pelos entes de direito público e de direito privado sob controle acionário da administração pública, Phòng Ủy quyền của Brazil, tháng 12/1999) [185]. Dự án này đã quan tâm tới việc sử dụng PMTD trong cơ quan hành chính nhà nước và trong các công ty tư nhân với Nhà nước như là người nắm giữ cổ phần đa số.

Nó khuyến cáo sử dụng PMTD đối với các cơ quan này mà không có những hạn chế về việc cho vay mượn, sửa đổi hoặc phân phối. Các bài viết về luật mô tả chi tiết cách mà PMTD được định nghĩa và

làm thế nào mà các giấy phép đi với nó phải là. Các định nghĩa trùng khớp với định nghĩa kinh điển của PMTD của dự án GNU. Bản ghi nhớ có giải thích xem xét lại lịch sử của dự án GNU, phân tích những ưu điểm và thành tựu của nó. Nó cũng tham chiếu tới hiện trạng của PMTD, sử dụng hệ điều hành GNU/Linux như một ví dụ.

Một trong những phần thú vị nhất của bài viết, xác định rất rõ ràng lĩnh vực mà ở đó việc sử dụng PMTD được đề xuất (nhớ trong đầu rằng định nghĩa được đưa ra trong các bài viết sau cho "chương trình mở" là, như đã được nhắc tới, y hệt như PMTD):

> "Cơ quan hành chính nhà nước ở tất cả các cấp, sức mạnh của nước Cộng hòa, Nhà nước và các doanh nghiệp công tư hợp doanh, các công ty nhà nước và tất cả các cơ quan khác của nhà nước tuân thủ sự kiểm soát của Nhà nước Brazil có bổn phận phải sử dụng một cách ưu tiên, trong các hệ thống và trang bị máy tính của họ, các chương trình mở, tự do khỏi những hạn chế sở hữu độc quyền với quan tâm về sự nhượng lại, sửa đổi và phân phối".

6.3.3 Các dự luật tại Peru

Tại Peru, một vài dự án dự thảo đã được đệ trình về sử dụng PMTD của các cơ quan hành chính nhà nước ("GNU Peru. Các dự luật về PMTD trong hành chính nhà nước của chính phủ Peru", Quốc hội của nước Cộng hòa) [184]. Cái đầu tiên và nổi tiếng nhất đã được đệ trình bởi nghị sỹ quốc hội Edgar Villanueva Núñez vào tháng 12/2001 (dự luật về PMTD số 1609, tháng 12/2001) [222]. Nó định nghĩa *PMTD* theo định nghĩa kinh điển của 4 quyền tự do (có lẽ bổ sung chính xác hơn về pháp lý, với một định nghĩa chỉ định 6 đặc tính của một chương trình tự do) và đề xuất chỉ sử dụng nó trong các cơ quan hành chính nhà nước của Peru.

> "Điều 2. Các cơ quan hành pháp, lập pháp và tư pháp, các cơ quan phân quyền địa phương và các công ty nơi mà Nhà nước là người chiếm cổ phần đa số, sẽ chỉ sử dụng các chương trình hoặc PMTD trong các hệ thống và thiết bị máy tính của họ".

Bất chấp, sau đó, các điểm 4 và 5 đưa vào một số ngoại lệ cho qui định này.

Trong những ngày đó dự luật này đã có một sự tác động toàn cầu. Một mặt, nó là lần đầu tiên mà chỉ sử dụng PMTD trong các cơ quan hành chính nhà nước đã được đề xuất. Nhưng còn quan trọng hơn cho tác động này về tính mới lạ của nó, là sự trao đổi thư từ giữa nghị sỹ Villanueva và đại diện của Microsoft tại Peru, mà nó đã viện các lý do chống lại đề xuất này. Dự luật này cũng thú vị trong mối liên quan tới quan điểm được chấp nhận của sứ quán Mỹ, mà đã gửi qua các kênh chính thức một lưu ý (gắn vào một báo cáo được Microsoft chuẩn bị) cho Quốc hội Peru bày tỏ "sự lo lắng về những đề xuất gần đây của Quốc hội nước Cộng hòa về mua sắm của Chính phủ Peru chỉ giới hạn trong PMNM hoặc PMTD" ("Bức thư cho chủ tịch Quốc hội nước Cộng hòa này", 2002 [147]. Trong số những vận động khác, những viện lý của cả Microsoft và Sứ quán Mỹ đã cố chứng minh rằng dự luật có thể phân biệt đối xử giữa các công ty khác nhau làm cho những đầu tư được yêu cầu sẽ không thể tạo ra được một nền công nghiệp phần mềm của quốc gia. Villanueva đã tranh luận ngược lại rằng dự luật không phân biệt hay ưu tiên bất kỳ công ty đặc biệt nào bằng bất kỳ cách thức nào, vì nó đã không chỉ định ai có thể là nhà cung cấp cho các cơ quan hành chính nhà nước, mà là cách nào (theo những điều kiện gì) phần mềm có thể phải được cung cấp. Lý lẽ này là rất rõ ràng để hiểu làm thế nào sự khuyến khích về

PMTD của các cơ quan hành chính nhà nước không gây hại bằng bất kỳ cách nào cho sự cạnh tranh tự do giữa các nhà cung cấp.

Sau đó, các nghị sỹ quốc hội Peru là Edgar Villanueva Núñez và Jacques Rodrich Ackerman đã đệ trình một dự luật mới, số 2485, ngày 08/04/2002 (Dự luật về Sử dụng PMTD trong các cơ quan hành chính nhà nước số 2485, 2002) [223], mà trong tháng 08/2003 vẫn còn trong những bản lưu của quốc hội. Dự luật này đã là một sự tiến bộ của Dự luật 1609 [222], từ đó nó đưa ra vài bình luận thực hiện một số cải tiến, và có thể được xem là một ví dụ tốt về một dự luật đề xuất sử dụng chỉ các PMTD trong các cơ quan hành chính nhà nước, bảo lưu một số trường hợp ngoại lệ. Đưa ra sự xác đáng của nó, chúng ta đưa vào văn bản đầy đủ của nó (phần D.3 của phụ lục D). Đặc biệt, bản ghi nhớ có giải thích là một kết luận tốt về những đặc tính mà phần mềm được sử dụng trong các cơ quan hành chính nhà nước phải có và làm cách nào PMTD tuân theo được với những đặc tính tốt hơn của PMSHĐQ.

6.3.4 Các dự luật tại Tây Ban Nha

Tại Tây Ban Nha đã từng có một vài sáng kiến pháp lý có liên quan tới PMTD. Dưới đây, chúng ta trích ra một số trong đó:

- Sắc lệnh về các Biện pháp để Khuyến khích Xã hội Tri thức tại Andalucia. Một trong những sáng kiến pháp lý quan trọng nhất tại Tây Ban Nha (vì nó đã trở thành có hiệu lực) từng chắc chắn không nghi ngờ gì là một sáng kiến pháp lý được áp dụng bởi Andalucia. Sắc lệnh về các Biện pháp để Khuyến khích Xã hội Tri thức tại Andalucia (Sắc lệnh 72/2003, ngày 18/03 về các Biện pháp để Khuyến khích Xã hội Tri thức tại Andalucia, 2003) [99] là về việc sử dụng PMTD, về cơ bản (nhưng không chỉ) trong ngữ cảnh giáo dục.

- Ngoài ra, nó khuyến khích sử dụng ưu tiên PMTD trong các trung tâm giáo dục nhà nước, bắt buộc tất cả các trang bị được mua bởi các trung tâm này phải tương thích được với các hệ điều hành tự do, và cũng vậy cho các trung tâm của chính quyền vùng mà cung cấp truy cập Internet công cộng.

- Dự luật về PMTD trong ngữ cảnh của các cơ quan hành chính nhà nước của Catalonia. Các cộng đồng khác đã tranh luận về những đề xuất tham vọng hơn, nhưng không giành được đa số phiếu mà chúng đòi hỏi. Đề xuất nổi tiếng nhất của chúng có lẽ là đề xuất được tranh luận tại Quốc hội của Catalonia (Proposició de llei de programari lliure en el marc de l'Administració pública de Catalunya, 2002) [221], rất giống với đề xuất mà cũng đảng này (Esquerra Republicana de Catalunya) đã đệ trình cho Hạ viện, mà chúng ta sẽ nói về nó sau. Đề xuất này đã không thành công khi được đệ trình để biểu quyết.

- Dự luật của Puigcercós Boixassa trong Hạ viện. Cũng còn một sáng kiến nữa trong Hạ viện được đệ trình bởi Joan Puigcercós Boixassa (Esquerra Republicana de Catalunya) (Dự luật về các Biện pháp cho sự thâm nhập của PMTD vào các cơ quan hành chính nhà nước, 2002) [188]. Sáng kiến này đã đệ trình sử dụng ưu tiên PMTD trong các cơ quan hành chính nhà nước, và theo ý nghĩa này là tương tự như các sáng kiến khác mà chia sẻ mục tiêu này. Tuy nhiên, nó có đặc thù thú vị về việc nhấn mạnh tính sẵn sàng của các chương trình tự do được bản địa hóa cho

các ngôn ngữ đồng chính thức (tại các cộng đồng tự trị mà có chúng). Sáng kiến này đã không được phê chuẩn trong chương trình nghị sự của quốc hội.

7 Kỹ thuật của PMTD

> "Cách tốt nhất để có được một ý tưởng tốt là phải có chúng nhiều".
>
> Linus Pauling.

Trong các chương trước chúng ta đã chỉ ra vì sao thách thức của PMTD không phải là một thách thức của một đối thủ cạnh tranh mà tạo ra phần mềm nhanh hơn, rẻ hơn và chất lượng tốt hơn: PMTD là khác so với phần mềm "truyền thống" ở nhiều hơn các khía cạnh cơ bản, bắt đầu với những lý do triết học và những động lực, tiếp tục với thị trường và các mô hình kinh doanh mới, và kết thúc với một cách khác biệt về việc tạo ra phần mềm. Kỹ thuật phần mềm không thể không bị ảnh hưởng bởi tất cả các yếu tố đã được nói tới ở trên; vì thế, đối với sự nghiên cứu ít hơn 10 năm về cách mà PMTD được phát triển đã từng là tiêu điểm sâu sắc lớn hơn nhiều. Chương này hướng tới việc tranh luận những nghiên cứu và bằng chứng đáng kể nhất mà chúng cung cấp, với quan điểm đưa ra cho độc giả một tầm nhìn hiện đại và viễn cảnh tương lai của những gì chúng ta đã quyết định để gọi là *kỹ thuật của PMTD*.

7.1 Giới thiệu

Dù PMTD đã từng được phát triển vài thập kỷ tới nay, chỉ trong những năm gần đây chúng ta mới bắt đầu chú ý tới các mô hình và các qui trình phát triển của nó từ một viễn cảnh của kỹ thuật phần mềm. Theo cách y hệt như vậy khi mà không chỉ có một mô hình duy nhất cho sự phát triển của PMSHĐQ, cũng không có mô hình duy nhất cho sự phát triển PMTD[5], mà ngay cả như vậy chúng ta cũng có thể thấy những đặc tính thú vị mà hầu hết các dự án theo sự chia sẻ nghiên cứu và điều đó có thể bắt nguồn từ những thuộc tính của các chương trình tự do.

Vào năm 1997, Eric S. Raymond đã xuất bản bài viết được phát tán rộng rãi lần đầu tiên "*Nhà thờ lớn và cái chợ*". *Trầm ngâm về Linux và nguồn mở bởi một cuộc cách mạng ngẫu nhiên*, O'Reilly và Associates http://www.ora.com, 2001) [192], mô tả một số đặc tính của mô hình phát triển PMTD, đặc biệt nhấn mạnh tới những gì các mô hình này khác biệt với sự phát triển của PMSHĐQ. Kể từ đó bài viết này đã trở thành một trong những bài viết nổi tiếng (và bị chỉ trích) nhất trong thế giới PMTD, và tới thời điểm, là dấu hiệu của việc bắt đầu sự phát triển của kỹ thuật PMTD.

7.2 Nhà thờ lớn và cái chợ

Raymond so sánh sự tương tự giữa cách xây dựng các nhà thờ lớn thời trung cổ và cách thức kinh điển về sản xuất phần mềm. Viện lý rằng trong cả 2 trường hợp có một sự phân phối rõ ràng các nhiệm vụ và chức năng, nhấn mạnh sự tồn tại của một nhà thiết kế mà nhìn trước được mọi thứ và phải luôn kiểm soát được sự phát triển của hoạt động này. Cùng lúc, việc lên kế hoạch được kiểm soát nghiêm ngặt, đưa ra những qui trình được chi tiết hóa ở những nơi mà lý tưởng thì mỗi người tham gia trong hoạt

5 Bài viết "Sinh thái học của sự phát triển PMNM" (Kieran Healy and Alan Schussman, 2003) [140] chỉ ra một loạt lớn các dự án và sự đa dạng về số lượng các lập trình viên, sử dụng các công cụ và những bản tải về.

động này có một vai trò được xác định rõ.

Những gì Raymond lấy làm như mô hình cho việc xây dựng các nhà thờ lớn không chỉ có chỗ cho các qui trình lớn mà chúng ta có thể thấy trong nền công nghiệp phần mềm (mô hình thác nước đổ kinh điển, những khía cạnh khác biệt của Qui trình Thống nhất Hợp lý – RUP, ...), mà còn cho các dự án PMTD như GNU và NetBSD. Đối với Raymond, các dự án này được tập trung cao độ, vì chỉ một số ít người có trách nhiệm cho thiết kế và triển khai phần mềm. Các nhiệm vụ được triển khai bởi những người này, bổ sung thêm vào các chức năng của chúng, được xác định tốt, và bất kỳ ai mong muốn hình thành một phần của đội phát triển cần phải được chỉ định một nhiệm vụ và một chức năng theo những yêu cầu của dự án. Mặt khác, sự tung ra dạng các chương trình này theo đúng thời gian đặt ra theo một lịch trình khá nghiêm ngặt. Điều này có ý nghĩa là có ít các lần tung ra phần mềm và chu kỳ dài, tạo nên vài bước giữa các lần tung ra đó.

Mô hình đối nghịch lại đối với nhà thờ lớn là của cái chợ. Theo Raymond, một số chương trình PMTD, đặc biệt là nhân Linux, đã được phát triển theo một sơ đồ tương tự như việc của một cái chợ phương đông. Trong một cái chợ không có nhà chức trách lớn nhất để kiểm soát các qui trình mà sẽ được phát triển hoặc lên kế hoạch một cách nghiêm ngặt những gì phải xảy ra. Cùng một lúc, các vai trò của những người tham gia có thể thay đổi liên tục (những người bán có thể trở thành những khách hàng) và không có chỉ định ra bên ngoài.

Nhưng những gì là mới lạ nhất về "Nhà thờ lớn và cái chợ" là làm thế nào nó mô tả được quá trình để Linux đã trở nên thành công; đây là một danh sách các "kinh nghiệm thực tiễn tốt" để tạo ra nhiều cơ hội nhất được đưa ra bởi mã nguồn có sẵn sàng, và về tính có thể tương tác được thông qua việc sử dụng các hệ thống và công cụ từ xa.

Một dự án PMTD có xu hướng xuất hiện như một kết quả của một hành động cá nhân thuần túy; Lý do này có thể thấy được trong một lập trình viên mà thấy khả năng của anh ta để giải quyết một vấn đề có giới hạn. Lập trình viên cần phải có đủ sự hiểu biết để bắt đầu, ít nhất là, giải quyết nó. Một khi anh ta đã giành được thứ gì đó có thể sử dụng được, với một số chức năng, đơn giản, và nếu có thể, được thiết kế hoặc được viết tốt, thì điều tốt nhất mà anh ta có thể làm là chia sẻ giải pháp đó với thế giới PMTD. Đây là những gì được biết tới như là sự tung ra sớm, mà nó giúp lôi cuốn được sự chú ý của những người khác (thường là các lập trình viên), những người có vấn đề y hệt và những người có thể có quan tâm trong giải pháp đó. Một trong những nguyên tắc của mô hình phát triển này là nghĩ về những người sử dụng như những người cùng phát triển - cùng là các lập trình viên. Họ cần phải được đối xử với sự cẩn trọng, không chỉ vì họ có thể cung cấp "sự truyền khẩu" một cách công khai mà còn vì họ sẽ triển khai một trong những nhiệm vụ tốn tiền nhất mà có trong sự tạo ra phần mềm: việc kiểm thử. Không giống như sự đồng phát triển, mà là khó về phạm vi, việc dò tìm lỗi và các vụ kiểm thử có tính chất song song hóa được cao độ. Người sử dụng sẽ là một tính chất để nắm lấy phần mềm và kiểm thử nó trong máy của anh ta theo những điều kiện cụ thể (một kiến trúc, các công cụ cụ thể nào đó, …), một nhiệm vụ mà nó được nhân lên một số lượng rộng rãi của các kiến trúc và các môi trường có thể gây ra một hiệu ứng khổng lồ đối với đội phát triển.

Nếu chúng ta đối xử với người sử dụng như là đồng lập trình viên thì có thể xảy ra rằng một trong số họ thấy một lỗi và giải quyết nó bằng việc gửi một bản vá tới các lập trình viên của dự án sao cho vấn đề đó có thể được giải quyết trong phiên bản tiếp sau. Nó cũng có thể xảy ra, ví dụ, rằng ai đó khác

người mà phát hiện ra lỗi đó ngẫu nhiên hiểu nó và sửa nó. Trong trường hợp này, tất cả những hoàn cảnh là có lợi cho sự phát triển của PMTD, nghĩa là nó có lợi để tham gia vào một sự năng động của dạng này.

Tình huống này trở nên có hiệu quả hơn với những lần tung ra thường xuyên, vì động lực để tìm ra, lưu ý và sửa các lỗi là cao vì được giả thiết rằng chúng sẽ được tham dự ngay lập tức. Hơn nữa, những lợi ích thứ cấp sẽ đạt được như thực tế là sự tích hợp thường xuyên - lý tưởng một hoặc nhiều lần trong ngày - không đòi hỏi một pha cuối cùng về việc tích hợp các module cấu tạo nên chương trình. Điều này được gọi là sự tung ra thường xuyên và cho phép một sự module hóa cao (Alessandro Narduzzo và Alessandro Rossi, "Tính module hóa trong hành động: GNU/Linux và mô hình phát triển mở của PMTD/PMNM", tháng 05/2003) [176], cùng một lúc khi nó tối đa hóa hiệu ứng tuyên truyền được báo chí thực hiện đối với phiên bản mới nhất của phần mềm.

Lưu ý

Sự quản lý phiên bản mới phụ thuộc, một cách logic, vào kích thước của dự án, vì những vấn đề mà cần phải được giải quyết sẽ không y hệt khi đội phát triển có 2 thành viên và khi có hàng trăm thành viên. Trong khi đó, nói chung, đối với các dự án nhỏ thì qui trình này ít nhiều là không chính thống, thì sự quản lý các bản tung ra cho những dự án lớn có xu hướng tuân theo một qui trình được xác định, mà nó không là ngoại lệ từ một mức độ nào đó về tính phức tạp. Có một bài viết gọi là “Quản lý phiên bản trong các dự án nguồn mở” (Justin R. Ehrenkrantz, 2003) [110] mà nó mô tả chi tiết sự tuần tự được tuân thủ với máy chủ web Apache, nhân Linux và hệ thống thiết lập phiên bản phụ.

Để tránh “ra phiên bản thường xuyên” từ việc làm sợ cho những người sử dụng với một ưu tiên cho tính ổn định của phần mềm đối với tốc độ mà với nó phần mềm tiến hóa, một số dự án PMTD có vài nhánh phát triển chạy song song. Trường hợp nổi tiếng nhất của điều này là nhân Linux, mà về lịch sử đã có hướng tới những người mà họ đánh giá độ tin cậy của nó và những thứ không ổn định khác được thiết kế cho các lập trình viên với những đổi mới sáng tạo và những sự lạ thường mới nhất.

7.3 Sự lãnh đạo và ra quyết định trong cái chợ

Raymond cho rằng tất cả các dự án PMTD phải có một nhà độc tài nhân từ, một dạng lãnh tụ mà thường là người sáng lập ra dự án để chỉ dẫn dự án và luôn có lời cuối cùng khi liên quan tới ra quyết định. Những kỹ năng mà người này phải có liên quan chủ yếu tới việc biết cách để động viên và điều phối một dự án, hiểu những người sử dụng và các đồng lập trình viên, tìm kiếm sự đồng thuận và việc tích hợp từng người mà có thứ gì đó để đóng góp. Như bạn có thể thấy, chúng ta đã không nhắc tới năng lực kỹ thuật trong số những yêu cầu quan trọng nhất này, dù nó không bao giờ là thừa.

Khi mà kích thước dự án và số lượng các lập trình viên có liên quan tới chúng tăng lên, các cách thức mới của việc tổ chức ra quyết định đã nổi lên. Ví dụ, Linux, có một cấu trúc thứ bậc dựa vào những trách nhiệm giao phó cho Linus Torvalds, “nhà độc tài nhân từ”. Và, chúng ta sẽ thấy rằng có những phần của Linux mà có “các nhà độc tài nhân từ” của riêng chúng, dù sức mạnh của họ sẽ bị hạn chế bởi thực tế là Linus Torvalds có lời nói cuối cùng. Trường hợp này là một ví dụ rõ ràng về cách mà một mức độ cao của tính module hóa trong một dự án PMTD đã tạo ra đối với một cách thức cụ thể về việc tổ chức mọi thứ và ra các quyết định (Alessandro Narduzzo và Alessandro Rossi, "Tính module hóa

trong hành động: Mô hình mở của sự phát triển GNU/Linux và PMTD/PMNM", 2003) [176].

Lưu ý

Một số người nói rằng cách mà các dự án PMTD được tổ chức là tương tự như một đội phẫu thuật, như được đề xuất bởi Harlan Mills (của IBM) trong những năm đầu thập kỷ 70 được phổ biến bởi Brooks trong cuốn sách nổi tiếng của ông "*Người - tháng thần thoại*" (Frederick P. Brooks Jr., 1975) [150].

Dù có thể có những trường hợp nơi mà đội phát triển của một ứng dụng PMTD cụ thể nào đó tạo thành từ một người thiết kế/người lập trình (nhà phẫu thuật) và nhiều đồng lập trình viên mà họ thực hiện những nhiệm vụ phụ trợ (quản trị hệ thống, duy trì, các nhiệm vụ đặc thù, viết tài liệu), sẽ không bao giờ có một sự chính xác như vậy và sự chia rẽ được xác định khi một người được gợi ý bởi Mills và Brooks. Tất tần tật, như Brooks chỉ ra trong trường hợp của đội phẫu thuật, trong PMTD số lượng các lập trình viên mà cần giao tiếp để tạo ra được một hệ thống lớn và phức tạp - những người tích cực nhất - là thấp hơn nhiều so với tổng số lượng các lập trình viên.

Trong trường hợp của Quỹ Apache, chúng ta có một *chế độ nhân tài*, khi mà cơ quan này có một ủy ban các giám đốc hình thành từ những người mà đã đóng góp vào theo một cách đáng chú ý cho dự án. Trong thực tế, không phải một chế độ nhân tài nghiêm khắc theo nghĩa của những người nào mà đóng góp nhiều nhất sẽ điều hành, vì ủy ban các giám đốc được bầu một cách dân chủ và thường bởi các thành viên của Quỹ (có trách nhiệm cho việc quản lý một loạt các dự án PMTD, như Apache, Jakarta, …). Để trở thành một thành viên của Quỹ Apache, bạn cần phải đóng góp theo một cách quan trọng và liên tục cho một trong vài dự án của Quỹ. Hệ thống này cũng được sử dụng bởi các dự án khác, như FreeBSD hoặc GNOME.

Trường hợp thú vị khác về tổ chức chính thống là Ban Chỉ đạo GCC. Nó đã được tạo ra vào năm 1998 để tránh bất kỳ ai giành được sự kiểm soát đối với dự án GCC (Bộ sưu tập các trình biên dịch của GNU, hệ thống biên dịch của GNU) và được sự ủng hộ của FSF (người sáng lập ra dự án GNU) ít tháng sau đó. Theo một ý nghĩa nhất định nào đó, ban này tiếp tục truyền thống của một ban tương tự mà dự án EGCS đã có (mà cùng một lúc quản lý song song cho dự án GCC, nhưng sau đó đã ra nhập nó). Nhiệm vụ cơ bản của nó là để đảm bảo rằng dự án GCC đáp ứng đầy đủ được tuyên bố về nhiệm vụ của dự án này. Các thành viên của ban này là các thành viên theo một khả năng cá nhân, và được chọn bởi bản thân dự án theo một cách như để đại diện một cách trung thực cho các cộng đồng khác nhau mà hợp tác trong sự phát triển của GCC (hỗ trợ các lập trình viên đối với vài ngôn ngữ lập trình, các lập trình viên có liên quan tới nhân, các nhóm có quan tâm trong việc lập trình nhúng, …).

Cũng con người ấy không nhất thiết phải là lãnh đạo của một dự án PMTD vĩnh viễn. Về cơ bản, có thể có 2 hoàn cảnh trong đó lãnh đạo của dự án thôi là như vậy. Trường hợp đầu là thiếu sự quan tâm, thời gian hoặc động lực để tiếp tục. Trong trường hợp này, gậy chỉ huy phải được truyền cho lập trình viên khác mà sẽ đóng vai trò của lãnh đạo dự án. Các nghiên cứu gần đây (Jesús M. González Barahona và Gregorio Robles, 2003) [87] chỉ ra rằng, nói chung, sự lãnh đạo dự án thường thay đổi người, theo một cách mà chúng ta có thể thấy vài thế hệ các lập trình viên qua một thời gian. Trường hợp thứ 2 là có vấn đề hơn: nó liên quan tới sự rẽ nhánh. Các giấy phép PMTD cho phép mã nguồn được nắm lấy, được sửa và được phận phối lại bởi bất kỳ ai mà không cần có sự phê chuẩn của người lãnh đạo dự án. Điều này có xu hướng xảy ra một cách không bình thường, ngoại trừ trong các trường hợp nơi mà ý tưởng này là để tránh người lãnh đạo dự án một cách có chủ tâm (và sự phủ quyết tiềm năng của người lãnh đạo chống lại một sự đóng góp). Điều này là tương tự một mặt đối với một dạng "đảo chính", mà

một mặt khác là hoàn toàn đúng và hợp pháp. Đối với lý do này, một trong những mục đích của một nhà lãnh đạo dự án trong việc giữ cho các đồng lập trình viên được thỏa mãn là giảm tối đa khả năng rẽ nhánh.

7.4 Các qui trình của PMTD

Dù PMTD không nhất thiết có liên quan tới một qui trình phát triển phần mềm đặc biệt nào, thì vẫn có một sự đồng thuận rộng rãi về những qui trình mà nó sử dụng một cách rộng rãi nhất. Điều này không có nghĩa là không có dự án PMTD nào từng được tạo ra có sử dụng các qui trình kinh điển, như mô hình thác nước đổ. Nói chung, mô hình phát triển của các dự án PMTD có xu hướng sẽ rất không chính thức, chủ yếu vì thực tế là một sự chia sẻ rộng rãi của đội phát triển thực hiện các nhiệm vụ này một cách tự nguyện và trong sự trao đổi không vì phần thưởng về tài chính, ít nhất theo một cách trực tiếp. Cách thức để nắm bắt những yêu cầu trong thế giới PMTD phụ thuộc nhiều vào "tuổi" cũng như kích cỡ của dự án. Trong những giai đoạn đầu, người sáng lập và người sử dụng của dự án có xu hướng là cùng chính những người đó. Sau đó, và nếu dự án mở rộng, thì sự nắm bắt được các yêu cầu có xu hướng diễn ra thông qua các danh sách thư điện tử và một sự phân biệt có xu hướng sẽ đạt được giữa đội phát triển, hoặc ít nhất, những lập trình viên và những người sử dụng tích cực hơn. Đối với những dự án lớn, với nhiều người sử dụng và nhiều lập trình viên, thì những yêu cầu sẽ được nắm bắt có sử dụng cùng một công cụ như là công cụ được sử dụng cho việc quản lý các lỗi của dự án. Trong trường hợp này, thay vì làm việc với các lỗi, họ tham chiếu tới các hoạt động, dù cơ chế được sử dụng cho việc quản lý chúng là y hệt cơ chế cho việc dò tìm và sửa lỗi (chúng sẽ được phân loại về tầm quan trọng, sự phụ thuộc, …, và nó sẽ có khả năng để giám sát liệu chúng đã được giải quyết hay chưa). Việc sử dụng công cụ lập kế hoạch này là khá gần đây, nên chúng ta có thể thấy cách mà thế giới của PMTD đã tiến hóa một chút từ một sự thiếu hụt hoàn toàn, tới một hệ thống tập trung cho việc quản lý các hoạt động này theo nghĩa kỹ thuật, ngay cả nếu nó chắc chắn bị hạn chế hơn. Tất tần tật, không thường thấy một tài liệu nào mà nó tập hợp các yêu cầu, như là trường hợp thông thường trong mô hình thác nước đổ.

Như đối với việc thiết kế toàn cầu các hệ thống, thì chỉ những dự án lớn có xu hướng sẽ được viết thành tài liệu một cách chi tiết tổng thể. Còn với những dự án còn lại, lập trình viên chính (hoặc nhóm của các lập trình viên chính) hầu như có lẽ chỉ có một (nhóm) người duy nhất có nó, trong đầu họ; đôi khi, điều này còn không xảy ra, và hệ thống hình thành khi phần mềm tiến hóa. Sự thiếu hụt một thiết kế chi tiết không chỉ ngụ ý tới những hạn chế về khả năng sử dụng lại các module, mà còn là một cản trở lớn khi xảy ra việc phải để các lập trình viên mới truy cập vào, khi họ sẽ phải đối mặt với một qui trình đắt giá và học chậm. Việc có một thiết kế chi tiết cũng không phải là rất thường có. Sự thiếu hụt của nó có nghĩa là nhiều cơ hội cho việc sử dụng lại mã nguồn đã bị mất.

Sự cài đặt triển khai là giai đoạn, nơi mà các lập trình viên PMTD tập trung hầu hết mọi nỗ lực, lý do là theo quan điểm của họ thì rõ ràng đây là điều thú vị nhất. Để làm được điều này, mô hình lập trình kinh điển về kiểm thử và lỗi thường được giám sát cho tới khi các kết quả mong muốn đạt được từ quan điểm về mục tiêu của các lập trình viên. Về mặt lịch sử, hiếm khi những kiểm thử sẽ được đưa vào với mã nguồn, ngay cả nếu chúng có thể thực hiện sự sửa đổi hoặc đưa vào mã nguồn tới sau từ các lập trình viên khác là dễ dàng hơn. Trong trường hợp của các dự án thực sự là lớn, ví dụ như Mozilla, thì chúng là các máy chuyên dụng một cách riêng biệt cho việc tải về các kho chứa hầu hết mã nguồn gần

đây nhất và để biên dịch nó cho các kiến trúc khác nhau ("Tổng quan về qui trình và các công cụ của kỹ thuật phần mềm trong dự án Mozilla", 2002) [193]. Các lỗi được dò tìm ra sẽ được lưu ý tới một danh sách thư của các lập trình viên.

Tuy nhiên, các kiểm thử tự động không phải là một thực tế đóng kín. Nói chung, bản thân những người sử dụng, với một dãy lớn khổng lồ của họ về việc sử dụng, các kiến trúc và những kết hợp, sẽ triển khai chúng. Điều này có ưu điểm về quản lý chúng song song với một giá thành tối thiểu đối với đội phát triển. Vấn đề với mô hình này là làm thế nào để giành được ý kiến phản hồi từ những người sử dụng và để tổ chức nó có hiệu quả nhất có thể. Vì sự duy trì phần mềm trong thế giới của PMTD được quan tâm, được hiểu như là sự duy trì của các phiên bản trước, nên việc có được nhiệm vụ này sẽ phụ thuộc vào dự án. Đối với các dự án mà cần tính ổn định, như là các nhân của hệ điều hành, thì các phiên bản trước đó sẽ được duy trì, vì việc thay đổi tới một phiên bản mới có thể sẽ gây lỗi. Nhưng nói chung, đối với hầu hết các dự án của PMTD, nếu một lỗi được tìm thấy trong một phiên bản trước đó, thì các lập trình viên thường sẽ bỏ qua nó và khuyến cáo sử dụng phiên bản mới nhất trong hy vọng rằng lỗi đó đã biến mất với sự tiến hóa của phần mềm.

7.5 Chỉ trích về "Nhà thờ lớn và cái chợ"

"Nhà thờ lớn và cái chợ" chịu sự chỉ trích về tính không có hệ thống và thiếu tính chặt chẽ từ giới phóng viên hơn là từ bản chất tự nhiên của giới khoa học. Chỉ trích thường xuyên nhất tham chiếu tới thực tế là nó về cơ bản giải thích cho trường hợp cụ thể của trải nghiệm của Linux và hướng tới để mở rộng những kết luận này cho tất cả các dự án PMTD. Theo nghĩa này, trong "Hang động hay cộng đồng? Một sự kiểm tra theo kinh nghiệm của 100 dự án nguồn mở chín muồi" [160] chúng ta có thể thấy rằng sự tồn tại của một cộng đồng rộng lớn như cộng đồng của nhân Linux là một ngoại lệ hơn là một lệ thường.

Còn nhiều chỉ trích hơn là từ những người mà họ tin rằng Linux là một ví dụ của mô hình phát triển của nhà thờ lớn. Họ viện lý rằng rõ ràng có một động lực, hoặc ít nhất là một người với quyền uy tối đa, và một hệ thống thứ bậc có tôn ti trật tự đại diện cho trách nhiệm xuống tới tận những người lao động/lập trình viên. Hơn nữa, có một sự phân công nhiệm vụ, dù là tuyệt đối. "Cái nhìn thứ 2 vào nhà thờ lớn và cái chợ" [91] đi vượt ra khỏi sự kiêng nể, ở một mức độ nào đó có sự cay đắng và ngạo mạn trong việc viện lý của mình, rằng phép ẩn dụ của cái chợ là sự trái ngược mang tính nội tại bên trong.

Những điểm bị chỉ trích nhiều nhất khác của "Nhà thờ lớn và cái chợ" là sự khẳng định rằng luật của Brooks, mà nói rằng "việc bổ sung thêm các lập trình viên cho một dự án phần mềm đã bị chậm trễ sẽ làm chậm trễ nó còn nhiều hơn" (*Người - tháng thần thoại. Tiểu luận về kỹ thuật phần mềm*, 1975) [150], là không có giá trị trong thế giới của PMTD. Trong [148] chúng ta có thể đọc được cách mà những gì đã xảy ra trong thực tế là việc những ngữ cảnh về môi trường sẽ là khác và rằng những gì theo nguyên lý dường như sẽ là không phù hợp với luật Brooks, sau một phân tích tổng hợp hơn, chỉ là một ảo vọng.

7.6 Các nghiên cứu có tính định lượng

PMTD làm cho có khả năng để đi sâu hơn vào nghiên cứu mã nguồn và các tham số khác mà chúng can thiệp vào sự phát sinh của nó nhờ việc có được sự truy cập tới nhiều nguồn thông tin công khai. Điều này cho phép các lĩnh vực của kỹ thuật phần mềm truyền thống như là kỹ thuật phần mềm theo chủ nghĩa kinh nghiệm được khuyến khích nhờ sự tồn tại của một số lượng lớn các thông tin mà có thể truy cập được mà không cần thiết phải thâm nhập bừa một cách nặng nề vào sự phát triển của PMTD. Các tác giả bị thuyết phục rằng tầm nhìn này có thể đóng góp lớn lao cho việc phân tích và tổng hợp những hiện tượng có liên quan tới PMTD (và phần mềm nói chung), và rằng nó còn có thể, trong số những khả năng khác, sản sinh ra các mô hình phần mềm có thể dự báo trước được với các ý kiến phản hồi trong thời gian thực.

Ý tưởng đằng sau nó là rất đơn giản: "đưa ra rằng chúng ta có cơ hội để nghiên cứu một số lượng khổng lồ các chương trình PMTD, hãy làm thế". Và để bổ sung vào hiện trạng của một dự án, sự tiến hóa của nó trong quá khứ là công cộng, nghĩa là tất cả các thông tin này, được trích ra, được phân tích và được đóng gói đúng lúc, thì có thể phục vụ như một nền tảng tri thức mà nó cho phép chúng ta đánh giá một tình trạng lành mạnh của dự án, giúp hướng tới việc ra quyết định và đoán trước được những sự phức tạp trong hiện tại và tương lai.

Nghiên cứu lượng hóa đầu tiên về bất kỳ tầm quan trọng nào trong thế giới của PMTD ngược về năm 1998, dù nó đã được xuất bản vào đầu năm 2000 ("Khảo sát về PMTD Orbiten") [127]. Mục đích của nó là để tìm ra trong những khái niệm theo chủ nghĩa kinh nghiệm cho sự tham gia của các lập trình viên trong PMTD. Để làm vậy họ đã xử lý về mặt thống kê những chỉ định quyền tác giả mà các tác giả có xu hướng đặt trong đầu đề các tệp mã nguồn. Các kết quả chỉ ra rằng sự tham gia là ổn định với luật Pareto ("Tiến trình của Kinh tế Chính trị", Lausana, 1896) [182]: 80% mã nguồn tương ứng với hầu như 20% các lập trình viên tích cực nhất, trong khi 80% các lập trình viên còn lại đóng góp 20% tổng số mã nguồn. Nhiều nghiên cứu sau đó đã khẳng định và mở rộng tính đúng đắn của kết quả này theo các cách thức khác nhau về việc tham gia trong sự đóng góp mã nguồn (các danh sách thư, các lưu ý lỗi hoặc ngay cả là số lượng các bản tải về, như chúng ta có thể thấy được trên http://www-mmd.eng.cam.ac.uk/people/fhh10/Sourceforge/Sourceforge%20paper.pdf [145]).

Lưu ý

Thực tế là nhiều khái niệm về kinh tế xuất hiện trong nghiên cứu về kỹ thuật PMTD là một kết quả của sự quan tâm mà một số nhà kinh tế học đã chỉ ra trong việc học và hiểu những động cơ nào mà những người tình nguyện làm ra được các sản phẩm có giá trị cao mà không giành được theo thường lệ một lợi ích trực tiếp để trao đổi. Bài viết nổi tiếng nhất là "Thị trường của các nồi nấu nước: một mô hình kinh tế cho sự buôn bán trong các hàng hóa và dịch vụ tự do trên Internet" [125], mà nó giới thiệu ý tưởng của *nền kinh tế quà biếu* trên Internet. Tại http://www.wikipedia.org/wiki/Pareto [232] chúng ta có thể có được nhiều hơn những chi tiết về nguyên lý của Pareto và sự tổng quát hóa của nó đối với sự phân bổ của Pareto. Đường cong Lorenz (http://www.wikipedia.org/wiki/Lorenz_curve) [231], mà bằng đồ thị chỉ ra sự tham gia của các lập trình viên trong một dự án, cũng là thú vị như hệ số Gini (http://www.wikipedia.org/wiki/Gini_coefficient) [230], được tính trên cơ sở của đường cong Lorenz và nó tạo ra một số lượng chỉ ra tính không bình đẳng của hệ thống.

Công cụ được sử dụng để tiến hành nghiên cứu này đã được xuất bản bởi các tác giả của nó theo một

giấy phép tự do, nghĩa là các kết quả của nó có thể được làm lại và nó có thể được sử dụng để tiến hành các nghiên cứu mới.

Trong một nghiên cứu sau đó, Koch ("Các kết quả từ nghiên cứu kỹ thuật phần mềm trong các dự án phát triển nguồn mở có sử dụng các dữ liệu công khai", 2000) [158] đã đi xa hơn và cũng đã phân tích những tương tác trong một dự án PMTD. Các nguồn thông tin là các danh sách thư và kho của các phiên bản của dự án GNOME. Nhưng khía cạnh thú vị nhất trong nghiên cứu của Koch là phân tích về kinh tế.

Koch tập trung vào việc kiểm tra tính đúng đắn của những dự báo giá thành theo cách kinh điển (các điểm chức năng, mô hình COCOMO …) và chỉ ra những vấn đề có liên quan trong việc áp dụng chúng, dù nó thừa nhận rằng các kết quả có được phải được nắm lấy với sự dè dặt một phần phù hợp với thực tế. Ông đã kết luận rằng PMTD đòi hỏi các mô hình và các phương pháp nghiên cứu của riêng nó, vì những thứ được biết không áp dụng được cho tính tự nhiên cơ bản của nó. Tuy nhiên, rõ ràng có khả năng đạt được nhiều dữ liệu có liên quan tới sự phát triển của PMTD một cách công khai, cho phép chúng ta lạc quan về việc đạt được những mục tiêu này trong tương lai gần. Phân tích của Koch có thể được coi là phân tích định lượng đầy đủ đầu tiên, dù nó chắc chắn thiếu một phương pháp rõ ràng, và *đặc biệt* một số công cụ hiện đại mà chúng có thể làm cho có khả năng để xác minh các kết quả của nó và để nghiên cứu các dự án khác.

Vào năm 2000, Mockus và những người khác đã trình bày nghiên cứu đầu tiên về các dự án PMTD có chứa đựng một mô tả đầy đủ các qui trình phát triển và cấu trúc tổ chức, với cả bằng chứng định tính và định lượng ("Một trường hợp điển hình về sự phát triển của PMNM: máy chủ Apache") [172]. Để làm thế, họ đã sử dụng các báo cáo thay đổi nhật ký (changelog) và lỗi của phần mềm để lượng hóa các khía cạnh tham gia của các lập trình viên, kích cỡ của nhóm cốt lõi, tác giả của mã nguồn, năng suất, mật độ lỗi, và khoảng thời gian giải quyết các vấn đề. Theo một cách thức, nghiên cứu này vẫn là một nghiên cứu kỹ thuật phần mềm kinh điển, bảo vệ thực tế rằng các dữ liệu đã có được hoàn toàn từ sự điều tra bán tự động các dữ liệu mà các dự án đưa ra công khai trên net. Như trong trường hợp của "Các kết quả từ nghiên cứu kỹ thuật phần mềm trong các dự án phát triển nguồn mở có sử dụng các dữ liệu công khai", 2000 [158], bài viết này đã không cung cấp bất kỳ công cụ nào hoặc qui trình tự động nào mà có thể sử dụng lại được trong tương lai bởi các đội nghiên cứu khác.

Trong "Đánh giá kích cỡ của Linux", 2000 [227], và "Hơn là một gigabuck: đánh giá của GNU/Linux"[228] chúng ta thấy một phân tích có định lượng về các dòng mã lệnh và các ngôn ngữ lập trình được sử dụng trong phát tán Red Hat. González Barahona et al. đã lần theo các bước này trong một loạt các bài viết về phát tán Debian (ví dụ "Mổ xẻ 2 phát tán GNU/Linux" [88]). Tất cả những thứ này được cung cấp bởi một công cụ mà nó tính số dòng mã lệnh của chương trình (SLOC, các dòng mã lệnh mà không phải là các dòng trống hoặc các bình chú). Bên cạnh kết quả ngoạn mục theo tổng số dòng mã lệnh), chúng ta có thể thấy số lượng các dòng được phân bổ như thế nào cho mỗi ngôn ngữ lập trình. Có khả năng nghiên cứu sự tiến hóa của các phiên bản khác nhau của Debian qua thời gian đã đưa ra được một số kết quả thú vị [88]. Đáng để lưu ý rằng trong 5 năm gần đây kích thước các gói trung bình trên thực tế đã được giữ nguyên không đổi, nghĩa là xu hướng tự nhiên gia tăng đã được trung lập hóa bằng sự bao gộp cả các gói nhỏ hơn. Cùng một lúc, chúng ta có thể thấy tầm quan trọng như thế nào của ngôn ngữ lập trình C, dù vẫn là chiếm ưu thế trội hơn hẳn, lại đang giảm đi theo thời gian, trong khi các ngôn ngữ script (Python, PHP và Perl) và Java đang trải nghiệm một sự gia tăng

bùng nổ. Các ngôn ngữ được biên dịch "kinh điển" (Pascal, Ada, Module...) rõ ràng đang mất dần. Cuối cùng, những bài viết này bao gồm cả một phần mà nó chỉ ra những kết quả có được nếu chúng ta áp dụng mô hình đánh giá nỗ lực COCOMO kinh điển có từ những năm đầu thập kỷ 80 (*Kinh tế học của Kỹ thuật Phần mềm*, 1981) [93] và thứ được sử dụng bởi PMSHĐQ để đánh giá những nỗ lực, lịch trình và giá thành của dự án.

Dù là những kẻ đến trước, hầu hết các nghiên cứu được trình bày trong phần này khá là hạn chế đối với các dự án theo các phân tích. Phương pháp được sử dụng đã được áp dụng cho dự án được phân tích, một phần bằng tay và một cách ngẫu nhiên một phần tự động hóa có thể được sử dụng nói chung với các dự án PMTD khác. Điều này có nghĩa là nỗ lực được đòi hỏi để nghiên cứu một dự án mới là lớn hơn nhiều, vì phương pháp này cần phải được áp dụng lại và những nhiệm vụ làm bằng tay sẽ phải được lặp lại.

Vì lý do này, những nỗ lực mới gần đây nhất ("nghiên cứu sự tiến hóa của các dự án PMTD có sử dụng các dữ liệu có sẵn một cách công khai", trong : *Kỷ yếu* của Hội thảo lần thứ 3 về Kỹ thuật của PMNM, Hội nghị Quốc tế lần thứ 25 về Kỹ thuật Phần mềm, Portland, Mỹ [196] hoặc "Việc tự động hóa đo đếm của các dự án nguồn mở", 2003 [124]) tập trung vào việc tạo ra một hạ tầng phân tích mà nó tích hợp một số công cụ sao cho qui trình đó có thể được tự động hóa tối đa. Có 2 lý do khá rõ ràng để làm việc này: lý do đầu tiên là việc một khi nhiều thời gian và nỗ lực được đầu tư vào việc tạo ra một công cụ để phân tích một dự án với sự nhấn mạnh đặc biệt vào việc làm cho nó thành chung, thì nỗ lực có liên quan trong việc sử dụng nó cho các dự án PMTD khác là tối thiểu. Lý do thứ 2 là việc phân tích sử dụng một loạt các công cụ mà chúng nghiên cứu các chương trình từ các quan điểm khác nhau và đôi khi bổ sung cho nhau, cùng một lúc không cho phép chúng ta có được một tầm nhìn rộng hơn về dự án. Trên website về Kỹ thuật của PMTD [86] chúng ta có thể tuân theo những sáng kiến này chi tiết hơn.

7.7 Công việc tương lai

Được mô tả ngắn gọn nhưng mãnh liệt lịch sử của việc nghiên cứu kỹ thuật phần mềm trên PMTD, chúng ta có thể nói rằng vẫn còn đang nói về những bước đầu tiên của nó. Nhiều khía cạnh quan trọng vẫn còn đang treo việc thẩm tra phân tích và chi tiết cho tới khi chúng ta có thể thấy được một mô hình mà ít nhất nó một phần giải thích được làm thế nào mà PMTD lại sinh ra được. Những vấn đề sẽ cần phải được nắm bắt trong tương lai gần bao gồm sự phân loại các dự án PMTD, sự tạo ra một phương pháp dựa trên những phân tích có thể dựa vào đó được và việc sử dụng những hiểu biết có được để xây dựng những mô hình mà giúp cho chúng ta hiểu được PMTD phát triển như thế nào cùng một lúc với việc tạo điều kiện thuận lợi cho việc ra quyết định trên cơ sở của kinh nghiệm có được.

Một khía cạnh khác mà phải được xem xét kỹ và điều đó đang bắt đầu được xem xét hiện nay là tính đúng đắn của các phương pháp về kỹ thuật kinh điển trong lĩnh vực của PMTD trong khắp tất cả những điều nổi lên của kỹ thuật phần mềm. Do đó, ví dụ, các luật của sự tiến hóa của phần mềm được tuyên bố bởi Hehman ("Những giá trị và luật của sự tiến hóa của phần mềm - quan điểm những năm 90" [165]) ở đầu những năm 1970 và được cập nhật và mở rộng trong những năm 80 và 90 dường như không được thỏa mãn vô điều kiện trong sự phát triển của một số dự án PMTD ("Hiểu về sự tiến hóa của PMNM: việc áp dụng, phá vỡ và nghĩ lại các luật về sự tiến hóa của phần mềm", 2003 [199]).

Hiện nay, một trong những sự thiếu hụt nghiêm trọng nhất là sự thiếu hụt một sự phân loại nghiêm ngặt sao cho các dự án PMTD có thể được phân loại trong những chủng loại khác nhau. Hiện tại, các tiêu chí phân loại là quá rộng, và các dự án với những đặc tính rất khác biệt về tổ chức, kỹ thuật và những đặc tính khác tất cả đặt trong cùng một giỏ. Lý lẽ rằng Linux, với một cộng đồng rộng lớn và số lượng lớn các lập trình viên, có một sự tự nhiên khác biệt và không hành xử theo cùng một cách như với một dự án bị hạn chế hơn nhiều về số lượng các lập trình viên và người sử dụng, là rất đúng. Tất tần tật, một sự phân loại chi tiết hơn có thể làm cho có khả năng để sử dụng lại kinh nghiệm có được trong những dự án tương tự (nói cách khác, với những đặc tính tương tự), làm dễ dàng hơn cho các dự báo, và làm cho có khả năng thấy trước được các rủi ro, …

Khía cạnh quan trọng thứ 2 là kỹ thuật PMTD cần phải ràng buộc, kết nối chặt chẽ với điểm trước đó và các xu thế hiện hành, là sự sáng tạo ra một phương pháp và các công cụ để hỗ trợ nó. Một phương pháp rõ ràng và súc tích sẽ làm cho có khả năng để nghiên cứu tất cả các dự án trên một nền tảng công bằng như nhau, phát hiện ra hiện trạng của chúng, học cách mà chúng đã tiến hóa, và tất nhiên, phân loại chúng. Các công cụ là cần thiết khi làm việc với vấn đề này, vì một khi được tạo ra thì chúng làm cho có khả năng để phân tích hàng ngàn dự án với nỗ lực bổ sung là tối thiểu. Một trong những mục tiêu của kỹ thuật PMTD là để làm cho có khả năng nghiên cứu một dự án thật sâu trên cơ sở của một tập hợp có hạn chế các tham số chỉ ra nơi mà thông tin trong dự án có thể thấy được trên Net (địa chỉ của kho các phiên bản phần mềm, nơi mà ở đó các lưu trữ danh sách thư được lưu trữ, vị trí của hệ thống quản lý lỗi, và một khảo sát tối thiểu). Những người quản lý dự án có lẽ sau đó chỉ như là một cái núm nằm tách khỏi một phân tích hoàn chỉnh, một dạng của phân tích buồng bệnh mà nó giúp được để chuẩn đoán một tình trạng sức khỏe của dự án bao gồm những chỉ số cùng một lúc trong những lĩnh vực cần cải tiến.

Một khi chúng ta đã có được các phương pháp, một sự phân loại và các mô hình, thì những cơ hội gia tăng từ sự mô phỏng, và sẽ trở nên chính xác hơn, thì những tác nhân tri thức, có thể là khổng lồ. Việc coi điểm bắt đầu đó của chúng ta là một hệ thống phức tạp khét tiếng, có thể là thú vị để tạo ra những mô hình năng động trong đó các thực thể khác nhau tham gia vào sự tạo ra phần mềm có thể mô hình hóa được. Rõ ràng, chúng ta càng biết về những yếu tố khác biệt nhiều hơn bao nhiêu, thì càng áp dụng được cho mô hình thực tế của chúng ta sẽ tốt hơn bấy nhiêu. Mặc dù một số đề xuất cho sự mô phỏng PMTD được biết tới, thì chúng khá là đơn giản và không hoàn chỉnh. Ở một mức độ nào đó, điều này là vì thực tế rằng vẫn còn một sự thiếu hụt trầm trọng sự hiểu biết về những tương tác mà chúng diễn ra trong sự tạo ra PMTD. Nếu chúng ta định đóng gói và xử lý đúng đắn những thông tin của các dự án thông qua lịch sử của chúng, thì những tác nhân này có thể trở thành cốt tử cho sự hiểu biết sự tiến hóa trong tương lai của chúng sẽ là gì. Mặc dù có nhiều đề xuất như là làm thế nào tiếp cận được vấn đề này, thì một trong những đề xuất tiến bộ nhất cho tới nay có thể thấy tại http://wwwai.wuwien.ac.at/~koch/oss-book/ [82].

7.8 Tóm lược

Trong phần tóm lược, chúng ta đã cố gắng chỉ ra trong chương này rằng kỹ thuật PMTD vẫn là một lĩnh vực còn non trẻ và chưa được khám phá. Những bước đầu của nó là nhờ vào sự cố gắng của các phóng viên đã đề xuất, không phải không thiếu sự chặt chẽ khoa học nhất định nào đó, một mô hình

phát triển hiệu quả hơn, nhưng qui trình dần dần được làm hướng tới một nghiên cứu có tính hệ thống của PMTD từ một viễn cảnh kỹ thuật. Hiện nay, sau vài năm của các báo cáo và phân tích định tính và định lượng về các dự án tự do, một nỗ lực khổng lồ đang được thực hiện để đạt được một hạ tầng toàn cầu mà nó làm cho có khả năng để phân loại, phân tích và mô hình hóa dự án trong một không gian hạn chế về thời gian và một phần theo cách tự động hóa. Khi việc phân tích các dự án PMTD dừng ở việc quá mất thời gian và nỗ lực như nó hiện nay, thì hình như là một giai đoạn mới trong kỹ thuật phần mềm đang xuất hiện trong khung cảnh được thiết kế chủ yếu để đoán trước sự tiến hóa của phần mềm và thấy trước được những phức tạp tiềm tàng.

8 Các môi trường và công nghệ phát triển

> “Các công cụ mà chúng ta sử dụng có một ảnh hưởng sâu sắc (và thủ đoạn!) tới những thói quen suy nghĩ của chúng ta, và, vì thế lên những khả năng suy nghĩ của chúng ta”.
>
> Edsger W. Dijkstra, "Làm thế nào chúng ta nói lên được những sự thực mà có thể gây tổn thương?"

Lùi về những năm mà các dự án PMTD đã tạo ra những công cụ (cũng tự do) và những hệ thống của riêng họ để đóng góp vào quá trình phát triển. Dù mỗi dự án tuân theo những qui định của nó và sử dụng tập hợp các công cụ của riêng nó, có những thực tế, những môi trường và những công nghệ nhất định mà chúng có thể được coi là thông thường trong thế giới của sự phát triển PMTD. Trong chương này chúng ta sẽ xem những thứ chung nhất đó và thảo luận ảnh hưởng của chúng lên sự quản lý và tiến hóa của các dự án.

8.1 Mô tả các môi trường, công cụ và hệ thống

Trước khi giải thích về các công cụ cụ thể nào đó, chúng ta sẽ xác định những đặc tính chung và những thuộc tính của chúng theo nhiệm vụ sẽ được thực hiện và theo cách mà các lập trình viên được tổ chức.

Đầu tiên, dù không nhất thiết là một yếu tố quyết định, thì thông thường đối với môi trường, các công cụ phát triển (và ngay cả máy tính ảo đích, nếu có) đều là tự do. Điều này không phải lúc nào cũng vậy. Ví dụ, dự án GNU, với mục tiêu để thay thế Unix, phải được phát triển trong và cho những hệ thống Unix sở hữu độc quyền cho tới khi Linux và FreeBSD xuất hiện. Hiện nay, đặc biệt khi PMTD được phát triển như một phần của một mô hình kinh doanh, thì xu thế là việc máy tính đích cũng có thể là một hệ thống sở hữu độc quyền, thường thông qua những máy ảo được xen kẽ (Java, Python, PHP, …). Trong mọi trường hợp, môi trường và máy ảo cần phải đủ chung và rẻ để cùng mang đến cho các đồng lập trình viên đủ để có cùng các công cụ.

Thứ 2, cũng để lôi cuốn số lượng lớn nhất có thể các đồng lập trình viên, các công cụ cần phải là *đơn giản*, *nổi tiếng* và có khả năng hoạt động trong các máy *tiết kiệm*. Có lẽ vì những lý do này mà thế giới PMTD khá là bảo thủ khi nói về các ngôn ngữ, công cụ và môi trường.

Thứ 3, mô hình phát triển của PMTD có xu hướng được phân tán cao độ, với nhiều người cộng tác tiềm năng trải rộng ra khắp thế giới. Vì lý do này thường các công cụ cộng tác không đồng bộ là cần thiết, mà chúng cùng một lúc cho phép sự phát triển tiến bộ dễ dàng, bất chấp số lượng và nhịp độ công việc của từng cộng tác viên với hàng loạt kiến trúc khác nhau trong đó chúng có thể biên dịch và kiểm thử các chương trình của chúng.

8.2 Các ngôn ngữ và công cụ có liên quan

Hầu hết PMTD được viết trong ngôn ngữ C, không chỉ vì C là ngôn ngữ tự nhiên của bất kỳ biến thể Unix nào (nền tảng thường thấy của PMTD), nhưng cũng vì nó được phổ biến rộng rãi, cả trong tâm trí của mọi người và trong các máy (GCC là một trình biên dịch tiêu chuẩn được cài đặt mặc định trong

hầu hết mọi phát tán). Chính xác vì những lý do này và vì sự hiệu quả của nó, Stallman đã khuyến cáo sử dụng nó trong các dự án GNU ("Các chuẩn lập trình GNU") [203]. Các ngôn ngữ khá tương tự khác là C++, cũng được hỗ trợ ngầm định bởi GCC, và Java, mà có sự tương tự nhất định và là phổ biến vì nó cho phép những phát triển cho các máy tính ảo sẵn sàng trong một dải rộng lớn các nền tảng. Nói chung, các lý do của kỹ thuật phần mềm là không được tính tới: theo SourceForge (phần 8.9.1), trong năm 2004, cứ mỗi 160 dự án được viết trong ngôn ngữ C thì có 1 dự án được viết trong Ada, dù ngôn ngữ sau được cho là một ngôn ngữ phù hợp hơn cho việc phát triển các chương trình có chất lượng. Cùng một lúc, tiếng Anh là *ngôn ngữ chính* của các lập trình viên PMTD, dù thực tế là Esperanto là một ngôn ngữ dễ hơn nhiều để học với một cấu trúc logic hơn nhiều. Các ngôn ngữ biên dịch được thiết kế cho việc tạo mẫu nhanh chóng các ứng dụng thông thường và các dịch vụ web như Perl, Python và PHP cũng là phổ biến.

Chỉ có C là ngôn ngữ tiêu chuẩn, *make* trở thành công cụ tiêu chuẩn để xây dựng chương trình, được đưa ra trong các tệp mã nguồn của nó. Một lập trình viên tự do sẽ thường sử dụng phiên bản GNU (GNU make) [36] hơn là thứ tương thích với BSD (Adam de Boor, "PMake - một sách chỉ dẫn") [100]. Chúng có thể được sử dụng để chỉ định các cây phụ thuộc giữa các tệp, và các qui định cho việc tạo ra các tệp phụ thuộc từ những thứ mà chúng phụ thuộc. Vì thế, chúng ta có thể chỉ định rằng một tệp đối tượng *x.o* phụ thuộc vào các tệp nguồn *x.c* và *x.h* và rằng để xây dựng nó chúng ta cần chạy *gcc -c x.c*. Hoặc rằng sự chạy chương trình của chúng ta phụ thuộc vào một bộ các đối tượng và được liên kết theo một cách nhất định nào đó. Khi chúng ta sử mã nguồn và sau đó chạy *make*, chỉ các module bị ảnh hưởng sẽ được biên dịch lại và đối tượng cuối cùng sẽ được liên kết lại. Đây là một công cụ mức rất thấp, vì, ví dụ, không có khả năng tìm ra bản thân nó khi một module cần phải được biên dịch lại trong C, mặc dù thực tế là có thể làm thế bằng việc kiểm tra các chuỗi của *includes*. Nó cũng rất mạnh, vì nó có thể kết hợp tất cả các công cụ biến đổi tệp để xây dựng các mục tiêu rất phức tạp của một dự án đa ngôn ngữ. Nhưng nó rất phức tạp và rất phụ thuộc vào các môi trường dạng Unix. Những giải pháp thay thế được đề xuất tốt hơn khác, như *jam* (Jam Product Information) [41], *aap* (Dự án Aap) [1] hoặc *ant* (Dự án Apache Ant) [7] rất hiếm khi được sử dụng (cái sau đang giành được sự phổ biến đặc biệt trong thế giới của Java).

Đưa ra tính hỗn hợp không đồng nhất của các hệ thống đang tồn tại ngay cả trong thế giới của Unix, chúng ta cũng sử dụng các công cụ được thiết kế để giúp tạo ra các chương trình khả chuyển được của chúng ta. Các công cụ GNU *autoconf* (http://www.gnu.org/software/autoconf) [10], *automake* (http://www.gnu.org/software/automake) [32] và *libtool* (http://www.gnu.org/software/libtool) [35] làm cho những nhiệm vụ này dễ dàng hơn trong các môi trường C và Unix. Đưa ra sự đa dạng của các ngôn ngữ, các tập hợp ký tự và các ngữ cảnh văn hóa, các lập trình viên C (và các lập trình viên sử dụng nhiều ngôn ngữ khác nhau) thường sử dụng *gettext* (http://www.gnu.org/software/gettext) [31] và những lựa chọn quốc tế hóa của thư viện tiêu chuẩn của C (http://www.gnu.org/software/libc) [34] cho việc lập trình các ứng dụng mà chúng có thể dễ dàng được bản địa hóa cho bất kỳ môi trường văn hóa nào trong thời gian thực.

Vì thế, khi chúng ta nhận được một gói nguồn, hầu hết có lẽ được viết trong C, được đóng gói với *tar*, được nén với *gzip*, được làm khả chuyển được với *autoconf* và các công cụ liên kết được, và có thể được xây dựng và cài đặt với *make*. Việc cài đặt của nó sẽ được triển khai trong một quá trình rất tương tự như sau:

```
tar xzvf package-1.3.5.tar.gz
cd package-1.3.5
./configure
make make install
```

8.3 Các môi trường phát triển tích hợp

Một *môi trường phát triển tích hợp* IDE (integrated development environment) là một hệ thống mà nó làm cho công việc của lập trình viên phần mềm dễ dàng hơn bằng việc tích hợp chắc chắn việc xuất bản có định hướng các ngôn ngữ, trình biên dịch hoặc phiên dịch, dò tìm và sửa lỗi, đo lường sự thực thi, hợp nhất các mã nguồn tới một hệ thống kiểm soát nguồn, …, thường theo dạng các module.

Không phải tất cả các lập trình viên PMTD đều thích các công cụ này, dù việc sử dụng chúng dần dần đã được mở rộng. Trong thế giới của PMTD, môi trường đầu tiên được sử dụng một cách tích cực là GNU Emacs (http://www.gnu.org/software/emacs/) [33], công việc ngôi sao của Richard Stallman, được viết và mở rộng được trong Emacs Lisp, mà có nhiều sự đóng góp cho nó.

Eclipse (Eclipse - Một nền tảng phát triển mở) [23] có thể được coi là IDE tham chiếu ngày nay trong thế giới của PMTD, với nhược điểm là nó làm việc tốt hơn (khoảng tháng 07/2007) trên một máy Java ảo không tự do (của Sun mà được hy vọng sẽ trở thành tự do sớm). Các môi trường khác là Kdevelop (http://www.kdevelop.org) [42] cho KDE, Anjuta (http://www.anjuta.org) [6] cho GNOME, Netbeans (http://www.netbeans.org) [51] của Sun cho Java và Code::Blocks (http://www.codeblocks.org) [18] cho các ứng dụng của C++.

8.4 Các cơ chế cộng tác cơ bản

PMTD là một hiện tượng được làm cho có khả năng bởi sự cộng tác của các cộng đồng phân tán và vì thế, đòi hỏi các công cụ để làm cho sự cộng tác đó có hiệu quả. Mặc dù từ lâu các băng từ về mặt vật lý đã được gửi đi, thì sự phát triển nhanh chóng của PMTD đã bắt đầu khi nó trở nên có khả năng giao tiếp một cách nhanh chóng với nhiều người và để phân phối mã chương trình cho họ hoặc trả lời với các bình luận và bản vá. Để cho tiện, thay vì việc gửi đi mã nguồn, các thông điệp có thể được sử dụng để gửi các thông tin về site mà từ đó mã nguồn có thể được thu thập. Trên thực tế, ngay từ đầu của những năm 70, thư điện tử đã là một mở rộng của giao thức truyền tệp của ARPANET.

Trong thế giới của Unix, trong những năm giữa thập kỷ 70, *uucp*, giao thức truyền tệp của Unix, đã được phát triển cho các máy tính giao tiếp thông qua quay số dial - up và các đường dây chuyên dụng, và trên đó thư điện tử đã được xây dựng, và vào năm 1979, kết nối đầu tiên của USENET qua UUCP. thông tin của USENET, một hệ thống diễn đàn có cấu trúc thứ bậc được phân tán bằng việc gây ngập tràn tới các site được sắp xếp theo thứ bậc, đã đóng một vai trò cơ bản trong sự phát triển của PMTD, gửi đi mã nguồn của các chương trình hoàn chỉnh tới các nhóm `comp.sources`.

Cùng một lúc, các danh sách thư đã được phát triển, trong số đó có BITNET (1981) danh sách thư mà các nhà quản lý đáng nhắc tới. Ngày nay xu thế là ưa các danh sách thư qua các nhóm tin dạng của

USENET. Lý do chính từng là sự lạm dụng cho các mục đích thương mại và thâm nhập trái phép của những người "thiếu suy nghĩ", can thiệp với sự ầm ĩ trong các cuộc thảo luận. Hơn nữa, các danh sách thư cung cấp sự kiểm soát nhiều hơn và có thể đạt tới nhiều người hơn. Những người nhận cần phải đăng ký và bất kỳ địa chỉ thư điện tử nào cũng là hợp lệ, ngay cả nếu không có sự truy cập trực tiếp tới Internet. Người quản trị danh sách thư có thể chọn để biết ai đăng ký hoặc bỏ đăng ký cho ai đó.

Những đóng góp có thể bị hạn chế chỉ đối với những thành viên hoặc lập trình viên có thể chọn để điều tiết các bài viết trước khi cho chúng xuất hiện[6].

Theo truyền thống, việc quản trị các danh sách thư được thực hiện bởi thư điện tử, sử dụng các thông điệp đặc biệt với một mật khẩu, cho phép người quản trị không có sự truy cập vĩnh viễn tới Internet, dù điều này đang trở thành một hiện tượng ngày càng hiếm, nghĩa là những người quản lý các danh sách thư phổ biến nhất ngày nay (Mailman, Người quản lý Danh sách Thư GNU) [46]) không thể được quản trị bởi thư điện tử, mà là sự cần thiết thông qua web. Các danh sách thư đóng một vai trò cốt tử trong thế giới của PMTD và trong nhiều trường hợp[7] chúng có thể là cách duy nhất để đóng góp.

Hiện hành, với sự phổ biến của web, nhiều nhóm thảo luận là những nhóm thảo luận thuần túy web hoặc weblogs, mà không có giao diện khác hơn là một giao diện được cung cấp bởi trình duyệt. Chúng có thể là chung, như ShashDot phổ biến (Slashdot: News for Nerds) [58] hoặc Barrapunto của Tây Ban Nha (http://barrapunto.com) [11], nơi mà PMTD mới được công bố hoặc thảo luận. Hoặc chúng có thể được chuyên môn hóa trong một chương trình đặc biệt nào đó; trong trường hợp này chúng thường được tích hợp với vài công cụ bổ sung trong các site cộng tác (xem phần 8.6.2). Cũng có các giao diện web cho các nhóm tin và các danh sách truyền thống.

Một cơ chế cộng tác khác mà đã trở thành phổ biến cùng một lúc, được dựa trên wikis, đặc biệt khi ý tưởng sẽ là để xây dựng một tài liệu cộng tác, như là đặc tả kỹ thuật cho một chương trình, một module hoặc một hệ thống. Chúng ta sẽ thảo luận điều này trong phần 8.6.2.

Cuối cùng, chúng ta phải nhắc tới các cơ chế tương tác được sử dụng bởi các lập trình viên để nói chuyện trong thời gian thực. Đối với PMTD nó không có xu hướng sẽ là một cơ chế thực tế, vì với tất cả các lập trình viên được phân tán khắp thế giới thì không dễ dàng để tìm ra một thời gian thuận lợi cho tất cả mọi người. Dù vậy, có vài dự án mà chúng sử dụng các công cụ chat bằng văn bản, hoặc thường hoặc tại các hội nghị ảo trong những ngày đã được định sẵn. Công cụ thường được sử dụng nhất là IRC (Internet Relay Chat, http://www.ietf.org/rfc/rfc2810.txt) [151], mà nó thường kết nối mọi người thông qua các "kênh" theo chủ đề được thiết lập trên cơ sở của một loạt các máy chủ cộng tác. Không là phổ biến cho các công cụ đa phương tiện để được sử dụng (âm thanh, hình ảnh, …) có lẽ vì các kết nối có chất lượng sẽ được yêu cầu mà không phải bất kỳ ai cũng có thể có và nó có thể dẫn tới các vấn đề với tính sẵn sàng của PMTD, và khó khăn của việc đăng ký và soạn thảo các kết quả của các hội thoại vì các lý do làm tài liệu.

6 Cũng có những nhóm tin có điều tiết

7 Ví dụ, những đóng góp cho Linux phải được thực hiện như các bản vá bằng văn bản cho danh sách linuxkernel@vger.kernel.org

8.5 Quản lý mã nguồn

Khuyến cáo cho bất kỳ dự án phát triển chương trình nào phải lưu trữ lịch sử của nó, vì một sửa đổi có thể tạo ra một lỗi ẩn được phát hiện sau này, ví dụ thế, và bản gốc cần phải được phục hồi lại, ít nhất để phân tích vấn đề. Nếu dự án được phát triển bởi vài người, thì tác giả của từng thay đổi cũng sẽ cần phải được ghi lại, vì cùng những lý do như đã giải thích ở trên. Nếu các phiên bản có đánh số của một dự án được thực hiện, thì chúng ta cần biết chính xác những phiên bản nào của từng module tạo nên một phần của từng phiên bản. Thường thì, một dự án sẽ giữ một phiên bản ổn định và phiên bản thực nghiệm khác; cả 2 phiên bản này cần phải được duy trì, gỡ rối, và sửa các lỗi được truyền từ phiên bản này sang phiên bản khác. Tất cả những điều này có thể được thực hiện bằng việc lưu và dán nhãn cho mỗi và mọi phiên bản của các tệp một cách đúng đắn, mà thường được xem là một giá thành quá thừa, mặc dù với các ổ đĩa hiện hành thì điều này đang trở nên ít đúng hơn. Những gì mà một *hệ thống kiểm soát nguồn*, còn được biết tới như một *hệ thống quản lý phiên bản*, thường làm, là để lưu giữ lịch sử các tệp như một tập hợp các khác biệt đối với một phiên bản, thường là phiên bản gần nhất, vì tính hiệu quả, cũng như việc dán nhãn cho từng sự khác biệt với siêu dữ liệu cần thiết.

Nhưng chúng ta cũng muốn một hệ thống các đặc tính này sẽ phục vụ cho nhiều lập trình viên để cộng tác một cách có hiệu quả mà không phải dẫm lên chân của nhau, mà không có việc gây trở ngại cho sự tiến bộ của nhau. Vì thế, chúng ta cần phải có khả năng để cho phép vài lập trình viên làm việc cùng một lúc, nhưng với một mức nhất định nào đó về kiểm soát. Sự kiểm soát này có thể là lạc quan hoặc bi quan. Với sự kiểm soát bi quan, một lập trình viên có thể dự trữ một số tệp cho bản thân mình để cải tiến trong một thời gian, trong khi đó không ai khác có thể động tới những tệp này. Điều này là rất an toàn, nhưng sẽ khóa các lập trình viên khác và có thể gây chậm trễ cho dự án, đặc biệt nếu lập trình viên mà đã khóa các tệp lại bận với những việc khác hoặc thậm chí đã quên về chúng. Việc cho phép những người khác để cải tiến là năng động hơn, nhưng nguy hiểm hơn, vì những sửa đổi không tương thích có thể xảy ra. Một hệ thống lạc quan cho phép cải tiến được thực hiện, nhưng cảnh báo cho chúng ta khi có những xung đột và trao cho chúng ta các công cụ để giải quyết chúng.

8.5.1 Hệ thống Phiên bản Đồng thời

Hệ thống phiên bản đồng thời CVS (Concurrent Version System) là một hệ thống quản lý nguồn lạc quan được thiết kế vào cuối những năm 80 và được sử dụng bởi đa số các dự án tự do (Hệ thống Phiên bản Đồng thời [20], *Phát triển mã nguồn mở bằng CVS* phiên bản 2 [113], Quản lý phiên bản với CVS [95]). Nó sử dụng một kho trung tâm được truy cập thông qua một hệ thống máy khách/máy chủ. Người quản trị site quyết định ai có sự truy cập tới kho này, hoặc tới những phần nào của kho, dù bình thường, thì một khi một lập trình viên đã được chấp nhận trong một chu kỳ tin cậy, thì anh ta sẽ truy cập được tới tất cả các tệp. Sự truy cập nặc danh, theo phương thức chỉ đọc, cũng có thể được cho phép cho bất kỳ ai.

Người cộng tác nặc danh

CVS *nặc danh* là một công cụ sống còn để thỏa mãn khái niệm "phiên bản sớm và thường xuyên" được bảo vệ bởi Eric Raymond. Bất kỳ người sử dụng nào cũng khắc khoải cố thử phiên bản mới nhất của một chương trình có thể trích nó ra từ CVS, phát hiện các lỗi và báo cáo về chúng, ngay cả ở dạng các

miếng vá với sự sửa cho đúng. Và nó có thể kiểm tra lịch sử đầy đủ của sự phát triển.

Hãy xem một chút vào cơ chế này. Một người sử dụng cao cấp muốn có được phiên bản mới nhất của module *mod* từ một kho truy cập được một cách nặc danh trong `progs.org`, thư mục `/var/lib/cvs` và giao thức `pserver`. Lần đầu tiên anh ta đưa ra ý định để gõ vào:

```
cvs -d:pserver:anonymous@progs.org:/var/lib/cvs login
```

Nếu một mật khẩu được yêu cầu, thì nó sẽ là người sử dụng nặc danh, mà nó sẽ được đăng ký trong một tệp cục bộ (hoạt động này là không thực sự cần thiết cho sự truy cập nặc danh, nhưng chương trình sẽ kêu nếu tệp này với mật khẩu không tồn tại). Sau đó, thứ quan trọng là có được bản sao đầu tiên của module:

```
cvs -d:pserver:anonymous@progs.org:/var/lib/cvs co mod
```

Điều này sẽ tạo ra một thư mục *mod* với tất cả các tệp của module này và các thư mục và một số siêu dữ liệu (các nội dung trong các thư mục con được gọi là CVS), mà nó sẽ cho phép, cùng với những thứ khác, không phải lặp đi lặp lại thông tin đã được cung cấp.

Người sử dụng cao cấp của chúng ta vào trong thư mục được tạo ra, tạo gói và kiểm thử nó:

```
cd mod
./configure
make
make install
```

Khi anh ta muốn có được một phiên bản mới, anh ta sẽ cập nhật một cách đơn giản bản sao của anh ta trong `mod`.

```
cd mod
cvs update
./configure
make
make install...
```

Nếu anh ta thấy một lỗi, anh ta có thể sửa nó ngay tại chỗ và sau đó gửi một bản vá thông qua thư điện tử cho người duy trì (cá nhân hoặc danh sách thư) chương trình:

```
cvs diff -ubB | mail -s "My patches" mod-maint@progs.org
```

Lập trình viên bình thường

Lập trình viên bình thường sẽ có một tài khoản trên máy chủ. Anh ta có thể sử dụng cơ chế y hệt và giao thức y hệt như người sử dụng nặc danh, thay thế sự *nặc danh* cho tên tài khoản của anh ta.

Một khi anh ta có một bản sao làm việc của module này, thì anh ta có thể tiến hành những thay đổi cần thiết, và khi anh ta xem xét rằng chúng đã là ổn định rồi, thì ủy thác những thay đổi đó tới kho. Ví dụ, nếu anh ta sửa các tệp `part.h` và `part.c`, anh ta sẽ ủy thác chúng như sau:

```
cvs ci part.h part.c
```

Trước khi hoàn tất hoạt động này, CVS sẽ hỏi anh ta cho một sự giải thích về những gì anh ta đã làm, mà nó sẽ được gắn tới cả các tệp log. Hơn nữa *số sửa lại* của mỗi tệp sẽ được tăng lên một đơn vị. Số này xác định mỗi thời điểm quan trọng trong lịch sử của một tệp và có thể được sử dụng để phục hồi từng thời điểm này một.

Khi nào thì một lập trình viên phải tiến hành một ủy thác? Đây là một câu hỏi của phương pháp mà các thành viên của dự án cần phải đồng ý, nhưng nó dường như chắc chắn rằng những thay đổi mà không biên dịch phải không được ủy thác. Nhưng cũng ưu tiên để cho qua một đoạn kiểm thử tối thiểu. Trong nhiều dự án thì sự phê chuẩn của một người giám sát của một dự án hoặc dự án phụ mà người này kiểm tra sự sửa đổi cũng được yêu cầu.

Trong việc phát triển sửa đổi, ai đó có thể đã sửa các tệp khác, hoặc thậm chí chính các tệp này. Vì thế được khuyến cáo cho những lập trình viên hãy làm một cập nhật khá thường xuyên cho bản sao của họ (*cvs cập nhật*). Nếu những tệp khác đã được sửa đổi, thì môi trường cũng có thể đã thay đổi và các kiểm thử mà đã được cho qua trước đó bây giờ có thể bị hỏng. Nếu cùng các tệp này đã được sửa đổi, thì có thể là những thay đổi này đã xảy ra hoặc ngay tại chỗ hoặc theo lệ thường mà chúng ta đã không động tới hoặc trong mã nguồn mà chúng ta đã sửa đổi. Trong trường hợp đầu không có xung đột (ít nhất có vẻ không) và hoạt động sửa đổi này "trộn" phiên bản của chúng ta với phiên bản của kho, tạo ra những tệp được kết hợp, với tất cả những thay đổi. Nếu không thì sẽ có một xung đột, trong trường hợp đó chúng ta cần phải thảo luận với lập trình viên mà đã thực hiện những thay đổi khác và đồng ý đối với một phiên bản cuối cùng.

Để nhận diện tốt hơn đối với mỗi thành phần của dự án, được khuyến cáo mang trực tiếp các thông tin sửa lại có liên quan. CVS có thể dán nhãn cho các mã nguồn và các đối tượng một cách tự động, theo điều kiện của việc tuân thủ một nguyên tắc nhất định nào đó. Ví dụ, nếu trong một chú giải của mã nguồn chúng ta viết từ khóa `$Id$`, thì mỗi lần tệp này được ủy thác tới kho, thì từ này sẽ được thay thế bằng một chuỗi xác định mà nó sẽ chỉ ra tên tệp, số sửa lại[8], ngày tháng và thời gian của ủy thác và tác giả:

8 Trong CVS các số sửa lại thường có 2 thành phần (chính và phụ), nhưng chúng có thể có 4, 6, ...

```
$Id: part.c,v 1.7 2003/07/11 08:20:47 joaquin Exp $
```

Nếu chúng ta đưa từ khóa này vào trong một chuỗi của chương trình, khi được biên dịch thì chuỗi này sẽ xuất hiện trong đối tượng và trong chỗ thi hành được, làm cho có khả năng để xác định nó với một công cụ (*ident*).

Lưu ý

Vì những lý do an ninh, đối với những tài khoản với các quyền ghi, ssh có xu hướng sẽ được sử dụng, vì nó cung cấp một kênh được xác thực và mã hóa.

Người quản trị

Chắc chắn, người quản trị có trách nhiệm về phần phức tạp nhất của việc duy trì kho. Ví dụ, họ cần phải đăng ký chương trình, đưa ra các quyền cho các lập trình viên và điều phối họ, gắn nhãn cho các phiên bản, …

Thực tế thông thường đối với tất cả các dự án phải có một phiên bản ổn định và một phiên bản thực nghiệm. Để làm điều này chúng ta tạo ra các nhánh. Trong khi những người chuyên tâm để duy trì các lỗi được sửa đúng trong nhánh ổn định, thì những phát triển mới được thực hiện trong các nhánh thực nghiệm. Khi nhánh thực nghiệm ổn định hóa, nó sẽ được đưa qua nhánh ổn định, nhưng không phải không áp dụng trước đó những chỉnh sửa cho đúng được thực hiện đối với nhánh ổn định trước đó. Hoạt động này được gọi là việc *trộn*, điều này là tinh vi và được hỗ trợ bởi CVS, mặc dù theo một cách hơi sơ đẳng. Ý tưởng này có thể được mở rộng cho khái niệm của các nhánh thực nghiệm mà chúng tiến hóa theo các hướng khác nhau, mà chúng có hoặc không thể trở thành kết cục tốt, và rằng trong mọi trường hợp, trừ phi chúng là những kết cục chết, sẽ phải được tích hợp một phần hoặc toàn phần vào trong sản phẩm ổn định, với những sự trộn phù hợp.

Một quyền mà PMTD trao cho chúng ta là để sửa đổi một chương trình cho việc sử dụng cá nhân. Mặc dù được mong muốn để đóng góp tất cả những cải tiến cho kho chung, thường những sửa đổi mà chúng ta muốn thực hiện là quá đặc chủng và không thú vị cho công chúng nói chung. Nhưng chúng ta có quan tâm trong việc kết hợp sự tiến bộ trong chương trình gốc. Điều này có thể được thực hiện với một dạng đặc biệt của việc chia nhánh và trộn (*các nhánh của nhà cung cấp*).

Người quản trị cũng có thể tạo điều kiện thuận lợi cho sự phối hợp của đội thông qua các cơ chế tự động, như là bằng việc tạo ra các thông điệp thư điện tử khi những sự kiện nhất định nào đó xảy ra, như là sự ủy thác, hoặc việc áp đặt những hành động tự động nhất định nào đó để triển khai trước một sự ủy thác, như là những kiểm tra tự động của kiểu style, sự biên dịch, hoặc những thử nghiệm.

8.5.2 Các hệ thống quản lý nguồn khác

Bất chấp là hệ thống kiểm soát phiên bản được sử dụng rộng rãi nhất, CVS có một số nhược điểm đáng chú ý:

1. CVS không hỗ trợ việc đổi tên hoặc thay đổi thư mục các tệp, hoặc siêu dữ liệu (người chủ, các quyền, …) hoặc các kết nối tượng trưng.
2. Vì nó là một tiến hóa của một hệ thống kiểm soát phiên bản cho các tệp riêng rẽ, về mặt tự nhiên nó không hỗ trợ kiểm soát phiên bản cho các nhóm đầy đủ.
3. CVS không hỗ trợ các tập hợp những thay đổi cố kết. Quả thực, việc bổ sung thêm một tính năng hoặc việc sửa cho đúng một lỗi có thể liên quan tới việc thay đổi một số tệp. Những thay đổi này phải là nguyên tử.
4. Trong CVS việc sử dụng các nhánh và trộn khá phức tạp. Trên thực tế, nếu chúng ta tạo ra một nhánh thực nghiệm của một dự án và muốn đưa vào những sửa đổi cho đúng được thực hiện cho phiên bản ổn định, thì chúng ta cần biết chi tiết các sửa đổi nào đã được thực hiện rồi và cái nào chưa, sao cho không thực hiện chúng vài lần.
5. CVS phụ thuộc vào một máy chủ trung tâm, và dù nó là có khả năng để làm việc mà không có một kết nối, thì chúng ta phải cần một kết nối cho việc tạo ra những phiên bản, việc so sánh và trộn chúng.
6. CVS không tạo, mà không trợ giúp các công cụ một cách riêng rẽ, tệp `changelog`, mà nó chỉ ra lịch sử toàn cầu về những thay đổi của một dự án.
7. CVS không hỗ trợ tốt các dự án với một số lượng lớn các tệp, như trong trường hợp của nhân Linux.

Và vâng, còn có những hệ thống tự do khác mà nó giải quyết một vài trong số các vấn đề này. Chúng ta có thể nhấn mạnh tới người kế nghiệp đã được nhắc tới của CVS, Subversion - Phiên bản phụ (http://subversion.tigris.org) [62], (http://svnbook.red-bean.com/) [96], mà nó giải quyết một cách hoàn toàn những vấn đề cơ bản của CVS và có thể sử dụng những mở rộng HTTP (WebDAV) để vượt qua các chính sách an ninh hay sinh sự.

Mô hình phát triển này dựa trên một kho trung tâm, dù phù hợp cho công việc cộng tác, không thỏa mãn cho tất cả những mong đợi, vì có khả năng tạo ra những nhánh phát triển của riêng mà chúng ta dựa trên, một mặt, vào tính có thể truy cập được tới máy chủ và việc vận hành tốt và, mặt khác, dựa vào việc trao cho các quản trị viên của máy chủ đó. Đôi khi các kho phân tán được yêu cầu để cho phép bất kỳ ai có được một kho với một nhánh riêng tư hoặc công cộng mà nó có thể trộn được hoặc không với nhánh chính thống. Điều này giải thích cách mà công việc GNU *arch* (Hệ thống kiểm soát xem xét lại Arch) [8] hoặc *bazaar* (Phần mềm Kiểm soát Xem xét lại Phân tán GPL của Bazaar) [12], cũng như hệ thống sở hữu độc quyền BitKeeper (Quản lý Nguồn Bitkeeper) [14], được chọn bởi Linus Torvalds để duy trì Linux từ tháng 02/2002, vì theo ông đã không có công cụ tự do nào phù hợp. Điều này nói rằng việc sử dụng Bitkeeper làm tăng nhanh gấp đôi bước phát triển của Linux. Bất chấp, quyết định này đã tới dưới sự chỉ trích nặng nề vì nó là sở hữu độc quyền, với một giấy phép mà đã cho phép các dự án tự do giành được nó miễn phí trong điều kiện là tất cả những thay đổi ủy thác với siêu dữ liệu của chúng được ghi lại nhật ký trên một máy chủ công cộng được chỉ định bởi những người chủ và có thể truy cập được đối với bất kỳ ai, và luôn trong điều kiện rằng người được cấp phép đã không cố gắng phát triển hệ thống kiểm soát nguồn khác để cạnh tranh với nó. Chính xác đã có dự định phát triển một sản phẩm tự do tương thích được bởi một nhân viên của cùng công ty nơi mà Linus Torvalds đã làm

việc mà đã làm bùng nổ sự thay đổi trong hệ thống quản lý nguồn. Linus đã nhanh chóng phát triển một thay thế tạm thời, *git* (“Trang sổ tay của Git”) [218], mà nó sớm trở thành khẳng định, cô đọng tất cả kinh nghiệm của sự phát triển cộng tác và phi tập trung của Linux: nó hỗ trợ các dự án kích cỡ rộng lớn theo một cách phi tập trung, tạo điều kiện thuận lợi ở một mức độ nào đó cho sự phát triển của các nhánh thực nghiệm và việc trộn của chúng với những nhánh khác hoặc với nhánh chính, với các cơ chế an ninh có mã hóa mà nó ngăn ngừa được việc sửa tệp log. Vào tháng 04/2005, Linux được duy trì bằng việc sử dụng *git* hoặc các đồ bọc của nó (ví dụ, *cogito* “Trang sổ tay của Cogito”) [90].

8.6 Tài liệu

Trong thế giới của PMTD, các trình xử lý văn bản dạng WYSIWYG (nhìn thấy gì có cái nấy) và các công cụ bộ phần mềm văn phòng khác mà là rất thành công trong các môi trường khác được sử dụng một cách công khai, ngay cả dù đã có những công cụ tự do như OpenOffice.org. Điều này nhờ vào một số yếu tố quan trọng:

- Được khuyến cáo để duy trì tài liệu theo kiểm soát nguồn, và các hệ thống kiểm soát nguồn, như CVS, dù chúng chấp nhận các định dạng nhị phân, ưu tiên và xử lý được với các công cụ được phát triển cho các chương trình mà chúng cho phép chúng ta thấy được những khác biệt giữa các phiên bản một cách dễ dàng, để tạo ra và áp dụng các bản vá dựa trên những khác biệt đó, và để triển khai các phần trộn.
- Một số giấy phép tài liệu tự do, đặc biệt là GFDL (phần 10.2.1), đòi hỏi các định dạng minh bạch, đặc biệt vì chúng làm cho công việc dễ dàng hơn cho những người mà chuẩn bị các tài liệu dẫn xuất.
- Các công cụ WYSIWYG thường không chứa bất kỳ thông tin nào khác so với sự mường tượng chính xác, làm cho nó rất khó khăn, nếu không nói là không thể, để xác định các tác giả, hoặc các đầu đề, hoặc sự chuyển đổi sang các định dạng khác. Ngay cả nếu chúng cho phép sự chuyển đổi sang các định dạng khác, thì điều này có xu hướng sẽ được thực hiện một cách tương tác, và thường không thể tự động hóa được (việc sử dụng *make*, ví dụ vậy).
- Nói chung, các ứng dụng văn phòng tạo ra những định dạng tệp kích thước lớn, mà nó là một tính năng không mong muốn cho cả các lập trình viên và các kho.

  Lưu ý

  Trong Unix các công cụ thông thường nhất cho những hoạt động này là *diff, diff3*, và và *trộn*.

Đối với tất cả những thứ ở trên, các lập trình viên tự do, đã quen với việc lập trình và biên dịch, thích các định dạng tài liệu minh bạch hơn, trong mọi trường hợp văn bản đơn giản thuần túy và trong nhiều định dạng tài liệu có thể xử lý được khác.

Các định dạng có thể xử lý được được sử dụng không nhiều. Theo truyền thống, trong thế giới của các chương trình Unix được làm thành tài liệu trong những định dạng được mong đợi bởi họ các trình xử lý *roff*, với một phiên bản tự do (GNU *troff*) [37] của Norman Walsh. Dù sao, thực tế này cũng đã dần dần bị quên lãng, ngoại trừ các trang làm bằng tay theo truyền thống, vì nó hầu như buộc phải chuẩn bị các

trang bằng tay cho các công cụ cơ bản nhất của hệ thống. Vì nhiều trang làm bằng tay đã gia tăng quá nhiều nên nó hoàn toàn chỉ phù hợp để gọi chúng là *các trang*, cần phải chuẩn bị định dạng siêu văn bản thay thế vì nó cho phép các tài liệu cấu trúc được với các chỉ mục và tham chiếu chéo. Dự án GNU đã thiết kế định dạng *texinfo* (Texinfo - Hệ thống làm Tài liệu GNU) [63] và biến nó thành chuẩn. Định dạng này cho phép các tài liệu có thể điều khiển được làm được bằng công cụ *info* hoặc bên trong trình soạn thảo *emacs*, và đổi lại, để có được các bản in tài liệu chất lượng sử dụng trình xử lý văn bản TeX, của Donald Knuth (The TeXbook) [156].

Định dạng *texinfo* có thể được dịch thành nhiều trang HTML nếu cần, và nhiều người thích xem thông tin hơn với một trình duyệt web, nhưng khả năng tìm kiếm cho các từ trong một tài liệu bị mất. Đây là một trong những kết quả không mong muốn của sự phổ biến của HTML, khi mà các trình duyệt không triển khai khái niệm *tài liệu nhiều trang*, dù thực tế là có các yếu tố liên kết cho phép các phần được kết nối nội bộ với nhau.

Có một yêu cầu áp đảo cho khả năng xem các tài liệu phức tạp như những trang web nhiều trang có thể duyệt được dễ dàng. Có những người viết tài liệu trong LaTeX (*Chỉ dẫn và sổ tay tham chiếu của người sử dụng LaTeX*) [163], cũng là một ứng dụng của TeX, rất phổ biến trong các nhà khoa học, diễn đạt được nhiều hơn so với Texinfo và có thể chuyển đổi được sang nhiều trang HTML với các công cụ nhất định nào đó (The LaTeX Web Companion) [130], trong điều kiện một nguyên tắc nhất định nào đó được duy trì. Quả thực, các ứng dụng của TeX là những tập hợp các macros mà chúng phối hợp với các toán tử in được ở mức thấp để chuyển đổi chúng thành các ngôn ngữ trừu tượng làm việc được với các khái niệm mức cao (tác giả, đầu đề, tóm lược, chương, phần, …). Nếu chúng ta chỉ sử dụng các macro cơ bản, thì sự chuyển đổi là đơn giản. Nhưng vì không ai ngăn cản việc sử dụng các toán tử ở mức thấp và, hơn nữa, có những số lượng khổng lồ các gói macro nằm ngoài khả năng duy trì của những người duy trì các công cụ chuyển đổi nên khó mà đạt được những chuyển đổi tốt.

8.6.1 DocBook

Vấn đề bắt nguồn từ thực tế là không có sự phân biệt giữa nội dung và trình bày, cả trong TeX lẫn trong *nroff*, khi sự trừu tượng được xây dựng trong các lớp. Sự phân biệt này được làm bởi các ứng dụng SGML (*ngôn ngữ đánh dấu tổng quát theo chuẩn*) [81] và XML (ngôn ngữ đánh dấu mở rộng) [224], nơi mà sự trình bày được chỉ định với các bảng kiểu (*style sheet*) riêng biệt hoàn toàn. Các ứng dụng rất đơn giản của SGML đã sớm trở nên được sử dụng, như `linuxdoc` và `debiandoc`, nhưng vì khả năng diễn tả hạn chế của chúng, DocBook đã được chọn. (*DocBook: chỉ dẫn cuối cùng*) [225].

DocBook là một ứng dụng SGML ban đầu được phát triển cho các tài liệu kỹ thuật về công nghệ thông tin và bây giờ có một phương án của XML. Hiện tại, DocBook là chuẩn định dạng tài liệu tự do cho nhiều dự án (Linux Documentation Project, KDE, GNOME, Mandriva Linux, etc.) và là một mục tiêu để đạt được đối với những dự án khác (Linux, *BSD, Debian, etc).

Tuy nhiên, DocBook là một ngôn ngữ phức tạp, gây ra bởi các thẻ, mà nó có nghĩa là hữu dụng để có các công cụ để trợ giúp việc soạn thảo, ngay cả nếu chúng rất cơ bản và hiếm hoi; một trong những công cụ phổ biến nhất dạng này là phương thức psgml của emacs. Còn khó đối với qui trình và các trình xử lý tự do mà vẫn tạo ra được một chất lượng rất hấp dẫn các tài liệu.

8.6.2 Wikis

Nhiều người thấy quá phức tạp để viết tài liệu với những ngôn ngữ phức tạp như DocBook và các cơ chế cộng tác như CVS. Điều này giải thích vì sao một cơ chế mới về sự cộng tác cho sự chuẩn bị tài liệu trực tuyến thông qua web đã trở nên phổ biến, được gọi là *wiki*, và được sáng tạo bởi Ward Cunningham("Các nguyên lý thiết kế Wiki") [97]. Lần đầu tiên được đưa vào phục vụ năm 1995 và bây giờ được sử dụng nhiều trong một dãy rộng rãi của nhiều biến thể cho việc chuẩn bị các tài liệu rất năng động, không được thiết kế cho việc in ấn và thường với một vòng đời ngắn (ví dụ, tổ chức hội nghị).

Không giống như DocBook, một *wiki* có một ngôn ngữ rất đơn giản và đánh dấu trùng khớp mà nó được làm nhớ lại về trình bày cuối cùng, không chính xác phải giống nó. Ví dụ, các đoạn sẽ được tách bởi một dòng trống, các yếu tố của một danh sách được bắt đầu với một dấu gạch nối nếu không được đánh số và với một số 0 nếu chúng được đánh số, và các ô của bảng được cách nhau bởi các đường thẳng đứng và nằm ngang.

Không tồn tại khái niệm của một "tài liệu đầy đủ", mà một *wiki* là hơn một tập hợp của các tài liệu nhỏ được liên kết nội bộ với nhau được tạo ra khi cần thiết phải giải thích một khái niệm hoặc chủ đề mới. Các tài liệu này được tạo ra hầu như tự động, vì công cụ soạn thảo chỉ rất rõ ràng rằng chúng ta phải đưa vào một khái niệm (thông qua một tên *WikiName*, hầu như luôn 2 từ được liên kết với nhau với ký tự đầu được viết hoa). Khó mà bất kỳ *wiki* nào cho phép các siêu liên kết bên trong cùng một trang.

Không giống như CVS, bất kỳ ai cũng có thể viết trong một *wiki*, dù được khuyến cáo cho tác giả để xác định bản thân mình bằng việc đăng ký trước đó. Khi chúng ta viếng thăm một *wiki* chúng ta có thể thấy rằng tất cả các trang có một núm cho phép các trang soạn sửa được. Nếu được nhấn, trình duyệt sẽ chỉ cho chúng ta một mẫu với mã nguồn của tài liệu, mà chúng ta sẽ có khả năng thay đổi. Đây là một trình soạn thảo WYSIWYG, mà không khuyến khích bất kỳ ai chỉ muốn can thiệp vào, nhưng đủ đơn giản cho bất kỳ ai có quan tâm có khả năng sửa đổi các tài liệu với nỗ lực rất ít.

Wikis có sự kiểm soát phiên bản tài liệu của riêng nó, theo một cách thức mà tất cả các phiên bản của chúng thường có thể truy cập được, chỉ ra ai đã tạo ra chúng và khi nào. chúng cũng có thể dễ dàng so sánh được. Hơn nữa, chúng có xu hướng đưa vào cơ chế tìm kiếm, ít nhất cho mỗi tên trang và nội dung từ.

Thông thường, tác giả ban đầu của một trang sẽ muốn biết những thay đổi gì được thực hiện với nó. Để làm thế, anh ta có thể đăng ký cho những thay đổi đó và nhận được những lưu ý của chúng bằng thư điện tử. Đôi khi người thấy một tài liệu sẽ không dám thay đổi gì cả, nhưng có thể đưa vào một bình luận. Thông thường, tất cả *các trang wiki* có một nhóm thảo luận các bình luận có liên quan được dán ở cuối của tài liệu, mà hoặc tác giả ban đầu hoặc bất kỳ ai mà đóng vai trò của người biên tập có thể sử dụng để thiết lập lại văn bản gốc ban đầu, có thể bằng việc chuyển các mệnh đề từ các bình luận vào các chỗ phù hợp.

Khuyến cáo

Cách tốt nhất để hiểu khái niệm wiki là truy cập vào một *wiki* và trải nghiệm trên một trang được thiết kế

cho mục đích này, thường được gọi là một Hộp Cát (SandBox).

8.7 Quản lý lỗi và các vấn đề khác

Một trong những điểm mạnh của mô hình phát triển tự do là việc cộng đồng đóng góp với những báo cáo lỗi và cảm thấy rằng những báo cáo hoặc giải pháp này đưa ra được sự chú ý. Điều này đòi hỏi một cơ chế báo cáo lỗi đơn giản, sao cho các lập trình viên có thể nhận được những thông tin đầy đủ, theo một cách có hệ thống và chứa đựng tất cả các chi tiết cần thiết, hoặc được cung cấp bởi người đóng góp, với một sự giải thích về những gì đang xảy ra, mức độ quan trọng và giải pháp có thể, hoặc thông qua một cơ chế tự động mà nó xác định, ví dụ, phiên bản và môi trường của chương trình trong đó nó hoạt động. Các lỗi cũng phải được lưu giữ trong một cơ sở dữ liệu mà có thể tra cứu được, để xem liệu một lỗi đã được giao tiếp, được sửa hay chưa, mức độ quan trọng của nó, …

Có một vài hệ thống như thế này, với các triết lý khác nhau. Một số thông qua web, số khác thông qua thư điện tử, thông qua một số chương trình trung gian. Tất cả chúng có một giao diện web cho sự tư vấn. Một số cho phép các báo cáo nặc danh, trong khi số khác yêu cầu sự xác thực (một địa chỉ thư điện tử hợp lệ) để ngăn ngừa nhiễu. Mặc dù những thủ tục web dường như là đơn giản nhất, chúng không dễ dàng có được thông tin một cách tự động trong môi trường lỗi. Ví dụ, hệ thống của Debian cung cấp các chương trình như báo cáo lỗi reportbug, mà sau khi yêu cầu tên của gói mà chúng ta muốn báo cáo về nó, sẽ tra cứu lỗi cho máy chủ về các lỗi được báo cáo cho nó. Nếu không ai tham chiếu tới vấn đề của chúng ta, chúng ta sẽ được yêu cầu cho một mô tả về nó, mức độ quan trọng của nó (“nguy kịch”, “nghiêm trọng”, “hệ trọng”, “quan trọng”, “không thể tạo lại được từ mã nguồn”, “thông thường”, “không quan trọng” hoặc “gợi ý”) và các nhãn về chủng loại của nó (ví dụ, “an ninh”). Tiếp sau điều này, nếu chúng ta khẳng định yêu cầu này, thì nó sẽ tự động tìm ra phiên bản của gói và những thứ mà nó phụ thuộc vào, bổ sung cho cả phiên bản và kiến trúc của nhân. Rõ ràng, nó biết địa chỉ thư điện tử, nên nó gửi cho đúng site một báo cáo tương tự thứ sau đây:

```
Package: w3m-ssl
Version: 0.2.1-4
Severity: important
After reloading a page containing complex tables several dozen times, w3m had used
all physical memory and thrashing commenced. This is an Alpha machine.
--System Information
Debian Release: testing/unstable
Kernel Version: Linux romana 2.2.19 #1 Fri Jun 1 18:20:08 PDT 2001 alpha unknown
Versions of the packages w3m-ssl depends on:
ii libc6.1 2.2.3-7 GNU C Library: Shared libraries and Timezone data
ii libgc5 5.0.alpha4-8 Conservative garbage collector for C
ii libgpmg1 1.19.3-6 General Purpose Mouse Library [libc6]
```

```
ii libncurses5 5.2.20010318-3 Shared libraries for terminal handling

ii libssl0.9.6 0.9.6a-3 SSL shared libraries ii w3m 0.2.1-2 WWW browsable pager
with

tables/frames support
```

Thông điệp này sinh ra một con số của lỗi mà được trả lại cho chúng ta, gửi tới người duy trì và được lưu trong cơ sở dữ liệu. Khi lỗi này được giải quyết, chúng ta cũng sẽ nhận được một thông báo. Mỗi lỗi có một địa chỉ thư điện tử được chỉ định tới nó mà có thể được sử dụng để cung cấp thông tin bổ sung, ví dụ vậy. Chúng ta có thể tra cứu trong cơ sở dữ liệu lỗi http://bugs.debian.org bất kỳ lúc nào.

Đôi khi các hệ thống giám sát lỗi có các cơ chế để chỉ định ai đó giải quyết lỗi và thiết lập một thời hạn chót. Cũng có những vấn đề khác, như là các công việc treo, những cải tiến theo yêu cầu, các bản dịch, …. , mà đòi hỏi các cơ chế quản lý tương tự. với PMTD chúng ta không thể sử dụng chung các cơ chế rất cứng nhắc cho việc quản lý các nhiệm vụ mà từng lập trình viên phải làm. Sau tất cả, nhiều cộng tác viên là những tình nguyện viên và không thể bị bắt buộc phải làm bất kỳ thứ gì. Dù vậy, các nhiệm vụ có thể được xác định và chúng ta có thể chờ cho ai đó đăng ký vào hệ thống và nhận chúng trong một khoảng thời gian đã được công bố. Liệu có sự kiểm soát với những gì mà những người nhất định nào đó có thể làm hay không làm, luôn được khuyến cáo để kiểm soát tất cả các nhiệm vụ mà chúng cần phải được thực hiện, ai và những gì chúng phụ thuộc vào, mức độ quan trọng của chúng, và ai đang làm việc về chúng. Nhiều dự án quan trọng quản lý những khía cạnh này bằng việc sử dụng Bugzilla (*Chỉ dẫn về Bugzilla*) [89] hoặc những dẫn xuất từ nó. Đôi khi ai đó làm việc trong một dự án có thể phát hiện ra một lỗi trong một dự án khác mà công việc của anh ta phụ thuộc vào, nhưng nó có hệ thống quản lý lỗi khác với hệ thống mà anh ta đã quen. Điều này là đặc biệt đúng cho những người sử dụng của các phát tán mà họ muốn sử dụng một công cụ duy nhất cho việc báo cáo và giám sát việc giải quyết lỗi. Để tạo điều kiện thuận lợi cho việc báo cáo và giám sát các lỗi này, được tư vấn để tổ chức lại thành *liên đoàn* các hệ thống khác nhau, như được thực hiện bởi *Malone* (Trình giám sát lỗi Malone) [47].

8.8 Hỗ trợ cho các kiến trúc khác

Sự hỗ trợ tối thiểu được yêu cầu để làm việc với một chương trình khả chuyển là truy cập tới các *trại biên dịch*, mà nó cho phép chương trình được biên dịch trên các kiến trúc và các hệ điều hành khác nhau. Ví dụ, SourceForge (phần 8.9.1) đã đưa ra một thời gian các môi trường của Debian GNU/Linux cho Intel x86, DEC Alpha, PowerPC và SPARC, bổ sung thêm vào các môi trường của Solaris và Mac OS/X.

Cũng hữu dụng để có khả năng kiểm thử (không chỉ biên dịch) chương trình trong các môi trường đó. Nhưng dịch vụ này đòi hỏi nhiều hơn các tài nguyên và nhiều hơn về thời gian của người quản trị. Dịch vụ biên dịch có thể là đã thực dụng, vì thường thì chúng ta cần phải cung cấp các môi trường biên dịch cho vài ngôn ngữ, với một số lượng lớn các thư viện. Nếu những gì chúng ta muốn làm là để kiểm thử bất kỳ chương trình nào, thì các khó khăn gia tăng theo hàm số mũ, không chỉ vì nó rất khó khăn để có được những tài nguyên cần thiết sẵn sàng, mà còn vì những lý do an ninh, mà chúng có thể làm cho nó

trở nên cực kỳ phức tạp để quản trị các hệ thống này. Tuy vậy, có một số ít các dịch vụ của *trại biên dịch*, với các cài đặt chuẩn của nhiều kiến trúc, mà chúng có thể cho phép chúng ta kiểm thử vài thứ.

Những trại công khai được nhắc tới ở trên thường là một dịch vụ mà nó đòi hỏi sử dụng bằng tay. Lập trình viên được mời sao chép các tệp của anh ta vào một trong những máy tính, biên dịch chúng và kiểm thử kết quả. Anh ta sẽ có thể phải thực hiện nó lần này lần khác, trước khi để đưa ra được một phiên bản quan trọng của chương trình. Có thể thú vị hơn nhiều cho những biên dịch và thi hành các kiểm thử theo lối đi giật lùi được triển khai một cách có hệ thống, theo một cách thức được tự động hóa, ví dụ như là mỗi đêm, nếu có những thay đổi trong mã nguồn. Đây là cách mà một số dự án quan trọng hoạt động, mà nó cung cấp hạ tầng của riêng chúng cho các lập trình viên bên ngoài, mà có xu hướng được gọi là *tinderbox* (hộp bùi nhùi). Đây là trường hợp của Mozilla, được cấp vốn bởi Netscape, nhóm bùi nhùi của họ (http://www.mozilla.org/tinderbox.html) [50] có một giao diện web cho các kết quả trong đó nó hoạt động. Giao diện này có quan hệ chặt chẽ với CVS và chỉ ra những kết quả cho các tình trạng khác nhau (giữa các lần ủy thác), xác định người có trách nhiệm đối với các lỗi, và tạo điều kiện thuận lợi cho sự tiến bộ, bằng việc vượt qua vấn đề cho tới khi nó được giải quyết. Các hộp bùi nhùi cũng được sử dụng bởi các dự án như OpenOffice và FreeBSD, ít nhất là vậy.

8.9 Các site hỗ trợ phát triển

Các site hỗ trợ phát triển đưa ra, ít nhiều theo cách được tích hợp, tất cả các dịch vụ được mô tả ở trên cộng với một vài dịch vụ bổ sung mà chúng cho phép các dự án tìm kiếm được theo các chủng loại và để phân loại chúng theo một số thuộc tính đơn giản của hoạt động. Điều này là dành cho lập trình viên để thiết lập và quản trị toàn bộ một hạ tầng cho sự cộng tác, cho phép anh ta tập trung vào dự án.

8.9.1 SourceForge

Về dạng dịch vụ này, một trong những thứ đầu tiên đã được thiết lập, và nổi tiếng nhất, là SourceForge (http://sourceforge.net) [61], được quản lý bởi OSDN (Mạng Phát triển Phần mềm Mở), một bộ phận của VA Software, mà tới tháng 03/2007 chứa hơn 144,000 dự án. Nó được cấu trúc hóa xung quanh một tập hợp các chương trình với cùng tên, và các tên với phiên bản cho tới 2 là PMTD.

SourceForge, như một mẫu cho dạng site này, đưa ra một giao diện web hoặc cổng truy cập toàn cầu (http://sourceforge.net/) và một cổng phụ cho từng dự án (http://proyecto.sourceforge.net). Giao diện toàn cầu đưa ra các tin, quảng cáo, đường liên kết, và một lời mời để trở thành một thành viên hoặc tham gia nếu chúng ta đã là thành viên. Để cộng tác trên site này, được khuyến cáo trở thành một thành viên, và là bổn phận nếu chúng ta muốn tạo ra một dự án mới hoặc để tham gia vào một dự án đang tồn tại. Để trở thành một khán giả thì điều này là không cần thiết, và như vậy, chúng ta có thể xem những dự án nào đang trải qua sự phát triển tích cực nhất hoặc được tải về thường xuyên nhất, và tìm ra những dự án theo chủng loại hoặc từ khóa mô tả, và chúng sẽ xuất hiện theo trật tự các mức hoạt động. Đối với từng dự án chúng ta có thể thấy mô tả, tình trạng (alpha, beta, sản phẩm) của nó, những nhận diện của nó (ngôn ngữ lập trình, hệ điều hành, chủ đề, dạng người sử dụng, ngôn ngữ, giấy phép của nó...), các lỗi và các khía cạnh treo hoặc được phục hồi, các mức hoạt động qua thời gian..., hoặc tải nó về.

Chúng ta có thể cũng tham gia vào các nhóm thảo luận hoặc báo cáo lỗi, thậm chí là một cách nặc danh, mà không được khuyến cáo cho lắm (vì, ví dụ, chúng ta có thể không có được câu trả lời).

Bất kỳ người sử dụng được xác thực nào cũng có thể yêu cầu để đăng ký một dự án, mà những người quản trị sẽ chấp nhận có điều kiện rằng nó thỏa mãn các chính sách của site, mà trong trường hợp của SourceForge là khá tự do. Một khi được xác thực, người tạo nên có thể đăng ký cho những người sử dụng khác như những quản trị viên bổ sung hoặc như các lập trình viên, với sự truy cập để sửa mã nguồn. Tiếp sau sự xác thực, sẽ không có nhiều sự kiểm soát nào hơn nữa đối với dự án, mà nó có nghĩa là có nhiều dự án chết. Điều này không làm bối rối cho những người sử dụng quá nhiều, vì việc tìm kiếm dự án sẽ phân loại các dự án theo mức hoạt động, nghĩa là các dự án có mức hoạt động bằng 0 hoặc thấp hầu như sẽ không thấy. Những dự án này có rủi ro sẽ bị hạn chế bởi các chủ site này. Các dịch vụ mà SourceForge đưa ra cho một dự án, và chúng ta có thể mong đợi từ bất kỳ dịch vụ tương tự nào khác là như sau:

- Hosting cho các trang web của cổng của dự án, tại địa chỉ *project.sourceforge.net*, để công chúng có thể xem được. Các trang này có thể là tĩnh hoặc động (với CGI hoặc PHP), trong trường hợp đó chúng có thể sử dụng một cơ sở dữ liệu (MySQL). Chúng được đưa vào một cách trực tiếp thông qua các lệnh sao chép từ xa và có thể được quản lý bằng việc sử dụng các phiên tương tác đầu cuối từ xa (SSH).
- Một cách tùy ý không bắt buộc, một máy chủ ảo mà nó trả lời cho địa chỉ từ một miền có được một cách tách biệt, giống như là *www.project.org or cvs.project.org*.
- Nhiều bao nhiêu nhóm thảo luận bằng web và/hoặc danh sách thư là tùy ý theo ý kiến của người quản trị.
- Một dịch vụ tin nơi mà những người quản trị công bố những tiến bộ có liên quan tới dự án.
- *Những người giám sát* cho việc báo cáo và giám sát lỗi, các yêu cầu về hỗ trợ, các yêu cầu về những cải tiến hoặc tích hợp các miếng vá. Những người quản trị trao cho từng vấn đề một mức ưu tiên và chỉ định một lập trình viên để tìm ra giải pháp.
- Những người quản lý nhiệm vụ, tương tự như những người giám sát, mà họ cho phép các dự án phụ được xác định với một loạt các nhiệm vụ. Những nhiệm vụ này, bổ sung vào một mức ưu tiên, sẽ đưa ra một thời hạn chót. Cùng với thời gian, các lập trình viên được chỉ định cho những nhiệm vụ này có thể chỉ ra các phần trăm hoàn thành nhiệm vụ.
- Một CVS hoặc Subversion (phiên bản phụ) với các quyền truy cập ban đầu cho tất cả các lập trình viên.
- Dịch vụ tải lên và tải về đối với các gói phần mềm. Nó đăng ký các phiên bản đầu vào khi các bên sử dụng và quan tâm có thể nhận một thông báo khi điều này xảy ra. Hơn nữa, việc tải lên ban đầu có liên quan tới sự tạo ra của vài bản nhân bản khắp thế giới, tạo điều kiện thuận lợi cho sự phân phối.
- Dịch vụ cho việc xuất bản các tài liệu ở định dạng HTML. Bất kỳ ai cũng có thể đăng ký chúng, nhưng họ sẽ chỉ nhìn thấy sau khi được phê chuẩn bởi một người quản trị.

- Bản sao sao lưu cho việc phục hồi thảm họa, như khi ổ bị hỏng, không có lỗi của người sử dụng, hoặc như việc ngẫu nhiên xóa mất một tệp.
- Cơ chế tích hợp cho những hiến tặng của những người sử dụng, cho các dự án và cho SourceForge.

Một người sử dụng được xác thực sẽ có một trang cá nhân chứa tất cả các thông tin liên quan, như các dự án mà người sử dụng có liên quan, các mẫu themes hoặc các nhiệm vụ đang treo, cũng như các nhóm thảo luận và các tệp mà anh ta đã nói anh ta muốn giám sát. Hơn nữa, vì anh ta không phải chăm sóc trang cá nhân của anh ta, người sử dụng sẽ nhận được những thông báo tới thư điện tử của mình về những thứ anh ta muốn kiểm soát.

8.9.2 Những người kế thừa của SourceForge

Vào năm 2001, VA Software gần như phá sản, trong sự rung lắc toàn phần của khủng hoảng dotcom. Sau đó hãng đã công bố một phiên bản mới của phần mềm SourceForge của mình với một giấy phép không tự do, trong một mong muốn để cứu nguồn doanh thu bằng việc bán nó cho các công ty vì những phát triển nội bộ của họ. Cùng một lúc, hãng đã hạn chế các cơ chế mà đã cho phép một dự án sẽ được bỏ đấy để chuyển sang site khác. Cả 2 sự kiện này đã được coi như là một mối đe dọa mà hàng ngàn dự án được đặt trên SourceForge có thể bị mắc bẫy trong tay của một công ty duy nhất, mà có thể sử dụng nền tảng này cho việc đưa ra các phần mềm không tự do. Trước điều này và khả năng của site bị đóng cửa, con cháu của phiên bản tự do đã được phát triển và các cổng dựa trên nó đã được mở, đặc biệt là Savannah (http://savannah.gnu.org) [57], chuyên dụng cho dự án GNU và cho các chương trình khác với các giấy phép dạng *copyleft*, hoặc BerliOS (BerliOS: Người điều đình của Nguồn Mở) [13], đã hình thành như một điểm họp cho các lập trình viên và các công ty PMTD. Tuy nhiên, đây chỉ là một bước trong hướng phát triển một nền tảng phân tán và nhân bản, nơi mà không ai có sự kiểm soát tuyệt đối đối với các dự án (Savannah Thế hệ Tiếp theo, 2001) [98].

Một ví dụ khác của một hệ thống quản lý dự án PMTD là Launchpad (https://launchpad.net) [43], được sử dụng bởi Ubuntu cho việc phát triển từng phiên bản của phát tán này. Launchpad không phải là một kho cho mã nguồn, mà nó được thiết kế để đưa ra sự hỗ trợ cho việc giám sát mã nguồn, những sự việc xảy ra và các bản dịch. Để đạt được điều này nó sử dụng công cụ Malone đã được nhắc tới, mà nó cho phép những sự việc được định hướng lại tới từng kho mã nguồn của các module bị ảnh hưởng.

8.9.3 Các site và chương trình khác

Một cách tự nhiên, các hệ thống cộng tác từng được và tiếp tục được phát triển, và một số công ty để công việc kinh doanh của họ dựa trên việc duy trì và cung cấp dịch vụ cho các site này. Ví dụ, dự án Tigris (Tigris.org: Các công cụ Kỹ thuật PMNM) [64], mà nó không chỉ duy trì các dự án kỹ thuật PMTD, mà còn sử dụng một cổng cộng tác (SourceCast) được duy trì bởi một công ty dịch vụ (CollabNet), mà nó cũng duy trì các site riêng rẽ của các dự án, như OpenOffice.org. Các site mới đang nổi lên áp dụng các PMTD mới, như GForce (http://gforge.org) [30], được sử dụng bởi dự án Debian (http://alioth.debian.org) [5]. Chúng ta có thể thấy một sự so sánh chi tiết của nhiều site trong “So sánh

các site hosting tự do/nguồn mở (FOSPhost) có sẵn cho các dự án hosting ra bên ngoài từ những chủ nhân các dự án" [202].

9 Các trường hợp điển hình

> "GNU, mà nó nghĩa là 'GNU' không phải là Unix, là tên cho hệ thống phần mềm tương thích hoàn toàn với Unix mà tôi đang viết sao cho tôi có thể trao nó đi một cách tự do cho bất kỳ ai cũng có thể sử dụng nó. Một vài tình nguyện viên đang giúp tôi. Những đóng góp về thời gian, tiền bạc, các chương trình và trang thiết bị là rất cần thiết".
>
> Richard Stallman, "Tuyên ngôn của GNU" (1985).

Chương này đưa ra một nghiên cứu sâu hơn một số dự án PMTD về mặt ảnh hưởng trong thế giới PMTD, những kết quả đạt được, các mô hình quản lý, sự phát triển lịch sử, … Tất nhiên, số lượng các dự án mà chúng ta có thể thảo luận ở đây là nhỏ hơn nhiều so với tổng số các dự án PMTD (hàng chục ngàn), mà nó có nghĩa là chương này không nên nghĩ nó là toàn diện, cũng không bao giờ có thể là như vậy. Dù vậy, chúng ta hy vọng rằng các độc giả, đọc xong chương này, sẽ ít nhất có được một sự hiểu biết cơ bản về cách mà những lý thuyết mà chúng ta đã thảo luận qua cuốn sách này đã được đưa vào trong thực tế như thế nào.

Các dự án mà chúng ta đã chọn trải từ những ứng dụng mức thấp hơn, những ứng dụng mà sự tương tác nhiều hơn với hệ thống vật lý của các máy tính hơn là với người sử dụng, cho tới những môi trường làm việc được thiết kế cho người sử dụng đầu cuối. Chúng ta cũng đã đưa vào các dự án PMTD mà, về nguyên tắc, không phải là những dự án phát triển một cách nghiêm ngặt. Điều này chủ yếu áp dụng đối với những phát tán, mà chúng có xu hướng sẽ được sử dụng như các hệ thống tích hợp, vì chúng chủ yếu nắm một tập hợp rộng rãi nhưng có giới hạn các ứng dụng độc lập và việc sử dụng chúng để tạo ra một hệ thống mà trong đó mọi thứ tương tác được một cách có hiệu quả, bao gồm cả những lựa chọn cho việc cài đặt, cập nhật và xóa bỏ các ứng dụng, khi mà người sử dụng mong muốn.

Các dự án mức thấp nhất mà chúng ta sẽ đề cập tới là Linux, nhân của hệ điều hành tự do nổi tiếng nhất ngày nay và FreeBSD, mà nó kết hợp được nhân từ họ BSD với một loạt các ứng dụng và tiện ích được làm bởi các bên thứ 3. Các môi trường làm việc cho những người sử dụng đầu cuối mà chúng ta sẽ nghiên cứu sẽ là KDE và GNOME, mà chúng chắc chắn là phổ biến và được sử dụng rộng rãi nhất. Đối với các máy chủ, một trong những khía cạnh chính trong các hệ thống tự do, chúng ta sẽ đề cập tới là Apache, người dẫn đầu trong thị trường máy chủ WWW, trong chương này. Cũng như vậy, chúng ta sẽ giới thiệu Mozilla, một trong những máy trạm WWW (trên thực tế, còn hơn thế nữa) mà chúng ta có thể tin cậy dựa vào trong thế giới PMTD. Dự án cuối cùng mà chúng ta sẽ đề cập trong chương này là OpenOffice.org, một gói (*bộ*) IT cho văn phòng tự do.

Chúng ta nghĩ có thể phù hợp để nghiên cứu các chi tiết của 2 trong số những phát tán phổ biến nhất, Red Hat Linux và Debian GNU/Linux, và để so sánh các kích thước của chúng với các hệ thống được sử dụng rộng rãi khác, như Microsoft Windows hoặc Solaris. Cuối cùng, môi trường phát triển phần mềm đa ngôn ngữ Eclipse cũng sẽ được đưa vào.

Sau việc thảo luận các trường hợp điển hình khác nhau, chúng ta sẽ đưa ra một bảng chỉ ra những đặc tính quan trọng nhất của từng ứng dụng hoặc dự án. Một trong những yếu tố mà các độc giả có thể sẽ thấy ngạc nhiên nhất sẽ là các kết quả của những đánh giá về giá thành và khoảng thời gian và số lượng các lập trình viên được yêu cầu. Chúng ta đã có được các kết quả này bằng việc sử dụng các phương

pháp tiêu biểu được sử dụng trong lĩnh vực của kỹ thuật phần mềm, đặc biệt là mô hình đánh giá giá thành của phần mềm COCOMO. Mô hình COCOMO (*Kinh tế kỹ thuật của phần mềm*, 1981) [93] lấy số lượng các dòng mã lệnh như là thước đo ban đầu và sinh ra những ước lượng về tổng giá thành, thời gian phát triển và nỗ lực được yêu cầu để tạo ra phần mềm. COCOMO là một mô hình được thiết kế cho các quá trình tạo ra phần mềm "kinh điển" (mô hình phát triển thác nước đổ hoặc V) và cho các dự án kích thước trung bình hoặc phạm vi rộng; vì thế, các con số mà nó sẽ tạo ra cho một số các trường hợp mà chúng ta phân tích phải được nắm bắt với một số hạn chế. Trong bất kỳ trường hợp nào, thì các kết quả này cũng có thể giúp cho chúng ta ý tưởng về phạm vi tuyệt đối mà trong đó chúng ta đang làm việc và về số lượng của nỗ lực cố gắng mà có thể sẽ cần thiết để đạt được những kết quả hệt như vậy với một mô hình phát triển của PMSHĐQ.

Nói chung, đây là những ước lượng giá thành mà chúng là nổi bật nhất của tất cả các con số kết quả từ mô hình COCOMO. 2 yếu tố được tính tới trong đánh giá này: lương trung bình của một lập trình viên và *tổng chi phí*. Đối với việc tính toán giá thành ước lượng, lương trung bình cho một lập trình viên hệ thống toàn thời gian được lấy từ "Khảo sát về lương năm 2000" [235] từ năm 2000. *Tổng chi phí* sẽ là giá thành phải trả thêm mà tất cả các công ty phải trả sao cho sản phẩm có thể được tung ra, độc lập đối với lương được trả cho các lập trình viên. Điều này trải từ lương của các thư ký và đội marketing tới giá thành các máy photocopy, đèn chiếu sáng, trang bị phần cứng, … Để tổng kết, giá thành được tính bằng COCOMO là tổng giá thành mà một công ty có thể phải chịu để tạo ra phần mềm có kích cỡ được chỉ định và phải ghi nhớ rằng chỉ một phần của số tiền này các lập trình viên được nhận cho việc thiết kế phần mềm. Một khi điều này là yếu tố được đưa vào, thì giá thành không còn được coi là quá đáng nữa.

9.1 Linux

Nhân Linux là, không còn nghi ngờ gì, ứng dụng ngôi sao của PMTD, ở một mức độ nào đó, trong khi chỉ là một phần nhỏ của hệ thống, thì tên của nó được sử dụng để xác định toàn bộ. Hơn nữa, còn có thể nói rằng bản thân PMTD bị lẫn lộn với Linux trong nhiều trường hợp, mà là một sai lầm khá lớn, biết rằng có những PMTD mà chúng chạy trên các hệ thống không phải dựa trên Linux (trên thực tế, một trong những mục đích lớn nhất của phong trào này và của nhiều dự án PMTD là để tạo ra những ứng dụng có thể chạy trong nhiều môi trường).

Trong một lưu ý khác, còn có những ứng dụng mà chúng làm việc trong Linux và chúng thực sự không phải là PMTD (như là Acrobat Reader, trình đọc các tài liệu PDF sở hữu độc quyền, mà đối với nó cũng có một phiên bản cho Linux).

Lưu ý

Thực sự có các dự án khác nhau mà chúng tích hợp và phân phối các ứng dụng tự do chạy trên các hệ thống Windows, để tránh cho PMTD trở nên chỉ có liên quan tới các hệ thống Linux. Một trong những ứng dụng tiên phong trong lĩnh vực này (và là một ứng dụng mà có lẽ đã trở thành nổi tiếng và toàn diện nhất) là GNUWind, mà nó đã được phân phối trên những đĩa CD tự khởi động được với hơn 100 ứng dụng tự do cho các hệ thống Win32. Hầu hết các ứng dụng này cũng sẵn sàng trong các phát tán thông thường của GNU/Linux, mà đã làm cho GNUWin thành một công cụ tốt cho việc chuẩn bị chuyển dần dần và dễ dàng từ hệ thống Windows sang hệ thống GNU/Linux. Vào đầu năm 2007, có các hệ thống

tương tự khác sẵn sàng, như là WinLibre.

9.1.1 Lịch sử của Linux

Lịch sử của Linux là một trong những lịch sử nổi tiếng nhất trong thế giới của PMTD, có lẽ hầu như vì nó có những nét của một câu chuyện thần thoại hơn là những nét lịch sử của một chương trình máy tính. Vào năm 1991, một sinh viên Phần Lan có tên là Linus Torvalds đã quyết định rằng anh ta đã muốn học cách để sử dụng chế độ bảo vệ 386 trên một máy tính mà thu nhập ít ỏi của anh ta đã cho phép anh ta mua. Vào thời điểm đó, đã có một nhân trong hệ điều hành được gọi là Minix, được thiết kế cho các mục đích hàn lâm và để sử dụng trong các khóa học của trường đại học về các hệ điều hành; Nó vẫn còn được sử dụng ngày nay. Andrew Tanenbaun, một trong những giáo sư nổi tiếng nhất tại đại học này, là lãnh đạo của đội làm việc về sự phát triển của Minix, dựa vào các hệ thống Unix truyền thống. Minix là một hệ thống hạn chế, nhưng hoàn toàn có khả năng và được thiết kế tốt, và ở tại trung tâm của một viện hàn lâm rộng lớn và một cộng đồng các kỹ sư.

Minix đã có một giấy phép phân phối tự do và có thể được sử dụng cho các mục đích hàn lâm, nhưng nó đã được làm một cách độc lập, thường sử dụng các bản vá. Điều này có nghĩa rằng trong thực tế, đã có một phiên bản chính thức của Minix mà mọi người đã sử dụng và sau một loạt dài các bản vá mà đã được áp dụng sau này để có được các chức năng bổ sung.

Giữa năm 1991, Linus, khi này là một sinh viên Phần Lan vô danh, đã gửi một thông điệp cho nhóm tin Minix công bố rằng anh ta đang bắt đầu làm việc về một nhân hệ điều hành dựa trên Minix, từ đầu, viết lại mã nguồn. Khi đó, dù Linus đã không dứt khoát nói rằng anh ta định xuất bản nó với một giấy phép của PMTD, thì anh ta đã lưu ý rằng hệ thống mà anh ta định tạo ra sẽ không có những *rào cản* mà Minix đã có; điều này có lẽ chỉ ra rằng, còn chưa rõ đối với anh ta, và có lẽ không thực muốn thế, anh ta đã tiến hành bước đầu tiên hướng tới việc tạo ra cộng đồng được tập hợp xung quanh Minix vào lúc đó của anh ta.

Phiên bản 0.0.2, mà đề ngày từ tháng 10/1991, dù còn rất hạn chế, đã có thể chạy *thử* được trên các máy đầu cuối (terminal) và trình biên dịch GCC. Qua quá trình các tháng sau đó, số lượng các đóng góp từ bên ngoài đã gia tăng tới độ vào tháng 03/1992, Linus đã có thể xuất bản phiên bản 0.95, mà đã được thừa nhận rộng rãi là hầu như ổn định. Tuy nhiên, vẫn còn cả một con đường phải đi, trước khi ra phiên bản 1.0, mà nó thường được coi là phiên bản ổn định đầu tiên. Ví dụ, vào tháng 12/1993, phiên bản 0.99pl14 đã được xuất bản (có thể là phiên bản được sửa cho đúng thứ 14 của phiên bản 0.99); vào tháng 03/1994, Linux 1.0 cuối cùng đã ra đời. Vào lúc đó, Linux đã được xuất bản theo những điều khoản của giấy phép GPL; theo bản thân Torvalds, đây là một trong những quyết định tốt nhất từ trước tới nay mà ông đã làm, vì nó là cực kỳ hữu ích trong việc phân phối và phổ dụng nhân của ông. Trong "Sự tiến hóa trong PMNM: một trường hợp điển hình", [128] có một phân tích thấu đáo về sự tiến hóa của các phiên bản khác nhau của nhân Linux, tập trung vào phạm vi và tính module hóa.

Lưu ý

Sự kiện đáng kể khác trong biên niên sử của PMTD là tranh luận đã diễn ra vào cuối tháng 01/1992 trên nhóm tin Minix giữa Andrew Tanenbaun và Linus Torvalds. Tanenbaum, người có lẽ một chút khó chịu đối với thành công của Torvalds với "đồ chơi" của anh ta, đã tấn công Linux và Linus theo một cách

không cân xứng. Điểm cơ bản của anh ta là việc Linux là một hệ thống nguyên khối (nhân tích hợp tất cả các điều khiển và phần còn lại) và không phải là một hệ thống nhân vi mô – microkernel (nhân có một thiết kế theo module, mà có nghĩa là nó có thể nhỏ hơn nhiều và các module có thể được tải theo yêu cầu). Lý lẽ ban đầu có thể chỉ được đọc như nó đã xảy ra trong nhóm tín "tranh luận giữa Tanenbaum-Torvalds" [214].

9.1.2 Cách thức làm việc của Linux

Cách mà Torvalds đã làm là rất không thông thường khi đó. Sự phát triển chủ yếu dựa trên một danh sách thư[9]. Danh sách thư này là nơi mọi người không chỉ tranh luận, mà còn là nơi các phát triển cũng diễn ra. Và điều này là vì Torvalds cực kỳ sắc bén về việc có toàn bộ cuộc sống của dự án được phản ánh trên danh sách thư, mà nó giải thích vì sao anh ta có thể yêu cầu mọi người gửi các bản vá của họ cho danh sách này. Đối ngược với những gì một người có thể đã mong đợi (các bản vá được gửi như những tệp đính kèm), Linus đã thích có mã nguồn được gửi vào thân của thông điệp hơn sao cho anh ta và những người khác có thể bình luận trên mã nguồn đó. Trong mọi trường hợp, dù nhiều người có thể đưa ra những ý kiến của họ và gửi những bản sửa cho đúng hoặc các chức năng mới, thì lời cuối cùng luôn đi tới Linus Torvalds, người sẽ quyết định trên mã nguồn nào sẽ hợp nhất vào trong Linux. Ở một mức độ nào đó, điều này vẫn là cách mà nó làm việc trong năm 2007.

Lưu ý

Sự nhất quán của Linus Torvalds như một "kẻ độc tài nhân từ" đã tạo ra một số lượng lớn các chuyện cười trong dự án. Ví dụ, được nói rằng nếu một ý tưởng là đúng, thì nó phải được triển khai. Nếu nó là không đúng, thì cũng phải được triển khai. Kết quả tất yếu, vì thế, là các ý tưởng tốt không được sử dụng tí nào (mà không có mã nguồn, tất nhiên rồi). Lưu ý khác, nếu sự triển khai cài đặt là không được đúng, thì về bản chất phải cố nài. Một trường hợp nổi tiếng là việc Gooch, mà đối với anh ta thì Saint Job chỉ là một người học trò. Gooch được làm từ 146 miếng vá song song cho tới khi Linus cuối cùng đã quyết định tích hợp chúng vào nhánh chính thống của nhân.

Một chuyện khác về những ý tưởng sáng kiến của Torvalds là để phát triển 2 nhánh của nhân song song nhau: nhánh ổn định (số thứ 2 của phiên bản thường là chẵn, như là 2.4.18) và nhánh không ổn định (số thứ 2 của phiên bản là lẻ, như 2.5.12). Như mọi khi, Torvalds là người quyết định những gì sẽ đi vào nhánh nào (nhiều quyết định gây tranh cãi nhất có liên quan chính xác tới điểm này). Trong mọi trường hợp, Linux đã không lên bất kỳ kế hoạch phân phối nào trong khung thời gian cố định cả: nó sẽ là sẵn sàng khi nó sẵn sàng và trong thời gian chờ đợi thì chúng ta sẽ chỉ có mà chờ. Chắc chắn bây giờ, hầu hết các độc giả đã nhận thức được rằng quyết định về liệu hệ thống là sẵn sàng hay chưa sẽ được thực hiện bởi chỉ Linus.

Phương pháp phát triển được sử dụng trong Linux đã chứng minh là rất hiệu quả về mặt kết quả: Linux là rất ổn định và bất kỳ lỗi nào cũng sẽ được sửa cực kỳ nhanh (đôi khi trong vài phút), vì nó có hàng ngàn lập trình viên. Trong trường hợp này, khi có một lỗi, thì tính có thể là ai đó sẽ tìm thấy nó là rất cao, và nếu người mà tìm ra nó không có khả năng sửa nó, thì ai đó sẽ xuất hiện và sẽ đưa ra giải pháp rất nhanh. Để kết luận, điều này chỉ ra làm thế nào Linux có hàng ngàn người làm việc về sự phát triển

9 Thư điện tử của danh sách này là linuxkernel@vger.kernel.org. Các thông điệp lịch sử có thể được thấy tại http://www.uwsg.indiana.edu/hypermail/linux/kernel/.

của nó mỗi tháng, mà nó giải thích vì sao sự thành công của nó là không ngạc nhiên tí nào.

Tuy nhiên, phải lưu ý, rằng cách thức làm việc này là rất đắt giá nơi mà những tài nguyên được quan tâm. Không bình thường đối với việc sẽ có nhiều đề xuất qua lại đặc biệt cho một chức năng mới hoặc hàng tá các bản vá sẽ được nhận cho cùng một lỗi. Trong hầu hết các trường hợp, chỉ 1 trong những bản vá cuối cùng sẽ được đưa vào trong nhân, mà nó có nghĩa là những thứ còn lại của thời gian và nỗ lực được đặt vào các bản vá bởi những lập trình viên khác sẽ là vô ích. Mô hình phát triển của Linux, vì thế, là một mô hình mà nó làm việc rất tốt trong Linux nhưng không phải tất cả các dự án có thể tự cho phép chúng làm thế được.

9.1.3 Hiện trạng của Linux

Vào đầu năm 2007, Linux ở phiên bản 2.6, mà nó bao gồm những cải tiến được làm cho phiên bản 2.4, NUMA (Truy cập bộ nhớ không thống nhất, được sử dụng trong đa vi xử lý), các hệ thống tệp mới, những cải tiến cho giao tiếp trong các mạng không dây và các kiến trúc âm thanh (ALSA) và nhiều cải tiến khác (nếu bạn có quan tâm tới những chi tiết của những thay đổi trong các phiên bản trước, bạn có thể tra cứu "Thế giới tuyệt vời của Linux 2.6" [186]). Mô hình phát triển của Linux đã và đang có những thay đổi trong những năm gần đây. Dù danh sách thư của sự phát triển này vẫn còn là linh hồn của dự án, thì mã nguồn không còn phải truyền qua danh sách này nữa, là tất yếu. Một trong những thứ mà đã đóng góp cho điều này theo một nghĩa rộng là BitKeeper, một hệ thống sở hữu độc quyền mà nó thực hiện việc kiểm soát việc rà soát lại, được phát triển bởi công ty BitMover, tuân thủ một cách nghiêm khắc những khuyến cáo của Linus Torvalds. Việc sử dụng công cụ sở hữu độc quyền này đã làm nảy sinh ra nhiều tranh cãi, trong đó Linus đã thể hiện một lần nữa tính thực dụng của mình, vì đối với ông và nhiều người khác, thì hệ thống kiểm soát phiên bản CVS là lỗi thời. Những bất đồng đã đưa tới một kết thúc với sự phát triển của git, một hệ thống kiểm soát việc rà soát lại với những tính năng tương tự như BitKeeper mà nó hiện đang được sử dụng trong phát triển Linux. Đặc biệt hơn, quá trình phát triển của Linux tuân theo tôn ti trật tự hình chóp nón, trong đó các lập trình viên đề xuất các bản vá, chia sẻ thông qua thư giữa các mức, mà phải được phê chuẩn bởi mức tiếp sau cao hơn, được định hình bởi người kiểm soát và những người duy trì. Những người duy trì các hệ thống thứ cấp là ở một mức cao hơn, trong khi Linus Torvalds và Andrew Morton là ở mức đỉnh và có lời cuối cùng khi có liên quan tới sự chấp nhận các bản vá.

Để tổng kết lại, bảng sau đây đưa ra một bức trang X quang của dự án Linux, chỉ ra làm cách nào nó có hơn 5 triệu dòng mã lệnh và nó có thể vì thế được đưa vào trong số những dự án PMTD lớn nhất (cùng với Mozilla và OpenOffice.org). Đối với những đánh giá về thời gian nó có thể cần để thiết kế một dự án như vậy và số lượng trung bình các lập trình viên có thể là cần thiết, chúng ta phải lưu ý rằng thời gian chắc chắn là ít hơn nhiều so với thời gian mà Linux đã từng có. Mặt khác, điều này được bù đắp nhiều hơn bởi chi tiết sau, đưa ra rằng số lượng trung bình các lập trình viên làm việc toàn thời gian mà có thể cần thiết cho một dự án như vậy là cao hơn so với từ trước tới nay, sẵn có đối với Linux.

Lưu ý

Ước tính giá thành mà COCOMO chỉ ra là trong khoảng 215 triệu USD, một tổng số mà, nếu chúng ta đặt nó vào ngữ cảnh của các con số hàng ngày mà chúng ta có thể nghĩ được, thì có thể nhân đôi những

gì mà các câu lạc bộ bóng đá tốt nhất trả cho một ngôi sao bóng đá tuyệt vời.

Bảng 4. Phân tích về Linux

Website	http://www.kernel.org
Bắt đầu của dự án	Thông điệp đầu tiên trên news.comp.os.minix: Tháng 08/1991
Giấy phép	GNU GPL
Phiên bản được phân tích	2.6.20 (phiên bản ổn định vào ngày 20/02/2007)
Số dòng mã lệnh	5,195,239
Ước tính giá thành (Cơ bản theo COCOMO)	$215,291,772.00
Ước tính thời gian thiết kế (cơ bản theo COCOMO)	8.83 years (105.91 tháng)
Ước tính số lượng trung bình các lập trình viên (cơ bản theo COCOMO)	180.57
Số lượng áng chừng các lập trình viên	Được ước lượng là hàng ngàn (dù chỉ hàng trăm là đáng tin cậy [219)]
Các công cụ hỗ trợ phát triển	Danh sách thư và git

Thành phần của Linux về các ngôn ngữ lập trình chỉ ra một sự áp đảo rõ ràng về C, mà nó được coi là ngôn ngữ lý tưởng cho việc thiết kế các hệ thống tới hạn về tốc độ. Khi tốc độ là một yêu cầu nghiêm ngặt như vậy mà C còn chưa thể đạt được nó, thì ngôn ngữ assembly trực tiếp được sử dụng cho việc lập trình và điều này, như chúng ta có thể thấy, xảy ra khá thường xuyên. Nhược điểm của ngôn ngữ assembly, so với C, là việc nó không thật khả chuyển được. Mỗi kiến trúc có tập hợp các lệnh đặc biệt của nó, có nghĩa là nhiều mã nguồn được viết cho một kiến trúc trong ngôn ngữ assembly phải được chuyển sang các kiến trúc khác. Phạm vi ảnh hưởng của các ngôn ngữ còn lại, như được chỉ ra trong bảng đính kèm, là có giới hạn và chúng bị hạn chế đối với các chức năng cài đặt và các tiện ích phát triển. Phiên bản được phân tích cho cuốn sách này là Linux 2.6.20, được xuất bản vào ngày 20/02/2007 (không bao gồm bất kỳ bản vá nào sau đó).

Bảng 5. Các ngôn ngữ lập trình được sử dụng trong Linux

Ngôn ngữ lập trình	Số dòng lệnh	Phần trăm
C	4,972,172	95.71%
Assembler	210,693	4.06%
Perl	3,224	0.06%

Ngôn ngữ lập trình	Số dòng lệnh	Phần trăm
Yacc	2,632	0.05%
Shell	2,203	0.04%

9.2 FreeBSD

Như chúng ta đã nhắc tới trong chương này về lịch sử của PMTD, có những dạng khác của các hệ điều hành PMTD, khác với GNU/Linux phổ biến. Một họ của những "người thừa kế" của các phát tán từ Đại học Berkeley, California (Mỹ): các hệ thống dạng BSD. Hệ thống BSD lâu đời và nổi tiếng nhất là FreeBSD, mà nó được tạo ra vào đầu năm 1993, khi Bill Jolitz đã dừng xuất bản những cập nhật không chính thức cho 386BSD. Với sự trợ giúp của công ty Walnut Creek CDROM, mà sau này đã đổi tên thành BSDI, một nhóm các tình nguyện viên đã quyết định triển khai việc tạo ra hệ điều hành tự do này.

Mục đích chính của dự án FreeBSD là tạo ra một hệ điều hành mà nó có thể được sử dụng mà không có bất kỳ dạng bổn phận hoặc ràng buộc nào, nhưng lại có tất cả các ưu điểm về mã nguồn sẵn sàng và được xử lý một cách cẩn trọng để đảm bảo chất lượng của sản phẩm. Người sử dụng có sự tự do để làm bất kỳ thứ gì mà họ thích với phần mềm, hoặc bằng việc sửa đổi nó theo những mong muốn của họ hoặc bằng việc phân phối lại nó theo một dạng mở hoặc ngay cả theo một dạng đóng, theo những điều khoản mà họ muốn, có hoặc không có những sửa đổi. Như bản thân cái tên của nó chỉ ra, dự án FreeBSD, vì thế, dựa trên triết lý của các giấy phép BSD.

9.2.1 Lịch sử của FreeBSD

Phiên bản 1.0 đã xuất hiện vào cuối năm 1993 và đã dựa trên 4.3BSD Net/2 và 386BSD. 4.3BSD Net/2 đã có mã nguồn mà đã được tạo ra trong những năm 70, khi Unix đã được phát triển bởi AT&T, mà, hóa ra là, đã liên quan tới một loạt vấn đề về pháp lý mà đã chưa được giải quyết cho tới tận năm 1995, khi FreeBSD 2.0 được xuất bản mà không có mã nguồn gốc ban đầu được phát triển bởi AT&T mà được dựa vào 4.4BSD Lite, một phiên bản *nhẹ* của 4.4BSD (trong đó nhiều module đã bị hạn chế vì những lý do pháp lý, tách biệt khỏi thực tế là bản *chuyển sang* cho các hệ thống của Intel vẫn còn chưa hoàn chỉnh) và đã được tung ra bởi Đại học California.

Lịch sử của FreeBSD có thể không hoàn chỉnh nếu chúng ta bỏ quên nhắc tới các phát tán "chị em" của nó, NetBSD và OpenBSD. NetBSD đã xuất hiện như là phiên bản 0.8 vào giữa năm 1993. Mục tiêu chính là vì nó rất khả chuyển (dù ban đầu nó chỉ là một sự thích nghi cho i386); vì thế, phương châm của sản phẩm là: "Tất nhiên nó chạy NetBSD". OpenBSD nổi lên từ bộ phận của NetBSD với những khác biệt về triết lý (cũng như những khác biệt cá nhân) giữa các lập trình viên vào giữa năm 1996. Sự tập trung chủ yếu là vào an ninh và mật mã và họ nói rằng đây là hệ điều hành an toàn nhất đang tồn tại, dù, vì nó dựa trên NetBSD, nó cũng khả chuyển cao.

9.2.2 Sự phát triển trong FreeBSD

Mô hình phát triển được sử dụng bởi dự án FreeBSD dựa chủ yếu trên 2 công cụ: hệ thống kiểm soát phiên bản CVS và phần mềm theo dõi lỗi GNATS. Toàn bộ dự án dựa trên 2 công cụ này, như được khẳng định bởi thực tế là một tôn ti trật tự đã được tạo ra trên cơ sở của các công cụ này. Thực tế, chính *những người được ủy thác* (các lập trình viên với quyền truy cập ghi được vào kho CVS), những người có nhiều quyền nhất đối với dự án, hoặc trực tiếp hoặc gián tiếp thông qua lựa chọn của nhóm nòng cốt, như chúng ta sẽ thấy trong phần tiếp sau.

Bạn không buộc phải là một *người được ủy thác* để thực hiện các báo cáo lỗi trong GNATS, mà nó có nghĩa là bất kỳ ai muốn đều có thể báo cáo về một lỗi được. Tất cả những đóng góp (mở) trong GNATS sẽ được đánh giá bởi một *người được ủy thác*, người có thể chỉ định nhiệm vụ (được phân tích) cho *người được ủy thác* khác hoặc yêu cầu nhiều thông tin hơn từ người mà ban đầu đã thực hiện báo cáo lỗi này (ý kiến phản hồi). Có những tình huống trong đó lỗi đã từng được sửa cho một vài nhánh gần đây, mà sau đó chúng sẽ được chỉ định với tình trạng *treo*. Trong mọi trường hợp, mục đích là báo cáo này phải được *đóng*, một khi lỗi đã được sửa.

FreeBSD phân phối phần mềm của mình ở 2 dạng: một mặt, các bản khả chuyển, một hệ thống mà nó tải về các mã nguồn, biên dịch chúng và cài đặt ứng dụng lên máy tính cục bộ, và mặt khác, các gói, mà chúng đơn giản là các mã nguồn của các bản khả chuyển được biên dịch sẵn trước và, vì thế, ở dạng nhị phân. Ưu điểm quan trọng nhất của các bản khả chuyển đối với các gói là việc các bản khả chuyển cho phép người sử dụng thiết lập cấu hình và tối ưu hóa phần mềm cho máy tính của họ. Mặt khác, trong hệ thống các gói, vì chúng đã được biên dịch trước đó, nên cần ít thời gian hơn nhiều để cài đặt các phần mềm.

9.2.3 Qui trình ra quyết định trong FreeBSD

Ban lãnh đạo của FreeBSD, thường được gọi là đội nòng cốt, có trách nhiệm về việc xác định đường hướng của dự án và đảm bảo rằng các mục tiêu sẽ đạt được, cũng như việc dàn xếp được trong những trường hợp trong đó có những xung đột giữa những người được ủy quyền. Cho tới tháng 10/2000, nó từng là một nhóm kín, mà chỉ có thể được tham gia bởi một lời mời rõ ràng từ bản thân đội nòng cốt. Vào tháng 10/2000, các thành viên được bầu một cách định kỳ và dân chủ bởi những *người được ủy thác*. Qui định quan trọng nhất cho sự bầu cử của đội nòng cốt là như sau:

1. Những *người được ủy thác* đã làm ít nhất một sự *ủy thác* vào năm vừa qua có quyền bầu cử.
2. Ban Giám đốc sẽ được bầu mới mỗi 2 năm một lần.
3. Các thành viên của ban giám đốc có thể bị "trục xuất" bởi một cuộc bỏ phiếu của 2/3 tổng số những người được ủy thác.
4. Nếu số lượng các thành viên của ban giám đốc ít hơn 7, thì bầu cử mới sẽ được diễn ra.
5. Cuộc bầu cử mới sẽ diễn ra khi 1/3 các phiếu của những *người được ủy thác* đồng ý.
6. Bất kỳ thay đổi nào trong các qui định đòi hỏi 2/3 những *người được ủy thác* đồng ý.

9.2.4 Các công ty làm việc xung quanh FreeBSD

Có nhiều công ty đưa ra các dịch vụ và sản phẩm dựa trên FreeBSD, mà các danh sách của FreeBSD có trên các website của dự án. Trong trình bày về FreeBSD chúng ta sẽ học nhiều hơn về những khía cạnh đáng kể nhất: BSDI và Walnut Creek CDROM.

FreeBSD đã được sinh ra một phần nhờ quỹ của công ty BSDI vào năm 1991 bởi những người từ Nhóm Nghiên cứu các Hệ thống Máy tính CSRG (Computer Systems Research Group) của Đại học Berkeley, mà nó cung cấp hỗ trợ thương mại cho hệ điều hành mới này. Tách biệt khỏi phiên bản thương mại của hệ điều hành FreeBSD, BSDI cũng đã phát triển các sản phẩm khác, như là máy chủ Internet và một máy chủ cổng (gateway).

Walnut Creek CDROM đã được tạo ra với mục tiêu thương mại hóa FreeBSD như là sản phẩm cuối cùng, theo một cách mà nó có thể được coi là một phát tán ở dạng của những thứ đang tồn tại với GNU/Linux, nhưng là với FreeBSD. Vào tháng 11/1998, Walnut Creek đã tạo ra cổng FreeBSD Mall, mà nó có thể thương mại hóa tất cả các dạng sản phẩm dựa trên FreeBSD (từ việc tự phân phối tới áo t-shirt, tạp chí, sách, …), và công bố các sản phẩm của bên thứ 3 trên website của nó và cung cấp hỗ trợ chuyên nghiệp cho FreeBSD.

Vào tháng 03/2000, BSDI và Walnut Creek đã sát nhập theo tên BSDI để làm việc cùng nhau chống lại hiện tượng Linux, mà đã rõ ràng bỏ lại các hệ thống BSD nói chung và đặc biệt là FreeBSD, đứng trong bóng tối. Một năm sau, vào tháng 05/2001, Wind River đã mua phần chuyên tạo ra phần mềm của BSDI, với mong muốn rõ ràng để khuyến khích sự phát triển của FreeBSD cho sử dụng của hãng trong các hệ thống nhúng và các thiết bị thông minh được kết nối với mạng.

9.2.5 Hiện trạng của FreeBSD

Theo những dữ liệu mới nhất từ thăm dò mà Netcraft thực hiện định kỳ, thì số lượng các máy chủ chạy FreeBSD khoảng 2 triệu. Một người sử dụng mới mà mong muốn cài đặt FreeBSD có thể chọn giữa phiên bản 6.2 (mà có thể được coi như là phiên bản "ổn định") hoặc cao cấp hơn hoặc nhánh "phát triển". Trong khi phiên bản 6.2 cung cấp sự ổn định hơn, đặc biệt trong những lĩnh vực như đa xử lý bất đối xứng, mà từng được phát triển lại hoàn toàn trong các phiên bản mới hơn, thì cái sau cho phép những người sử dụng thụ hưởng những phát triển mới nhất. Cũng quan trọng phải nhớ trong đầu rằng những phiên bản được phát triển có xu hướng đưa vào các mã nguồn thử nghiệm, mà chúng khá ảnh hưởng tới tốc độ của hệ thống. Một trong những tính năng ngôi sao của FreeBSD là những gì được biết tới như là jails (nhà tù). Jail tối thiểu hóa thiệt hại mà có thể gây ra bởi một cuộc tấn công trên các dịch vụ cơ bản của mạng, như Sendmail cho thư điện tử hoặc BIND (Tên miền Internet của Berkeley) cho quản lý tên mền. Các dịch vụ được đặt trong một jail sao cho chúng chạy trong một môi trường bị cách ly. Jails có thể được quản trị bằng việc sử dụng một loạt các tiện ích được đưa vào trong FreeBSD.

9.2.6 Bức tranh X quang của FreeBSD

Như chúng ta đã nhắc tới trong phần cuối này, các chức năng của FreeBSD không bị hạn chế chỉ cho việc phát triển một nhân hệ điều hành, mà còn đưa vào sự tích hợp của vô số các tiện ích mà chúng được phân phối cùng nhau ở một dạng của các phát tán GNU/Linux. Thực tế là quá trình phát triển của FreeBSD được liên kết rất chặt chẽ với các phương tiện của hệ thống kiểm soát phiên bản CVS mà bằng việc nghiên cứu hệ thống này, chúng ta có thể có được một ý tưởng tốt về những gì cấu thành nên FreeBSD. Các con số chỉ ra bên dưới tương ứng với phân tích FreeBSD được thực hiện vào ngày 21/08/2003.

Một trong những khía cạnh thú vị của FreeBSD là việc các con số này rất tương tự với những con số trong KDE và GNOME: kích cỡ của phần mềm dễ dàng vượt quá 5 triệu dòng mã lệnh, số lượng các tệp khoảng 250,000 và tổng số lượng các lần *ủy thác* khoảng 2 triệu. Tuy nhiên, thú vị để quan sát rằng sự khác biệt chính giữa GNOME và KDE so với FreeBSD là độ tuổi của dự án. FreeBSD gần đây được 10 năm và nó là vào khoảng gần như gấp 2 lần lâu hơn các môi trường máy tính để bàn mà với nó chúng đang được so sánh. Mà kích thước chúng là tương tự, dù thực tế là giai đoạn phát triển phải là dài hơn, một phần vì thực tế là FreeBSD đã không lôi cuốn được nhiều lập trình viên. Có một danh sách khoảng 400 lập trình viên với quyền truy cập ghi tới CVS (những *người được ủy thác*), trong khi số lượng các lập trình viên được liệt kê trong sách hướng dẫn của FreeBSD là khoảng 1,000. Điều này giải thích vì sao hoạt động được đăng ký trong CVS của FreeBSD là thấp hơn so với trung bình (500 *ủy thác* trong một ngày) so với những gì được đăng ký cả trong GNOME (900) và KDE (1700, bao gồm cả những *ủy thác* tự động).

Chúng ta đã coi như hệ thống cơ bản của FreeBSD tất cả được đặt trong thư mục src/src của module gốc (root) của CVS. Hoạt động mà đã được đăng ký trong hệ thống cơ bản trong vòng 10 năm qua là lớn hơn nửa triệu lần *ủy thác*. Có hơn 5 triệu dòng mã lệnh, dù chúng ta phải nhớ rằng điều này không chỉ bao gồm nhân, mà nhiều tiện ích bổ sung nữa, bao gồm cả các trò chơi. Nếu chúng ta chỉ lấy nhân để tính (mà nó nằm trong thư mục con sys), thì phạm vi là 1.5 triệu dòng mã lệnh, chủ yếu được viết bằng ngôn ngữ C.

Thú vị để xem ước tính thời gian được đưa ra thế nào bởi COCOMO tương ứng chính xác cho thời gian thực của dự án FreeBSD, mặc dù ước tính số lượng trung bình các lập trình viên là cao hơn nhiều so với số lược thực sự. Chúng ta cũng phải chỉ ra rằng trong năm vừa qua, chỉ 75 *người được ủy thác* có hoạt động, trong khi COCOMO cho rằng hơn 10 năm phát triển, thì số lượng các lập trình viên phải là 235.

Cuối cùng, chúng ta phải nhớ, như chúng ta đã nhắc tới, rằng hoạt động chính của FreeBSD dựa xung quanh kho CVS và các hoạt động của hệ thống kiểm soát lỗi GNATS.

Bảng 6. Phân tích của FreeBSD

Website	http://www.FreeBSD.org
Bắt đầu dự án	1993

Giấy phép	Dạng của BSD
Phiên bản được phân tích	4.8
Số dòng mã lệnh	7,750,000
Số dòng mã lệnh (chỉ trong nhân)	1,500,000
Số lượng các tệp	250,000
Ước tính giá thành	$ 325,000,000
Ước tính thời gian thực hiện	10.5 năm (126 tháng)
Ước tính số lượng trung bình các lập trình viên	235
Số lượng các lập trình viên khoảng	400 được ủy thác (1,000 cộng tác viên)
Số lượng người được ủy thác tích cực trong năm cuối	75 (ít hơn 20% tổng số)
Số lượng người được ủy thác tích cực trong 2 năm cuối	165 (khoảng 40% tổng số)
Số lượng ủy thác trong CVS	2,000,000
Số lượng trung bình các ủy thác (tổng) trong 1 ngày	Khoảng 500
Công cụ hỗ trợ phát triển	CVS, GNATS, danh sách thư và site tin

C là ngôn ngữ chủ yếu trong FreeBSD và nó giữ một khoảng cách lớn hơn so với C++ so với các trường hợp khác mà chúng ta đã nghiên cứu trong chương này. Thú vị để lưu ý rằng số lượng các dòng mã lệnh trong ngôn ngữ assembly có trong FreeBSD, khớp với, về phạm vi, như trong Linux, dù chúng tương ứng với nhân chỉ là 25,000, tổng số. Để kết luận, chúng ta có thể nói rằng trong FreeBSD, các ngôn ngữ kinh điển hơn bên trong các phần mềm, C, Shell và Perl áp đảo, và rằng các ngôn ngữ khác mà chúng ta đã thấy trong các ứng dụng và dự án khác, như C++, Java, Python, đã không được tích hợp.

Bảng 7. Các ngôn ngữ lập trình được sử dụng trong FreeBSD

Ngôn ngữ lập trình	Số dòng lệnh	Phần trăm
C	7,080,000	92.0%
Shell	205,000	2.7%
C++	131,500	1.7%
Assembler	116,000	1.5%
Perl	90,900	1.20%
Yacc	5,800	0.75%

9.2.7 Các nghiên cứu hàn lâm về FreeBSD

Bất chấp việc chắc chắn là một dự án rất thú vị (chúng ta có thể thấy lịch sử của nó bằng việc phân tích hệ thống phiên bản, quay về 10 năm trước), thì FreeBSD đã không truyền được cảm hứng nhiều trong cộng đồng khoa học. Tuy nhiên, có một đội nghiên cứu mà đã chỉ ra mối quan tâm trong dự án FreeBSD, từ một loạt quan điểm ("Sự tích hợp dần dần và phi tập trung hóa trong FreeBSD") [149], mà nó đặc biệt tập trung vào cách mà các vấn đề tích hợp phần mềm được giải quyết theo một cách dần dần và phi tập trung.

9.3 KDE

Mặc dù nó là giải pháp đầu tiên của các môi trường máy tính để bàn thân thiện với người sử dụng, sự phổ biên của hệ điều hành Windows 95 vào giữa năm 1995 đã gây ra một thay đổi cơ bản theo cách mà những người sử dụng không đặc biệt gì cũng tương tác được với các máy tính. Từ những hệ thống một chiều của các dòng lệnh (các máy đầu cuối), phép ẩn dụ của môi trường máy tính để bàn 2 chiều đã ra đời, nơi mà chuột đã bắt đầu được sử dụng nhiều hơn là bàn phím. Windows 95, hơn là một sự đổi mới sáng tạo công nghệ, được tạo ra như là hệ thống mà bao trùm lên tất cả các môi trường cá nhân và văn phòng, thiết lập các chuẩn mà có thể được tuân theo trong tương lai (các qui định về kỹ thuật và xã hội mà, chúng ta vẫn còn, trong một số trường hợp, chịu đựng trong đầu thế kỷ 21 này).

Trước khi các hệ thống máy tính để bàn được tạo ra, mỗi ứng dụng đã quản lý sự thể hiện và cách thức của riêng nó trong việc tương tác với người sử dụng, một cách tự quản. Tuy nhiên, trên các máy tính để bàn, các ứng dụng phải có những thuộc tính và một sự thể hiện chung mà nó được chia sẻ bởi các ứng dụng, mà nó làm nhẹ đi áp lực lên người sử dụng, những người có thể *sử dụng lại cách thức tương tác* đã học khi sử dụng một ứng dụng này, để sử dụng cho một ứng dụng khác. Điều này cũng làm nhẹ đi áp lực lên các lập trình viên các ứng dụng, khi họ đã không phải làm việc với vấn đề về tạo ra những yếu tố tương tác bắt đầu từ số 0 (mà luôn là một nhiệm vụ phức tạp), mà có thể bắt đầu từ một khung công việc và các qui định được xác định trước.

9.3.1 Lịch sử của KDE

Những người theo Unix đã nhanh chóng nhận ra được thành công nổi bật của Windows 95 và, biết rằng các môi trường giống Unix đã không có các hệ thống trực giác trong khi vẫn là tự do, nên họ đã quyết định tiến hành công việc này. Dự án môi trường máy tính để bàn K - KDE (K Desktop Environment) đã ra đời từ nỗ lực này vào năm 1996; nó đã được thiết kế bởi Matthias Ettrich (người tạo ra LyX, một chương trình soạn thảo trong sắp chữ TeX) và các cao thủ khác. Dự án KDE đã đề xuất những mục tiêu sau:

- Để đưa ra cho các hệ thống như Unix với một môi trường thân thiện với người sử dụng mà, cùng một lúc, mở, ổn định, đáng tin cậy và mạnh.
- Để phát triển một tập hợp các thư viện cho việc viết các ứng dụng chuẩn trong một hệ thống đồ họa cho Unix X11.

- Để tạo ra một loạt các ứng dụng mà chúng có thể cho phép người sử dụng đạt được các mục tiêu của họ một cách có hiệu quả và hiệu lực.

Một sự tranh cãi đã nảy sinh khi các thành viên của dự án mới được tạo ra KDE đã quyết định sử dụng một thư viện hướng đối tượng được gọi là Qt, thuộc về hãng Trolltech của Nauy, mà nó đã không theo được bất kỳ giấy phép PMTD nào.

Hóa ra là, dù các ứng dụng của KDE đã được cấp phép theo GPL, thì chúng đã liên kết với thư viện này, mà có nghĩa là nó không thể phân phối lại chúng. Hậu quả là, một trong 4 quyền tự do được thiết lập bởi Richard Stallman trong Tuyên ngôn về PMTD đã bị vi phạm [117]. Vì thế từ phiên bản 2.0, Trolltech phân phối Qt theo một giấy phép đôi mà nó chỉ định rằng nếu ứng dụng mà sử dụng thư viện hoạt động theo GPL, thì giấy phép hợp lệ cho Qt là GPL. Nhờ điều này, một trong những tranh luận nóng và giận dữ nhất trong thế giới của PMTD đã kết thúc một cách hạnh phúc.

Lưu ý

Ban đầu, cái tên KDE có nghĩa là Môi trường Để bàn Kool, nhưng sau nó đã được thay đổi đơn giản thành Môi trường Để bàn K. Giải thích chính thức này là việc chữ cái K chỉ là đứng trước L, cho Linux, trong bảng chữ cái latin.

9.3.2 Sự phát triển của KDE

KDE là một trong ít dự án PMTD mà thường tuân theo một lịch trình tung ra phiên bản mới (chúng ta nhớ, ví dụ, sẽ có một phiên bản mới của Linux “khi nó là sẵn sàng”, trong khi, như chúng ta sẽ bàn luận sau này, GNOME luôn chịu sự chậm trễ đáng kể khi nói về việc đưa ra các phiên bản mới). Việc đánh số của các phiên bản mới tuân theo một chính sách được xác định tuyệt vời. Các phiên bản KDE có 2 con số cho phiên bản: con số cao nhất và 2 con số thấp hơn. Ví dụ, trong KDE 3.1.2, số cao nhất là số 3, trong khi 1 và 2 là các con số thấp hơn. Các phiên bản với cùng số cao có tính tương thích nhị phân, có nghĩa là không cần thiết phải biên dịch các ứng dụng. Cho tới nay, những thay đổi trong con số cao hơn đã xảy ra song song với những thay đổi trong thư viện của Qt, mà nó chỉ ra cách mà các lập trình viên đã muốn tận dụng ưu thế của các chức năng mới trong thư viện của Qt trong phiên bản sắp ra đời của KDE.

Ở những nơi mà các số thấp hơn có liên quan, thì các phiên bản với một số thấp hơn duy nhất là những phiên bản trong đó chúng đã đưa vào cả những tính năng mới và trong đó các lỗi mà đã được phát hiện, đã được sửa. Các phiên bản với một số thứ 2 thấp hơn, chỉ chứa những sửa lỗi. Ví dụ sau đây sẽ giải thích điều này tốt hơn: KDE 3.1 là một phiên bản thế hệ thứ 3 của KDE (Số cao hơn là 3) mà đối với nó với cùng các chức năng, nhưng với tất cả các lỗi mà đã được tìm thấy đã được sửa. KDE đã được xây dựng, ngay sau khi dự án đã bắt đầu, trong một hiệp hội được đăng ký tại Đức (KDE e.V.) và, vì thế, các bài viết về hiệp hội này có nghĩa là phải có một ban quản lý. Ảnh hưởng của ban quản lý này, đặc biệt nơi mà những quyên góp mà dự án nhận được được sự quan tâm. Để khuyến khích và phổ biến KDE, Liên đoàn KDE, mà nó bao gồm tất cả các công ty có quan tâm, đã được tạo ra, như chúng ta sẽ thảo luận dưới đây.

9.3.3 Liên đoàn KDE

Liên đoàn KDE là một nhóm các công ty và cá nhân từ KDE mà họ có mục đích xúc tác sự thúc đẩy, phân phối và phát triển của KDE. Các công ty và cá nhân mà tham gia vào Liên đoàn KDE không phải buộc phải tham gia trực tiếp vào trong sự phát triển của KDE (mặc dù các thành viên được khuyến khích để tham gia), mà đơn giản đại diện cho một khung công việc công nghiệp và xã hội mà là thân thiện đối với KDE. Các mục tiêu của Liên đoàn KDE là như sau:

- Khuyến khích, cung cấp và tạo điều kiện thuận lợi cho giáo dục chính thống và không chính thống về các chức năng, khả năng và các phẩm chất khác của KDE.
- Khuyến khích các đoàn thể, chính phủ, công ty và cá nhân sử dụng KDE.
- Khuyến khích các đoàn thể, chính phủ, công ty và cá nhân tham gia vào sự phát triển của KDE.
- Cung cấp kiến thức, thông tin, sự quản lý và định vị xung quanh KDE trong việc sử dụng và phát triển nó.
- Khuyến khích giao tiếp và cộng tác giữa các lập trình viên của KDE.
- Khuyến khích giao tiếp và cộng tác giữa các lập trình viên và công chúng nói chung thông qua các xuất bản phẩm, bài báo, website, hội họp, tham gia các hội nghị và triển lãm, thông cáo báo chí, phỏng vấn, các tư liệu và ủy ban thúc đẩy.

Các công ty tham gia trong Liên đoàn KDE chủ yếu là các nhà thiết kế các phát tán (SuSE, bây giờ là Novell, Mandriva, TurboLinux, Lindows và Hancom, một phát tán PMTD của Hàn quốc), các công ty phát triển (Trolltech và Klarälvdalens Datakonsult AB), cộng với người khổng lồ IBM và một công ty được tạo ra với mục đích thúc đẩy KDE (KDE.com). Trong tất cả những công ty này, chúng ta phải đặc biệt chú ý tới Trolltech, Novell và Mandriva Software, sự tham gia của họ là cần thiết và các mô hình kinh doanh của họ là có liên quan chặt chẽ tới dự án KDE:

- Trolltech là một công ty của Nauy có trụ sở ở Oslo mà nó phát triển Qt, thư viện mà có thể được sử dụng như một giao diện đồ họa cho người sử dụng và một giao diện lập trình ứng dụng API cho các lập trình viên, dù nó cũng có thể làm việc như một yếu tố được nhúng trong thiết bị số cá nhân PDA (như là Sharp Zaurus). Tầm quan trọng của dự án KDE đối với Trolltech là hiển nhiên bởi 2 yếu tố cơ bản trong chiến lược thương mại của hãng: một mặt, hãng nhận thức được KDE như phương pháp quảng cáo chính của hãng, khuyến khích sự phát triển máy tính để bàn và việc chấp nhận và triển khai những cải tiến hoặc sửa đổi được đề xuất; mặt khác, một số lập trình viên quan trọng nhất của KDE làm việc chuyên nghiệp cho Trolltech; ví dụ nổi tiếng nhất là việc bản thân Matthias Ettrich, người đã sáng lập ra bản thân công ty này. Sự tham gia của Trolltech trong dự án KDE không bị hạn chế một cách đặc biệt đối với thư viện Qt, như được chứng minh bởi thực tế rằng một trong những lập trình viên chính của KOffice, gói phần mềm văn phòng của KDE, hiện có một hợp đồng bán thời gian với họ.
- SuSE (bây giờ là một phần của Novell) đã luôn luôn chỉ ra sự ưa thích riêng đặc biệt cho hệ thống để bàn KDE, một phần nhờ vào yếu tố rằng hầu hết các lập trình viên của nó là những người Đức hoặc gốc Trung Âu, như bản thân hãng. SuSE, hiểu một thực tế rằng môi trường để

bàn tốt hơn và dễ dàng hơn mà phát tán của hãng đưa ra, sự triển khai lớn hơn của hãng và, vì thế, việc bán hàng và các yêu cầu hỗ trợ, đã luôn có một chính sách rất tích cực về ngân sách được phân bổ cho việc chuyên nghiệp hóa những vị trí chủ chốt bên trong dự án KDE. Như một ví dụ, người quản trị hiện hành của hệ thống kiểm soát phiên bản và 2 lập trình viên chính khác tất cả đều nhận lương của SuSE. Cũng vậy, trong nhân lực của SuSE có một tá các lập trình viên mà có thể bỏ một số thời gian làm việc của họ ra trong việc phát triển KDE.

- Phát tán Mandriva là một trong những người ủng hộ lớn nhất khác của KDE và một số lượng các lập trình viên chính làm việc cho hãng. Tình trạng tài chính của hãng, mà đã phá sản từ 2003, có nghĩa là hãng đã đánh mất ảnh hưởng trong ít năm vừa qua.

9.3.4 Hiện trạng của KDE

Sau lần xuất bản của KDE 3 vào tháng 05/2002, ý kiến chung là việc các máy để bàn là đi cặp đôi với các đối thủ cạnh tranh sở hữu độc quyền của chúng. Một số thành tựu tuyệt vời nhất của nó bao gồm sự kết hợp một hệ thống các thành phần (Kparts) mà chúng làm cho có khả năng nhúng được một số ứng dụng trong những ứng dụng khác (một mẩu của bảng tính KSpread trong trình xử lý văn bản Kword) và sự phát triển của DCOP, một hệ thống đơn giản các qui trình để giao tiếp với nhau, với sự xác thực.

DCOP là một cam kết của dự án mà nó đã hành động làm tổn hại tới các công nghệ của CORBA, một chủ đề gây tranh cãi rộng rãi bên trong thế giới của máy để bàn tự do, đặc biệt đối với GNOME, nơi mà nó đã quyết định rằng CORBA có thể được sử dụng. Lịch sử dường như đã đặt mỗi công nghệ vào chỗ của nó, như có thể được thấy từ đề xuất DBUS (một dạng cải tiến của DCOP) từ FreeDesktop.org, một dự án có quan tâm trong việc khuyến khích tính tương hợp và sử dụng của các công nghệ được kết hợp với nhau trong các máy để bàn tự do, mà nó, một cách trùng khớp, đã được dẫn dắt bởi một trong những *cao thủ* nổi tiếng nhất của GNOME.

Bảng sau tổng kết những đặc tính quan trọng nhất của dự án KDE. Các giấy phép mà dự án này chấp nhận phụ thuộc vào việc liệu chúng có là vì một ứng dụng hoặc một thư viện hay không. Các giấy phép cho thư viện đưa ra "tính mềm dẻo" tuyệt vời cho các bên thứ 3, nói một cách khác, chúng làm cho có khả năng đối với bên thứ 3 để tạo ra các ứng dụng sở hữu độc quyền mà những ứng dụng này được liên kết tới các thư viện.

Phiên bản mới nhất của KDE là, vào đầu năm 2007, phiên bản 3.5.6 và thế hệ 4, KDE 4, mà nó sẽ dựa trên Qt4, được dự kiến ra mắt vào giữa năm 2007. Sự thay đổi thế hệ liên quan tới nhiều nỗ lực về việc thích nghi cho phiên bản, mà là một nhiệm vụ nặng nề và tốn thời gian. Tuy nhiên, điều này không có nghĩa là các ứng dụng "cũ" sẽ không còn làm việc được nữa. Nói chung, để làm cho chúng vẫn còn làm việc được, thì các phiên bản cũ của các thư viện trong đó chúng dựa vào cũng sẽ được đưa vào, dù điều này có nghĩa là các phiên bản khác nhau về các thư viện phải được tải cùng một lúc vào bộ nhớ, với việc đảm bảo lãng phí các tài nguyên hệ thống. Các lập trình viên KDE coi hiệu ứng này như một phần vốn có của sự phát triển của dự án và, vì thế, như một tai hại nhỏ.

9.3.5 Hình ảnh X quang của KDE

Ở những nơi mà phạm vi của KDE được quan tâm, thì những con số mà chúng ta bây giờ sẽ thảo luận tương ứng với tình trạng của CVS vào tháng 08/2003, có nghĩa là chúng phải được lấy với những hạn chế thông thường mà chúng ta đã thảo luận, cộng với một thứ nữa: một số các module từng được sử dụng trong nghiên cứu này vẫn còn đang được phát triển và còn chưa thỏa mãn được các tiêu chí đối với một sản phẩm hoàn chỉnh. Điều này thực sự không có bất kỳ ảnh hưởng nào cho các mục tiêu của chúng ta, vì chúng ta có quan tâm hơn vào phạm vi của các kết quả hơn là các con số chính xác.

Mã nguồn được đưa vào trong CVS của KDE tổng cộng là 6 triệu dòng mã lệnh trong các ngôn ngữ lập trình khác nhau, như chúng ta sẽ chỉ ra bên dưới. Thời gian cần thiết để tạo ra KDE có thể khoảng 9 năm rưỡi, mà là lớn hơn 7 năm của dự án, và số lượng trung bình các lập trình viên làm việc toàn thời gian được ước lượng có thể là 200. Nếu chúng ta tính tới thực tế là KDE có khoảng 800 người với quyền truy cập ghi vào CVS vào năm 2003 (trong số đó thì một nửa là không hoạt động trong vòng 2 năm qua) và thực tế là số lượng các lập trình viên KDE với các hợp đồng toàn thời gian đã không lớn hơn 20 vào bất kỳ thời gian nào được đưa ra, thì chúng ta có thể thấy mức độ năng suất của KDE là cao, cao hơn nhiều so với ước tính được cung cấp bởi COCOMO.

> **Lưu ý**
>
> Một công ty mà đã muốn phát triển một sản phẩm có phạm vi như thế này bắt đầu từ số 0 có thể phải đầu tư hơn 250 triệu USD; vì những lý do để so sánh, con số tổng này có thể tương đương với một sự đầu tư của một nhà sản xuất ô tô trong việc tạo ra một nhà máy sản xuất mới tại Đông Âu hoặc những gì mà một công ty dầu khí nổi tiếng đang lên kế hoạch để chi tiêu cho 200 điểm bán xăng tại Tây Ban Nha.

Cũng thú vị để thấy rằng một phần lớn của nỗ lực, hầu như một nửa của nó được mở rộng trong sự phát triển của dự án KDE, có thể tương ứng với việc dịch giao diện người sử dụng và các tài liệu. Dù rất ít (khoảng 1,000) các dòng lập trình được quan tâm với với nhiệm vụ này, thì số lượng các tệp chuyên dụng cho mục đích này là 75,000 cho việc dịch (một số lượng mà gia tăng tới 100,000 nếu chúng ta đưa vào tài liệu ở các định dạng khác nhau), mà nó tạo nên hầu như 1/4 (1/3) trong số 310,000 tệp có trong CVS. Hoạt động được tổng hợp lại này của CVS là 1,200 ủy thác trong một ngày, có nghĩa là thời gian trung bình giữa các lần ủy thác là khoảng 1 phút[10].

Ở những nơi mà các công cụ, những vị trí có thông tin và những sự kiện hỗ trợ phát triển được quan tâm, thì chúng ta sẽ thấy rằng dãy các khả năng được đưa ra bởi KDE là rộng hơn nhiều so với những thứ như vậy được sử dụng trong Linux. Bỏ qua hệ thống kiểm soát phiên bản và các danh sách thư, thì KDE có một loạt các website cung cấp thông tin và tài liệu cả kỹ thuật và không kỹ thuật về dự án.

Cũng có một site thông tin trong số những site nơi mà các giải pháp mới được trình bày và các đề xuất được tranh luận. Tuy nhiên, site mới này, không thể được coi như một thay thế cho các danh sách thư, mà, như xảy ra với Linux, là nơi mà những tranh luận diễn ra và các quyết định được thực hiện và những chiến lược của tương lai được nghĩ ra; site thông tin này thực sự hơn là một điểm gặp gỡ cho những người sử dụng. Cuối cùng, KDE đã và đang tổ chức các cuộc gặp thường xuyên trong 3 năm,

10 Hai quan sát phải được thực hiện ở điểm này: một là khi một ủy thác có các tệp khác nhau được thực hiện, thì nó dường như đã có một ủy thác riêng rẽ cho từng tệp; hai là số lượng các ủy thác là một tổng số ước lượng, vì dự án này có một loạt các scripts mà chúng thực hiện các ủy thác một cách tự động.

trong đó các lập trình viên và các cộng tác viên gặp nhau khoảng một tuần để trình bày những đổi mới sáng tạo, sự phát triển, tranh luận mới nhất và để biết nhau và vui vẻ (không nhất thiết theo trật tự này).

Bảng 8. Phân tích KDE

Website	http://www.kde.org
Bắt đầu dự án	1996
Giấy phép (cho các ứng dụng)	GPL, QPL, MIT, Artistic
Giấy phép (cho các thư viện)	LGPL, BSD, X11
Phiên bản được phân tích	3.1.3
Số dòng mã lệnh	6,100,000
Số lượng các tệp (mã nguồn, tài liệu, ...)	310,000 tệp
Ước tính giá thành	$ 255,000,000
Ước tính thời gian thực hiện	9.41 năm (112.98 tháng)
Ước tính số lượng trung bình các lập trình viên	200.64
Số lượng các lập trình viên khoảng	Khoảng 900 người được ủy thác
Số lượng người được ủy thác tích cực trong 1 năm	Khoảng 450 (khoảng 50% tổng số)
Số lượng người được ủy thác tích cực trong 2 năm qua	Khoảng 600 (khoảng 65% tổng số)
Số lượng người dịch khoảng (tích cực)	Khoảng 300 người dịch cho hơn 50 ngôn ngữ (kể cả Esperanto).
Số lượng các ủy thác (của các lập trình viên) trong CVS	Khoảng 2,000,000 (con số ước lượng không bao gồm các ủy thác)
Số lượng các ủy thác (của những người dịch) trong CVS	Khoảng 1,000,000 (con số ước lượng không bao gồm các ủy thác tự động)
Số lượng trung bình các ủy thác (tổng số) trong 1 ngày	1,700
Các công cụ, tài liệu và các sự kiện hỗ trợ phát triển	CVS, các danh sách thư, website, site thông tin, các cuộc gặp hàng năm

Những nơi mà các ngôn ngữ lập trình được sử dụng trong KDE được quan tâm, thì C++ là áp đảo. Điều này chủ yếu vì thực tế đây là ngôn ngữ gốc của Qt, dù một nỗ lực lớn được mở rộng vào việc cung cấp các ràng buộc để cho phép những phát triển theo các ngôn ngữ khác. Chắc chắn, số lượng các dòng mã lệnh trong các ngôn ngữ thiểu số tương ứng với hầu hết những ràng buộc này, dù điều này không có nghĩa là chúng hoàn toàn không được sử dụng, như có nhiều dự án ngoài KDE mà sử dụng chúng.

Bảng 9. Các ngôn ngữ lập trình được sử dụng trong KDE

Ngôn ngữ lập trình	Số dòng mã lệnh	Phần trăm
C++	5,011,288	82.05%
C	575,237	9.42%
Objective C	144,415	2.36%
Shell	103,132	1.69%
Java	87,974	1.44%
Perl	85,869	1.41%

9.4 GNOME

Mục đích chính của dự án GNOME là để tạo ra một hệ thống máy để bàn cho người sử dụng đầu cuối mà nó là hoàn chỉnh, tự do và dễ sử dụng. Cũng vậy, ý tưởng này là để GNOME sẽ là một nền tảng rất mạnh cho các lập trình viên. Các chữ cái đầu của GNOME là viết tắt từ *GNU Network Object Model Environment, có nghĩa là Môi trường Mô hình Đối tượng Mạng GNU*. Từ tên của nó, chúng ta thấy rằng GNOME là một phần của dự án GNU. Hiện hành, tất cả các mã nguồn có trong GNOME phải được cấp phép theo GNU GPL hoặc GNU LGPL. Chúng ta cũng có thể thấy rằng các mạng và việc mô hình hóa hướng đối tượng là cực kỳ quan trọng.

9.4.1 Lịch sử của GNOME

Trong khi sự tự do của KDE vẫn còn đang được tranh luận, thì vào mùa hè năm 1997, như định mệnh muốn có, Miguel de Icaza và Nat Friedman đã gặp nhau tại Redmond trong một số cuộc hội thảo được tổ chức bởi Microsoft. Có lẽ là cuộc gặp này đã gây ra một sự chuyển biến đột ngột trong cả 2 người, gây nên trong sự tạo ra GNOME của Miguel de Icaza khi ông trở về Mexico (cùng với Federico Mena Quintero) và sự thán phục của ông về các công nghệ đối tượng phân tán. De Icaza và Mena đã quyết định tạo ra một môi trường mà có thể là một giải pháp thay thế cho KDE, khi họ đã hiểu rằng một sự triển khai lại của một thư viện sở hữu độc quyền có thể sẽ là một nhiệm vụ đã định trước thất bại. Và vì thế GNOME đã ra đời.

Kể từ thời điểm cổ xưa trong năm 1997, GNOME đã lớn lên dần dần và tiếp tục phát triển, với những xuất bản liên tục của nó. Phiên bản 0.99 đã được tung ra vào tháng 11/1998, nhưng phiên bản phổ biến thực sự đầu tiên, được phân phối thực tế với bất kỳ phát tán GNU/Linux nào, là GNOME 1.0, được tung ra vào tháng 03/1999. Phải lưu ý rằng kinh nghiệm của phiên bản ổn định đầu tiên của GNOME từng không phải là rất thỏa mãn, như nhiều người đã coi nó có đầy lỗi. Vì lý do này, GNOME October - tháng 10 (GNOME 1.0.55) được đối xử như là phiên bản đầu của môi trường máy để bàn GNOME mà nó thực sự là ổn định. Như chúng ta có thể thấy, với GNOME October, các lập trình viên đã không sử dụng phiên bản xuất bản được đánh số sao cho để tránh việc đưa vào một "cuộc đua của các phiên bản" chống lại KDE. GUADEC đầu tiên, hội nghị của châu Âu của những người sử dụng và các lập

trình viên GNOME, đã diễn ra tại Paris năm 2000 và việc trung khớp suýt chút nữa bị bỏ qua với sự tung ra phiên bản mới của GNOME, được gọi là GNOME April (GNOME tháng 4) đã được tung ra sau GNOME October, dù nó có thể được tha thứ vì việc giả thiết điều ngược lại). Vào tháng 10 của năm đó, sau một loạt tranh cãi qua nhiều tháng trong các danh sách thư khác nhau, thì Quỹ GNOME, mà chúng ta sẽ thảo luận trong các phần sau, đã được tạo ra.

GNOME 1.2 đã là một bước tiến về kiến trúc được sử dụng bởi GNOME, một kiến trúc mà nó đã tiếp tục được sử dụng trong GNOME 1.4. Kỷ nguyên này đã được đặc trưng bởi GUADEC thứ 2, mà nó diễn ra tại Copenhagen. Những gì đã bắt đầu như một cuộc họp nhỏ của một ít các cao thủ, đã trở thành một sự kiện lớn mà nó đã gây được sự chú ý của toàn bộ nền công nghiệp phần mềm.

Trong khi chờ đợi, lý do về sự tự do của KDE đã được giải quyết với sự thay đổi quan điểm của Trolltech, khi hãng đã kết thúc bằng việc cấp phép cho Qt theo một giấy phép đôi, mà nó dùng cho PMTD đối với các ứng dụng mà làm việc với PMTD. Hôm nay, không nghi ngờ gì rằng cả GNOME và KDE là những môi trường máy để bàn tự do, mà nó có nghĩa là chúng ta có thể nói rằng sự phát triển của GNOME đã khuyến khích sự tạo ra của không chỉ một môi trường máy để bàn tự do, mà là 2.

9.4.2 Quỹ GNOME

Vấn đề khó khăn nhất phải đưa ra khi bạn nghe về GNOME lần đầu tiên là việc tổ chức của nhiều hơn 1,000 cộng tác viên cho dự án này. Thật nghịch lý rằng một dự án với một cấu trúc mà có xu thế hướng tới sự hỗn loạn vô chính phủ, sẽ phải là thành công và đạt được các mục tiêu phức tạp này mà chỉ một số ít những người đa quốc gia trong khu vực công nghệ thông tin có khả năng đạt được. Dù GNOME đã được tạo ra với mục tiêu rõ ràng về việc cung cấp một môi trường thân thiện với người sử dụng và mạnh, mà đối với nó thì những chương trình mới có thể dần dần sẽ được bổ sung vào, nó sớm trở nên rõ ràng rằng cần thiết phải tạo ra một thực thể mà có thể có những trách nhiệm nhất định nào đó mà có thể cho phép chúng khuyến khích và thúc đẩy việc sử dụng, phát triển và phổ biến GNOME: hệ quả là, Quỹ GNOME đã được tạo ra vào năm 2000; tổng hành dinh của nó đặt tại Boston, Mỹ.

Quỹ GNOME là một tổ chức phi lợi nhuận và không phải là một nhóm các nhà công nghiệp; nó có các chức năng sau:

- Phối hợp các xuất bản phẩm.
- Quyết định những dự án nào là một phần của GNOME.
- Là người phát ngôn chính thức (cho báo chí và cho cả các tổ chức thương mại và phi thương mại) của dự án GNOME.
- Khuyến khích các hội nghị có liên quan tới GNOME (như GUADEC).
- Đại diện cho GNOME tại các hội nghị khác.
- Tạo ra các chuẩn kỹ thuật.
- Khuyến khích sử dụng và phát triển GNOME.

Bổ sung thêm, Quỹ GNOME nhận được tài chính cho việc khuyến khích và thúc đẩy các chức năng nêu trên, vì điều này đã không thể làm được theo một cách minh bạch trước khi quỹ này được tạo ra.

Hiện hành, Quỹ GNOME có một nhân viên làm việc toàn thời gian mà có trách nhiệm giải quyết tất cả các nhiệm vụ hành chính và tổ chức mà phải được thực hiện trong một tổ chức phi lợi nhuận mà nó tổ chức các cuộc họp và hội nghị thường xuyên. Nói chung, Quỹ GNOME được chia thành 2 ban lớn: ban quản lý và ban tư vấn.

Ban quản lý (*Ban Giám đốc*) được hình thành bởi 14 thành viên được bầu một cách dân chủ bởi các thành viên của Quỹ GNOME. Một mô hình "theo chế độ nhân tài" được tuân thủ, mà nó có nghĩa là, để trở thành một thành viên của Quỹ GNOME, một người phải đã cộng tác theo một cách nào đó với dự án GNOME. Sự đóng góp không nhất thiết phải liên quan tới mã nguồn; cũng có những nhiệm vụ mà đòi hỏi dịch, tổ chức, phổ biến, …, mà một người có thể thực hiện và sau đó áp dụng cho cơ chế thành viên của Quỹ GNOME, để có quyền biểu quyết. Vì thế, đây là những thành viên của Quỹ mà có thể tự đề cử họ cho ban giám đốc và đây là những thành viên mà, một cách dân chủ hóa, chọn ra những đại diện của họ trong ban này từ những người mà họ đã tự đề cử vào. Hiện nay, việc bầu cử là bằng thư điện tử. Nhiệm kỳ là thành viên của ban giám đốc là 1 năm, sau đó các cuộc bầu cử sẽ lại diễn ra.

Có một số điều chỉnh cơ bản cho việc đảm bảo tính minh bạch của ban giám đốc. Điều chính đáng chú ý nhất là sự hạn chế về số lượng thành viên có liên quan tới cùng một công ty, mà không thể vượt quá 4 nhân viên. Điều quan trọng để nhấn mạnh rằng các thành viên của ban giám đốc luôn là theo khả năng cá nhân của họ, và không bao giờ theo sự đại diện của một công ty. Tuy nhiên, sau một thảo luận dài, được thống nhất rằng điều này được đưa vào để tránh bất kỳ sự ngờ vực nào.

Ủy ban khác trong Quỹ GNOME là ủy ban tư vấn, mà nó không có quyền ra các quyết định nhưng nó phục vụ như một phương tiện cho việc giao tiếp với ủy ban quản lý. Nó được hình thành bởi các công ty thương mại làm việc trong nền công nghiệp phần mềm, cũng như các tổ chức phi thương mại. Hiện hành, các thành viên của nó bao gồm Red Hat, Novell, Hewlett-Packard, Mandrake, Sun Microsystems, Red Flag Linux, Wipro, Debian và FSF. Tất cả các công ty với hơn 10 nhân viên được yêu cầu trả phí để trở thành một phần của ban tư vấn.

9.4.3 Giới công nghiệp làm việc xung quanh GNOME

GNOME đã muốn tham gia vào giới công nghiệp một cách thực chất, theo một cách mà các công ty khác nhau đã tham gia rất tích cực vào sự phát triển của nó. Trong tất cả những công ty này, các trường hợp quan trọng nhất là của hãng Ximian, Eazel, RHAD Labs của Red Hat và, gần đây hơn, là Sun Microsystems. Chúng ta bây giờ sẽ mô tả, cho từng trường hợp, những động lực của các công ty này cũng như những đóng góp quan trọng nhất của họ cho môi trường máy để bàn GNOME:

- Ximian Inc. (ban đầu được gọi là Helix Inc.) là tên của một công ty được thành lập năm 1999 bởi Miguel de Icaza, đồng sáng lập của GNOME, và Nat Friedman, một trong những cao thủ của GNOME. Mục tiêu chính là để cùng mang các lập trình viên quan trọng nhất của GNOME dưới cùng một cái ô để tối đa hóa sự phát triển, mà nó giải thích vì sao không ngạc nhiên là các nhân viên hiện nay cũng như trước đây đã được đưa vào khoảng 20 lập trình viên tích cực nhất

của GNOME. Ứng dụng mà Ximian đặt nhiều nỗ lực vào nhất ngay từ đầu là Evolution, một hệ thống quản lý thông tin cá nhân kiểu như của Microsoft Outlook, mà nó đã đưa vào một trình thư điện tử cho máy trạm, chương trình nghị sự và một sổ địa chỉ liên lạc. Các sản phẩm mà Ximian đã bán từng là máy để bàn Ximian (một phiên bản của GNOME với nhiều mục tiêu cho doanh nghiệp hơn), Red Carpet (chủ yếu, do không bị hạn chế đối với một hệ thống phân phối phần mềm của GNOME) và cuối cùng là MONO (một sự triển khai lại của nền tảng phát triển .NET), dù dự án này sau này không có liên quan theo bất kỳ cách thức nào tới GNOME. Ximian cũng đã phát triển một ứng dụng mà nó cho phép Evolution tương tác được với Exchange 2000 Server. Ứng dụng này, trong khi khá nhỏ, nhưng rất gây tranh cãi vì nó đã được xuất bản với một giấy phép không tự do (rồi thì, trong năm 2004, thành phần này cũng đã được cấp phép như là PMTD). Vào tháng 08/2003, Novell, như một phần của chiến lược của công ty để đưa vào máy để bàn GNU/Linux, đã mua Ximian.

- Eazel đã được thành lập vào năm 1999 bởi một nhóm người mà đã quen làm việc cho Apple, với mục tiêu làm cho môi trường GNU/Linux dễ dàng như môi trường của Macintosh. Ứng dụng mà họ tập trung những nỗ lực của họ vào đó được gọi là Nautilus và nó dường như sẽ là trình quản lý tệp mà có thể chắc chắn cho về hưu trình Midnight Commander thần thoại, được phát triển bởi Miguel de Icaza. Sự thiếu hụt một mô hình kinh doanh và sự khủng hoảng của dotcom, mà đã gây ra rủi ro cho các nhà đầu tư để rút toàn bộ vốn đã được yêu cầu cho công ty này để triển khai công việc, làm cho Eazel phải công bố phá sản vào ngày 15/05/2001 và đóng cửa của mình. Nó đã có thời gian để tung ra Nautilus phiên bản 1.0 trước sự kiện này tuy nhiên, dù việc đánh số này đúng hơn là giả tạo, đưa ra rằng tính ổn định mà một người có thể mong đợi trong một phiên bản 1.0 không được thấy ở đâu cả. Hai năm sau sự phá sản của Eazel, chúng ta đã có khả năng thấy Nautilus đã phát triển như thế nào và đã trở thành một trình quản lý tệp hoàn chỉnh và có thể quản lý được, được tích hợp vào GNOME; điều này có nghĩa là câu chuyện của Eazel và Nautilus có thể được coi như một trường hợp của hệ biến hóa của một chương trình mà nó đã sống sót được sau sự biến mất của công ty đã tạo ra nó; thứ gì đó mà hầu như chỉ có thể trong thế giới của PMTD.

- Red Hat đã tạo ra Phòng thí nghiệm Phát triển Tiên tiến Red Hat, RHAD, với mục đích đảm bảo rằng máy để bàn GNOME có thể giành được tính thân thiện với người sử dụng và sức mạnh. Để đạt được điều này, Red Hat đã sử dụng nửa tá những cao thủ quan trọng nhất từ GNOME và đã trao cho họ sự tự do để phát triển bất kể thứ gì mà họ đã quyết định là phù hợp. Từ RHAD Labs chúng ta có ORBit, sự triển khai cài đặt của CORBA được sử dụng bởi dự án GNOME, được biết như là “nhanh nhất ở phương tây”. Khía cạnh quan trọng khác là công việc mà đã được triển khai trên phiên bản mới của GTK+ và trên hệ thống cấu hình của GNOME, GConf.

- Sun Microsystems đã có liên quan trong sự phát triển của GNOME ở giai đoạn sau, khi mà GNOME đã là một sản phẩm khá chín muồi vào tháng 09/2000. Dự định của Sun là để sử dụng GNOME như một hệ thống máy để bàn cho hệ điều hành Solaris. Vì thế hãng đã tạo ra một đội để làm việc với GNOME, mà những giá trị quan trọng nhất của nó bao gồm tính có thể sử dụng được và tính có thể truy cập được của GNOME. Vào tháng 06/2003, Sun đã công bố rằng hãng có thể phân phối GNOME 2.2 với phiên bản 9 của Solaris.

9.4.4 Hiện trạng của GNOME

GNOME, như vào đầu năm 2007, là ở phiên bản 2.18. Hầu hết các công nghệ chủ chốt mà nó được dựa vào đã chín muồi, như là bằng chứng từ số của phiên bản. Ví dụ, *CORBA broker* được sử dụng bây giờ là ORBit2, trong khi môi trường đồ họa và API, GTK +, còn đang trải qua những thay đổi được để lại từ kinh nghiệm tích tụ lại trong các phiên bản trước của GNOME. Một tính năng mới quan trọng là việc đưa vào một thư viện cho tính có thể truy cập được, được đề xuất bởi Sun, mà nó cho phép mọi người với các vấn đề về tính có thể truy cập được có khả năng sử dụng được môi trường GNOME. Một nhắc nhở đặc biệt cũng tới cho Bonobo, hệ thống các thành phần của GNOME. Bonobo để lại dấu ấn của nó trong một kỷ nguyên bên trong GNOME, trong khi chương trình quản lý thông tin cá nhân Evolution còn đang được phát triển. Tuy nhiên, thời gian đã chứng minh rằng những mong đợi được dấy lên bởi Bonobo là quá cao và rằng sự sử dụng lại của những nỗ lực được bỏ ra cho nó bằng việc sử dụng các thành phần của nó đã không rộng rãi như ban đầu được mong đợi.

Lưu ý

Thư viện ATK là một thư viện của các lớp trừu tượng mà chúng làm cho các ứng dụng có thể truy cập được. Điều này có nghĩa là mọi người với một số dạng khuyết tật (mù lòa, mù màu, những người có vấn đề về mắt, những người mà không thể sử dụng chuột, bàn phím, …) vẫn có thể sử dụng GNOME. Sự quan tâm của Sun về việc đảm bảo tính có thể truy cập được là do thực tế là nếu hãng muốn đưa ra các dịch vụ của hãng cho chính phủ Mỹ, thì hãng phải đáp ứng được một loạt chuẩn về tính có thể truy cập được. Họ đã nắm lấy công việc này thực sự nghiêm túc mà có ngay cả một lập trình viên mù trong đội phát triển GNOME làm việc tại Sun. Vào tháng 09/2002, kiến trúc về tính có thể truy cập được của GNOME đã được trao tặng giải thưởng Helen Keller Achievement.

9.4.5 Hình ảnh X quang của GNOME

Các dữ liệu và con số chỉ ra trong bảng 10 đưa chúng ta tới phần kết về trình bày GNOME của chúng ta. Các con số này tương ứng cho tình trạng của CVS của GNOME ngày 14/08/2003. Vào ngày đó, đã có hơn 9 triệu dòng mã lệnh được tổ chức trong kho của CVS được sở hữu bởi dự án GNOME. Mặc dù vậy thì thứ tự nhiên cơ bản nhất có thể so sánh GNOME với KDE, chúng ta phải cảnh báo cho các độc giả rằng những khác biệt về cách mà những dự án này được tổ chức làm cho điều này không thể khuyến cáo được nếu chúng ta muốn tiến hành sự so sánh theo những điều kiện như nhau. Ví dụ, điều này là do thực tế là CVS của GNOME đưa vào GIMP (một chương trình cho việc tạo và quản lý các hình ảnh), mà, bản thân chúng, đại diện cho hơn 660,000 dòng mã lệnh, hoặc thư viện GTK+, trong đó sự phát triển của GNOME được tập trung vào, và nó có 330,000 dòng mã lệnh. Nếu chúng ta bổ sung thực tế rằng kho CVS của GNOME là có khuynh hướng để mở các module mới cho các chương trình mà chúng ta có thể hiểu vì sao GNOME có nhiều dòng mã lệnh hơn là KDE, dù trẻ hơn 1,5 năm. Kho của GNOME chứa hơn 225,000 tệp, mà nó đã được bổ sung và sửa đổi hầu như 2 triệu lần (xem số lượng các *ủy thác* một số hàng bên dưới, trong bảng).

Lưu ý

Một công ty mà muốn tạo ra phần mềm có kích cỡ như phần mềm của GNOME, có thể phải ký hợp

đồng trung bình với khoảng 250 lập trình viên trong hơn 11 năm, để có được một sản phẩm với một sự mở rộng tương tự, theo mô hình của COCOMO, được sử dụng trong khắp chương này. Giá thành có liên quan có thể khoảng 400 triệu USD, một con số tương đương cho những gì mà một công ty điện thoại được thiết lập tốt đã đầu tư trong năm 2003 để tăng cường khả năng mạng của nó, hoặc tương tự với con số mà một hãng sản xuất ô tô có thể trả để mở một nhà máy sản xuất tại Barcelona. Các nguồn tài nguyên con người của GNOME bao gồm hầu như là 1,000 lập trình viên với quyền truy cập ghi tới hệ thống kiểm soát soát sửa lại CVS, đối với nó hầu như 20 người làm việc cho GNOME một cách chuyên nghiệp (toàn thời gian và bán thời gian). Trong số này, chỉ 25% là tích cực trong năm vừa qua và 40% từng tích cực trong vòng 2 năm qua. Số lượng trung bình các ủy thác trong một ngày, được đăng ký kể từ khi dự án được bắt đầu gần như là 1,000. Các công tụ hỗ trợ phát triển được sử dụng bởi dự án GNOME về cơ bản là y hệt như những công cụ được sử dụng trong dự án của KDE, và vì thế chúng ta sẽ không đi vào chúng trong phần này.

Bảng 10. Phân tích của GNOME

Website	http://www.gnome.org
Bắt đầu của dự án	Tháng 09/1997
Giấy phép	GNU GPL and GNU LPGL
Phiên bản được phân tích	2.2
Số dòng mã lệnh	9,200,000
Số lượng các tệp (mã nguồn, tài liệu, ...)	228,000
Ước tính giá thành	$ 400,000,000
Ước tính thời gian thực hiện	11.08 năm (133.02 tháng)
Ước tính số lường trung bình các lập trình viên	Khoảng 250
Số lượng các dự án phụ	Hơn 700 modules trong CVS
Số lượng các lập trình viên khoảng	Gần 1,000 với quyền truy cập ghi CVS
Số lượng người được ủy thác tích cực trong năm qua	Khoảng 500 (khoản 55% của tổng số)
Số lượng người được ủy thác tích cực trong 2 năm qua	Khoảng 700 (75% của tổng số)
Số lượng các ủy thác trong CVS	1,900,000
Số lượng trung bình các ủy thác (tổng) trong 1 ngày	Khoảng 900
Các công cụ hỗ trợ phát triển	CVS, các danh sách thư, website, site tin, gặp mặt hàng năm

Trong khi trong KDE, thì C++ là không nghi ngờ gì là ngôn ngữ được sử dụng rộng rãi nhất, thì trong GNOME, ngôn ngữ đó là C. Trong GNOME, như xảy ra trong KDE, điều này là vì thực tế là thư viện chính được viết trong C, có nghĩa là ngôn ngữ tự nhiên là C, trong khi các lập trình viên mong muốn sử dụng các ngôn ngữ khác phải chờ cho những ràng buộc tương xứng xuất hiện. Ngôn ngữ tiên tiến nhất ràng buộc trong GNOME là ngôn ngữ mà nó được đưa vào trong gnome - , mà nó là một thứ khác C+

+, nó giải thích vì sao không ngạc nhiên đó là ngôn ngữ thứ 2 trong sự phân loại. Perl luôn được chấp nhận một cách rộng rãi bên trong cộng đồng GNOME và là một ví dụ của thực tế rằng trong GNOME có khả năng lập trình bằng nhiều ngôn ngữ. Tuy nhiên, sự triển khai cài đặt của nó không được rộng rãi như có thể như được mong đợi nhưng nó khá là rộng rãi hơn là Shell. Mặt khác, Python và Lisp cũng được chấp nhận khá rộng rãi trong GNOME, như được minh chứng bởi tầm quan trọng tương đối của sự phân loại này, trong khi Java đã không thực sự được cất cánh có lẽ là do một liên kết không hoàn chỉnh.

Bảng 11. Các ngôn ngữ lập trình được sử dụng trong GNOME

Ngôn ngữ lập trình	**Số dòng mã lệnh**	**Phần trăm**
C	7,918,586	86.10%
C++	576,869	6.27%
Perl	199,448	2.17%
Shell	159,263	1.73%
Python	137,380	1.49%
Lisp	88,546	0.96%

9.4.6 Các nghiên cứu hàn lâm về GNOME

Các nghiên cứu quan trọng nhất về GNOME trong khu vực hàn lâm là 2 cái sau: “Những kết quả từ nghiên cứu kỹ thuật phần mềm trong các dự án phát triển nguồn mở có sử dụng các dữ liệu công cộng” [158] và “Sự tiến hóa của GNOME” [123].

- [158] là một trong những nghiên cứu ở phạm vi rộng đầu tiên về phần mềm trong khu vực của PMTD. Các tác giả của nghiên cứu này đã tận dụng ưu thế thực tế là các chi tiết của sự phát triển thường truy cập được một cách công khai để đo đếm những nỗ lực và so sánh chúng đối với các mô hình ước tính giá thành, và các đo đếm nỗ lực và thời gian theo truyền thống. Một trong những mô hình kinh điển với nó họ đã so sánh chúng là mô hình được sử dụng trong chương này, mô hình COCOMO.
- [123] đi qua một cách vắn tắt các mục tiêu của GNOME và lịch sử ngắn gọn của nó, cũng như việc sử dụng công nghệ của dự án GNOME.

9.5 Apache

Máy chủ Apache HTTP là một trong những ứng dụng ngôi sao của thế giới PMTD, vì nó là máy chủ web mà được sử dụng rộng rãi nhất, theo khảo sát thời gian thực của Netcraft (http://news.netcraft.com/archives/2003/08/01/august_2003_web_server_survey.html) [167]. Ví dụ, vào tháng 05/1999, 57% các máy chủ web đã làm việc với Apache, trong khi vào tháng 05/2003, số phần trăm này đã gia tăng lên tới 68% . Apache là sẵn sàng cho tất cả các dạng Unix (BSD,

GNU/Linux, Solaris...), Microsoft Windows và các nền tảng thiểu số khác.

9.5.1 Lịch sử của Apache

Vào tháng 03/1989, Tim Berners Lee, một nhà khoa học Anh mà đã làm việc tại CERN (Thụy Sĩ) đã đề xuất một phương pháp mới cho việc quản lý số lượng khổng lồ các thông tin từ các dự án của CERN. Phương pháp này có thể là một mạng của các tài liệu được siêu liên kết (siêu văn bản, như Ted Nelson đã gọi nó từ năm 1965); và WWW đã ra đời. Tuy nhiên, nó chưa thể cho tới tận tháng 11/1990 mà phần mềm WWW đầu tiên đã được hé lộ: một gói được gọi là World Wide Web đã đưa vào một trình duyệt với một giao diện đồ họa và một trình soạn thảo WYSIWYG ("những gì bạn nhìn thấy là những gì bạn có"). Hai năm sau, danh sách của các máy chủ WWW đã xấp xỉ 30, bao gồm NCSA HTTPd.

Lịch sử thực tế của Apache đã bắt đầu khi Rob McCool rời NCSA vào tháng 03/1995. Apache 0.2 có thể đã ra đời vào ngày 18/03/1995, dựa trên máy chủ NCSA HTTPd 1.3, được xây dựng bởi bản thân Rob McCool trong khi ông còn ở NCSA. Trong những tháng đầu tiên đó, Apache từng là một bộ sưu tập của các bản vá được áp dụng cho máy chủ của NCSA, cho tới tận khi Robert Thau tung ra Shambhala 0.1, một triển khai lại hầu như hoàn toàn mà nó đã đưa vào API cho các module mà sau này hóa ra là sẽ thành công.

> **Lưu ý**
>
> Tên của dự án Apache là dựa trên triết lý của sự phát triển và tổ chức của nó. Như là trường hợp của bộ lạc Apache, các lập trình viên của Apache đã quyết định rằng phương pháp tổ chức của họ phải dựa trên những giá trị của các lập trình viên trong sự so sánh với phần còn lại của cộng đồng Apache. Tuy nhiên, có một chuyện thần thoại mà nó đã lan truyền nói rằng tên Apache thực sự tới từ thực tế là trong những giai đoạn đầu, nó đơn giản là một máy chủ được vá của NCSA, hoặc *một máy chủ patchy* (vá).

Phiên bản ổn định đầu tiên của Apache đã không xuất hiện cho tới tháng 01/1996, khi Apache 1.0 đã được tung ra, mà nó đã đưa vào việc tải các module trong chế độ thử thời gian thực, cũng như các chức năng thú vị khác. Những tháng đầu của năm đó đã là đặc biệt thành công cho dự án này, khi phiên bản 1.1, mà đã có các module xác thực mà chúng có thể được kiểm tra đối với các cơ sở dữ liệu (như MySQL) đã được tung ra chỉ 2 tháng sau. Từ thời điểm đó cho tới ngày hôm nay, những sự kiện quan trọng nhất cho dự án này là sự giới thiệu của sự tuân thủ toàn phần với chuẩn HTTP 1.1 (được đưa vào tháng 04/1997 trong Apache 1.2), sự đưa vào của nền tảng Windows NT (mà nó đã bắt đầu vào tháng 07/1997 với các phiên bản thử nghiệm của Apache 1.3), sự thống nhất của các tệp cấu hình trong một tệp duy nhất (mà nó đã không xảy ra cho tới tháng 10/1998, trong Apache 1.3.3) và sự tung ra, vẫn còn trong giai đoạn thử nghiệm, của thế hệ tiếp sau của Apache, Apache 2.

Trong khi chờ đợi, vào tháng 06/1998, IBM đã quyết định rằng, thay vì việc phát triển HTTP của riêng hãng, hãng có thể sử dụng Apache như là động cơ của sản phẩm WebSphere của hãng. Điều này đã được hiểu như một sự chứng thực cực lớn cho dự án Apache từ Big Blue và cho PMTD nói chung, dù nó có thể là cần thiết phải sửa đổi giấy phép gốc của Apache một chút để làm cho điều này làm việc được.

9.5.2 Sự phát triển của Apache

Máy chủ Apache HTTP là dự án chính trong số nhiều dự án mà Quỹ Phần mềm Apache (Apache Software Foundation) quản lý. Thiết kế theo module của Apache đã làm cho nó có khả năng phục vụ cho một loạt các dự án vệ tinh, dựa vào quanh Apache, một số trong số đó còn lớn hơn cả bản thân Apache. Ví dụ, máy chủ Apache HTTP chứa nhân của hệ thống với các chức năng cơ bản, trong khi các chức năng bổ sung được cung cấp bởi các module khác. Các module nổi tiếng nhất là mod_perl (một trình biên dịch ngôn ngữ script của Perl được nhúng vào máy chủ web) và Jakarta (một máy chủ ứng dụng mạnh). Trong các đoạn dưới, chúng ta sẽ chỉ mô tả quá trình phát triển được tuân thủ đối với máy chủ HTTP, mà không tính tới các module khác mà chúng có thể có những quá trình tương tự hoặc không.

Sự phát triển của máy chủ Apache HTTP dựa trên công việc của một nhóm nhỏ các lập trình viên được gọi là nhóm Apache. Nhóm Apache được hình thành từ các lập trình viên mà họ đã làm việc cùng nhau trong dự án này một thời gian dài, thường hơn 6 tháng. Lập trình viên mà được mời bởi một thành viên khác để tham gia vào nhóm Apache, được biểu quyết bởi tất cả các thành viên khác. Trong những giai đoạn đầu, nhóm Apache có 8 lập trình viên; con số này đã gia tăng sau đó thành 12 và hiện có 25 thành viên.

Nhóm Apache có trách nhiệm cho sự phát triển của máy chủ web và, do đó, đối với những quyết định đặc biệt về sự phát triển tại bất kỳ thời điểm nào. Điều quan trọng để phân biệt nhóm Apache với các lập trình viên trong nhóm cốt lõi, mà nó là tích cực mọi thời gian. Bản chất tự nhiên tự nguyện của công việc đã được thể hiện bởi hầu hết các lập trình viên làm cho có lẽ tất cả mọi người tạo nên nhóm Apache cũng sẽ tích cực mọi thời gian, mà nó có nghĩa là cốt lõi được xác định như là những người có thể chăm sóc cho các nhiệm vụ tại Apache trong một giai đoạn nhất định nào đó. Nói chung, các quyết định phải được thực hiện bởi các lập trình viên thuộc về nhóm cốt lõi được hạn chế đối với việc biểu quyết để đưa hay không mã nguồn vào, dù trong thực tế điều này được dành chỉ cho những thay đổi và những câu hỏi về thiết kế ở phạm vi rộng. Một lưu ý nữa, là họ thường có quyền truy cập ghi tới kho CVS, có nghĩa là họ hành động như những người lính gác cho mã nguồn đầu vào, đảm bảo rằng nó đúng và có chất lượng tốt.

9.5.3 Hình ảnh X quang của Apache

Các con số bên dưới tương ứng cho phiên bản máy chủ Apache HTTP mà đã sẵn sàng cho việc tải về từ máy chủ CVS của dự án Apache vào ngày 18/04/2003. Không có con số các module mà dự án Apache có, được lấy để tính ở đây. Nhưng chúng ta sẽ thấy, dự án Apache khá nhỏ so với các trường hợp được nghiên cứu khác trong chương này. Dù điều này đã được nhắc tới, điều quan trọng phải nhấn mạnh tới tính module hóa của Apache, mà có những ưu điểm đặc biệt sau: nhân là nhỏ và có thể quản lý được. Kho CVS của dự án Apache, mà nó chứa nhân của máy chủ web và nhiều module bổ sung, có hơn 4 triệu dòng mã lệnh, một con số mà khá là thấp so với của các dự án như KDE và GNOME.

Phiên bản 1.3 của Apache đã có ít hơn 85,000 dòng mã lệnh; theo mô hình COCOMO, điều này có thể đã đòi hỏi công việc của trung bình 20 lập trình viên làm việc toàn thời gian cho 1 năm rưỡi. Giá thành tổng cộng của dự án có thể, vào lúc đó, xấp xỉ khoảng 4 triệu USD. Để chuẩn bị cho máy chủ web

Apache, tới 60 ủy thác khác nhau có thể đã cần thiết, trong khi số lượng các lập trình viên cung cấp đầu vào, theo tính toán, có thể khoảng 400.

Bảng 12. Phân tích của Apache

Website	http://www.apache.org
Bắt đầu của dự án	1995
Giấy phép	Giấy phép PMTD Apache
Phiên bản được phân tích	2.2.4
Số dòng mã lệnh	225,065
Số lượng các tệp	2,807
Ước tính giá thành	$ 7,971,958
Ước tính thời gian thực hiện	2.52 năm (30.27 tháng)
Ước tính số lượng trung bình các lập trình viên	23.4
Số lượng các lập trình viên khoảng	60 ủy thác (400 lập trình viên)
Các công cụ hỗ trợ phát triển	CVS, danh sách thư, hệ thống báo cáo lỗi

Apache 1.3 được viết hầu như hoàn toàn bằng ngôn ngữ C và không có bất kỳ ngôn ngữ lập trình nào khác, đặc biệt nếu chúng ta tính tới thực tế là hầu hết các dòng lệnh được viết trong ngôn ngữ thứ 2, Shell, tương ứng với các tệp cấu hình và hỗ trợ sự biên dịch.

Bảng 13. Các ngôn ngữ lập trình được sử dụng trong Apache

Ngôn ngữ lập trình	Số dòng mã lệnh	Phần trăm
C	208,866	92.80%
Shell	12,796	5.69%
Perl	1,649	0.73%
Awk	874	0.39%

9.6 Mozilla

Dự án Mozilla làm việc về một tập hợp các ứng dụng tích hợp cho Internet, mà là tự do và đa nền tảng, và các sản phẩm đáng chú ý nhất là trình duyệt web Firefox và trình thư điện tử và tin cho máy trạm Thunderbird. Tập hợp này cũng được thiết kế như một nền tảng cho việc phát triển các ứng dụng khác, mà có nghĩa là có nhiều trình duyệt mà sử dụng Gecko, máy HTML của Mozilla (như là Galeon).

Dự án này được quản lý bởi Quỹ Mozilla, một tổ chức phi lợi nhuận mà nó tạo ra các PMTD và là "chuyên để gìn giữ sự lựa chọn và khuyến khích đổi mới sáng tạo trên Internet". Vì những lý do này,

các sản phẩm của Mozilla dựa trên 3 nguyên tắc: chúng phải là PMTD, tôn trọng các chuẩn và khả chuyển cho các nền tảng khác.

9.6.1 Lịch sử của Mozilla

Lịch sử của Mozilla là dài và xoáy nhưng cũng rất thú vị, khi nó cho phép chúng ta đi theo lịch sử của bản thân WWW. Lý do cho điều này là việc nếu chúng ta dõi theo tất cả những người và cơ quan mà đã có liên quan trong sự phát triển của Mozilla, thì chúng ta sẽ tới điểm bắt đầu của Internet, với sự tung ra trình duyệt Internet hoàn chỉnh đầu tiên.

Đó là trường hợp với người tiền nhiệm của Apache, nó là NCSA nơi mà trình duyệt Internet đầu tiên hoàn chỉnh, Mosaic, đã "được ra đời" vào năm 1993. Nhiều thành viên của đội phát triển, với Marc Andreessen và Jim Clark lèo lái, đã tạo ra một công ty nhỏ để viết, bắt đầu từ số 0 (khi đã có những vấn đề với bản quyền về mã nguồn của Mosaic và thiết kế kỹ thuật của chương trình đã có những hạn chế của nó, xem *Tăng tốc cho Net: câu chuyện bên trong của Netscape và cách mà nó đã thay đổi Microsoft* [189]), những gì có thể sau này đã trở thành trình duyệt Netscape Communicator, mà không phải tranh luận, đã từng là người dẫn đầu của thị trường các trình duyệt Internet cho tới khi ra đời Microsoft Internet Explorer.

Bỏ qua một bên sự đổi mới sáng tạo thuần túy về công nghệ mà trình duyệt của Netscape đã thể thiện, thì hãng netscape Inc. cũng đã là đổi mới sáng tạo theo cách mà hãng định dồn thị trường vào góc tường. Hoàn toàn đối nghịch lại với những gì đã được gìn giữ theo nghĩa thông thường khi đó, ứng dụng ngôi sao của hãng, trình duyệt WWW, đã sẵn sàng một cách tự do (và còn có thể được phân phối với những hạn chế nhất định nào đó). Tiếp cận này, mà nó là hoàn toàn chưa từng được nghe thấy trong thế giới các doanh nghiệp khi đó, đã gây ra một số ngạc nhiên nhất định, mà hóa ra là nó lại đúng cho chiến lược của hãng Netscape Inc., và chỉ có người khổng lồ Microsoft có khả năng vượt được nó với những chiến thuật hung hăng hơn (và có lẽ gây hại cho sự cạnh tranh của thị trường tự do).

Khoảng năm 1997, thị phần của Netscape đã rớt nhanh chóng vì sự lan truyền của Microsoft Internet Explorer; hậu quả là, Netscape Inc. đã nghiên cứu những cách thức mới cho việc khôi phục lại sự áp đảo trước đó của nó. Một báo cáo kỹ thuật đã được xuất bản bởi kỹ sư Frank Hecker ("Việc thiết lập cửa hàng: sự kinh doanh của PMNM", 1998) [142] đã đề xuất rằng giải pháp tốt nhất cho vấn đề này là tung ra mã nguồn của trình duyệt và lợi ích từ những ảnh hưởng của cộng đồng PMTD, như được mô tả bởi Eric Raymond trong "Nhà thờ lớn và cái chợ".

Vào tháng 01/1998, Netscape Inc. chính thức đã tuyên bố rằng hãng có thể tung ra công khai mã nguồn trình duyệt của hãng, đánh dấu một cột mốc cực kỳ quan trọng trong lịch sử ngắn của PMTD: một công ty đã xuất bản toàn bộ mã nguồn của một ứng dụng mà đã từng là một sản phẩm sở hữu độc quyền cho tới tận khi đó, theo một giấy phép của PMTD. Ngày tung ra đã được định lịch là 31/03/1998.

Trong 2 tháng giữa tháng 1 và 3, mọi người tại Netscape đã tích cực một cách điên cuồng ("Việc giải phóng mã nguồn: lịch sử của Mozilla", 1999) [134]. Ở mức độ kỹ thuật, đã cần thiết để liên hệ với các công ty mà đã làm những module để hỏi họ về sự đồng ý của họ để thay đổi giấy phép: nếu câu trả lời là tiêu cực, thì module đó phải bị giới hạn. Bổ sung thêm, tất cả các phần được viết trong Java phải

được triển khai lại, vì nó đã được coi rằng Java là không phải tự do. Họ sau đó đã quyết định gọi dự án tự do này là Mozilla, chỉ vì các lập trình viên của Netscape đã gọi thành phần chính của họ là Mozilla, và miền Mozilla.org đã được mua để xây dựng một cộng đồng các lập trình viên và các cộng tác viên dựa xung quanh website này. Ở cuối của quá trình này, hơn 1,5 triệu dòng mã lệnh đã được tung ra.

Lưu ý

Cái tên Mozilla là một kiểu chơi chữ, với một chút hài hước từ đội phát triển của Netscape Inc. Tên Mozilla tới từ việc áp dụng cái tên Godzilla, con quái vật mà đã gây ra tai họa trong các phim kinh dị của Nhật Bản từ những năm 50, để cho nó nghe giống như là Mosaic Killer (Sát thủ Mosaic), như là trình duyệt mới, với công nghệ tiên tiến hơn, được hỗ trợ để thay cho Mosaic lỗi thời.

Một lưu ý khác, đã có câu hỏi về pháp lý. 3 giấy phép đang tồn tại khi đó đã không thuyết phục được các lãnh đạo của Netscape, những người có thể không thấy làm sao điều này có thể "tương thích" được với bản chất tự nhiên thương mại của một doanh nghiệp. Netscape đã muốn một giấy phép mềm dẻo hơn, mà nó có khả năng đạt được những thỏa thuận với các bên thứ 3 sao cho để đưa mã nguồn của họ bất chấp giấy phép hoặc liệu các lập trình viên thương mại khác có đóng góp vào cho nó, sao cho họ có thể bảo vệ được những lợi ích tài chính của họ bằng một cách nào đó mà họ chọn. Và dù họ đã không lên kế hoạch ban đầu để tạo ra một giấy phép mới, thì cuối cùng họ đã đạt được kết luận rằng điều này là cách duy nhất mà họ có thể đạt được những gì mà họ muốn.

Điều này giải thích làm thế nào mà Giấy phép Công cộng Netscape – NPL (Netscape Public License) đã được tạo ra: một giấy phép đã dựa trên những nguyên tắc cơ bản của các giấy phép của PMTD, nhưng cũng trao những quyền bổ sung nhất định nào đó cho Netscape Inc., mà cũng đã làm cho nó trở thành một giấy phép không tự do, từ quan điểm của FSF. Khi bản dự thảo của NPL đã được xuất bản cho thảo luận công khai, thì mệnh đề đưa ra các quyền bổ sung cho Netscape đã bị chỉ trích nặng nề. Hãng Netscape đã nhanh chóng phản ứng để trả lời cho những chỉ trích đó và đã tạo ra một giấy phép bổ sung, Giấy phép Công cộng Mozilla – MPL (Mozilla Public License), mà nó y hệt NPL, ngoại trừ trong đó Netscape đã không có các quyền bổ sung nào. Quyết định cuối cùng là đưa ra các mã nguồn của Netscape theo giấy phép NPL, được cung cấp các quyền bổ sung cho Netscape, còn bất kỳ mã nguồn mới nào được đưa vào thì sẽ được phân phối theo giấy phép MPL (hoặc một giấy phép tương thích). Những sửa đổi đối với mã nguồn ban đầu (được cấp phép theo NPL) cũng được phân phối theo giấy phép này.

Lưu ý

Hiện hành, Mozilla chấp nhận những đóng góp theo 3 giấy phép: MPL, GPL và LGPL. Việc thay đổi giấy phép tất cả là không dễ dàng, khi mà họ đã phải tìm tất cả mọi người mà đã đóng góp mã nguồn ở bất kỳ thời điểm nào sao cho họ có thể đưa ra sự ưng thuận của họ cho bất kỳ sự thay đổi nào từ NPL/MPL sang MPL/GPL/LGPL. Để cấp phép cho toàn bộ mã nguồn, một website, mà chứa một danh sách 300 cao thủ "bị mất tích", đã được tạo ra ("Bạn đã bao giờ thấy các cao thủ này chưa vậy?") [38]. Vào tháng 05/2007, họ vẫn còn tìm kiếm 2 trong số những lập trình viên này.

Việc phát triển mã nguồn ban đầu của Netscape Communicator đã, không nghi ngờ gì, phức tạp hơn so với dự kiến ban đầu. Những điều kiện ban đầu đã là tồi để bắt đầu, vì những gì đã được tung ra, một cách ngẫu nhiên, là không hoàn chỉnh (tất cả các module của bên thứ 3 mà không có sự đồng ý đã được đưa ra cho phiên bản đã được loại bỏ) và nó khó mà làm việc được. Dường như là điều đó đã không đủ,

bỏ qua vấn đề kỹ thuật để làm cho Mozilla làm việc trên một số lượng lớn hơn các hệ điều hành và nền tảng, đã có những lỗi được lấy từ hãng Netscape, với chu kỳ tung ra mà đã quá lâu và không hiệu quả đối với thế giới của Internet và nó đã không phân biệt được giữa những lợi ích của riêng nó và của cộng đồng được hình thành xung quanh Mozilla. Tất cả điều này đổ lên đầu chính xác một năm sau khi một trong những lập trình viên tích cực nhất từ trước và sau phiên bản đó, Jamie Zawinsky, đã quyết định ra đi sau một bức thư cay đắng ("Sự từ chức và sau cái chết", 1999) [237] trong đó ông đã nêu rõ sự thất vọng và sự cô độc của mình.

Vào ngày 15/07/2003, hãng Netscape (bây giờ là tài sản của America On Line) đã công bố rằng hãng đã không còn tiếp tục phát triển trình duyệt Netscape nữa và, vì thế, không còn chăm sóc tích cực cho dự án Mozilla nữa. Như một dạng của "giải quyết hậu quả" Netscape đã chấp nhận sự ra đời của Quỹ Mozilla, mà nó đã hỗ trợ với một sự đóng góp 2 triệu USD. Như vậy, tất cả các mã nguồn mà theo NPL đã được hiến tặng cho Quỹ này và được phân phối lại với các giấy phép trước kia đã được xuất bản bởi dự án Mozilla: MPL, LGPL và GPL.

Vào ngày 10/03/2005, Quỹ Mozilla đã tuyên bố rằng quỹ có thể không xuất bản bất kỳ phiên bản chính thức nào nữa của Bộ Ứng dụng Mozilla (Mozilla Application Suite), mà nó có thể được thay thế bởi Mozilla SeaMonkey, mà nó đã đưa vào một trình duyệt web, một trình thư điện tử máy trạm, một sổ địa chỉ, một trình soạn thảo HTML và một trình chat IRC máy trạm. Trong một lưu ý khác, dự án Mozilla chứa một loạt các ứng dụng độc lập, đáng chú ý nhất trong đó là Mozilla Firefox (trình duyệt web), mà không nghi ngờ gì nó là nổi tiếng nhất, Mozilla Thunderbird (trình thư điện tử và tin cho máy trạm), Mozilla Sunbird (lịch), Mozilla Nvu (trình soạn thảo HTML), Camino (trình duyệt web được thiết kế cho Mac OS X) và Bugzilla (công cụ theo dõi lỗi dựa trên web).

Thời gian đã qua, mặc dù nhiều nghi ngờ và một giai đoạn dài trong đó dường như là số phận đã an bài, dự án này bây giờ dường như đang đi tốt. Nhờ có sự linh hoạt và tính khả chuyển của các ứng dụng của nó, dù đòi hỏi nhiều tài nguyên thời gian thực trong nhiều trường hợp, chúng được sử dụng (nói chung, đặc biệt là Firefox) như cặp đôi của OpenOffice.org trong các máy tính để bàn của người sử dụng đầu cuối.

9.6.2 Hình ảnh X quang của Mozilla

Những con số mà chúng ta sẽ thảo luận trong phần này tương ứng với một nghiên cứu về Firefox, ứng dụng nổi tiếng nhất trong số các ứng dụng của dự án này. Theo đánh giá của mô hình COCOMO, một công ty mà muốn tạo ra phần mềm với phạm vi này có thể phải đầu tư khoảng 111 triệu USD để có nó. Thời gian nó có thể bỏ ra là khoảng 7 năm và số lượng trung bình các lập trình viên làm việc toàn thời gian mà công ty có thể phải sử dụng là khoảng 120 người.

Bảng 14. Hiện trạng của Mozilla Firefox

Website	www.mozilla-europe.org/es/products/firefox/
Bắt đầu của dự án	2002
Giấy phép	MPL/LGPL/GPL

Phiên bản	2
Số dòng mã lệnh	2,768,223
Ước tính giá thành	$ 111,161,078
Ước tính thời gian thực hiện	6.87 năm (82.39 tháng)
Ước tính số lượng trung bình các lập trình viên	120
Số lượng các lập trình viên khoảng	50 người được ủy thác
Các công cụ hỗ trợ phát triển	CVS, các danh sách thư, IRC, Bugzilla.

C++ và C là các ngôn ngữ được sử dụng nhiều nhất, theo trật tự ưu tiên tương ứng. Perl được sử dụng và điều này chủ yếu là do thực tế là các công cụ hỗ trợ phát triển được tạo ra bởi dự án Mozilla, như là BugZilla hoặc Tinderbox, đều được thiết kế trong ngôn ngữ này. Điều ngạc nhiên là số lượng lớn các dòng lệnh trong ngôn ngữ assembly trong một ứng dụng cho người sử dụng đầu cuối. Một nghiên cứu mã nguồn trong kho chỉ ra rằng, để hiệu quả, có nhiều tệp được viết mã trong ngôn ngữ assembly.

Bảng 15. Các ngôn ngữ được sử dụng trong Mozilla Firefox

Ngôn ngữ lập trình	Số dòng mã lệnh	Phần trăm
C++	1,777,764	64.22%
C	896,551	32.39%
Assembler	34,831	1.26%
Perl	26,768	0.97%
Shell	16,278	0.59%
C#	6,232	0.23%
Java	5,352	0.19%
Python	3,077	0.11%
Pascal	459	0.02%

9.7 OpenOffice.org

OpenOffice.org là một trong những ứng dụng ngôi sao hiện nay trong thế giới PMTD. Nó là một *bộ* ứng dụng văn phòng đa nền tảng mà bao gồm các ứng dụng chính trong một môi trường máy để bàn văn phòng, như trình xử lý văn bản (Writer), bảng tính (Calc), chương trình trình chiếu (Impress), một trình soạn thảo đồ họa (Draw), một công cụ cho việc tạo và sửa các công thức toán học (Math) và, cuối cùng, một trình soạn thảo ngôn ngữ HTML (được đưa vào trong Writer). Giao diện được cung cấp bởi OpenOffice.org là thuần nhất và trực giác, với một sự thể hiện và các chức năng tương tự như của các ứng dụng văn phòng khác, đặc biệt với Microsoft Office, bộ phần mềm văn phòng được sử dụng rộng

rãi nhất hiện nay.

Được viết trong C++, OpenOffice.org đưa vào các API của Java và có những thành phần riêng của nó cho các hệ thống nhúng, mà làm cho có khả năng để đưa vào, ví dụ, các bảng từ một bảng tính vào trong trình xử lý văn bản theo một cách rất đơn giản và trực giác. Một trong những ưu điểm của nó là việc nó có thể điều khiển được một số lượng lớn các định dạng tệp, bao gồm cả những định dạng tệp của Microsoft Office. Các định dạng tệp bẩm sinh của nó, không giống như của bộ phần mềm văn phòng của Microsoft, là dựa trên XML, mà nó chỉ ra rằng chúng được cam kết một cách rõ ràng cho tính linh hoạt, sự dễ dàng biến đổi và minh bạch. Hiện hành, OpenOffice.org đã được dịch trong hơn 25 ngôn ngữ và nó chạy trên Solaris (hệ thống bẩm sinh), GNU/Linux và Windows. Các phiên bản cho FreeBSD, IRIX và Mac OS X được mong đợi trong tương lai không xa.

OpenOffice.org lấy tên chắc chắn của nó (OpenOffice, như mọi người biết nó, cộng với thẻ .org) sau một vụ kiện, trong đó nó bị tố cáo về vi phạm thương hiệu của một công ty khác.

9.7.1 Lịch sử của OpenOffice.org

Vào giữa những năm 1980, công ty StarDivision đã được thành lập tại Cộng hòa Liên bang Đức, với mục tiêu cơ bản là tạo ra một bộ ứng dụng văn phòng: StarOffice. Vào mùa hè năm 1999, Sun Microsystems đã quyết định mua công ty StarDivision và đã đưa ra một cam kết đáng kể cho StarOffice, với dự định rõ ràng về việc vặn ra một phần của thị phần được chinh phục bởi Microsoft tại thời điểm đó. Vào tháng 06/2000, hãng này đã tung ra phiên bản 5.2 của StarOffice, mà có thể tải về được miễn phí từ Internet.

Tuy nhiên, thành công của StarOffice là hạn chế, vì thị trường đã bị áp đảo bởi gói văn phòng của Microsoft. Sun đã quyết định thay đổi chiến lược của mình và, như đã xảy ra với Netscape và dự án Mozilla, đã quyết định tận dụng ưu thế của PMTD để giành được tầm quan trọng và sự triển khai cho các hệ thống của hãng. Kết quả là, các phiên bản trong tương lai của StarOffice (một sản phẩm sở hữu độc quyền của Sun) có thể được tạo ra bằng việc sử dụng OpenOffice.org (một sản phẩm tự do) như một nguồn, tuân theo các giao diện lập trình ứng dụng (API) và các định dạng tệp và phục vụ như là triển khai cài đặt chuẩn.

9.7.2 Tổ chức của OpenOffice.org

OpenOffice.org có mục tiêu để có một cấu trúc ra quyết định trong đó tất cả các thành viên của cộng đồng cảm thấy là những người tham gia. Kết quả là, một hệ thống đã được tạo ra sao cho qui trình ra quyết định có thể có được sự đồng thuận lớn nhất có thể được. Dự án OpenOffice.org được chia thành một loạt các dự án phụ mà chúng được nắm bởi các thành viên của dự án, những cộng tác viên và một lãnh đạo duy nhất. Tất nhiên, các thành viên của một dự án có thể làm việc trên hơn một dự án, như lãnh đạo cũng có thể. Tuy nhiên, không ai có thể lãnh đạo hơn một dự án cùng một lúc. Các dự án được chia thành 3 loại:

- Các dự án được chấp nhận. Những dự án này có thể là kỹ thuật hoặc phi kỹ thuật. Những lãnh

đạo của từng dự án được chấp nhận có 1 phiếu khi cần có quyết định mang tính toàn cầu.

- Các dự án *ngôn ngữ bẩm sinh*. Tất cả những dự án quốc tế hoá và bản địa hóa của OpenOffice.org. Hiện hành, như chúng ta đã nhắc tới, có hơn 25 đội mà họ đang làm việc về dịch các ứng dụng của OpenOffice.org sang các ngôn ngữ và qui ước khác nhau. Như một tập hợp, các dự án ngôn ngữ bẩm sinh không có phiếu bầu nào trong các quyết định mang tính toàn cầu.
- Các dự án vườn ươm: Những dự án này được khuyến khích bởi cộng đồng (nói chung, chúng là thí điểm hoặc nhỏ). Chúng có thể trở thành các dự án được chấp nhận sau một giai đoạn 6 tháng. Để hiệu quả, cộng đồng OpenOffice.org có thể đảm bảo rằng các dự án được chấp nhận sẽ dựa trên một sự quan tâm thực sự, vì tỷ lệ tử vong của những dự án mới trong thế giới của PMTD là rất cao. Tổng cộng, các dự án vườn ươm có một phiếu trong các quyết định được đưa ra.

9.7.3 Hình ảnh X quang của OpenOffice.org

Bộ văn phòng OpenOffice.org có khoảng 4 triệu dòng mã lệnh được phân bổ qua 45,000 tệp.

Mô hình COCOMO ước lượng rằng công việc được yêu cầu để xây dựng một "mô phỏng" của OpenOffice.org có thể phải được cung cấp bởi 180 lập trình viên làm việc toàn thời gian trong hầu như 8 năm. Theo COCOMO ước tính, thì giá thành cho sự phát triển có thể khoảng 215 triệu USD. Kết quả được thảo luận trong phần này đã được lấy từ một nghiên cứu về mã nguồn của phiên bản ổn định 2.1 của OpenOffice.org.

Bảng 16. Hiện trạng của OpenOffice.org

Website	http://www.openoffice.org
Bắt đầu của dự án	Tháng 06/20000 (các phiên bản tự do đầu tiên)
Giấy phép	LGPL và SISSL
Phiên bản	2.1
Số dòng mã lệnh	5,197,090
Ước tính giá thành	$ 215,372,314
Ước tính thời gian thực hiện	8.83 năm (105.93 tháng)
Ước tính số lượng trung bình các lập trình viên	180
Ước tính số lượng các lập trình viên	200 người được ủy thác
Các công cụ hỗ trợ phát triển	CVS, các danh sách thư

Trong số các ngôn ngữ lập trình được sử dụng trong OpenOffice.org được quan tâm, thì hầu hết là C++. Thú vị để lưu ý là việc Sun mua một công ty khác lại cho kết quả là một sự tích hợp rất nhiều mã

nguồn Java vào trong bộ văn phòng này, mà nó còn vượt qua cả số lượng của ngôn ngữ C.

Bảng 17. Các ngôn ngữ lập trình được sử dụng trong OpenOffice.org

Ngôn ngữ lập trình	Số dòng mã lệnh	Phần trăm
C++	4,615,623	88.81%
Java	385,075	7.41%
C	105,691	2.03%
Perl	54,063	1.04%
Shell	12,732	0.24%
Yacc	6,828	0.13%
C#	6,594	0.13%

9.8 Red Hat Linux

Red Hat Linux là một trong những phát tán thương mại đầu tiên của GNU/Linux. Ngày nay, nó có lẽ là một trong những phát tán nổi tiếng nhất, và chắc chắn là một phát tán mà có thể được coi là "hợp chuẩn" trong tất cả các phát tán thương mại. Công việc của những người phân phối về cơ bản có liên quan tới các nhiệm vụ tích hợp và không nhiều tới sự phát triển phần mềm. Tất nhiên, Red Hat và các phát tán khác có thể có các lập trình viên làm việc cho họ, nhưng công việc của họ là thứ yếu với những mục tiêu của một phát tán. Nói chung, giả thiết rằng nhiệm vụ của các phát tán đơn giản là lấy các gói nguồn (thường các tệp được xuất bản bởi bản thân các lập trình viên) và đóng gói chúng sao cho chúng thỏa mãn được những tiêu chí nhất định nào đó (cả kỹ thuật lẫn tổ chức). Sản phẩm của quá trình này là một phát tán: một loạt các bó được tổ chức một cách phù hợp mà nó làm cho có khả năng để người sử dụng cài đặt, bỏ cài đặt và cập nhật chúng.

Các phát tán cũng có trách nhiệm về chất lượng của sản phẩm cuối cùng, mà là khía cạnh rất quan trọng nếu chúng ta coi rằng nhiều ứng dụng được đưa vào từng được phát triển bởi các tình nguyện viên trong lúc rảnh của họ. Hệ quả là, những khía cạnh an ninh và ổn định là cơ bản cho một phát tán.

9.8.1 Lịch sử của Red Hat

Hãng Red Hat Software Inc. đã được thành lập bởi Bob Young và Marc Ewing năm 1994. Mục đích chính là để biên dịch và thương mại hóa một phát tán GNU/Linux mà được gọi (và vẫn còn được gọi) là Red Hat Linux [236]. Về cơ bản, đây là một phiên bản đóng gói của những gì đã tồn tại trên Internet khi đó, bao gồm cả tài liệu và hỗ trợ. Phiên bản 1.0 của phát tán này đã ra đời vào mùa hè năm 1995. Ít tháng sau đó, vào mùa thu, phiên bản 2.0 , mà nó đã đưa vào công nghệ RPM (*trình quản lý gói RPM*) đã được xuất bản. Trình quản lý gói RPM đã trở thành chuẩn de facto cho các gói trong các hệ thống GNU/Linux. Vào năm 1998, phiên bản 5.2 của Red Hat đã được tung ra cho công chúng. Đối với lịch sử đầy đủ của những cái tên của các phiên bản khác nhau của Red Hat hãy đọc "Sự thực đằng sau

những cái tên của Red Hat" [201].

Lưu ý

Như của phiên bản 1.1 của Nền tảng Chuẩn Linux - Linux Standard Base (một đặc tả kỹ thuật được thiết kế để đạt được tính tương thích nhị phân giữa các phát tán GNU/Linux, mà nó được chăm sóc bởi Nhóm Tiêu chuẩn Mở - Free Standard Group), RPM đã được chọn như trình quản lý gói chuẩn. Dự án Debian tiếp tục với định dạng gói của riêng nó, như nhiều phát tán phụ thuộc vào hệ thống quản lý gói của Debian, và chúng được chỉnh để tiêu chuẩn hóa định dạng sử dụng một công cụ chuyển đổi được gọi là alien (người khác lạ).

Trước khi hệ thống quản lý RPM đã tồn tại, hầu hết tất cả các phát tán GNU/Linux đã đưa ra khả năng cài đặt phần mềm thông qua một thủ tục dựa trên thực đơn (menu), nhưng việc tiến hành những sửa đổi đối với một cài đặt đang tồn tại, đặc biệt việc bổ sung thêm các gói phần mềm mới sau khi cài đặt, là không dễ dàng. RPM đã làm bước đó vượt ra khỏi khả năng có thể bằng việc cung cấp cho những người sử dụng bằng khả năng để quản lý các gói của riêng họ ("RPM cực đại. Việc nắm lấy trình quản lý gói của Red Hat tới giới hạn", 1998) [83], mà nó đã làm cho có thể xóa được, cài đặt được hoặc cập nhật được bất kỳ gói phần mềm nào đang tồn tại trong phát tán theo một cách dễ dàng hơn nhiều. Hệ thống gói RPM tiếp tục là hệ thống quản lý gói được sử dụng rộng rãi nhất trong các phát tán GNU/Linux khác nhau. Những con số thống kê của các phát tán Linux, "Con số và sự kiện", 2003 [92]), một website chứa các thông tin định tính và định lượng trong một số lượng lớn các phát tán, chỉ ra rằng vào tháng 05/2003, đa số lớn (65) trong số 118 phát tán đã sử dụng để tính toán, đã sử dụng RPM (khoảng 55% tổng số). Để so sánh, định dạng gói của Debian (được biết như là deb) chỉ được sử dụng trong 16 phát tán (khoảng 14% tổng số).

Tuy nhiên, hãng Red Hat không chỉ được biết vì phát tán phần mềm của nó dựa trên Linux. Vào tháng 08/1999, Red Hat đã ra công khai và cổ phiếu của nó đã đạt được 8 ngày đầu cao nhất trong toàn bộ lịch sử của Phố Uôn. 4 năm sau, giá trị của cổ phiếu Red Hat đã giảm xuống 100 lần so với giá trị cực đại mà hãng đã đạt được trước khủng hoảng dotcom. Dù vậy, thành công ban đầu của hãng trên thị trường chứng khoán đã đặt Red Hat lên các trang nhất của cáo tờ báo và tạp chí mà không trực tiếp có sự chuyên môn hóa theo các vấn đề của công nghệ thông tin. Trong mọi trường hợp, dường như là Red Hat đã định vượt qua những vấn đề mà các công ty khác trong thế giới doanh nghiệp đã có với PMTD và các con số mà hãng đã xuất bản trong quý cuối của năm 2002 đã là màu đen lần đầu tiên trong lịch sử của hãng. Những sự kiện lịch sử quan trọng nhất có liên quan tới Red Hat là việc mua Cygnus Solutions vào tháng 11/1999, một công ty được thành lập từ một thập kỷ trước đã được chứng minh cách mà nó đã có khả năng kiếm tiền với một chiến lược tổng hợp dựa trên PMTD ("Tương lai của Cygnus Solutions. Một tài khoản doanh nghiệp") [216]. Cygnus chọn thị thường phức tạp của các trình biên dịch để ghi dấu của hãng. Chiến lược thương mại của hãng đã dựa trên sự phát triển và áp dụng các công cụ phát triển phần mềm của GNU (cơ bản là GCC và GDB) được tùy biến cho những nhu cầu của các khách hàng.

Vào tháng 09/2003, Red Hat đã quyết định tập trung vào công việc phát triển vào phiên bản doanh nghiệp của phát tán của hãng và đã ủy quyền phiên bản cộng đồng cho Fedora Core, một dự án nguồn mở độc lập với Red Hat.

Vào tháng 06/2006, Red Hat đã mua công ty JBoss, trở thành công ty có trách nhiệm về việc phát triển

máy chủ ứng dụng nguồn mở quan trọng nhất, J2EE.

9.8.2 Hiện trạng của Red Hat

Hiện hành, các sản phẩm quan trọng nhất của hãng Red Hat là Fedora Core và Red Hat Network, và dịch vụ cập nhật phần mềm Internet. Các dạng dịch vụ này được thiết kế nhiều hơn với trọng tâm hướng vào người sử dụng đầu cuối và không có nhiều cho môi trường của các doanh nghiệp, nhưng chúng là tốt cho Red Hat và quảng cáo cho bản thân hãng và để tăng cường chiến lược thương hiệu của hãng.

Chiến lược thương mại "thực sự" của Red Hat là dựa trên các sản phẩm mà hãng thiết kế cho thế giới các công ty. Những dạng sản phẩm này còn ít nổi tiếng hơn nhiều, nhưng chúng là phần chủ yếu của doanh số của Red Hat, lớn hơn nhiều so với doanh số của các sản phẩm ngôi sao phổ biến nhất của hãng theo nghĩa đen.

Red Hat có một phát tán mà nó là hướng doanh nghiệp, được tích hợp xung quanh một máy chủ ứng dụng được gọi là Red Hat Enterprise Linux AS. Các khách hàng mà mua phần mềm này cũng nhận được sự hỗ trợ. Tương đương với Red Hat Network cho những người sử dụng thương mại là Red Hat Enterprise Network, mà nó bao gồm sự quản lý và lựa chọn hệ thống đối với việc cập nhật. Một lưu ý nữa là, Red Hat cũng đưa ra các dịch vụ tư vấn công nghệ thông tin và một chương trình cấp chứng chỉ tương tự như thứ được đưa ra bởi Microsoft trong thế giới của Windows.

9.8.3 Bức tranh X quang của Red Hat

Red Hat gần đây đã đi qua được một cột mốc 50 triệu dòng mã lệnh, mà nó làm thành một trong những phát tán phần mềm lớn nhất đã tồn tại từ trước tới nay, vượt qua, như chúng ta sẽ thấy sau trong chương này, kích thước của các hệ điều hành sở hữu độc quyền. Red Hat phiên bản 8.1 có 792 gói, nên chúng ta có thể giả thiết rằng phiên bản mới nhất có thể đã có hơn 800 gói, nếu chúng ta biết rằng số lượng này có xu hướng gia tăng khá từ phiên bản này tới phiên bản khác. Như trong các ví dụ trước của chúng ta, mô hình COCOMO đã từng được sử dụng để ước tính đầu tư và nỗ lực mà có thể cần thiết để tạo ra một thế hệ các phần mềm phạm vi y như vậy. Tuy nhiên, trong trường hợp của Red Hat, chúng ta đã tính tới thực tế đây là một sản phẩm được chuẩn bị có sử dụng một loạt các ứng dụng độc lập. Hậu quả là, một ước tính độc lập của COCOMO đã được sử dụng cho mỗi ứng dụng trong các gói của Red Hat, và sau đó chúng ta đã bổ sung các giá thành và nhân lực ước tính mà có thể đã cần thiết. Để phân tích thời gian thiết kế tối ưu cho Red Hat, chúng ta đã chọn gói lớn nhất, như, theo lý thuyết, tất cả các gói là độc lập và có thể vì thế được thiết kế cùng một lúc. Vì lý do này, thời gian thiết kế tối ưu cho Red Hat là tương tự đối với thời gian của các dự án khác được trình bày trong các phần trước của chương này.

Theo COCOMO, khoảng 7,5 năm và một đội các lập trình viên, trung bình khoảng 1,800 lập trình viên, có thể đã cần thiết để thiết kế phát tán Red Hat Linux 8.1, bắt đầu từ số 0. Giá thành cuối cùng của sự phát triển này có thể khoảng 1,800 triệu USD.

Lưu ý

1,800 triệu USD là tổng số tiền mà Bộ Quốc phòng Tây Ban Nha đã phân bổ cho việc làm mới lại tàu sân bay cho máy bay trực thăng trong một ngân sách lớn nhất. Ngoài số tiền đó ra, một nửa sẽ được đầu tư để mua 24 chiếc máy bay trực thăng, nên chúng ta có thể nói rằng giá của Red Hat có thể tương đương với giá của 48 chiếc máy bay trực thăng chiến đấu. Cũng vậy, 1,800 triệu USD là tổng số tiền kiếm được từ bộ phim Titanic.

Bảng 18. Hiện trạng của Red Hat Linux

Website	http://www.redhat.com
Bắt đầu của dự án	1993
Giấy phép	
Phiên bản	9.0
Số dòng mã lệnh	Hơn 50,000,000
Số lượng các gói	792
Ước tính giá thành	$ 1,800,000,000
Ước tính thời gian thực hiện	7.35 năm (88.25 tháng)
Ước tính số lượng trung bình các lập trình viên	1,800
Số lượng các lập trình viên khoảng	Các nhân viên của Red Hat (chỉ tích hợp chung)
Các công cụ hỗ trợ phát triển	CVS, các danh sách thư

Do thực tế là có nhiều gói, nên các ngôn ngữ trong Red Hat là đa dạng hơn so với các ứng dụng mà chúng ta đã thấy trong các ứng dụng PMTD quan trọng nhất. Nói chung, C là rất quan trọng, với hơn 60% số lượng các dòng mã lệnh. Đứng thứ 2, với hơn 10 triệu dòng mã lệnh, chúng ta có C++, theo sau một khoảng lớn là Shell. Thú vị để lưu ý rằng sau Perl chúng ta có Lisp (chủ yếu vì sử dụng của nó trong Emacs), ngôn ngữ assembly (trong số 1/4 tương ứng đối với ngôn ngữ mà đi với Linux) và một ngôn ngữ mà việc sử dụng của nó thành thực mà nói là đang giảm, Fortran.

Bảng 19. Các ngôn ngữ lập trình được sử dụng trong Red Hat

Ngôn ngữ lập trình	Số dòng mã lệnh	Phần trăm
C	30,993,778	62.13%
C++	10,216,270	20.48%
Shell	3,251,493	6.52%
Perl	1,106,082	2.22%
Lisp	958,037	1.92%

Assembler	641,350	1.29%
Fortran	532,629	1.07%

9.9 Debian GNU/Linux

Debian là một hệ điều hành PMTD mà hiện hành sử dụng nhân Linux cho phát tán của nó (dù nó được mong đợi rằng sẽ có các phát tán Debian dựa trên các nhân khác trong tương lai, như trường hợp của "HURD"). Hiện nó đang có sẵn (trong năm 2007) cho nhiều kiến trúc khác nhau, bao gồm Intel x86, ARM, Motorola, 680x0, PowerPC, Alpha và SPARC.

Debian không chỉ là phát tán GNU/Linux lớn nhất đang tồn tại, mà còn là một trong những phát tán ổn định nhất và nó đã nhận được một loạt giải thưởng cho thực tế là nó được ưa thích hơn bởi người sử dụng. Dù nền tảng người sử dụng của nó là khó đánh giá, vì dự án Debian không bán các CD hoặc bất kỳ phương tiện nào khác với phần mềm của mình và những phần mềm mà nó có đều có thể được phân phối bởi bất kỳ ai muốn, nên chúng ta có thể giả thiết là, với một mức độ chấp nhận được nào đó, thì nó là một phát tán quan trọng bên trong thị trường của GNU/Linux.

Có một sự phân hóa trong Debian mà nó phụ thuộc vào các yêu cầu về giấy phép và sự phân phối của các gói. Nhân của phát tán Debian (phần được gọi là chính mà nó bao phủ một loạt lớn các gói) hình thành chỉ từ PMTD tuân theo với DFSG (những Chỉ dẫn của PMTD Debian) [104]. Nó có thể được tải về từ Internet và nhiều nhà phân phối lại bán nó trên các đĩa CD hoặc trên các phương tiện khác.

Các phát tán Debian được tạo ra bởi hầu như 1,000 tình nguyện viên (thường là những người chuyên nghiệp và các chuyên gia công nghệ thông tin). Công việc của những tình nguyện viên này cấu thành từ việc lấy các chương trình nguồn, trong hầu hết các trường hợp từ các tác giả gốc, việc thiết lập cấu hình cho chúng, việc biên dịch chúng và bó chúng lại sao cho một người sử dụng trung bình của phát tán Debian chỉ phải chọn gói này và hệ thống sẽ cài đặt nó mà không có các vấn đề nào hơn nữa. Những gì có thể ban đầu khi mới xuất hiện là đơn giản thì có thể sẽ trở nên phức tạp ngay sau khi các yếu tố khác, như những phụ thuộc giữa các gói khác nhau (gói A cần gói B để làm việc) và những phiên bản khác nhau của tất cả các gói này, sẽ phải được tính tới.

Công việc được thực hiện bởi các thành viên của dự án Debian là y hệt như công việc được thực hiện trong bất kỳ phát tán nào khác: sự tích hợp các phần mềm sao cho tất cả làm việc được cùng với nhau một cách phù hợp. Bỏ qua công việc thích nghi và đóng gói, các lập trình viên Debian có trách nhiệm về duy trì một hạ tầng các dịch vụ dựa trên Internet (website, các tệp trực tuyến, hệ thống quản lý lỗi, các danh sách thư hỗ trợ, sự hỗ trợ và phát triển, …), hàng loạt dự án dịch thuật và quốc tế hóa, sự phát triển của một loạt các công cụ đặc chủng cho Debian và, nói chung, có trách nhiệm về mọi thứ mà được yêu cầu để làm cho phát tán Debian làm việc.

Bỏ qua bản chất tự nhiên tự nguyện của nó, dự án Debian có một tính năng mà là đặc biệt độc nhất vô nhị: liên hệ xã hội của Debian (http://www.debian.org/social_contract.html) [106]. Tài liệu này không chỉ mô tả các mục tiêu chính của dự án Debian, mà còn các phương tiện mà nó sẽ được sử dụng để đạt được chúng. Debian cũng được biết tới vì có một chính sách về các gói và các phiên bản rất nghiêm ngặt, được thiết kế để đạt được chất lượng tốt nhất trong sản phẩm này ("Sách chỉ dẫn về chính sách

của Debian") [105]. Theo cách này, có 3 dạng khác nhau của Debian ở bất kỳ lúc nào: một phiên bản ổn định, một phiên bản không ổn định và một phiên bản thử nghiệm. Như bản thân tên của của nó chỉ ra, phiên bản ổn định là phiên bản được khuyến cáo cho các hệ thống và người sử dụng mà đòi hỏi tính ổn định hoàn toàn. Phần mềm phải tuân thủ một giai đoạn đóng băng, trong giai đoạn này bất kỳ lỗi nào cũng được sửa cho đúng. Qui định chung là việc phiên bản Debian ổn định phải không có bất kỳ lỗi nguy kịch nào được biết tới. Mặt khác, phiên bản ổn định này thường không có những phiên bản mới nhất của phần mềm (các bổ sung mới nhất).

Có 2 phiên bản khác của Debian mà nó tồn tại cùng với phiên bản ổn định cho những ai mà họ mong muốn có được phần mềm mới nhất. Phiên bản không ổn định bao gồm các gói mà đang được ổn định hóa, trong khi phiên bản thử nghiệm, như tên của nó chỉ ra, là phiên bản mà có một xu hướng lớn sẽ hỏng và nó chứa những thứ mới nhất của phần mềm mới nhất.

Khi nghiên cứu lần đầu đã được thực hiện, thì phiên bản ổn định của Debian là Debian 3.0 (cũng còn được gọi là Woody), còn phiên bản không ổn định là Sid và phiên bản thử nghiệm là Sarge. Tuy nhiên, Woody cũng đã đi qua một giai đoạn không ổn định, và trước đó, một giai đoạn thử nghiệm. Điều này là quan trọng, vì những gì chúng ta sẽ xem xét trong bài viết này sẽ bao trùm các phiên bản ổn định khác nhau của Debian, từ trước tới nay kể từ phiên bản 2.0 đã được xuất bản, vào năm 1998. Ví dụ, chúng ta có Debian 2.0 (tên mã là Hamm), Debian 2.1(Slink), Debian 2.2 9(Potato) và, cuối cùng, Debian 3.0 (Woody).

Lưu ý

Những cái tên hiệu của các phiên bản Debian tương ứng với những đặc tính trong câu chuyện Toy (đồ chơi) của phim hoạt hình, một truyền thống mà đã được bắt đầu, nửa đùa nửa thật, khi phiên bản 2.0 đã được xuất bản và Bruce Perens, người đứng đầu của dự án này khi đó và sau này là nhà sáng lập của OSI và người đại diện phát biểu cho nguồn mở, đã làm việc cho một công ty mà hãng này đã thiết kế bộ phim này. Chi tiết hơn về lịch sử của Debian và phát tán Debian nói chung, chúng ta khuyến cáo "Tóm tắt lịch sử của Debian" [122].

9.9.1 Hình ảnh X quang của Debian

Debian GNU/Linux có lẽ là biên dịch lớn nhất của PMTD mà nó làm việc theo một cách thức phối hợp và, không còn nghi ngờ gì nữa, một trong những sản phẩm phần mềm lớn nhất từ trước tới này được xây dựng. Phiên bản 4.0, được tung ra vào tháng 04/2007 (được gọi là Etch), cấu tạo từ hơn 10,000 gói nguồn, với hơn 288 triệu dòng mã lệnh.

Số lượng các dòng mã lệnh trong Debian 3.0 là 105 triệu. Theo mô hình COCOMO, tổng số khoảng 3,600 triệu USD có thể phải chi ra để có được phần mềm tương tự như thế được đóng bó với phát tán này. Như với Red Hat, nỗ lực này đòi hỏi phải xây dựng mỗi gói riêng rẽ phải được tính toán và các con số kết quả phải được bổ sung vào sau đó cho từng gói. Vì lý do y hệt này, thời gian nó có thể phải bỏ ra để phát triển Debian là 7 năm, vì các gói tất cả có thể được xây dựng cùng một lúc được. Tuy nhiên, trung bình khoảng 4,000 lập trình viên có thể đã phải được huy động trong suốt 7 năm này.

Bảng 20. Hiện trạng của Debian

Website	http://www.debian.org
Bắt đầu của dự án	16/08/1993
Giấy phép	Những giấy phép thỏa mãn cho DFSG
Phiên bản được sử dụng	Debian 4.0 (còn được gọi là Etch)
Số dòng mã lệnh	288,500,000
Số lượng các gói	10,106
Ước tính giá thành	$ 10,140 triệu
Ước tính thời gian thực hiện	8.84 năm
Số lượng những người duy trì khoảng	Khoảng 1,500
Các công cụ hỗ trợ phát triển	Các danh sách thư, hệ thống báo cáo lỗi

Ngôn ngữ được sử dụng phổ biến nhất trong Debian 4.0 là C, với hơn 51% số lượng các dòng mã lệnh. Tuy nhiên, như chúng ta sẽ chỉ ra một chút sau đây trong phần này, tầm quan trọng của C đang giảm với thời gian, khi mà 80% các mã nguồn trong các phiên bản đầu của Debian, là C. Ngôn ngữ được sử dụng phổ biến thứ 2, là C++, chia sẻ một phần khá của "khiếu nại" cho sự đi xuống của C; tuy nhiên, C đặc biệt đã bị ảnh hưởng bởi sự gia tăng của các ngôn ngữ scripting như Perl, Python và PHP. Mặt khác, các ngôn ngữ như Lisp hoặc Java (mà được trình bày không đúng mức trong Debian vì chính sách của nó không chấp nhận mã nguồn mà phụ thuộc vào máy ảo sở hữu độc quyền của Sun) đôi khi cũng được đưa vào.

Lưu ý

3,600 triệu USD là ngân sách được phân bổ bởi Chương trình Khung công việc của Ủy ban châu Âu lần thứ 6 cho nghiên cứu và phát triển về xã hội thông tin. Nó cũng là số tiền mà Telefónica định đầu tư vào Đức để triển khai các dịch vụ UMTS.

Bảng 21. Các ngôn ngữ lập trình được sử dụng trong Debian GNU/Linux 4.0

Ngôn ngữ lập trình	Số dòng mã lệnh (triệu dòng)	Phần trăm
C	155	51%
C++	55	19%
Shell	30	10%
Perl	8.1	2.9%
Lisp	7.7	2.7%
Python	7.2	2.5%

Java	6.9	2.4%
PHP	3.5	1.24%

Bảng 22 chỉ ra cách mà những ngôn ngữ quan trọng nhất được phát triển trong Debian.

Bảng 22. Các ngôn ngữ được sử dụng nhiều nhất trong Debian

Ngôn ngữ	Debian 2.0		Debian 2.1		Debian 2.2		Debian 3.0	
C	19,400,000	76.67%	27,800,000	74.89%	40,900,000	69.12%	66,500,000	63.08%
C++	1,600,000	6.16%	2,800,000	7.57%	5,980,000	10.11%	13,000,000	12.39%
Shell	645,000	2.55%	1,150,000	3.10%	2,710,000	4.59%	8,635,000	8.19%
Lisp	1,425,000	5.64%	1,890,000	5.10%	3,200,000	5.41%	4,090,000	3.87%
Perl	425,000	1.68%	774,000	2.09%	1,395,000	2.36%	3,199,000	3.03%
Fortran	494,000	1.96%	735,000	1.98%	1,182,000	1.99%	1,939,000	1.84%
Python	122,000	0.48%	211,000	0.57%	349,000	0.59%	1,459,000	1.38%
Tcl	311,000	1.23%	458,000	1.24%	557,000	0.94%	1,081,000	1.02%

Có những ngôn ngữ mà chúng ta có thể coi là thiểu số mà chúng đạt được những vị thế khá cao trong sự phân loại. Điều này là do thực tế rằng, trong khi chúng chỉ thể hiện trong một số lượng nhỏ các gói, thì các gói theo yêu cầu là khá lớn. Như trường hợp của Ada, mà trong khi chỉ có trong 3 gói (GNAT, một trình biên dịch của Ada; libgtkada, một liên kết tới thư viện GTK và ASIS, một hệ thống cho việc quản lý các nguồn của Ada) bao trùm 430,000 trong tổng số 576,000 dòng mã lệnh mà đã được tính trong Debian 3.0 cho Ada. Trường hợp tương tự khác là Lisp, mà nó chỉ xuất hiện trong GNU Emacs và XEmacs, nhưng có hơn 1,200,000 dòng mã lệnh của khoảng 4 triệu trong toàn bộ phát tán.

9.9.2 So sánh với các hệ điều hành khác

Có câu tục ngữ nói rằng tất cả những so sánh đều khập khiễng; điều này là đặc biệt đúng khi so sánh PMTD với PMSHĐQ. Các hình ảnh X quang chi tiết của Red Hat Linux và Debian đã có thể vì chúng là những ví dụ của PMTD. Việc có sự truy cập vào mã nguồn (và tới các thông tin khác đã được đưa ra trong chương này) là cơ bản cho việc nghiên cứu số các dòng lệnh, các gói, các ngôn ngữ lập trình... của các phiên bản khác nhau. Nhưng những ưu điểm của PMTD là vượt ra ngoài điều này, vì, bổ sung thêm vào, chúng làm cho dễ dàng hơn đối với các bên thứ 3, bất kể họ là những đội nghiên cứu hay đơn giản chỉ là những người mà có quan tâm, để phân tích chúng.

Trong các hệ thống sở hữu độc quyền nói chung, một nghiên cứu như thế này có thể là hoàn toàn không thể. Trong thực tế, những con số được cung cấp bên dưới đã có được từ bản thân các công ty

đằng sau sự phát triển PMSHĐQ, mà nó có nghĩa là chúng ta không ở vào vị trí để đảm bảo được tính trung thực của chúng. Trong nhiều trường hợp chúng ta không biết liệu chúng có đang nói về các dòng lệnh của mã nguồn vật lý hay không, khi mà chúng ta đã thực hiện trong chương này, hay liệu chúng có đưa vào các dòng trống và bình luận hay không. Hơn nữa, chúng ta không biết chắc chắn những gì họ đưa vào trong phần mềm của họ, có nghĩa là chúng ta không biết liệu các phiên bản nhất định nào đó của Microsoft Windows có đưa vào bộ Microsoft Office hay không.

Trong mọi trường hợp, coi tất cả những thứ chúng ta đã thảo luận về vấn đề này trong các đoạn trên, chúng ta tin tưởng rằng sẽ thú vị để đưa vào sự so sánh này, vì nó giúp chúng ta thấy được vị trí trong đó những phát tán khác nhau của Red Hat và Debian ra sao, trong một ngữ cảnh rộng lớn hơn. Những gì là không thể hỏi được là việc cả Debian và Red Hat, mà đặc biệt là Red Hat, là những bộ sưu tập lớn nhất của phần mềm từ trước tới nay được thấy bởi loài người hôm nay.

Các con số được trích ra bên dưới tới từ Mark Lucovsky [168] cho Windows XP và Bruce Schneier [200] cho tất cả các hệ thống khác. Bảng 23 đưa ra một sự so sánh, từ nhỏ nhất cho tới lớn nhất.

Bảng 23. So sánh với các hệ thống sở hữu độc quyền

Hệ điều hành	Ngày xuất bản	Số dòng mã lệnh (khoảng)
Microsoft Windows 3.1	Tháng 04/1992	3000000
SUN Solaris 7	Tháng 10/1998	7,500,000
SUN StarOffice 5.2	Tháng 06/2000	7,600,000
Microsoft Windows 95	Tháng 08/1995	15,000,000
Red Hat Linux 6.2	Tháng 03/2000	18,000,000
Debian 2.0	Tháng 07/1998	25,000,000
Microsoft Windows 2000	Tháng 02/2000	29,000,000
Red Hat Linux 7.1	Tháng 04/2001	32,000,000
Debian 2.1	Tháng 03/1999	37,000,000
Windows NT 4.0	Tháng 07/1996	40,000,000
Red Hat Linux 8.0	Tháng 09/2002	50,000,000
Debian 2.2	Tháng 08/2000	55,000,000
Debian 3.0	Tháng 07/2002	105,000,000

9.10 Eclipse

Nền tảng Eclipse được cấu tạo từ một IDE (*môi trường phát triển tích hợp*) mở và được mở rộng. Một IDE là một chương trình được cấu tạo từ một tập hợp các công cụ mà chúng hữu dụng cho một lập

trình viên phần mềm. Các yếu tố cơ bản của một IDE bao gồm một trình soạn thảo mã nguồn, một trình biên dịch/phiên dịch và một trình gỡ lỗi (debugger). Eclipse là một IDE trong Java và cung cấp hàng loạt các công cụ phát triển phần mềm. Nó cũng hỗ trợ các ngôn ngữ lập trình khác, như C/C++, Cobol, Fortran, PHP hoặc Python. Các trình cài cắm (plug-in) có thể được bổ sung vào nền tảng cơ bản của Eclipse để gia tăng chức năng.

Khái niệm Eclipse cũng tham chiếu tới cộng đồng PMTD mà nó phát triển nền tảng Eclipse. Công việc này được chia thành các dự án với mục đích để cung cấp một nền tảng mạnh mẽ, có thể thay đổi được theo phạm vi có chất lượng để phát triển của phần mềm với IDE của Eclipse. Công việc được phối hợp bởi Quỹ Eclipse, mà nó là một tổ chức phi lợi nhuận được tạo ra để thúc đẩy và phát triển nền tảng Eclipse và hỗ trợ cả cộng đồng và hệ sinh thái Eclipse.

9.10.1 Lịch sử của Eclipse

Nhiều chương trình của Eclipse đã được triển khai bởi IBM trước khi dự án Eclipse được tạo ra. Người tiền nhiệm của Eclipse là VisualAge và nó đã được xây dựng có sử dụng Smalltalk trong một môi trường phát triển được gọi là Envy. Sau khi Java xuất hiện vào những năm 90, IBM đã phát triển một máy ảo mà nó đã làm việc với cả Smalltalk và Java. Sự phát triển nhanh chóng của Java và những ưu điểm của nó với sự tập trung vào Internet mà nó đã đang mở rộng mạnh mẽ đã buộc IBM phải xem xét tới việc bỏ máy ảo đôi này và xây dựng một nền tảng mới dựa trên Java từ đầu. Sản phẩm cuối cùng là Eclipse, mà nó đã ngốn của IBM khoảng 40 triệu USD vào năm 2001.

Về cuối năm 2001, IBM, cùng với Borland, đã tạo ra quỹ Eclipse phi lợi nhuận, bằng cách đó mở ra với thế giới PMNM. Nhóm các doanh nghiệp này đã dần dần liên kết được với các công ty phát triển phần mềm toàn cầu quan trọng như: Oracle, Rational Software, Red Hat, SuSE, HP, Serena, Ericson và Novell, và những công ty khác. Không có 2 công ty là Microsoft và Sun Microsystem.

Microsoft đã bị loại bỏ vì sự độc quyền của hãng đối với thị trường và Sun Microsystem đã có IDE của riêng hãng, và là sự cạnh tranh chính của Eclipse: NetBeans. Trên thực tế, cái tên Eclipse đã được chọn vì mục tiêu là để tạo ra một IDE có khả năng “làm lu mờ (eclipse) Visual Studio” (của Microsoft) và để “làm lu mờ mặt trời” (Sun Microsystem).

Phiên bản ổn định mới nhất của Eclipse là sẵn sàng cho các hệ điều hành Windows, Linux, Solaris, AIX, HP-UX và Mac OS X. Tất cả các phiên bản của Eclipse cần phải có một máy chủ ảo Java – JVM (Java Virtual Machine) được cài đặt trong hệ thống, ưa hơn thì là môi trường thời gian thực Java - JRE (Java Runtime Environment) hoặc bộ công cụ cho lập trình viên Java - JDK (Java Developer Kit) của Sun, mà, vào đầu năm 2007, còn chưa là tự do (dù Sun đã công bố rằng JVM của họ sẽ là tự do).

9.10.2 Hiện trạng của Eclipse

Tất cả công việc được chuẩn bị cho nhóm Eclipse được tổ chức trong các dự án khác nhau. Có những dự án tới lượt nó được chia thành các dự án phụ và các dự án phụ thành các thành phần. Các dự án mức cao được quản lý bởi các ban của Quỹ Eclipse PMC (*Project management committee - Ban quản lý dự*

án). Danh sách sau đây chỉ ra các dự án mức cao:

- Eclipse. Nền tảng cơ bản cho những thành phần còn lại. Nền tảng này sẽ là tự do, khỏe mạnh, hoàn chỉnh và có chất lượng tốt cho sự phát triển của các nền tảng máy trạm giàu – RCP (*rich client platform*) và các công cụ được tích hợp (plug-ins). Nhân thời gian thực của nền tảng Eclipse được gọi là Equinox và nó là một triển khai cài đặt của đặc tả kỹ thuật OSGi (Open Services Gateway Initiative - Sáng kiến Cổng dịch vụ mở), mà nó mô tả một kiến trúc hướng dịch vụ (SOA) cho các ứng dụng.
- Các công cụ (ETP, *dự án các công cụ của Eclipse*). Hàng loạt các công cụ và các thành phần cho nền tảng Eclipse.
- Web (WTP, *dự án các công cụ web*). Các công cụ để phát triển các ứng dụng web và JEE (Java Enterprise Edition - Phiên bản Java cho doanh nghiệp).
- *Dự án thử nghiệm và các công cụ thực thi* (TPTP). Các công cụ thử nghiệm và đo đạc mức độ thực thi sao cho các lập trình viên có thể theo dõi được các ứng dụng của họ và làm cho họ làm việc năng suất hơn.
- Các báo cáo Web (BIRT, *các công cụ báo cáo và tri thức doanh nghiệp*). Hệ thống tạo báo cáo Web.
- Việc mô hình hóa (EMP, *dự án mô hình hóa Eclipse*). Các công cụ phát triển dựa theo mô hình.
- Dữ liệu (DTP, *nền tảng các công cụ về dữ liệu*). Hỗ trợ cho các công nghệ quản lý các dữ liệu.
- Các thiết bị nhúng (DSDP, *nền tảng phát triển phần mềm cho các thiết bị*). Các công cụ cho sự phát triển các ứng dụng mà sẽ được chạy trên các thiết bị với phần cứng hạn chế, nói một cách khác, các thiết bị nhúng.
- *Kiến trúc hướng dịch vụ* (SOA). Các công cụ cho việc phát triển các dự án hướng dịch vụ.
- Công nghệ Eclipse. Nghiên cứu, phổ biến và phát triển nền tảng Eclipse.

Các nguyên tắc mà chúng chỉ dẫn cho sự phát triển của cộng đồng Eclipse là như sau:

- Chất lượng. Phần mềm được phát triển tại Eclipse phải đáp ứng được các chuẩn về chất lượng của kỹ thuật phần mềm.
- Sự phát triển. Nền tảng Eclipse, và tất cả các công cụ dựa trên nó, phải phát triển một cách năng động phù hợp với các yêu cầu của người sử dụng.
- Chế độ nhân tài. Ai đó mà càng đóng góp nhiều, thì các trách nhiệm của anh ta hoặc chị ta càng nhiều.
- Hệ sinh thái Eclipse. Sẽ có những tài nguyên được hiến tặng bởi cộng đồng nguồn mở cho nhóm các công ty của Eclipse. Những tài nguyên này sẽ được sử dụng theo cách cách thức mà có lợi cho cộng đồng.

Qui trình phát triển của Eclipse tuân theo các pha được xác định trước. Đầu tiên, có một pha gọi là pha đề xuất trước, ở đó một cá nhân hoặc công ty công bố mối quan tâm của họ trong việc thiết lập ra một dự án. Nếu đề xuất được chấp thuận, thì nó sẽ được quyết định liệu nó sẽ là một dự án mức cao hay là một dự án phụ. Bước tiếp theo là sẽ kiểm tra tính đúng đắn của dự án về tính có thể áp dụng được và chất lượng. Sau một pha ở đó dự án được ươm thử, sẽ có một sự rà soát lại cuối cùng. Nếu dự án vượt qua được sự rà soát này, thì nó sẽ được phê chuẩn tính đúng đắn của nó trước khi cộng đồng Eclipse và nó sẽ đi vào pha triển khai.

9.10.3 Hình ảnh X quang của Eclipse

Eclipse được phân phối theo một giấy phép EPL (Eclipse Public License). Giấy phép này được cho là tự do bởi FSF và OSI. Theo giấy phép EPL, có khả năng sử dụng, sửa đổi, sao chép và phân phối các phiên bản mới của sản phẩm được cấp phép. Người tiền nhiệmcủa EPL là CPL (Common Public License - Giấy phép Công cộng chung). CPL đã được viết bởi IBM, trong khi EPL là công việc của nhóm các công ty của Eclipse.

Việc đánh giá sự đầu tư và nỗ lực được đặt vào Eclipse là một nhiệm vụ không dễ dàng. Điều này là vì thực tế là mã nguồn mà tạo nên hệ sinh thái Eclipse được phân tán trong vô số các dự án và các kho phần mềm.

Bên dưới là các kết quả của việc áp dụng mô hình COCOMO cho nền tảng của Eclipse, mà nó được sử dụng như là cơ sở cho phần còn lại của các trình cài cắm bổ sung (plug-ins).

Bảng 24. Phân tích về Eclipse

Website	http://www.eclipse.org
Bắt đầu của dự án	2001
Giấy phép	Giấy phép Công cộng Eclipse (EPL)
Phiên bản được phân tích	3.2.2
Số dòng mã lệnh	2,163,932
Số lượng các tệp	15,426
Ước tính giá thành	$ 85,831,641
Ước tính thời gian thực hiện	6.22 năm (74.68 tháng)
Ước tính số lượng trung bình các lập trình viên	102.10
Số lượng các lập trình viên khoảng	133 người được ủy thác
Các công cụ hỗ trợ phát triển	CVS, các danh sách thư, hệ thống theo dõi lỗi (Bugzilla)

Bảng sau chỉ ra các ngôn ngữ lập trình trong Eclipse 3.2.2:

Bảng 25. Các ngôn ngữ lập trình được sử dụng trong Eclipse

Ngôn ngữ lập trình	Số dòng mã lệnh	Phần trăm
Java	2,066,631	95.50%
C	85,829	3.97%
Perl	3,224	0.06%
C++	5,442	0.25%
JSP	3,786	0.17%
Perl	1,325	0.06%
Lex	1,510	0.03%
Shell	849	0.04%
Python	46	0.00%
PHP	24	0.00%

10 Các nguồn tự do khác

> "Nếu bạn muốn tạo ra một cái bánh táo từ bàn tay trắng, thì bạn trước hết phải tạo ra vũ trụ đã".
>
> Carl Sagan.

Những ý tưởng đằng sau các chương trình tự do liệu có thể được mở rộng cho các tài nguyên khác không? chúng ta có thể coi rằng các nguồn thông tin khác mà có thể dễ dàng được sao chép bằng điện tử là tương tự như các chương trình và với những sự tự do, những qui định, những mô hình phát triển và kinh doanh y hệt có thể áp dụng cho chúng. Tuy nhiên có một số sự khác biệt và những ảnh hưởng của những khác biệt này có nghĩa là chúng đã không phát triển với cùng sức mạnh như các chương trình. Sự khác biệt chính là việc tất cả mọi thứ mà một người phải làm là sao chép các chương trình này để làm cho chúng làm việc được, trong khi các dạng thông tin khác sẽ được sao chép, chúng phải đi qua được ít nhiều một quá trình có giá thành trước khi chúng có thể bắt đầu trở thành hữu dụng theo bất kỳ cách gì, mà có thể đi từ việc học một tài liệu cho tới pha sản xuất phần cứng được mô tả trong một ngôn ngữ phù hợp.

10.1 Những tài nguyên tự do quan trọng nhất

Chúng ta đã thảo luận về tài liệu của các chương trình và những tài liệu kỹ thuật khác ở phần 3.2.5. Ở đây chúng ta sẽ đề cập tới những dạng sáng tạo khác, mà chúng cũng có thể là văn bản, nhưng chúng không có liên quan tới phần mềm, mà tới các lĩnh vực khoa học, kỹ thuật và nghệ thuật.

10.1.1 Các tài liệu khoa học

Con đường mà theo đó khoa học tiến bộ là, ở một mức độ nào đó, do thực tế là các nhà nghiên cứu mà làm cho nó tiến bộ vì lợi ích của con người, xuất bản các kết quả công việc của họ trong các tạp chí mà tới được với công chúng rộng rãi. Nhờ sự phổ biến này, các nhà nghiên cứu phát triển một hồ sơ theo dõi mà cho phép chúng tiến hóa theo những vị thế cao hơn và trách nhiệm cao hơn, trong khi họ nhận được doanh thu từ các hợp đồng nghiên cứu mà họ giành được nhờ uy tín phát triển của họ.

Cách phổ biến tài liệu này đại diện cho một mô hình kinh doanh mà đã chứng minh được là rất hiệu quả. Đối với mô hình công việc này, chất lượng của công việc phải được đảm bảo và các tài liệu phải được phổ biến rộng rãi. Cản trở mà nó ngăn cản sự phổ biến là số lượng lớn các tạp chí đang tồn tại, đối với một giá thành đáng kể, mà chỉ có thể được mua với những ngân sách hào phóng. Chất lượng được đảm bảo bởi thực tế là các tài liệu này được xem xét lại bởi các chuyên gia hoặc các đồng nghiệp.

Liên quan tới điều này, hàng loạt các tạp chí trực tuyến đã nổi lên, trong số đó chúng ta có thể nhắc tới những thứ kỳ cựu *Ngày thứ hai Đầu tiên* ("First Monday: tạp chí được xem xét lại của các đồng nghiệp trên Internet") [26] hoặc dự án *Thư viện Công của Khoa học* (PLOS http://www.publiclibraryofscience.org [55]). "Thư mục các Tạp chí Truy cập Mở" [22] trính ra nhiều hơn nhiều. Liệu mọi người không phải là các tác giả có được phép xuất bản những sửa đổi cho các

dạng tài liệu này không? Có những sự phản đối mà nó trải từ khả năng của chất lượng dưới mức chuẩn hoặc mập mờ nước đôi về những ý kiến hoặc những kết quả, cho tới sự nguy hiểm của những người mà họ có thể dễ dàng ăn cắp các tài liệu và làm nổi lên mà không có nỗ lực nào, trong khi lại phủ nhận các tác giả đích thực của những giá trị khó kiếm được của họ. Tuy nhiên, thực tế là tất cả những người viết có bổn phận về việc trích dẫn tác giả gốc và đưa tài liệu đó cho sự rà soát lại của các đồng nghiệp để xuất bản trong một tạp chí có uy tín có thể bù đắp được cho các vấn đề này (xem phần 10.2.2).

Một sự tương tự cũng đã được thiết lập giữa PMTD và khoa học, khi mà mô hình phát triển của PMTD đòi hỏi số lượng lớn nhất về phổ biến, rà soát lại của các đồng nghiệp (giả thiết là các chuyên gia) và sử dụng lại các kết quả ("PMTD/khoa học tự do", 2001) [154].

10.1.2 Luật pháp và chuẩn

Có những tài liệu của một sự tự nhiên điều chỉnh mà nó xác định làm thế nào mà mọi thứ phải được thực hiện, như là để cải thiện sự cùng tồn tại giữa mọi người hoặc sao cho các chương trình hoặc các máy có thể hoạt động được cùng với nhau. Những tài liệu này cần phải được phổ biến một cách rộng rãi, mà nó có nghĩa rằng bất kỳ cản trở nào cũng sẽ là phản sản xuất. Vì lý do này, có thể hiểu được rằng chúng nhận được sự đối xử đặc biệt, như được minh họa trong Luật Sở hữu Trí tuệ của Tây Ban Nha:

> "Những đề xuất pháp lý và điều chỉnh và những phác thảo của chúng, các cơ quan xét xử và tài phán và các luật, thỏa thuận, những cân nhắc thận trọng và những quyết định của các cơ quan nhà nước, và những bản dịch chính thức của tất cả các văn bản như vậy, phải không là đối tượng của sở hữu trí tuệ".

Sự tương tự về công nghệ của các luật này có thể là những qui tắc tiêu chuẩn hoặc các chuẩn. Trong việc lập trình, các giao thức giao tiếp, hoặc giữa các máy từ xa hoặc giữa các module trong cùng một máy tính, là đặc biệt quan trọng. Rõ ràng là chúng ta phải không hạn chế sự phổ biến của chúng, đặc biệt nếu chúng ta muốn các chương trình tự do mà chúng hoạt động với những người khác để thịnh vượng, nhưng, bất chấp điều này, theo truyền thống, các cơ quan mà điều chỉnh các vấn đề này, như là ISO[11] và ITU[12], bán các điều chỉnh và chuẩn của họ ngay cả ở các định dạng điện tử, và ngăn cấm sự phân phối lại của chúng.

Dù điều này có thể được hợp thức hóa ở một mức độ nào đó, cho là nhu cầu để bù đắp một phần giá thành, thì sự phổ biến tự do của các chuẩn sẽ là có hiệu suất hơn nhiều; đây là trường hợp của những chỉ dẫn của W3C[13] và, đặc biệt nơi mà các chuẩn Internet có liên quan, thì các tài liệu được gọi là RFC (*yêu cầu cho các bình luận*) mà đã tồn tại từ đầu, ở các định dạng điện tử mà có thể đọc được bằng việc sử dụng bất kỳ trình soạn thảo văn bản nào.

Tuy nhiên, sự thành công của các giao thức Internet là không chỉ vì tính sẵn sàng của chúng. Những yếu tố khác bao gồm *mô hình phát triển*, mà rất tương tự như PMTD vì tính mở của nó đối với sự tham gia của bất kỳ người nào có quan tâm và sử dụng các danh sách thư và các yếu tố tương tự. Qui trình này được mô tả trong "Qui trình của các chuẩn Internet - sự rà soát lại lần 3" [94] và "Tao của IETF:

11 Tổ chức Quốc tế về Tiêu chuẩn hóa
12 Liên minh Truyền thông Quốc tế
13 Nhóm các công ty về World Wede Web

Một chỉ dẫn mới cho đội đặc nhiệm về kỹ thuật Internet" [136].

Liệu việc sửa đổi các văn bản luật và các điều chỉnh có được cho phép không? Rõ ràng không nếu nó dẫn tới sự nhầm lẫn. Ví dụ, một RFC chỉ phải được sửa đổi để giải thích nó hoặc bổ sung những bình luận làm sáng tỏ, trong khi điều này còn không được phép mà không có một sự cho phép rõ ràng cho những khuyến cáo của W3C (http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231) [65].

Bản thân các giấy phép cũng là các tài liệu pháp lý mà chúng không thể được sửa đổi. Liệu có khả năng để tạo ra những điều chỉnh mới xuất phát từ những giấy phép đang tồn tại có sử dụng các tài liệu gốc hay không? Điều này có thể dẫn tới sự truyền bá thụ động của những điều chỉnh tương tự và không tương thích mà chúng có thể tạo ra sự nhầm lẫn và có thể giúp các công ty mà áp đảo thị trường để khuyến khích những sự biến đổi không tương thích của riêng họ, khi trên thực tế đang xảy ra, đặc biệt trong lĩnh vực của Internet. Mặc dù vậy, nơi mà pháp luật của Nhà nước được quan tâm, thì rất thường xuyên các luật được sao chép theo nghĩa đen từ những người ở các quốc gia khác và được áp dụng với những sửa đổi nhỏ cho các đặc thù của bản địa.

Liệu có một mô hình kinh doanh cho các luật và những điều chỉnh? Có vô số những người chuyên nghiệp mà họ làm việc về các luật, có trách nhiệm về việc thiết kế, phiên dịch và tăng cường cho chúng (các nhà làm luật, các luật sư, các cố vấn pháp lý, các quan tòa, …). Có những phòng thí nghiệm mà cung cấp các chứng chỉ tuân thủ cho những điều chỉnh. Các cơ quan điều chỉnh tồn tại, hoặc phải tồn tại, trong những đóng góp của các thành viên của họ mà những thành viên này mong muốn thúc đẩy các chuẩn, ví dụ, vì việc kinh doanh của họ là dựa vào các sản phẩm mà tương hợp được.

Theo cách y như vậy rằng thuận tiện để có một định nghĩa về *PMTD* hoặc *PMNM*, cũng cần thiết phải có làm việc cho một định nghĩa về các *chuẩn mở*. Bruce Perens (http://perens.org/OpenStandards) [15] đã đề xuất định nghĩa sau đây dựa trên các nguyên lý sau:

1. Tính sẵn sàng: nếu có thể, các chuẩn mở phải là sẵn sàng cho tất cả để đọc và triển khai.
2. Tối đa hóa sự lựa chọn của người sử dụng đầu cuối.
3. Các chuẩn mở phải là tự do cho tất cả để triển khai mà không có chi phí bản quyền hoặc phí nào (các chứng thực về tuân thủ có thể liên quan tới một thứ phí, dù Bruce Perens khuyến cáo rằng nên có các *công cụ tự chứng thực* tự do sẵn sàng).
4. Không có sự phân biệt đối xử có lợi cho một nhà triển khai cài đặt nào so với những người khác.
5. Những quyền mở rộng hoặc phụ thêm (không chứng thực được).
6. Tránh thực tế ăn cắp bởi các nhà sản xuất áp đảo. Tất cả các mở rộng sở hữu độc quyền phải có một triển khai cài đặt theo chuẩn mở.

10.1.3 Bách khoa toàn thư

Vào năm 1999, Richard Stallman đã đề xuất ý tưởng về một bách khoa toàn thư ("Bách khoa toàn thư

tự do và tài nguyên học tập", 2001) [210] như một cơ chế để tránh sự chiếm đoạt tri thức và cung cấp sự truy cập vạn năng tới các tài liệu học tập và có liên quan. Nó có thể được cấu tạo từ các bài báo được cung cấp bởi cộng đồng, mà không có sự kiểm soát tập trung, nơi mà các tác nhân khác nhau có thể nhận làm các nhiệm vụ khác nhau, bao gồm, như một khuyến cáo nhưng không phải là một bổn phận, rằng đối với việc rà soát lại hoặc kiểm tra lại các bài báo. Bách khoa toàn thư này có thể không chỉ chứa văn bản mà còn cả đa phương tiện và các phần mềm giáo dục tự do. Hàng loạt các sáng kiến đã nổi lên để biến điều này thành hiện thực. Ví dụ, Nupedia (http://www.nupedia.com) [178] đã cố gắng xây dựng một bách khoa toàn thư có chất lượng, nhưng dự định này đã thất bại, có lẽ vì nó đã yêu cầu một định dạng mà khá khó để học (TEI), dù có lẽ nhiều hơn là vì yêu cầu về việc có tất cả các bài báo được soạn thảo, được rà soát lại bởi các nhà khoa học và được kiểm tra về kiểu dạng, …

Hậu bối của Nupedia, mà đã thành công hơn nhiều, là Wikipedia (http://www.wikipedia.org) [69]. Wikipedia là một bách khoa toàn thư tự do đa ngôn ngữ dựa trên công nghệ wiki. Wikipedia được viết một cách cộng tác bởi những tình nguyện viên và đa số lớn các bài viết có thể được sửa đổi bởi bất kỳ ai với một trình duyệt web. Sự thành công của nó dựa trên cấu trúc của nó, mà nó là mềm dẻo hơn nhiều về việc soạn thảo, mà nó hạn chế được những trở ngại mà Nupedia đã có và nó làm cho gần gũi hơn với những gì mà Stallman đã có trong đầu. Từ *wiki* tới từ một từ của Haiwai có nghĩa là ('nhanh'). Công nghệ wiki cho phép bất kỳ ai cũng sửa đổi được bất kỳ tài liệu nào có sử dụng hệ thống văn bản có cấu trúc, mà là cực kỳ đơn giản như chúng ta đã thấy trong phần 8.6.2. Vào tháng 02/2007, số lượng bài viết bằng tiếng Anh trên Wikipedia đã là hơn 1,500,000.

Lưu ý

Wikipedia là một dự án của tổ chức phi lợi nhuận Wikimedia, mà nó cũng có các dự án sau, dựa vào cùng mô hình như của Wikipedia:

- Wiktionary (http://www.wiktionary.org) [66]. Đây là một dự án cộng tác mà nó có mục tiêu để tạo ra một từ điển tự do đa ngôn ngữ, với những định nghĩa, từ nguyên học và những phát âm, trong các ngôn ngữ theo yêu cầu.
- Wikibooks (http://www.wikibooks.org/) [67]. Đây là một dự án mà nó có mục tiêu để cung cấp các sách giáo khoa, sổ tay hướng dẫn, sách hướng dẫn hoặc các văn bản sư phạm cho bất kỳ ai yêu cầu các yếu tố này, một cách tự do.
- Wikiquote (http://www.wikiquote.org) [70]. Đây là một sự biên dịch các câu thành ngữ nổi tiếng trong tất cả các ngôn ngữ, mà nó bao gồm cả các nguồn khi những thứ này được biết.
- Wikisource. Đây là một thư viện của các văn bản gốc mà chúng là trong miền công cộng hoặc chúng đã được xuất bản với một giấy phép tài liệu tự do GNU (GFDL).
- Wikispecies (http://species.wikimedia.org/) [71]. Đây là một kho tài liệu về các loại động vật, các loài rau quả, nấm, vi khuẩn và tất cả các dạng sự sống được biết.
- Wikinews (http://wikinews.org/) [68]. Đây là một nguồn của các nội dung tin tự do trong đó người sử dụng là những người soạn thảo.
- Commons (http://commons.wikimedia.org/) [19]. Đây là một kho các nội dung tự do của hình ảnh và đang phương tiện tự do.

- Wikiversity (http://wikiversity.org/) [72]. Đây là một nền tảng giáo dục mở và tự do, dựa trên các dự án dạy học ở tất cả các mức giáo dục..
- Meta-Wiki (http://meta.wikimedia.org/) [48]. Đây là website mà hỗ trợ tất cả các dự án của Quỹ Wikimedia.

Chúng ta cũng nên nhắc tới *Bách khoa toàn thư Ngắn gọn về Toán học*, mà nó có một khái niệm được giới hạn hơn về những gì tự do có nghĩa (nó chỉ có thể được tư vấn trên Internet) và là một mô hình phát triển trong đó cần thiết phải đề xuất tất cả các đóng góp cho một ủy ban soạn thảo trước khi xuất bản.

10.1.4 Các khóa học

Với cùng mục đích như các bách khoa toàn thư, có khả năng sản xuất các tư liệu dạy học tự do, như các ghi chép, slide, bài tập, sách, các phần mềm đọc âm tiết hoặc để dạy học. Có một xu hướng coi các trường đại học như các doanh nghiệp mà chúng sản xuất và bán kiến thức, mà chúng đối nghịch với các nguyên lý cơ bản. Những lý do giải thích vì sao một trường đại học có thể làm ra những tư liệu này sẵn sàng cho tất cả như sau:

- Thỏa mãn nhiệm vụ của trường, như một đại lý phân phối kiến thức.
- Giá thành thấp của các tư liệu đang tồn tại sẵn sàng trên toàn thế giới.
- Thực tế là những tư liệu này không thể thay thế việc dạy học bằng con người.
- Ý tưởng về các tư liệu này được công khai mà có thể lôi cuốn được các sinh viên và đóng góp cho uy tín của đại học.
- Khả năng tạo ra một cộng đồng các giáo viên mà họ rà soát lại và cải tiến các tư liệu này.

Sáng kiến nổi tiếng nhất trong lĩnh vực này là việc của MIT (http://ocw.mit.edu) [174], mà nó có mục tiêu tạo ra nhiều hơn 2,000 nguồn được xếp thành các catalog tốt có thể truy cập được theo một cách thống nhất và mạch lạc.

10.1.5 Các bộ sưu tập và các cơ sở dữ liệu

Chỉ là sự biên dịch của các thông tin tuân theo các tiêu chí được xác định, việc tổ chức và làm cho sẵn sàng, bản thân nó, là một sản phẩm của các thông tin có giá trị, bất chấp bản thân thông tin ra sao, mà vì thế sản phẩm cả của các tác giả các thông tin đó, hệ quả là, tuân theo những hạn chế về sự tự do truy cập, sửa đổi hoặc phân phối lại các nội dung. Vì thế, nếu chúng ta muốn các thông tin tự do, thì chúng ta cũng có thể muốn các bộ sưu tập tự do.

Ví dụ, chúng ta có thể muốn phân loại các thông tin quan trọng trên Internet, tổ chức và bình luận các liên kết. Đây là những gì mà Dự án Thư mục Mở - ODP (Open Directory Project http://dmoz.org [109]) làm; nó được vận hành bởi Netscape và được duy trì bởi những người soạn thảo là các tình nguyện viên

được tổ chức theo một kiến trúc có tôn ti trật tự.

Thư mục đầy đủ có thể được tự do sao chép ở định dạng RDF và được xuất bản với những sửa đổi nhất định nào đó, như Google và nhiều máy tìm kiếm khác làm. Netscape, chủ nhân của thư mục này, đảm bảo cho một "hợp đồng xã hội của Dự án Thư mục Mở" [53] đã tạo cảm hứng cho phát tán Debian (http://www.debian.org/social_contract.html) [106], mà nó tạo điều kiện thuận lợi cho những đóng góp bên ngoài cho việc đảm bảo rằng Dự án Thư mục Mở sẽ luôn là tự do, với các chính sách công cộng, tự điều chỉnh bởi cộng đồng và những người sử dụng như là ưu tiên đầu tiên.

Những ví dụ khác của các bộ sưu tập mà chúng có thể làm cho chúng ta quan tâm là những phát tán PMTD, với những chương trình được sửa đổi sao cho chúng phù hợp cũng sẽ được biên dịch trước một cách tuyệt vời sao cho chúng có thể chạy được một cách dễ dàng.

10.1.6 Phần cứng

Có 2 khía cạnh chính có liên quan trong sự tự do về phần cứng. Khía cạnh đầu tiên là nhu cầu cho các giao diện và các tập hợp lệnh sẽ là tự do, theo một cách mà mọi người có thể tạo ra được một trình điều khiển thiết bị hoặc một trình biên dịch cho một kiến trúc. Khía cạnh thứ 2 là việc phải có đủ thông tin và năng lượng sẵn sàng cho việc tạo lại được một thiết kế phần cứng, việc sửa đổi nó và kết hợp nó với những thứ khác. Những thiết kế này có thể được coi là phần mềm theo một ngôn ngữ phù hợp (VHDL, Verilog, …). Tuy nhiên, việc làm cho chúng làm việc được không dễ dàng, vì chúng phải được sản xuất ra, mà nó là đắt giá và chậm chạp. Tuy nhiên, có những sáng kiến theo nghĩa này, trong số đó chúng ta có thể nhắc tới OpenCores (http://www.opencores.org) [52], cho các mạng tích hợp.

10.1.7 Văn học và nghệ thuật

Để kết thúc việc xem xét của chúng ta về các nguồn tự do, chúng ta không thể quên nghệ thuật và văn học, mà mục tiêu cuối cùng của chúng không là vụ lợi nhiều như là thẩm mỹ.

Những lý do gì có thể để một nghệ sĩ phải trao cho mọi người sự tự do để sao chép, sửa đổi hoặc phân phối lại tác phẩm của họ nhỉ? Một mặt, có thể giúp làm cho họ nổi tiếng và có ích cho sự phổ biến tác phẩm của họ, mà cho phép họ có được thu nhập từ các hoạt động khác, như những sự phối hợp và các tác phẩm ủy thác, và một mặt khác, nó có thể khuyến khích sự thử nghiệm và tính sáng tạo. Trong nghệ thuật, chúng ta có những tình huống y hệt như trong các vấn đề kỹ thuật. Sự đổi mới sáng tạo là từng chút một và đôi khi khó để phân biệt được giữa sự ăn cắp ý tưởng và một tác phẩm mà đại diện hoặc tuân theo một xu thế hoặc trào lưu nghệ thuật nào đó.

Rõ ràng, sự sáng tạo và sự làm sáng tỏ là không phải cùng một thứ, và không phải là âm nhạc hay văn học. Âm nhạc, hội họa, nhiếp ảnh và điện ảnh là rất tương tự như các chương trình, theo nghĩa là chúng có thể được làm để "làm việc" ngay lập tức trên một máy tính, trong khi thứ y như vậy không thể áp dụng được cho điêu khắc, ví dụ vậy. Không có nhiều những sáng kiến nguồn mở trong nghệ thuật và văn học trong khi những sáng kiến mà đang tồn tại thì lại rất khác nhau. Chúng ta có thể nhắc tới những tiểu thuyết của tập thể Wu Ming (http://www.wumingfoundation.com) [29].

10.2 Các giấy phép cho những tài nguyên tự do khác

Các giấy phép cho PMTD từng là một nguồn của sự truyền cảm hứng cho những nguồn trí tuệ khác, theo cách mà nhiều trong số chúng đã từng được áp dụng một cách trực tiếp, đặc biệt nơi mà các tài liệu được quan tâm, và trong những trường hợp khác, chúng từng được áp dụng khá yếu, như xảy ra với Giấy phép Âm thanh Mở - Open Audio License (http://www.eff.org/IP/Open_licenses/eff_oal.html) [114]. Hầu hết những giấy phép này là các giấy phép *copyleft*, nếu chúng cho phép các công việc dẫn xuất.

Giấy phép tài liệu tự do của GNU (xem phần 10.2.1) đã từng được sử dụng và thường được sử dụng cho tất cả mọi dạng văn bản, dù các giấy phép Chung Sáng tạo - Creative Commons (xem phần 10.2.2) dần dần đang được áp dụng.

Trên thực tế, các giấy phép cho các chương trình (GPL và LGPL) còn được sử dụng cho phần cứng, dù đối tượng này là phức tạp và khó để hài hòa với luật pháp hiện hành. Về ảnh hưởng, thì những thiết kế và biểu đồ có thể được sử dụng, mà không có việc sao chép một cách vật lý, để trích ra những ý tưởng được sử dụng cho những thiết kế đóng mới. Ví dụ, Giấy phép Công cộng Chung Phần cứng OpenIPCore ("OpenIPCore hardware general public license") [155] thiết lập rằng sự chiếm hữu này là không được phép, mà tính đúng đắn về pháp lý của tài liệu là còn đáng ngờ [209]. Cách duy nhất có thể đối với việc bảo vệ những ý tưởng này là việc sử dụng một số dạng của bằng sáng chế tự do, mà nó là thứ gì đó mà còn chưa được phát triển và chưa với tới được những người mà không có dự định hoặc không có khả năng thiết lập một doanh nghiệp được xây dựng trên những ý tưởng này.

10.2.1 Giấy phép tài liệu tự do GNU

Một trong những giấy phép *copyleft* nổi tiếng nhất cho tài liệu kỹ thuật, bất kể nó phù hợp với các chương trình hay bất kỳ vấn đề nào khác, thì nó vẫn là của FSF. Sau khi nhận thức được rằng một tài liệu không phải là y hệt như một chương trình, Richard Stallman đã thúc đẩy một giấy phép cho các tài liệu mà đi với các chương trình và cho các tài liệu khác mà bản chất tự nhiên của nó là về kỹ thuật và để dạy học.

Để làm ổn thỏa cho sự phát triển của các phiên bản dẫn xuất, một bản sao minh bạch của tài liệu phải được làm cho sẵn sàng cho bất kỳ ai cần nó, như được giải thích trong phần 3.2.5, cũng như các bản sao còn *mờ đục*, theo một cách tương ứng giữa mã nguồn và các đối tượng của các chương trình.

Một trong những lý do cho việc có một giấy phép là để thiết lập quyền tác giả và để đảm bảo rằng các ý tưởng hoặc ý kiến được trình bày bởi tác giả sẽ không bị mô tả sai về các đặc điểm. Điều này giải thích vì sao các công việc dẫn xuất phải có một đầu đề ở bài khác với các phiên bản trước (trừ phi quyền thể hiện đã được trao) và phải nói lên một cách rõ ràng nơi mà bản gốc có thể lấy được. Các cái tên của những tác giả chính của các tài liệu gốc cũng phải được liệt kê, cũng như những cái tên của những người mà họ đã làm ra bất kỳ sửa đổi nào, và tất cả những ghi chú về sở hữu trí tuệ phải được giữ lại. Cũng vậy, bất kỳ sự thừa nhận và sự cống hiến nào cũng phải được giữ lại và phần về lịch sử, nếu có, phải được tôn trọng khi những sửa đổi mới được bổ sung vào. Còn có thể, và điều này là sự tôn trọng

giấy phép mà bị chỉ trích nhiều nhất, để chỉ định ra những phần không thay đổi và các văn bản bao trùm, mà không ai có thể sửa hoặc giới hạn được, dù giấy phép này chỉ cho phép các văn bản *phi kỹ thuật* được coi như các phần bất biến, mà các phần này thì giấy phép tham chiếu tới như các phần *thứ cấp*.

Giấy phép này đã tạo ra nhiều tranh cãi trong thế giới của PMTD, ở điểm rằng dự án của phát tán Debian hiện hành (vào thời điểm cuốn sách này được xuất bản) đang tranh luận liệu để loại bỏ khỏi Debian các nội dung theo giấy phép này hoặc chỉ định tất cả các tài liệu mà có giấy phép như là không tự do và coi chúng như là không chính thức chăng. Ngay cả dù không có các phần bất biến, thì vì các công việc dẫn xuất có thể tuân theo những khái niệm của giấy phép y hệt, là quan trọng để ghi nhớ rằng chúng có thể được bổ sung vào sau đó. Ví dụ, được tranh luận, rằng có thể các phần bất biến đã lỗi thời hoặc không đúng hoặc, mà, dù thế, cũng phải được giữ lại. Trong bất kỳ trường hợp nào, thì giấy phép này là không tương thích với các chỉ dẫn về PMTD của Debian (http://www.debian.org/social_contract.html#guidelines) [104], mà câu hỏi này xoay quanh có lẽ về việc liệu tài liệu có phải tuân theo những chỉ dẫn này hay không (ví dụ, các văn bản của các giấy phép cũng không thể được sửa đổi).

Khuyến cáo

Các phiên bản đầu của văn bản này là theo giấy phép GFDL, nhưng các tác giả sau đó đã quyết định sử dụng một giấy phép Chung Sáng tạo (xem phần 10.2.2), mà nó là phù hợp hơn cho những đặc tính của một cuốn sách. Vì thế văn bản này là một tác phẩm có 2 giấy phép.

10.2.2 Các giấy phép chung sáng tạo

Chung Sáng tạo - Creative Commons (http://creativecommons.org) [21] là một tổ chức phi lợi nhuận mà đã được thành lập vào năm 2001 bởi các chuyên gia về sở hữu trí tuệ và pháp luật trong xã hội thông tin, với mục tiêu để thúc đẩy sự sáng tạo, sự gìn giữ và tính có thể truy cập được của các nguồn tri thức được nhượng lại cho cộng đồng theo nhiều cách. Nó dựa trên ý tưởng rằng một số người có thể không muốn sử dụng tất cả các quyền sở hữu trí tuệ mà pháp luật đã trao cho họ, khi điều này có thể gây cản trở cho sự phổ biến chúng một cách rộng rãi.

Các giấy phép Chung Sáng tạo đầu tiên cho các công việc sáng tạo, mà chúng là có nhiều phiên bản, bắt đầu xuất hiện khoảng cuối năm 2002. Những giấy phép này đã được thiết kế để:

- Đủ mạnh để chịu được một sự soi xét của tòa án, tại nhiều quốc gia.
- Đủ đơn giản cho những người không phải là luật sư để sử dụng.
- Đủ phức tạp để được xác định bởi các ứng dụng web khác nhau.

Các giấy phép khác nhau cho phép người sáng tạo lựa chọn những dạng tự do nào được phép, tách biệt khỏi việc sao chép, tuân theo với 4 điểm:

Trong phiên bản 1.x của các giấy phép Chung Sáng tạo, đã có 11 dạng giấy phép, mà chúng đã kết hợp 4 đặc tính cơ bản được nhắc tới ở trên. 98% các tác giả chọn lựa chọn "thẩm quyền"; kết quả là, như của phiên bản 2.x của các giấy phép Chung Sáng tạo, thẩm quyền là một yêu cầu. Điều này làm giảm 11 dạng giấy phép xuống còn 6, là như sau:

Bảng sau đây chỉ ra một phác họa của các giấy phép với những biểu tượng tương ứng. Biểu tượng này thường là một liên kết tới một kết luận của giấy phép, được quản lý trên website của Creative Commons [21].

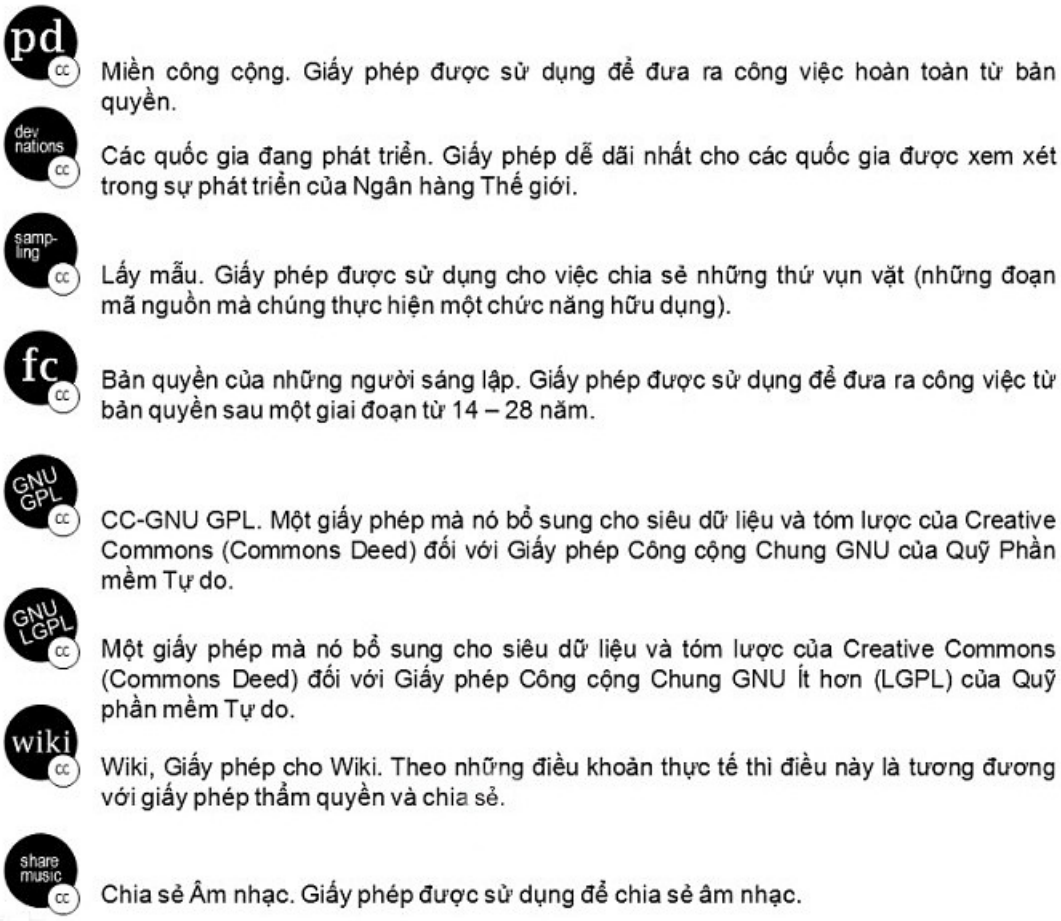
Có khả năng sử dụng biểu tượng[14] chung thay vì biểu tượng đại diện cho giấy phép, nhưng nó phải được liên kết tới giấy phép được chọn bởi tác giả. Mã HTML của đường liên kết này tới giấy phép có thể có được từ Creative Commons [21]. Một khi giấy phép được chọn và biểu tượng tương ứng được bổ sung, thì công việc này sẽ được cấp phép và bạn sẽ nhận được:

- *Chứng thư chung*. Một tóm tắt về giấy phép với những biểu tượng phù hợp cho nó. Tóm tắt này sẽ được chỉ ra khi nháy vào đường liên kết có được từ Chung Sáng tạo (Creative Commons) [21].
- *Mã pháp lý*. Đây là văn bản pháp lý đầy đủ về những gì mà giấy phép dựa vào. Văn bản này có thể truy cập được từ tóm tắt được nêu ở trên.
- *Mã số*. Đây là mô tả RDF (khung công việc mô tả nguồn), mà các máy tìm kiếm và các ứng dụng khác có thể sử dụng để xác định giấy phép và các điều khoản sử dụng.

Vào tháng 02/2007, phiên bản 3.0 của các giấy phép Chung Sáng tạo đã được xuất bản. Đây là một cập nhật mà nó sửa cho đúng nhiều lỗi mà mọi người đã xác định. Sự sửa đổi lớn đầu tiên là việc giấy phép chung không còn dựa vào mô hình của Mỹ và bây giờ dựa vào thuật ngữ của Công ước Berne. Thứ hai là, các quyền về đạo đức và các xã hội quản lý bằng các quyền được nhắc tới một cách đặc biệt, như những sự thống trị khác nhau đã được thực hiện trong từng phạm vi quyền hạn. Thứ ba và là cuối cùng, các văn bản của cả *chứng thư chung* và *mã pháp lý* mà đã đi với từng giấy phép đã được sửa đổi để làm cho nó rõ ràng rằng câu chữ trong sự thừa nhận quyền tác giả không cho phép người được cấp phép có ngụ ý hoặc đưa ra cảm giác rằng họ có một mối quan hệ hoặc có liên quan theo bất kỳ cách gì tới người cấp phép.

Để bổ sung, Chung Sáng tạo cung cấp các dạng giấy phép khác cho các ứng dụng đặc thù. Như là:

14 Xem hình



Không phải tất cả các giấy phép Chung Sáng tạo được coi là tự do bởi các khu vực có liên kết tới PMTD, như 4 quyền tự do cơ bản phải áp dụng trước khi các giấy phép được xác định như (xem phần 1.1.1) Benjamin "Kako" Hill (lập trình viên của Debian và Ubuntu) đã tạo ra website Freedomdefined.org (http://freedomdefined.org/) [28], mà nó có mục tiêu cung cấp một định nghĩa tốt hơn của những gì là văn hóa tự do và những gì là không phải. Trên cơ sở này, trong số 6 giấy phép cơ bản của Chung Sáng tạo, chỉ có 2 là tự do một cách nghiêm túc: chỉ Thẩm quyền (BY) và Thẩm quyền - Chia sẻ (BY-SA), thì cái sau của nó cũng có *copyleft*.

Thư mục tham khảo

[1] Aap Project: http://www.a-a-p.org

[2] Ada Core Technologies: http://www.gnat.com/

[3] Alcôve: http://www.alcove.com

[4] Alcôve-Labs: http://www.alcove-labs.org

[5] Alioth: http://alioth.debian.org

[6] Anjuta: http://www.anjuta.org

[7] The Apache Ant Project: http://ant.apache.org

[8] Arch Revision Control System: http://www.gnu.org/software/gnu-arch/

[9] artofcode LLC: http://artofcode.com/

[10] Autoconf: http://www.gnu.org/software/autoconf

[11] Barrapunto: http://barrapunto.com

[12] Bazaar GPL Distributed Version Control Software: http://bazaar-vcs.org/

[13] Berlios. The Open Source Mediator: http://berlios.de

[14] Bitkeeper Source Management: http://www.bitkeeper.com

[15] Bruce Perens: http://perens.com/OpenStandards/Definition.html

[16] Caldera: http://www.sco.com

[17] Cisco Enterprise Print System: http://ceps.sourceforge.net/

[18] Code::blocks: http://www.codeblocks.org

[19] Commons: http://commons.wikimedia.org/

[20] Concurrent Version System: http://ximbiot.com/cvs/

[21] Creative Commons. http://creativecommons.org

[22] Directory of Open Access Journals: http://www.doaj.org

[23] Eclipse - An Open Development Platform: http://www.eclipse.org

[24] eCos: http://sources.redhat.com/ecos/

[25] eCos license 2.0: http://www.gnu.org/licenses/ecos-license.html

[26] First Monday. Peer Rewiewed Journal on the Internet: http://firstmonday.org

[27] Free Software Foundation: http://www.fsf.org

[28] Freedom Defined (Free Cultural Works): http://freedomdefined.org/

[29] Fundación Wu Ming: http://www.wumingfoundation.com

[30] GForge: http://gforge.org

[31] Gettext: http://www.gnu.org/software/gettext

[32] GNU Automake: http://www.gnu.org/software/automake

[33] GNU Emacs: http://www.gnu.org/software/emacs/

[34] GNU Libc: http://www.gnu.org/software/libc

[35] GNU Libtool: http://www.gnu.org/software/libtool

[36] GNU Make: http://www.gnu.org/software/make/make.html

[37] GNU Troff: http://www.gnu.org/software/groff/groff.html

[38] "Have you seen these hackers?": http://www.mozilla.org/MPL/missing.html

[39] "History of TeX": http://www.math.utah.edu/software/plot79/tex/history.html

[40] IBM Public License Version 1.0: http://opensource.org/licesenses/ibmpl.php

[41] Jam Product Information: http://www.perforce.com/jam/jam.html

[42] KDevelop: http://www.kdevelop.org

[43] Launchpad: https://launchpad.net

[44] The Linux Documentation Project: http://www.tldp.org

[45] LinuxCare: http://www.levanta.com

[46] Mailman, the GNU Mailing List Manager: http://www.list.org

[47] The Malone Bug Tracker: https://launchpad.net/products/malone

[48] Metawiki: http://meta.wikimedia.org/

[49] Mozilla Public License 1.1: http://www.mozilla.org/MPL/MPL-1.1.html

[50] Mozilla Tinderbox: http://www.mozilla.org/tinderbox.html

[51] NetBeans: http://www.netbeans.org

[52] Open Cores: http://www.opencores.org

[53] Open Directory Project Social Contract:

[54] Open Source Initiative: http://www.opensource.org

[55] Public Library of Science: http://www.publiclibraryofscience.org

[56] Red Hat: http://www.redhat.com

[57] Savannah: http://savannah.gnu.org and http://savannah.nongnu.org

[58] Slashdot: News for Nerds. http://slashdot.org

[59] Sleepycat License: http://www.sleepycat.com/download/oslicense.html

[60] Sleepycat Software: http://www.sleepycat.com/

[61] SourceForge: Open Source Software Development Website: http://sourceforge.net

[62] Subversion: http://subversion.tigris.org

[63] Texinfo - The GNU Documentation System:

[64] Tigris.org: Open Source Software Engineering: http://tigris.org

[65] W3c Document License:
http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231

[66] Wiktionary: http://www.wiktionary.org

[67] Wikibooks: http://www.wikibooks.org/

[68] Wikinews: http://wikinews.org/

[69] Wikipedia: http://www.wikipedia.org

[70] Wikiquote: http://www.wikiquote.org

[71] Wikispecies: http://species.wikimedia.org/

[72] Wikiversity: http://wikiversity.org/

[73] X Window System Release 11 License: http://www.x.org/Downloads_terms.html

[74] Ximian: http://www.novell.com/linux/ximian.html

[75] Zope Corporation: http://www.zope.com/

[76] Zope Public License 2.0: http://www.zope.org/Resources/ZPL

[77] Law on Intellectual Property. Spanish Royal Legislative Decree 1/1996, of 12th April (April 1996):

[78] Affero General Public License, 2002: http://www.affero.org/oagpl.html

[79] Law on Intellectual Property. Spanish Law 23/2006, of 7th July (July 2006):

[80] Flossimpact Study. Technical Report, European Comission, 2007: http://flossimpact.eu

[81] ISO JTC 1/SC 34. Standard Generalised Markup Language (SGML, ISO 8879), 1986:

[82] Antoniades, I.; Samoladas, I.; Stamelos, I.; Bleris, G. L. "Dynamical simulation models of the open source development process" En: Koch [157]. http://wwwai.wu-wien.ac.at/~koch/oss-book/

[83] Bailey, E. C. (1998). Maximum RPM. Taking the Red Hat package manager to the limit. http://rikers.org/rpmbook/

[84] González Barahona, J. M. (2000). "Software libre, monopolios y otras yerbas". Todo Linux (3). http://sinetgy.org/~jgb/articulos/soft-libre-monopolios/

[85] González Barahona, J. M. (2002). "¿Qué se hace con mi dinero?". Todo Linux (17). http://sinetgy.org/~jgb/articulos/sobre-administracion/

[86] González Barahona, J. M.; Robles, G. Libre Software Engineering Web Site. http://libresoft.dat.escet.urjc.es/

[87] González Barahona, J. M.; Robles, G. (2003, mayo). "Unmounting the code god assumption". En: Proceedings of the Fourth International Conference on eXtreme Programming and Agile Processes in Software Engineering. Genoa, Italy.

[88] González Barahona, J. M.; Robles, G.; Ortuño Pérez, M. A.; Rodero Merino, L.; Centeno González, J.; Matellán Olivera, V.; Castro Barbero, E. M.; De las Heras Quirós; P. "Anatomy of two GNU/Linux distributions". En: Koch [157]. http://wwwai.wu-wien.ac.at/~koch/oss-book/

[89] Barnson, M. P. The Bugzilla guide. http://www.bugzilla.org/docs214/html/index.html

[90] Baudis, P. "Cogito manual page". http://www.kernel.org/pub/software/scm/cogito/docs/

[91] Bezroukov, N. (1998, diciembre). "A second look at the cathedral and the bazaar". First Monday, 4(12). http://www.firstmonday.org/issues/issue4_12/bezroukov/index.html

[92] Bodnar, L. (2003). "Linux distributions. Facts and figures". http://www.distrowatch.com/stats.php?section=packagemanagement

[93] Boehm, B. W. (1981). Software Engineering Economics. Prentice Hall.

[94] Bradner, S. (1996, october). "The Internet standards process. Revision 3 (rfc 2026, bcp 9)".
http://www.ietf.org/rfc/rfc2026.txt
[95] Cederqvist, P.; GNU (1993). "CVS - concurrent versions system". http://www.gnu.org/manual/cvs/index.html
[96] Collins-Sussman, B.; Fitzpatrick; B. W.; Pilato, C. M. (2004). Version control with Subversion. O'Reilly & Associates (http://www.ora.com).
http://svnbook.red-bean.com/
[97] Cunningham, W. "Wiki design principles".
[98] Dachary, L. (2001). "Savannah, the next generation".
http://savannah.gnu.org/docs/savannah-plan.html
[99] Autonomous Government of Andalucía (2003, March). Decree 72/2003, of 18th March, on Measures to Promote the Knowledge Society in Andalucía.
http://www.andaluciajunta.es/SP/AJ/CDA/Ficheros/ArchivosPdf/DecretoConocimiento.pdf
[100] De Boor, A. Pmake. A tutorial. http://docs.freebsd.org/44doc/psd/12.make/paper.html
[101] De Icaza, M. "The story of the GNOME Project".
http://primates.ximian.com/~miguel/gnome-history.html
[102] Senate of the Republic of France. Forum sur la proposition de loi tendant à généraliser dans l'administration l'usage d'Internet et de logiciels libres.
http://www.senat.fr/consult/loglibre/index.htm
[103] De las Heras Quirós, P.; González Barahona, J. M. (2000). "Iniciativas de las administraciones públicas en relación al software libre". Bole. TIC, ASTIC magazine (14).
[104] Debian. "Debian free software guidelines".
http://www.debian.org/social_contract.html#guidelines
[105] Debian. Debian policy manual.
http://www.debian.org/doc/debian-policy/
[106] Debian. "Debian social contract".
http://www.debian.org/social_contract.html

[107] Schriftenreihe der KBSt (2003, July). Leitfaden für die migration von basissoftwarekomponenten auf serverund arbeitsplatzsystemen. Technical report, Koordinierungsund Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (KBSt).
http://www.kbst.bund.de/download/mlf_v1_de.pdf
[108] DiBona, C.; Ockman, S.; Stone, M. (ed.) (1999). Open sources. Voices from the open source revolution. O'Reilly & Associates.
http://www.oreilly.com/catalog/opensource/
[109] Open Directory Project. http://dmoz.org
[110] Ehrenkrantz, J. R. (2003, May). "Release management within open source projects". In: Proceedings of the 3rd Workshop on Open Source Software Engineering at the 25th International Conference on Software Engineering. Portland, USA
[111] European Council (1991). Council Directive 91/250/CEE of 14th May 1991, on the legal protection of computer programs.
http://europa.eu.int/scadplus/leg/es/lvb/l26027.htm
[112] Feller, J.; Fitzgerald, B; Hissam, S.; Lakhani, K. (ed.) (2003). Making sense of the bazaar. O'Reilly.
[113] Fogel, K.; Bar, M. (2001). Open source code development with CVS (2nd edition). Paragliph Press.
http://cvsbook.red-bean.com
[114] Electronic Frontier Foundation. Open Audio.
http://www.eff.org/IP/Open_licenses/eff_oal.html
[115] Free Software Foundation. GPLv3.
http://gplv3.fsf.org
[116] Free Software Foundation. LGPLv3. First discussion draft.
http://gplv3.fsf.org/pipermail/info-gplv3/2006-July/000008.html
[117] Free Software Foundation (1985): "The GNU Manifesto".
http://www.gnu.org/philosophy/

[118] Free Software Foundation (1991, junio). GNU General Public License, version 2.
http://www.fsf.org/licenses/gpl.html
[119] Free Software Foundation (1999, February). GNU Lesser General Public License, version 2.1.
http://www.fsf.org/licenses/lgpl.html
[120] Free Software Foundation. "Free software definition".
http://www.gnu.org/philosophy/free-sw.html
[121] Free Software Foundation. "Free licenses".
http://www.gnu.org/licenses/license-list.html
[122] Garbee, B.; Koptein, H.; Lohner, N.; Lowe, W.; Mitchell, B.; Murdock, I.; Schulze, M.; Small, C. "A brief history of Debian". In the package: Debian-history.
[123] Germán, D. (2002, May). "The evolution of GNOME". In: Proceedings of the 2nd Workshop on Open Source Software Engineering at the 24th International Conference on Software Engineering. Florida, USA
[124] Germán, D.; Mockus, A. (2003, May): "Automating the measurement of open source projects". In: Proceedings of the 3rd Workshop on Open Source Software Engineering at the 25th International Conference on Software Engineering. Portland, USA
[125] Ghosh, R. A. (1998, March). "Cooking pot markets: an economic model for the trade in free goods and services on the Internet. First Monday, 3(3).
http://www.firstmonday.dk/issues/issue3_3/ghosh/index.html
[126] Ghosh, R. A.; Glott, R.; Krieger, B.; Robles, G. (2002). Free/libre and open source software: Survey and study. Part iv: "Survey of developers".
http://www.infonomics.nl/FLOSS/report/FLOSS_Final4.pdf
[127] Ghosh, R. A.; Prakash, V. V. (2000, July). "The orbiten free software survey". First Monday, 5(7).
http://www.firstmonday.dk/issues/issue5_7/ghosh/index.html
[128] Godfrey, M. W.; Tu, Q. (2000, August). "Evolution in open source software. A case study". In: Proceedings of the 2000 International Conference on Software Maintainance.

[129] González, J. A. (2002, March). "Carta al congresista Villanueva".
http://www.gnu.org. pe/mscarta.html
[130] Goosens, M.; Rahtz, S. (1999). The LaTeX Web Companion. Addison Wesley.
[131] Grad, B. (2002, January-March). "A personal recollection: IBM's unbundling of software and services". In: IEEE Annals of the History of Computing, 24(1):64-71.
[132] Working Group on Libre Software (1999). "Free software / open source. Information society opportunities for Europe?".
http://eu.conecta.it/paper.pdf
[133] GrULIC. "Legislation on the use of free software by the State ".
http://proposicion.org.ar/doc/referencias/index.html.es
[134] Hamerly, J; Paquin, T.; Walton, S. (1999). "Freeing the source. The story of Mozilla".
http://www.oreilly.com/catalog/opensources/book/netrev.html
[135] Hammel, M. J. (1991, December). "The history of xfree86". Linux Magazine.
http://www.linux-mag.com/2001-12/xfree86_01.html
[136] Harris, S. (2001, August). The Tao of IETF. A novice's guide to the Internet engineering task force (RFC 3160, FYI 17).
http://www.ietf.org/rfc/rfc3160.txt
[137] Harrison, P. (2002). "The rational street performer protocol".
http://www.logarithmic.net/pfh/RSPP
[138] Hasan, R. "History of Linux".
http://ragib.hypermart.net/linux/
[139] Hauben, M.; Hauben, R. (1997). Netizens. On the history and impact of Usenet and the Internet. IEEE Computer Society Press.
[140] Healy, K.; Schussman; A. (2003, January). "The ecology of open source software development". http://opensource.mit.edu/papers/healyschussman.pdf
[141] Hecker, F. (1998, May). "Setting up shop. The business of open-source software".
http://www.hecker.org/writings/setting-up-shop.html
[142] Hecker, F. (1998). "Setting up shop. The business of open-source software".

http://www.hecker.org/writings/setting-up-shop.html
[143] Hertel, G.; Niedner, S.; Herrmann, S. (2003). "Motivation of software developers in open source projects. An Internet-based survey of contributors to the Linux kernel". http://opensource.mit.edu/papers/rp-hertelniednerherrmann.pdf
[144] Himanen, P. (2001). The hacker ethic and the spirit of the information age. Random House.
http://www.hackerethic.org
[145] Hunt, F.; Johnson, P. (2002). "On the Pareto distribution of SourceForge projects. Technical report". Centre for Technology Management, Cambridge University Engineering Department, Mill Lane, Cambridge CB2 1RX.
http://www-mmd.eng.cam.ac.uk/people/fhh10/Sourceforge/Sourceforge%20paper.pdf
[146] Open Source Initiative. "History of the OSI".
http://www.opensource.org/docs/history.php
[147] Hamilton, J. R. (US ambassador to Peru) (2002, June). "Carta al presidente del Congreso de la República".
http://www.gnu.org.pe/lobbyusa-congreso.html
[148] Jones, P. (2000, May). "Brook's law and open source. The more the merrier?".
http://www-106.ibm.com/developerworks/opensource/library/osmerrier.html?dwzone=opensource
[149] Jorgensen, N. "Incremental and decentralized integration in FreeBSD". In: Feller et al. [112]. http://www.dat.ruc.dk/~nielsj/research/papers/bazaar-freebsd.pdf
[150] Brooks, F. P. (1975). The mythical man-month. Essays on software engineering. Addison-Wesley.
[151] Kalt, C. (2000, April). "Internet relay chat: architecture (RFC 2810)".
http://www.ietf.org/rfc/rfc2810.txt
[152] Kelsey, J.; Schneier, B. (1998, November). "The street performer protocol". In: Third USENIX Workshop on Electronic Commerce Proceedings. USENIX Press.
http://www.counterpane.com/street_performer.html

[153] Kelsey, J.; Schneier, B. (1999, June). "The street performer protocol and digital copyrights". First Monday, 4(6).
http://www.firstmonday.dk/issues/issue4_6/kelsey/
[154] Kelty, C. M. (2001, December). "Free software/free science". First Monday, 6(12).
http://firstmonday.org/issues/issue6_12/kelty/index.html
[155] Khatib, J. "OpenIPCore Hardware General Public License".
http://www.opencores.org/OIPC/OHGPL_17.shtml
[156] Knuth, D. (1989). The TeXbook. Addison Welsley.
[157] Koch, S. (ed.) (2003). Free/open source software development. Idea Group Inc.
http://wwwai.wu-wien.ac.at/~koch/oss-book/
[158] Koch, S.; Schneider, G. (2000). "Results from software engineering research into open source development projects using public data". In: Diskussionspapiere zum Tätigkeitsfeld Informationsverarbeitung und Informationswirtschaft, H.R. Hansen und W.H. Janko (Hrsg.), Nr. 22, WirtschaftsuniversitätWien.
[159] Kovács, G. L.; Drozdik, S.; Succi, G.; Zuliani, P. (2004). "Open source software for the public administration". In: Proceedings of the 6th International Workshop on Computer Science and Information Technologies (CIST 2004). Budapest, Hungary.
[160] Krishnamurthy, S. (2002, May). "Cave or community? An empirical examination of 100 mature open source projects". First Monday, 7(6).
http://www.firstmonday.dk/issues/issue7_6/krishnamurthy/index.html
[161] Laffitte; Trégouet; Cabanel (1999). Proposition de loi numéro 495. Senate of the Republic of France.
http://www.senat.fr/consult/loglibre/texteloi.html
[162] Laffitte; Trégouet; Cabanel (2000). Proposition de loi numéro 117. Senate of the Republic of France.
http://www.senat.fr/consult/loglibre/texteloi.html
[163] Lamport, L. (1994). LaTeX user's guide and reference manual (2nd edition). Addison Welsley, Reading, Mass.

[164] Lancashire, D. (2001, December). "Code, culture and cash. The fading altruism of open source development". First Monday, 6(12). http://www.firstmonday.dk/issues/issue6_12/lancashire/index.html

[165] Lehman, M. M.; Ramil, J. F; Wernick, P. D. (1997, November). "Metrics and laws of software evolution. The nineties view". In: Proceedings of the 4th International Symposium on Software Metrics. http://www.ece.utexas.edu/~perry/work/papers/feast1.pdf

[166] Leiner, B. M.; Cerf, V. G.; Kahn, R. E.; Clark, D. D.; Kleinrock, L.; Lynch, D. C.; Postel, J.; Roberts, L. G.; Wolff, S. (1997). "A brief history of the Internet". In: Communications of the ACM. http://www.isoc.org/internet/history/brief.shtml

[167] Netcraft Ltd. August 2003 Web Server Survey, 2003. http://news.netcraft.com/archives/2003/08/01/august_2003_web_server_survey.html

[168] Lucovsky, M. (2000). "From NT OS/2 to Windows 2000 and beyond. A software-engineering odyssey". http://www.usenix.org/events/usenix-win2000/invitedtalks/lucovsky_html/>

[169] McGraw, G. "Building secure software: how to avoid security problems the right way". Cited by: David A. Wheeler in http://www.dwheeler.com/sloc/

[170] McKusick, M. K. (1999). "Twenty years of Berkeley Unix. From AT&T owned to freely redistributable". In: DiBona et al. [108]. http://www.oreilly.com/catalog/opensources/

[171] SUN Microsystems (2000). "Sun microsystems announces availability of StarOffice source code on OpenOffice.org". http://www.collab.net/news/press/2000/openoffice_live.html

[172] Mockus, A.; Fielding, R. T.; Herbsleb, J. D. (2000, June). "A case study of open source software development: the Apache server". In: Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000), pages 263272. Limerick, Ireland ACM Press.

[173] Molenaar, B. "What is the context of charityware?".

http://www.moolenaar.net/Charityware.html
[174] MIT OpenCourseWare.
http://ocw.mit.edu
[175] Nagel, L. W. (1996, september). "The life of SPICE". In: 1996 Bipolar Circuits and Technology Meeting. Minneapolis, MN, US
http://www.icsl.ucla.edu/aagroup/Life%20of%20SPICE.html
[176] Narduzzo, A.; Rossi, A. (2003, May). "Modularity in action: GNU/Linux and free/ open source software development model unleashed".
http://opensource.mit.edu/papers/narduzzorossi.pdf
[177] Newman, N. (1999). "The origins and future of open source software".
http://www.netaction.org/opensrc/future/
[178] Nupedia.
http://www.nupedia.com
[179] Villanueva Núñez, E. (2002, April). "Letter to Microsoft Peru".
http://www.gnu.org.pe/rescon.html
[180] Danish Board of Technology (2002, October). "Open-source software in e-Government, analysis and recommendations drawn up by a working group under the danish board of technology. Technical report".
[181] Open Source Initiative. "Open source licenses".
http://www.opensource.org/licenses/index.html
[182] Pareto, W. (1896). "Course of Political Economy". Lausanne.
[183] Perens, P.; The Open Source Initiative (1998). "The open source definition". http:/ /www.opensource.org/docs/definition_plain.html
[184] GNU Peru. "Proyectos ley de software libre en la Administración pública del Gobierno peruano, Congreso de la República".
http://www.gnu.org.pe/proleyap.html
[185] Pinheiro, P. (1999, December). Proposição pl-2269/1999: Dispõe sobre a utilização de programas abertos pelos entes de direito público e de direito privado sob controle acionário

da administração pública. Câmara dos Deputados do Brasil.
http://www.camara.gov.br/Internet/sileg/Prop_Detalhe.asp?id=17879
http://www.fenadados.org.br/software.htm
[186] Pranevich, J. (2003). "The wonderful world of Linux 2.6".
http://www.kniggit.net/wwol26.html
[187] The Debian Project. "Debian developer map".
http://www.debian.org/devel/developers.loc
[188] Puigcercós Boixassa, J. (2002). Draft Bill on Measures for Implementing Free Software in Public Administration.
http://www.congreso.es/public_oficiales/L7/CONG/BOCG/B/B_244-01.PDF
[189] Quittner, J.; Slatalla, M. (1998). Speeding the net: the inside story of Netscape and how it challenged Microsoft. Atlantic Monthly Pr.
[190] Rasch, C. "A brief history of free/open source software movement".
http://www.openknowledge.org/writing/open-source/scb/brief-open-source-history.html
[191] Rasch, C. (2001, May). "The Wall Street performer protocol. Using software completion bonds to fund open source software development". First Monday, 6(6).
[192] Raymond, E. R. (2001, January). The cathedral and the bazaar. Musings on Linux and open source by an accidental revolutionary. O'Reilly & Associates (http://www.ora.com).
http://catb.org/~esr/writings/cathedral-bazaar/
[193] Reis, C R.; De Mattos Fortes, R. P. (2002, February). "An overview of the software engineering process and tools in the Mozilla Project".
http://opensource.mit.edu/papers/reismozilla.pdf
[194] Rideau, F. R. (2000). "Patents are an economic absurdity".
http://fare.tunes.org/articles/patents.html
[195] Roberts, L. (1978, November). "The evolution of packet switching". Proceedings of the IEEE, (66).
[196] Robles, G.; González Barahona, J. M.; Centeno González, J.; Matellán Olivera, V.; Rodero Merino, L. (2003, May). "Studying the evolution of libre software projects

using publicly available data". In: Proceedings of the 3rd Workshop on Open Source Software Engineering

at the 25th International Conference on Software Engineering. Portland, US.

[197] Robles, G.; Scheider, H.; Tretkowski, I.; Weber, N. (2001): "Who is doing it?

Knowing more about libre software developers".

http://widi.berlios.de/paper/study.pdf

[198] Rochkind, M. (1986, May). "Interview with Dick Haight". Unix Review.

[199] Scacchi, W. (2003). "Understanding open source software evolution. Applying, breaking

and rethinking the laws of software evolution".

http://www.ics.uci.edu/~wscacchi/Papers/New/Understanding-OSS-Evolution.pdf

[200] Schneier, B. (2000). "Software complexity and security".

http://www.counterpane.com/crypto-gram-0003.html

[201] Smoogen, S. J. "The truth behind Red Hat names".

http://www.smoogespace.com/documents/behind_the_names.html

[202] Haggen So. "Comparison of free/open source hosting (FOSPhost) sites available for

hosting projects externally from project owners".

http://www.ibiblio.org/fosphost/exhost.htm

[203] Stallman, R. "GNU coding standards".

http://www.gnu.org/prep/standards.html

[204] Stallman, R. "Why free software is better than open source".

http://www.fsf.org/philosophy/free-software-for-freedom.html

[205] Stallman, R. (1998). "Copyleft: pragmatic idealism".

http://www.gnu.org/philosophy/pragmatic.html

[206] Stallman, R. (1998). "Why free software is better than open source".

http://www.gnu.org/philosophy/free-software-for-freedom.html

[207] Stallman, R. (1998). "Why software should not have owners".

http://www.gnu.org/philosophy/why-free.html

[208] Stallman, R. "The GNU Operating System and the Free Software Movement". In:

DiBona et al. [108].

http://www.fsf.org/gnu/thegnuproject.html
[209] Stallman, R. (1999, June). "On free hardware". Linux Today.
http://features.linuxtoday.com/news_story.php3?ltsn=1999-06-22-005-05-NW-LF
[210] Stallman, R. (2001). "The free universal encyclopedia and learning resource".
http://www.gnu.org/encyclopedia/free-encyclopedia.html
[211] Stallman, R. (2002). Free software, free society. Selected essays of Richard M. Stallman. Joshua Gay.
[212] Stallman, R. (2003). "Some confusing or loaded words and phrases that are worth avoiding".
http://www.gnu.org/philosophy/words-to-avoid.html
[213] Stoltz, M. (1999). "The case for government promotion of open source software".
http://www.netaction.org/opensrc/oss-report.html
[214] Tanenbaum, A.; Torvalds, L. (1999). "The Tanenbaum-Torvalds debate".
http://www.oreilly.com/catalog/opensources/book/appa.html
[215] The Open Source Initiative. "The open source definition".
http://www.opensource.org/docs/definition_plain.html
[216] Tiemann, M. "Future of Cygnus Solutions. An entrepreneur's account". In: DiBona et al. [108].
http://www.oreilly.com/catalog/opensources/book/tiemans.html
[217] Torvalds, L; Diamond; D. (2001). Just for fun: the story of an accidental revolutionary. Texere.
[218] Linus Torvalds, Hamano, J. C.; Ericsson, A. "Git manual page".
http://www.kernel.org/pub/software/scm/git/docs/
[219] Tuomi, I. (2002). "Evolution of the Linux credits file: methodological challenges and reference data for open source research".
http://www.jrc.es/~tuomiil/articles/EvolutionOfTheLinuxCreditsFile.pdf
[220] Several authors. "Open letter to WIPO".
http://www.cptech.org/ip/wipo/kamil-idris-7july2003.pdf

[221] Vigo i Sallent, P.; Benach i Pascual, E.; Huguet i Biosca; J. (2002, May). Proposició de llei de programari lliure en el marc de l'Administració pública de Catalunya.
http://www.parlament-cat.es/pdf/06b296.pdf
http://www.hispalinux.es/
modules.php?
op=modload&name=Sections&file=index&req=viewarticle&artid=49
[222] Villanueva Núñez, E. (2001, December). Free software project bill, number 1609.
http://www.gnu.org.pe/proley1.html
[223] Villanueva Núñez, E.; Rodrich Ackerman, J. (2002, April). Bill on the use of free software by the Public Administration, number 2485.
http://www.gnu.org.pe/proley4.html
[224] W3C (2000). Extensible markup language (xml) 1.0 (2nd edition).
[225] Walsh, N.; Muellner, L.; Stayton, B. (2002). DocBook: the definitive guide. O'Reilly.
http://docbook.org/tdg/en/html/docbook.html
[226] Welke, L; Johnson, L. (1998). How the ICP Directory began.
http://www.softwarehistory.org/history/Welke1.html
[227] Wheeler, D. A. (2000, July). "Estimating Linux's size".
http://www.dwheeler.com/sloc
[228] Wheeler, D. A. (2001, June). "More than a gigabuck: estimating GNU/Linux's".
http://www.dwheeler.com/sloc
[229] Wiesstein, E. "Concise encyclopedia of mathematics".
http://mathworld.wolfram.com/
[230] Wikipedia. "Gini coefficient".
http://www.wikipedia.org/wiki/Gini_coefficient
[231] Wikipedia. "Lorenz curve".
http://www.wikipedia.org/wiki/Lorenz_curve
[232] Wikipedia. "Pareto".
http://www.wikipedia.org/wiki/Pareto
[233] Wikipedia. "TeX".

http://www.wikipedia.org/wiki/TeX

[234] Wilson, B. "Netscape Navigator".

http://www.blooberry.com/indexdot/history/netscape.htm

[235] Computer World (2000). "Salary survey 2000".

http://www.computerworld.com/cwi/careers/surveysandreports

[236] Young, R. (1999). "Giving it away. how Red Hat software stumbled across a new economic model and helped improve an industry".

http://www.oreilly.com/catalog/opensources/book/young.html

[237] Zawinsky, J. W. (1999). "Resignation and postmortem".

http://www.jwz.org/gruntle/nomo.html

GIỚI THIỆU PHẦN MỀM TỰ DO
INTRODUCTION TO FREE SOFTWARE

CHỊU TRÁCH NHIỆM XUẤT BẢN:
NGUYỄN THỊ THU HÀ

BIÊN TẬP: PHẠM VĂN GIÁP

SỬA BẢN IN: TẠP CHÍ TIN HỌC VÀ ĐỜI SỐNG

TRÌNH BÀY: TẠP CHÍ TIN HỌC VÀ ĐỜI SỐNG

NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG

Trụ sở chính: 75 ngõ 5 Hoàng Tích Trí, Hà Nội
ĐT: 04-35772143, 35772136 Fax: 04-35772037
E-mail: nxb.tttt@mic.gov.vn Website: www.nxbthongtintruyenthong.vn

Chi nhánh TP Hồ Chí Minh: 8A đường D2, phường 25, Q.Bình Thạnh, TP Hồ Chí Minh
ĐT: 08-35127750 Fax: 08-35127751
E-mail: cnsg.nxbtttt@mic.gov.vn

Chi nhánh TP Đà Nẵng: 42 Trần Quốc Toản, Q. Hải Châu, thành phố Đà Nẵng
ĐT: 0511-3897467 Fax: 0511-3843359
Email: cndn.nxbtttt@mic.gov.vn

In 2000 cuốn khổ 21 x 29 cm tại Xưởng in Tạp chí Tin học và Đời sống.

Số đăng ký kế hoạch xuất bản: 327 – 2010/CXB/9-189/TTTT.

Quyết định xuất bản số: 79/QĐ-NXBTTTT ngày 13/04/2010 của Giám đốc Nhà xuất bản Thông tin và Truyền thông.

In xong và nộp lưu chiểu Quý II-2010.

Phần mềm tự do là gì? Nó là gì và những ảnh hưởng của một giấy phép của chương trình tự do là gì? Các dự án phần mềm tự do được cấp vốn như thế nào và các mô hình kinh doanh có liên quan tới chúng mà chúng ta đang trải nghiệm là gì? Cái gì là động lực cho các lập trình viên, đặc biệt là những tình nguyện viên, tham gia vào các dự án phần mềm tự do? Những lập trình viên này trông ra sao? Các dự án của họ được điều phối thế nào, và phần mềm mà họ làm ra trông ra sao? Ngắn gọn, toàn cảnh của phần mềm tự do là gì?

Đây là những dạng câu hỏi mà chúng ta sẽ cố gắng trả lời trong tài liệu này, vì mặc dù phần mềm tự do ngày một gia tăng sự hiện diện của nó trong các phương tiện truyền thông và trong các tranh luận giữa dân chuyên ngành công nghệ thông tin, và mặc dù ngay cả một vài người dân cũng bắt đầu nói về nó, thì nó vẫn còn chưa rõ với nhiều người, và ngay cả những người mà quen với nó cũng thường nhận thức được chỉ một số tính năng của nó, và hầu hết bỏ qua những tính năng khác.

What is free software? What is it and what are the implications of a free program licence? How is free software projects financed and what are the business models associated to them that we are experiencing? What motivates developers, especially volunteers, to become involved in free software projects? What are these developers like? How are their projects coordinated, and what is the software that they produce like? In short, what is the overall panorama of free software?

These are the sort of questions that we will try to answer in this document, because although free software is increasing its presence in the media and in debates among IT professionals, and although even citizens in several are starting to talk about it, it is still unknown for many people, and even those who are familiar with it are often aware of just some of its features, and mostly ignorant about others.

freeknowledgeinstitute
UOC Universitat Oberta de Catalunya www.uoc.edu
www.ou.nl
UNIVERSITETET I AGDER

Giá: 80.000đ