Chapter 4 – Community driven development

Phần mềm mã nguồn mở thường được phát triển bởi một nhóm người thông thường không có quyền lực quản lý hoặc hỗ trợ tổ chức. Họ sử dụng những chiến lược và phương pháp gì? Họ sử dụng các công cụ nào ở các giai đoạn khác nhau của vòng đời phát triển phần mềm? Những công cụ này có sẵn miễn phí không? Chương này cung cấp câu trả lời cho tất cả những câu hỏi trên.

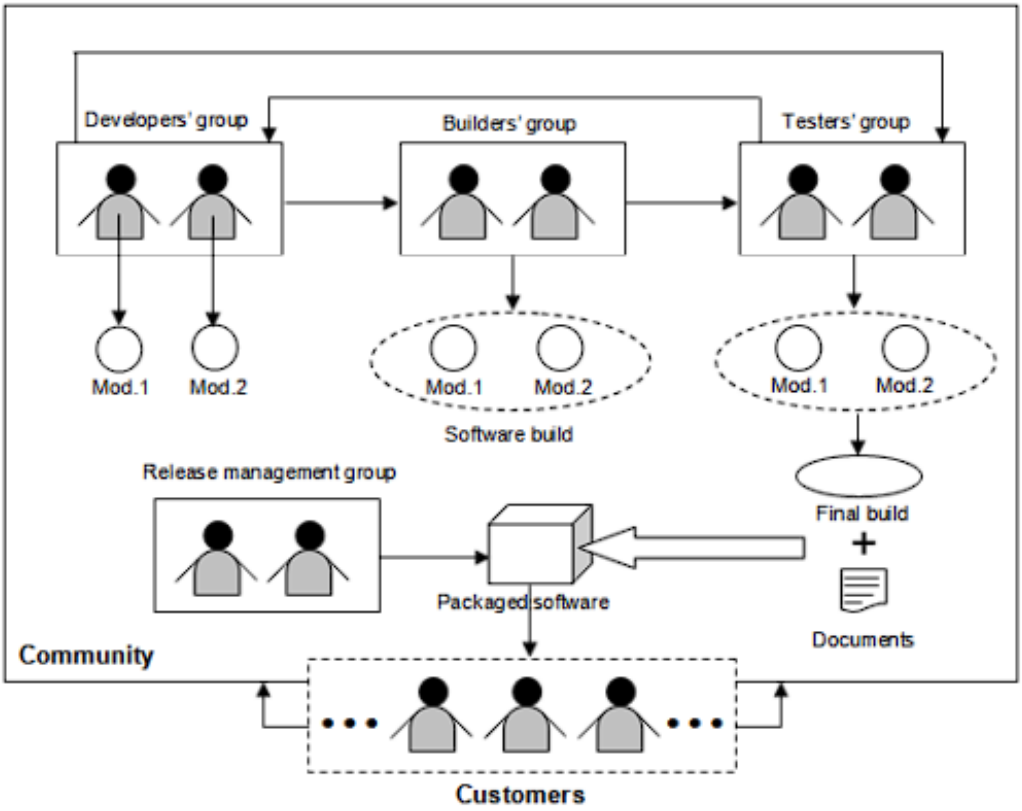
Trong chương này, bạn sẽ tìm hiểu về:

- Các công cụ khác nhau được sử dụng trong dự án phát triển phần mềm mã nguồn mở.
- Quy trình thiết kế và viết chương trình.
- Quản lý các phiên bản khác nhau của mã nguồn và tạo các phiên bản mới.
- Phương pháp kiểm thử và theo dõi vấn đề.
- Cách phần mềm mã nguồn mở được phát hành.

4.1 Phát triển dựa trên cộng đồng: Tổng quan chung

Một cộng đồng là một nhóm đa dạng người chia sẻ suy nghĩ, ý tưởng, công việc và kinh nghiệm tại một địa điểm chung. Thuật ngữ "cộng đồng" khi được sử dụng trong chương này không nằm ngoài định nghĩa này. Trong ngành công nghiệp phần mềm, phát triển dựa trên cộng đồng rộng rãi có nghĩa là một sáng kiến trong đó một nhóm kỹ sư công nghệ hợp tác với mục tiêu chung là tạo ra phần mềm mã nguồn mở.

Hầu hết các cộng đồng phát triển phần mềm mã nguồn mở hiện nay, tổ chức chúng theo cách tương tự như một tổ chức chuyên nghiệp hoặc một công ty phần mềm độc quyền. Điều này làm cho trong một cộng đồng OSS có cấu trúc tốt, bạn có thể thấy mỗi thành viên đều có một vai trò công việc trong nhóm. Các nhóm người thực hiện công việc tương tự hình thành các đội con bên trong cộng đồng. Hình 4.1 mang đến cái nhìn tổng quan về một cộng đồng như vậy, nơi nhiều đội con hợp tác để phát triển phần mềm mã nguồn mở.



Hình 4.1 - Phát triển phần mềm dựa trên cộng đồng

Như thể hiện trong hình, nhóm nhà phát triển chịu trách nhiệm viết mã cho các module độc lập khác nhau của phần mềm. Nhóm xây dựng lấy những module này từ nhà phát triển và kết hợp chúng để xây dựng một phiên bản mới của phần mềm. Các phiên bản phần mềm này được kiểm thử nội bộ bởi nhóm kiểm thử. Trong trường hợp phát hiện lỗi hoặc sự cố, họ báo cáo lại cho nhà phát triển thích hợp, người đó sẽ gỡ lỗi mã và thêm một sửa lỗi phù hợp. Đây là một quá trình lặp lại lặp lại cho đến khi sản phẩm đáp ứng tất cả yêu cầu của nó và trở thành tối đa có thể không có lỗi. Khi đạt được mục tiêu của dự án, nhóm quản lý phiên bản đóng gói phiên bản cuối cùng (bản xây dựng cuối cùng) của phần mềm với tất cả các tài liệu cần thiết, sau đó chuyển giao cho khách hàng.

Cộng đồng phát triển OSS, đặc biệt là những cộng đồng tự lái, cung cấp cho các thành viên trong nhóm sự linh hoạt trong việc thay đổi vai trò công việc của họ. Điều này có nghĩa là một nhà phát triển có thể thực hiện vai trò của một kiểm thử viên, một kiểm thử viên có thể làm việc như một thành viên nhóm xây dựng, và cứ như vậy. Quan trọng nhất, những cộng đồng này luôn mở cửa cho người dùng (khách hàng), để họ cũng có thể đóng góp một cách hiệu quả vào dự án và trở thành một phần của cộng đồng.

Công cụ phát triển phần mềm là một bộ sưu tập các chương trình hỗ trợ các nhiệm vụ cụ thể trong vòng đời phát triển phần mềm. Thường được sử dụng cho quản lý dự án và thực hiện các nhiệm vụ lặp đi lặp lại và tốn thời gian. Có cả công cụ phát triển có giấy phép và mã nguồn mở. Đối với dự án mã nguồn mở, rõ ràng là lựa chọn sau cùng là sự ưa thích. Cũng có một số trường hợp, nơi cộng đồng sử dụng các công cụ tự phát triển của họ. Các phần tiếp theo mô tả chi tiết về các quy trình và công cụ được sử dụng bởi các cộng đồng phát triển OSS.

4.1.1 Nhóm nhà phát triển: Thiết kế và phát triển phần mềm

Bạn có thể tưởng tượng thiết kế phần mềm như bản thiết kế xác định kiến trúc và hành vi của một sản phẩm, và lập trình như quá trình triển khai thiết kế.

Một dự án phát triển phần mềm dựa trên cộng đồng thường bắt đầu như một dự án để giải quyết các vấn đề phổ quát trong xã hội. Những vấn đề này kỹ thuật được gọi là điểm đau và bao gồm một danh sách yêu cầu mà phần mềm được kỳ vọng phải đáp ứng. Nhóm thiết kế, trong nhiều trường hợp cũng là một phần của nhóm nhà phát triển, lấy những yêu cầu này làm đầu vào và viết các thuật toán mà có chi phí hiệu quả và tối ưu về hiệu suất.

Nhà thiết kế phần mềm nên là những thành viên có kinh nghiệm và kiến thức sâu rộng nhất trong cộng đồng vì công việc của họ đòi hỏi sự hiểu biết kỹ lưỡng về toàn bộ hệ thống. Họ cũng phải tìm hiểu mọi lựa chọn để vượt qua tất cả các rào cản có thể xảy ra khi triển khai thiết kế. Dưới đây là hai điểm quan trọng nhất mà một nhà thiết kế phải phân tích cẩn thận:

- Nền tảng phần cứng: Nhà thiết kế phải biết về khả năng của phần cứng mà phần mềm sẽ tương tác. Tài nguyên như số lượng và tốc độ của các bộ xử lý, thời gian phản hồi của thiết bị I/O, bộ nhớ cache hệ thống có sẵn, bộ nhớ chính và không gian lưu trữ phụ, v.v., là những quan tâm quan trọng nhất. Hệ điều
- hành: Hệ điều hành (OS) cung cấp môi trường để chạy ứng dụng. Cơ chế xử lý tiến trình và luồng có sự biến động từ một hệ điều hành sang hệ điều hành khác. Hơn nữa, các thuật toán lập lịch công việc, phân bổ thời gian CPU, tránh tình trạng khóa hóa cũng khác nhau. Nhà thiết kế phải xem xét những yếu tố này.

Một thiết kế hoàn hảo nên là tự giải thích và dễ hiểu. Tất cả các thuật toán và biểu đồ luồng nên bao gồm các giải thích chi tiết và tài liệu cần thiết khác bao gồm độ phức tạp thời gian-không gian và hành vi dự kiến của hệ thống dưới các kịch bản tốt nhất và xấu nhất.

Ngôn ngữ Mô hình Hóa Thống nhất (UML) là một tiêu chuẩn được chấp nhận rộng rãi được sử dụng cho thiết kế phần mềm. Nó cho phép nhà thiết kế mô tả cấu trúc, hành vi và tương tác giữa các đơn vị con khác nhau của hệ thống thông qua các biểu đồ đơn giản khác nhau.

Sau giai đoạn thiết kế, vòng đời phát triển phần mềm chuyển sang giai đoạn lập trình. Tại giai đoạn này, thiết kế được chia thành nhiều đơn vị nhỏ hơn. Các nhóm phát triển khác nhau mã hóa những đơn vị nhỏ này một cách độc lập như là các module riêng lẻ.

Lập trình là quá trình chuyển các bước của một thuật toán thành nhiều hướng dẫn máy tính bằng cách sử dụng một ngôn ngữ lập trình. Việc chọn ngôn ngữ phù hợp để viết mã nguồn là quan trọng. Một cộng đồng nên luôn khuyến khích sử dụng các ngôn ngữ đã được ngành công nghiệp tin cậy và mà đội phát triển có kinh nghiệm. Một số cộng đồng thích sử dụng các ngôn ngữ thông dịch, vì chúng tiêu tốn ít thời gian để tạo ra các module thực thi so với các ngôn ngữ biên dịch.

Các công cụ tự động tạo mã thường hữu ích trong việc phát triển đoạn mã lớn và phức tạp. iCodeGenerator là một công cụ tạo mã nguồn mã nguồn mở được cung cấp bởi SourceForge (http://sourceforge.net/projects/codegenerator/).

Chuyển đổi giữa các giai đoạn thiết kế và tạo mã thường có xu hướng chồng lấn lẫn nhau trong một dự án phát triển dựa trên cộng đồng. Các nhà phát triển đôi khi bắt đầu mã hóa một số phần của hệ thống,

trong khi các phần khác vẫn chưa được thiết kế. Mặc dù cách tiếp cận này chạy hai giai đoạn song song giúp tiết kiệm thời gian, nhưng nó có thể dẫn đến sự trùng lặp công việc nếu có sự thay đổi trong thiết kế.

4.1.1.1 Kiểm soát phiên bản

Phát triển phần mềm là một quá trình tăng cường mà chức năng được thêm vào hoặc loại bỏ một cách từng bước một. Tại bất kỳ thời điểm nào, mã nguồn của các module khác nhau có thể trải qua các sửa đổi thường xuyên bởi các cá nhân hoặc nhóm khác nhau, tạo ra các phiên bản khác nhau của cùng một mã nguồn. Kiểm soát phiên bản (hoặc kiểm soát sửa đổi) là cơ chế để quản lý tất cả các phiên bản này một cách hiệu quả.

Có hai loại hệ thống kiểm soát phiên bản:

- Hệ thống Kiểm soát Phiên bản Tập trung (CVCS): Đây có lẽ là hệ thống kiểm soát phiên bản được sử dụng rộng rãi nhất cho đến nay. Nó hoạt động theo cách tập trung, có nghĩa là các nhà phát triển và người kiểm thử trên khắp thế giới phải kết nối với một máy chủ, nơi mà kho chứa tập trung cho toàn bộ dự án được lưu trữ.

  Concurrent Versions System (CVS) là một hệ thống kiểm soát phiên bản tập trung mã nguồn mở rất nổi tiếng (http://ftp.gnu.org/non-gnu/cvs/). Để có một bản sao riêng tư của một tệp cụ thể, bạn cần kiểm tra nó ra khỏi kho chứa. Khi hoàn tất với những thay đổi của bạn, bạn có thể commit hoặc lưu công việc của mình vào kho chứa bằng cách kiểm tra tệp. CVS sẽ tự động tăng số phiên bản (hoặc revision) lên 1 và ghi lại tất cả thông tin cần thiết về sự thay đổi trong log của nó. Nếu ai đó đã commit những thay đổi khác vào cùng một tệp, CVS sẽ tự động hợp nhất thay đổi của bạn với thay đổi của người khác. Hệ thống này cũng cho phép người dùng theo dõi các phiên bản cũ của một tệp. Bằng cách so sánh hai phiên bản liên tiếp của một tệp, bạn có thể dễ dàng xác định sự thay đổi đã được thực hiện

  Một trong những hạn chế chính của CVCS là các nhà phát triển có thể làm phiền môi trường làm việc của nhau. Ví dụ, một lỗi nhỏ được nhập vào thông qua một sự thay đổi mã nguồn, trở thành một vấn đề lớn khi nó làm cho nhiều module khác nhau của toàn bộ mã nguồn gặp sự cố.

- Hệ thống Kiểm soát Phiên bản Phi tập trung / Phân tán (DVCS): Loại kiểm soát phiên bản này cho phép một nhà phát triển hoặc người kiểm thử tạo ra một nhánh mã nguồn của riêng mình, nghĩa là họ có thể duy trì các phiên bản của mã nguồn theo cách phi tập trung. Một hệ thống kiểm soát phiên bản phân tán cung cấp tất cả các chức năng của CVCS, cộng với tính linh hoạt của việc tạo ra một kho chứa địa phương.
  Git, Bazaar, Monotone, Mercurial và Darcs là một số ví dụ về phần mềm kiểm soát phiên bản phân tán mã nguồn mở.

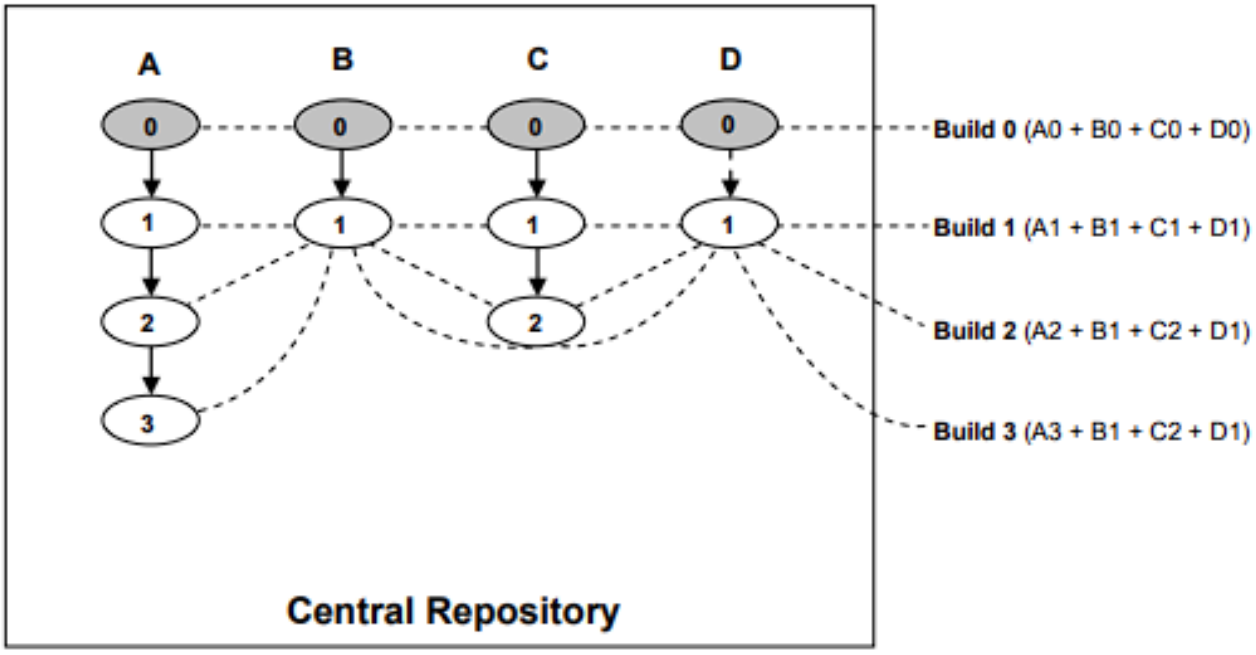
Bạn có biết không?

Trong giai đoạn đầu của việc phát triển hạt nhân Linux®, Linus Torvalds đã sử dụng CVS cho quản lý phiên bản. Đến năm 2002, Linus chuyển sang sử dụng BitKeeper; tuy nhiên, do BitKeeper trở thành phần mềm độc quyền vào năm 2005, Linus bắt đầu viết một hệ thống kiểm soát phiên bản phân tán hoàn toàn mới, được gọi là Git. Trang web chính thức của Git (http://git-scm.com/) mô tả nó như sau: "... một hệ thống kiểm soát phiên bản phân tán, miễn phí và mã nguồn mở, được thiết kế để xử lý mọi thứ từ dự án nhỏ đến rất lớn với tốc độ và hiệu suất. Mỗi bản sao Git là một kho chứa đầy đủ với lịch sử và khả năng theo dõi sửa đổi đầy đủ, không phụ thuộc vào quyền truy cập mạng hay một máy chủ trung tâm. Việc nhánh và hợp nhất diễn ra nhanh chóng và dễ dàng."

4.1.2 Nhóm xây dựng: Xây dựng phần mềm

Chương trình thường được viết dưới dạng các đơn vị nhỏ trước khi có thể được thực thi. Các đơn vị kết quả sau đó được liên kết trong quá trình được biết đến là liên kết để tạo thành một hệ thống phần mềm hoàn chỉnh. Việc xây dựng phần mềm đề cập đến tổng hợp của việc biên dịch và liên kết tất cả mã nguồn của một dự án.

Xây dựng phần mềm là một quá trình động: Mọi người liên tục thêm, xóa và sửa đổi các tệp mã nguồn dưới kho chứa trung tâm của dự án. Để làm cho những thay đổi này có hiệu quả, mã nguồn đã được sửa đổi cần phải được biên dịch lại và liên kết thường xuyên. Các cộng đồng phát triển thường có một đội xây dựng riêng để thực hiện công việc này. Quan trọng để đề cập rằng quá trình xây dựng chỉ biên dịch lại những tệp mã nguồn đã được sửa đổi kể từ khi phiên bản trước đó được phát hành. Đối với các phần còn lại không thay đổi, các mô-đun đã biên dịch từ trước được sử dụng lại. Hình 4.3 minh họa điều này.



Hình 4.3 – Quá trình xây dựng phần mềm

Trong hình vẽ, kho chứa trung tâm chứa bốn tệp mã nguồn A, B, C và D. Phiên bản cơ bản của những tệp này được minh họa với số không, do đó A0 đại diện cho phiên bản cơ bản của tệp mã nguồn A chẳng hạn. Khi biên dịch và liên kết, chúng tạo ra 'Build 0'. Trong vòng lặp tiếp theo, tất cả các tệp được sửa đổi và phiên bản của chúng được nâng cấp lên 1. Vì tất cả các tệp đã được sửa đổi, chúng được biên dịch và liên kết để tạo ra 'Build 1'. Trong vòng lặp tiếp theo, chỉ có tệp mã nguồn A và C được sửa đổi, tạo ra phiên bản thứ hai của chúng (A2 và C2). Do đó, khi tạo 'Build 2', chỉ A2 và C2 được biên dịch, sau đó liên kết với B1 và D1. Tương tự, 'Build 3' được tạo dựa trên A3 mới biên dịch và B1 đã được biên dịch trước, C2 và D1.

Berczuk và Appleton đã chia quá trình xây dựng phần mềm thành ba danh mục riêng biệt trong cuốn sách "Software Configuration Management Patterns" [2], và một bài báo bổ sung với Konieczka, "Build Management for an Agile Team" [3]:

- Xây dựng riêng tư (Private build): Là một phiên bản mới của nhánh cá nhân của một nhà phát triển trong hệ thống kiểm soát phiên bản phân tán (DVCS). Tự nhiên, loại xây dựng này chỉ có thể được nhìn thấy bởi chủ sở hữu của nó và anh ta có thể truy cập nó cục bộ ngay cả khi làm việc ngoại tuyến.
- Xây dựng tích hợp (Integration build): Là một phiên bản được tạo ra từ việc biên dịch và liên kết của nhánh chính của dự án và có sẵn cho tất cả trong cộng đồng để sử dụng.
- Xây dựng phát hành (Release build): Là phiên bản cuối cùng của phần mềm để gửi đi. Đây về cơ bản là một bản chụp của một xây dựng tích hợp ổn định được người dùng sử dụng như một sản phẩm.

Việc xây dựng thường xuyên giúp mã nguồn trải qua ít sự thay đổi hơn. Điều này giảm bớt công sức cần thiết cho việc gỡ lỗi và sửa lỗi mới. Ngược lại, quá trình xây dựng liên quan đến nhiều công việc lặp lại có thể được tự động hóa thông qua các công cụ mã nguồn mở như make, ant, maven và scons.

Cho đến nay, chúng ta đã thảo luận về quá trình xây dựng trong môi trường dự án. Tuy nhiên, do mã nguồn mở, bất cứ lúc nào người dùng có thể tùy chỉnh nó tại địa điểm của họ và có được các phiên bản tùy chỉnh của sản phẩm. Một phần sau sẽ thảo luận về cách bạn có thể xây dựng và cài đặt phần mềm mã nguồn mở sau khi thay đổi mã nguồn của nó.

4.1.3 Nhóm kiểm thử: Kiểm thử phần mềm

Kiểm thử phần mềm, như được định nghĩa bởi Glenford J. Myers trong cuốn sách "The Art of Software Testing" [4], là "quá trình thực thi một chương trình với ý định tìm ra lỗi".

Theo truyền thống, có hai loại kiểm thử phần mềm - hộp đen (black box) và hộp trắng (white box).

- Kiểm thử hộp đen (Black box testing): Trong loại kiểm thử này, kết quả thực tế của mã nguồn được so sánh với kết quả mong đợi. Người kiểm thử cần cung cấp hệ thống với một tập hợp đầu vào và ghi lại các kết quả tương ứng. Trong trường hợp không khớp giữa kết quả thu được và kết quả mong đợi, họ báo cáo cho nhà phát triển mà không phân tích mã nguồn.
- Kiểm thử hộp trắng (White box testing): Ngược lại với kiểm thử hộp đen, trong loại kiểm thử này, người kiểm thử xem xét mã nguồn của sản phẩm và tiến hành một số điều tra khi xảy ra lỗi.

Điều này yêu cầu nhà phát triển đầu tư ít công sức và thời gian hơn cho việc phân tích lỗi trước khi sửa chúng.

Với phần mềm mã nguồn mở, mọi người, dù có là thành viên của cộng đồng hay không, đều có cơ hội bằng nhau để thực hiện kiểm thử hộp trắng vì mã nguồn mở có sẵn cho tất cả.

Kiểm thử cũng đòi hỏi kế hoạch và cơ sở hạ tầng phù hợp. Tại các giai đoạn khác nhau của quá trình phát triển, mã nguồn trải qua các cấp độ kiểm thử khác nhau:

- Kiểm thử đơn vị (Unit testing): Cấp độ này được sử dụng để kiểm thử từng đơn vị cá nhân tạo nên phần mềm.
- Kiểm thử tích hợp (Integration testing): Tại cấp độ này, tất cả các mô-đun ổn định được tích hợp vào nhau để tạo thành toàn bộ hệ thống và được kiểm thử như một thể thống nhất.
- Kiểm thử hệ thống (System testing): Kiểm thử này đảm bảo rằng phần mềm hoạt động và đáp ứng tất cả yêu cầu của nó.
- Kiểm thử Alpha (Alpha testing): Tại giai đoạn này, phần mềm được giao cho người dùng nội bộ trong cộng đồng để đánh giá hiệu suất của nó khi triển khai thực tế.
- Kiểm thử Beta (Beta testing): Sau khi sửa các lỗi phát hiện trong kiểm thử alpha, cộng đồng phát hành phần mềm cho người dùng ngoại vi như một phiên bản beta, với một thông báo từ chối trách nhiệm rằng nó có thể gặp sự cố trong các trường hợp sử dụng không dự kiến. Cộng đồng mong đợi người dùng ngoại vi sẽ cung cấp phản hồi về những tình huống không dự kiến này.

Phần mềm mã nguồn mở quy mô nhỏ mà không có bất kỳ quản lý tổ chức nào có thể không trải qua kiểm thử hình thức. Tuy nhiên, phần mềm này thường được chấp nhận nhanh chóng vì nó cung cấp một giải pháp cho một vấn đề phổ biến. Khi gặp lỗi, người dùng thường sẽ kiểm tra mã nguồn để làm cho mọi thứ hoạt động trở lại, hoặc họ có thể tùy chỉnh nó theo nhu cầu của mình. Đây là cách nhiều người trở thành thành viên của cộng đồng phần mềm mã nguồn mở. Khi phần mềm trở nên hữu ích và có đủ người theo dõi, một số người dùng sẽ đưa phần mềm lên một tầm cao mới bằng cách cải thiện nó và phát hành nó một cách chính thức cho cộng đồng.

Phát triển phần mềm mã nguồn mở thường tiêu tốn ít thời gian và ngân sách cho việc kiểm thử sản phẩm của họ. Thay vào đó, họ tận dụng cộng đồng để tự kiểm thử và sửa lỗi mã nguồn; và để đưa ra phản hồi cho các tính năng mới. Hơn nữa, phương pháp độc đáo này làm cho phần mềm mã nguồn mở rất mạnh mẽ.

4.1.4 Nhóm quản lý phát hành: Đóng gói

Người dùng ưa thích một sản phẩm được đóng gói đẹp. Trong phần mềm mã nguồn mở, mục cốt lõi của một gói là mã nguồn. Một gói tiêu chuẩn bao gồm các phụ kiện như hướng dẫn cài đặt, hướng dẫn sử dụng và thông tin khác. Gói phần mềm được viết bằng các ngôn ngữ lập trình cao như C hoặc C++ và yêu cầu biên dịch mã nguồn, tùy chọn có thể chứa một phiên bản thực thi đã được biên dịch trước của sản

phẩm. Tất cả các yếu tố này được đặt cùng nhau trong một thư mục duy nhất và được nén vào một tệp duy nhất.

Định dạng của tệp nén phụ thuộc vào hệ điều hành mà phần mềm được thiết kế để chạy. Trên Linux hoặc UNIX®, các gói được hiển thị trong các định dạng như .tar, .gz, v.v. được tạo bằng các tiện ích nén như tar, gzip, bgzip, bzip2. Trên Windows®, gói phân phối là zip. Nếu phần mềm có các phiên bản dành riêng cho hệ điều hành cụ thể, mỗi phiên bản sẽ được phát hành trong các gói riêng biệt.

Khi gói đã được chuẩn bị, nó trải qua một bài kiểm tra nhỏ bởi một số nhà phát triển để đảm bảo rằng phiên bản cuối cùng của sản phẩm có thể được giải nén, cài đặt và hoạt động đúng cách. Sau đó, nó được xuất bản trên một hoặc nhiều trang web internet để tải xuống. Trong một số trường hợp, nếu có ngân sách, CD hoặc DVD có thể được đặt hàng trực tuyến mà không mất phí.

4.1.5 Nhóm quản lý phát hành: Phát hành

Phát hành phần mềm có nghĩa là đưa một phiên bản hoàn toàn mới hoặc nâng cấp của một phần mềm có sẵn cho người dùng. Một phiên bản mới giới thiệu một sản phẩm mới vào thị trường và một phiên bản nâng cấp cung cấp các sửa lỗi cho các lỗi được tìm thấy trong phiên bản đã phát hành trước đó và tùy chọn thêm các tính năng mới.

Để phân biệt các phiên bản với nhau, mỗi cộng đồng mã nguồn mở tuân theo quy ước của mình bằng cách sử dụng các số duy nhất cho mỗi phiên bản. Dưới đây là một mẫu của kiểu đánh số như vậy cho tên tệp của gói nén. Phần mở rộng tệp không được hiển thị, nhưng nó có thể là .tar, .gz, .zip và vân vân như đã thảo luận trong phần trước:

Abcdefg x.y.z < Alpha | Beta >

Trong ví dụ trên, Abcdefg là tên của phần mềm và x, y và z đại diện cho các thành phần chính, phụ và micro của số phiên bản phát hành duy nhất tương ứng. Một phiên bản chỉ chứa các sửa lỗi nhỏ cho phiên bản trước đó được coi là một bản phát hành nhỏ hoặc phụ và do đó liên kết với cùng một số phiên bản như của phiên bản trước đó, tăng giá trị của số cuối cùng (thành phần micro) lên 1. Ví dụ, nếu một sản phẩm được phát hành với tên là Abcdefg 2.0.5, một phiên bản nhỏ sẽ được biểu thị là Abcdefg 2.0.6. Một sửa lỗi quan trọng được áp dụng bằng cách tăng giá trị của thành phần phụ và thiết lập lại thành phần micro, khiến cho phiên bản mới được xác định là Abcdefg 2.1.0. Nếu các tính năng mới được thêm vào, số phiên bản sẽ được nâng cấp thành Abcdefg 3.0.0. Khi phần mềm với chức năng giới hạn được cung cấp cho người dùng nội bộ để kiểm thử, phiên bản trung gian này được gọi là Alpha như trong Abcdefg 3.0.0 Alpha. Một phiên bản alpha giúp cộng đồng phát triển thu thập báo cáo lỗi từ người dùng. Tùy thuộc vào số lượng và tính nhạy cảm của các lỗi phát hiện được, cộng đồng cũng có thể quyết định phát hành nhiều phiên bản alpha như Abcdefg 3.0.0 Alpha 1, Abcdefg 3.0.0 Alpha 2, và cứ tiếp tục như vậy. Khi chu kỳ alpha kết thúc, sản phẩm hoàn chỉnh về tính năng được cung cấp cho bên thứ ba ngoài cộng đồng. Điều này được gọi là một phiên bản Beta. Theo ví dụ, phiên bản sẽ được gọi là Abcdefg 3.0.0 Beta. Với đủ kiểm thử và gỡ lỗi qua các chu kỳ alpha và beta, phần mềm cuối cùng trở nên sẵn sàng để được chính thức phát hành. Ở giai đoạn quyết định này, cộng đồng thông báo "Tính sẵn có tổng quát (GA)" của sản phẩm.

Sau một bản phát hành thành công, công việc của nhóm phát triển được chia thành hai phần: Phần đầu tiên là duy trì & hỗ trợ phiên bản đã được phát hành, và phần thứ hai là cải thiện thêm sản phẩm với các tính năng mới để bao gồm trong phiên bản tiếp theo. Tại điểm này, ngành điều khiển phiên bản trung tâm của hệ thống quản lý phiên bản được chia thành hai. Nhánh nơi công việc duy trì sẽ diễn ra được gọi là nhánh phát hành. Các bản snapshot ổn định của nhánh này được phát hành như các phiên bản nhỏ của sản phẩm trong cùng một dòng chính (ví dụ, Abcdefg 3.1.0, Abcdefg 3.2.0, và cứ tiếp tục như vậy). Sửa lỗi cho các lỗi có khả năng gây ra thất bại cao mà không thể đợi đến phiên bản nhỏ tiếp theo được phân phối cho khách hàng qua một nhánh con của dòng phát hành hiện tại như Abcdefg 3.2.1, Abcdefg 3.2.2, vv.

Nhánh nơi nhà phát triển làm việc trên phiên bản mới (Abcdefg 4.0.0 trong ví dụ này) được gọi là nhánh phát triển chính. Nhánh này tiến triển song song với nhánh phát hành.

Một cộng đồng nên tiếp tục cung cấp các sửa lỗi cho các phiên bản trước cho đến khi đa số người dùng hiện tại bắt đầu sử dụng phiên bản mới nhất của phần mềm.

4.2 Cài đặt và theo dõi vấn đề

Sau khi một phần mềm mã nguồn mở (OSS) được phát hành và có sẵn cho người dùng, người dùng phải cài đặt sản phẩm. Khác với phần mềm độc quyền, OSS có thể yêu cầu người dùng biên dịch mã nguồn và thực hiện một số cấu hình trước khi cài đặt. Phần này mô tả điều này. Phần này cũng mô tả cách người dùng có thể cung cấp phản hồi cho cộng đồng bằng cách sử dụng theo dõi vấn đề.

4.2.1 Cài đặt

Trước khi người dùng có thể cài đặt phần mềm trên hệ thống của mình, anh ta cần giải nén nó bằng cách giải nén gói. Nhiều người dùng thường cố gắng cài đặt phần mềm mà không đọc hướng dẫn. Đề xuất ít nhất là đọc qua các tệp README hoặc INSTALL. Các bước cài đặt điển hình cho phần mềm mã nguồn mở là:

- Hiểu cấu hình hệ thống
- Biên dịch mã nguồn
- Thực thi tệp cài đặt.

Ví dụ, trên một hệ thống dựa trên Linux hoặc UNIX, chuỗi lệnh dưới đây thực hiện ba bước này:

```
# ./configure

# make

# make install
```

Lệnh đầu tiên được thực thi từ vị trí nơi gói phần mềm được giải nén. Nó gọi một chương trình nhỏ gọi là configure (thông thường được bao gồm trong gói) để phát hiện cấu hình phần cứng và phần mềm của máy và tạo một tệp Makefile chứa bộ lệnh để hoàn thành quá trình cài đặt.

Lệnh thứ hai (make) biên dịch mã nguồn của phần mềm theo hướng dẫn của Makefile.

Lệnh thứ ba (make install) đọc thông số cấu hình của máy tính từ Makefile và cài đặt phần mềm vào đó. Mặc dù hai lệnh đầu có thể được thực thi bởi bất kỳ người dùng thông thường nào, nhưng lệnh cuối cần quyền root.

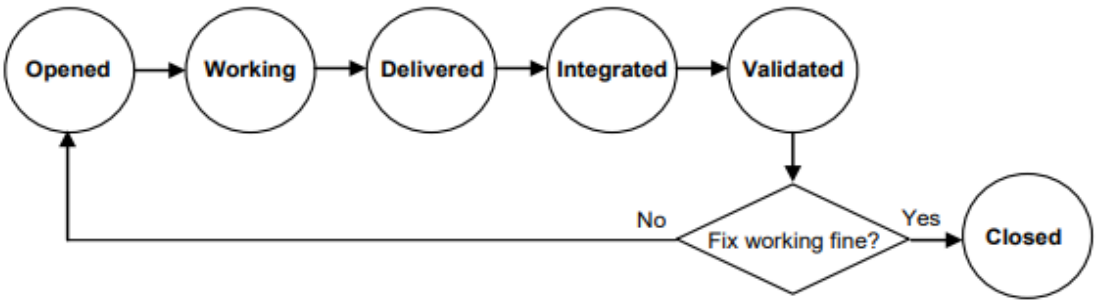
Nếu gói OSS không chứa tập lệnh configure, người dùng sẽ được cung cấp một tệp Makefile mà họ có thể chỉnh sửa với cấu hình máy tính của mình và bắt đầu quá trình thiết lập từ bước thứ hai.

Lưu ý rằng người dùng có thể thay đổi mã nguồn bất kỳ lúc nào và thực hiện lại quá trình này để cài đặt phiên bản được sửa đổi của sản phẩm.

4.2.2 Theo dõi vấn đề

Mặc dù thuật ngữ "vấn đề" có thể được hiểu là một lỗi hoặc vấn đề trong thế giới phát triển phần mềm, thực tế nó còn hơn thế. Một vấn đề có thể liên quan đến một lỗi, nhưng cũng có thể là một yêu cầu thay đổi, nơi thay đổi liên quan đến một phần mã nguồn hiện tại, thiết kế, tính năng hoặc thậm chí tài liệu phần mềm.

Các dự án OSS thường sử dụng các công cụ theo dõi vấn đề miễn phí như Abuky, Bugzilla, Buggit, BugRat, GNATS, JitterBug và Mantis. Một vấn đề có thể được một nhà phát triển, một người kiểm thử hoặc thậm chí là người dùng phát hiện. Khi xác định, nó được báo cáo cho cộng đồng bằng cách đăng nhập một bản ghi trong cơ sở dữ liệu theo dõi vấn đề, được quản lý bởi công cụ, với tất cả các chi tiết cần thiết. Mỗi bản ghi vấn đề được liên kết với một số duy nhất để nhận diện. Hình 4.2 hiển thị các trạng thái khác nhau của một vấn đề trong toàn bộ vòng đời của nó.



Hình 4.2 – Chu kỳ sống của một vấn đề.

Khi một người ("the originator") phát hiện một vấn đề, anh ta đăng nhập một lỗi vào cơ sở dữ liệu theo dõi vấn đề, và trạng thái của vấn đề được đặt **Opened**. Sau đó, một người từ đội phát triển cố gắng mô phỏng lỗi và đảm nhận vấn đề nếu anh ta xác nhận đó thực sự là một lỗi. Tại thời điểm đó, trạng thái

của vấn đề được cập nhật thành **Working**. Khi nhà phát triển (chủ sở hữu) sửa lỗi, anh ta thực hiện các thay đổi cần thiết vào kho chứa trung tâm và cập nhật trạng thái thành **Delivered**. Khi các thay đổi này được bao gồm trong bản xây dựng của nhánh dự án, trạng thái chuyển sang **Intergrated**. Nếu bản xây dựng đạt qua tất cả các kiểm thử cần thiết, nó sẵn có cho toàn bộ cộng đồng và trạng thái vấn đề chuyển sang **Validated**. Người sáng tạo có thể kiểm tra rằng bản xây dựng mới giải quyết vấn đề. Nếu lỗi tái xuất, trạng thái được chuyển trở lại **Opened** và toàn bộ chu kỳ bắt đầu lại; nếu không, vấn đề **Closed**.

Có những biến thể trong vòng đời được minh họa trong Hình 4.2. Thông thường có thể thấy các vấn đề mới **Opened** được **Closed** rất nhanh vì các lý do sau đây:

- **Duplicate:** Nếu ai đó mở một vấn đề cho một lỗi mà đội phát triển đã biết trước, vấn đề đó sẽ được đóng làm trùng lặp.
- **Non-reproducible:** Khi một vấn đề không thể được tái tạo trong một môi trường tương tự, vấn đề đó sẽ được đóng là không tái tạo được.

Các tình huống như vậy thường xuyên xảy ra vì người dùng không theo dõi chặt chẽ dự án mã nguồn mở cũng có thể báo cáo vấn đề. Hơn nữa, có những trường hợp người dùng mở vấn đề mà họ không hiểu rõ về một tính năng. Điều này có thể làm cho cơ sở dữ liệu theo dõi vấn đề của một dự án OSS nổi tiếng phát triển đáng kể với các vấn đề không hợp lệ. Để tránh tình trạng này, trách nhiệm của các thành viên tích cực trong cộng đồng là theo dõi các vấn đề ngay sau khi chúng được mở bằng cách chấp nhận hoặc từ chối chúng với các giải thích đúng đắn. Ngược lại, người dùng nên phân tích một vấn đề trước khi mở nó. Họ ít nhất nên thực hiện một số nghiên cứu cơ bản bằng cách thảo luận vấn đề trong các blog hoặc diễn đàn thảo luận, hoặc tìm kiếm trong cơ sở dữ liệu theo dõi vấn đề. Một lựa chọn khác là gửi email trực tiếp cho đội phát triển và chỉ mở vấn đề khi họ xác nhận đó là một vấn đề mới.

4.3 Bài tập

1. Truy cập hệ thống theo dõi vấn đề của Mozilla tại https://bugzilla.mozilla.org/ và xem cách một cộng đồng mã nguồn mở theo dõi vấn đề. Mozilla chấp nhận báo cáo lỗi và yêu cầu cải tiến tính năng cho tất cả các sản phẩm của họ thông qua cổng thông tin này.

2. Tìm hiểu thêm về kiểm thử phần mềm mã nguồn mở tại http://www.opensourcetesting.org/.

3. Tìm hiểu về "Lịch sử và Khái niệm Đóng gói Phần mềm Linux" từ http://wiki.rpath.com/wiki/Appliance_Development:Linux_Software_Packaging_History_and_Concepts.

4. Cộng đồng Apache sử dụng hệ thống quản lý phiên bản Subversion (SVN) và cho phép mọi người truy cập mã nguồn qua internet. Tìm hiểu cách bạn có thể truy cập kho mã nguồn của họ và làm việc với nội dung từ http://www.apache.org/dev/version-control.html#source-code-repositories.

4.4 Tóm tắt

Trong chương này, bạn đã học cách phát triển phần mềm nguồn mở như một cộng đồng sáng kiến và cách một dự án đi từ ý tưởng đến sản phẩm nổi bật. Chương đã thảo luận tất cả các giai đoạn của vòng đời phát triển phần mềm, chẳng hạn như thiết kế, viết mã, kiểm thử, xây dựng, đóng gói và phát hành; tập trung vào các nhóm khác nhau tham gia vào từng giai đoạn và cả về các công cụ phần mềm nguồn mở được sử dụng. 4.5 Câu hỏi ôn tập

1. Thuật ngữ "phát triển hướng tới cộng đồng" trong phần mềm có ý nghĩa gì ngành công nghiệp?

2. Nêu sự khác biệt chính giữa CVCS và DVCS.

3. Kể tên một số công cụ theo dõi vấn đề nguồn mở.

4. Các mục thường có trong gói OSS là gì?

5. Cài đặt phần mềm nguồn mở vào máy tính gồm những bước nào?

6. Định dạng nào sau đây là định dạng hợp lệ của gói OSS dành riêng cho Windows?

A. .tar

B. .zip

C. .gz

D. A và C

E. Tất cả những điều trên.

7. Điều nào sau đây được sử dụng trong quá trình thiết kế một PMNM?

A. UML

B. XML

C. CVS

D. Git

E. Không có điều nào ở trên

8. myproduct 3.4.5 – trong sơ đồ đánh số phiên bản này, là "thứ yếu" thành phần?

A. myproduct

B. 3

C. 4

D. 5

E. Không có điều nào ở trên

9. Hai công cụ nào sau đây có thể được sử dụng để tự động hóa quá trình xây dựng phần mềm?

A. Make

B. Apache

C. Ant

D. Darcs

E. Không có điều nào ở trên