# **LỜI CẢM ƠN**

Lời đầu tiên chúng em xin tỏ lòng biết ơn sâu sắc đến Thầy Trần Sơn Hải, người thầy đã cho chúng em những định hướng và những ý kiến rất quý báu về đề tài này. Cảm ơn Thầy đã chỉ dạy và hướng dẫn chúng em rất tận tình từ lúc lựa chọn đề tài cũng như trong quá trình thực hiện đồ án.

Chúng em cũng xin gửi lời cảm ơn đến Khoa CNTT trường Đại Học Công Nghệ TP HCM đã tạo điều kiện cho chúng em thực hiện đồ án môn học này. Cảm ơn các Thầy, Cô trong khoa đã tận tâm giảng dạy, cho chúng em những kiến thức thật hữu ích trong những năm qua.

Chúng em xin cảm Thầy Cô, bạn bè cùng khoá đã dìu dắt, giúp đỡ chúng em tiến bộ trong suốt những năm học qua. Xin cảm ơn gia đình và bè bạn, những người luôn bên cạnh, khuyến khích và giúp đỡ chúng em trong mọi hoàn cảnh khó khăn.

Được hoàn thành trong thời gian rất hạn hẹp, đồ án này chắc chắn còn nhiều khiếm khuyết, kính mong quý Thầy Cô và các bạn tận tình chỉ bảo. Chúng em xin cảm ơn những thầy cô, bạn bè và người thân đã và sẽ có những góp ý chân tình cho nội dung của đồ án này, để chúng em có thể tiếp tục đi sâu tìm hiểu và ứng dụng trong thực tiễn công tác.

Một lần nữa chúng em xin chân thành cảm ơn ./

TP.HCM, ngày 17 tháng 10 năm 2016

Nhóm sinh viên thực hiện.

**MỤC LỤC**

[**LỜI CẢM ƠN i**](#_Toc464476130)

[**MỤC LỤC ii**](#_Toc464476131)

[**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 1**](#_Toc464476132)

[**1.1. Khái niệm 1**](#_Toc464476133)

[**1.2. Yêu cầu hệ thống 1**](#_Toc464476134)

[**1.3. Tải và cài đặt Tortoise SVN 2**](#_Toc464476135)

[**1.4. Kho lưu trữ (Repository) 5**](#_Toc464476136)

[**1.5. Vấn đề chia sẻ tập tin 6**](#_Toc464476137)

[**1.6. Giải pháp Khóa-Sửa-Mở khóa 7**](#_Toc464476138)

[**1.7. Giải pháp Sao chép-Sửa đổi-Hợp nhất 9**](#_Toc464476139)

[**1.8. Subversion làm gì? 11**](#_Toc464476140)

[**1.9. Một số khái niệm chính trong SVN 12**](#_Toc464476141)

[**1.10. Một số action trong SVN 12**](#_Toc464476142)

[***1.10.1.* *Repo-browser.* 12**](#_Toc464476143)

[***1.10.2.* *Check Out.* 12**](#_Toc464476144)

[***1.10.3.* *Update.* 13**](#_Toc464476145)

[***1.10.4.* *Commit.* 13**](#_Toc464476146)

[***1.10.5.* *Conflict.* 13**](#_Toc464476147)

[***1.10.6.* *Create Patch*. 14**](#_Toc464476148)

[***1.10.7.* *Show Log.* 14**](#_Toc464476149)

[***1.10.8.* *Diff with previous version.* 14**](#_Toc464476150)

[**CHƯƠNG 2: ÁP DỤNG VÀO THỰC TIỄN 15**](#_Toc464476151)

[**2.1. Tạo project Github 15**](#_Toc464476152)

[**2.2. CheckOut project 16**](#_Toc464476153)

[**2.3. Xử lý Conflict 20**](#_Toc464476154)

[**2.4. Sử dụng 1 số action trong dự án nhóm 24**](#_Toc464476155)

# **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

## Khái niệm

TortoiseSVN là một trình máy khách Windows mã nguồn mở miễn phí cho hệ thống kiểm soát phiên bản Apache™ Subversion. TortoiseSVN quản lý tập tin và thư mục theo thời gian. Các tập tin được lưu trữ trong một kho trung tâm. Kho lưu trữ phần nhiều giống như một máy chủ tập tin bình thường, ngoại trừ việc nó nhớ tất cả các thay đổi đã từng được thực hiện trên các tập tin và thư mục của bạn. Điều này cho phép bạn khôi phục lại phiên bản cũ của tập tin của bạn và kiểm tra lịch sử của việc dữ liệu của bản đã được thay đổi như thế nào và khi nào, và ai đã thay đổi nó. Đây là lý do tại sao nhiều người nghĩ rằng Subversion và các hệ thống kiểm soát phiên bản nói chung là một loại của “cỗ máy thời gian”.

Một số hệ thống kiểm soát phiên bản cũng là phần mềm quản lý cấu hình (SCM). Các hệ thống này được thiết kế đặc biệt để quản lý cây mã nguồn, và có nhiều tính năng cụ thể cho phát triển phần mềm - chẳng hạn như sự hiểu biết bản xứ về ngôn ngữ lập trình, hoặc cung cấp các công cụ để xây dựng phần mềm. Tuy nhiên, Subversion không phải là một trong những hệ thống này, nó là một hệ thống tổng quát có thể được sử dụng để quản lý bất kỳ các tập tin, bao gồm cả mã nguồn.

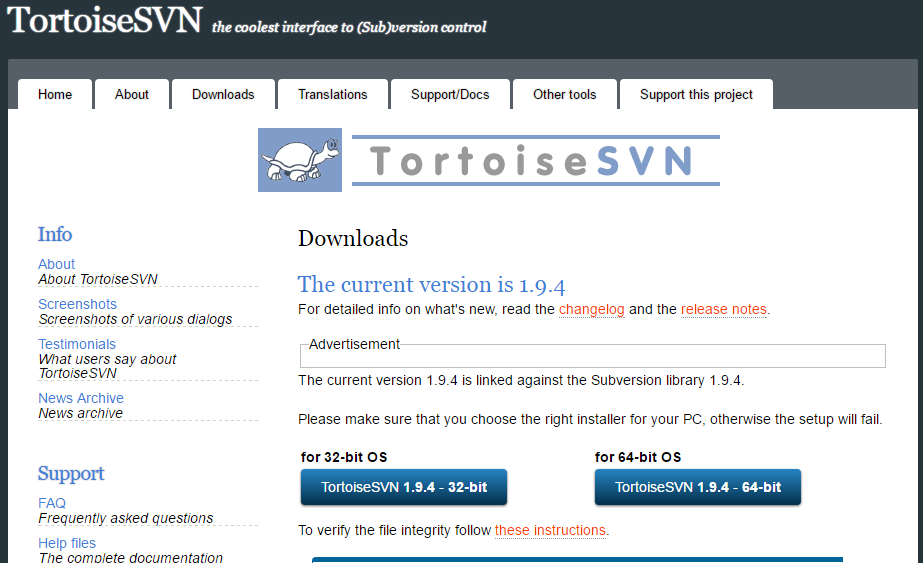
## Yêu cầu hệ thống

TortoiseSVN chạy trên Windows Vista hoặc cao hơn và đều hỗ trợ phiên bản 32-bit và 64-bit. Cài đặt cho Windows 64-bit cũng bao gồm các phần mở rộng 32-bit. Điều đó có nghĩa là bạn không cần phải cài đặt phiên bản 32-bit riêng biệt để có được menu ngữ cảnh TortoiseSVN và lớp phủ trong các ứng dụng 32-bit.

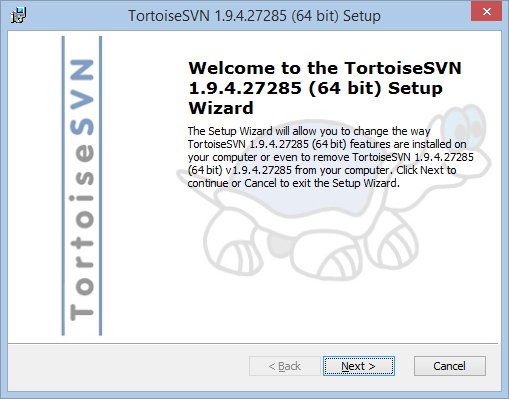
Hỗ trợ cho Windows 98, Windows ME và Windows NT4 ngừng hỗ trợ từ version 1.2.0, và Windows 2000 và XP SP2 ngừng hỗ trợ từ version 1.7.0. Hỗ trợ cho Windows XP SP3 ngừng hỗ trợ từ version 1.9.0. Bạn vẫn có thể tải về và cài đặt phiên bản cũ nếu bạn cần chúng.

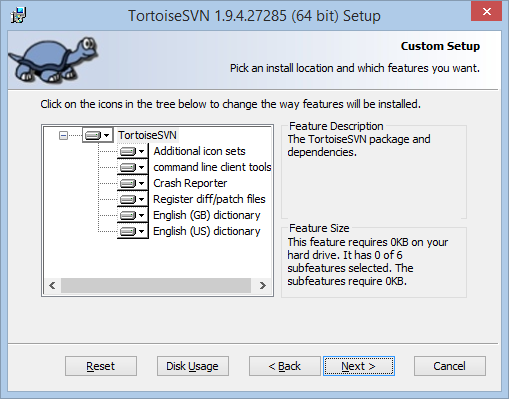
Dùng Github Repository

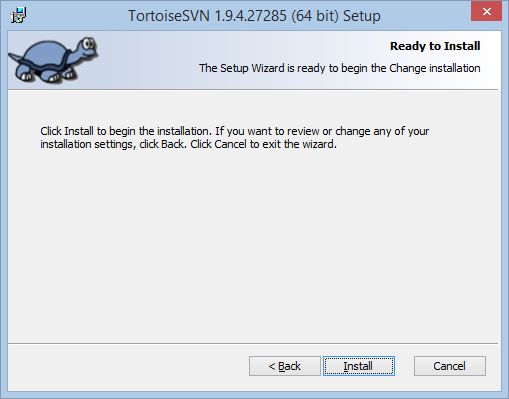
## Tải và cài đặt Tortoise SVN

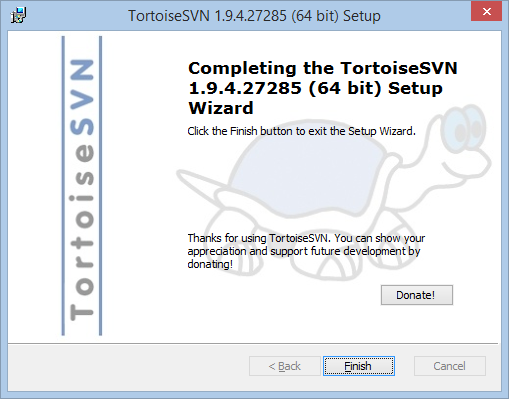


Tùy thuộc vào hệ điều hành của bạn là 64 hay 32 bit để download cho đúng.





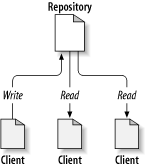




Việc cài đặt thành công, bạn cần phải khởi động lại máy tính.

## Kho lưu trữ (Repository)

Subversion là một hệ thống tập trung chia sẻ thông tin. Repository là một kho chưa mã nguồn, đó là trung tâm lưu trữ dữ liệu. Kho lưu trữ thông tin trong các hình thức của một cây hệ thống tập tin - Một hệ thống cấp bậc của các tập tin và thư mục. Số lượng bất kỳ các chương trình khách kết nối với kho lưu trữ, và sau đó đọc hoặc viết vào những tập tin này. Bằng cách ghi dữ liệu, chương trình khách hàng làm cho thông tin có sẵn cho những người khác; bằng cách đọc dữ liệu, chương trình khách hàng nhận được thông tin từ những người khác.



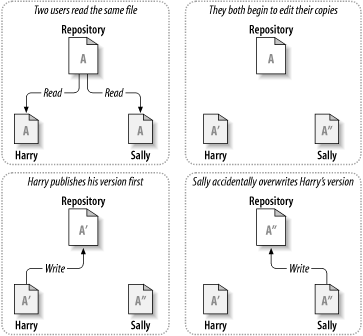
Hình 2.1. Một hệ thống Máy Khách/ Máy Chủ điển hình

Vì vậy, tại sao điều này thú vị? Cho đến nay, điều này nghe giống như định nghĩa của một máy chủ tập tin thông thường. Và quả thực, các kho lưu trữ là một loại máy chủ tập tin, nhưng nó không giống loại bình thường của bạn. Những gì làm cho kho lưu trữ Subversion đặc biệt là nó nhớ mọi sự thay đổi từng được viết vào nó: mỗi lần thay đổi cho mỗi tập tin, và thậm chí thay đổi cây thư mục, chẳng hạn như việc bổ sung, xóa, và sắp xếp lại các tập tin và thư mục.

When a client reads data from the repository, it normally sees only the latest version of the filesystem tree. But the client also has the ability to view previous states of the filesystem. For example, a client can ask historical questions like, “what did this directory contain last Wednesday?”, or “who was the last person to change this file, and what changes did they make?” These are the sorts of questions that are at the heart of any version control system: systems that are designed to record and track changes to data over time.

## Vấn đề chia sẻ tập tin

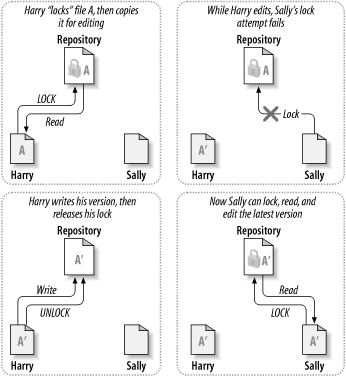
Hãy xem xét kịch bản này: giả sử chúng ta có hai đồng nghiệp, Harry và Sally. Mỗi quyết định chỉnh sửa các tập tin kho lưu trữ tại cùng một thời gian. Nếu Harry lưu thay đổi của mình vào kho lưu trữ đầu tiên, có thể một vài khoảnh khắc sau đó Sally vô tình có thể ghi đè lên chúng với phiên bản mới của tập tin. Trong khi Harry phiên bản của tập tin sẽ không bị mất mãi mãi (do hệ thống nhớ mỗi lần thay đổi), bất kỳ thay đổi nào Harry đã thực hiện sẽ không có mặt trong phiên bản mới hơn của Sally của tập tin, bởi vì cô không bao giờ nhìn thấy những thay đổi để bắt đầu với Harry. Tác phẩm của Harry vẫn còn có hiệu quả bị mất hoặc ít nhất là mất tích từ các phiên bản mới nhất của tập tin - và có lẽ do tai nạn. Điều này chắc chắn là một tình huống mà chúng ta muốn tránh!



Hình 2.2. Vấn đề cần lưu ý

## Giải pháp Khóa-Sửa-Mở khóa

Nhiều phiên bản kiểm soát hệ thống sử dụng một mô hình khoá-thay đổi-mở khóa để giải quyết vấn đề này, mà là một giải pháp rất đơn giản. Trong hệ thống như vậy, kho lưu trữ chỉ cho phép một người thay đổi một tập tin tại một thời điểm. Đầu tiên Harry phải khóa các tập tin trước khi có thể bắt đầu thay đổi nó. Khóa một tập tin là rất giống như mượn một cuốn sách từ thư viện, nếu Harry đã khóa một tập tin, thì sau đó Sally không thể thực hiện bất kỳ thay đổi với nó. Nếu cô ấy cố gắng để khóa các tập tin, các kho lưu trữ sẽ từ chối yêu cầu. Tất cả những gì có thể làm là đọc các tập tin, và chờ đợi cho Harry để kết thúc những thay đổi của mình và phát hành khóa của mình. Sau khi Harry mở các tập tin, hết lượt của mình, và bây giờ Sally có thể lần lượt của mình bằng cách khóa và chỉnh sửa.



Hình 2.3. Giải pháp Khóa-Sửa-Mở khóa

Vấn đề với mô hình khóa, sửa đổi, mở khóa là nó có một chút hạn chế, và thường trở thành một rào cản cho người sử dụng:

Khóa có thể gây ra các vấn đề hành chính. Đôi khi Harry sẽ khóa một tập tin và sau đó quên nó. Trong khi đó, bởi vì Sally là vẫn đang chờ đợi để chỉnh sửa các tập tin, bàn tay của cô bị bó buộc. Và sau đó Harry đi vào kỳ nghỉ. Bây giờ Sally phải nhờ một quản trị viên để phát hành khóa của Harry. Tình hình kết thúc lên gây ra rất nhiều sự chậm trễ không cần thiết và lãng phí thời gian.

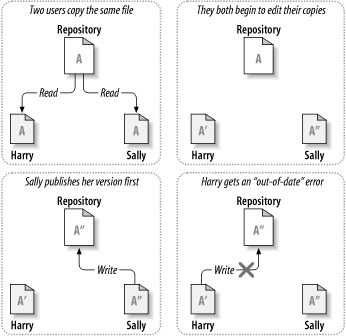
Khóa có thể gây ra tuần tự không cần thiết. Nếu Harry là chỉnh sửa bắt đầu của một tập tin văn bản, và Sally chỉ đơn giản là muốn chỉnh sửa cuối cùng một tập tin? Những thay đổi này không chồng chéo lên nhau chút nào. Họ có thể dễ dàng chỉnh sửa các tập tin cùng một lúc, và không có hại lớn, giả sử những thay đổi sáp nhập với nhau đúng cách. Không cần cho họ phải thay phiên nhau trong tình huống này.

Khóa có thể tạo ra một cảm giác an toàn sai lầm. Giả vờ rằng Harry khóa và chỉnh sửa tập tin A, trong khi đồng thời Sally khóa và chỉnh sửa các tập tin B. Nhưng giả sử rằng A và B phụ thuộc vào nhau, và các thay đổi được thực hiện cho từng ngữ nghĩa không tương thích. Đột nhiên A và B không làm việc cùng nhau nữa. Hệ thống khóa bất lực để ngăn chặn vấn đề nhưng nó bằng cách nào đó cung cấp một cảm giác an toàn sai. Thật dễ dàng cho Harry và Sally để tưởng tượng rằng các tập tin khóa, mỗi bắt đầu một công việc an toàn, cách biệt, và do đó ức chế sự thảo luận về những thay đổi không tương thích của họ sớm hơn.

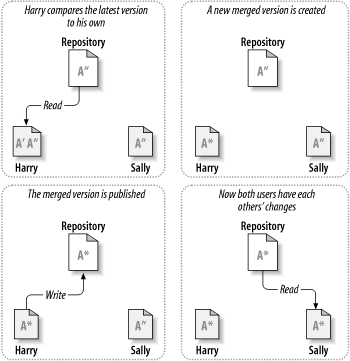
## Giải pháp Sao chép-Sửa đổi-Hợp nhất

Subversion, CVS, and other version control systems use a copy-modify-merge model as an alternative to locking. In this model, each user's client reads the repository and creates a personal working copy of the file or project. Users then work in parallel, modifying their private copies. Finally, the private copies are merged together into a new, final version. The version control system often assists with the merging, but ultimately a human being is responsible for making it happen correctly.

Dưới đây là một ví dụ. Hãy nói rằng Harry và Sally đều tạo ra các bản sao làm việc cùng một dự án, sao chép từ kho. Họ làm việc đồng thời, và thay đổi cùng một tập tin A trong bản sao của họ. Sally lưu thay đổi của mình vào kho lưu trữ đầu tiên. Khi Harry cố gắng để lưu các thay đổi của ông sau đó, kho lưu trữ thông báo cho ông rằng tập tin của mình quá cũ. Nói cách khác, tập tin A đó trong kho lưu trữ bằng cách nào đó thay đổi kể từ khi ông mới sao chép nó. Vì vậy, Harry yêu cầu khách hàng của mình để hợp nhất bất kỳ thay đổi mới nhất từ ​​kho vào bản sao hoạt động của tập tin A. Có thể là những thay đổi của Sally không chồng chéo lên nhau với bản của riêng ông ta, do đó, một khi ông có cả bộ những thay đổi tích hợp, ông lưu bản sao làm việc của mình trở lại vào kho.



Hình 2.4. Giải pháp Sao chép-Sửa đổi-Hợp nhất



Hình 2.5. Sao chép-Sửa đổi-Hợp nhất Tiếp theo

Nhưng điều gì sẽ xảy ra nếu thay đổi của Sally làm trùng với thay đổi của Harry? Những gì sau đó? Tình trạng này được gọi là một xung đột, Và nó thường là không phải là một vấn đề gì nhiều. Khi Harry yêu cầu khách hàng của mình để hợp nhất các thay đổi kho lưu trữ mới nhất vào bản sao làm việc của mình, bản sao của tập tin A là bằng cách nào đó gắn cờ là trong trạng thái xung đột: anh ta sẽ có thể nhìn thấy cả hai bộ những thay đổi xung đột, và tự lựa chọn giữa chúng. Lưu ý rằng phần mềm có thể không tự động giải quyết xung đột, con người chỉ có khả năng hiểu biết và thực hiện những lựa chọn thông minh cần thiết. Khi Harry đã giải quyết thủ công những thay đổi chồng chéo (có lẽ bằng cách thảo luận về cuộc xung đột với Sally), ông một cách an toàn có thể lưu các tập tin bị sáp nhập trở lại vào kho.

Mô hình sao chép, sửa đổi, sáp nhập có thể nghe một chút hỗn loạn, nhưng trong thực tế, nó chạy rất trơn tru. Người dùng có thể làm việc song song, không bao giờ chờ đợi nhau. Khi họ làm việc trên các tập tin tương tự, nó hóa ra rằng hầu hết các thay đổi đồng thời của họ không chồng chéo lên nhau ở tất cả các xung đột hiếm khi xảy ra. Và số lượng thời gian cần thiết để giải quyết xung đột là ít hơn so với thời gian đã mất bằng một hệ thống khóa.

Cuối cùng, nó đi xuống đến một yếu tố quan trọng: người sử dụng thông tin liên lạc. Khi người sử dụng giao tiếp kém, cả hai cú pháp và ngữ nghĩa xung đột gia tăng. Không có hệ thống có thể buộc người dùng để giao tiếp hoàn hảo, và không có hệ thống có thể phát hiện xung đột ngữ nghĩa. Vì vậy, không có điểm bị đẩy vào một lời hứa sai lầm rằng một hệ thống khóa bằng cách nào đó sẽ ngăn chặn xung đột, trong thực tế, khóa có vẻ ức chế năng suất hơn bất cứ điều gì khác.

Có một tình hình phổ biến nơi mô hình khóa-sửa đổi-mở khóa cho kết quả tốt hơn, và đó là nơi mà bạn có các tập tin không hợp nhất được. Ví dụ, nếu kho lưu trữ của bạn có chứa một số hình ảnh đồ họa, và hai người thay đổi hình ảnh tại cùng một thời gian, không có cách nào cho những thay đổi được sáp nhập với nhau. Hoặc là Harry hoặc Sally sẽ mất các thay đổi.

## Subversion làm gì?

Subversion sử dụng các bản sao chỉnh sửa, kết hợp giải pháp theo mặc định, và trong nhiều trường hợp này là tất cả những gì mà bạn cần. Tuy nhiên, phiên bản 1.2, Subversion cũng hỗ trợ các khóa tập tin, vì vậy nếu bạn có các tập tin không hợp nhất được, hoặc nếu bạn chỉ đơn giản là bị ép buộc vào một chính sách khóa bởi ban quản lý, Subversion vẫn sẽ cung cấp các tính năng bạn cần.

## Một số khái niệm chính trong SVN

Repository: server chứa SVN và đặt Project

* Head: phiên bản mới nhất trên server
* Master: thông thường khi tạo SVN, có một branch chính được tạo ra và gọi là master, các branch phụ thường được đặt theo tên feature mà branch đó được tạo. Với hình trên, đường nối các ô màu xanh lá thể hiện master, các đường nối ô màu vàng thể hiện branch được tạo ra
* Change: mô tả sự thay đổi cụ thể của 1 revision so với revision trước đó
* Working copy: bản copy của toàn bộ SVN tại máy của developer
* Conflict: xung đột xảy ra khi có nhiều developer cùng làm việc với 1 working copy

## Một số action trong SVN

1. *Repo-browser.*

Repo-Browser cho phép chúng ta xem cấu trúc và tình trạng của Repository. Với Repo-Browser mình có thể thực hiện các tác vụ như sao chép, di chuyển, tạo mới, xoá đổi tên thư mục, file trực tiếp trên Repository.

Chọn TortoiseSVN -> Repo-browser.

1. *Check Out.*

Để làm việc trên phiên bản copy, bạn cần phải check-out từ repository, đơn giản chỉ cần Click chuột phải vào thư mục cần lưu trữ phiên bản copy hoặc Click chuột phải vào thư mục cần copy trên Repo-Browser và chọn SVN Checkout.

Một hộp thoại xuất hiện yêu cầu chọn thư mục chứa. Nếu folder chưa được tạo, sau đó nó sẽ tự động tạo cho bạn. Chú ý bạn nên chọn thư mục rỗng.

1. *Update.*

Sau khi CheckOut phiên bản copy về máy để làm việc, chúng ta có thể thường xuyên cập nhật (Update) những phiên bản mới nhất về bằng cách Click chuột phải lên thư mục đã copy về máy và chọn SVN Update.

1. *Commit.*

Để lưu những thay đổi của bạn vào repository trên server, bạn chọn file đã được thay đổi và Click chuột phải chọn SVN Update. Sau khi không có bất kỳ xung đột nào xảy ra thì chúng ta chấp nhận thay đổi bằng cách Click chuột phải chọn SVN Commit.

Để lại message để sau này chúng ta dễ dàng lựa chọn phiên bản.

Sau mỗi lần bạn Commit thì Revision (phiên bản) sẽ tự động tăng lên một.

1. *Conflict.*

Thông thường khi một File mà có nhiều người cùng làm việc trên đó thì khi Update thường xảy ra tình trạng như sau:

Vì vậy để tránh việc xung đột thì người quản lý dự án cần phải phân quyền, phân việc sao cho hợp lý. Một File tại một thời điểm chỉ duy nhất một người làm việc với nó.

Giải quyết conflict:

Mở file bị xung đột, tìm kiếm những hàng bắt đầu với chuỗi <<<<<<<.

Trước hết bạn cần phải update. Click chuột phải, chọn SVN Update. Mở file bị xung đột và xóa đi vùng tương phản.

Sau khi chỉnh sửa xong file conflict, bạn phải thông báo cho SVN biết rằng xung đột đã được giải quyết bằng cách click chuột phải, chọn TortoiseSVN >> Resolved… Sau đó mới có thể commit được.

1. *Create Patch*.

Create Patch giúp các người quản lý và các thành viên trong dự án có thể kiểm soát được những thay đổi trong một file tại thời điểm nào đó như ai đã thay đổi và thay đổi những gì?

Để Create Patch thì trước khi Commit những thay đổi của bạn lên Server ta làm như sau: trên thư mục chứa File thay đổi ta Click chuột phải chọn TortoiseSVN ->Create Patch.

Sau đó ta sẽ lựa chọn những File mà muốn tạo Create Patch và nó sẽ tạo ra file có dạng như sau:

1. *Show Log.*

Click chuột phải và chọn TortoiseSVN -> Show Log để xem thay đổi revision. Nhìn vào bảng Log Messages ta có thể thấy được rất chi tiết từ ngày … , đến ngày … có những ai đăng nhập vào Server và làm những gì với file nào. Như vậy người quản lý dự án có thể biết được các thành viên trong dự án đang làm gì và từ đó đề xuất kế hoạch làm việc tiếp theo.

1. *Diff with previous version.*

Cho thấy được sự khác nhau của file hiện tại đang làm việc so với phiên bản liền kề trước đó.

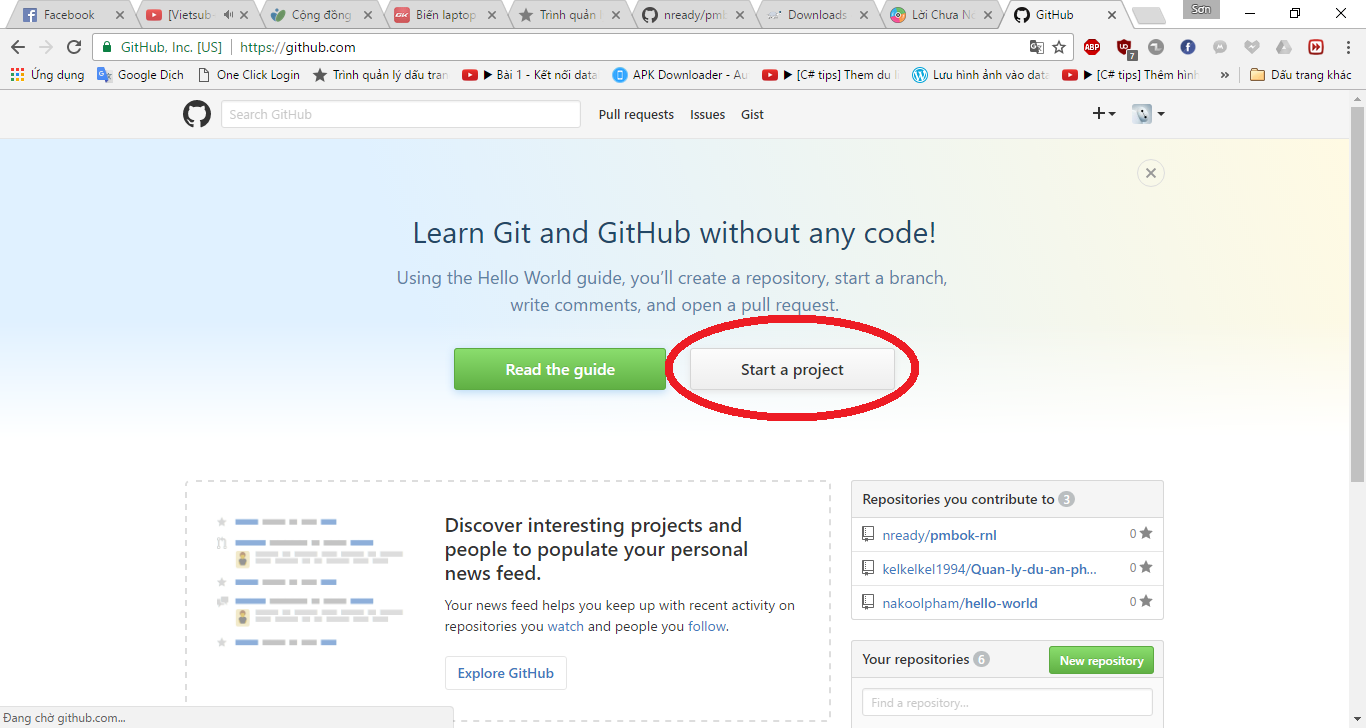
Click chuột phải vào File và chọn TortoiseSVN -> Diff.

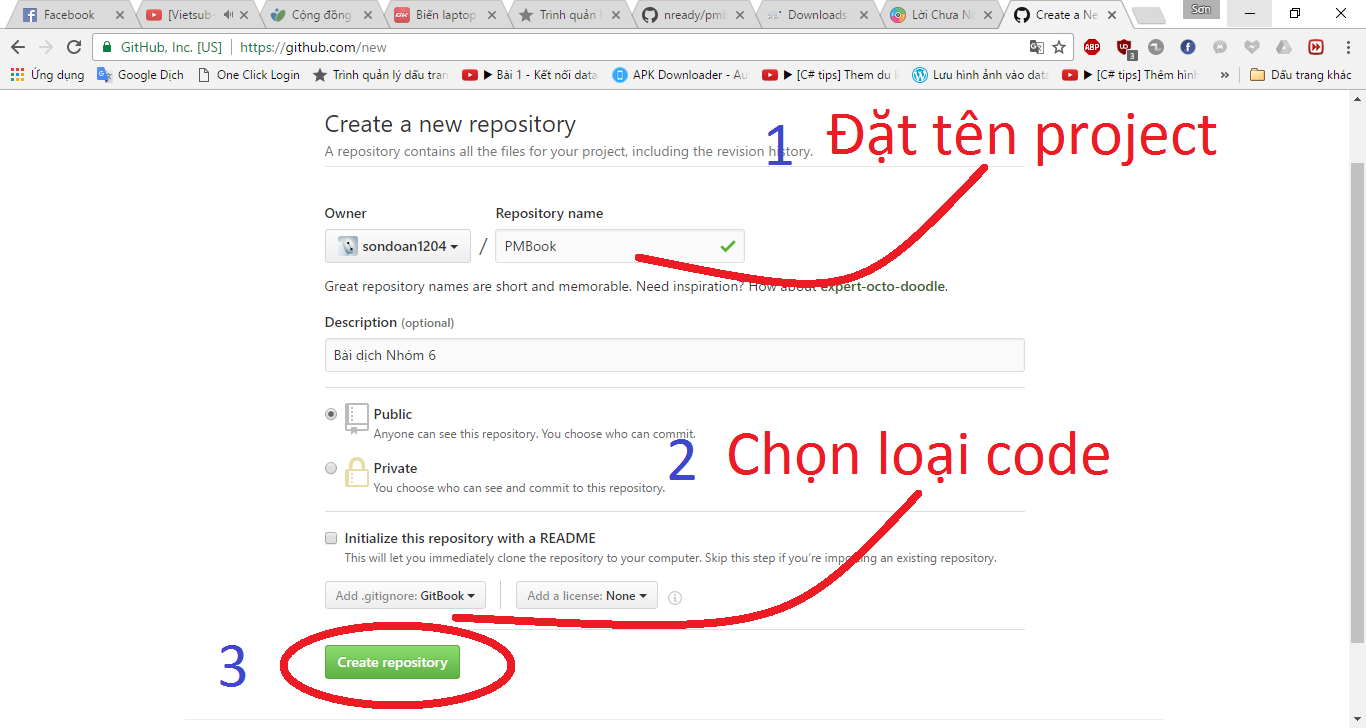
# **CHƯƠNG 2: ÁP DỤNG VÀO THỰC TIỄN**

Do dự án sử dụng chủ yếu sử dụng file txt để viết văn bản, chỉnh sửa ít nên không sử dụng được hết tính năng cùa SVN

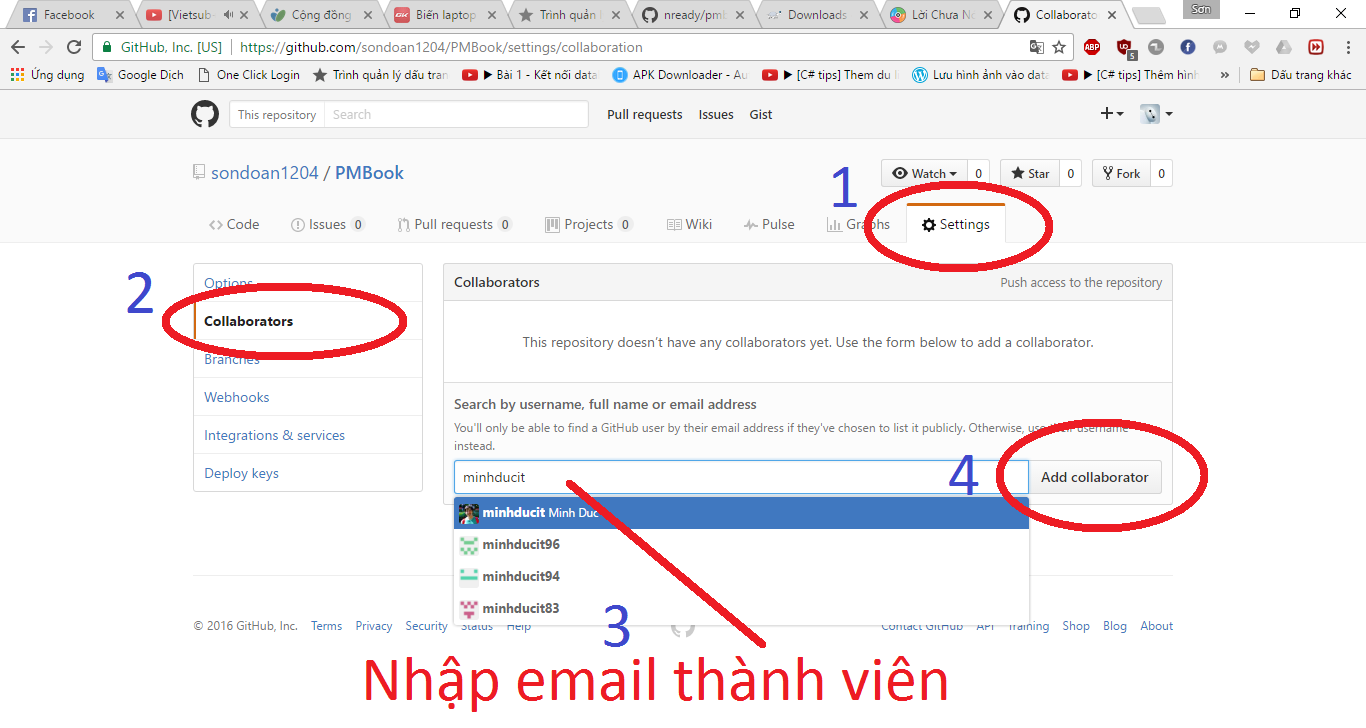
1. **Tạo project Github**

Truy câp <https://github.com/>, tạo accout, kích hoạt tài khoản, đăng nhập. Chọn Star a project



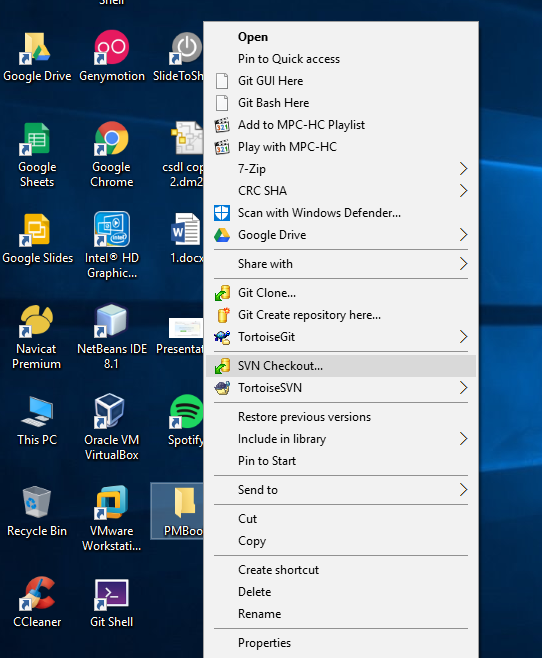


Thêm các thành viên nhóm



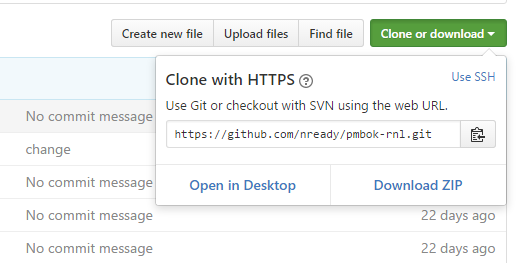
1. **CheckOut project**

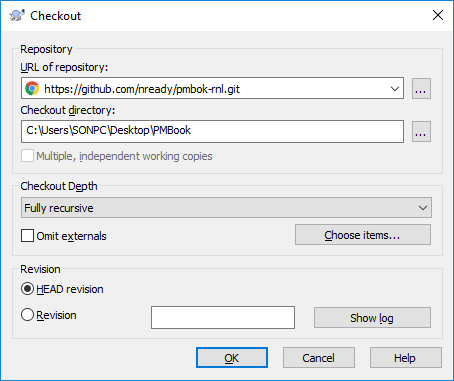
Tạo Folder chứa source dự án nhóm

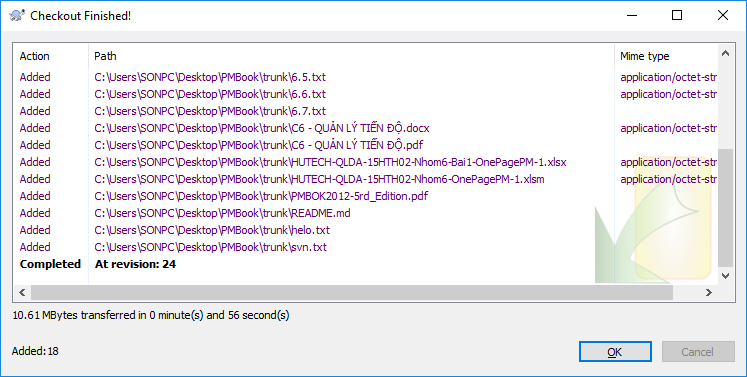


Checkout… (có thể sử dụng github, google code, …) ở đây, nhóm mình sử dụng gitub

Github “https://github.com/nready/pmbok-rnl.git”

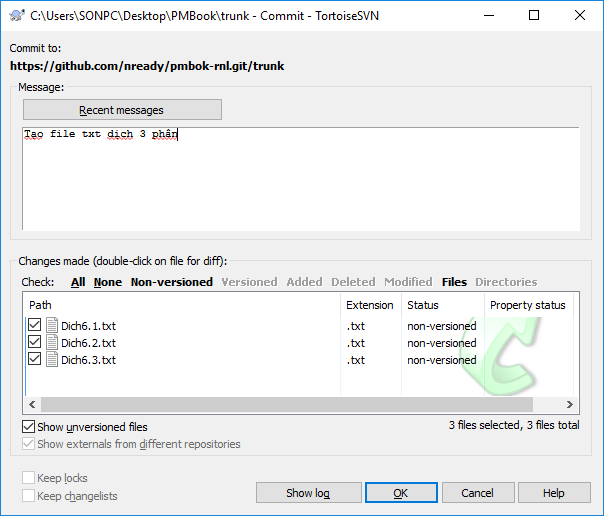


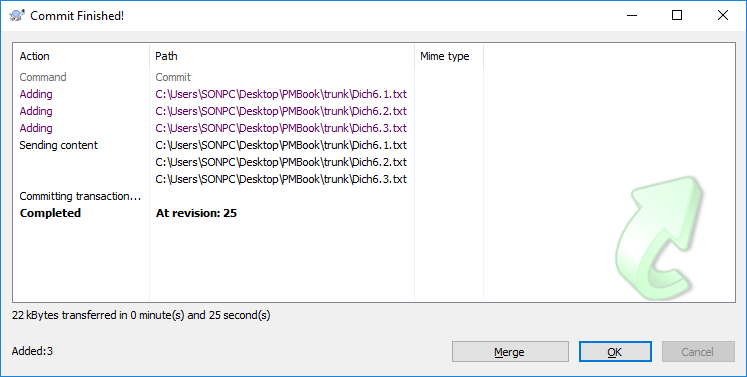




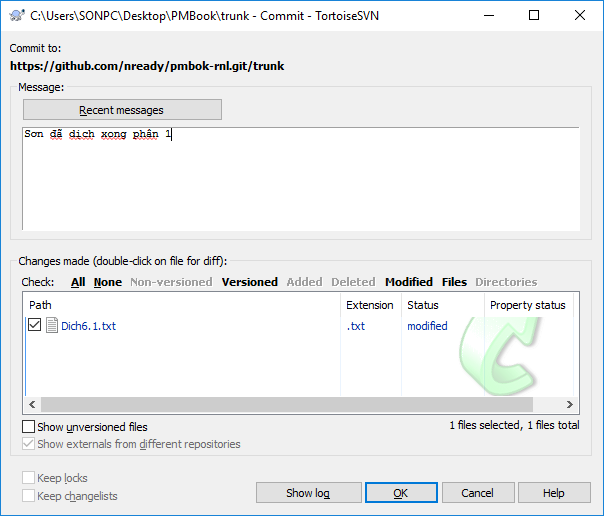
Tạo file đã dịch, sau đó Commit lên server (Repository)

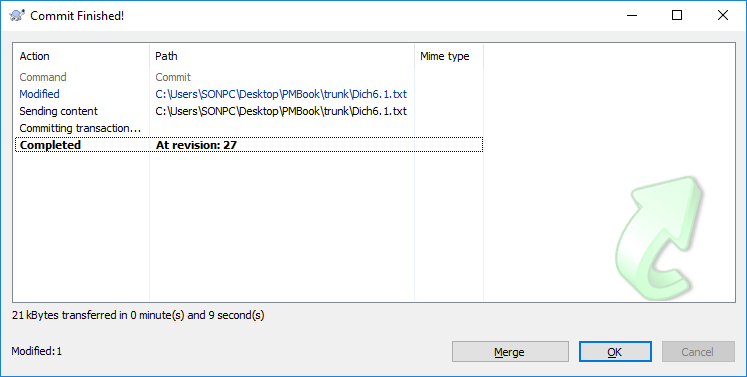
Nhập user và password tài khoản Repository sử dụng checkout về





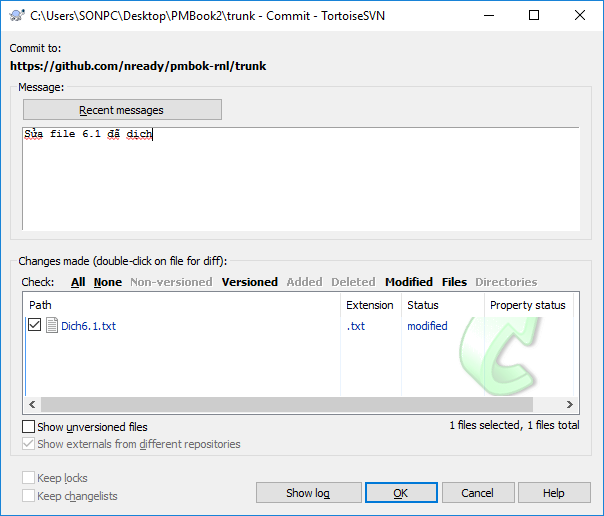
Mở file txt 6.1 tiến hành dịch bài từ file đã google translate, Sau đó thực hiện Commit lên lại server

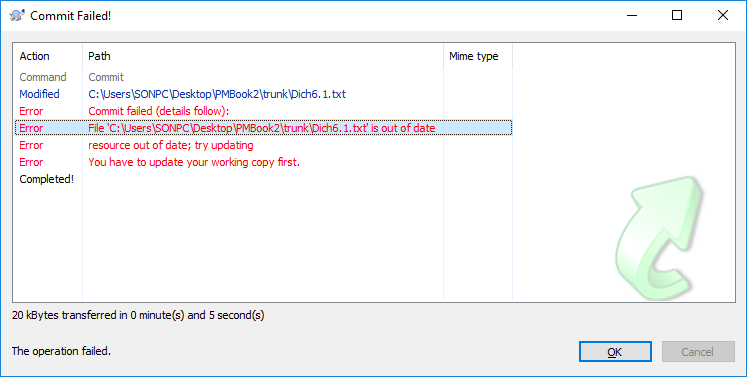




1. **Xử lý Conflict**

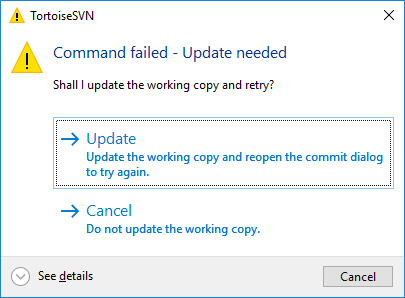
Lúc này, revision mới nhất là 27. Tại 1 máy khác nếu revision thấp hơn 27 mà sửa trực tiếp lên file chưa đc update mới nhất rồi Commit lên server sẽ bị báo lỗi Conflict



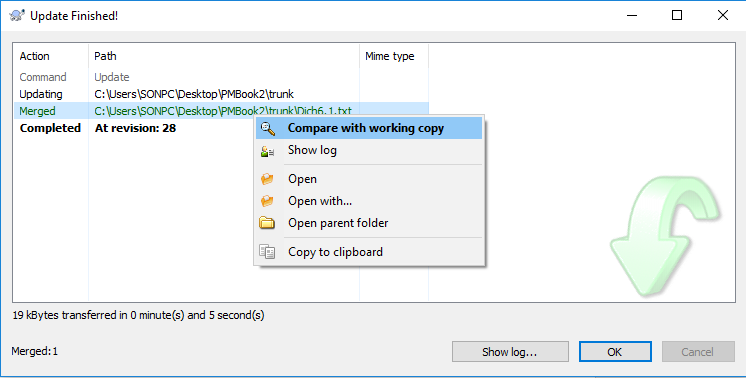


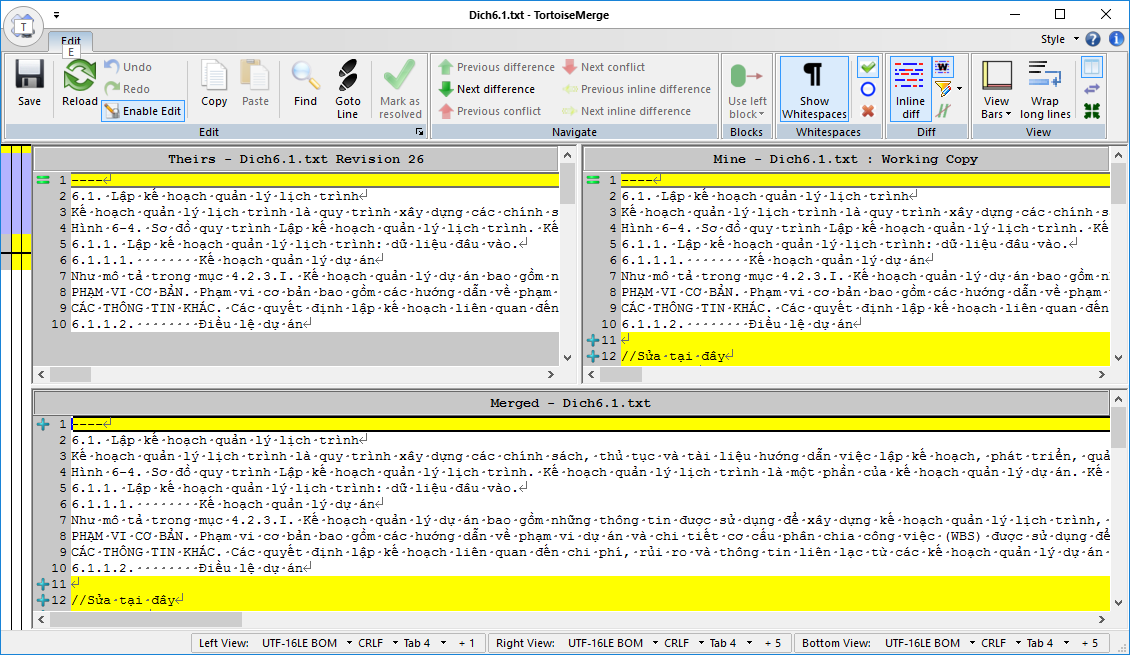
Sửa lỗi

Update lại phiên bản mới nhất của file.



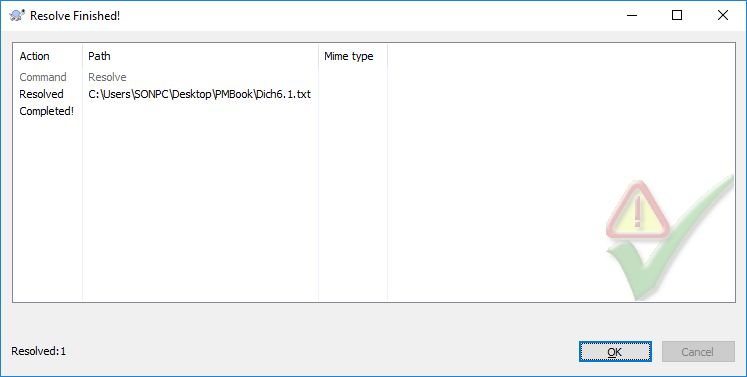
Tiến hành sửa Conflict bằng cách Compare, sửa lại, kiểm tra. Nếu không ảnh hưởng gì thì tiến hành Merged và Commit lên lại bản mới nhất

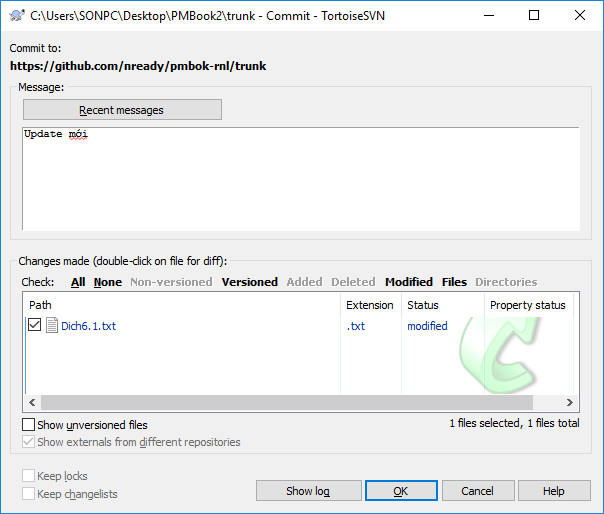


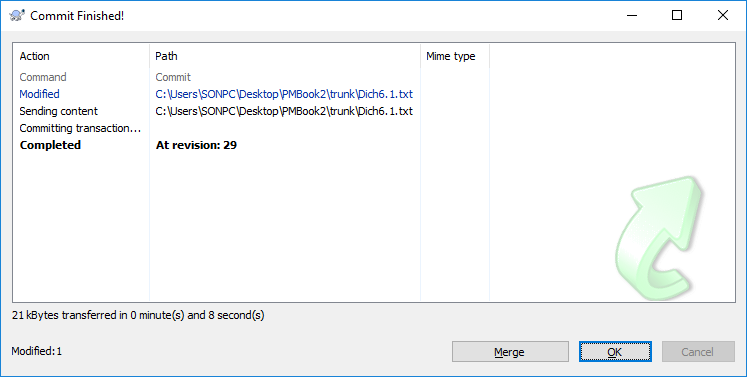


So Sánh sửa lỗi, Save lại và Commit lại

Trước khi Commit lai phai Resolve với SVN mới tiến hành Commit

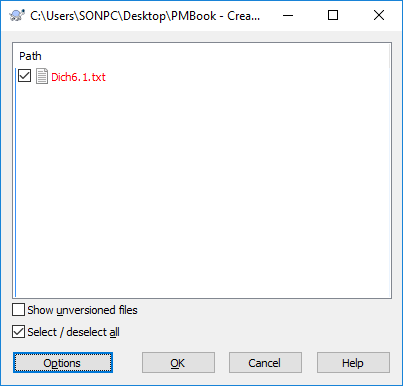


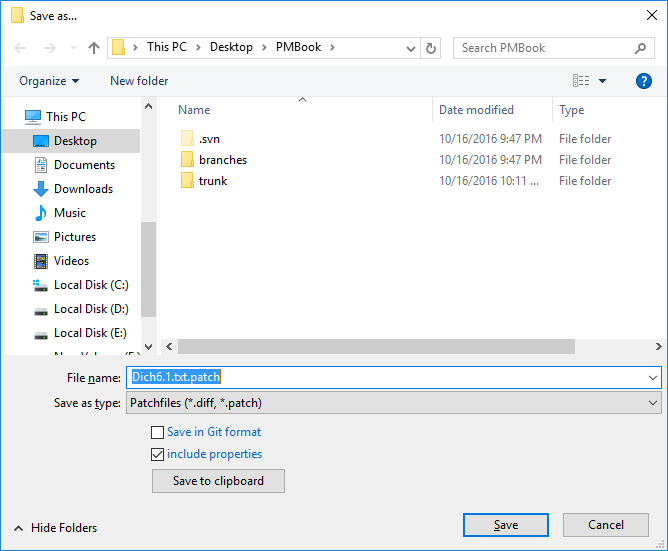


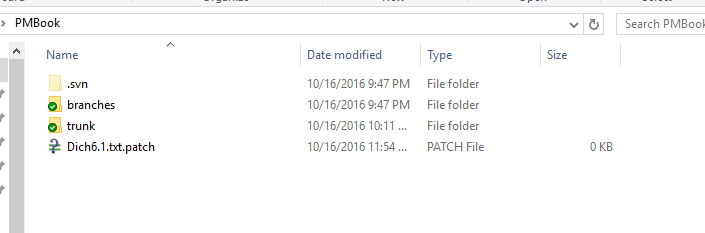


1. **Sử dụng 1 số action trong dự án nhóm**

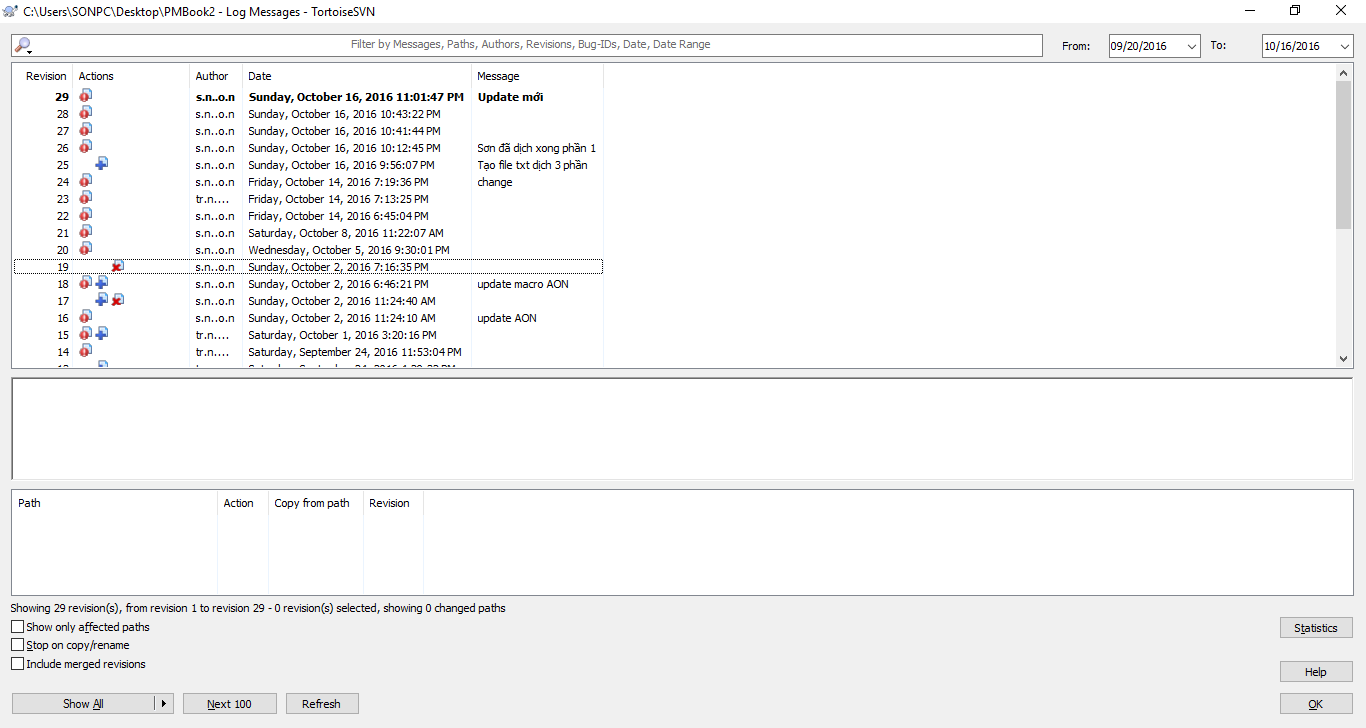
Create Patch – Trước khi Commit, các thành viên tạo patch để mọi người còn lại có thể thấy đc thay đổi của project







Show Log- xem lại các lịch sử và có thể quay lại phiên bản trước do Commit sai



Lock

