TortoiseSVN - Quick Guide

Written by : Marc Chevaldonné, IUT Informatique, Université d'Auvergne Clermont1

12 octobre 2009
Version 2.0

Sommaire

Introduction 3

Contexte 3

Documents de références 3

Table of modifications 3

1. Présentation rapide de Subversion 4
   - 1.1. Qu'est-ce que Subversion ? Qu'est-ce qu'un Système de Gestion de Versions ? 4
   - 1.2. Principe 4
   - 1.3. Problème du partage de fichiers 5
     - 1.3.1.1. Solution 1 : Lock - Modify - Unlock 5
   - 1.4. Le repository 7
     - 1.4.1.1. L'arbre des révisions 7
     - 1.4.1.2. Organisation du repository : trunk - branches - tags 8
2. Installation de Tortoise SVN 9
3. Config File 9
4. Utilisation quotidienne de TortoiseSVN 11
   - 4.1. Checkout 11
   - 4.2. Commit 12
   - 4.3. Add 13
   - 4.4. Delete 15
   - 4.5. Move 15
   - 4.6. Rename 16
   - 4.7. Update 16
   - 4.8. Conflicts 17
5. Utilisation avancée 22
   - 5.1. Brancher / Étiqueter (branching/tagging) 22
   - 5.2. Fusionner (Merge) 22
   - 5.3. Revision Graph 23

6. Quelques règles de bonne conduite 24

1. Introduction

1.1. Contexte

Ce document a été écrit pour les élèves de GI (2ème année de l'IUT d'Informatique de Clermont-Ferrand, Université d'Auvergne). Son but est de décrire succinctement les fonctionnalités de Subversion et les outils de base pour une utilisation via TortoiseSVN sous Windows.

1.2. Documents de références

Pour plus d'informations, le lecteur peut se documenter en lisant les documentations suivantes :

Title	Version	Lien
TortoiseSVN - A Subversion client for Windows	1.6.5	http://tortoisesvn.net/support
Version Control with Subversion	1.5	http://svnbook.red-bean.com/

1.3. Table of modifications

Author	Modification	Version	Date
Marc Chevaldonné	First draft	1.0	08 décembre 2008
Marc Chevaldonné	Première version complète	2.0	12 octobre 2009

2. Présentation rapide de Subversion

2.1. Qu'est-ce que Subversion ? Qu'est-ce qu'un Système de Gestion de Versions ?

Subversion est un système de gestion de versions, c'est-à-dire qu'il permet la gestion de dossiers et de fichiers ainsi que leurs modifications au cours du temps. Cela permet de récupérer des versions anciennes de documents ou d'examiner l'historique des modifications apportées. On parle aussi de «time machine».

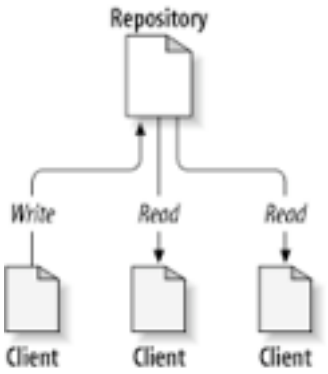
Subversion peut-être utilisé en réseau, ce qui permet son utilisation par des personnes distantes. Il est donc possible à un groupe de personnes de modifier et de gérer un même ensemble de données à distance et de manière collaborative. En effet, un outil tel que Subversion permet une production en parallèle et non linéaire, sans risque de confusion ou de pertes de données grâce au versionnage.

Les documents versionnés sont généralement du texte, et en particulier du code source, mais il peut s'agir de n'importe quel type de documents.

Subversion s'utilise en lignes de commande, mais il existe de nombreux shells permettant de les utiliser de manière plus convivial. Les plus connus et utilisés sont TortoiseSVN sous Windows et SCPlugin sous MacOSX.

2.2. Principe

Un serveur centralise tous les fichiers constituant un projet dans ce qu'on appelle la base centrale : le «repository».



Un utilisateur a des droits d'écriture et/ou de lecture sur les fichiers stockés.

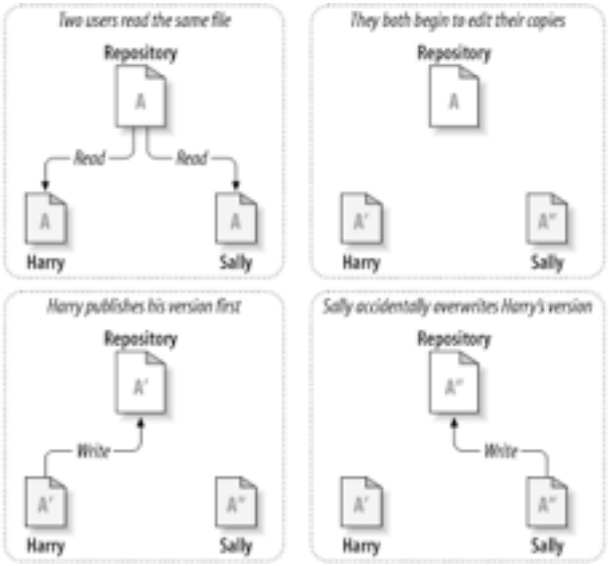
Un utilisateur rapatrie sur son poste de travail une version (généralement la dernière) des fichiers et travaille ainsi toujours sur une version local du projet.

Avant de pouvoir modifier un fichier du projet, l'utilisateur doit l'extraire, c'est-à-dire qu'il avertit le serveur qu'il en prend possession.

Une fois qu'il a terminé la modification, l'utilisateur archive le/les fichiers. Le fichier est renvoyé vers le serveur. Ce dernier fusionne les modifications effectuées par l'utilisateur à sa version courante du fichier.

2.3. Problème du partage de fichiers

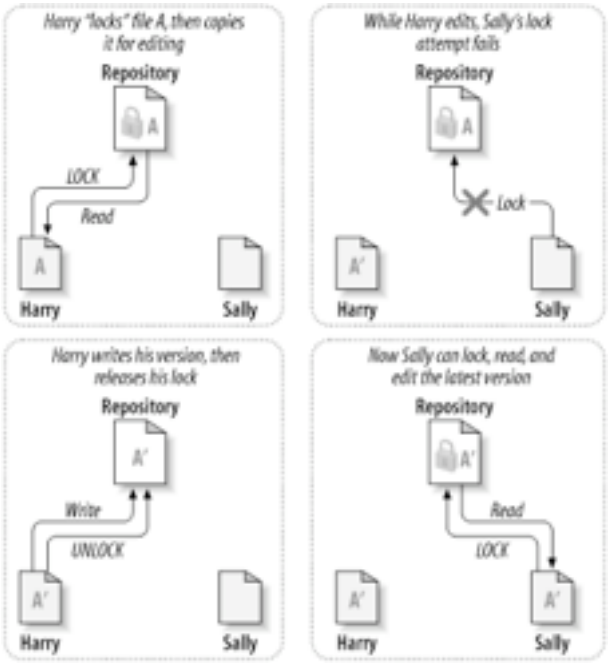
Un des problèmes majeurs des systèmes de gestion de versions est le partage de fichiers. Comment permettre à plusieurs utilisateurs de modifier le même fichier ?



- ➡ Harry et Sally prennent tous les deux une copie en local d'un fichier versionné sur le serveur.
- ➡ Harry et Sally le modifient de manière différente chacun de leur côté.
- ➡ Harry soumet ses modifications sur le serveur.
- ➡ Sally soumet ses modifications sur le serveur après Harry et efface toutes les modifications d'Harry accidentellement.

2.3.1. Solution 1 : Lock - Modify - Unlock

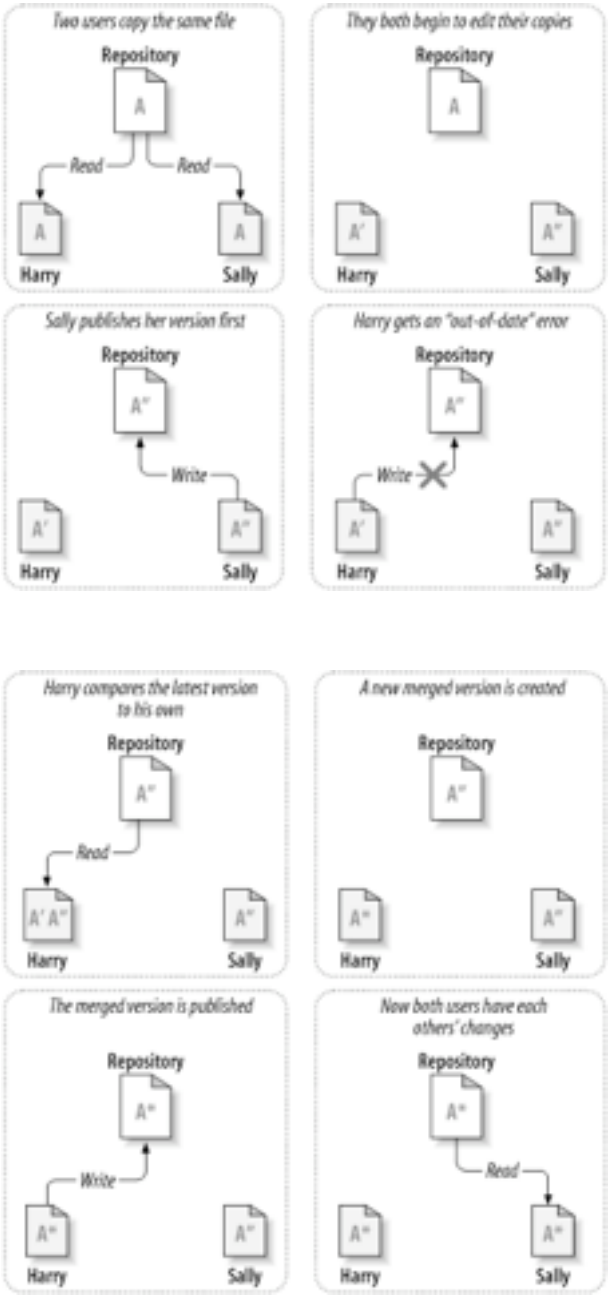
Avant Subversion, la solution unique consistait à verrouiller / modifier / déverrouiller.



- ➡ Harry récupère une copie du fichier sur le serveur et le verrouille.
- ➡ Sally essaye de récupérer une version du fichier pour le modifier, mais le fichier est verrouillé.
- ➡ Harry soumet ses modifications et déverrouille le fichier.
- ➡ Sally récupère une version du fichier maintenant déverrouillé et peut le modifier. Le fichier sur le serveur est verrouillé par Sally.

2.3.2. Solution 2 : Copy - Modify - Merge

Subversion a proposé une nouvelle solution qui consiste à copier en local une version du fichier, de la modifier, et de fusionner uniquement les différences avec la version du repository.



➡ Harry et Sally récupèrent une copie du fichier sur le serveur.

➡ Harry et Sally modifient chacun de leur côté et de manière différente le fichier.

➡ Sally soumet ses modifications et le fichier sur le repository est donc modifié.

➡ Harry veut soumettre ses modifications à son tour. Deux solutions : soit il n'y a aucun conflit entre les modifications d'Harry et les modifications de Sally, c'est-à-dire qu'ils ont modifié des parties très distinctes d'un même fichier (par exemple deux classes ou deux méthodes différentes), et dans ce cas, Subversion fusionne les modifications d'Harry au fichier modifié de Sally ; soit les modifications de Sally et de Harry recoupent les mêmes parties et Subversion ne sait donc pas faire la fusion. Dans ce dernier cas, Subversion empêche alors Harry de faire son commit car sa copie n'est pas à jour.

➡ Harry récupère alors la dernière version sur le repository et la compare avec sa version modifiée.

➡ Une nouvelle version qui fusionne les modifications d'Harry et la dernière version à jour du repository est créée, avec l'aide d'Harry.

➡ La version fusionnée (merged) devient la nouvelle dernière version du repository.

➡ Sally met à jour sa copie et les deux utilisateurs ont les modifications effectuées par l'un et l'autre des développeurs.

Subversion permet d'utiliser les deux solutions, même si la solution 2 est préférables (surtout quand le nombre de développeurs devient important). De plus, les développeurs ont généralement des tâches bien distinctes et les conflits sont très rares.

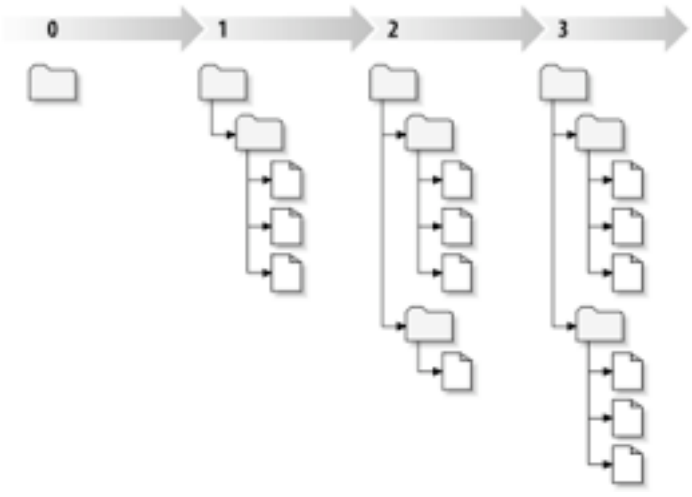
2.4. Le repository

Le repository est la base centrale où sont stockés les fichiers et les révisions. Un repository peut-être local ou distant, non sécurisé ou sécurisé. On accède à un repository via un URL.

Schema	Access method
`file:///`	accès direct sur un disque (local)
`http://`	accès via le protocole WebDAV et un serveur Apache
`https://`	comme le précédent mais avec des données cryptées (SSL)
`svn://`	accès via un protocole perso à un serveur `svnserve`
`svn+ssh://`	comme le précédent mais à travers un tunnel SSH

Dans le cas d'accès distant, l'administrateur du repository peut donner des droits d'accès en lecture / écriture à tout le repository ou seulement une partie du repository à des utilisateurs. Il peut par exemple donner des droits en lecture anonyme (pas besoin de se logger) et en écriture à des utilisateurs enregistrés avec mot de passe.

2.4.1. L'arbre des révisions



À chaque fois qu'un commit est réalisé par un utilisateur, Subversion crée un nouvel état de l'arborescence des dossiers et fichiers du projet versionné. Ce nouvel état est appelé une révision. À chaque révision, Subversion attribute un nombre entier unique, supérieur à celui de la révision précédente. La première révision est la numéro 0, et ne contient rien.

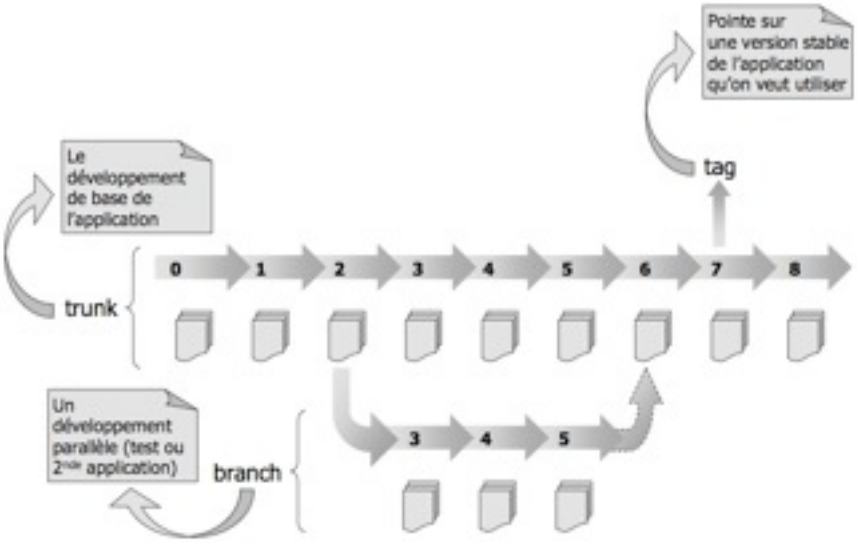
À chaque révision, Subversion n'enregistre que les différences avec la révision précédente. Pour récupérer une révision donnée, Subversion parcourt l'arbre des révisions et modifie les fichiers au fur et à mesure avec les différences enregistrées pour recréé l'état correspondant à cette révision.

Les révisions sont donc stockées sous forme d'arbre.

2.4.2. Organisation du repository : trunk - branches - tags

Subversion vous autorise à organiser votre repository comme vous l'entendez. Il existe de nombreuses bonnes organisations (mais encore plus de mauvaises...). Toutefois, il est fortement conseillé de suivre l'exemple du trunk - branches - tags (surtout pour les débutants).

Considérons un seul projet versionné via Subversion.



➡ Le dossier tronc (trunk) contient la ligne directrice de votre développement.

➡ Le dossier branches contient des copies du développement (une par branches). Les branches contiennent le même historique que le tronc au moment de la création de la branche. Ceci permet d'éviter d'avoir à refaire plusieurs fois la même chose pour des projets légèrement différents par exemple. Imaginons que nous souhaitons faire plusieurs versions d'un jeu : une en local et une autre pour le web. Nous développons d'abord le coeur du jeu qui sera commun à toutes les versions. Puis pour chaque version, nous créons une branche qui profitera de la partie commune déjà développée et de son historique et ajoutera ses propres particularités sans polluer le développement commun. Une branche peut également servir à faire un test de développement. Si ce test est approuvé, alors la branche peut être réintégrée dans le tronc, sinon, elle peut être coupée.

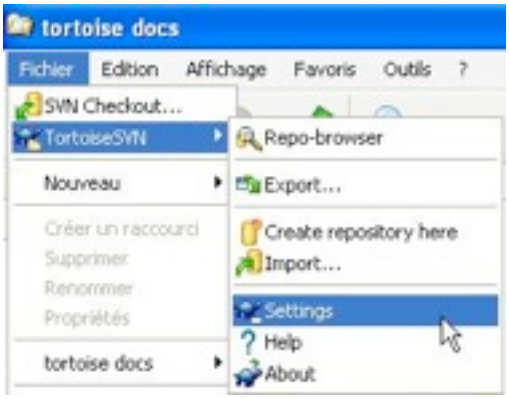
➡ Le dossier tags contient des «pointeurs» sur des copies à un instant donné. Un tag représente une espèce d'étiquette sur une version particulière de votre développement. Par exemple, la révision X fonctionne parfaitement, et si ce n'est pas la version finale, c'est en tout cas une très bonne version bêta. On peut alors la tagger, ce qui permettra de faire une première release ou des démos, sans empêcher l'avancement du projet. De même, on peut tagger ainsi la version 1.0, 1.1, 1.2 ... de notre logiciel. En pratique, le tag fonctionne exactement comme une branche. Subversion ne fait pas de différences pour des raisons de flexibilité. En effet, imaginez que vous taggez une version qui fonctionne parfaitement... enfin presque... après avoir taggé, vous découvrez un horrible bug ! Vous pouvez faire un commit sur un tag (c'est rare, mais ça se fait).

3. Installation de Tortoise SVN

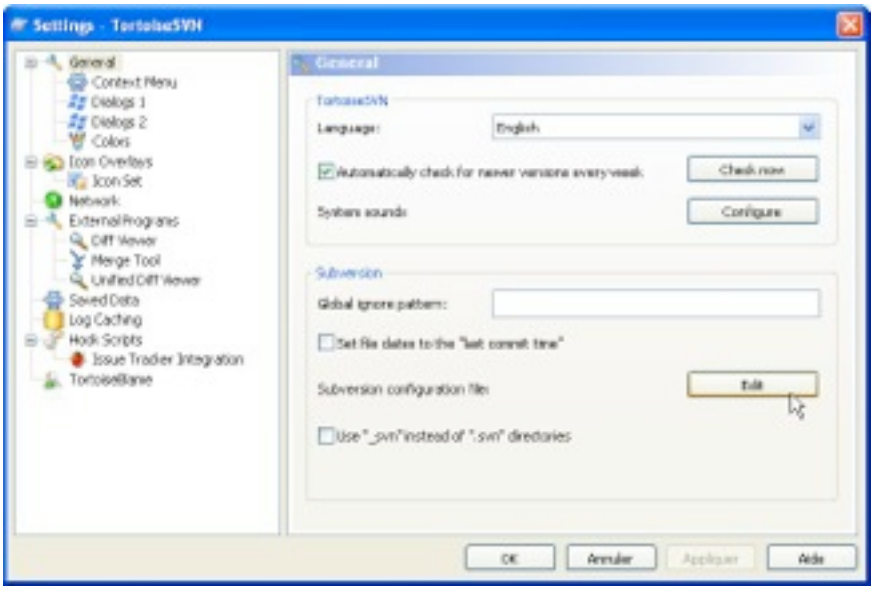
Vous trouverez la dernière release de TortoiseSVN (1.6.5) ici : http://tortoisesvn.net/downloads
Il y a différents packages de langue (français et autres...).

4. Config File

Une fois que vous avez installé Tortoise SVN 1.6.5, vous devrez modifier le fichier de configuration. C'est lui qui nous donnera la possibilité de signer automatiquement chaque fichier avec le nom de l'auteur, la date de modification, le numéro de révision ou d'autres informations.



Pour changer le fichier de configuration :
Dans l'explorer Windows, cliquez sur Fichier, puis TortoiseSVN puis Settings
Dans la nouvelle fenêtre qui s'ouvre, sur la gauche, cliquez sur General (par défaut) et sur la droite sur le bouton Edit.



Votre fichier de configuration est maintenant ouvert.
Dans ce fichier, copiez et collez le contenu de mon fichier de configuration ou ajoutez les lignes ci-dessous dans les sections adéquates :
trouvez la section `[miscellany]` et ajoutez (ou simplement décommentez) la ligne suivante :

```
enable-auto-props = yes
```

Ensuite, ajoutez les lignes suivantes dans la section `[auto-props]` placée généralement à la fin du fichier de configuration

```
*.c = svn:eol-style=native;svn:keywords= Id Rev Date Author
```

```
*.cpp = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.h = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.cs = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.sln = svn:eol-style=CRLF
*.vcproj = svn:eol-style=CRLF
*.csproj = svn:eol-style=CRLF
*.dsp = svn:eol-style=CRLF
*.dsw = svn:eol-style=CRLF
*.sh = svn:eol-style=native;svn:executable
*.txt = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.png = svn:mime-type=image/png
*.jpg = svn:mime-type=image/jpeg
Makefile = svn:eol-style=native

*.am = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.ac = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.xml = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.xsd = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.html = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.wsdl = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.composite = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.componentType = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.rb = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.py = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.php = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.js = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.java = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.properties = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.jelly = svn:eol-style=native;svn:keywords= Id Rev Date Author
*.ipr = svn:eol-style=native
*.iml = svn:eol-style=native
*.project = svn:eol-style=native
*.classpath = svn:eol-style=native
README = svn:eol-style=native;svn:keywords= Id Rev Date Author
LICENSE = svn:eol-style=native
NOTICE = svn:eol-style=native
```

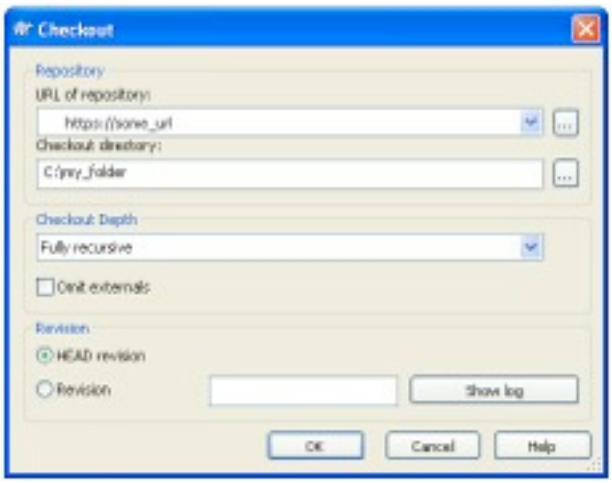
Ces modifications ne seront effectves que sur les fichiers qui seront ajoutées au repository après ces modifications. Ceci doit donc être fait en priorité lors de votre première utilisation de TortoiseSVN.

5. Utilisation quotidienne de TortoiseSVN

Dans cette section, les propriétés les plus utilisées de TortoiseSVN sont présentées. Elles devraient suffire pour une utilisation quotidienne. Si vous voulez en savoir plus, rendez vous sur le site http://tortoisesvn.net/support et téléchargez la documentation complète (assez facile à lire).

5.1. Checkout

C'est la première commande que vous utiliserez. Elle vous permet de récupérer une copie de la dernière révision du projet sur votre ordinateur. Vous pouvez avoir autant de copies que vous le voulez. Cependant, il est conseillé d'en avoir le moins possible pour éviter les confusions.
Une utilisation très pratique consiste à avoir une copie sur votre ordinateur de travail (à l'IUT par exemple la journée) et une autre sur votre ordinateur personnel (pour la nuit et le weekend ;).
Pour l'utiliser, créez simplement un nouveau dossier dans lequel vous laisserez votre copie (par exemple ici, my_folder).
Faites un clic droit sur ce dossier et cliquez sur `SVN Checkout...`



Vous voyez maintenant une nouvelle fenêtre.
Entrez l'URL du repository (généralement du type https://some_url ou svn://some_url pour des repositorys sur un serveur, ou file:///some_url for a local repository pour des repository en local).
Le checkout directory devrait déjà être le dossier que vous avez créé auparavant.
Cliquez sur `OK`
Par défaut, c'est la révision `HEAD Revision` qui est sélectionné. Cela signifie que vous allez télécharger une copie de la dernière révision du projet.
Si vous cliquez sur Revision, vous pourrez choisir un autre numéro de révision et télécharger une copie d'un état particulier du projet.
Généralement, vous téléchargerez la `HEAD Revision`, au moins en mode développement.

Vous aurez peut-être alors une fenêtre d'avertissement de la part de TortoiseSVN car il vous faudra un certificat pour accéder au repository. Si vous en avez un (donné par le propriétaire du repository) vous pouvez cliquer sur Accept permanently ou Accept Once. Si vous n'avez pas de login et de mot de passe, vous pouvez toujours essayer de demander au propriétaire du repository ;)

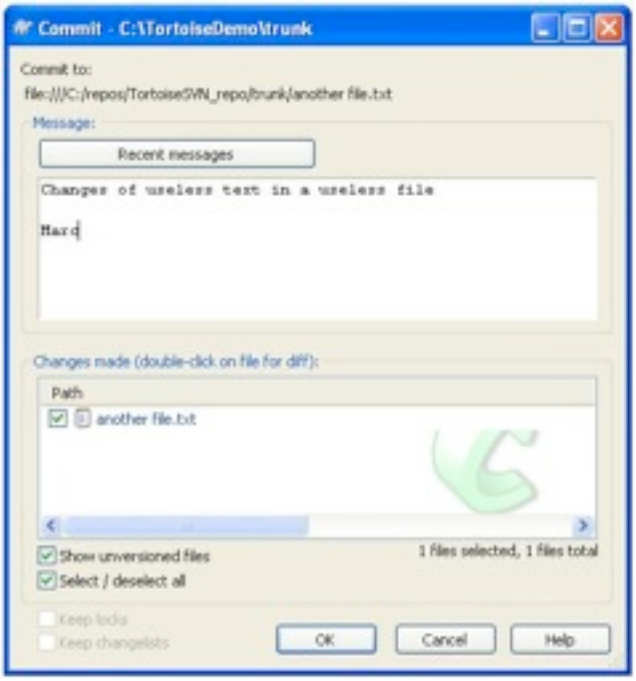
Maintenant vous avez un résumé du check out et vous pouvez voir une icône au-dessus de votre dossier, indiquant que vous n'avez pas modifié votre copie depuis le dernier check out ou la dernière mise à jour.

5.2. Commit

Pendant le développement, vous devriez changer quelques fichiers (bon... ne devriez vous pas ?). Dans ce cas, l'icône au-dessus du fichier que vous modifiez dans l'explorer windows, passe d'un joli vert à un horrible rouge et vous fait penser que vous avez fait une horrible chose.

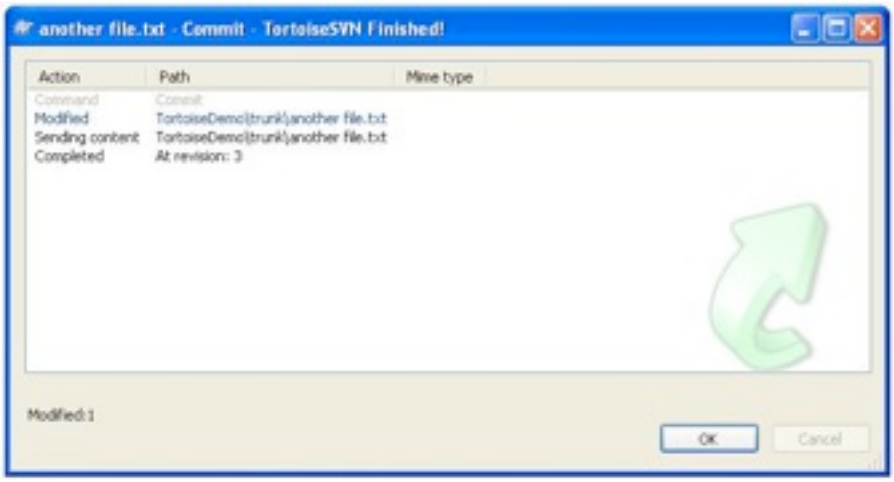
En réalité, cela veut seulement dire que ce fichier a été modifié depuis le dernier check out ou le dernier update. Cela signifie qu'il est très probablement différent du fichier qui est dans le repository. Cela signifie également que vos collègues ne savent rien de vos modifications. Quand vous avez fini vos modifications (vous devez d'abord vérifier que votre code compile correctement), vous pourriez avoir envie de le partager avec vos collègues. Vous devez pour cela faire un commit.

Cliquez simplement sur le fichier (ou le dossier et dans ce cas, tous les fichiers et dossiers enfants seront commités également) qui vous intéresse et cliquez sur `SVN Commit...`



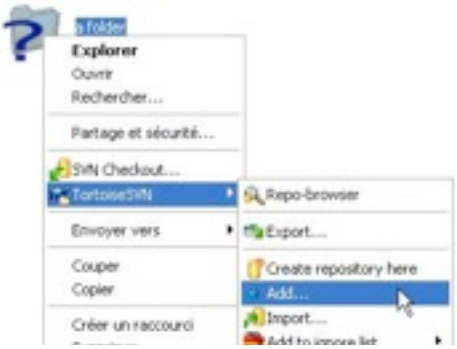
Une nouvelle fenêtre apparaît dans laquelle vous pouvez sélectionner ou désélectionner les fichiers que vous voulez envoyer sur le repository. Il y a quelques options qui vous aideront à déselectionner les fichiers inutiles et sélectionner rapidement les fichiers utiles.
Vous devriez également toujours rentrer un message qui explique à vos collègues le sens de vos modifications (cf. règles).

Cliquez sur OK. Vous devriez maintenant voir le résumé de votre commit. Votre fichier doit être revenu à son joli vert (sauf en cas de conflits, voir plus bas).



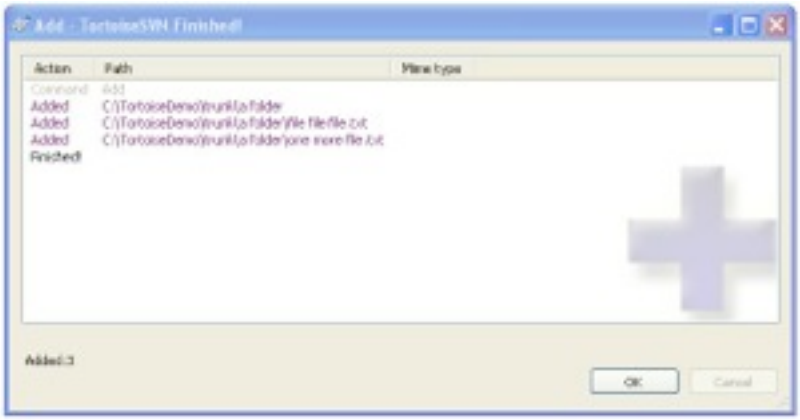
5.3. Add

Si vous voulez ajouter un nouveau fichier ou un nouveau dossier, créez simplement comme d'habitude vos documents dans l'explorer de windows. Si vous les créez dans un dossier versionné, vous verrez un gros point d'interrogation bleu sur votre document. Tortoise vous avertit que ce document n'est pas versionné dans le repository. «Peut-être souhaitez-vous ajoutez ces fichiers au repository ?» «Oui je le veux !».



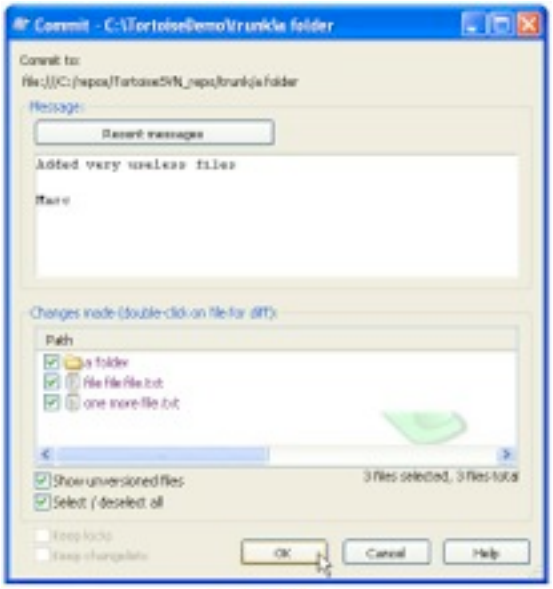
Pour dire à Tortoise que vous voulez les ajouter, faites un clic droit sur le fichier ou le dossier et sélectionnez `TortoiseSVN -> Add...`

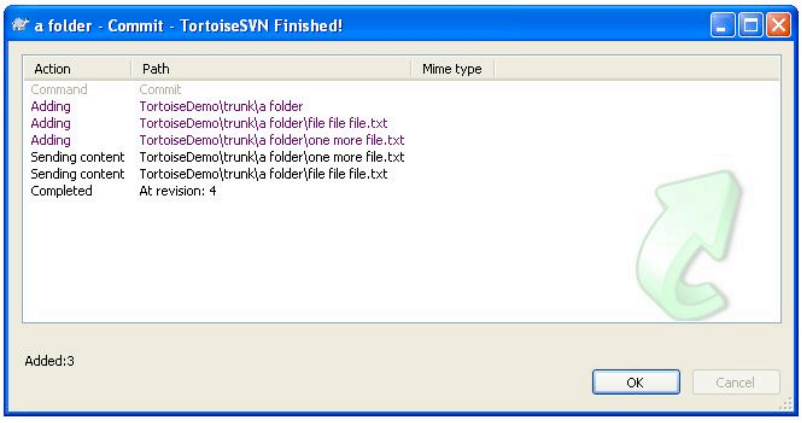
Une nouvelle fenêtre apparaît avec les fichiers que vous vouliez ajouter.
Sélectionnez ceux qui vous intéressent, déselectionnez ceux qui ne vous intéressent pas et cliquez sur `OK`.
Tortoise vous confirme que les fichiers sont ajoutés, mais les fichiers ne sont pas encore dans le repository. Vous avez juste indiqué que vous voulez les ajouter au prochain commit.



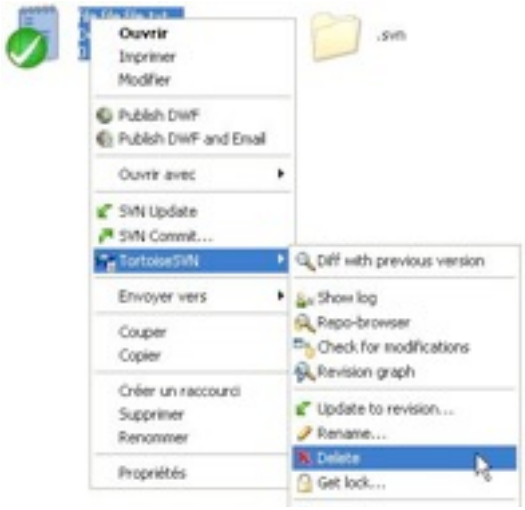
Vous pouvez d'ailleurs noter que l'icône est un gros + bleu et pas encore la belle icône verte.

Faites simplement un commit sur ces fichiers, et ils seront définitivement ajoutés au repository. N'oubliez pas d'ajouter le message !





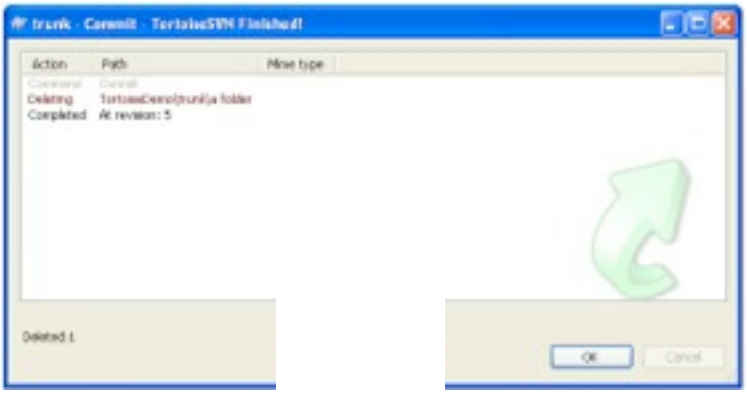
5.4. Delete



Le delete fonctionne un peu comme le Add. D'abord vous indiquer à Tortoise que vous voulez supprimer un fichier ou un dossier ... au prochain commit.

Si c'est un fichier, il va directement disparaître de l'explorer windows. Si c'est un dossier, il sera marqué d'une croix rouge vif. Dans tous les cas, ils sont supprimés de votre copie mais pas encore du repository.

Faites un commit quelque part dans un dossier parent (n'oubliez pas le message...) et le tour est joué.

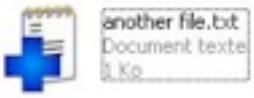


5.5. Move

Pour changer l'emplacement d'un fichier ou d'un dossier, sélectionnez le fichier ou le dossier, et glissez le avec le bouton droit de la souris enfoncé dans le nouvel emplacement.

Un popup menu apparaît alors, dans lequel vous pouvez sélectionner «SVN Move versioned files here».
Le fichier est alors supprimé et ajouté par Tortoise.

C'est pour cette raison que le dossier qui contient les fichiers a une icône rouge and le fichier qui a bougé a un + bleu. Faites un commit sur le dossier parent (sans oublier le message) et le fichier aura changé de place dans le repository.



5.6. Rename

Pour renommer un fichier, faites un clic droit sur le fichier concerné et sélectionnez `TortoiseSVN -> Rename...`

Dans la nouvelle fenêtre, changer le nom et cliquez sur `OK`.

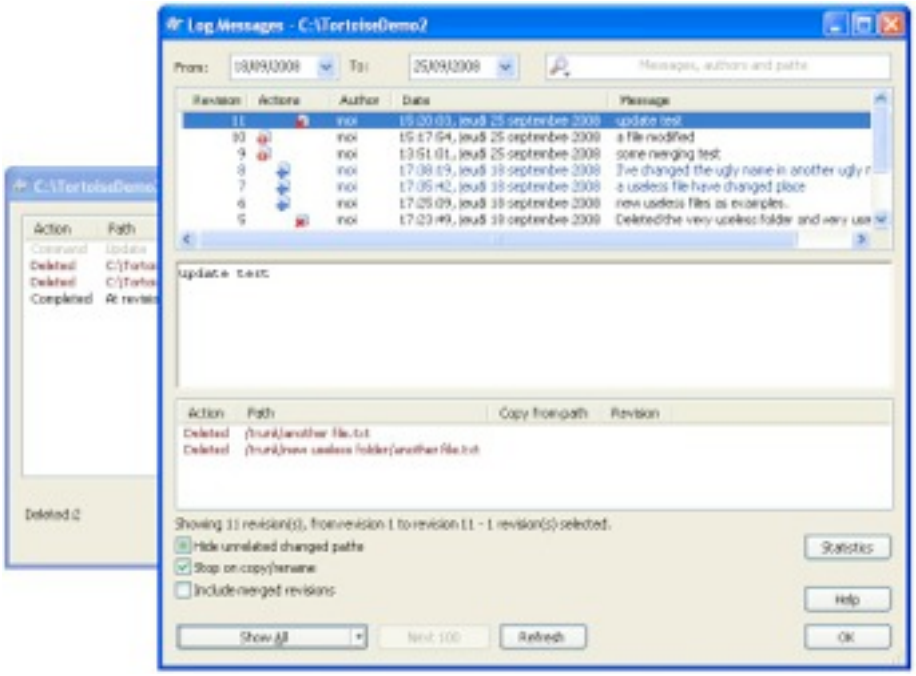
Le fichier est supprimé puis ajouté avec le nouveau nom. C'est pour cela qu'il a un gros + bleu sur lui. N'oubliez donc pas de faire le commit.

5.7. Update

Une à deux fois par jour, vous devriez faire un update de votre copie. Cela signifie que vous devriez télécharger les modifications faites par vos collègues sur le projet. Pour cela, vous devez faire un clic droit sur le dossier que vous voulez mettre à jour (généralement la racine, mais vous n'avez pas nécessairement besoin de tout mettre à jour si le repository contient par exemple plusieurs projets) et sélectionner `SVN Update`.
Notez que même si les fichiers de votre copie ont une icône verte, cela ne signifie pas qu'ils sont à jour : cela garantit uniquement que vous ne les avez jamais modifiés après le dernier update ou le check out. C'est la raison pour laquelle il faut continuer à faire l'update régulièrement.

Une fenêtre apparaît indiquant quel est le numéro de révision de votre update et quels sont les fichiers modifiés, ajoutés ou supprimés par cet update.

Vous pouvez également lire les messages des commits qui ont lieu jusqu'à cette révision en cliquant sur `Show Log...` Cela vous permet de voir qui a fait le commit, ce qui a été changé et le message laissé par celui qui l'a fait.



5.8. Conflicts

Prenons l'exemple du fichier "a file.txt", modifié par 2 personnes en même temps, à deux endroits différents. Le premier a changé le contenu de

```
"
a file that contains some text
"
```

à

```
"
a file that contains some text
blablabla
plouf plouf plouf
argh argh argh
"
```

et a fait un commit.
Vous ne le savez pas encore, parce que vous n'avez pas mis à jour votre copie depuis 3 heures et vous modifiez également le même fichier, mais dans votre copie, de :

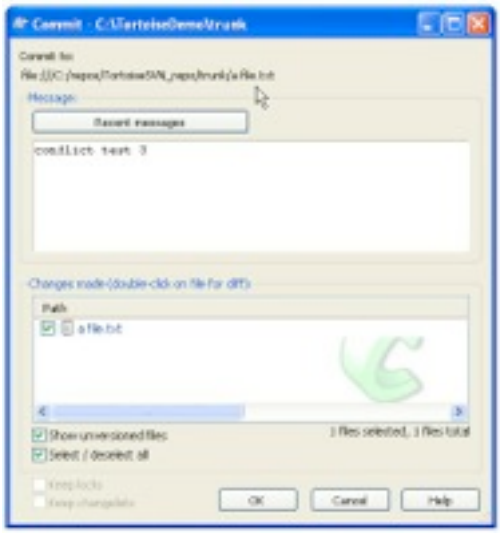
```
"
a file that contains some text
"
```

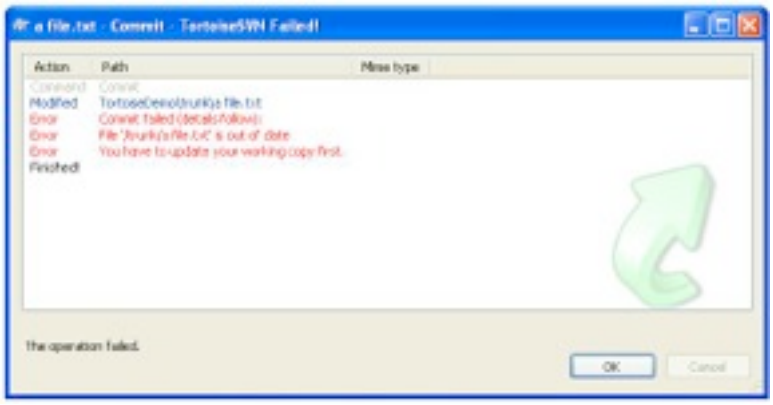
à

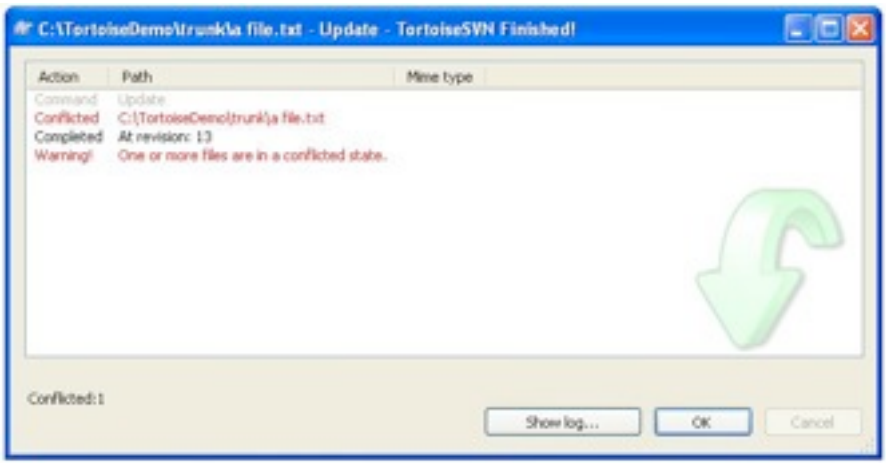
```
"
a file that contains some text
here is some text
"
```

Maintenant vous voulez également faire un commit avec vos modifications... que se passe-t-il ?

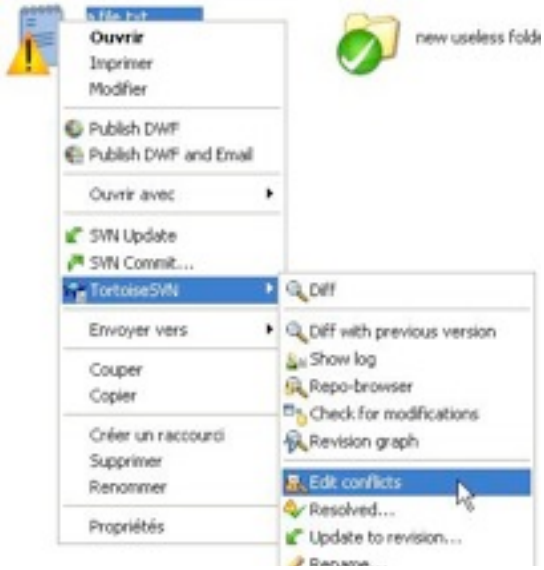
Tout d'abord, vous faites le commit comme d'habitude et une erreur s'affiche indiquant que votre copie n'est pas à jour, et que vous devriez faire un update avant de faire un commit. Votre commit est annulé.





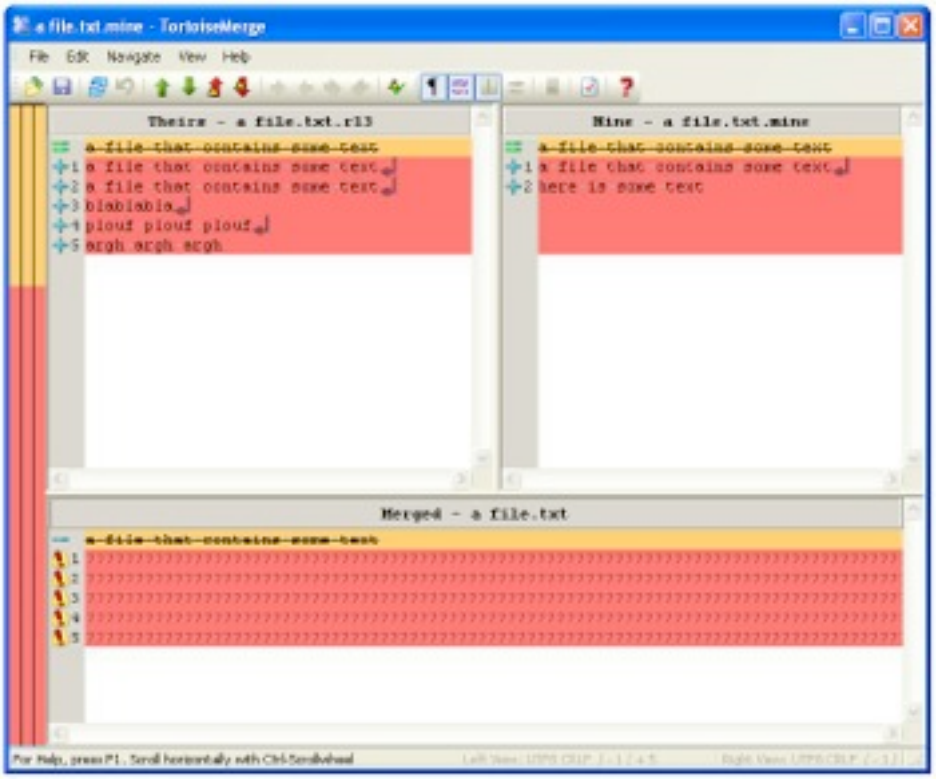


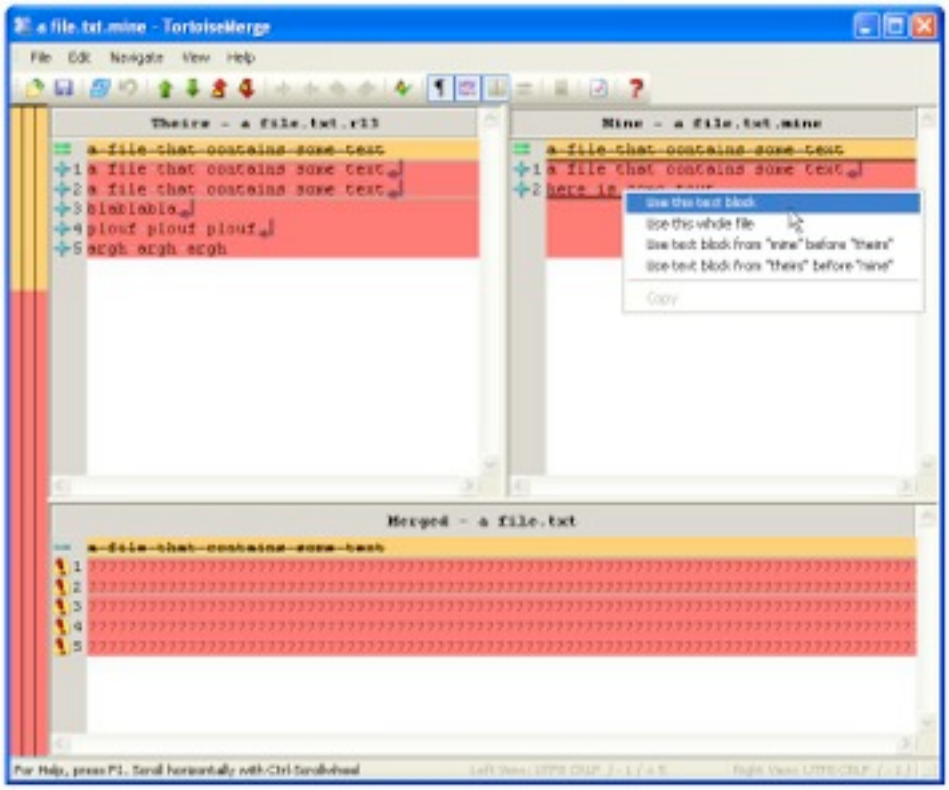
Ce n'est pas grave, vous cliquez sur OK, et vous faites un update. Le fichier en question est indiqué comme «conflicted». Donnez un coup d'oeil aux fichiers créés et aux icônes... l'un d'entre eux est "a file.txt.mine" (il s'agit de votre copie modifée par vous) "a file.txt.r12" (la copie de la révision numéro 12, c'est-à-dire le dernier update que vous aviez fait) "a file.txt.r13" (la copie de la révision numéro 13, la dernière révision de ce fichier sur le repository que vous n'aviez pas encore téléchargée). Ces fichiers vont être utilisés pour éditer les conflits. Oubliez-les.



Faites simplement un clic droit sur "a file.txt" et sélectionnez `Tortoise SVN --> Edit Conflicts...`

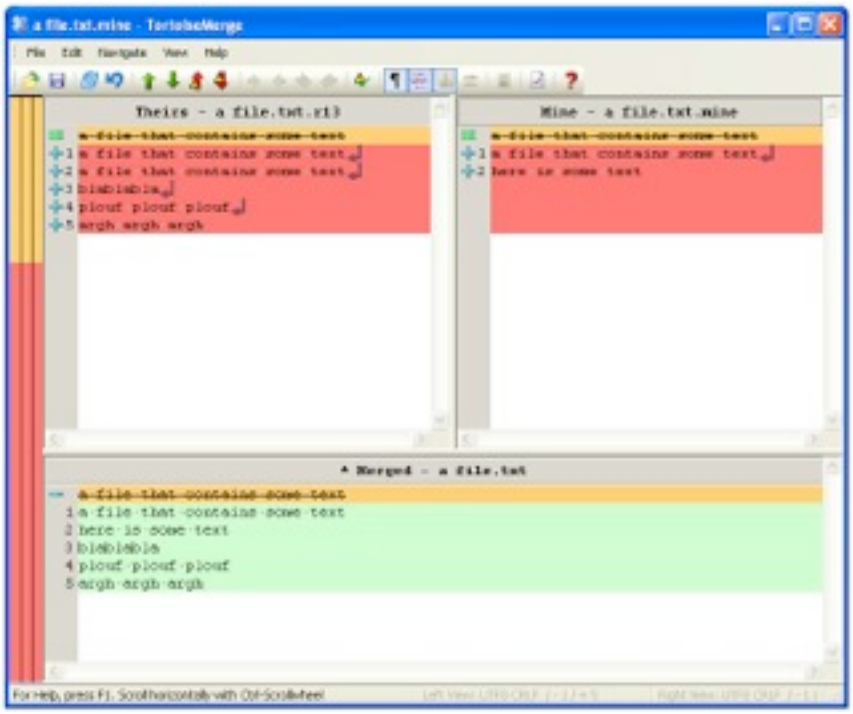
Une nouvelle fenêtre apparaît, et montre sur la gauche le fichier qui est sur le serveur (sur le repository) et sur la droite votre copie modifiée par vos soins. En couleurs sont indiquées les différences et vous pouvez les éditer. Plus c'est rouge, plus c'est grave. Si c'est rouge, vos modifications manuelles sont obligatoires (le reste peut être fait automatiquement). En bas, vous avez le fichier résultat après fusion.

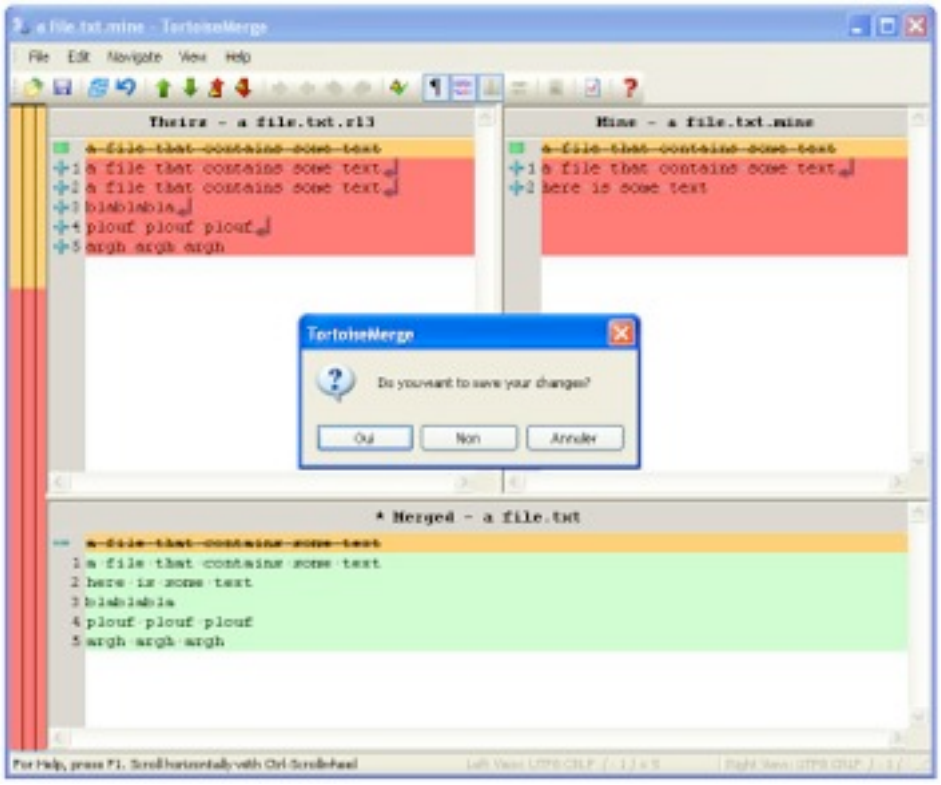




Cliquez sur les parties sur une des deux copies pour indiquer quels blocs il faut garder entre la version du repository et votre copie modifiée. Pour cela, cliquez simplement droit sur une partie du document et sélectionnez «Use this text block».

Vous pouvez voir les résultats dans le fichier fusionné. En rouge sont indiquées les parties à modifier.

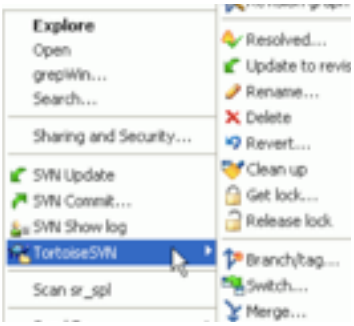




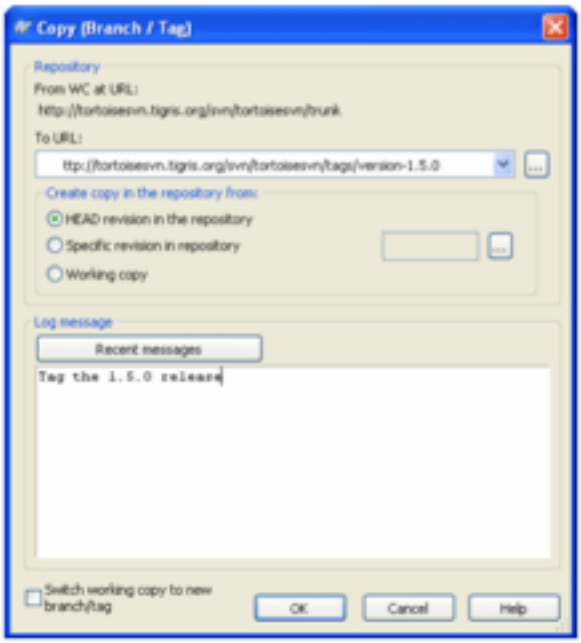
Sauvegardez vos modifications quand vous avez fini, et faites votre commit.

6. Utilisation avancée

6.1. Brancher / Étiqueter (branching/ tagging)



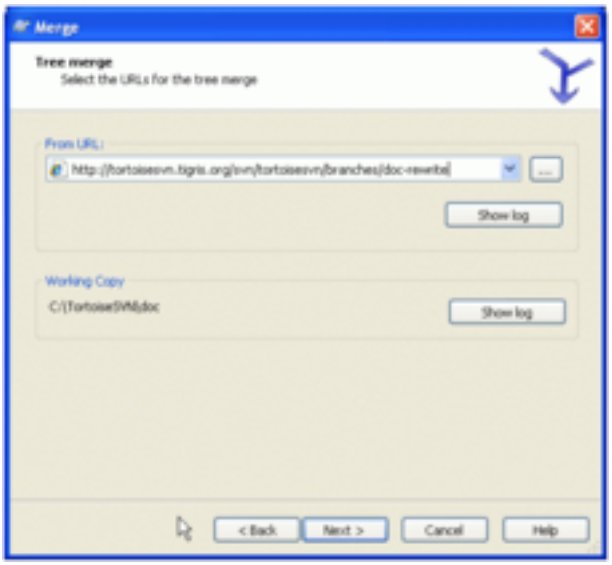
Créer une branche ou une étiquette se fait exactement de la même manière avec TortoiseSVN. Faites un clic droit sur le dossier du tronc (trunk) puis cliquez sur TortoiseSVN puis Branch/ tag.



Dans le fenêtre qui s'ouvre, la ligne «From WC at URL» représente l'adresse qui va être branchée ou étiquetée (généralement le tronc, ou une autre branche). La ligne «To URL» représente le nom de votre nouvelle branche ou étiquette. Si vous voulez créer une branche, donnez lui un nom et placez-la dans le dossier branches. Pour une étiquette, donnez lui un nom et placez-la dans le dossier tags.

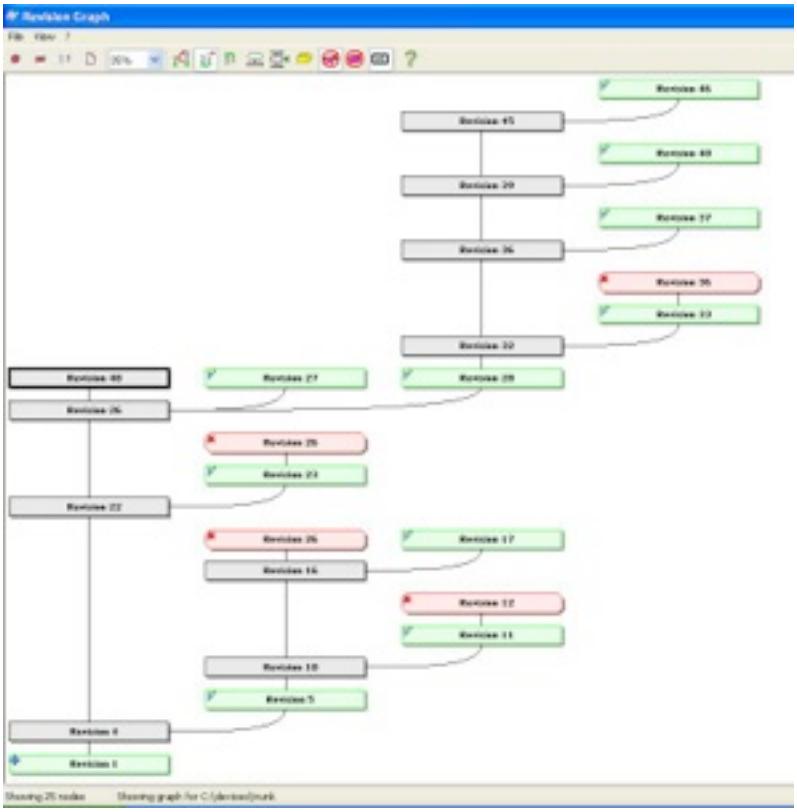
6.2. Fusionner (Merge)

Cette fonctionnalité vous permet de réintégrer une branche dans votre tronc une fois que vous l'avez validée. Il existe d'autres modes de fusion, mais pour une utilisation simple, c'est la seule que nous présenterons dans ce document.



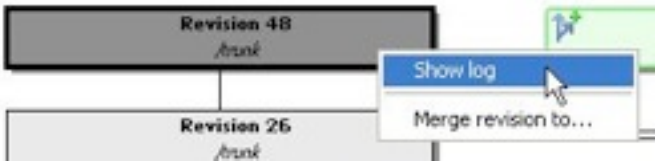
6.3. Revision Graph

TortoiseSVN propose un outil graphique de visualisation du graphe de révisions pour vous aider à retrouver votre chemin dans l'arbre des révisions. Faites un clic droit sur votre copie (trunk) et sélectionnez `TortoiseSVN > Revision Graph`.



Une nouvelle fenêtre s'ouvre et vous pouvez y voir un graphe de révisions. Si vous avez peu de révisions, peut-être que vous n'y verrez rien, parce que votre arbre ne sera qu'un tronc. Si votre arbre possède des branches, cet outil vous aidera à retrouver un tag ou une branch avec le numéro de révision recherché.

En cliquant avec le bouton droit de la souris sur une révision, vous pouvez sélectionner `Show Log` et lire le message laissé par l'auteur de ce commit.



7. Quelques règles de bonne conduite

Ce chapitre contient un ensemble de règles et de conventions à respecter pour une utilisation en groupe de Subversion.

- Le projet stocké dans la base (et encore plus, celui du tronc), doit toujours être compilable.
  - Avant de faire un commit, vérifiez toujours que votre projet compile sur votre copie.
  - Si vous n'êtes pas sûr de vous, faites un checkout dans un dossier temporaire et vérifiez que votre commit compile.
  - Si vous devez faire exceptionnellement un commit qui ne compile pas, prévenez tous les autres utilisateurs.

- Faites des updates très régulièrement.

- Faites des commits réguliers.
  - Si vous gardez trop longtemps des fichiers extraits modifiés, vous augmentez le risque de conflits qui seront de plus en plus compliqués à résoudre.
  - Sans commits réguliers, vous perdez également le bénéfice de l'historique.

- Utilisez le header suivant au début de chaque fichier afin de garder une trace de la dernière date de modification, de la dernière personne ayant modifié le fichier et du numéro de révision

```
// ============================================================================
//
// Copyright (C) 2008-2009 IUT CLERMONT1 - UNIVERSITE D'AUVERGNE
//                         www.iut.u-clermont1.fr
//
// Module       : Board - source file
// Author       : Marc Chevaldonné
// Creation date: 2008-11-25
// Modified by  : $Author: chevaldonne $
// Last update  : $Date: 2008-11-25 01:30:30 +0200 (tue, 25 nov 2008) $
// Revision     : $Revision: 18 $
//
// @author Marc Chevaldonné
//
// ============================================================================
```