# Gestion de versions : Subversion

Emmanuel Grolleau

Observatoire de Paris – LESIA – Service d'Informatique Scientifique

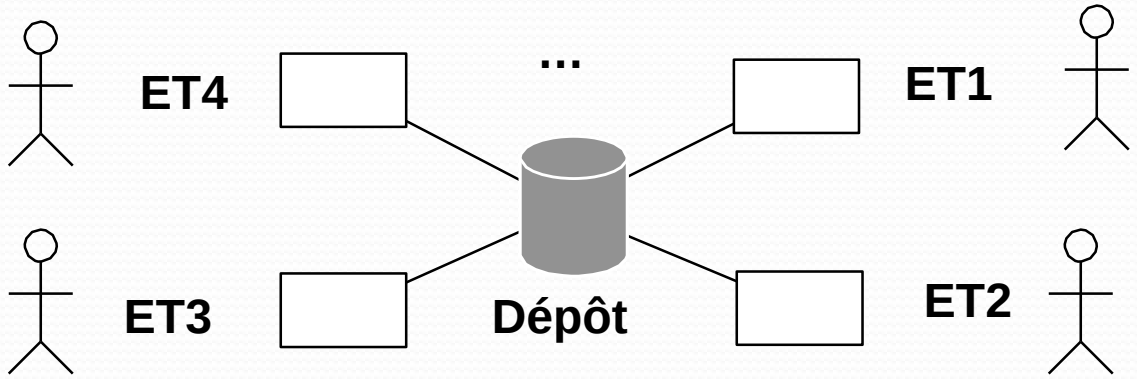
**Master 2 « Outils et Systèmes de l'Astronomie et de l'Espace »**

- **Pourquoi un outil de gestion de version ?**

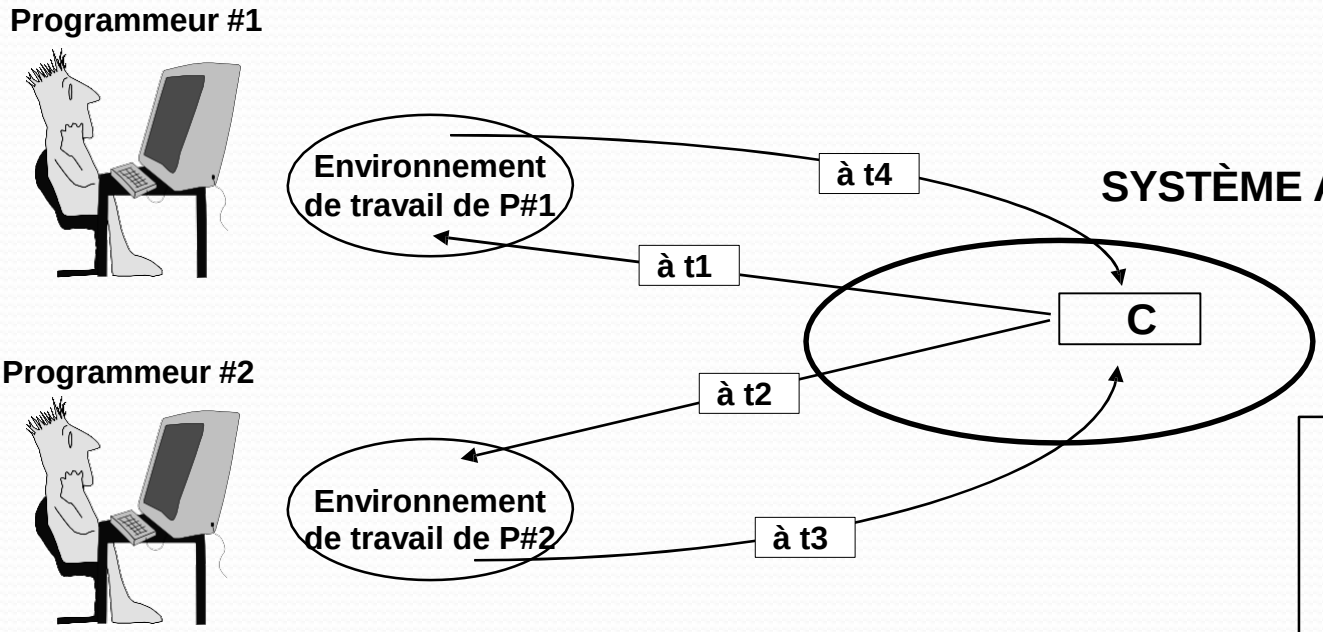
# Solution : utiliser un outil de gestion de version

- Permettre à plusieurs personnes de travailler en parallèle
  - Partage intègre de fichiers
    - Gérer les accès
    - Indiquer les conflits
    - Notifier les modifications
- Gérer les versions des composants
- Garantir traçabilité source à exécutable
- Revenir à une version antérieure

# Permettre à plusieurs personnes de travailler en parallèle



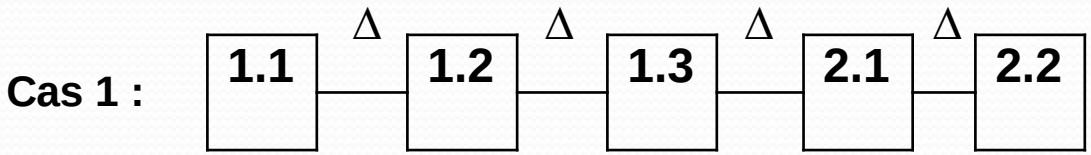
# Problème des mises a jour simultanées

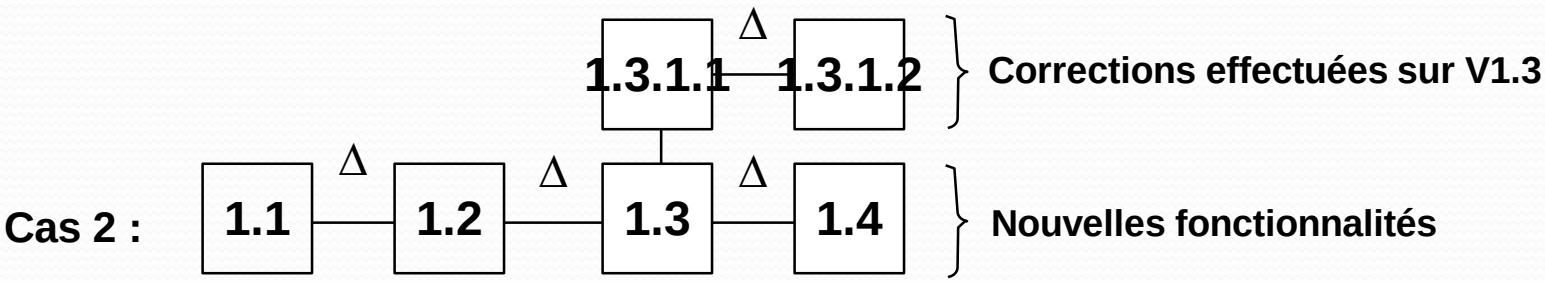


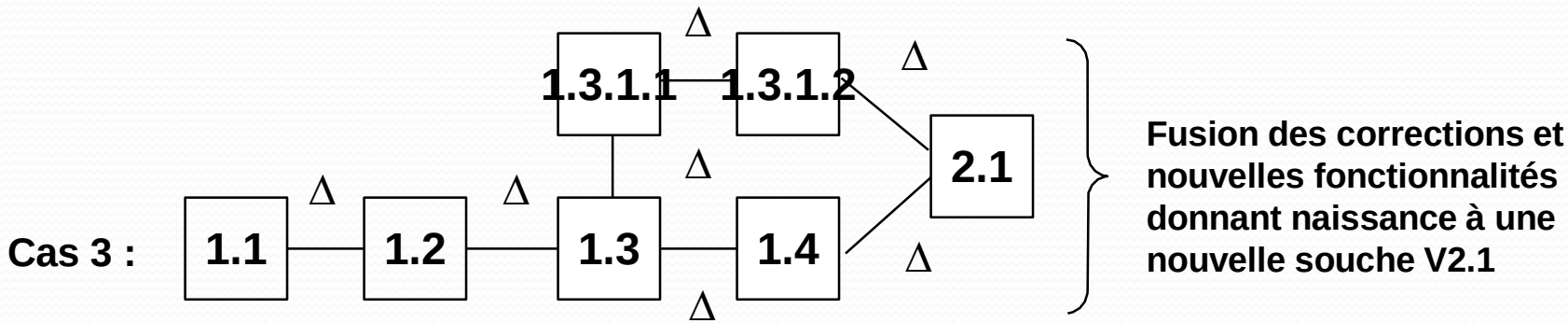

**SOLUTION**

**Un "secrétaire" doit garder trace des copies multiples et *synchroniser les mises à jour***

# Gérer les versions des composants



# Les principes

- Espaces de travail dédiés aux utilisateurs
  - Un espace de travail par utilisateur en local
  - Chaque utilisateur fait ses modifications en local
- Dépôt contenant l'ensemble des versions
  - Serveur
  - Garantissant l'intégrité
    - Ne pouvant être accédé que par les commandes de l'outil
  - Sécurité de gestion des accès
  - Contenant l'ensemble des versions successives

# Un outil de gestion de version : Subversion

Subversion est un logiciel de gestion de sources et contrôle de versions.

Il est principalement utilisé pour maintenir le code source ou la documentation.

Ses principales fonctionnalités sont :

- garder un historique des différentes versions des fichiers d'un projet
- permettre le retour à une version antérieure quelconque
- garder un historique des modifications
- permettre un accès à ces fichiers, en local ou via un réseau
- permettre à des utilisateurs distincts de travailler ensemble sur les mêmes fichiers

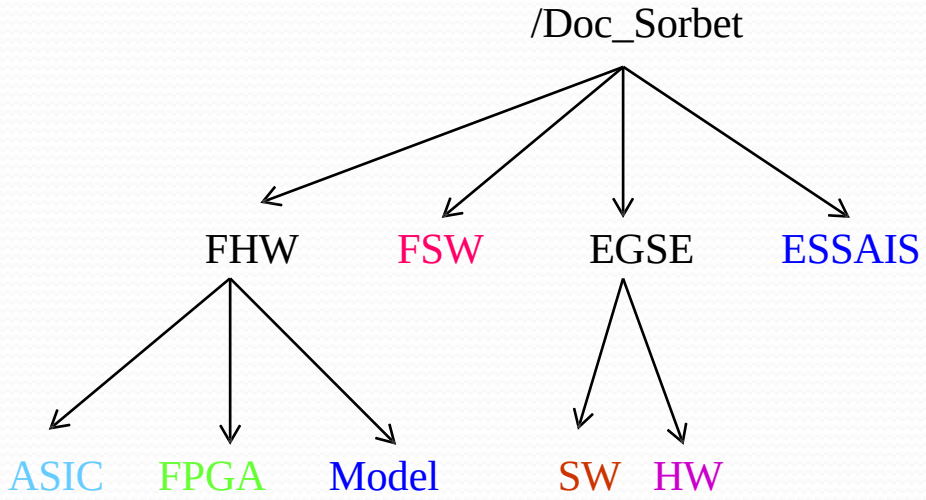
# Notions générales

- Dépôt (Repository)
  - Emplacement central où sont stockées les données relatives aux projets (Serveur)
  - On y accède via une URL distante  Ex : https://nomserveur/svn/repo1
- Projets
  - Répertoire situé à la racine du dépôt contenant les fichiers et dossiers du projet.
- Copie de travail
  - Répertoire situé en local sur le poste utilisateur et qui contient une copie d'un projet.
  - Modification d'un projet local avant d'être importé.
- Révision
  - Modification faite au dépôt
  - Indicateur s'incrémentant à chaque opération et permettant de revenir à une version donnée d'un ou plusieurs fichiers

# Exemple de dépôt (repository)

Le Dépôt de la documentation du projet SORBET-BepiColombo (Mercure).

/Doc_Sorbet (par défaut tous les acteurs du projet ont le droit en lecture)




Les accès en écriture :

Dekkali :

All :

Davy :

Astier :

Grolleau :

Boughedada :

# Opérations de base

- import :
  - Permet de placer dans le dépôt des fichiers locaux existant ;
  - Opération à ne réaliser qu'une fois par projet ;
  - Génération d'un nouveau projet sur le dépôt.

  *svn import --username groupeXX <chemin_projet> <url_repository> -m "import initial »*

- checkout :
  - Permet de récupérer les fichiers existants au sein d'un projet du dépôt ;
  - Opération à ne réaliser qu'une fois par projet ;
  - Génération d'une copie de travail

  *svn co --username groupeXX <url_repository>*

# Opérations de base

- update :
  - Permet de synchroniser la copie de travail avec le dépôt ;
  - Génération éventuel de conflits de version.

  *svn update*

- commit :
  - Permet la mise à jour du dépôt à partir de la copie de travail ;
  - Si le commit est accepté => une nouvelle révision ;
  - Sinon il faut effectuer un update et résoudre les conflits éventuels afin d'effectuer un nouveau commit.

  *svn commit –m « message de commit »*

# Scénario nominal

1) Import des fichiers existants dans le dépôt (initialisation du dépôt)
> svn import <*répertoire à importer*> <*url_repository_parent*> -m "*import initial*"
*Pour importer trunk dans* https://version-lesia.obspm.fr/repos/OSAE/Groupes/GroupeXX :
*cd trunk/..; svn import trunk* https://version-lesia.obspm.fr/repos/OSAE/Groupes/GroupeXX/ -m "*import initial*"
*Ou*
*cd <rep_parent>; svn import .* https://version-lesia.obspm.fr/repos/OSAE/Groupes/GroupeXX/ -m "*import initial*"

2) Check out du projet à partir du dépôt
> *svn co https://version-lesia.obspm.fr/repos/OSAE/Groupes/GroupeXX*

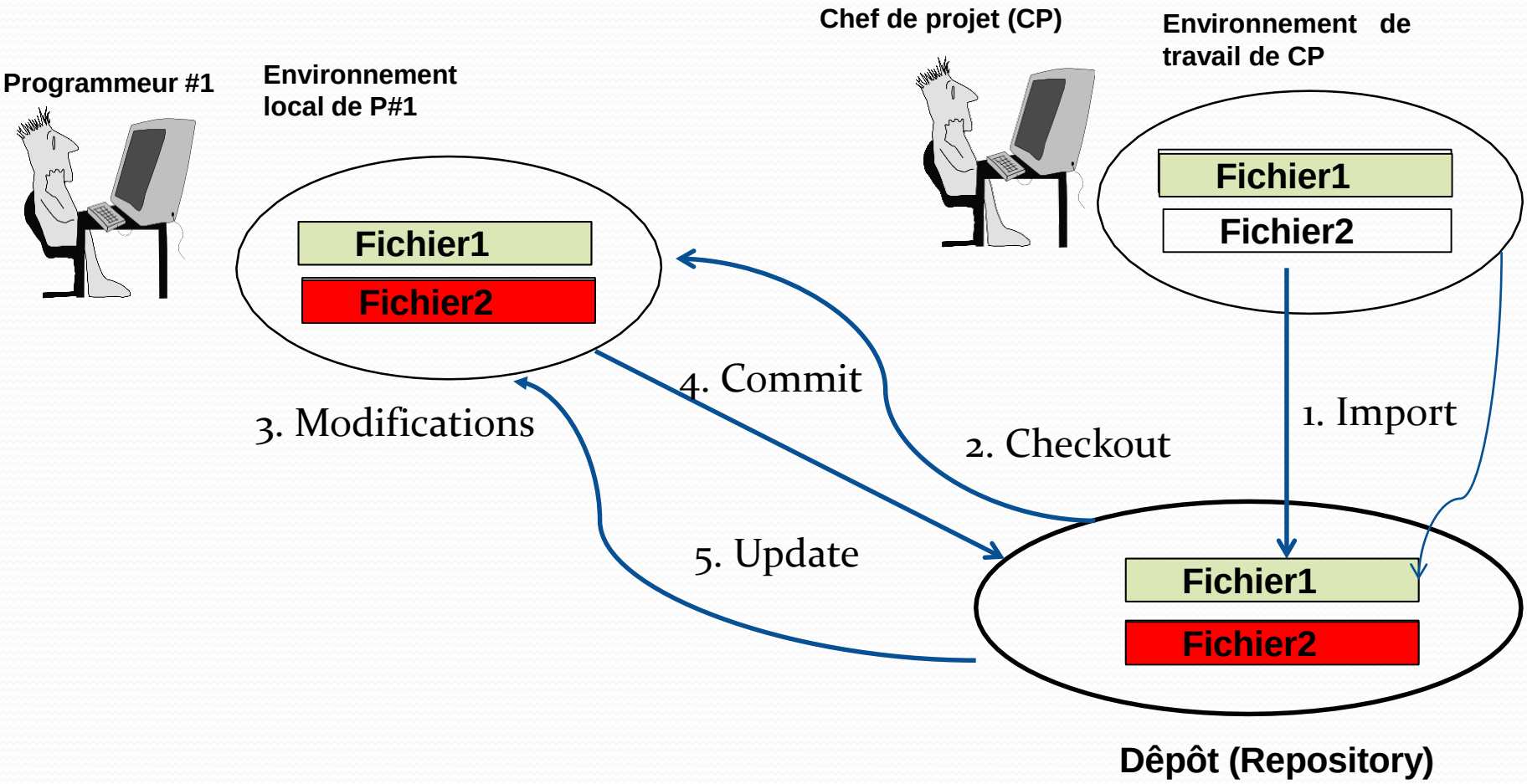
3) Dans le répertoire du projet, créer, supprimer ou renommer des fichiers ou des répertoires.
> *svn add file | svn delete file1 | svn move file1 file2*
Retourner à l'étape 2. Ou, si l'on est prêt à diffuser ses modifications, passer à l'étape 4

4) Commit de vos changements vers le dépôt. Retourner à l'étape 2.
> *svn commit -m " message sur le commit "*

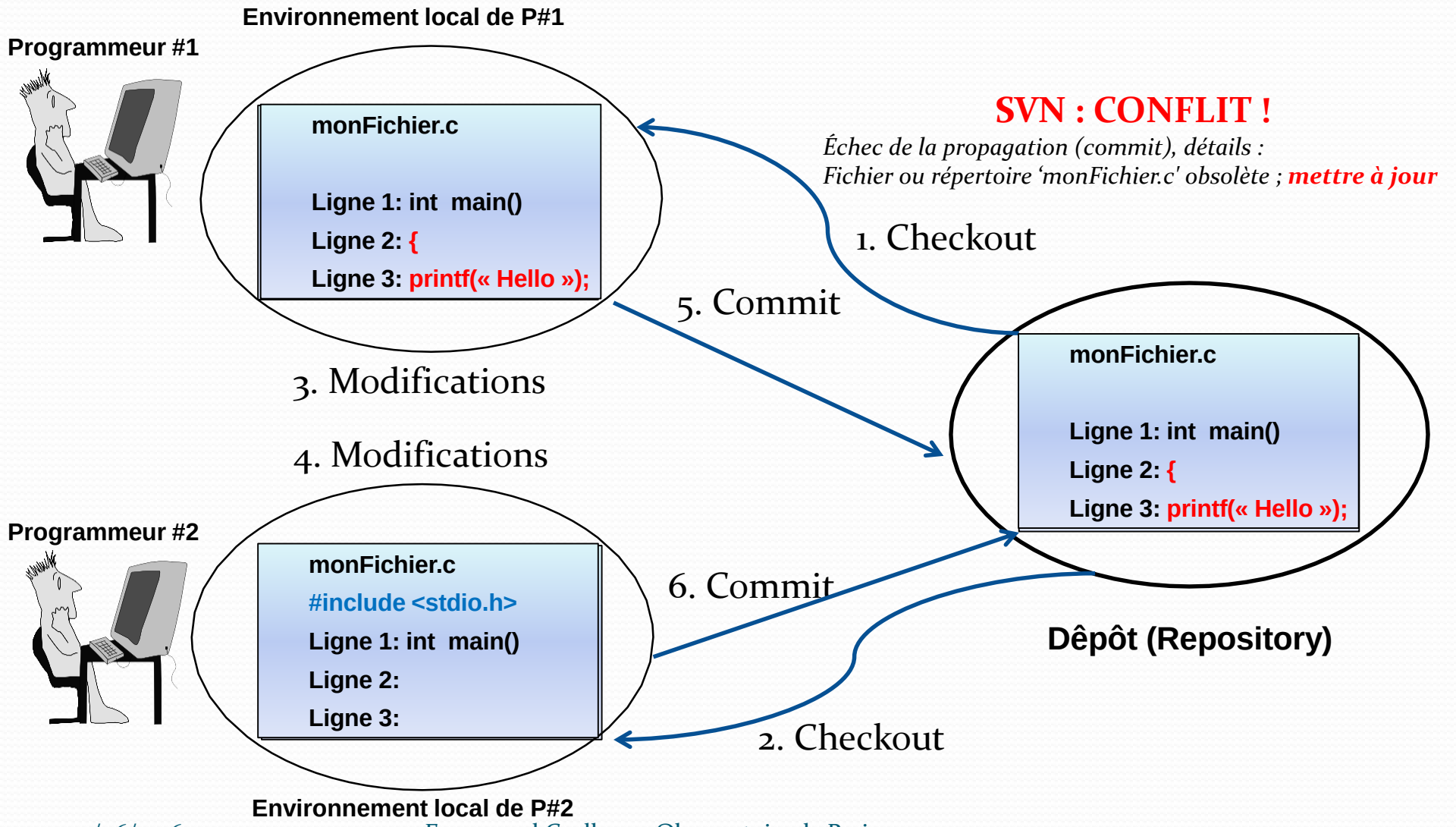
5) Mettre à jour la copie locale depuis le dépôt (récupérer les modifications des autres développeurs)
> *svn update*

Une commande indispensable : svn help <nom de la commande>

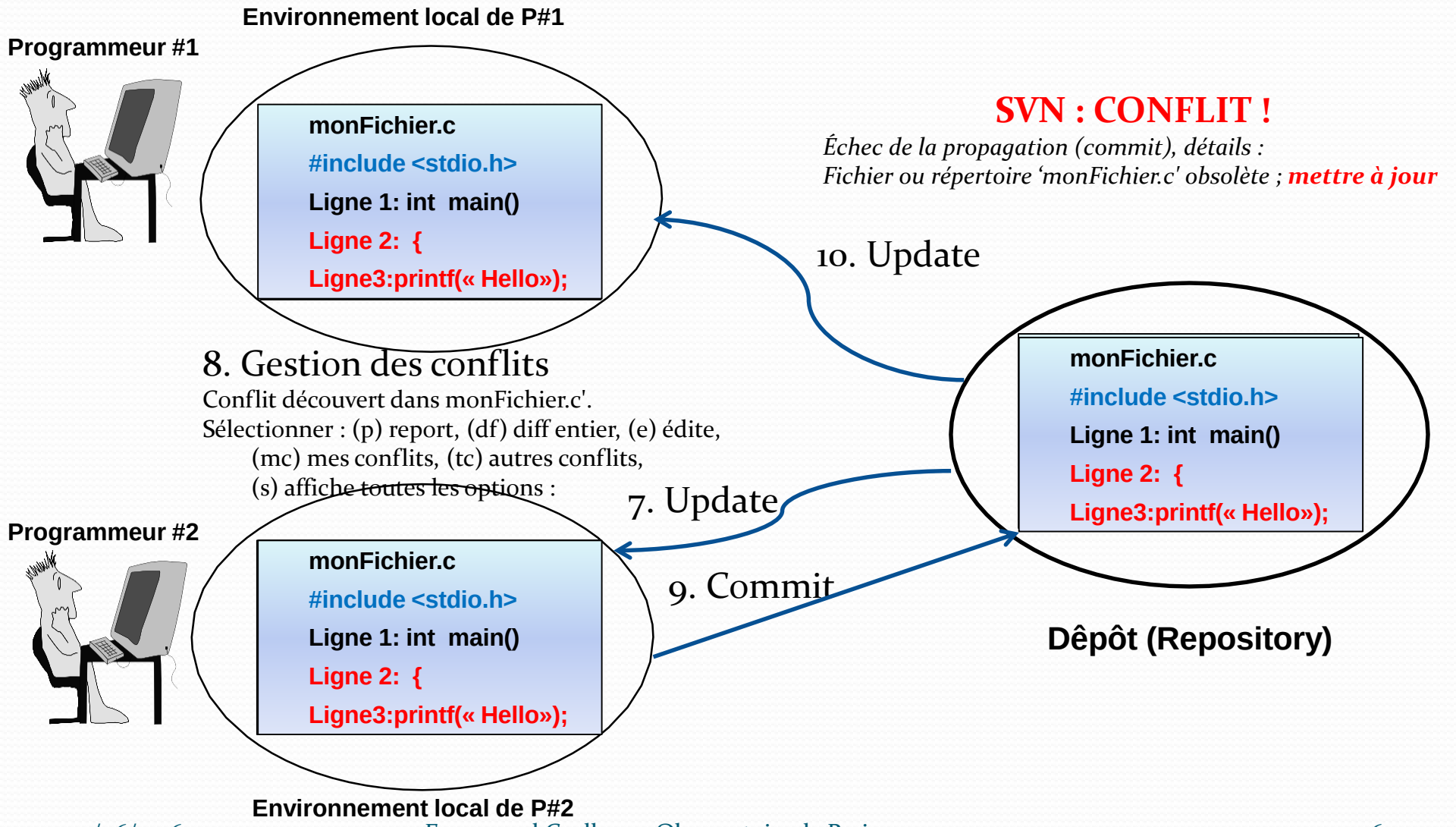
# Scénario nominal



# Les conflits



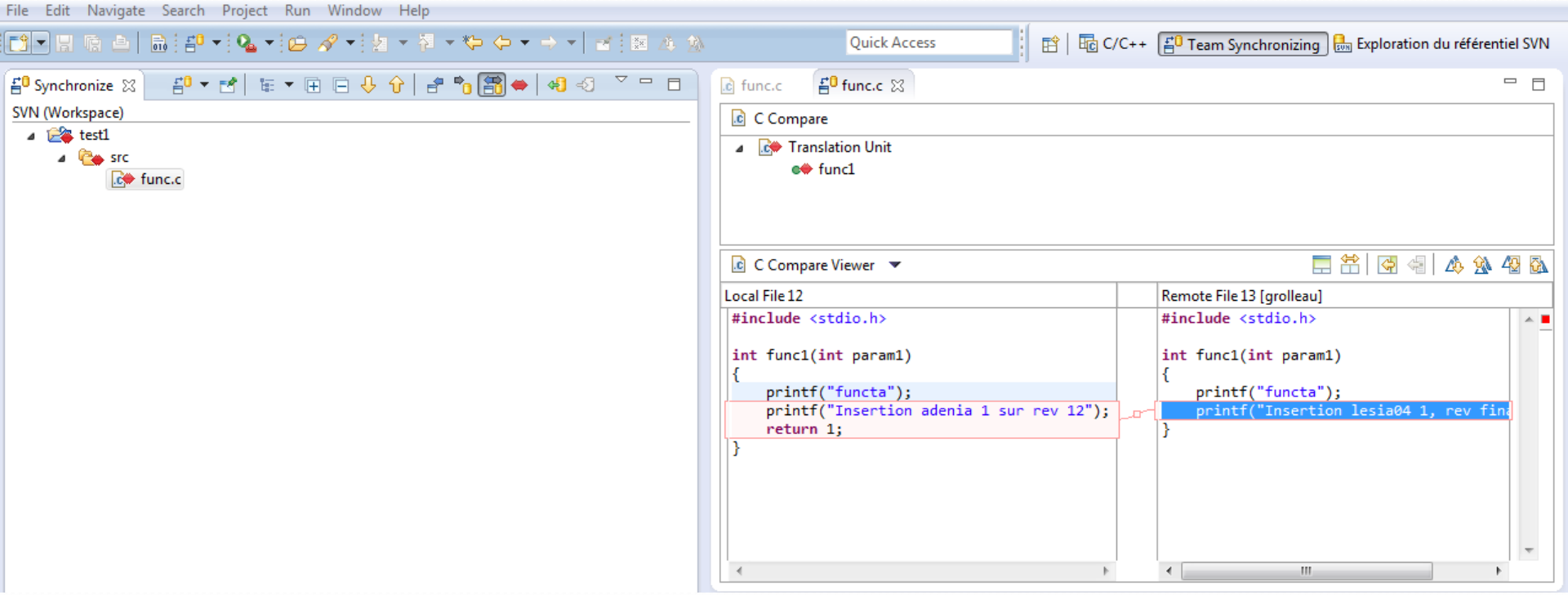
# Les conflits



# Conflits : Eclipse, perspective « Team synchronising »

- Dans la fenêtre de gauche est affichée un navigateur dans lequel ne sont affichés que les fichiers différents de la version du serveur SVN.
- Les fichiers peuvent avoir trois types d'icône :
  - Une icône avec une flèche grise orientée vers la droite : Cela signifie que le fichier a été modifié en local (par vous) depuis le dernier commit. Si vos modifications sont correctes, il convient de faire un « Commit » sur ce fichier.
    - Clic droit sur le fichier puis « Validez » dans le menu.
  - Une icône avec une flèche bleue orientée vers la gauche : Cela signifie que le fichier a été modifié par quelqu'un d'autre depuis votre dernière mise à jour. Autrement dit, la version du fichier sur le serveur SVN est plus récente que votre version. Il faut dans ce cas mettre à jour le fichier.
    - Clic droit sur le fichier puis cliquez sur « Mettre à jour » dans le menu.
- Une icône avec une double flèche rouge : Cela signifie que le fichier est en conflit avec la version du serveur SVN. Vous avez probablement modifié le fichier en même temps qu'une autre personne.

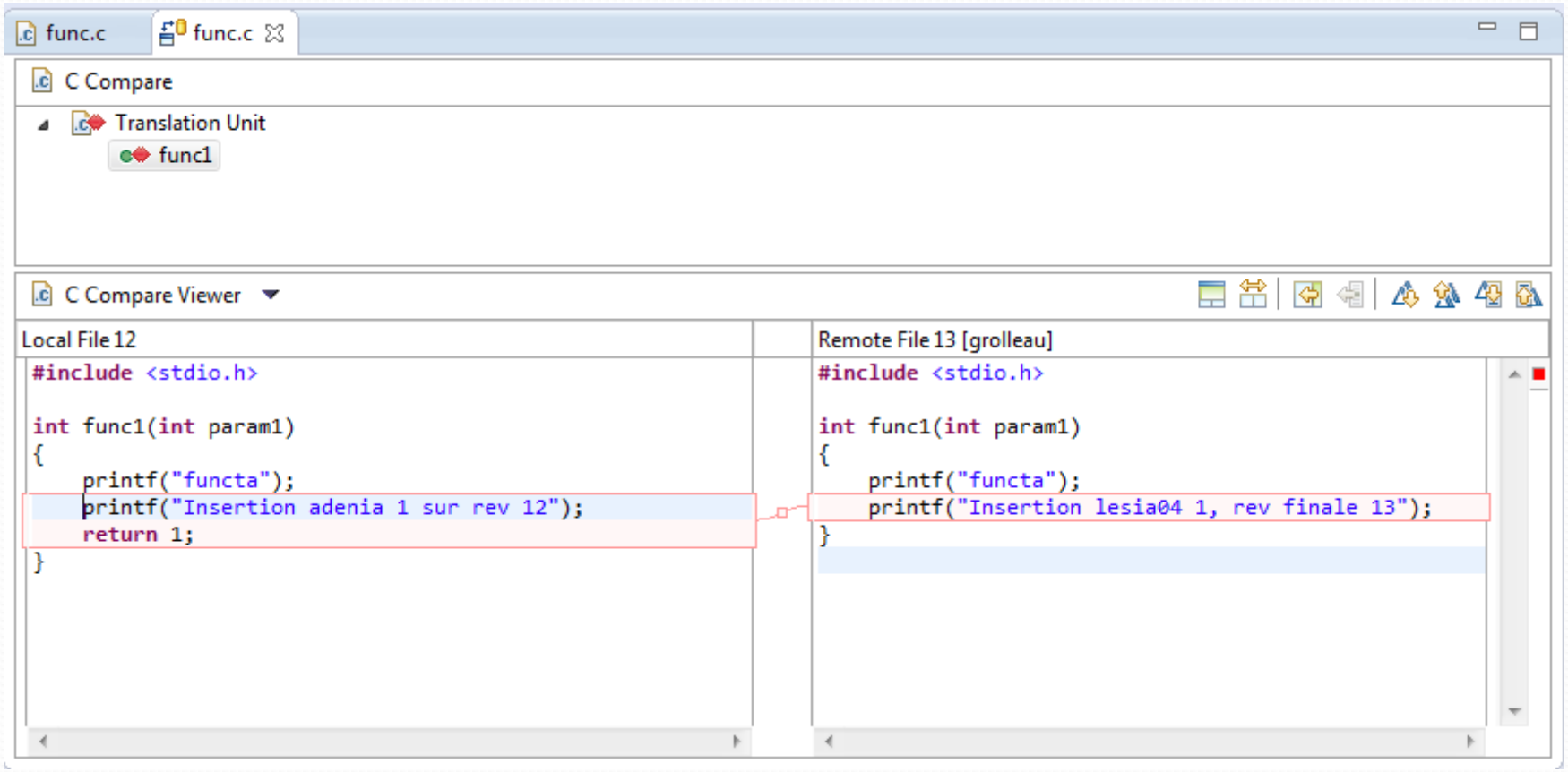
# Conflits


File Edit Navigate Search Project Run Window Help
Quick Access
C/C++
Team Synchronizing
Exploration du référentiel SVN
Synchronize
SVN (Workspace)
test1
src
func.c
func.c
func.c
C Compare
Translation Unit
func1
C Compare Viewer
Local File 12
Remote File 13 [grolleau]
#include <stdio.h>
int func1(int param1)
{
printf("functa");
printf("Insertion adenia 1 sur rev 12");
return 1;
}
#include <stdio.h>
int func1(int param1)
{
printf("functa");
printf("Insertion lesia04 1, rev fin
}

# Conflits : Eclipse, perspective « Team synchronising »

Il faut donc résoudre le conflit. Plusieurs options s'offrent à vous.

- Soit vous décidez que votre version est la bonne et le travail effectué par l'autre personne sera archivé.
  - Clic droit sur le fichier puis « Remplacer et valider » dans le menu.
- Soit vous décidez que le travail effectué par l'autre personne est le bon, votre travail sera archivé.
  - Clic droit sur le fichier puis « Surcharger et mettre à jour » dans le menu.
- Soit vous décidez de joindre les modifications effectuées par l'autre personne avec les vôtres.
  - Ouvrez le fichier dans l'éditeur de comparaison en double cliquant dessus.
  - L'éditeur de comparaison possède deux fenêtres, une pour le fichier local (à gauche) , qui est utilisable pour modifier le fichier local et une fenêtre à droite pour le fichier distant (sur le serveur SVN) en lecture seule.


func.c
func.c
C Compare
Translation Unit
func1
C Compare Viewer
Local File 12
#include <stdio.h>
int func1(int param1)
{
printf("functa");
printf("Insertion adenia 1 sur rev 12");
return 1;
}
Remote File 13 [grolleau]
#include <stdio.h>
int func1(int param1)
{
printf("functa");
printf("Insertion lesia04 1, rev finale 13");
}

# Conflits : Eclipse, perspective « Team synchronising »

- Une fois que le fichier local a été modifié pour prendre en compte le travail des deux personnes,
  - Enregistrer le fichier.
  - Clic droit sur le fichier puis « Remplacer et valider » dans le menu.

- la perspective de synchronisation a une troisième fenêtre qui permet d'afficher l'historique d'un fichier.
  - Clic droit sur un fichier puis « Afficher l'historique des ressources »

# Les conflits – suite
# Cas 2 : modifications incompatibles

- Ce type de conflit est à gérer manuellement. Subversion modifie le répertoire local du développeur en créant des fichiers supplémentaires :
  - file : contient maintenant l'ensemble des modifications
  - file.mine : contient le fichier file modifié par l'utilisateur
  - file.r#old (où #old est le numéro de l'ancienne révision) : c'est le fichier de la base avant que l'utilisateur ne fasse ses propres modifications ;
  - file.r#new (où #new est le numéro de la nouvelle révision) : c'est le fichier de la base le plus récent.
- L'utilisateur modifie le fichier en accord avec l'autre utilisateur et avertit le serveur que le conflit est résolu par la commande svn resolved.
- Ensuite, on peut soumettre les modifications par svn commit.

# Principales fonctions

| | |
|---|---|
| svn checkout | récupérer le contenu de la base |
| svn commit | soumettre des modifications à la base |
| svn import | importer un projet dans la base |
| svn resolved | indiquer que les conflits ont été résolus |
| svn revert | annuler toute modification locale |
| svn update | mettre à jour le répertoire local |
| svn cat | lire le contenu d'un fichier de la base |
| svn diff | regarder les différences entre des versions de la base |
| svn info | obtenir des infos sur le réperoire local |
| svn list | lister le contenu de la base |
| svn log | voir les messages accompagnant chaque révision de la base |
| svn status | afficher l'état des fichiers/dossiers du répertoire local |
| svn add | ajouter un fichier/dossier dans l'arborescence de la base |
| svn copy | copier des fichiers/dossiers dans l'arborescence de la base |
| svn delete | supprimer des fichiers/dossiers de l'arborescence de la base |
| svn mkdir | créer un dossier dans l'arborescence de la base |
| svn move | déplacer des fichiers/dossiers dans l'arborescence de la base |

# svn status

- With no arguments, it prints only locally modified items (no repository access)
- svn status --show-updates toto.c (ou svn status –u)
  - ' ‘ The item in your working copy is up-to-date.
  - '*'A newer revision of the item exists on the server.
  - '?‘ Item is not under version control.
  - ‘!’ Item is missing
  - 'A‘ Item is scheduled for Addition.
  - 'D‘ Item is scheduled for Deletion.
  - 'M‘ Item has been modified.
  - 'C‘ Item is in conflict with updates received from the repository.

# Tags et Branches : svn copy

- La commande svn copy sert à la fois pour la création de Tags et de branches
- Création d'un Tag à partir de la branche principale (trunk) :
  - *svn copy* https://XXX/OSAE/GroupeXX/trunk http://XXX/OSAE/GroupeXX/tags/test1-release-1.0 *-m "Tagging the 1.0 release of the 'test1' project."*
- Création d'une branche à partir de la branche principale (trunk) :
  - *svn copy* https://XXX/OSAE/GroupeXX/trunk http://XXX/OSAE/GroupeXX/branches/test1-dev1 *-m « Branch the of the 'test1' project."*

# Fusion de branches

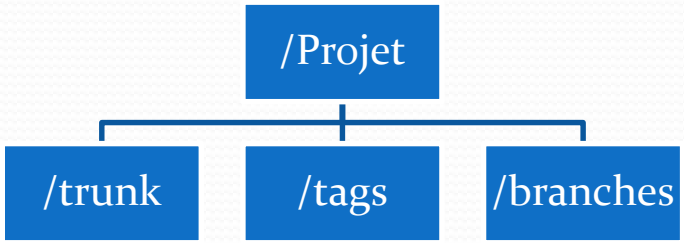
- Privilégier la ligne de commande
  - Se placer à l'intérieur du répertoire de la branche (cd test1-dev1)
    - svn log --stop-on-copy => affiche les modifications depuis la création de la branche,
      - noter la plus ancienne révision rXX
  - Se placer à l'intérieur de trunk (si nécessaire faire *svn co https://version-lesia.obspm.fr/repos/OSAE/trunk/src/*
    - Faire un svn upate
      - noter la révision YY (il s'agit de la dernière révision)
  - Faire le merge (depuis le répertoire trunk)
  - cd trunk
  - *svn merge --dry-run –rXX:YY https://version-lesia.obspm.fr/repos/OSAE/branches/src-br1*
  - *ou svn-merge –rXX:HEAD*
  - *Tester le process*
  - *svn commit –m "Merged branch XXX into trunk"*

# Bonnes pratiques Subversion (1)

- Interactions avec le dépôt
  - Un « update » doit toujours précéder un « commit ».
  - Mettre à jour l'ensemble de la copie locale.
  - Mettre à jour la copie locale périodiquement.
  - Ne versionner que les fichiers nécessaires au projet.
  - Un « commit » doit représenter un tout.
  - Chaque « commit » doit comporter un message représentatif de l'objectif et de la raison des modifications envoyées et non un résumé des modifications

# Bonnes pratiques Subversion (2)

- Gestion de projet
  - Il existe des schémas recommandés concernant le choix de l'arborescence d'un dépôt.
    - Exemple :



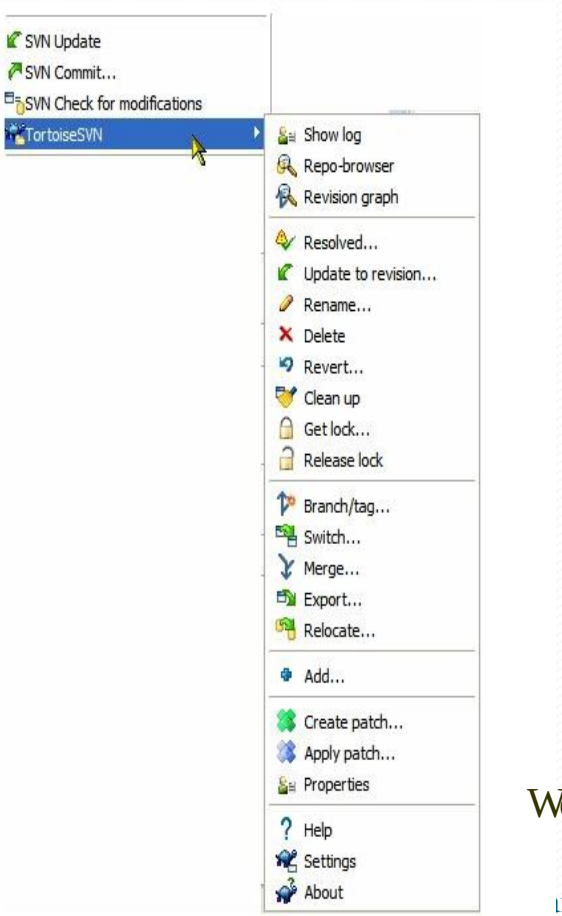

- trunk : Contient généralement la version la plus récente et en cours de développement.
- tags : Contient les différentes versions stables (« releases») d'un logiciel .
- branches : Contient des développements en parallèle.

  - Liste officielle : http://svn.apache.org/repos/asf/subversion/trunk/doc/user/svn-best-practices.html

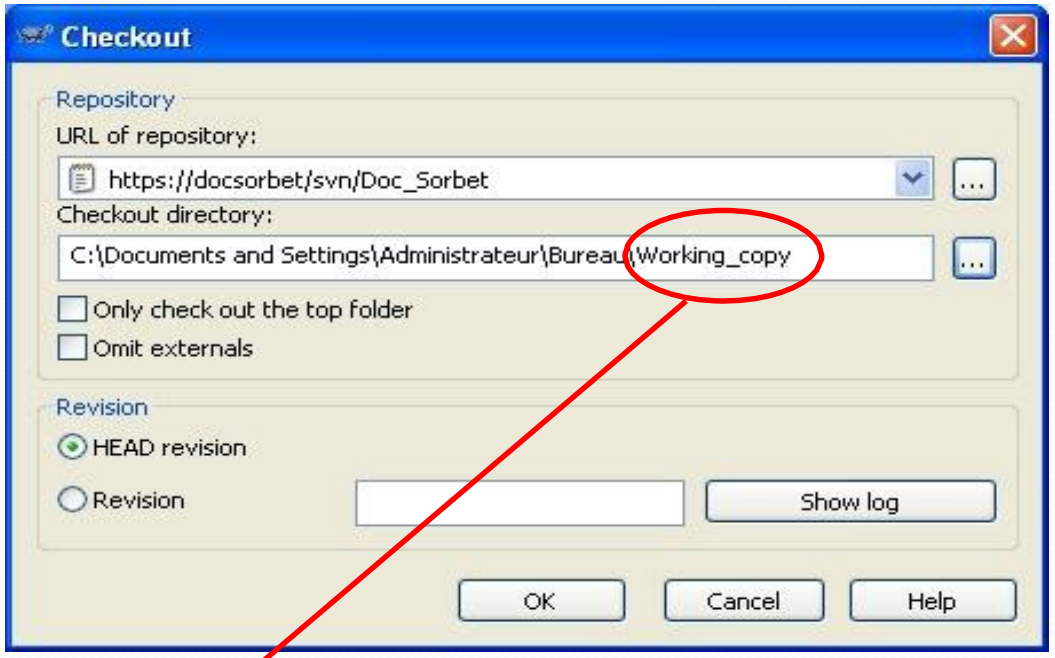
# Client Subversion Windows

- Aucune configuration particulière n'est nécessaire pour utiliser Subversion, il suffit juste de se procurer un client qui permette de se connecter et de communiquer avec le dépôt.

**TortoiseSVN** dans l'explorateur




Récupérer la version courante du projet (menu checkout)




Working_copy est un nouveau dossier vide crée en local par l'utilisateur.

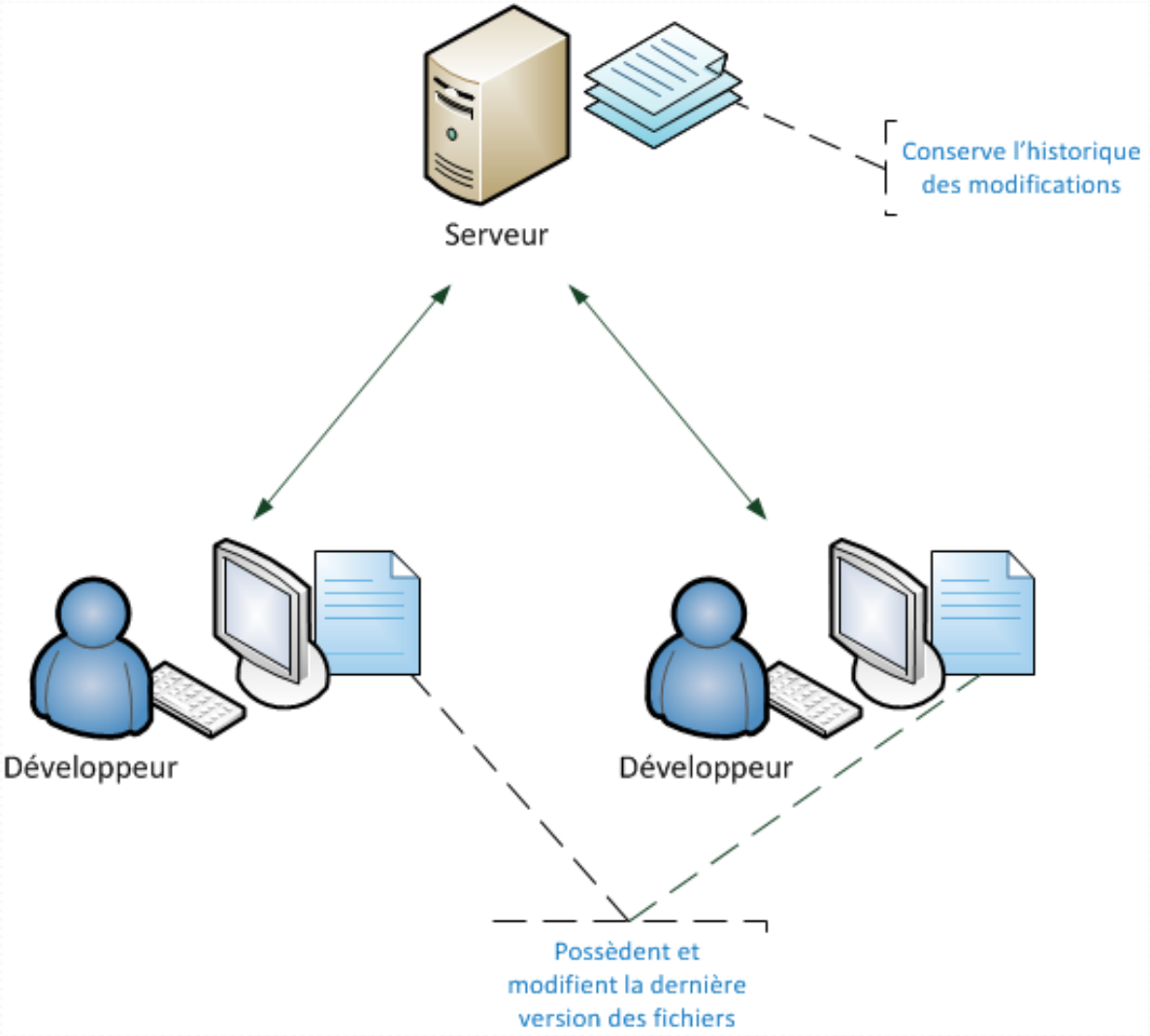
# Client Subversion Linux

- Il existe aussi des clients graphique svn sous Linux :
  - RapidSVN (http://rapidsvn.tigris.org/),
  - eSvn (http://esvn.umputun.com/) (multiplateforme),
  - Subversive (www.eclipse.org/subversive/ ) (*plugin* pour Eclipse),
  - psvn.el (http://xsteve.nit.at/prg/vc_svn/) (mode pour Emacs) . . .
- Une liste complète de clients est disponible : http://fr.wikipedia.org/wiki/Comparaison_des_clients_pour_Subversion

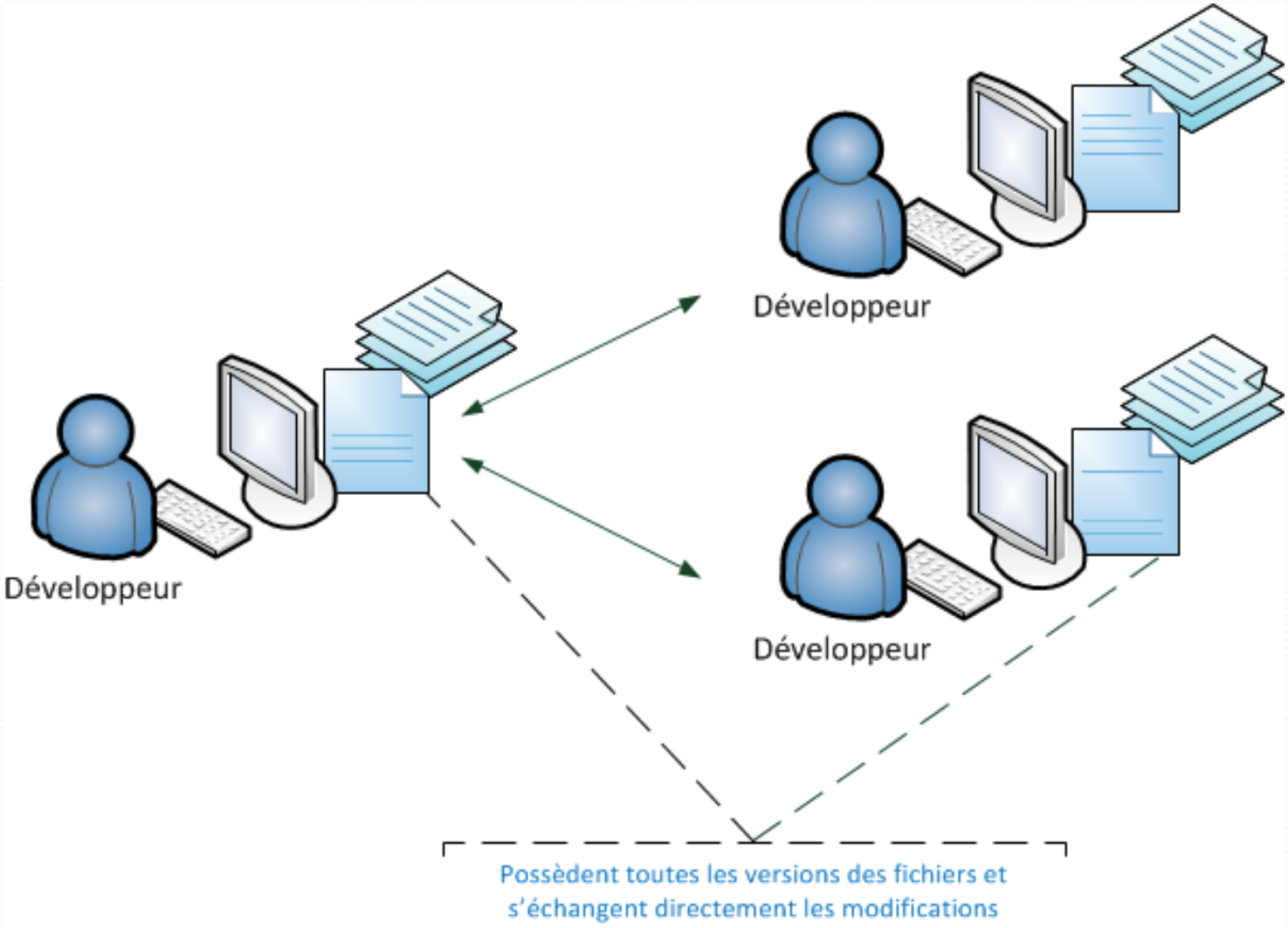
# Outils centralisés et distribués

- Il existe deux types principaux de logiciels de gestion de versions
- Les logiciels centralisés : un serveur conserve les anciennes versions des fichiers et les développeurs s'y connectent pour prendre connaissance des fichiers qui ont été modifiés par d'autres personnes et pour y envoyer leurs modifications.
  - Subversion, CVS
- Les logiciels distribués : il n'y a pas de serveur, chacun possède l'historique de l'évolution de chacun des fichiers. Les développeurs se transmettent directement entre eux les modifications, à la façon du peer-to-peer.
  - GIT

# Outil de gestion de version **centralisé** (CVS, SVN)



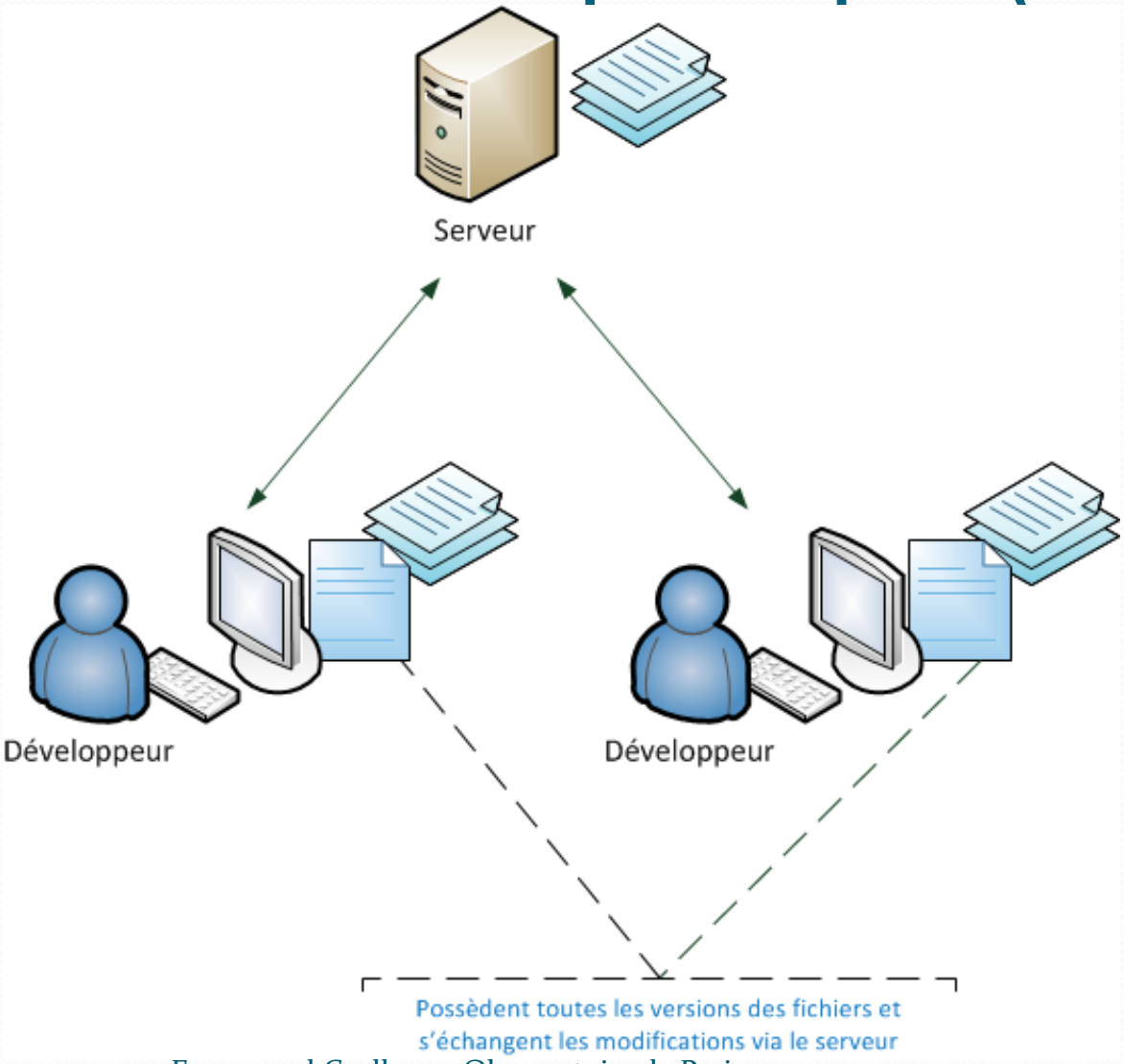
# Outil de gestion de version **distribué**



# Outil de gestion de version distribué : dans la pratique

- Dans la pratique, les logiciels distribués sont rarement utilisés comme sur le schéma précédent.
- Même lorsque les logiciels sont capables de fonctionner en mode distribué, on utilise très souvent un serveur qui sert de point de rencontre entre les développeurs.
- Le serveur connaît l'historique des modifications et permet l'échange d'informations entre les développeurs, qui eux possèdent également l'historique des modifications.
- Avantage : meilleure gestion des branches

# Outil de gestion de version distribué : dans la pratique (GIT)



# Sources

- SVN http://subversion.apache.org/
- Manuel SVN : http://svnbook.red-bean.com/
- Liste des clients SVN : http://fr.wikipedia.org/wiki/Comparaison_des_clients_pour_Subversion
- GIT
  - https://openclassrooms.com/courses/gerer-son-code-avec-git-et-github