

# MEDEV

Myriam Servières, Guillaume Moreau, Carito Guziolowski

21/11/19



# (Motivation) TP1 – bilan EI2

- **Bilan fait par les EI2**

- Points positifs :

- Tout a été fait, et **git** a été utile pour partager les fichiers.
    - Beaucoup de temps a été gagné en **parallélisant les tâches**.

- Points négatifs :

- certaines tâches ont été faites 2 fois, à cause d'un **manque de communication des différents groupes**.
    - La quasi-totalité des collaborateurs du projet n'avaient qu'une **vue réduite** du projet et n'avaient pas connaissance de l'avancement des autres. Le problème était surtout que personne ne savait réellement où en était précisément le projet.

# TP1 – bilan EI2 (continuation)

- **Observations de l'enseignant + rendu**
  - 8h30 Des interactions entre le groupe qui modélise le problème au tableau et les groupes « développeurs »
  - au moins 1 personne qui coordonne
  - 2ème moitié du TP : tout les personnes ont une tache à faire
  - IDE : Netbeans
  - Pas d'exécutable (e.g. .jar) fonctionnel
  - Code pas complètement fonctionnel (?) → code compile, initialise joueurs, mais sans moyen d'afficher les étapes du jeu

# (Motivation) TP1 – bilan EI3

- **Bilan fait par les EI3**

- Organisation :

- 1 personne qui coordonne la bonne repartition du travail à faire
    - 1 personne qui rassemblera les morceaux du code
    - Utilisation de discord (free voice chat pour gamers) pour échanger les documents
    - **GIT** a été utilisé pour déposer les sources, et pas pour se coordonner dans l'équipe de développeurs

- Problèmes rencontrés

- **Communication : coordination entre groupe qui fait le Diagramme de Classes est le groupe qui Développe**
    - **Déconnexion du code** : «Les gens ne pouvaient pas compiler ou tester leur code au fur et à mesure car chacun travaillait de se côté sur une classe »
    - **Déséquilibre de la charge du travail**
      - « Une grosse charge et une grosse pression a été concentré sur la personne qui rassemblait le code car le code était mal formé (erreur de syntaxe, erreur de nommage, spécification non suivies,...) »
      - Répartition inégale et un rendu qui ne fonctionne pas très bien.

# TP1 – bilan EI2 (continuation)

- **Observations de l'enseignant + rendu**

- 9h34 : 1 personne qui coordonne les sous-groupes, 1 personne qui récupère les codes et les teste
- 10h34 chaque personne a une tâche à faire
- Une longue période : ~8h15 à 9h34 où le groupe développeurs n'était pas investi dans le code → source d'inégalité, déséquilibre, surcharge, *free-time*
- IDE : Netbeans
- Pas d'exécutable (e.g. .jar) fonctionnel
- Code pas complètement fonctionnel → code compile, initialise plateau et l'affiche. Méthodes de joueurs implémentés mais pas testés (?)

# MEDEV

- **Texte en jaune** ... et autres
  - Outils de partage du code : GIT, SVN
  - IDE en collaboration : Netbeans + (GIT ou SVN)
  - Test Unitaires
  - Check bugs, code smells/vulnerabilities : Sonarqube
  - Building deploying projects, repetitive tasks to configure the code : Ant Jenkins (Java), Makefile

~not MEDEV



Source: Université Catholique de Louvain  
<https://uclouvain.be/fr/etudier/III/cahier-groupe.html>

## Texte en vert

- Le travail en équipe c'est beaucoup plus que la technique informatique
- Des techniques de communication sont très importantes à acquérir pour rendre la vie plus facile à tout le monde au sein d'une équipe (aussi de développeurs informatiques)

*Comment tenir compte des étudiant·es qui se reposent sur le travail de leurs pairs ?*

*Comment initier une dynamique de groupe propice à l'apprentissage ?*

*Fait-il partie de la responsabilité académique de se préoccuper des conflits relationnels qui surgissent parfois au sein des équipes étudiantes ?*

# Subversion – SVN (MEDEV)

Carito Guziolowski





# Plan

## Introduction SVN

Modèles de partage

SVN en action

Cycle de travail de base

Recommandations

SVN pour Windows

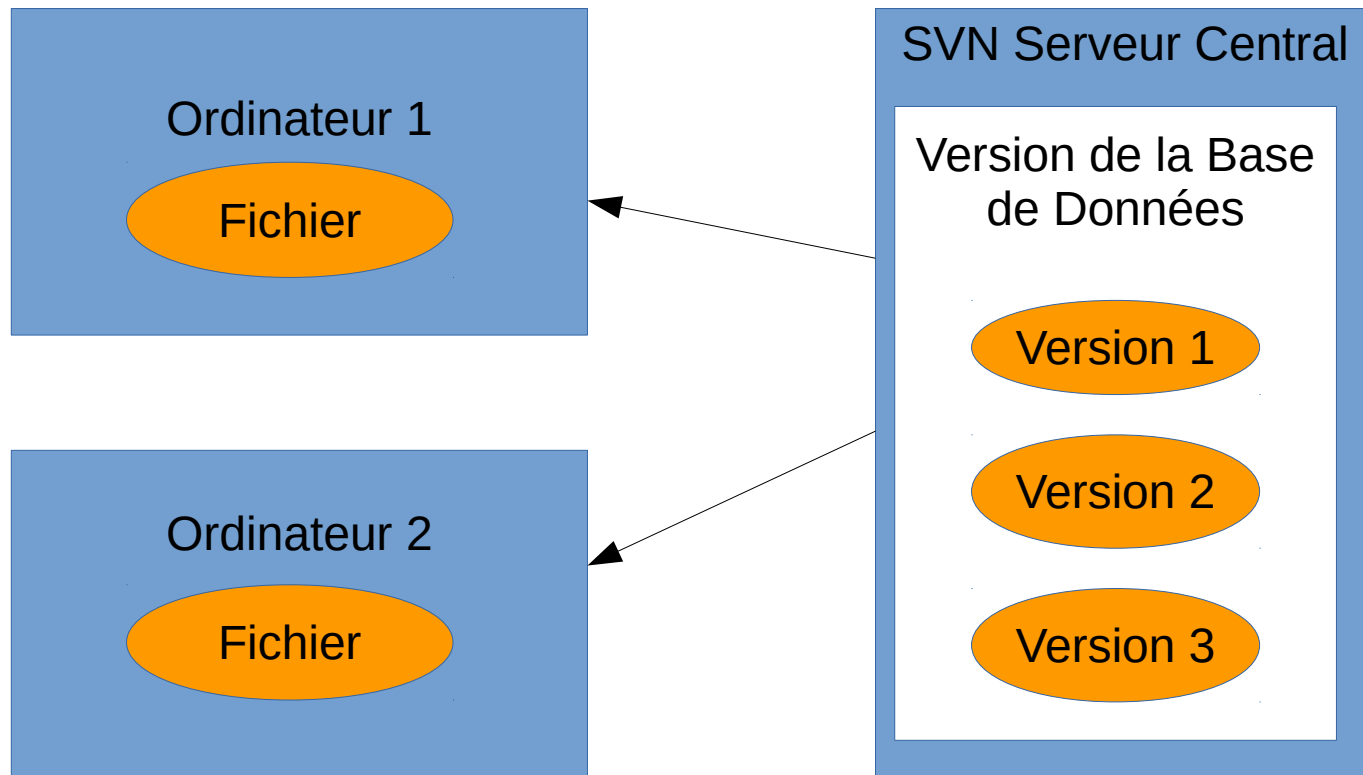
# Contrôle de versions

- Pour quoi ?
  - Travail à plusieurs simultané
  - Organisation du partage du code (orienté objet)
  - Gagner en temps du développement et test
  - Historique de changements effectués dans le code (qui, quoi et quand)
- Nécessite une bonne conception du projet

# Subversion (SVN)

- Subversion (SVN) est un système de Contrôle de Versions (*Version Control System* - VCS) gratuit et open-source
- But de systèmes du type VCS
  - Gérer des fichiers et dossiers et leurs évolutions au fil du temps
  - Systèmes : CVS, **SVN**, Perforce
- Distribué sur le réseaux → possibilité de travail à distance

# SVN - architecture



# SVN est le meilleur outil pour moi ?

- Le prix à payer :
  - S'adapter à copier, coller, déplacer, renommer, effacer des fichiers **différemment**
  - SVN duplique les données à tous les collaborateurs  
→ il ne s'agit pas seulement de partager des données statiques : photos, musique, logiciels.
  - SVN offre un historique local **peu profond des versions** (en comparaison à d'autres systèmes distribués du contrôle de versions, DVCS, e.g. GIT).

# (VCS vs. DVCS)

## SVN

- Système centralisé
- Stockage plus superficiel du système des versions
- Modèle plus simple à utiliser
  - Plus du confort pour la collaboration
  - Instructions plus simples
- Malgré son ancienneté :
  - 24 % de projets « open-source », source : <https://www.openhub.net/repositories/compare>
  - Entreprises

## GIT

- Besoin d'une **connexion au réseau rapide**
- **Coût de stockage bas**
- Le stockage de tout le projet (avec toutes les versions) sera local
- Bonne performance pour des opération journalières
- Meilleur support pour la gestion des « merges » entre les branches du développement

# Plan

Introduction SVN

[Modèles de partage](#)

SVN en action

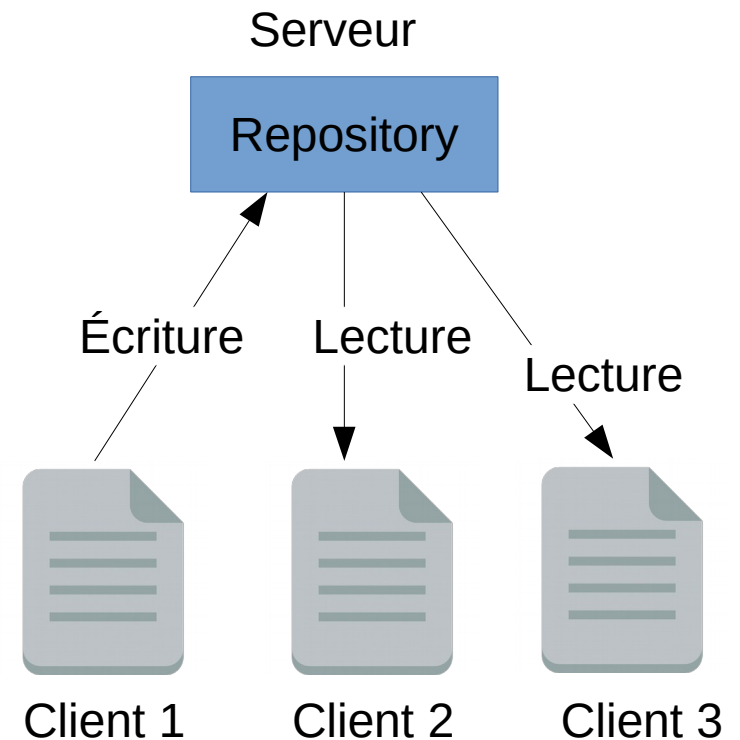
Cycle de travail de base

Recommandations

SVN pour Windows

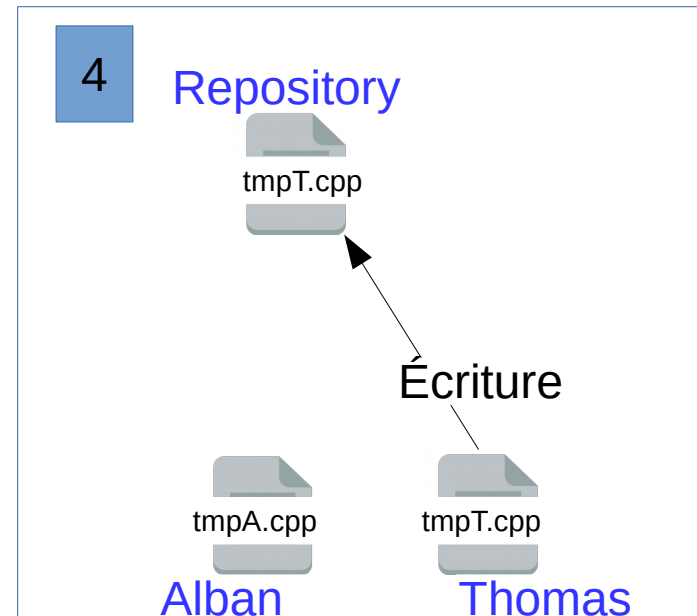
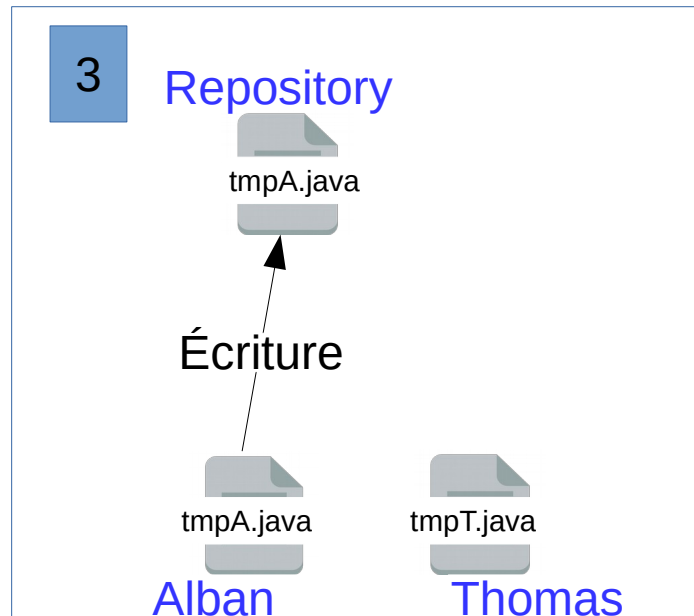
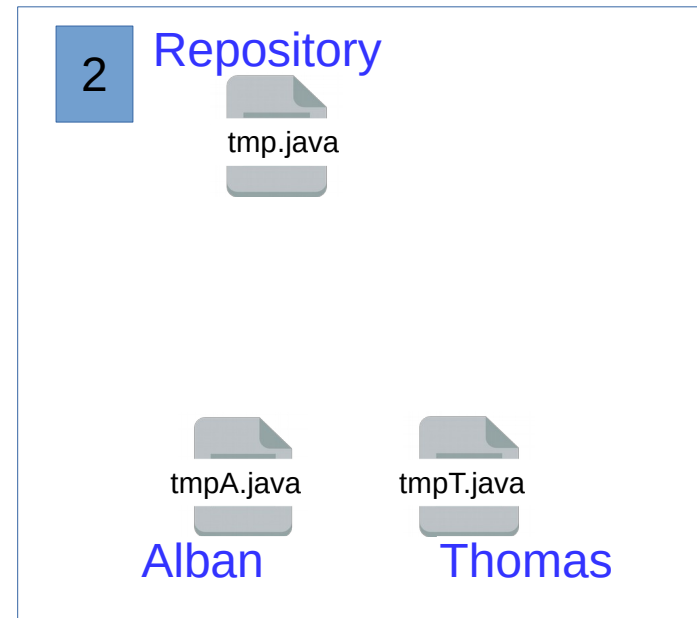
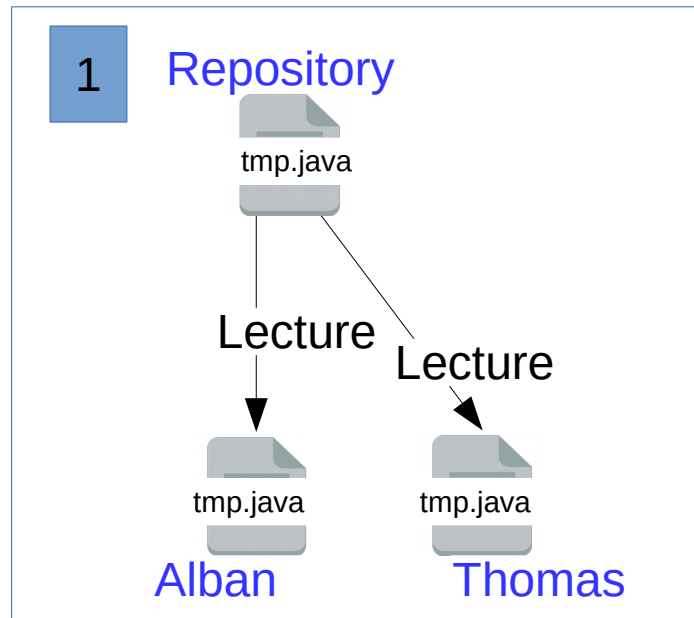
# Dépôt (*Repository*)

- Cœur du SVN : **stockage centralisé** (dépôt)
- Hiérarchie arborescente de fichiers et dossiers
- **Client**
  - Écriture : mettre à disposition des autres son fichier
  - Lecture : accéder à l'information des autres
- **Serveur** : stockage de fichiers avec leur historique
  - Écriture (client) → création de plusieurs versions
  - Lecture (client) → obtenir la version courante, version de hier, etc.

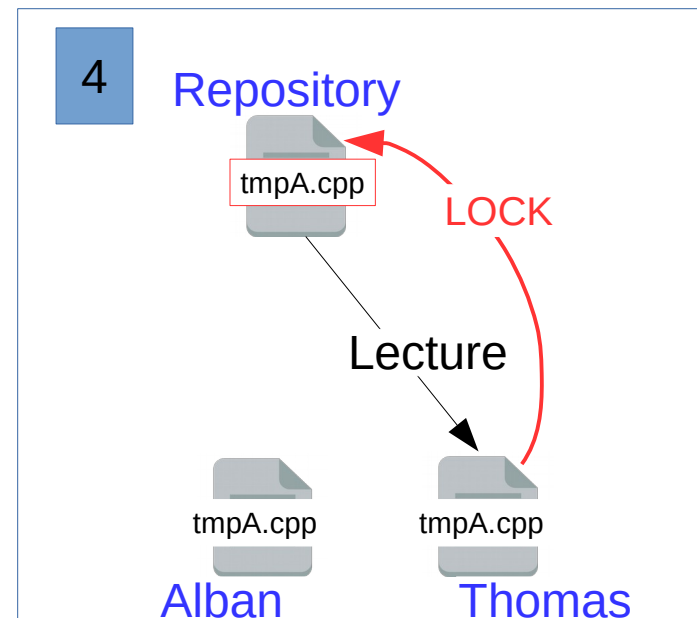
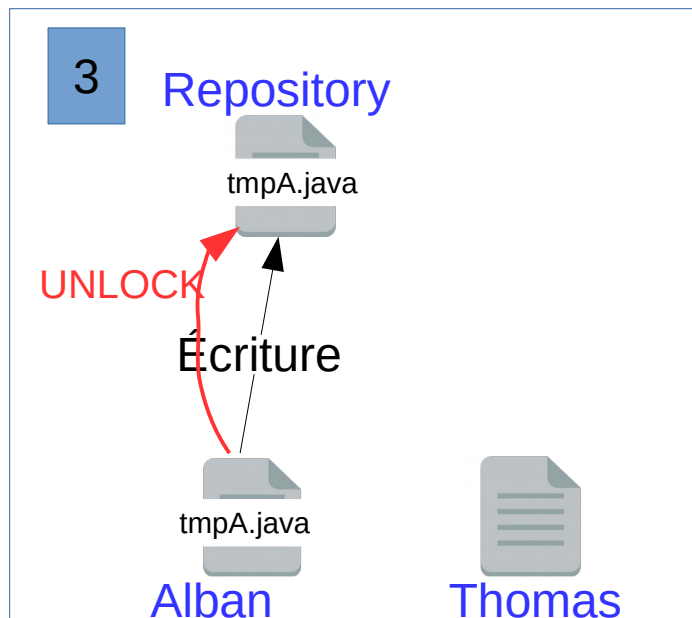
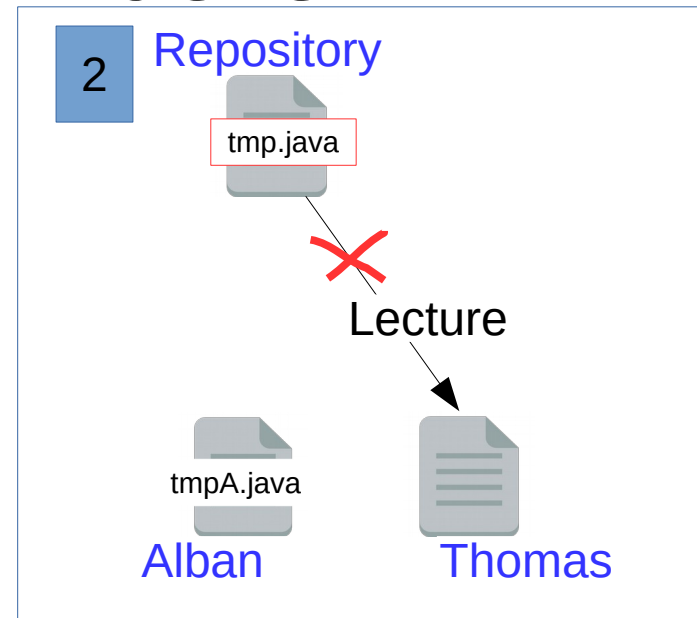
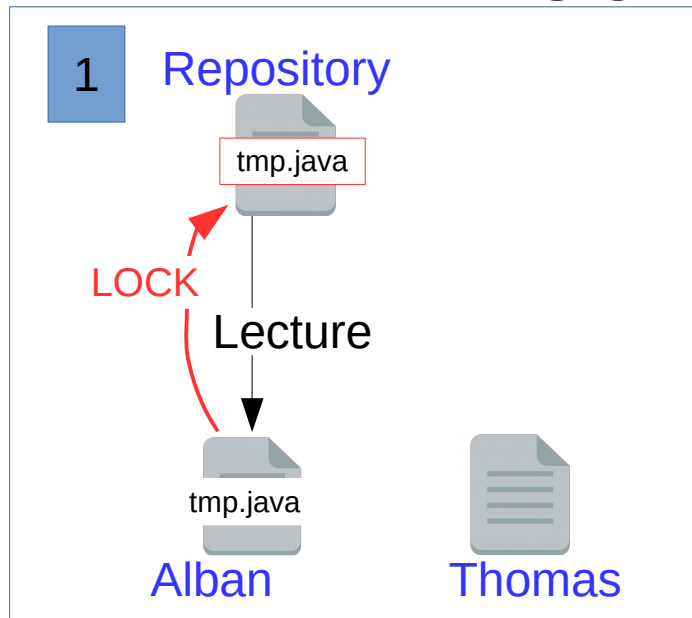




# Ce qui faut éviter



# Une solution : modèle verrouiller-modifier-libérer



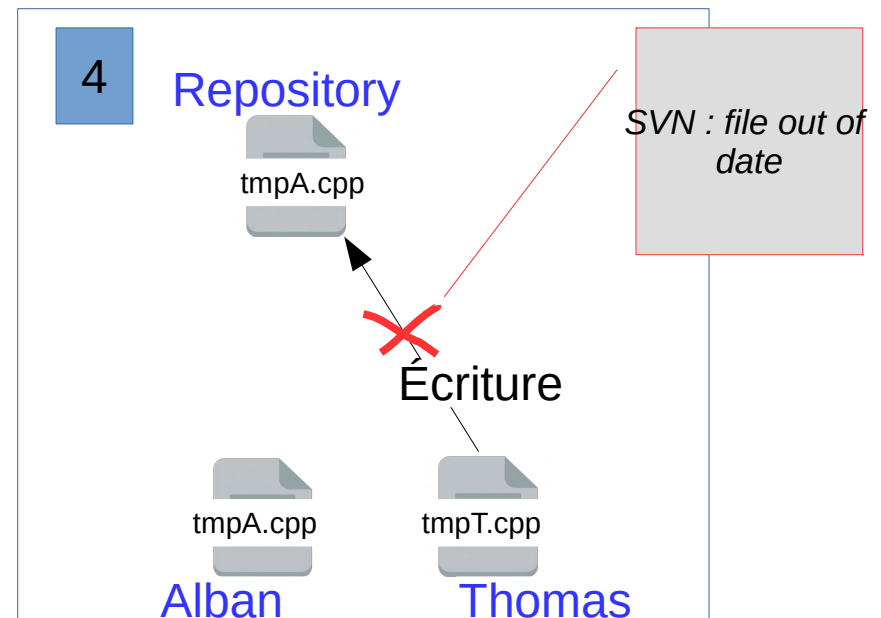
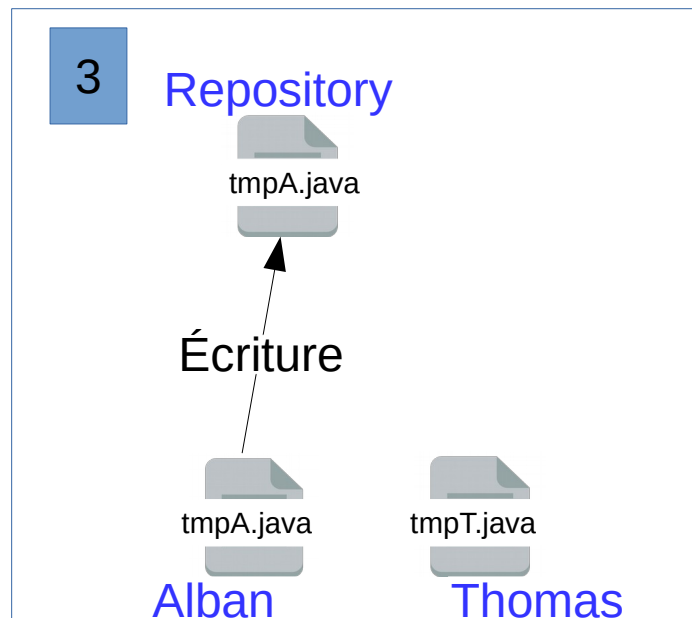
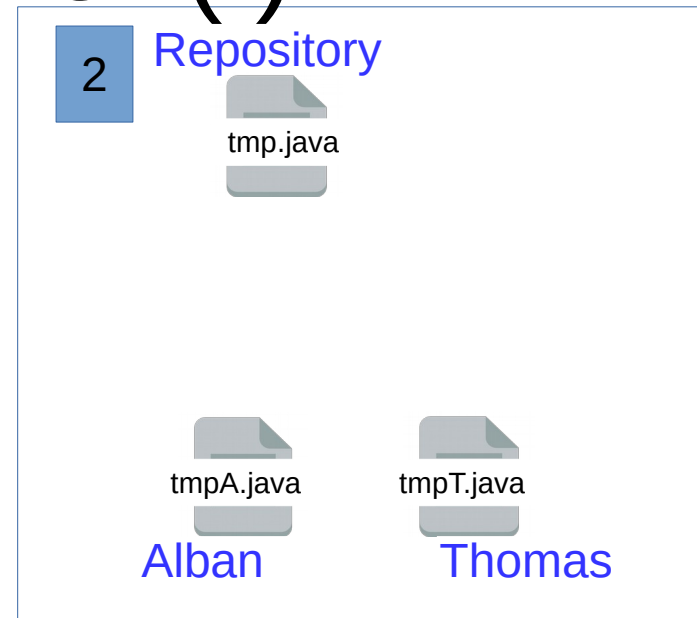
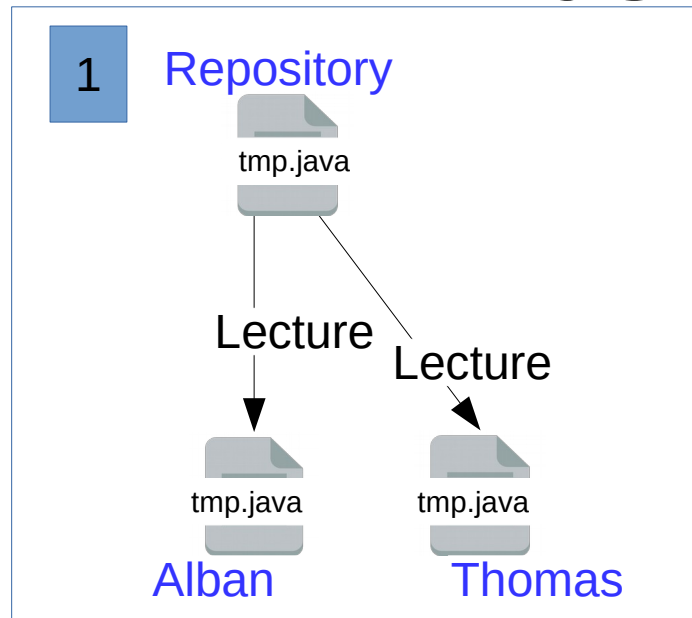
# Modèle verrouiller-modifier-libérer (quelques problèmes)

- Administratives
- Alban et Thomas peuvent éditer des fonctions différentes dans les fichiers
- Sens de sécurité irréal : éditer 2 fichiers différentes ne veut pas dire que le projet continuera à fonctionner

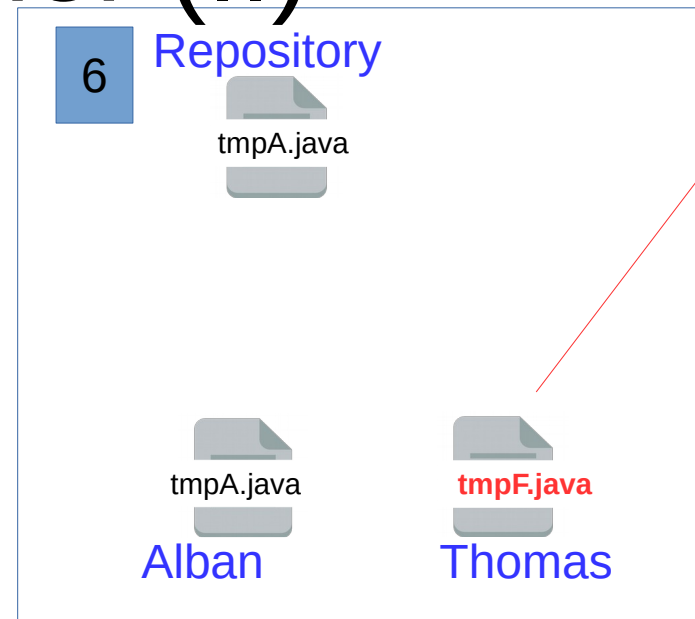
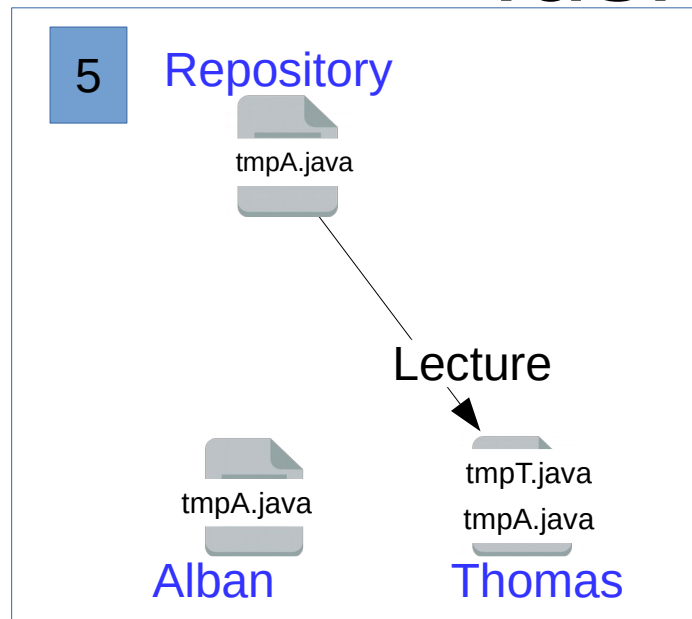
# Solution SVN : modèle copier-modifier-fusionner

1. Chaque client crée une copie locale (*personal working copy*) du Repository
2. Travail simultané de plusieurs clients sur le même fichier
3. Toutes les copies personnelles seront fusionnées dans une version finale
  - SVN assiste cette fusion
  - Implication humaine dans cette fusion

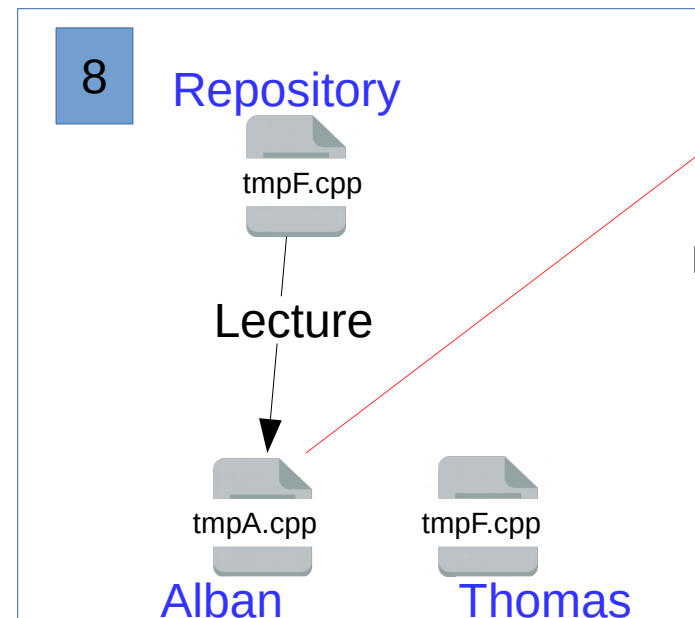
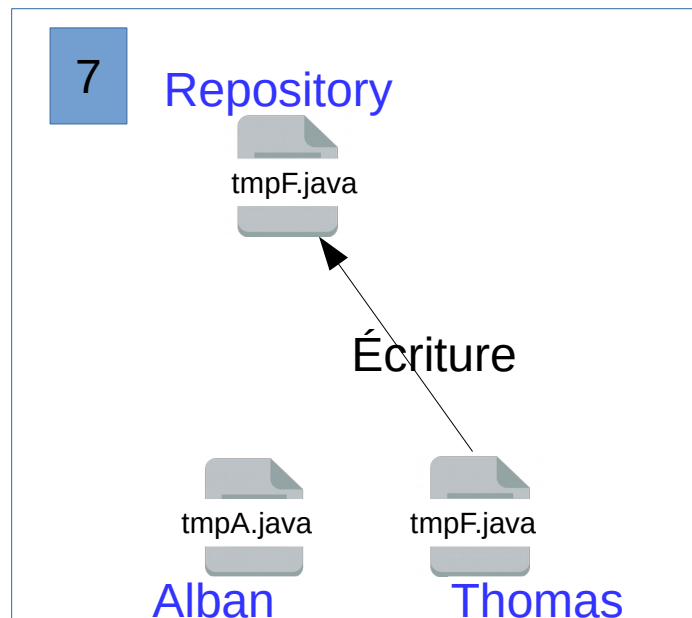
# Exemple du modèle copier-modifier-fusionner (I)



# Exemple du modèle copier-modifier-fusionner (II)



Une version fusionnée est créée  
(la fusion est faite par le svn ou par l'humain)



Les 2 clients partagent leur changements

# Plan

Introduction SVN

Modèles de partage

[SVN en action](#)

Cycle de travail de base

Recommandations

SVN pour Windows

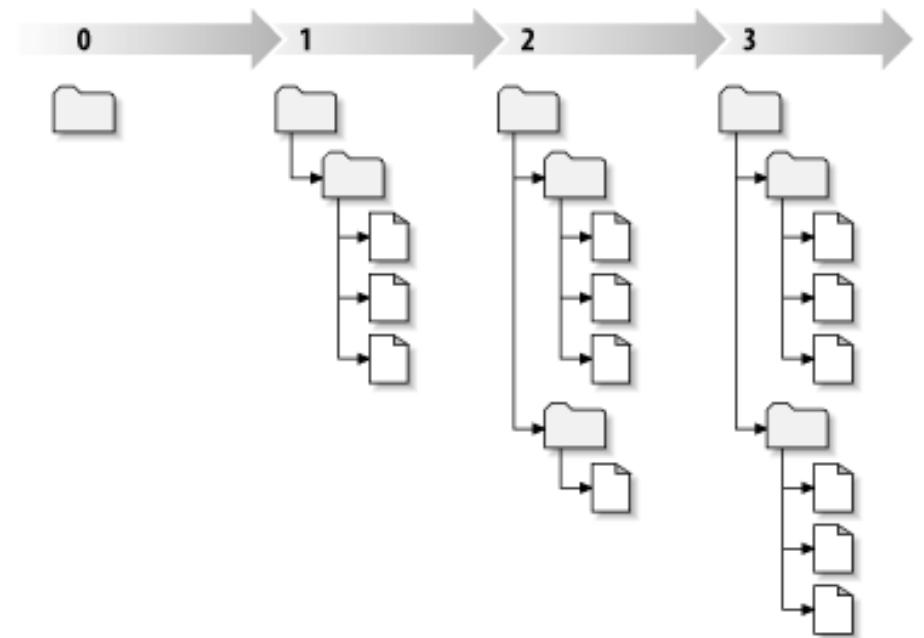
# SVN conflits

- Si la même ligne du fichier est éditée → *conflict*
- Les lignes en « conflit » sont étiquetées par le système lors de la fusion des fichiers.
- Thomas pourra voir les 2 versions et résoudre le conflit manuellement (après discussion avec Alban)
- Les conflits sont rares  
→ Communication parmi les collaborateurs



# SVN : commits, revisions

- **Commit** → changement à 1 fichier ou dossier
- Un client communique les changement faits à 1..n fichiers ou dossiers
- Chaque commit accepté par le Repository crée un état de l'arbre du système de données (filesystem tree) appelé *revision*
- Les états de cet arbre sont numérotées



Extraite de « Version Control with Subversion »  
Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato

# Utiliser SVN (client)

- Ubuntu (Linux)

```
$sudo apt-get install subversion
```

(exemples en svn 1.9.3)

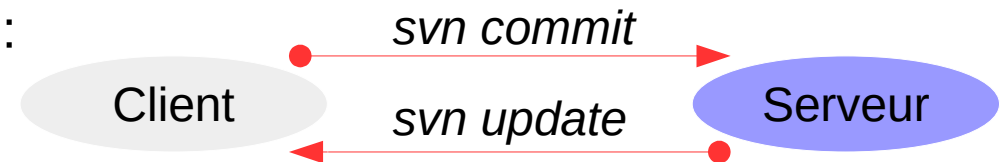
```
$svn checkout https://subversion.ecnantes.fr/svn/elevés
```

```
cguziolo@cguziolo-Latitude-7480:~/Cours/MEDEV/svn_ecn_etu/elevés$ ls
Info  MaisonConnectee  Nantes1900  PAPPL  R_et_D  UrbanISTIC  VilleNumerique
cguziolo@cguziolo-Latitude-7480:~/Cours/MEDEV/svn_ecn_etu/elevés$
```

Working copy du serveur SVN de l'ECN

# État d'un fichier

- Pour chaque fichier dans la copie locale, svn garde :
  - Revision
  - « timestamp » dernière mise à jour de la copie locale dans le Repository
- Un fichier peut être classé comme :



	LOCAL	SERVEUR	<i>svn commit</i>	<i>svn update</i>
Unchanged & Current	–	–	rien	rien
Locally changed & Current	modifié	–	succès	rien
Unchange & Out of Date	–	modifié	rien	mise à jour
Locally Changed & Out of Date	modifié	modifié	« out-of-date » error	Essaie fusion automatique, sinon <i>conflict</i>

# Copie Locale (*working copy*)

- **Checkout** → créer une « working copy » d'un dossier MEDEV svn

```
cguziolo@cguziolo-Latitude-7480:~/Cours/MEDEV/svn_ecn_medev$ svn checkout https://subversion.ec-nantes.fr/svn/elevés/MEDEV
A    MEDEV/test.txt
Checked out revision 2790.
cguziolo@cguziolo-Latitude-7480:~/Cours/MEDEV/svn_ecn_medev$ ls -a MEDEV
.  .. .svn test.txt
cguziolo@cguziolo-Latitude-7480:~/Cours/MEDEV/svn_ecn_medev$
```

- La liste d'éléments « A » montre les éléments ajoutés à la copie locale du SVN
- Le dossier .svn contient des informations qui aident au svn à gérer la trace des fichiers présents dans la copie locale (ceux qui ne sont pas publiés, ou mises à jour)

# Svn commit



- **commit** → pour publier des changements d'un fichier dans le Repository

```
cguziolo@cguziolo-Latitude-7480:~/Cours/MEDEV/svn_ecn_medev/MEDEV$ ls
test.txt
cguziolo@cguziolo-Latitude-7480:~/Cours/MEDEV/svn_ecn_medev/MEDEV$ vi test.txt
cguziolo@cguziolo-Latitude-7480:~/Cours/MEDEV/svn_ecn_medev/MEDEV$ svn commit test.txt -m "add lines"
Sending          test.txt
Transmitting file data .done
Committing transaction...
Committed revision 2791.
cguziolo@cguziolo-Latitude-7480:~/Cours/MEDEV/svn_ecn_medev/MEDEV$
```

- Si un autre utilisateur (Alban) édite simultanément « test.txt » et fait un commit après nous

```
Alban$ svn commit -m "changes"
Sending          test.txt
svn: E155011: Commit failed (details follow):
svn: E155011: File '/home/cguziolo/Cours/MEDEV/svn_ecn_etu/elevs/MEDEV/test.txt' is out of date
svn: E160024: resource out of date; try updating
Alban$
```

# Svn update



- Si le fichier a été modifié il faut alors faire une mise à jour du Repository avec la commande « **update** »

```
Alban$ svn update
Updating '.':
U test.txt
Updated to revision 2791
Alban$
```

- Ensuite :
  - Si Alban a modifié le fichier « test.txt » dans une ligne différente, alors il suffira de faire « svn commit »
  - Sinon, Alban devra résoudre les conflits manuellement

# Help !

```
$ svn help | more
usage: svn <subcommand> [options] [args]
Subversion command-line client.
Type 'svn help <subcommand>' for help on a specific subcommand.
Type 'svn --version' to see the program version and RA modules
  or 'svn --version --quiet' to see just the version number.
```

Most subcommands take file and/or directory arguments, recursing on the directories. If no arguments are supplied to such a command, it recurses on the current directory (inclusive) by default.

Available subcommands:

- add
- auth
- blame (praise, annotate, ann)
- cat
- changelist (cl)
- checkout (co)
- cleanup
- commit (ci)
- copy (cp)
- delete (del, remove, rm)
- diff (di)
- export

# Plan

Introduction SVN

Modèles de partage

SVN en action

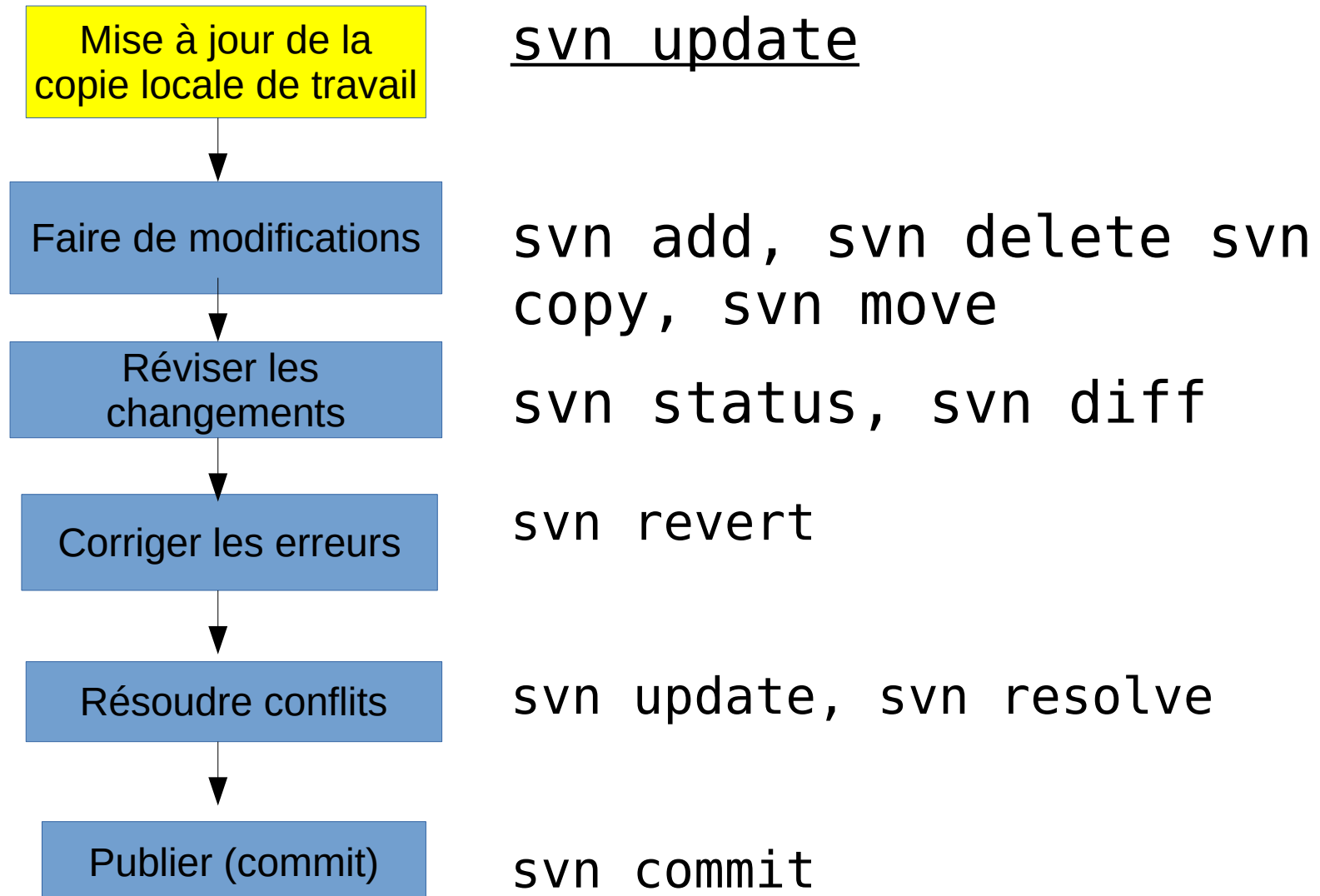
[Cycle de travail de base](#)

Recommandations

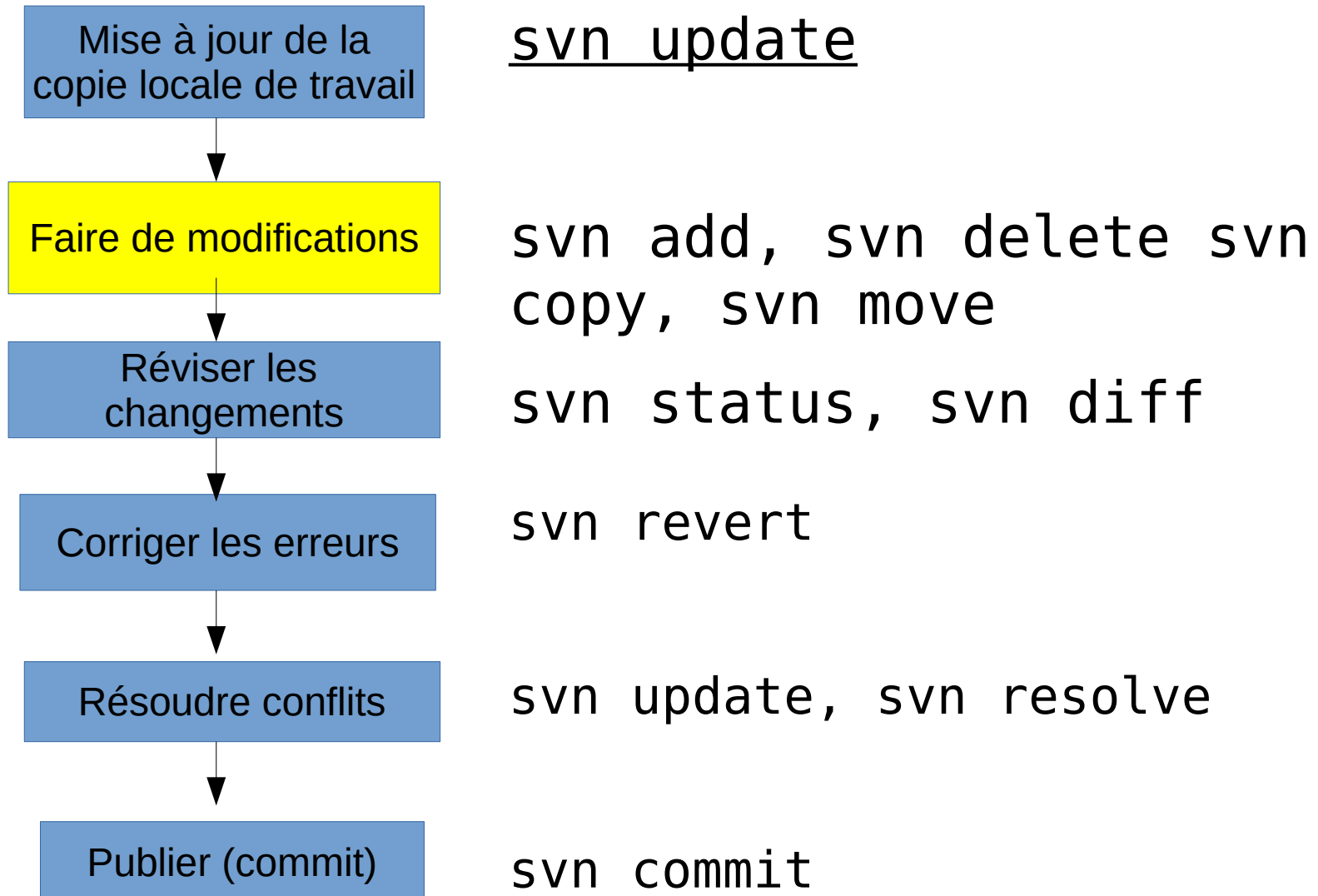
SVN pour Windows



# Cycle de Travail de base



# Cycle de Travail de base



# Les 5 instructions svn les plus utilisées

## svn add FOO

- Planifie l'ajout du fichier (ou dossier) FOO au Repository

```
$ vi FOO
$ svn add FOO ← ajout (étape 1) d'un fichier au SVN
A      FOO
$ ls
F00  test2.txt  test.txt
$ svn ls ← F00 n'est pas encore au SVN
test.txt
test2.txt
$ svn commit -m "added F00" ← ajout (étape 2) d'un fichier au SVN
Adding      F00
Transmitting file data .done
Committing transaction...
Committed revision 2798.
$ svn ls ← F00 est dans le SVN, sauf que ma « working copy » n'est pas mise à jour
test.txt
test2.txt
$ svn update
Updating '.':
At revision 2798.
$ ls ← F00 est aussi dans ma « working copy »
F00
test.txt
test2.txt
```

# Les 5 instructions svn les plus utilisées

## svn delete FOO

- Planifie la suppression du fichier (ou dossier) FOO au Repository

```
$svn delete FOO
D          FOO
$ls
test2.txt  test.txt
$svn ls
F00
test.txt
test2.txt
$svn commit -m "deleting FOO"
Deleting          F00
Committing transaction...
Committed revision 2801.
$svn ls
F00
test.txt
test2.txt
$svn update
Updating '.':
At revision 2801.
$svn ls
test.txt
test2.txt
$
```

Différence entre  
« working copy » et  
« Repository »

# Les 5 instructions svn les plus utilisées

## svn copy FOO BAR

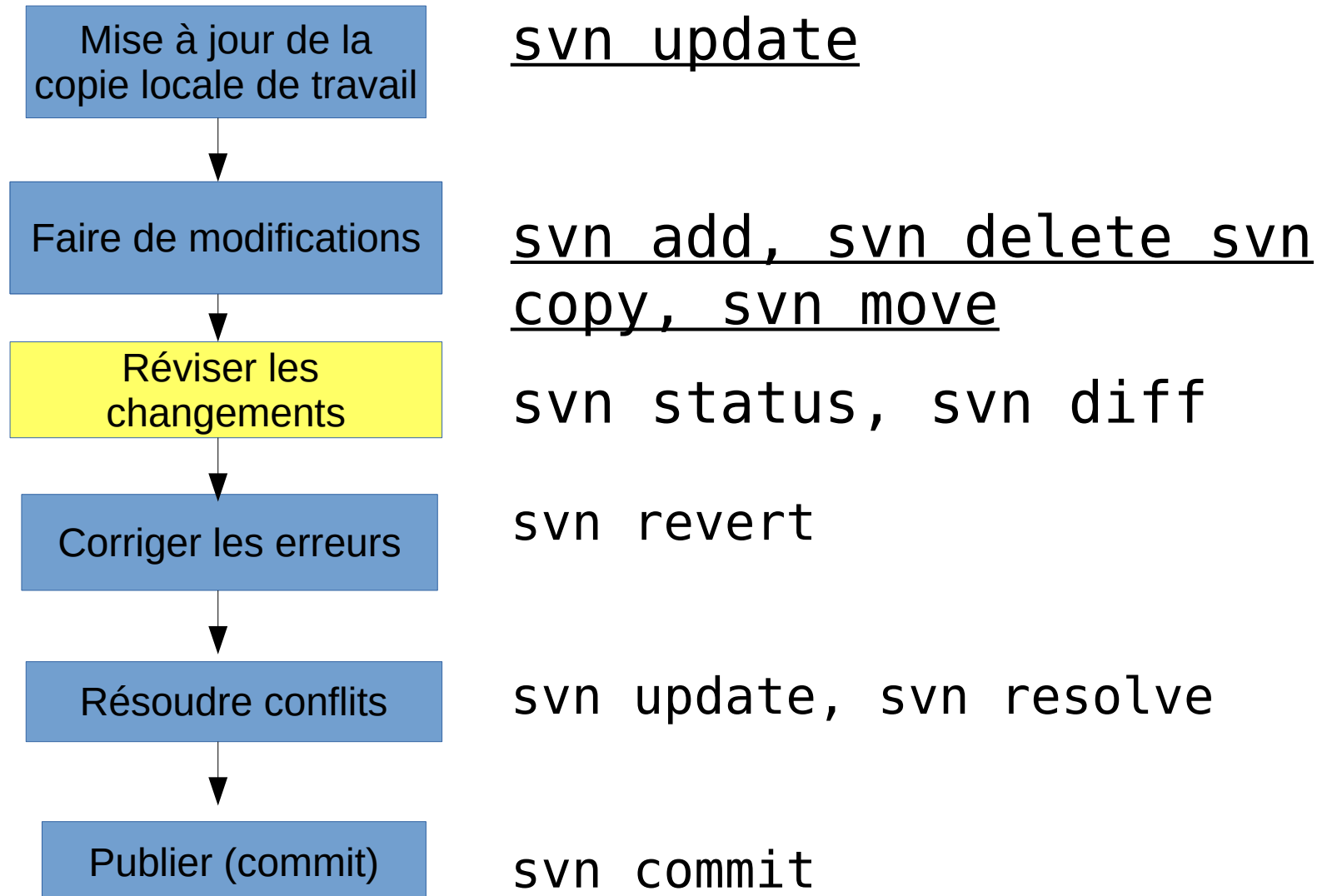
- Planifie la copie du fichier FOO au fichier BAR dans le Repository

```
$ls
FOO  test2.txt  test.txt
$svn ls
FOO
test.txt
test2.txt
$svn copy FOO BAR
A      BAR
$svn commit -m "copying FOO to BAR"
Adding      BAR
Committing transaction...
Committed revision 2803.
$svn ls
FOO
test.txt
test2.txt
$ls
BAR  FOO  test2.txt  test.txt
$svn update
Updating '.':
At revision 2803.
$svn ls
BAR
FOO
test.txt
test2.txt
```

# Les 5 instructions svn les plus utilisées

- `svn move F00 BAR` ↔ `svn copy F00 BAR ; svn delete F00`
- `svn mkdir F00` ↔ `mkdir F00 ; svn add F00`

# Cycle de Travail de base



# Réviser les changements

- *Good practice* : avant publier les modifications, regarder précisément ce que nous avons changé
- C'est possible d'utiliser `svn status`, `svn diff`, `svn revert` sans besoin d'être connecté au réseau



# Réviser les changements

**svn status** : Reporte tous les changements des fichiers faits dans le « working copy »

```
$svn status
M      F00 ← modifié
A      F003 ← planifié à être ajouté
?      test3.txt ← fichier qui n'est pas dans le SVN

$svn status --verbose
          2803      2803 cguziolo      .
          2803      2803 cguziolo      BAR
M      *      2805      2805 cguziolo      F00
          2805      2805 cguziolo      F002
A      -      ?      ?      F003
          *      2803      2792 cguziolo      test.txt
          2803      2793 alban      test2.txt
?                                     test3.txt
```

Dernière révision du  
Repository où l'élément  
avait changé

Utilisateur qui avait  
modifié l'élément

## Autres lettres :

D : Deleted  
(planifié pour suppression)  
C : en Conflit  
U : Updated (mis à jour)  
G : merGed (fusionné)

**svn status** rapporte l'état  
du SVN à la dernière  
mise à jour (svn update)

# Réviser les changements

**svn status -u -v:** Reporte tous les changements des fichiers faits dans le **SVN**

```
$svn status
M      F00 ← modifié
A      F003 ← planifié à être ajouté
?      test3.txt ← fichier qui n'est pas dans le SVN
```

```
$svn status --verbose
```

```
      2803      2803 cguziolo
M      2803      2803 cguziolo
      2805      2805 cguziolo
      2805      2805 cguziolo
A      -        ?      ?
      2803      2792 cguziolo
      2803      2793 alban
?
      *
```

```
.
BAR
F00
F002
F003
test.txt
test2.txt
test3.txt
```

Le fichiers F00 et test2.txt  
ont été modifiés depuis ma  
dernière mise à jour

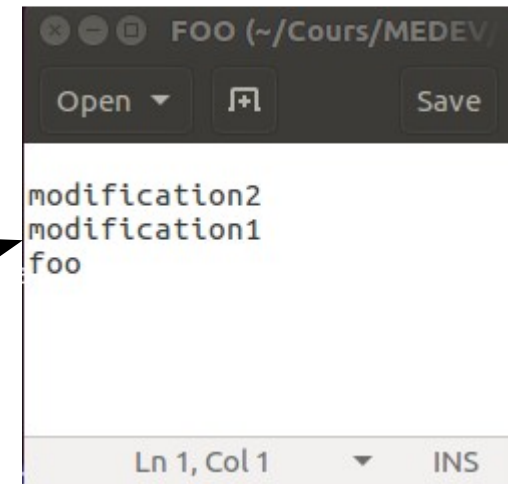
Notamment à cause de F00  
il faudra faire un update avant  
le commit

# Révision : *svn diff*

Montre les différences pour chaque ligne des fichiers dans le « working copy »

```
$svn diff
Index: F00
=====
--- F00      (revision 3775)
+++ F00      (working copy)
@@ -1,2 +1,4 @@
+
+modification2
+modification1
+foo
```

FOO (working copy)



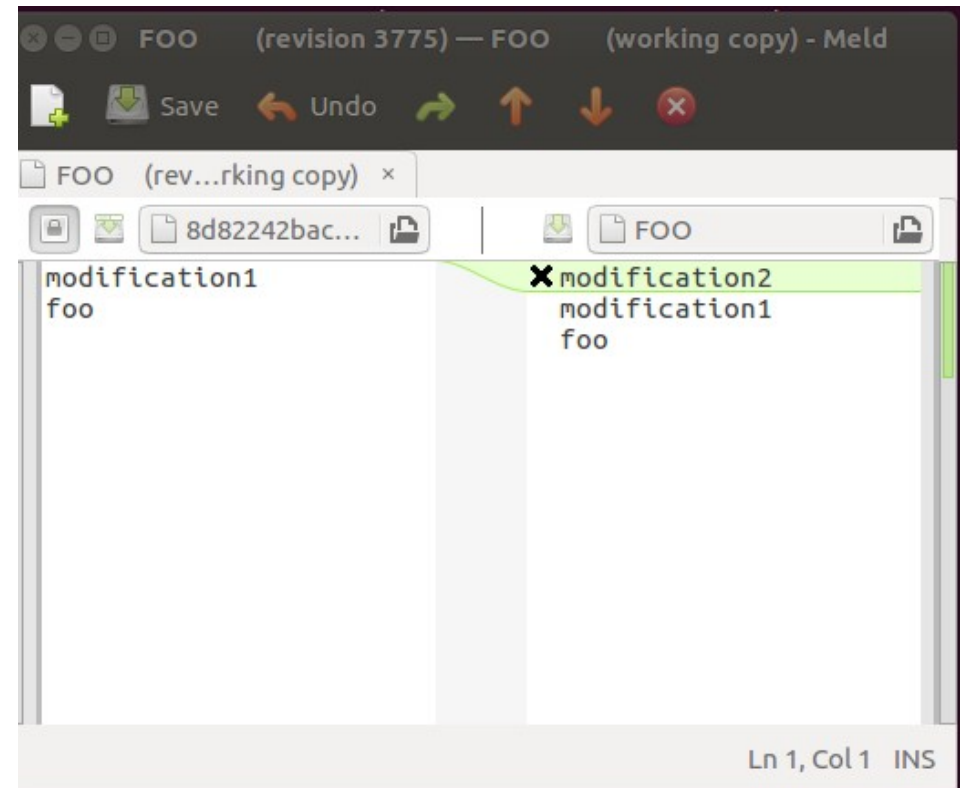
FOO (SVN)



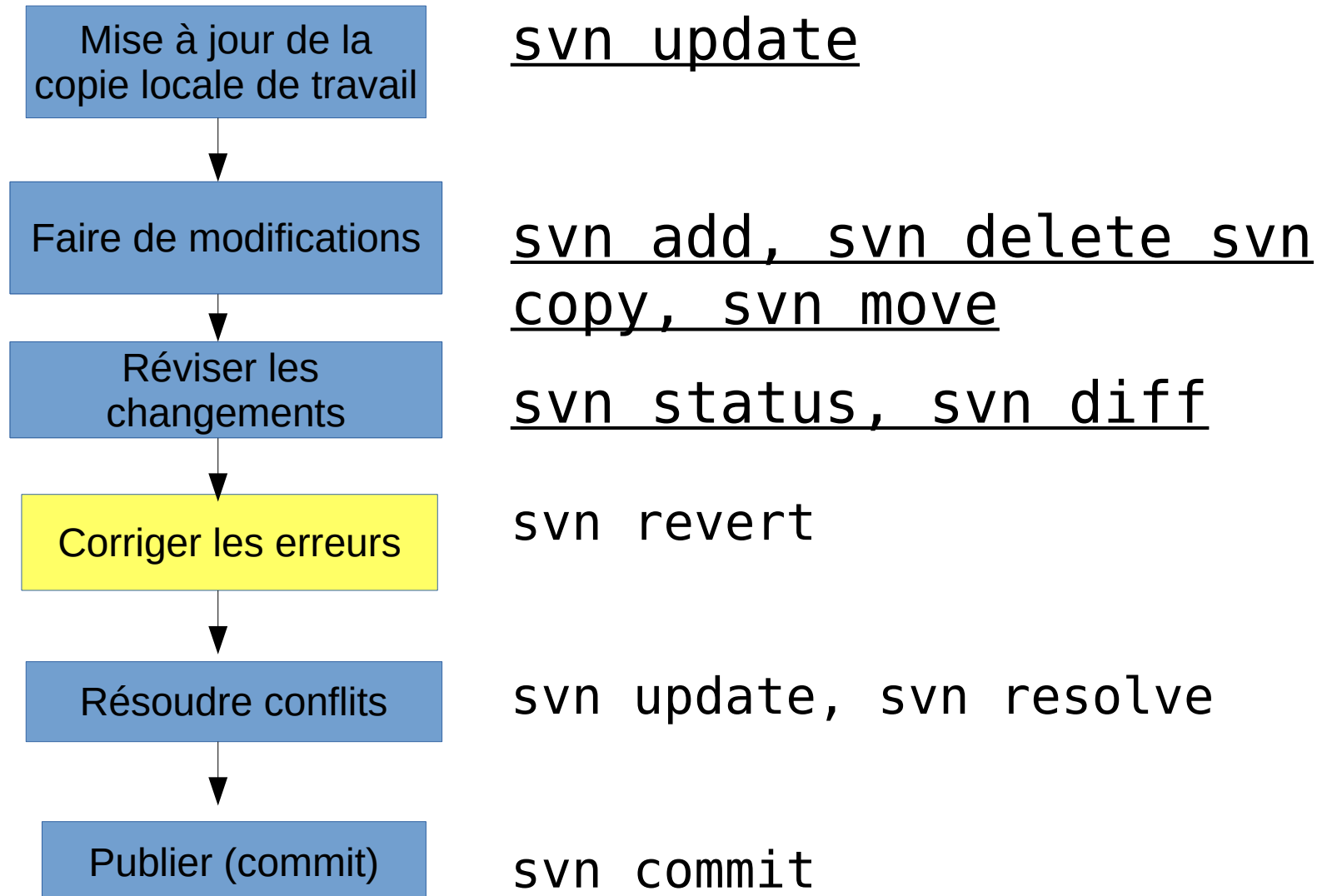
# Révision : *svn diff* (*interface graphique Meld*)

```
$sudo apt-get install meld ← Sous Ubuntu  
$svn diff --diff-cmd='meld' -r3775 F00
```

```
$svn diff  
Index: F00  
=====  
--- F00      (revision 3775)  
+++ F00      (working copy)  
@@ -1,2 +1,4 @@  
+  
+modification2  
 modification1  
 foo
```



# Cycle de Travail de base



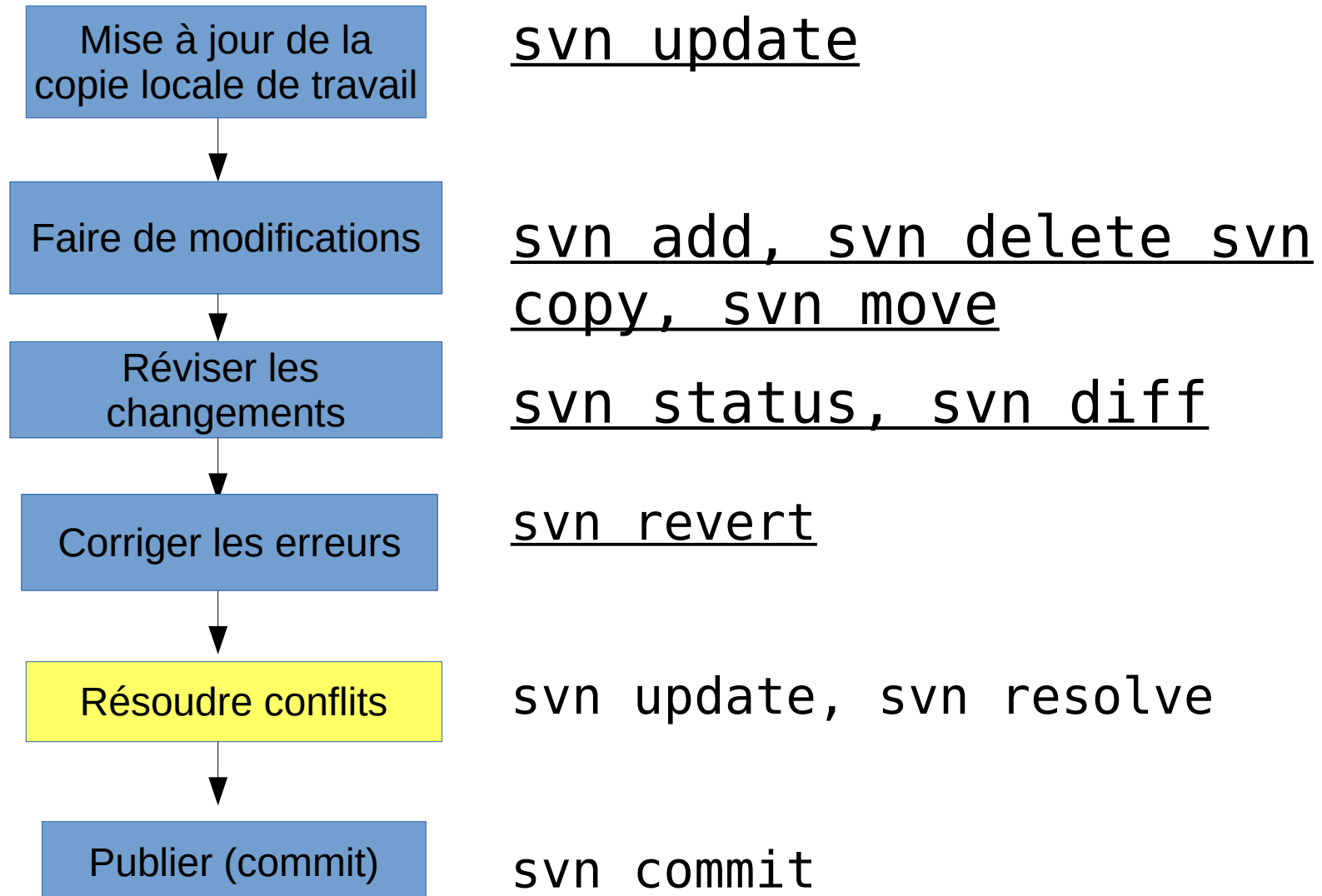
# Corriger les erreurs

Si jamais nous voulons revenir en arrière, **svn revert** récupère la version du fichier du *Repository* lors de notre dernière mise à jour

```
$cat F00
foo
$nano F00 ← fichier F00 est édité
$cat F00
modification 2
modification 1
foo
$svn revert F00
Reverted 'F00'
$cat F00
foo
$
```

L'instruction `svn revert` permet aussi d'effacer des ajouts (`svn add`) ou des suppressions (`svn delete`)

# Cycle de Travail de base



# Résoudre les conflits

Lors d'un update nous pouvons détecter des conflits avec la version que nous venons de modifier

Une autre personne  
a changé la même  
ligne du fichier

```
$svn update
Updating '.':
C    F00
Updated to revision 2812.
Summary of conflicts:
  Text conflicts: 1
Conflict discovered in file 'F00'.
Select: (p) postpone, (df) show diff, (e) edit file, (m) merge,
        (mc) my side of conflict, (tc) their side of conflict,
        (s) show all options:
```



# Options pour la résolution de conflits

(e) edit	Ouvrir le fichier avec l'éditeur dans la variable EDITOR
(df) diff-full	Montre les differences entre les 2 versions (locale et serveur) pour tout le fichier
(r ) resolved	Annoncer que le conflit a été resolu
(dc) display-conflict	Montrer les differences pour les zones du fichier en conflit
<u>(mc) mine-conflict</u>	(1) accepter tous les <i>merges</i> pour les zones sans conflit, et (2) rejeter la version dans le Repository pour la zone de conflits
<u>(tc) theirs-conflict</u>	(1) accepter tous les <i>merges</i> pour les zones sans conflit, et (2) rejeter la version dans la copie locale pour la zone de conflits
(mf) mine-full	Preserver seulement la version locale dans les zones sans et avec conflit du fichier
(tf) theirs-full	Preserver seulement la version du Repository dans les zones sans et avec conflit
(p) postpone	Laisser à plus tard la résolution du conflit
(l) launch	Executer un autre programme pour cette résolution du conflit
(s) show all	Lister toutes les instructions possibles à utiliser

# Résolution de conflits

Les instructions df et dc (pour la zone de conflit) permettent de visualiser les changements locaux vs serveur

```
Select: (p) postpone, (df) show diff, (e) edit file, (m) merge,  
        (r) mark resolved, (mc) my side of conflict,  
        (tc) their side of conflict, (s) show all options: df  
--- F00.r3781    - THEIRS  
+++ F00 - MERGED  
@@ -1,2 +1,8 @@  
+<<<<<<< .mine  
+test - user 1 completes line ← cette ligne est dans la copie locale  
+||||||| .r3780  
+test ← lors de la dernière mise à jour (r3780)  
+=====  
    user 2 changes this line (test)  
    user 2 adds this line  
+>>>>>>> .r3781 ← dans le Repository (r3781) la ligne a été éditée différemment
```

# Résolution de conflits

- Avec l'instruction (p) nous pouvons remettre à plus tard la résolution du conflit. Le fichier FOO aura un status « C » (svn status). La copie locale aura 3 fichiers temporaires pour FOO
  - FOO.mine : version du fichier avec mes changements
  - FOO.rOLDREV : version du fichier lors de ma dernière mise à jour (svn update). Un autre nom pour cette version : **base**
  - FOO.rNEWREV : version du fichier dans le Repository
- Il est impossible de publier les modifications sans résoudre les conflits avant

```
$svn commit F00 -m "test"  
svn: E155015: Commit failed (details follow):  
svn: E155015: Aborting commit:  
'/home/cguziolo/Cours/MEDEV/svn_ecn_medev/MEDEV/F00' remains in  
conflict
```

# Résolution de conflits : **svn resolve**

**svn resolve --accept ARG**. Les options possibles pour ARG:

- **base** : récupérer la version obtenue lors du mon dernière svn update
- **mine-full** : récupérer la version avec seulement mes modifications
- **theirs-full** : récupérer la version du fichier dans le Repository
- **working** : corriger à la main, l'état du fichier sera « resolved » et les 3 fichiers temporaires seront effacés

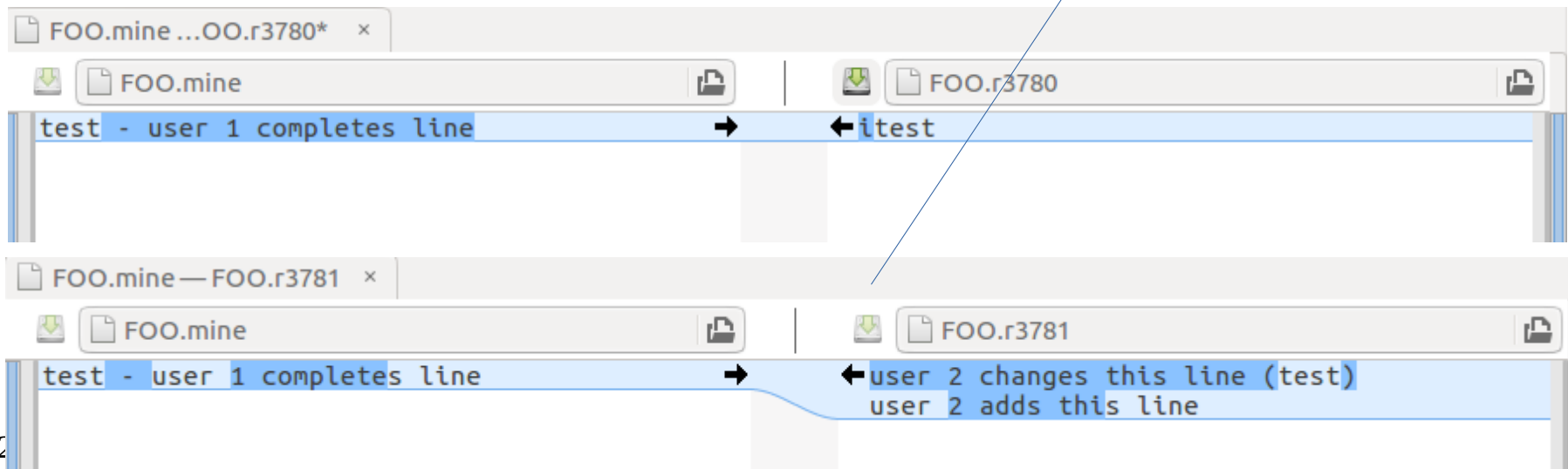
```
$ls
BAR  F00  F002  F003  F00.mine  F00.r2812  F00.r2813  test2.txt
test3.txt  test.txt
$svn status
C      F00
?      F00.mine
?      F00.r2812
?      F00.r2813
?      test3.txt
Summary of conflicts:
  Text conflicts: 1
$svn resolve --accept working F00
Resolved conflicted state of 'F00'
$svn status
M      F00
?      test3.txt
$ls
BAR  F00  F002  F003  test2.txt  test3.txt  test.txt
```

# Résolution de conflits : **svn resolve**

```
--- F00.r3781    - THEIRS
+++ F00 - MERGED
@@ -1,2 +1,8 @@
+<<<<<<< .mine
+test - user 1 completes line
+||||||| .r3780
+itest
+=====
+user 2 changes this line (test)
+user 2 adds this line
+>>>>>>> .r3781
```

meld FOO.mine FOO.r3780

meld FOO.mine FOO.r3781



# Résolution de conflits : édition manuelle

F00 (avant)

```
<<<<<< .mine  
test - user 1 completes line  
||||||| .r3780  
test  
=====  
user 2 changes this line  
(test)  
user 2 adds this line  
>>>>>> .r3781
```



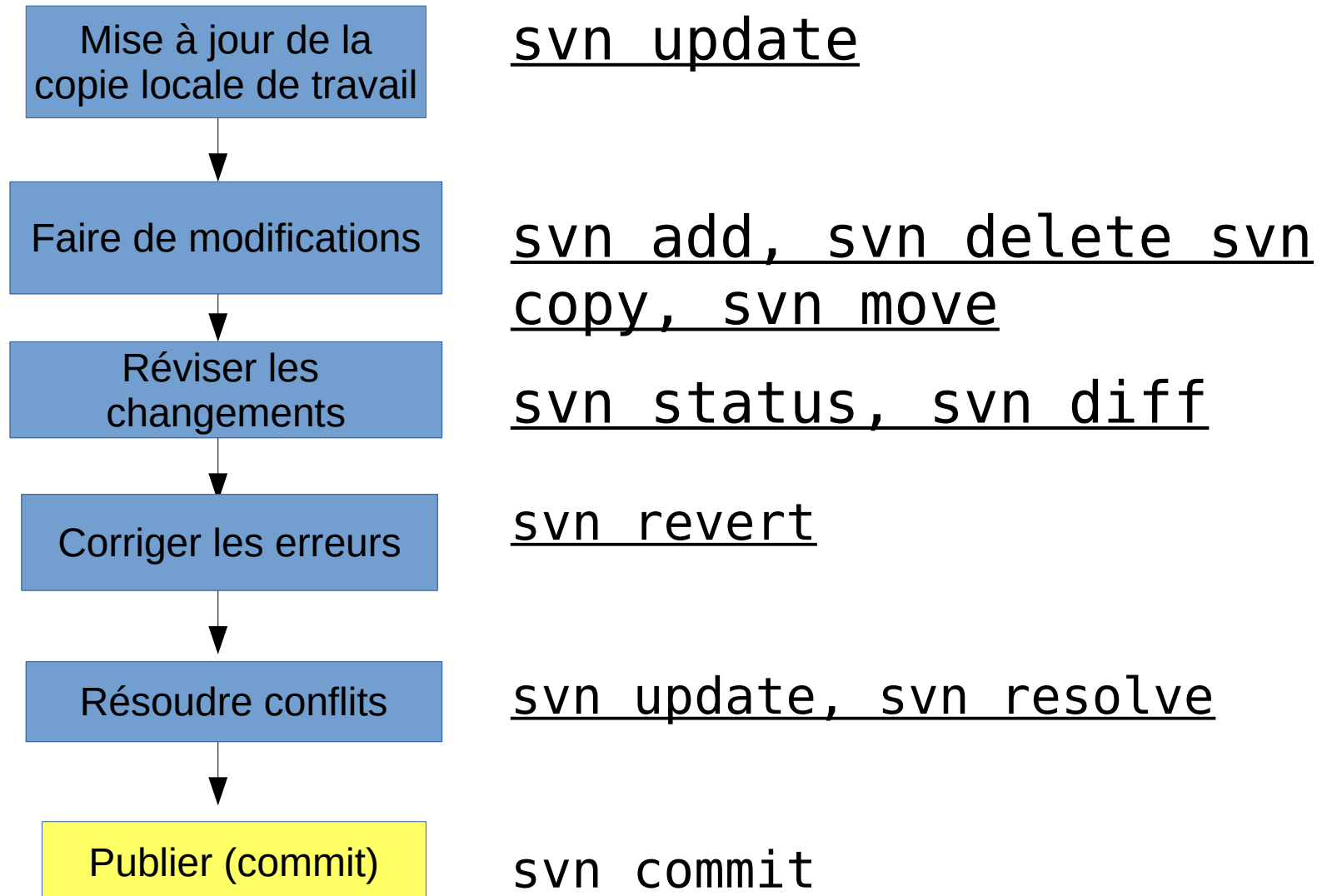
En coordination  
avec notre  
collaborateur

F00 (après)

```
test - user 1 completes line  
user 2 adds this line
```

Lorsque le fichier FOO ne contiendra plus des marqueurs « <<< », « |||| », « ===== », ou « >>>>> », le svn commit pourra être exécuté sans message d'erreur.

# Cycle de Travail de base



# Publier : `svn commit`

`svn commit -m « text »`

- Si *text* (*log* de la modification) n'est pas précisé alors un éditeur de texte s'ouvrira pour ajouter le message
- Le Repository n'a aucune notion d'exactitude ou fonctionnalité du code, ses erreurs concernent seulement le fait qu'une autre personne peut avoir changé la même ligne du code pendant qu'on ne regardait pas.

```
$svn commit -m "merged both versions"  
Sending          F00  
Transmitting file data .done  
Committing transaction...  
Committed revision 2817.  
$
```



# Historique : **svn log**

Décrit l'histoire des modifications d'un fichier ou dossier, en affichant la date, l'utilisateur modificateur, et le log

```
$svn log F00
-----
r2817 | cguziolo | 2017-11-15 11:03:22 +0100 (mer., 15 nov. 2017) | 1 line
merged both versions of line 5
-----
r2813 | cguziolo | 2017-11-14 18:42:06 +0100 (mar., 14 nov. 2017) | 1 line
line 5
-----
r2812 | cguziolo | 2017-11-14 18:00:40 +0100 (mar., 14 nov. 2017) | 1 line
line 5
-----
r2811 | cguziolo | 2017-11-14 17:59:33 +0100 (mar., 14 nov. 2017) | 1 line
my change
-----
r2810 | cguziolo | 2017-11-14 17:57:52 +0100 (mar., 14 nov. 2017) | 1 line
new line 3
-----
r2809 | cguziolo | 2017-11-14 17:53:53 +0100 (mar., 14 nov. 2017) | 1 line
new line 1
-----
r2808 | cguziolo | 2017-11-14 17:53:05 +0100 (mar., 14 nov. 2017) | 1 line
...

```

# Historique : **svn log**

<code>svn log -r N1 : N2</code>	Montre les logs pour les revisions de N1 à N2
<code>svn log -r N2 : N1</code>	Montre les logs pour les revisions de N2 à N1 (ordre décroissant)
<code>svn log -r N1</code>	Montre les logs pour seulement la revision N1

```
$svn log -r r2790
-----
r2790 | cguziolo | 2017-11-13 16:45:46 +0100 (lun., 13 nov. 2017) | 1 line
init
-----
```

# Plan

Introduction SVN

Modèles de partage

SVN en action

Cycle de travail de base

[Recommandations](#)

SVN pour Windows

# Recommandation pour organiser vos projets informatiques

- Chaque projet du développement contient un dossier **root** (qui est associé à un Repository)
- Chaque **root** contient 3 dossiers
  - **trunk**, pour le développement principale
  - **branches**, pour stocker copies divergents de la version du projet qui est en *trunk*
  - **tags**, avec de versions stables d'une ligne du développement

# Plan

Introduction SVN

Modèles de partage

SVN en action

Cycle de travail de base

Recommandations

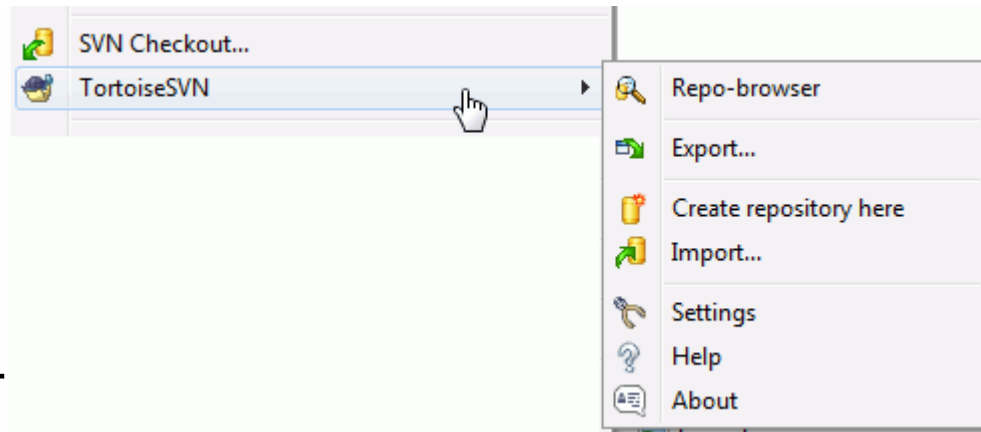
[SVN pour Windows](#)

# SVN pour Windows

Client svn TortoiseSVN

[https://tortoisesvn.net/docs/nightly/TortoiseSVN\\_en/](https://tortoisesvn.net/docs/nightly/TortoiseSVN_en/)

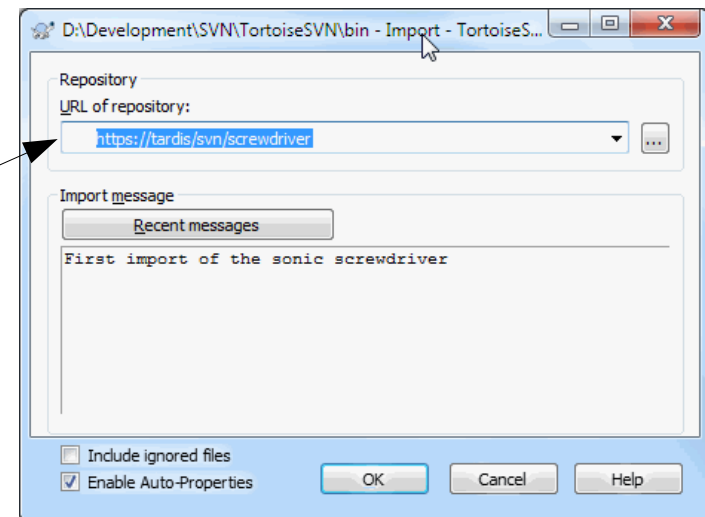
- Une fois installé, pour le lancer il faut faire click-droit sur un fichier quelconque



## Créer un Repository

- Créer un dossier
- Click-droit sur le dossier
  - TortoiseSVN → Create Repository here...
- Tortoise SVN → Import..

Ecrire l'adresse du  
svn



# Quelques Références

- Version Control with Subversion
  - Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. 2009. Version Control with Subversion - the Official Guide and Reference Manual. CreateSpace, Paramount, CA.
  - Version Control with Subversion. For Subversion 1.7. Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato <http://svnbook.red-bean.com/en/1.7/svn-book.html>
  - <http://svnbook.red-bean.com/> ← plusieurs formats de documentation, pour plusieurs version de svn