

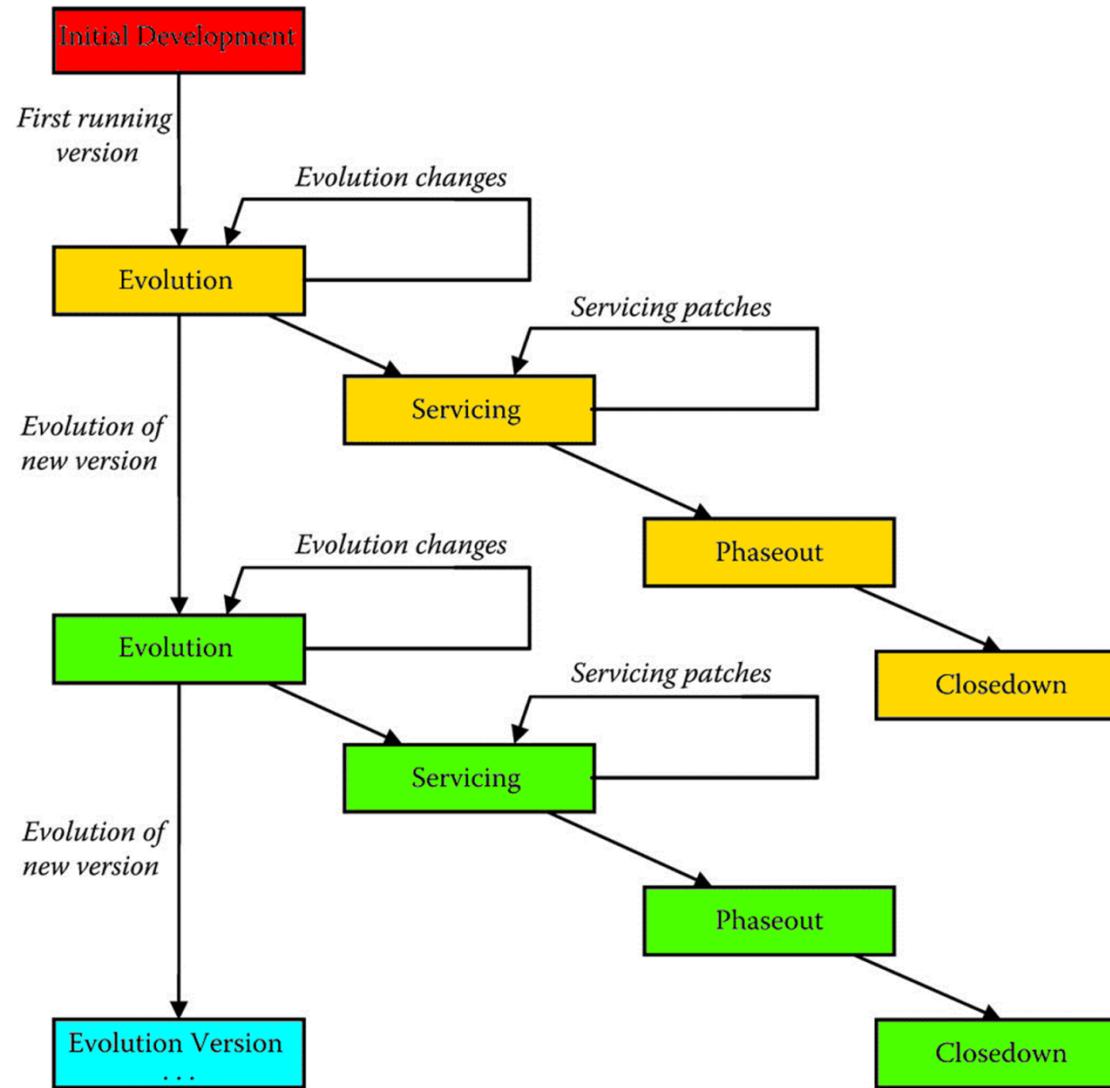
IFT 2255 - Genie Logiciel

Contrôle de versions du logiciel

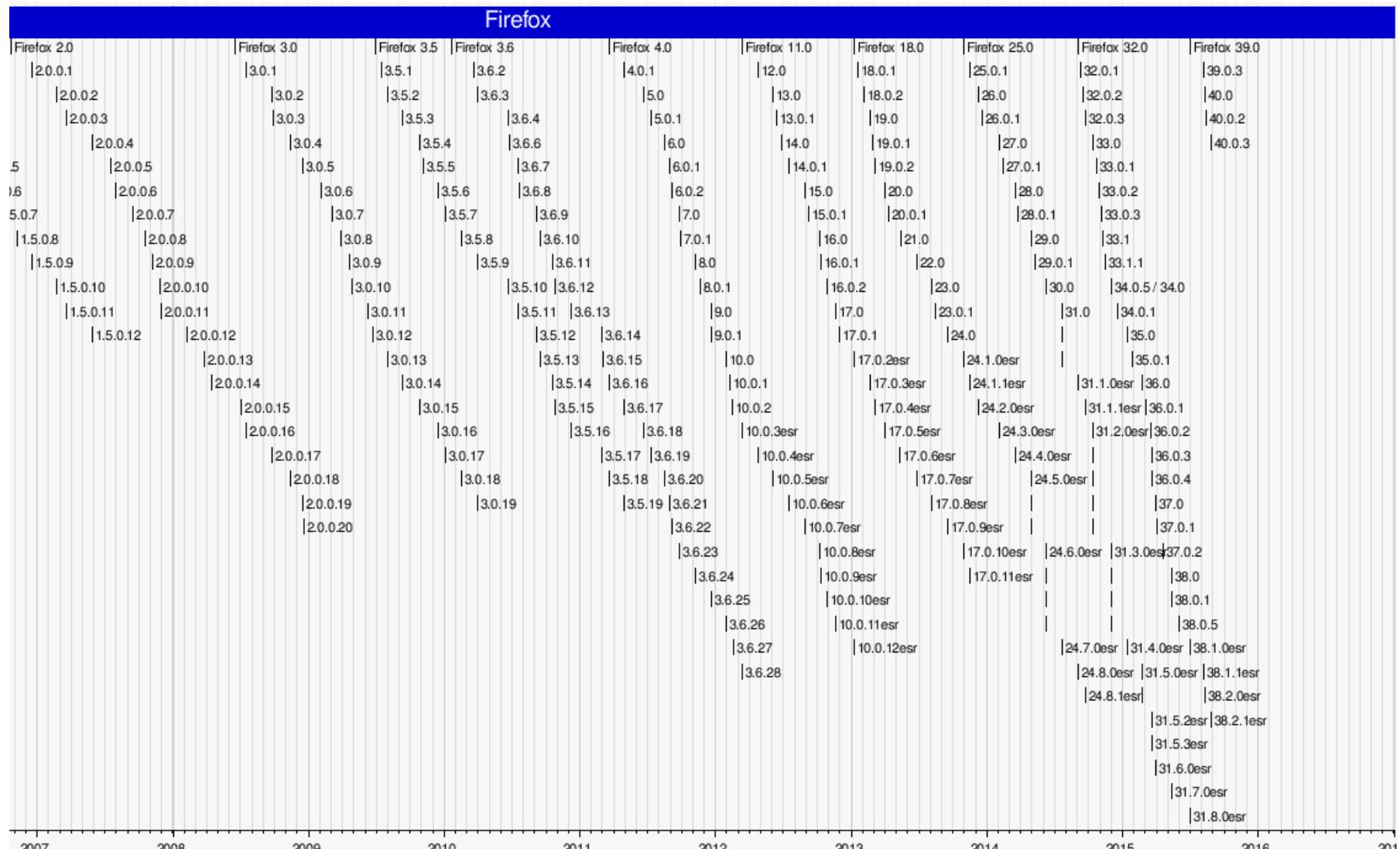
Versions du logiciel

- Plus d'un programmeur dans le projet ⇒ **collaboration**
 - Travailler en **parallèle**
 - Code de chacun doit être **consistant** avec le reste
- Besoin de **sauvegarde** sécuritaire du code source (*back-up*)
- Besoin de garder une **trace** des changements dans le code source
 - Besoin de revenir à une version antérieur de mon propre code
- Développement incrémental et itératif: plusieurs **versions** du code
 - Ex: Tests faits sur itération N, pendant que programmeur travail sur itération N+1
 - Documentation, artefacts de conception, spécifications, etc.

Évolution est un pilier du logiciel



Historique des versions de Firefox



Gestion de version à la main

"FINAL".doc



FINAL.doc!



FINAL_rev.2.doc



FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



FINAL_rev.18.comments7.
corrections9.MORE.30.doc FINAL_rev.22.comments49.
corrections.10.#@\$%WHYDID
ICOMETOGRAD SCHOOL????.doc

Return to Zero

WHAT'S THIS CUP OF USB DRIVES?

AH, THAT'S HOW I
VERSION-CONTROL
MY FILES...



IT'S BRILLIANT
REALLY. JUST
MAKE A NEW
COPY ON A
NEW ONE
EVERYDAY.



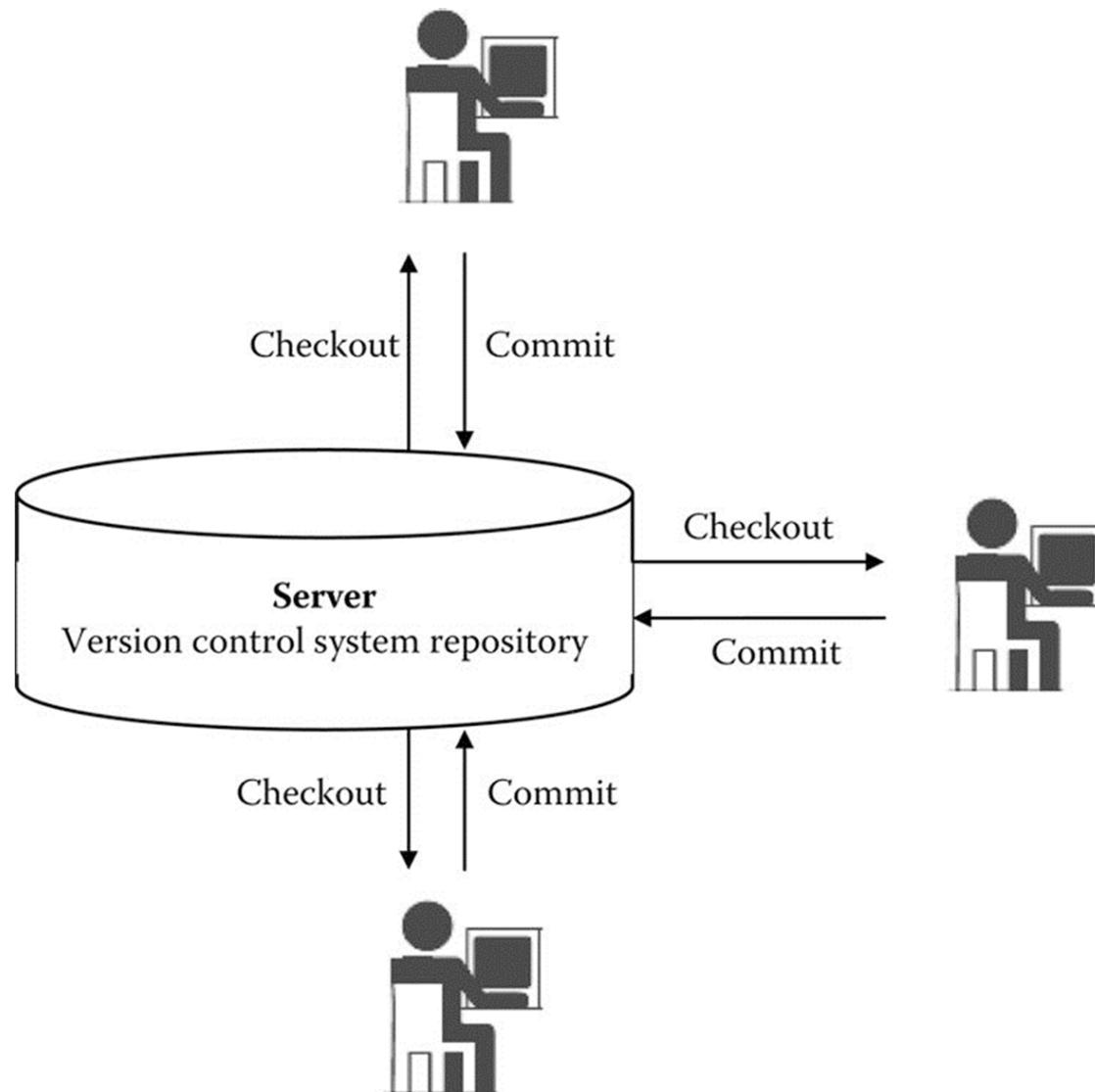
Système de contrôle de révisions

- Gérer les **changements** apportés à des fichiers
- Conserver une **trace**
 - Quels sont les changements?
 - Qui a fait quel changement ?
 - Voir l'arbre d'évolution complet
- **Collaborer** entre individus
 - Partager le travail dans une hiérarchie de fichiers
 - Synchronisation garantie
- **Protéger**
 - Contre la perte de données, documents et code source
 - Ignorer un changement et revenir à une version antérieure ou ultérieure

Contrôler les versions de quoi ?

- Fichier texte (source?)
 - Version de changement ligne par ligne
 - Conserve uniquement les changements
 - Visualisation des différences entre versions
- Fichier binaire (compilé?)
 - Granularité au niveau du fichier
 - Chaque version conserve le fichier au complet

Système de contrôle de révisions centralisé



Dépôt (*Repository*)

- Conserve tout le contenu associé à un projet
 - Sous forme de base de donnée
- Contient tout l'historique du projet
 - Contenu initial et modifié
 - Auteur des modifications
 - Date/heure d'une modification
- Dépôt centralisé
 - Le dépôt est sur le serveur en forme canonique
- Dépôt décentralisé
 - Chaque client a sa propre copie du dépôt provenant du serveur

Copie de travail (*working copy*)

- Programmeur qui veut apporter des modifications doit obtenir une copie de travail
 - Opération **check out** ou **clone** du dépôt pour la première fois
 - Opération **update** ou **pull** du dépôt pour avoir les nouveaux changements
- Programmeur apporte des modifications sur sa copie de travail, pas directement sur le dépôt
- Protège le contenu du dépôt jusqu'à créer une nouvelle révision

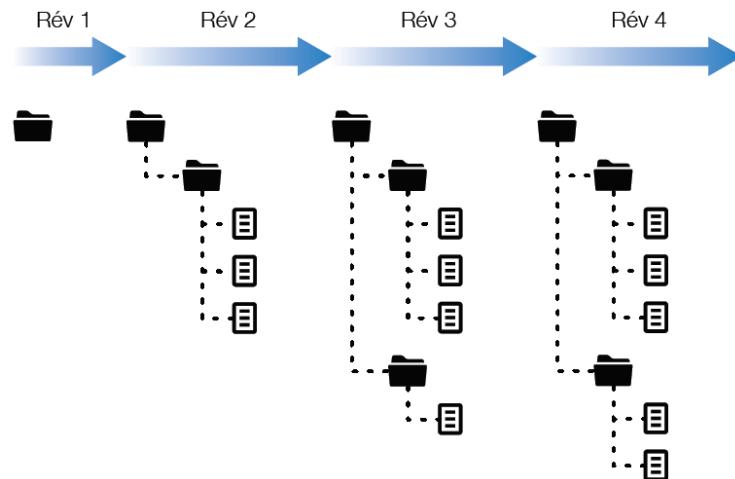
Suivi des changements (*track change*)

- Système fait le suivi des changements apportés à une copie de travail en arrière-plan
 - Possible à tout moment d'annuler les modifications apportées sur la copie de travail d'un ou plusieurs fichiers
- Changements suivis
 - Ajout, suppression, modification, déplacement, renommage un fichier ou dossier, etc.
- Processus transparent jusqu'à ce que les changements soient prêts à former la prochaine version

Soumission (*commit*)

- Lorsque les modifications sont complétées et **enregistrés en local** sur la copie de travail, le client soumet ses changements au dépôt
- Soumission d'un ou plusieurs changements dans une seule **transaction atomique**
- Une soumission provoque la création d'une nouvelle révision pour l'ensemble des changements
 - Calcule les **différences** entre la version de chaque fichier dans la copie locale avec celle du dépôt
 - Applique les changements à la nouvelle révision du dépôt
- La nouvelle révision devient la révision courante **HEAD**

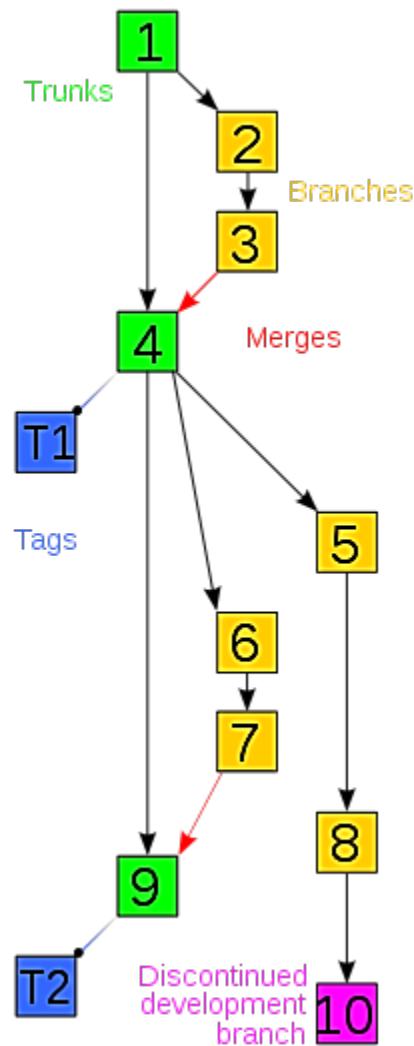
Création de révisions



- Une nouvelle révision obtient un **identifiant unique** qui permet d'y accéder et d'y revenir dans le futur
 - Numéro séquentiel (1, 2, 3, ...) ou code de hachage (5d368c0635de...)
- Majorité des systèmes supportent **l'étiquetage** des révisions
 - Nom plus parlant
- Lors de la soumission, le client peut ajouter un **message descriptif** du changement

Tronc, branches et tags

- Tronc (*trunk*)
 - Branche principale du contenu
 - Dernière version en cours de développement
- Branches
 - Copies du tronc pour une révision expérimentale qui risque de briser le tronc
 - Changements peuvent y être soumis pendant que d'autres changements sont soumis au tronc
 - Version en développement encore instable
- Tags
 - Branches en lecture seule, images du projet à un moment donné
 - Version stable d'un livrable
 - Autre mécanisme d'étiquetage



Fusion de fichiers (*merge*)

- Système connaît les changements apportés à chacun des fichiers depuis leur dernier ancêtre commun
- Fusion automatique quand il n'y a pas de conflits
- En cas de conflit, un humain doit décider de la fusion manuellement
 - Aidé par une vue de différences pour faciliter la fusion manuelle

Différences

The screenshot shows the TortoiseMerge application interface. The title bar reads "diffDemo.txt - TortoiseMerge". The menu bar includes File, Edit, Navigate, View, and Help. The toolbar contains various icons for file operations like Open, Save, and Diff.

The left pane is labeled "diffDemo.txt : Working Base, Revision 1331" and the right pane is labeled "diffDemo.txt : Working Copy". Both panes show the same text content:

```
1 This is my text file.  
2  
3 I have added a little bit of text to this file.  
4  
5 Here is some more text.  
6  
7 You don't just have to commit code to a repository!
```

Line 1 has a yellow background in the base pane and a blue background in the copy pane, indicating it is deleted in the copy. Line 3 has a yellow background in both panes, indicating it is modified. Line 5 has a yellow background in the base pane and a blue background in the copy pane, indicating it is inserted in the copy. Line 6 has a blue background in the base pane and a yellow background in the copy pane, indicating it is deleted in the base.

The screenshot shows the Java Source Compare feature. The left pane is labeled "Local: RuleShape.java" and the right pane is labeled "Index: RuleShape.java (editable)".

The Local version of RuleShape.java contains the following code:

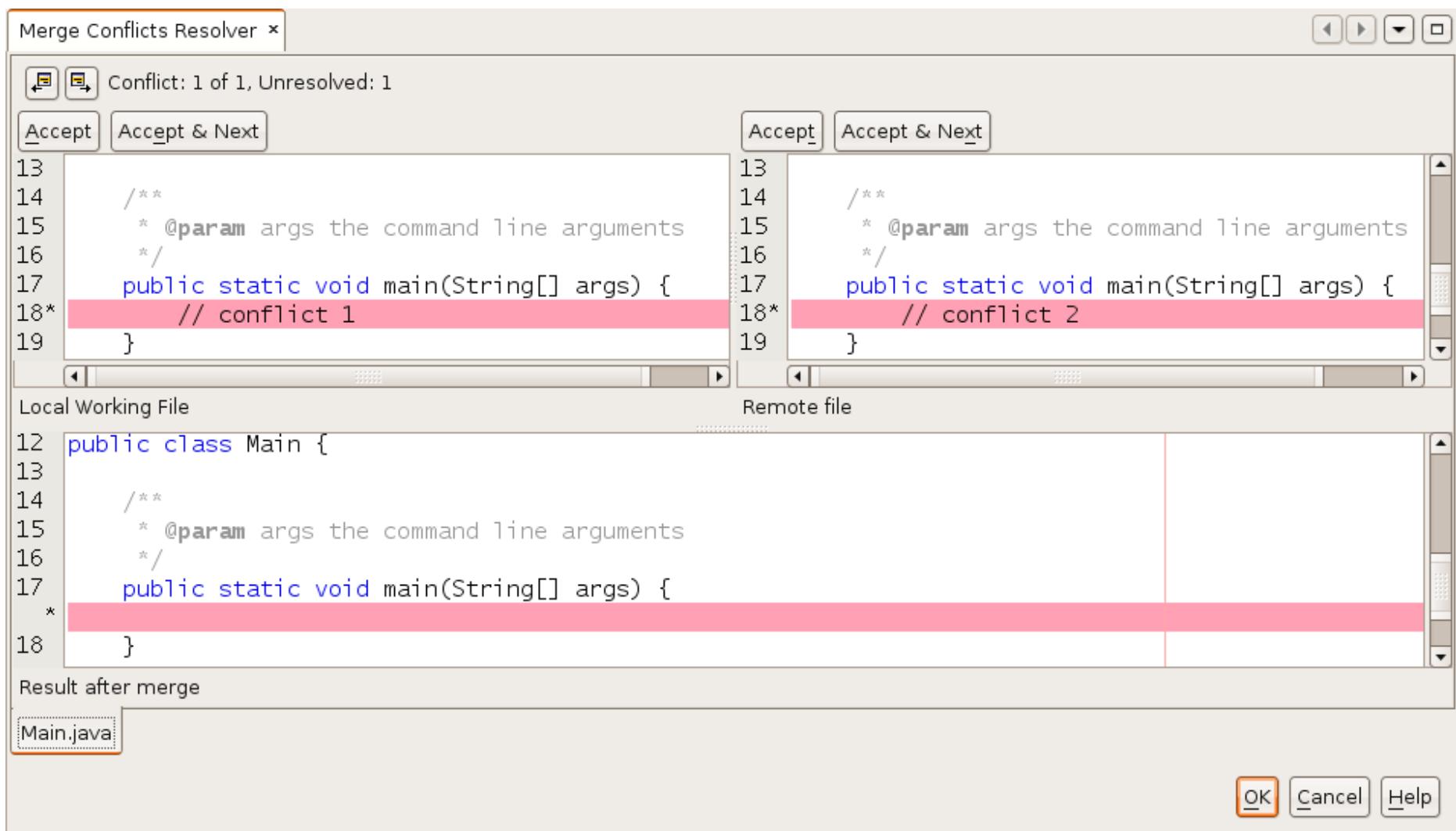
```
24     int h = rect.height;  
25  
26     int offset = 0;  
27  
28     // Rule is useless for now, remove in the next iteration  
29     Rule rule = (Rule) ((mxCell) state.getCell()).getValue();  
30  
31     Rule newRule = (Rule) ((mxCell) state.getCell()).getEdgeAt(0);  
32  
33     return new Rectangle2D.Float(x + 10 + offset, y + 10 + offset,  
34         - offset, h - 10 - offset);  
35  
36 }
```

The Index version of RuleShape.java contains the following code:

```
24     int h = rect.height;  
25  
26     int offset = 0;  
27  
28     Rule rule = (Rule) ((mxCell) state.getCell()).getValue();  
29  
30     if (rule.getBooleanAttr("isExhaustive"))  
31         offset = 10;  
32  
33     return new Rectangle2D.Float(x + 10 + offset, y + 10 + offset,  
34         - offset, h - 10 - offset);  
35  
36 }  
37  
38 @Override  
39 public void paintShape(mxGraphics2DCanvas canvas, mxCellState stat  
40     super.paintShape(canvas, state);
```

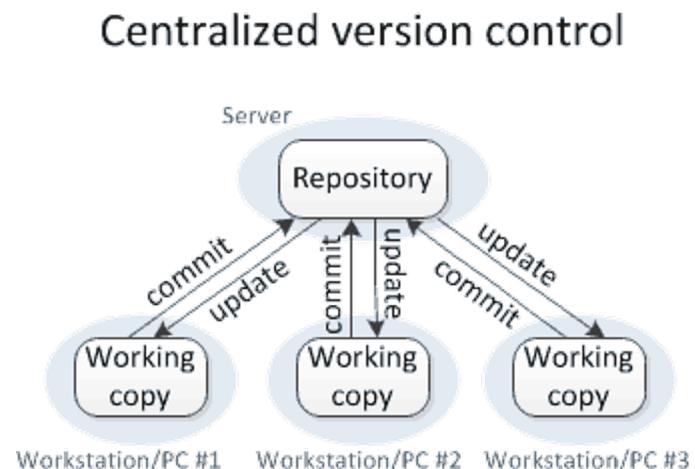
A diff highlighter shows differences between the two versions. Lines 28-31 are highlighted in blue, indicating they are inserted in the index version. Line 30 has a yellow background, indicating it is modified.

Résolution de conflits

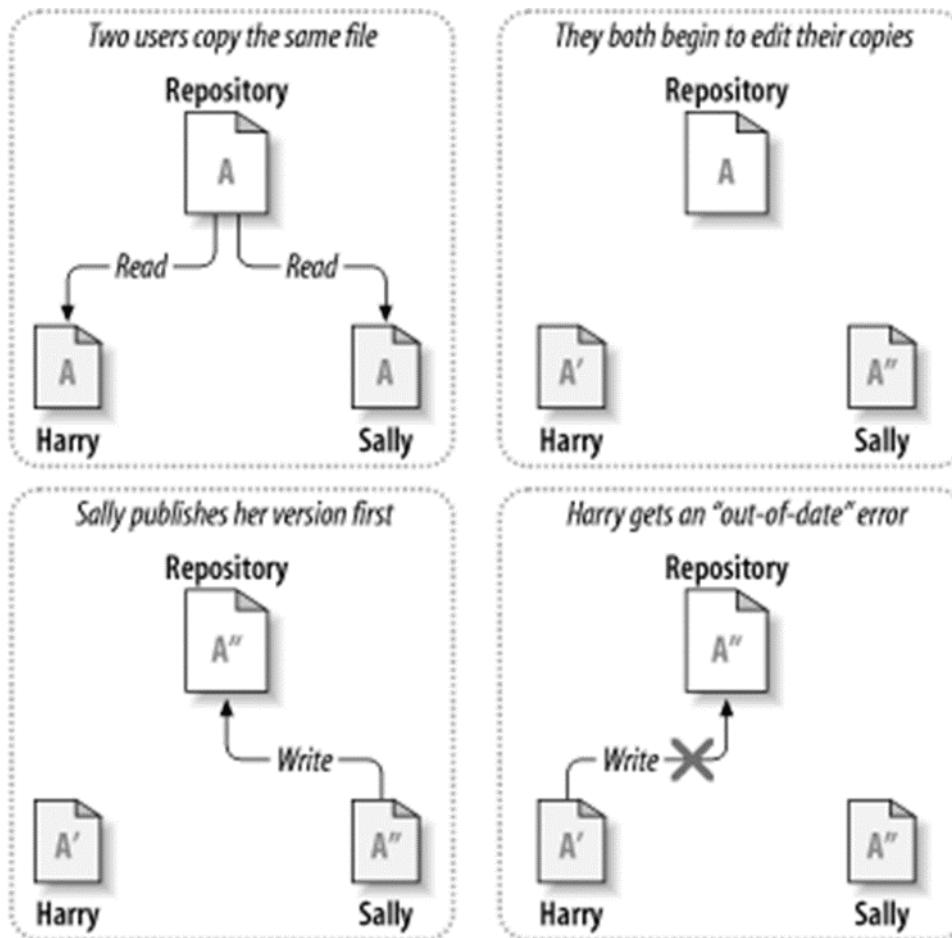


Mise à jour

- Des changements importants peuvent avoir été apportés au tronc pendant qu'un développeur travaille sur sa branche
- Pour importer les derniers changements, on fait une **mise à jour**
 - Faites souvent une mise à jour pour éviter des conflits lors de la fusion
- La mise à jour copie les fichiers du dépôt en local
 - Programmeurs changent la copie locale
 - Protège les données dans le dépôt jusqu'à la prochaine fusion
- **Checkout/clone:** première fois
- **Update/pull:** toutes les autres fois

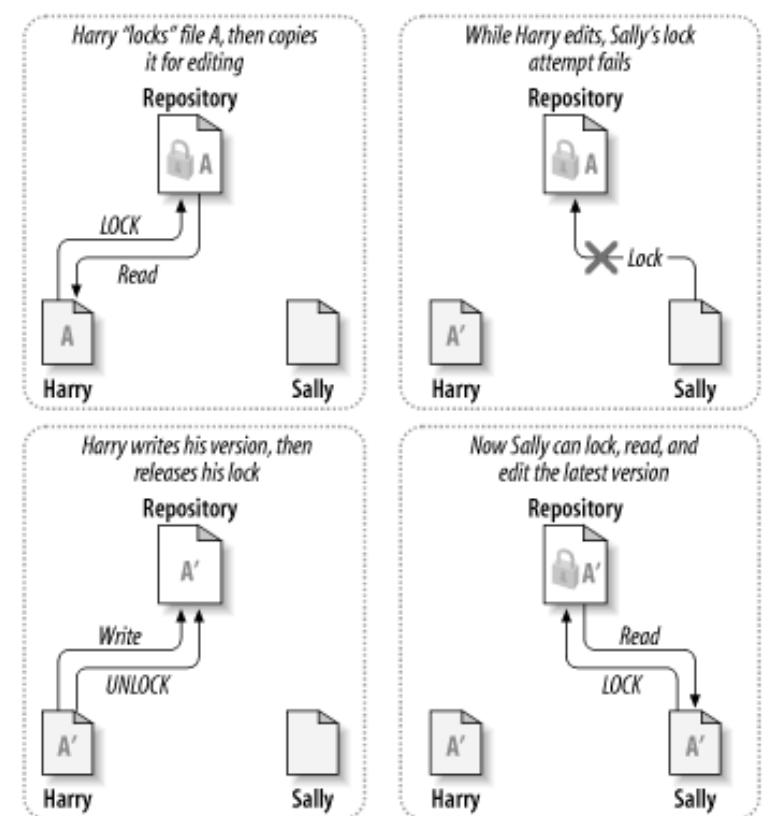


Scénario de collaboration



Solution: verrouiller (*lock*)

- Un fichier verrouillé ne peut être modifié que par un client à la fois
- Les autres doivent attendre qu'il soit déverrouillé
- Soumettre un fichier verrouillé le déverrouille automatiquement
- Limitations
 - Changements concurrents sur des fichiers différents seulement
 - Oubli de déverrouiller un fichier
 - Deux clients verrouillent deux fichiers interdépendants

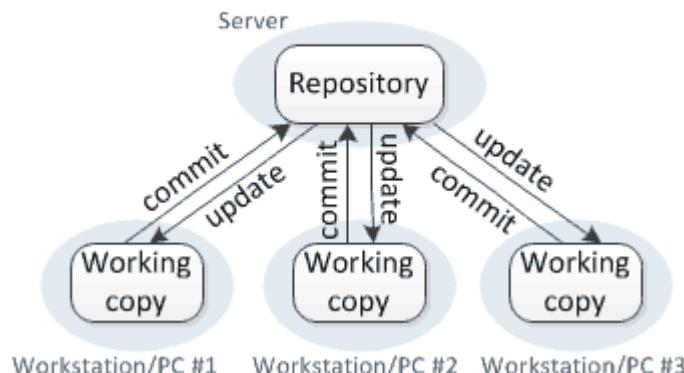


Systèmes de gestion de révisions

Centralisé

SVN, CVS

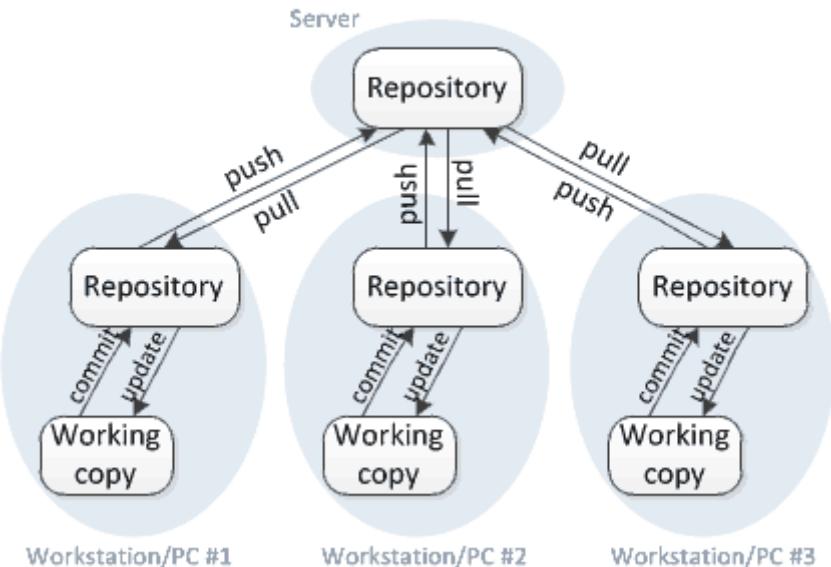
Centralized version control



Décentralisé

GIT, Mercurial

Distributed version control



Subversion (SVN)

- Système centralisé le plus populaire, depuis 2000
- Support natif pour les fichiers binaires
- Modèle simple, facile à apprendre et à utiliser
- Doit être connecté au serveur de dépôt pour soumettre
- Opérations peuvent être lentes
- Branches et fusions difficiles à utiliser
- Outils par ligne de commande ou graphique (TortoiseSVN)



GIT

- Système décentralisé le plus populaire, depuis 2005
- Serveur optionnel
- Opérations rapides car locales
- Branches efficaces utilisées fréquemment
- Modèle distribué plus complexe
- Beaucoup de fusion, donc plus de possibilités de conflits
- Pas d'outil graphique, seulement par ligne de commande



TortoiseSVN

