Systèmes de Gestion de Version

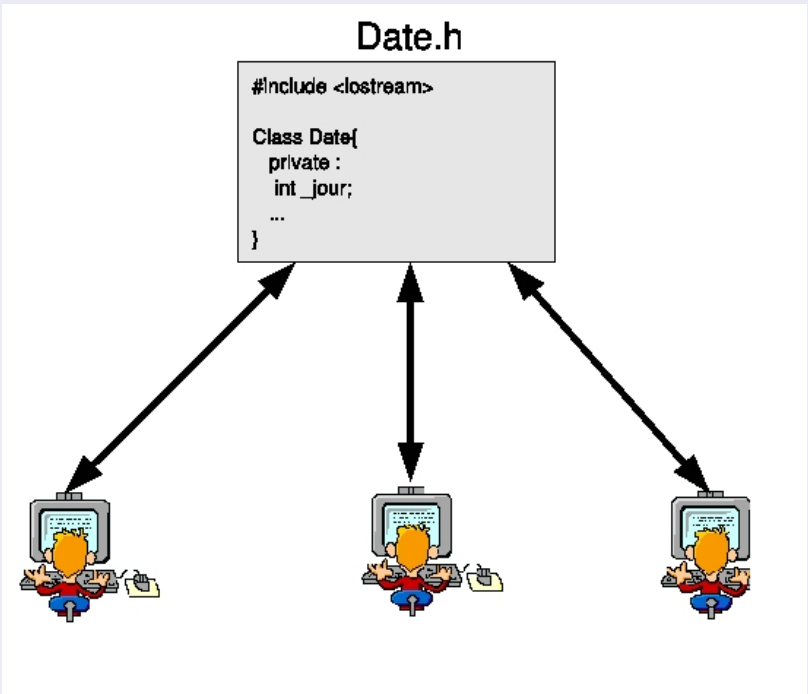
F. Langrognet

PLAN

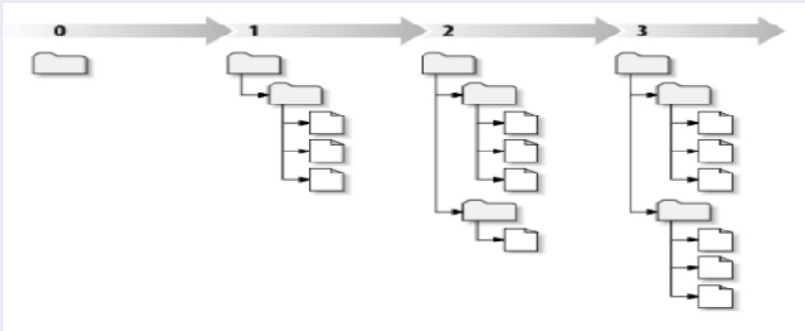
PLAN

Objectifs d'un Système de Gestion de Version

Travailler à plusieurs



Conserver l'historique



- Pouvoir revenir en arrière
- Qui a modifié pour la dernière fois ce fichier ?
- Quelles sont les différences entre 2 versions de ce fichier ?
- Quelle est la version du 15 mars 2007 ?

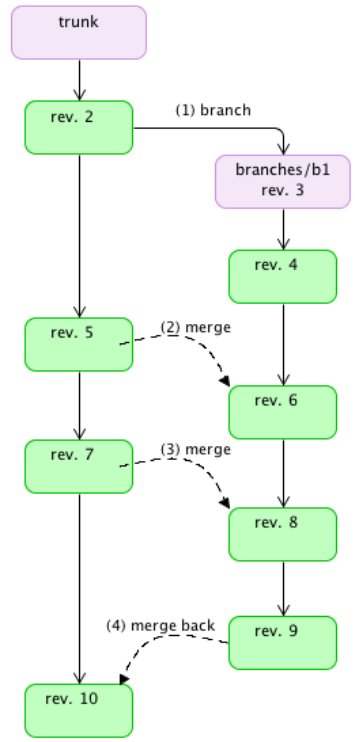
Objectifs d'un Système de Gestion de Version (suite)

Et aussi ...

- Gestion des branches
  Objectif : mener en parallèle plusieurs versions (stable, testing, ...)

- Utilisation de *tags*
  Objectif : donner un nom explicite à une version pour pouvoir y accéder facilement

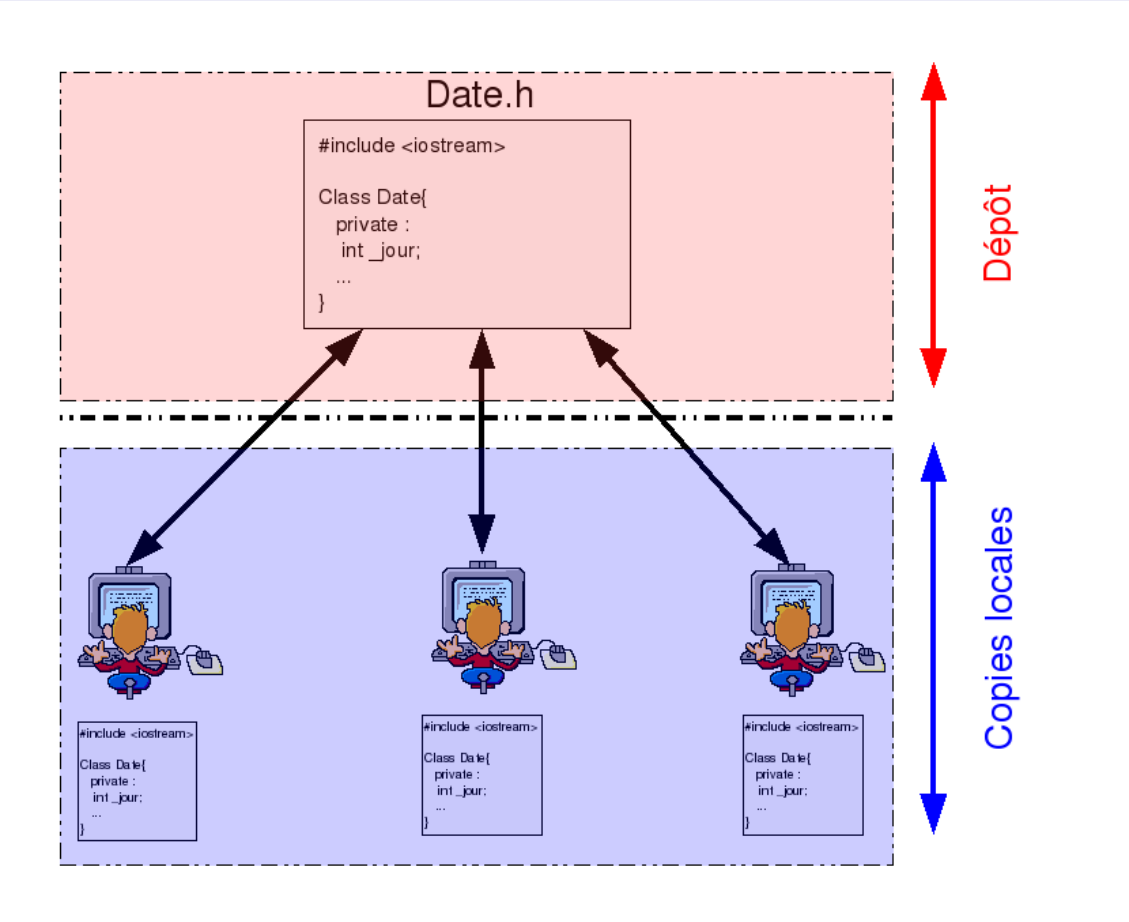
- Sécurité
  Intégrité, Disponibilité, Confidentialité



PLAN

Principe de base

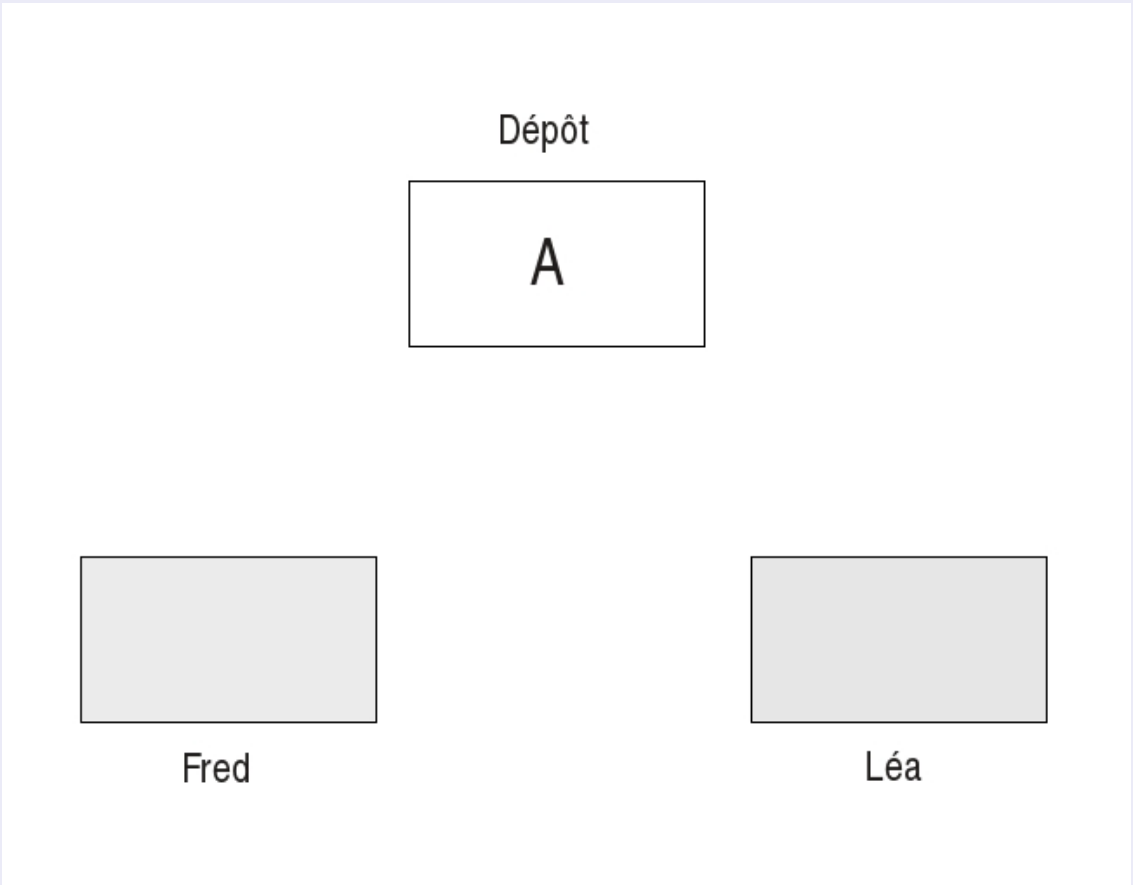
Notion de *dépôt* et *copie locale*



Les accès (écriture/lecture) se font via le système de gestion de version

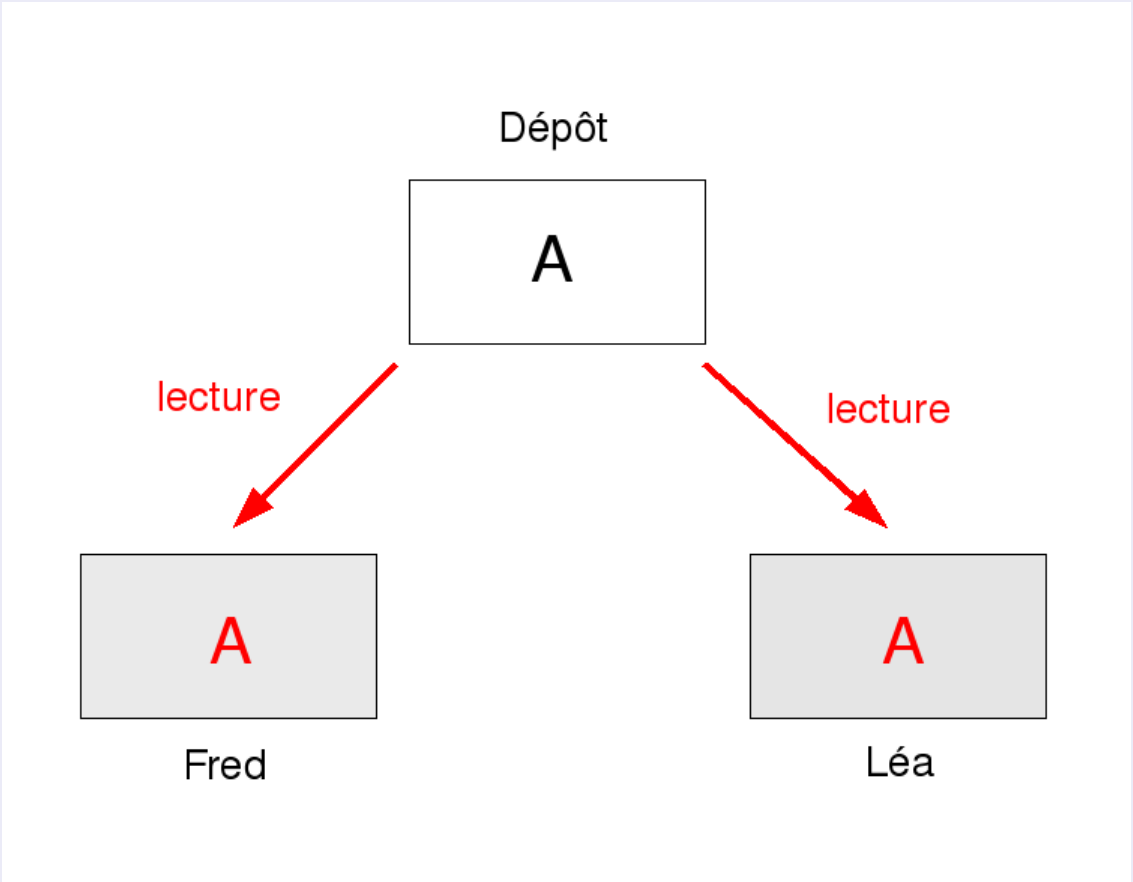
Le problème . . .

Problème



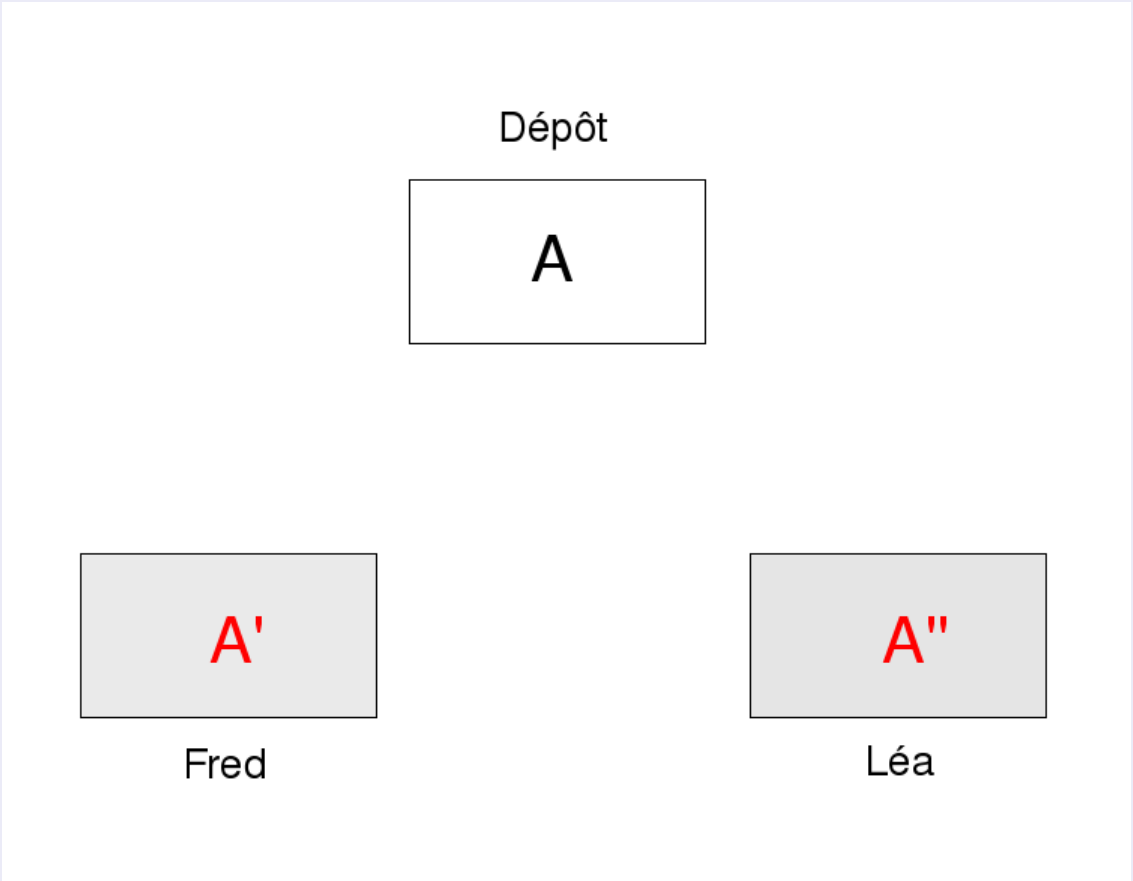
Fred et Léa veulent accèder au même fichier

Problème



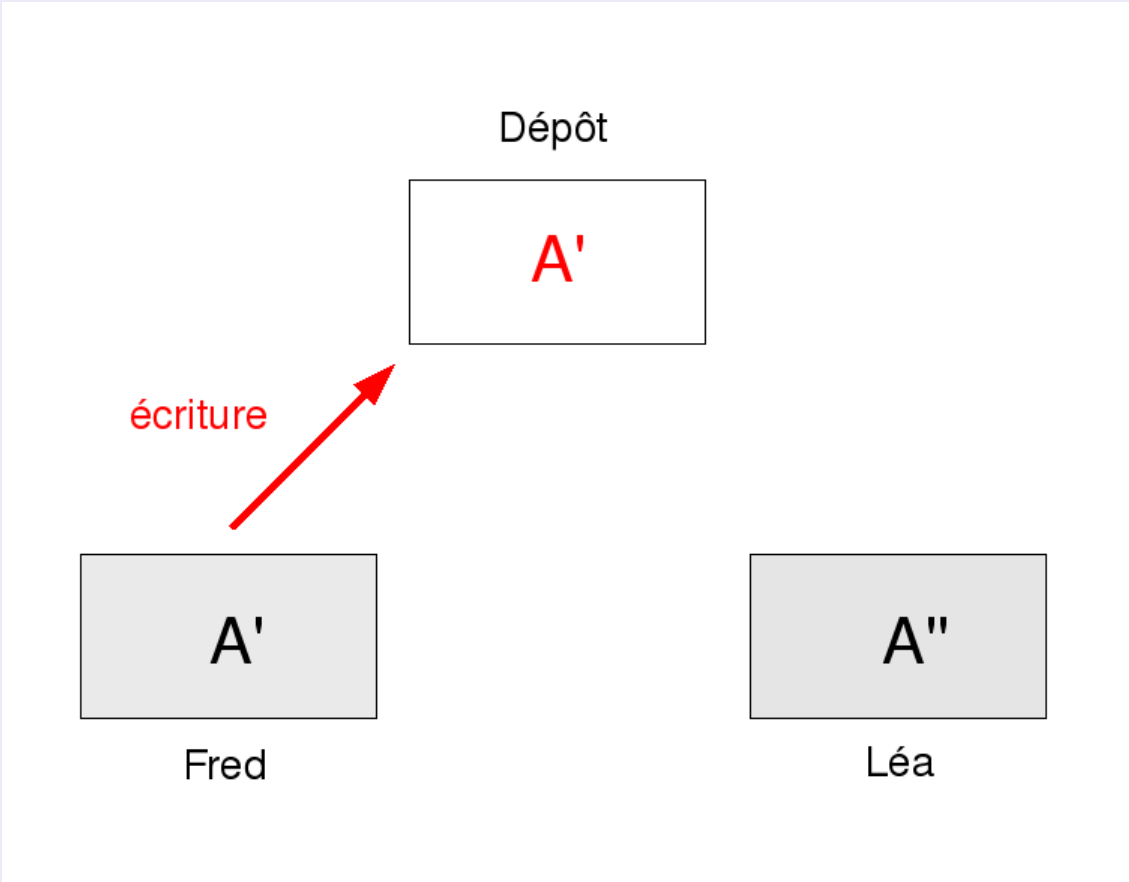
Fred et Léa accèdent au même fichier et le copient chez eux

Problème (suite)



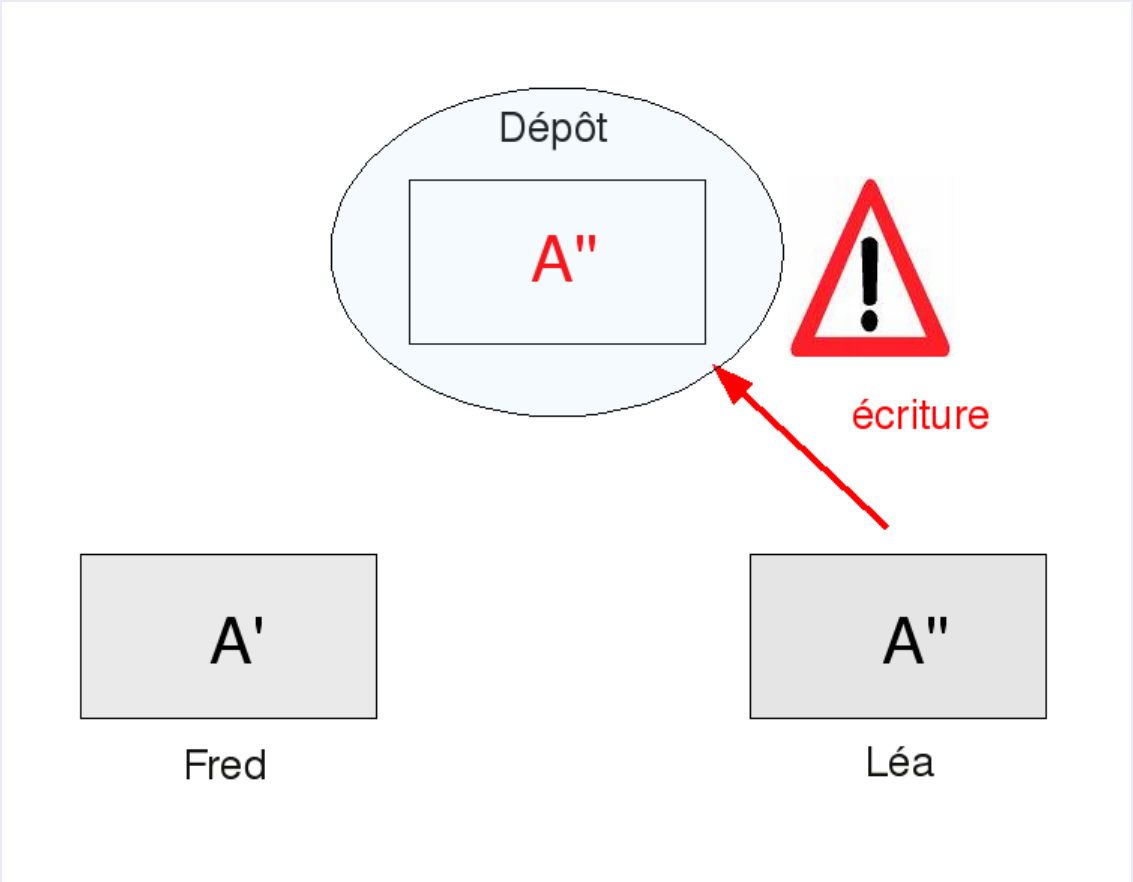
Fred et Léa font chacun des modifications

Problème (suite)



Fred écrit sur le dépôt

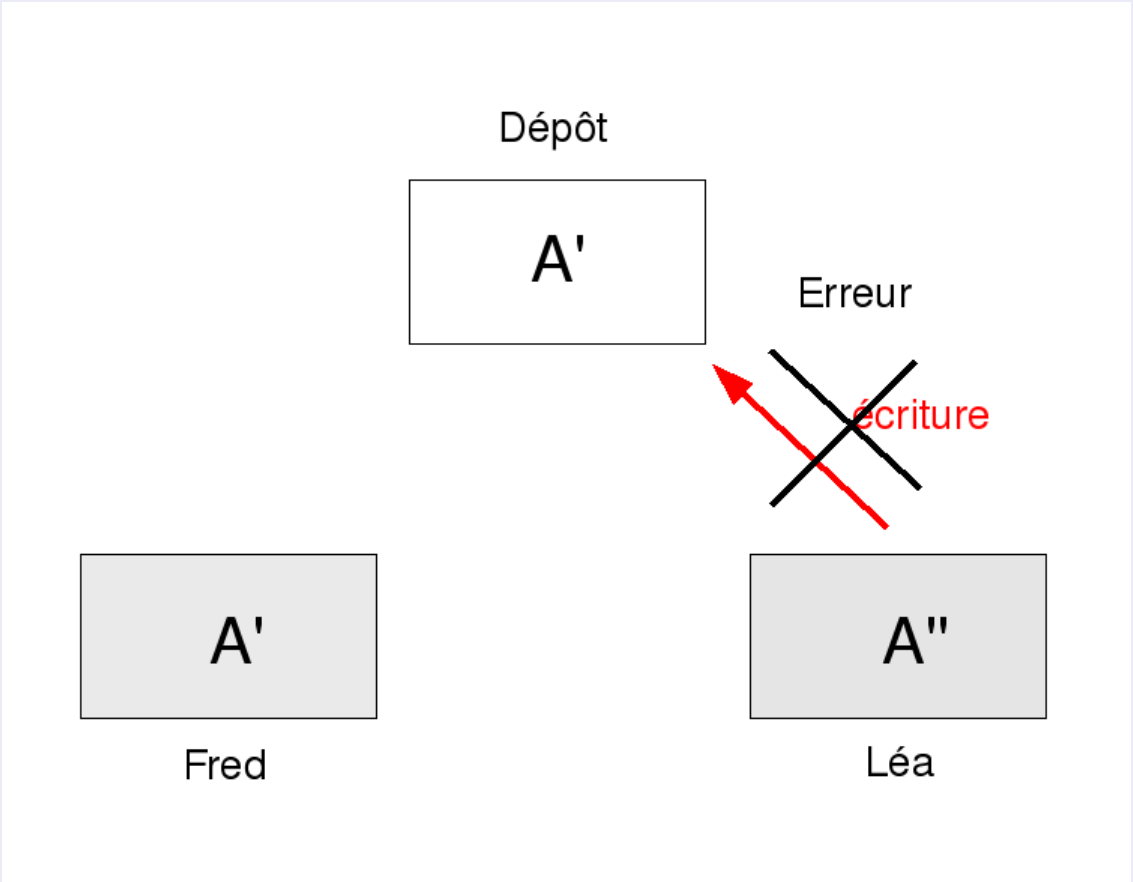
Problème (suite)



Léa écrit sur le dépôt en écrasant la version de Fred

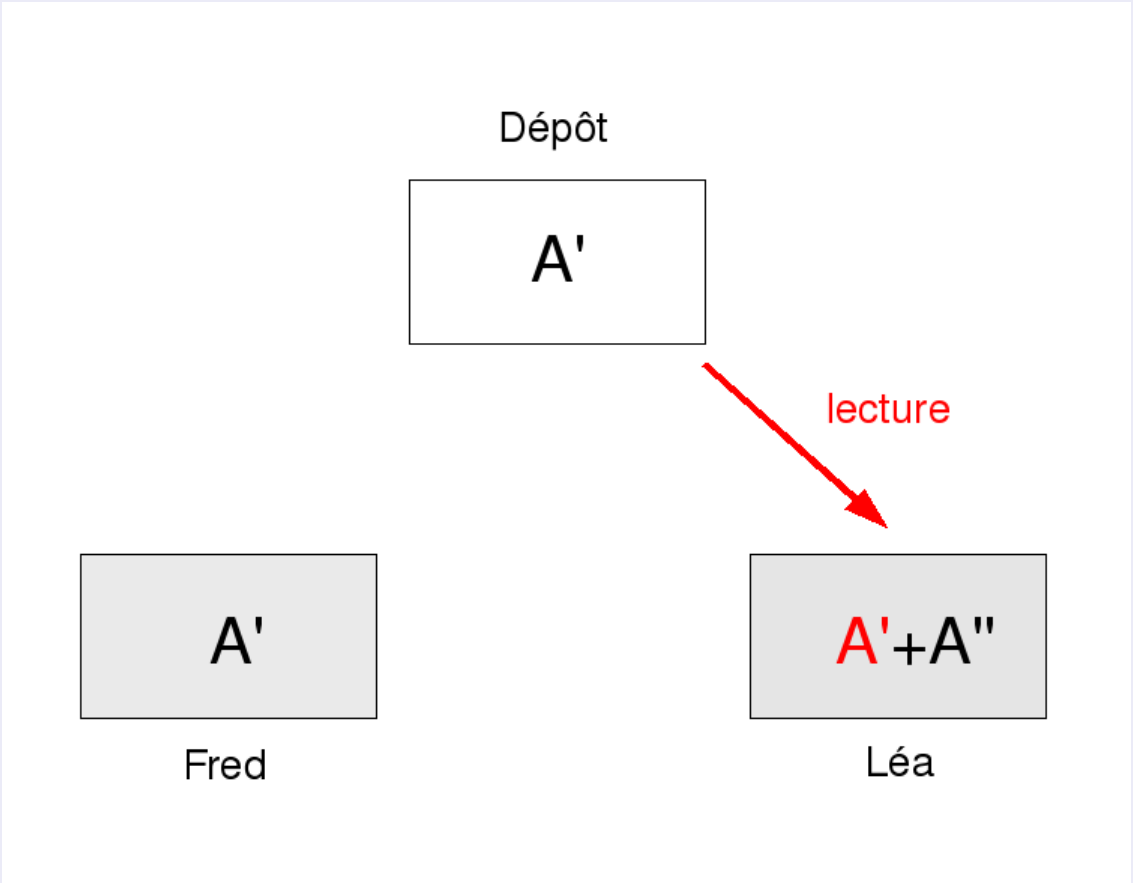
La solution !

Solution



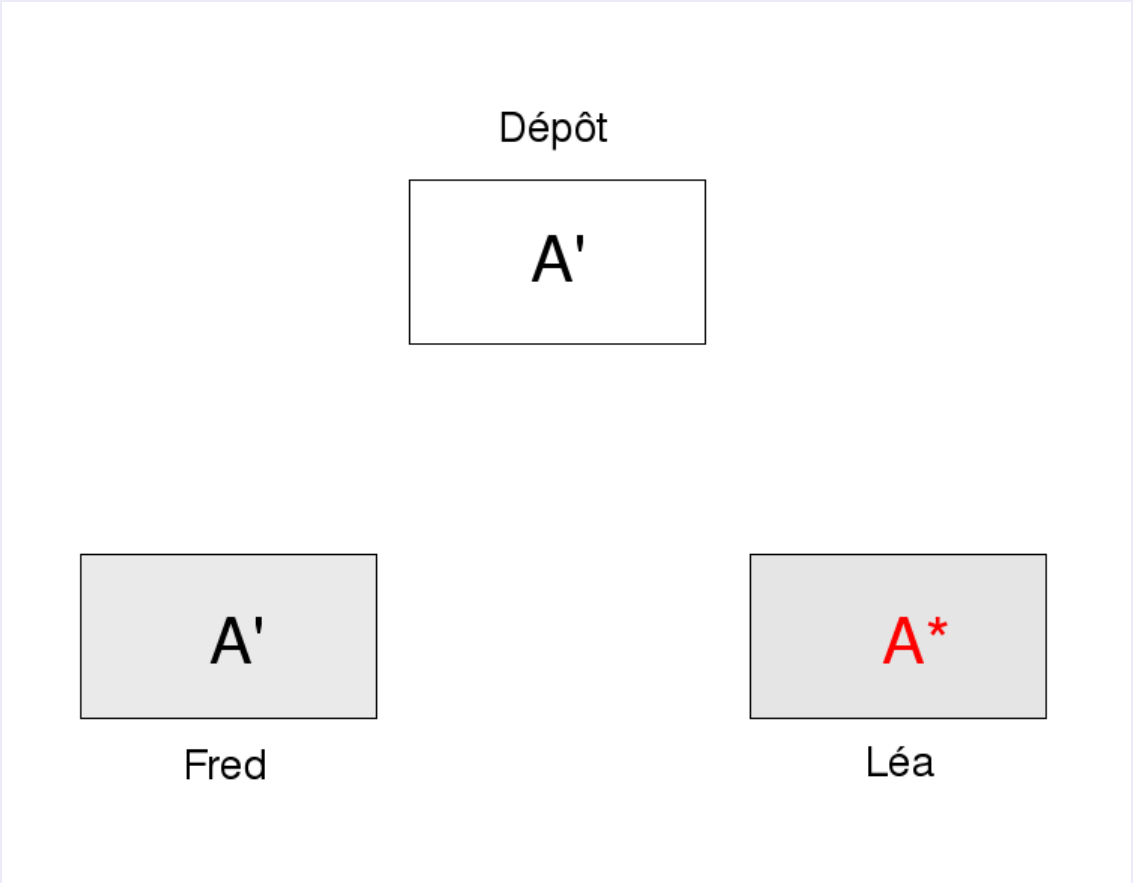
Léa ne peut pas écrire sur le dépôt car sa version n'est pas à jour

Solution (suite)



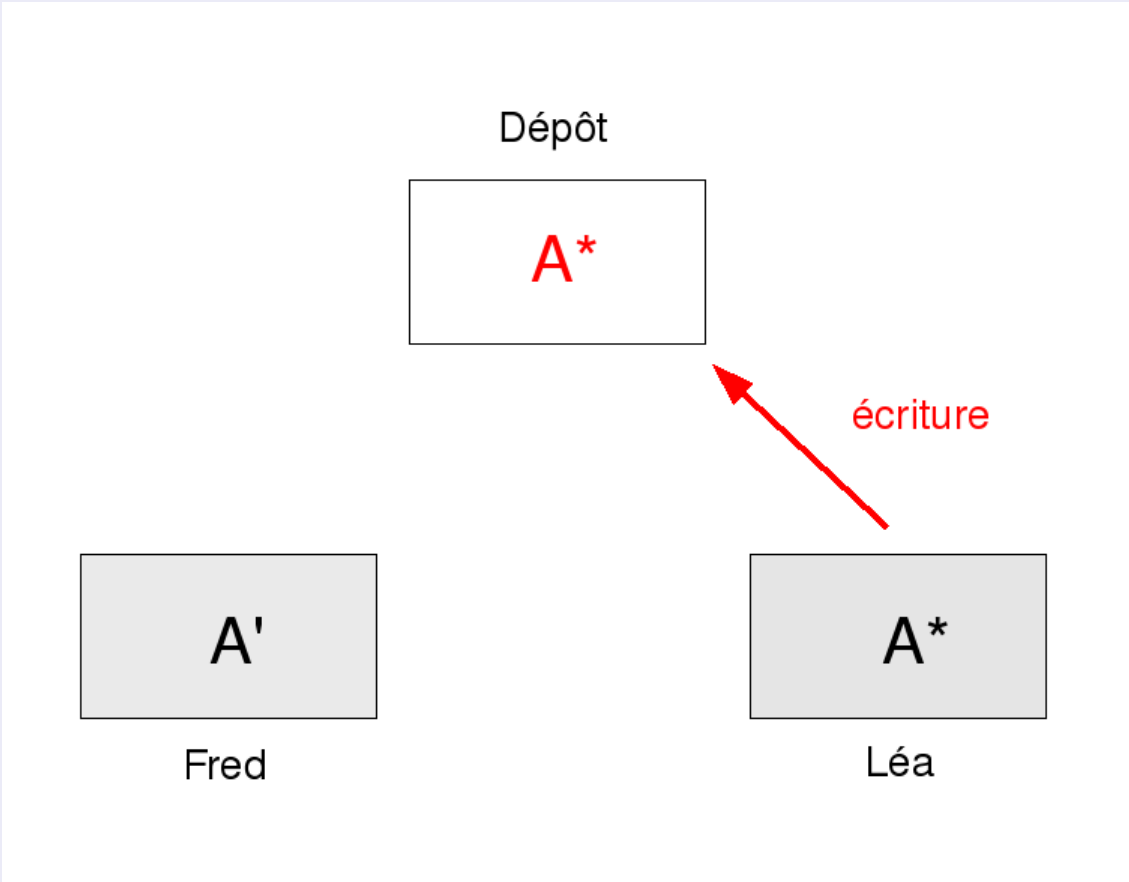
Léa met à jour : elle récupère la version du dépôt sans perdre ses modifications

Solution (suite)



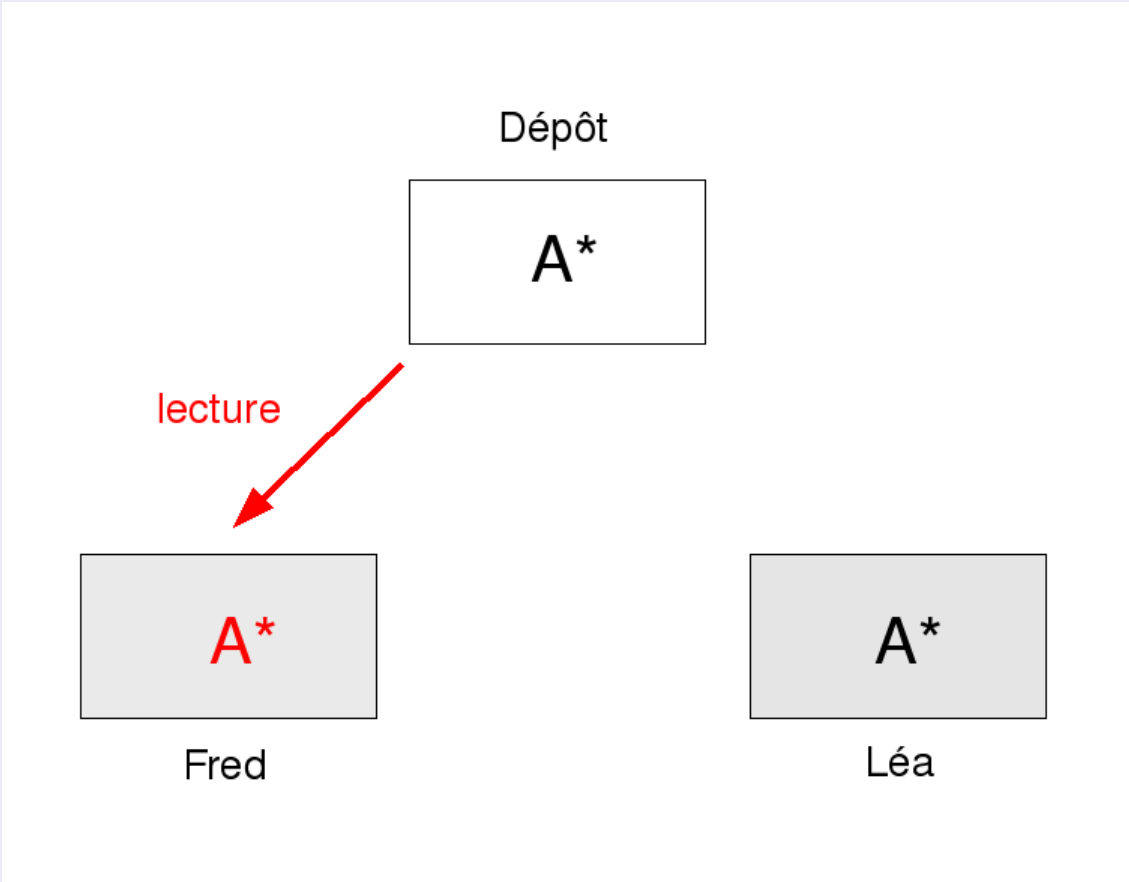
Léa fusionne la version du dépôt (A') avec sa version (A")
A',A" -> A*

Solution (suite)



Léa peut écrire sur le dépôt

Solution (suite)



Fred récupère la nouvelle version

Système de Gestion de Version

Système de Gestion de Version

Un SGV gère le mécanisme de lecture-fusion-écriture

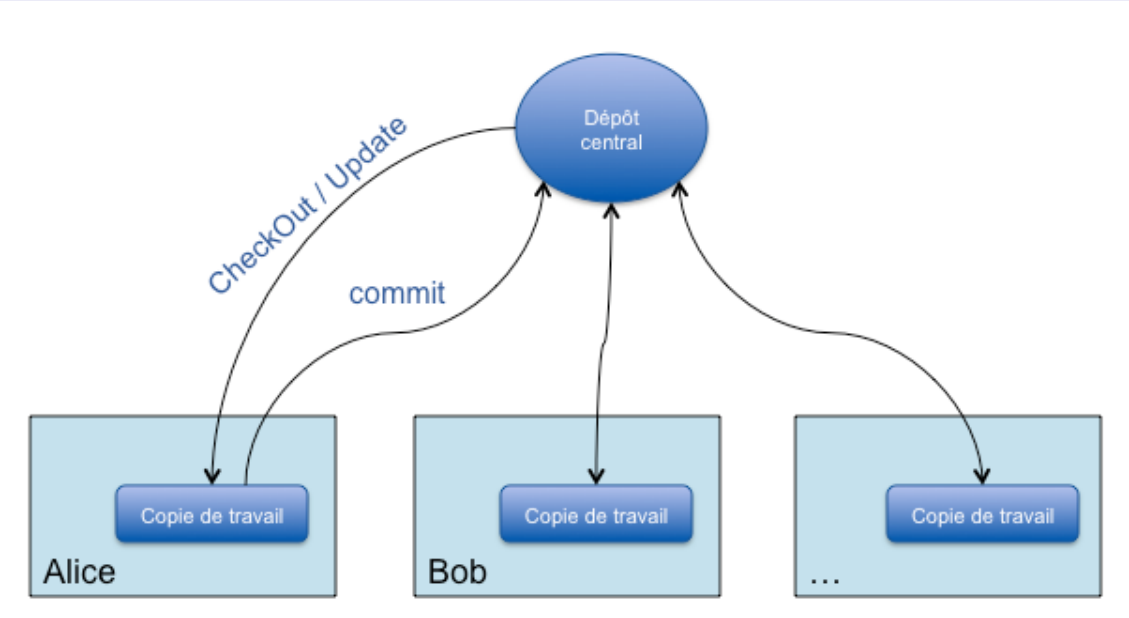
- Les demandes de lecture, écriture se font via le SGV
- La fusion automatique est possible si
  - il s'agit d'un fichier texte (ascii) (utilisation de diff)
  - les modifications ne touchent pas aux même contenus
- Le SGV conserve l'historique
- Et aussi : gestion des branches, tags, ...

PLAN

2 grandes catégories de SGV

1. Les systèmes centralisés

Un seul dépôt centralisé



Des qualités . . .

- Technologie éprouvée
- Largement disponible (IDE, Forges)
- Portabilité
- Sécurité

et des défauts !

- Échange entre les dépôts impossible
- Échange entre les copies locales impossible
- Travail hors connexion impossible
- Temps de mise à jour long pour de gros projets
- Et si le serveur tombe en panne ?

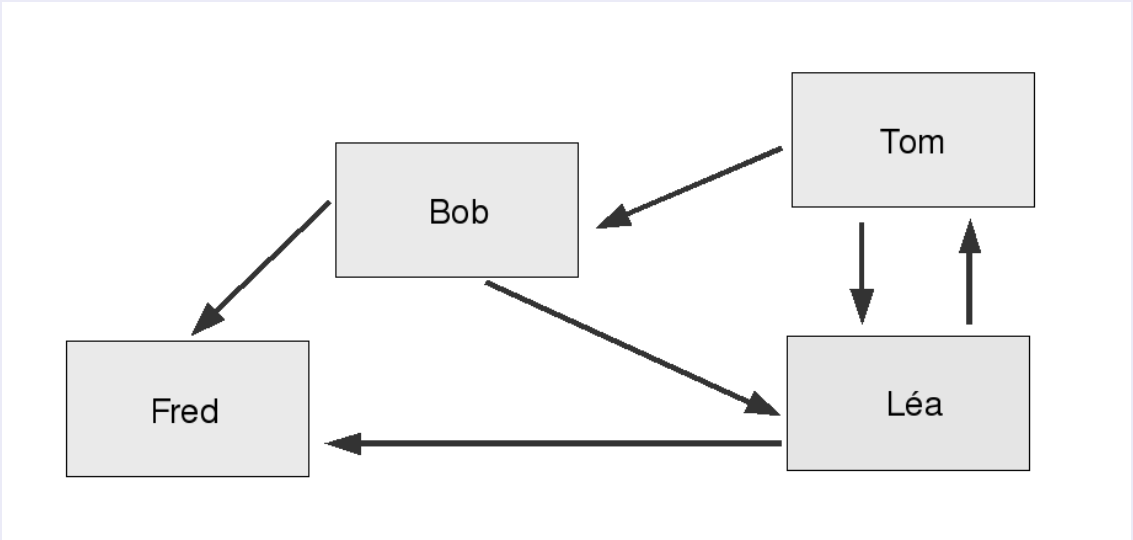
2 grandes catégories de SGV (suite)

2. Les systèmes décentralisés

Objectifs : pallier les limites/problèmes des systèmes centralisés

- Pouvoir utiliser ce système *hors connexion*
- Ne pas être dépendant d'un dépôt centralisé (panne, temps, . . .)
- Pouvoir échanger ses fichiers avec une partie des développeurs
- . . .

Chaque développeur possède son propre dépôt (et sa copie de travail)

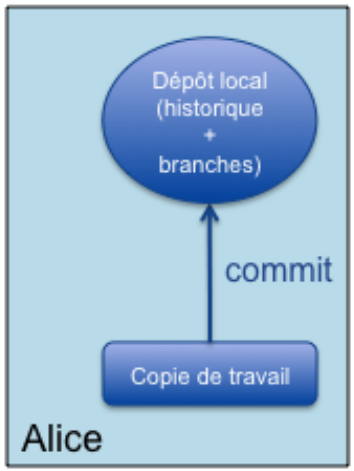


Les sytèmes décentralisés

Les avantages d'un système centralisé (en local)

Chaque développeur a son propre dépôt et sa copie de travail

Il peut donc utiliser un SGV décentralisé pour (par exemple)
conserver l'historique ou gérer des branches en local

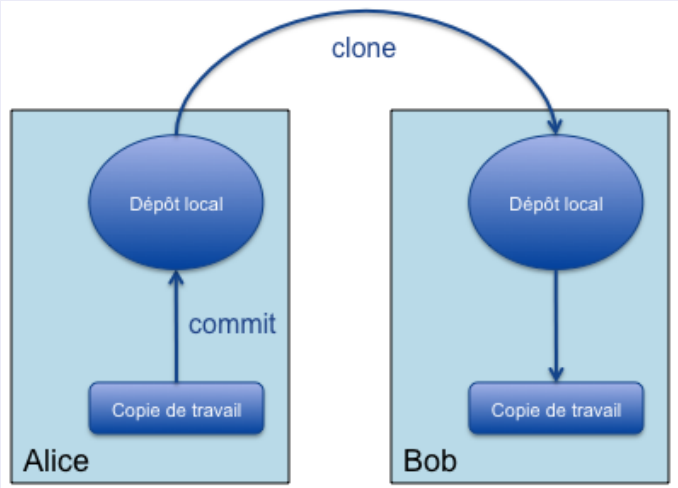


Les sytèmes décentralisés (suite)

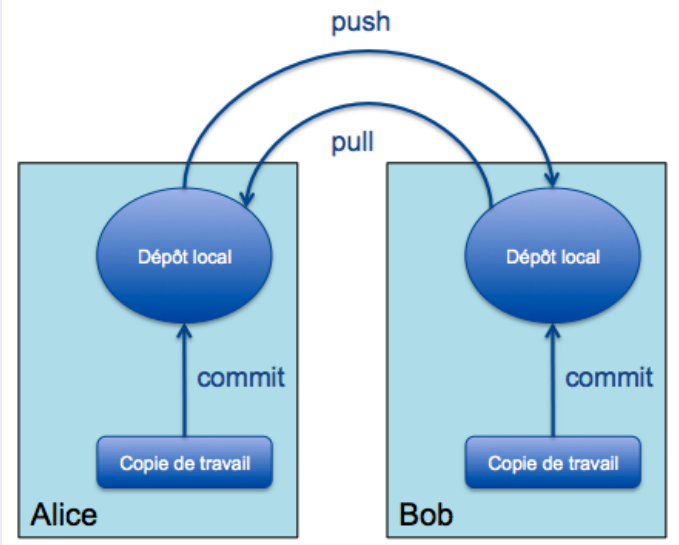
Travail entre dépôts

Les dépôts locaux peuvent communiquer

Clone d'un dépôt vers un autre (en conservant l'historique)



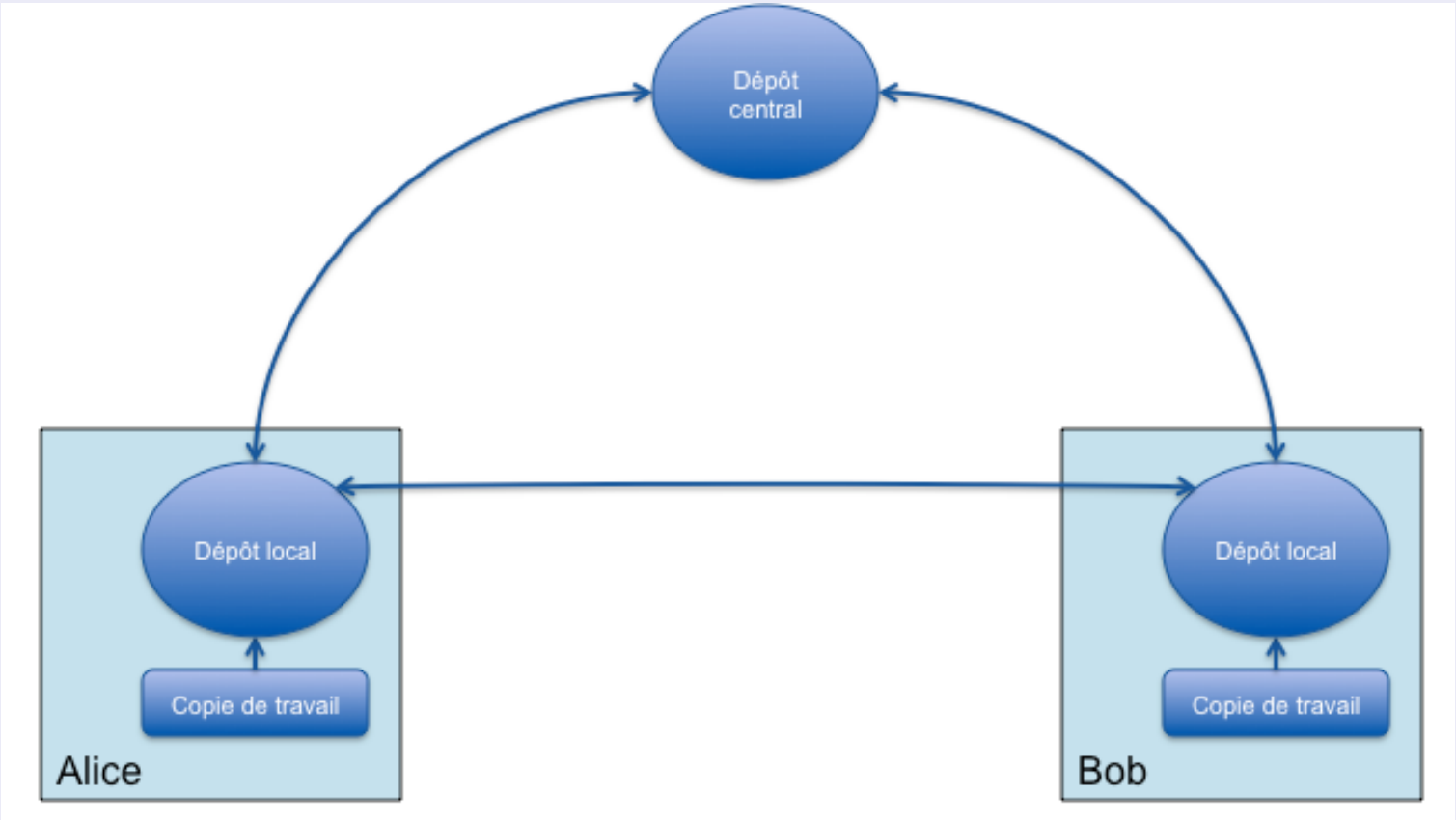
Écriture/Lecture d'un dépôt vers un autre



Les sytèmes décentralisés (suite)

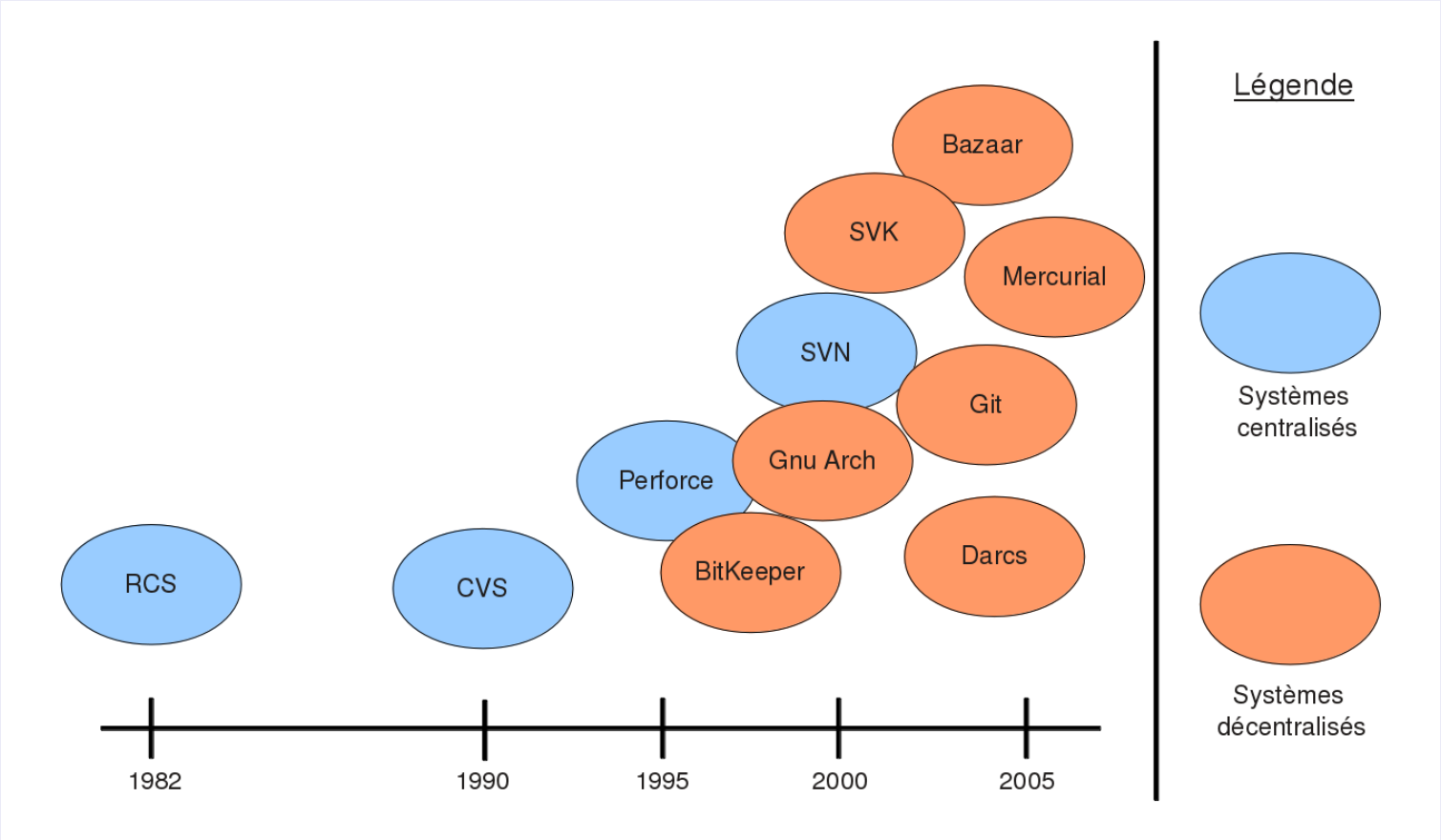
Avec un dépôt central ?

Si le nombre d'utilisateurs est grand, il peut être utile de mettre en pace un dépôt central pour stocker la version la plus à jour du système



Quel SGV choisir ?

Cartographie (incomplète) des SGV



Quel SGV choisir ?

Vaste choix

- Technologie en pleine évolution
- De nouveaux systèmes apparaissent régulièrement

Elements à prendre en compte

- Pérénité : systèmes *leaders* vs. systèmes *émergents*
- Intégré dans des *IDE*
- Proposé par des *Forges*
- Interfaces graphiques
- Portabilité (multi OS)
- Sécurité
- Documentations abondantes
- Outils connexes (ex : cvs2svn)

PLAN

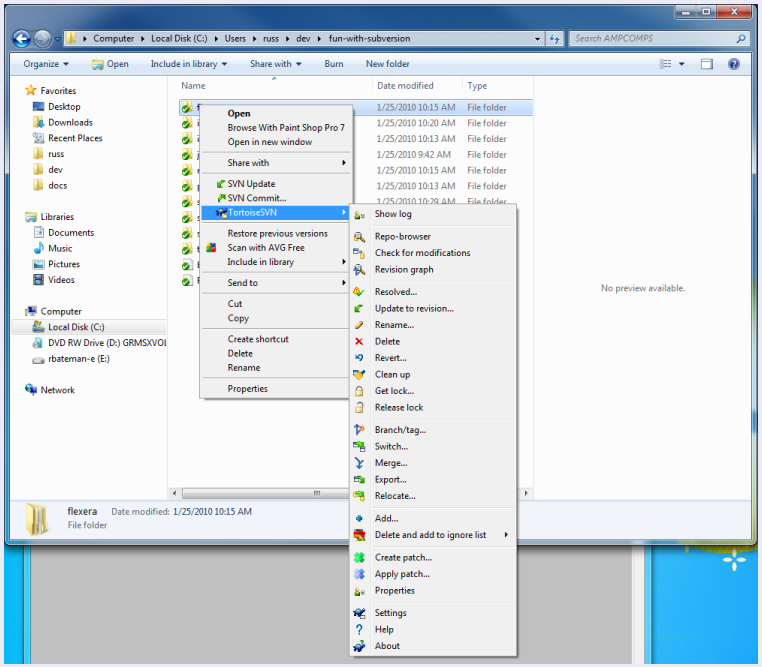
SVN et Git

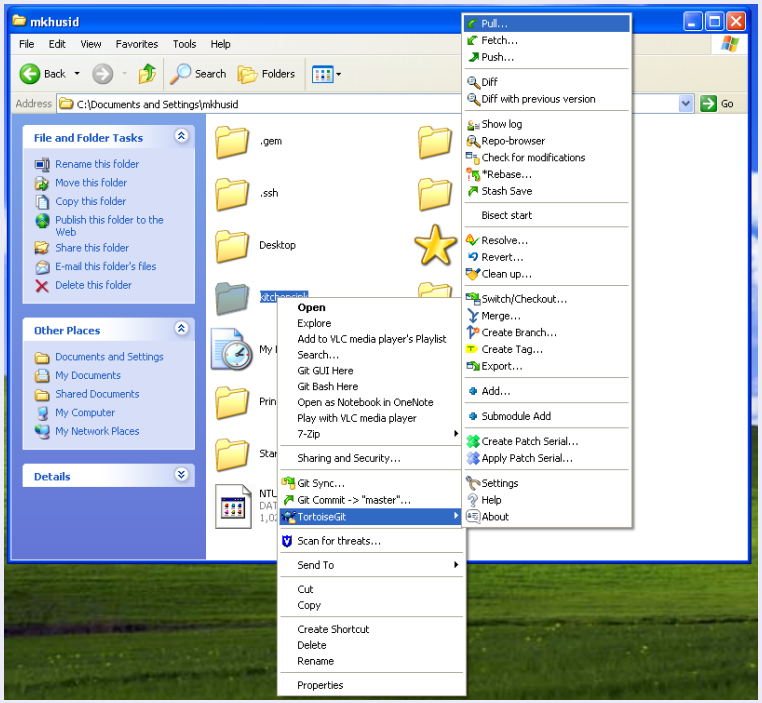
Caractéristiques communes

- Logiciels libres
- Multi OS (linux, windows, MacOS, ...)
- Très répandus (documentations abondantes, support, ...)
- Sécurisés (protocole https, ...)

Des outils utilisables simplement aussi sous windows

Plugins Tortoisegit et Tortoisesvn pour l'explorateur Windows





PLAN

Subversion (SVN)

Un SGV très répandu

- SGV centralisé
- Documentations très riches, forums actifs
- Interfaces graphiques
  - Linux : rapidsvn, kdesvn, esvn, Qsvn, ...
  - Windows : intégré à l'explorateur via le plugin TortoiseSVN
- Proposé dans les Forges et intégré dans certains IDE (Eclipse, Kdevelop)

Le *successeur* de CVS

Reprend le modèle de CVS en comblant certains manques :

- Renommage et déplacement de fichiers sans perte de l'historique
- Gestion des répertoires
- commits atomiques
- Gestion des *metadonnées* (ex : permissions)
- Possibilité de migrer de CVS vers SVN sans perte de l'historique (cvs2svn)
- Protocoles réseaux sécurisés (HTTPS)

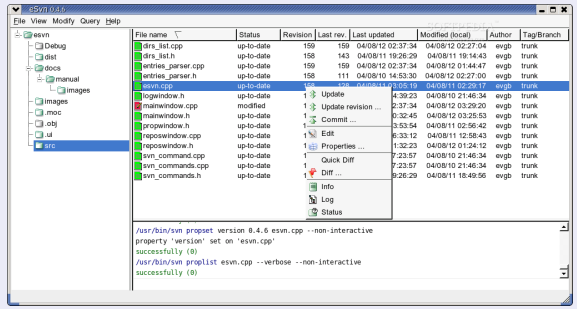
Subversion (suite)

Principales commandes

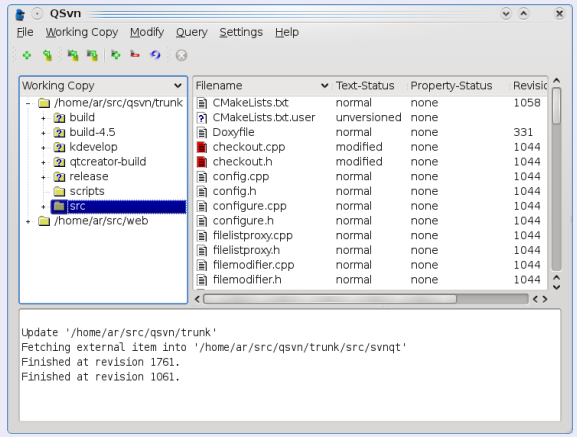
- svnadmin create : créer un nouveau dépôt
- svn import : importer un projet dans le dépôt
- svn checkout : lire tout un projet
- svn update : lire/mettre à jour depuis le dépôt
- svn commit : écrire/modifier le dépôt (nouvelle révision)
- svn status : état de la copie locale
- svn add : ajouter un fichier
- svn rm : enlever un fichier
- svn mv : déplacer un fichier
- svn mkdir : créer un répertoire

Interfaces graphiques

esvn



Qsvn



PLAN

git

GIT en 1 page...

- SGV décentralisé
- Très répandu (surtout dans le monde du logiciel libre)
- Développement actif
- Ressources croissantes (documentations, outils graphiques, ...)

Principales commandes

- git init : créer un dépôt local git
- git add : ajouter un élément
- git commit : propager sa version sur le dépôt
- git clone : copier un dépôt existant (/distant)
- git pull : tirer les modifications du dépôt distant vers le dépôt local
- git push : pousser les modifications du dépôt local vers le dépôt distant

PLAN

Conclusion

Utilisation d'un SGV

- Indispensable lorsque l'on travaille à plusieurs
- Sécurité, efficacité ... même quand on seul sur un projet
- Effort d'utilisation négligeable (avec ou sans interface graphique)

Ne pas utiliser de SGV est une faute professionnelle

Systèmes de Gestion de Version

F. Langrognet