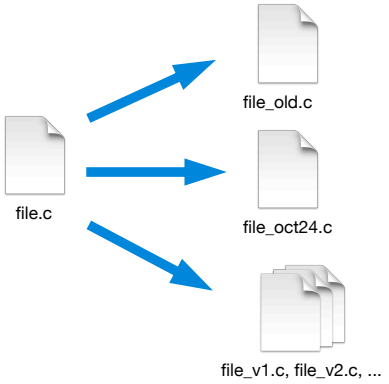
IFT2255 - Génie logiciel

Systèmes de contrôle de révisions

Bruno Dufour
dufour@iro.umontreal.ca

Gestion manuelle des changements



Systèmes de contrôle de révision

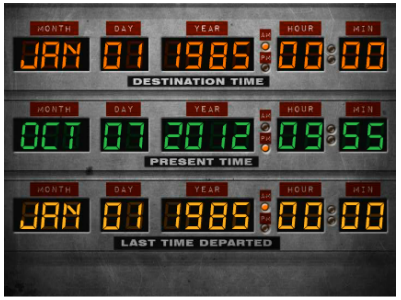
- Objectif: gérer les changements apportés à des fichiers
- Avantages
  - Permettre d'effectuer des changements en toute confiance, et le retour à une version antérieure si nécessaire
  - Permettre de déterminer qui a fait quels changements, et à quel moment
  - Permettre la collaboration à l'aide d'un hiérarchie de fichiers cohérente

Concepts

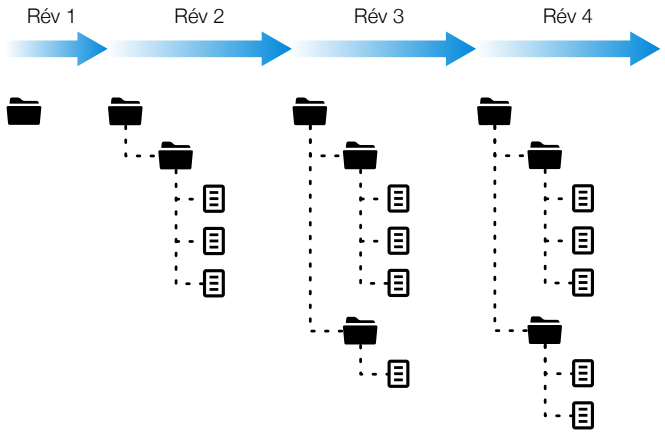
Système de contrôle de révisions

- Un système de contrôle de révisions ou versions (SCR) vise à faire le suivi des changements effectués sur un ensemble de fichiers et dossiers
- Comme serveur de fichiers conventionnel (ex: NFS), un SCR permet d'accéder à la version courante d'un projet
- Un SCR permet aussi d'accéder aux versions antérieures d'un projet
  - Chaque état d'un projet est appelé une **révision**

Système de contrôle de révisions



Révisions



Dépôt

- Un projet sous contrôle de révisions est associé à un **dépôt** (*repository*)
- Le dépôt contient l'historique du projet
  - contenu modifié
  - auteur des modifications
  - temps de modification
  - etc.
- Typiquement, le dépot est enregistré sous forme de base de données

Copie de travail

- Un client qui désire apporter des modifications doit obtenir une **copie de travail** (*working copy*)
  - L'opération de *check out* ou de clonage d'un dépôt permet d'obtenir une copie de travail
- Le SCR fait le suivi des changements apportés à une copie de travail en coulisse
  - Il est possible à tout moment d'annuler les modifications apportées la copie de travail d'un ou plusieurs fichiers

Suivi des changements

- Le SCR fait le suivi des changements apportés à une copie de travail en coulisses
  - ajout de fichier / dossier
  - suppression de fichier / dossier
  - modification du contenu
  - déplacement de fichiers
  - etc.
- Le processus est transparent jusqu'à ce que les changements soient prêts à former la prochaine révision

Soumission

- Au lieu d'enregistrer les changements individuels, un SCR attend que l'utilisateur enregistre un ou plusieurs changements dans une seule transaction atomique
  - Motivation: un changement logique est souvent composé de plusieurs changements individuels
- Lorsqu'un ensemble de changements est prêt, l'utilisateur peut les **soumettre** (*commit*, *check in*)
  - L'utilisateur pourra aussi associer un commentaire descriptif à sa soumission

Création d'une révision

- Suite à une soumission, une nouvelle révision est créée pour l'ensemble de changements
  - Le SCR enregistre tous les détails (fichiers affectés, auteur, moment du changement, etc.)
  - La nouvelle révision obtient un identifiant unique qui permet d'accéder à la cette révision dans le futur
    - ex: un numéro séquentiel (1,2,3,4,...), un code de hashage (5d368c0635de051...)
    - La majorité des SCR supportent aussi l'**étiquetage** de révisions
  - La nouvelle révision devient la révision courante (*HEAD revision*)

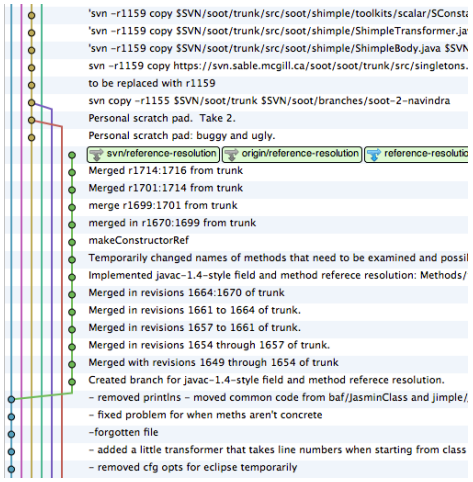
Branches

- Il est souvent désirable d'enregistrer des révisions expérimentales qui pourraient briser la révision principale (*main, trunk*)
  - Les SCR permettent de créer des copies d'une révision, ou **branches**, qui peuvent être modifiées en parallèle
  - Des changements peuvent être soumis dans une branche pendant que d'autres changements sont soumis dans la révision principale.
- Une fois qu'une branche est jugée stable, elle peut être **fusionnée** (*merged*) à la révision principale.

Fusion de fichiers

- Lors d'une fusion, le SCR connaît les changements apportés à chacun des fichiers depuis leur dernier ancêtre commun
  - Le SRC peut effectuer la fusion automatiquement lorsque les changements ne sont pas conflictuels
  - En cas de conflit, un humain devra effectuer l'opération manuellement
  - Le SCR peut offrir une vue des différences pour faciliter la fusion manuelle
  - Typiquement, un usager est averti par le SCR en cas de conflit

Exemple de branches

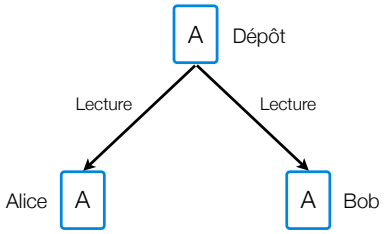


Mise à jour

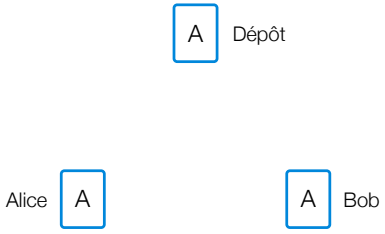
- Lorsqu'un développeur travaille dans une branche, des changements importants peuvent être apportés dans la branche principale
- Les SCR peuvent importer les changements d'une branche dans une autre avec l'opération de **mise à jour** (*update*, *pull*)
  - Importe les derniers changements d'une branche
- Importer les derniers changements fréquemment permet généralement d'éviter les conflits lors de la fusion éventuelle de la branche
  - Facilite la fusion de fichiers

Collaboration

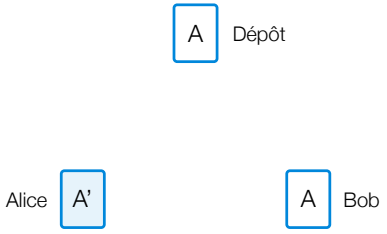
Changements concurrents



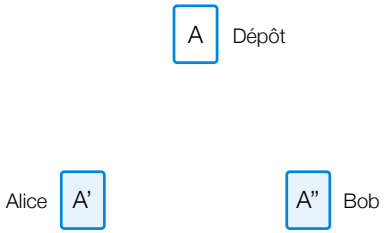
Changements concurrents



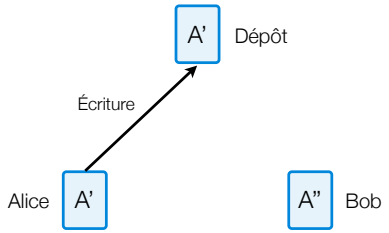
Changements concurrents



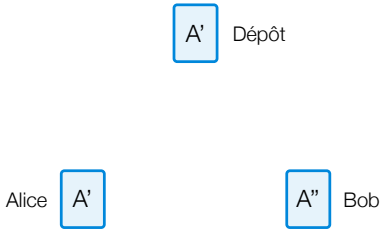
Changements concurrents



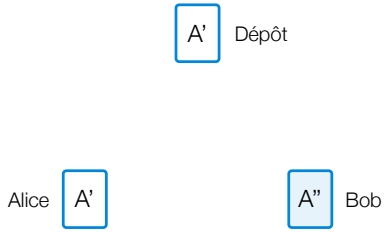
Changements concurrents



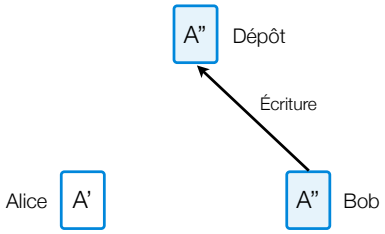
Changements concurrents



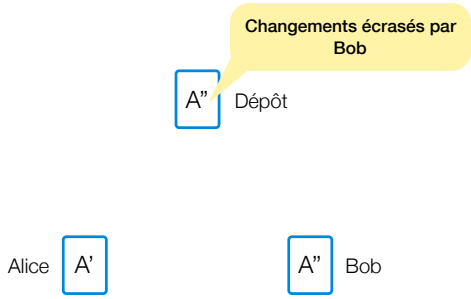
Changements concurrents



Changements concurrents



Changements concurrents



Solutions

- Verrouiller - modifier - déverrouiller
- Copier - modifier - fusionner

Verrouiller - modifier - déverrouiller

- Un fichier ne peut être modifié que par un client à la fois
  - Les autres utilisateurs doivent attendre que le fichier soit déverrouillé avant de le verrouiller et effectuer des modifications à leur tour
- Problèmes
  - Modèle très restrictif, ne permet pas des changements concurrents mais non-conflictuels à un fichier
  - Possibilité d’oublier de déverrouiller un fichier
  - Possibilité pour deux clients des verrouiller deux fichiers interdépendants

Copier - modifier - fusionner

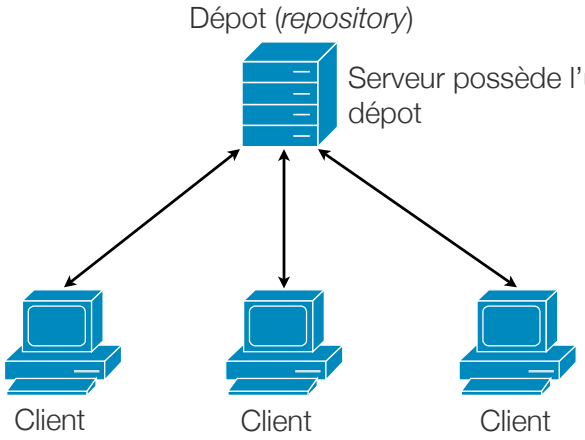
- Les clients obtiennent des copies locales d'un même fichier
  - Des modifications peuvent être effectuées en parallèle
  - La version finale est une fusion des fichiers modifiés

Architectures

Architectures

- Deux architectures principales
  - SCR Centralisé
    - ex: CVS, Subversion (SVN)
  - SCR Décentralisé
    - ex: Git, Mercurial

SCR centralisé

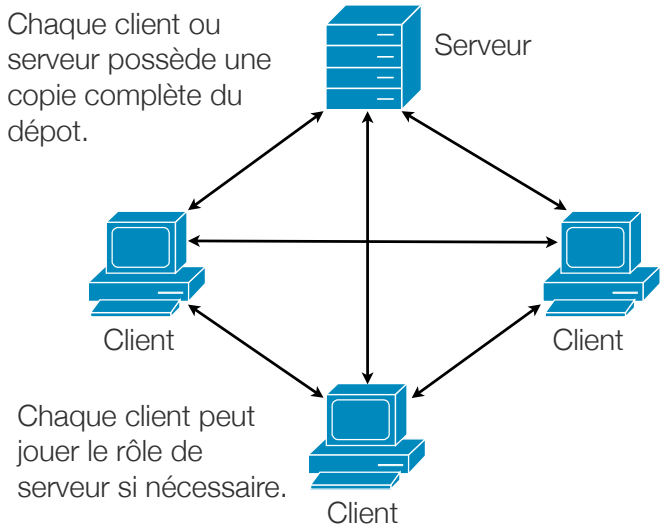


Clients communiquent avec le serveur pour toutes les opérations.

Subversion

- SCR centralisé le plus populaire en ce moment
- Avantages
  - Modèle facile à comprendre
  - Stable et supporté par une variété d'outils
  - Facile à apprendre
- Inconvénients
  - Dépend d'un accès à un serveur
  - Opérations peuvent être lentes
  - Branches et fusion difficiles à utiliser

SCR décentralisé



Git

- Un des SCR décentralisés les plus populaires
  - Créé et utilisé par les développeurs de Linux
- Avantages
  - Serveur optionnel
  - Rapide (opérations locales)
  - Branches efficaces, et donc utilisées fréquemment
- Inconvénients
  - Le modèle distribué peut être difficile à comprendre
  - Apprentissage plus difficile
    - ex: utilisation des codes de hashage les rend plus difficiles à référencer