VCS ou DVCS ?

'Faut-il abandonner Svn pour Git ?'

Journées Informatique de l'IN2P3
Obernai

Antoine Pérus - Laurent Garnier

1er octobre 2008

- Pourquoi un VCS ?

- Quel type de VCS ?
- VCS centralisé
- VCS décentralisé

- Subversion 1.5
- Git
- Comment choisir ...

- Conclusion

Rappel : pourquoi un 'Version Control System' (VCS)

- VCS : *Version Control System*

  un des principaux outils du travail collaboratif
- Pour qui ?
  - développeurs : gestion du code, des projets
  - administrateurs : gestion des fichiers de configuration
  - rédacteurs d'une documentation, d'un support de cours
- Pour quoi ?
  - sauvegarder et restaurer
  - partager et synchroniser
  - revenir en arrière
  - accéder à l'historique documenté de toutes les opérations
  - permettre le développement concurrent avec la gestion des conflits
  - gérer les développements parallèles
- En fait, indispensable même pour un utilisateur isolé !

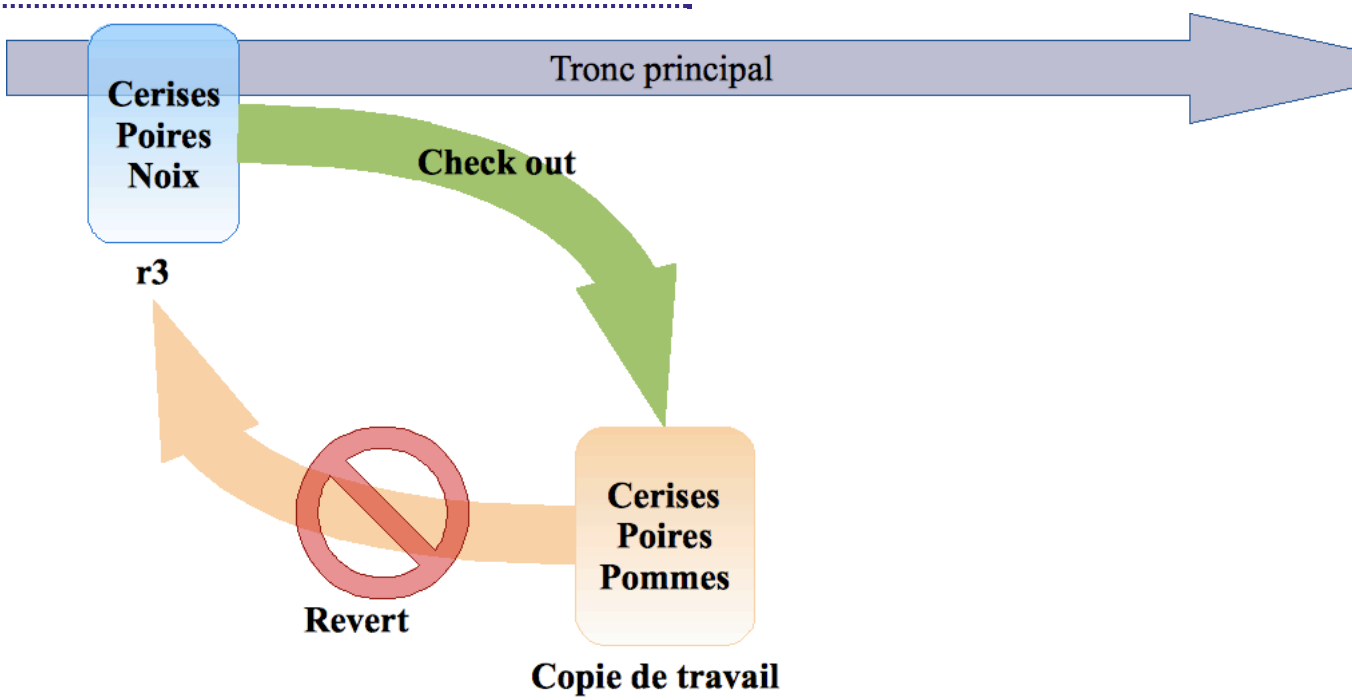
Les mots clés de base :

- le dépôt ou référentiel ou encore 'repository'
- l'espace de travail ou 'working area' ou encore 'working set'
- 'check out'
  - chargement, copie de fichier(s) depuis le dépôt dans l'espace de travail
- ajout - 'add'
  - ajout de fichier(s) dans le mécanisme de gestion de version
- 'check in' ou 'commit'
  - envoi de fichier(s) - si modifié(s) - dans le dépôt; le fichier acquiert une nouvelle révision et cette dernière version peut être rechargée par d'autres utilisateurs
- message de 'commit'
  - un message associé au 'commit' permettant la description de la modification enregistrée dans le dépôt
- révision
  - identifiant d'un fichier versionné, résultat d'un 'commit'
- HEAD
  - la dernière révision dans le dépôt
- historique - 'log' ou 'Changelog'
  - la liste des modifications apportées à un fichier depuis sa création
- 'update' ou 'synch'
  - synchronisation des fichiers locaux avec la dernière révision du dépôt
- 'revert'
  - permet de revenir sur la dernières modifications locales et de recharger la dernière version du dépôt

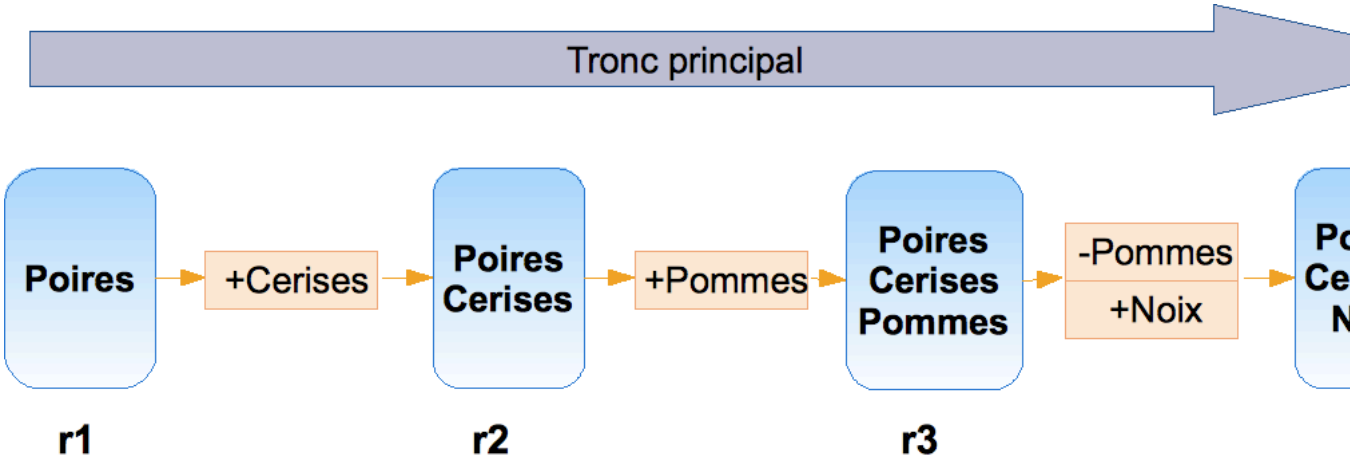
Les mots clés pour des opérations plus complexes :

- branche - 'branch'
  - une copie séparée pour un usage privé : 'bug fix', test, nouvelle fonctionnalité
- 'diff'
  - différence entre fichiers; en particulier entre révisions différentes
- fusion - 'merge' ou 'patch'
  - applique les modifications effectuées d'un fichier à un autre
- conflit
  - situation résultant de l'application de modifications contradictoires
- verrous - 'lock'
  - mécanisme par lequel on protège un fichier de modifications extérieures

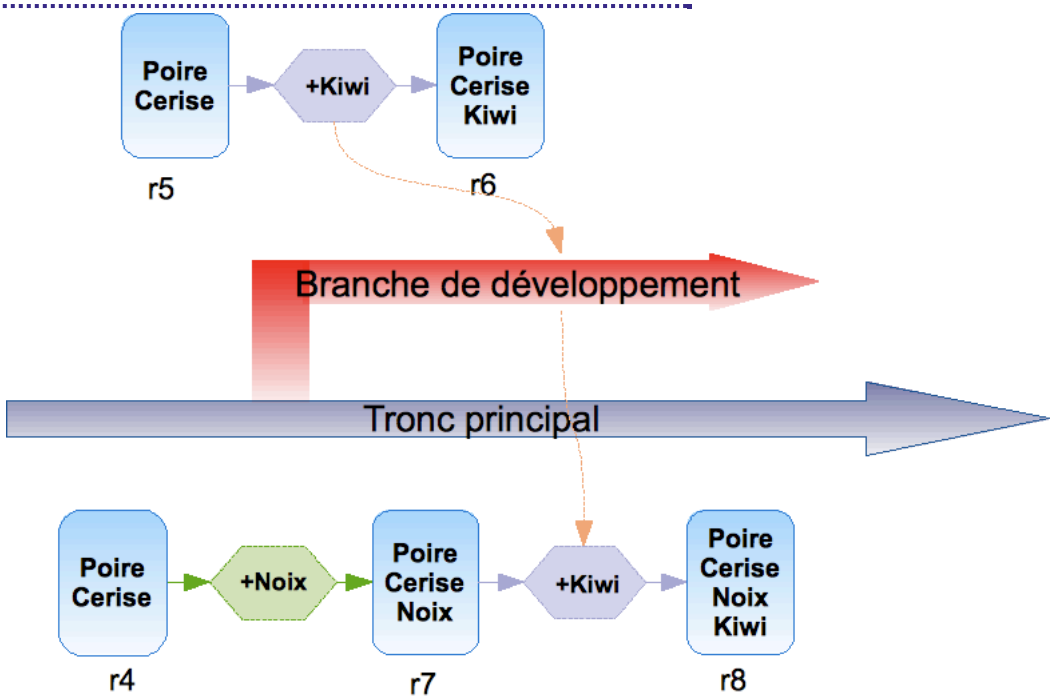
Les notions de base : '*check out*', édition et '*commit*'



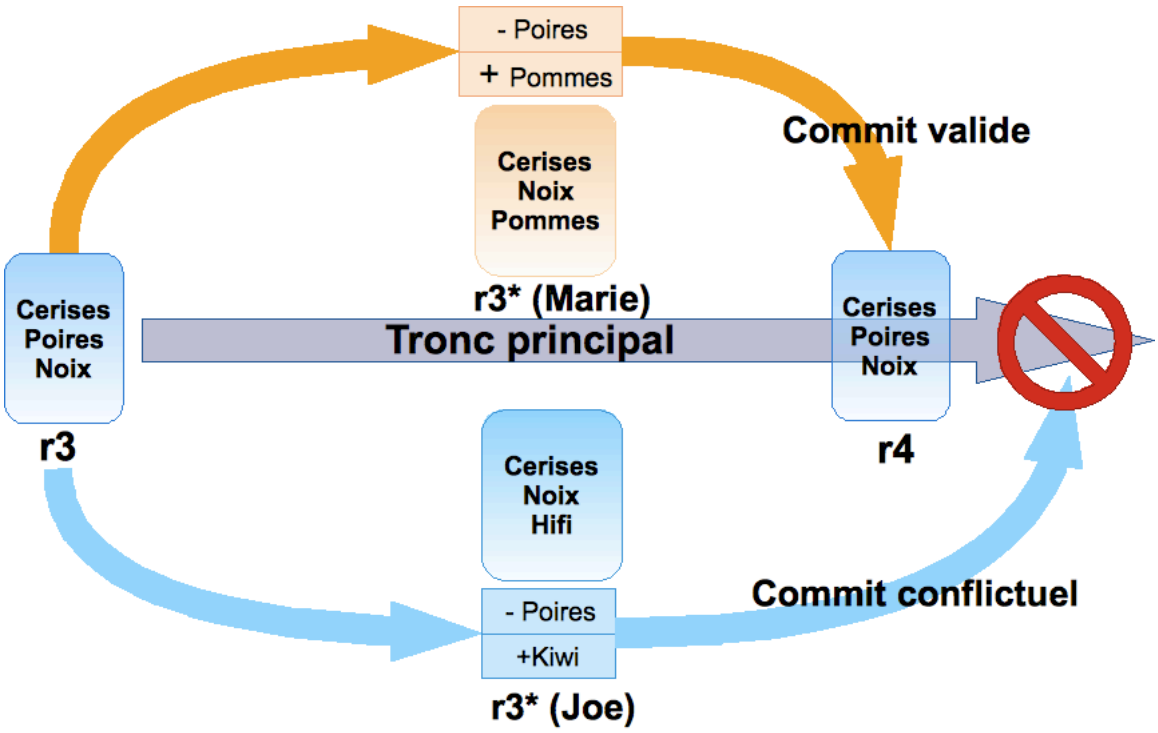
Les notions de base : '*diffs*' simples



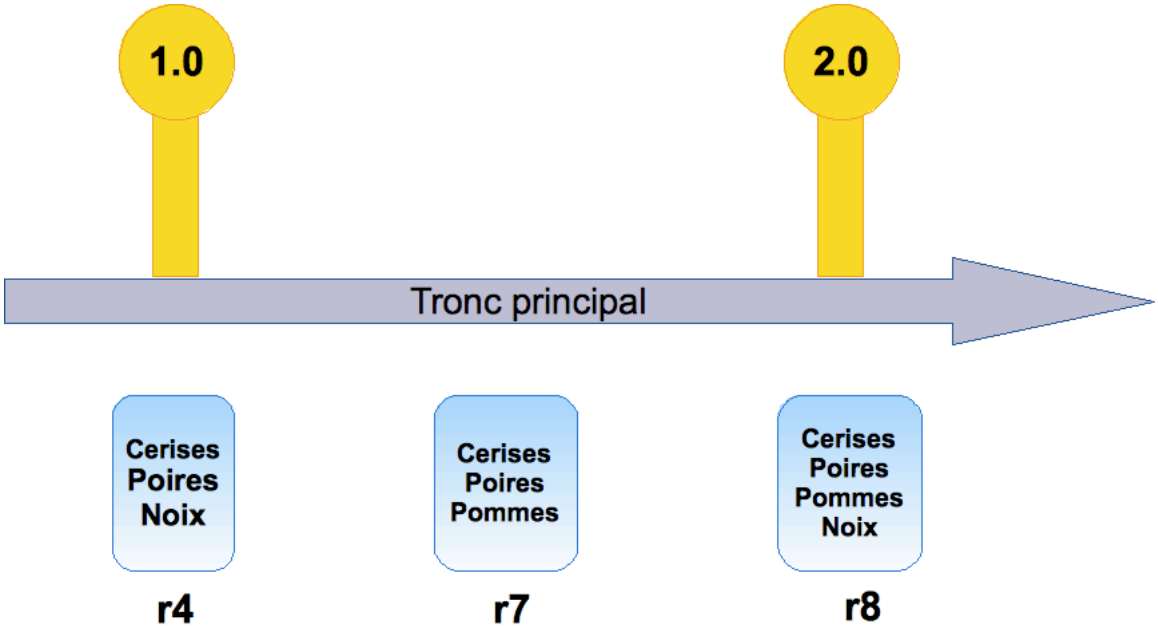
Les notions de base : '*branche*' et '*fusion*' ('*merge*')



Les notions de base : '*conflits*'



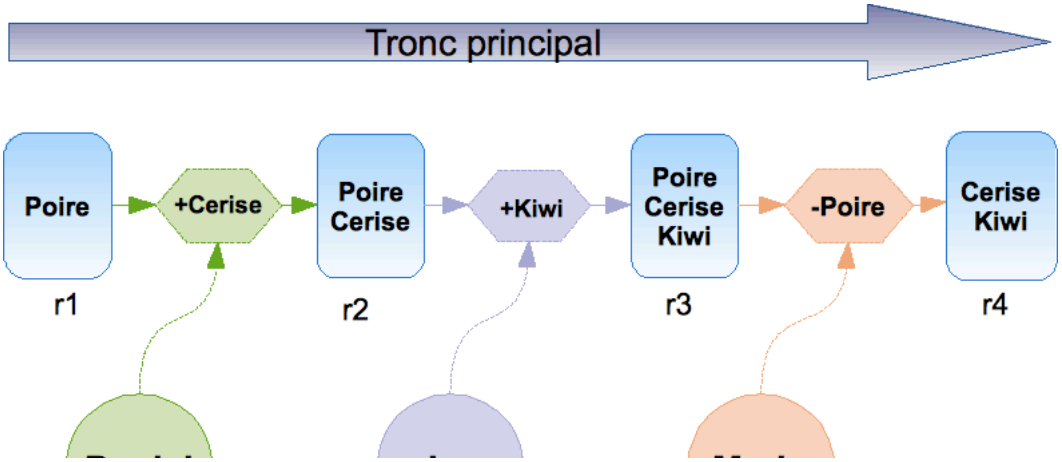
Les notions de base : '*étiquettes*' ou '*tags*'



Quels types de VCS ?

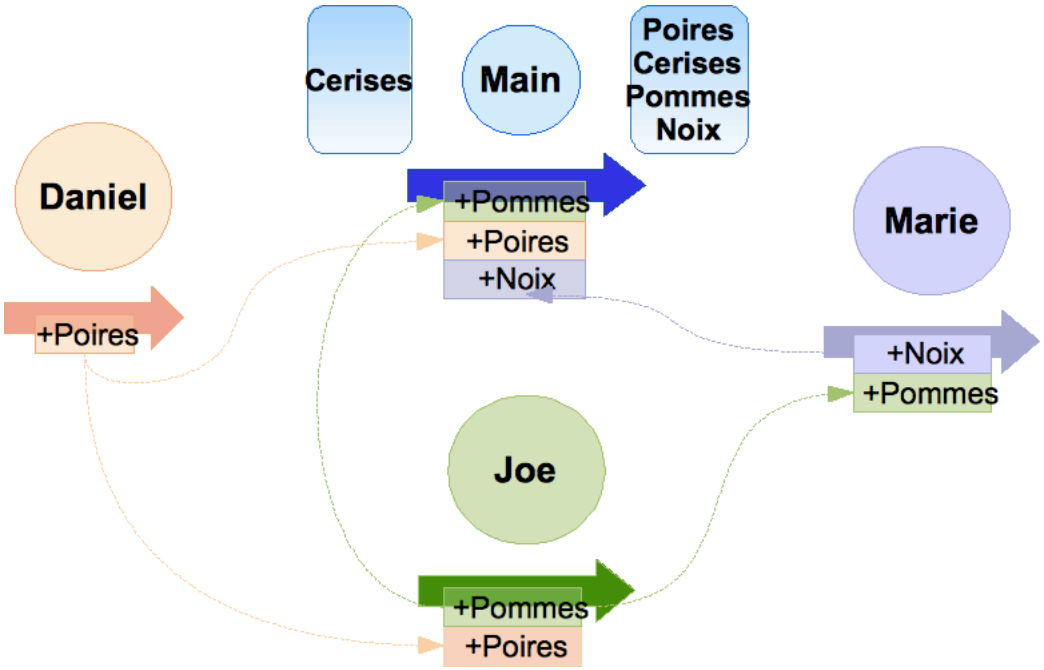
- Le modèle centralisé
  - caractérisé par un dépôt (*repository*) privilégié
  - CVS, Subversion (Svn)
  - politique stricte d'accès au dépôt
- Le modèle décentralisé ou réparti
  - sans dépôt de référence ou privilégié
  - Git, Mercurial, Bazaar, Darcs, Arch et d'autres !
  - chaque développeur ou chaque projet a son dépôt.
    Outre un *commit* dans son dépôt personnel ou local, il lui faudra éventuellement synchroniser 2 dépôts ou *commiter* également dans un autre dépôt de centralisation, de publication par exemple.
  - VCS des développements Open Source et de l'ère *laptop* dans le train ...

Le modèle *centralisé*



Daniel Joe Marie

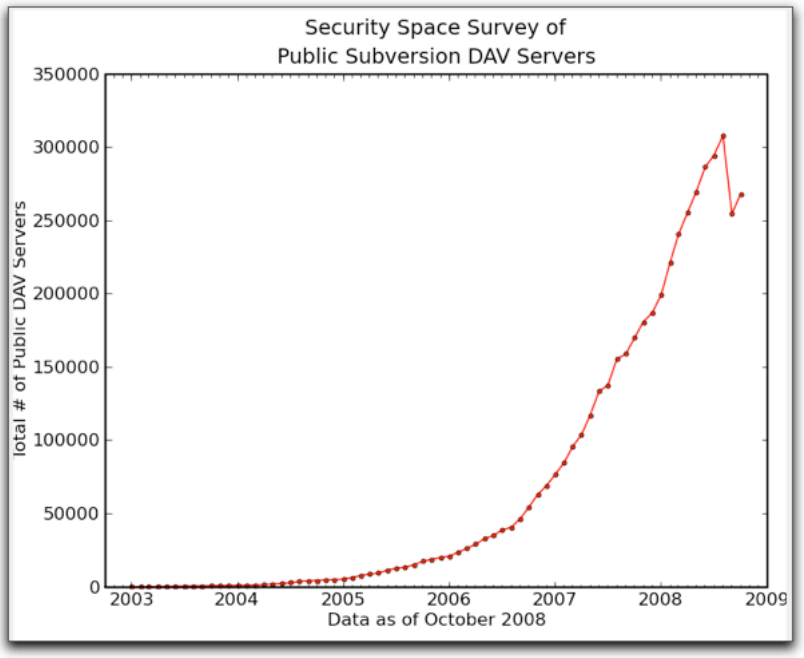
Le modèle *décentralisé*



Les points forts respectifs (1)

- Le VCS centralisé
  - Familiarité et simplicité du modèle
  - Maturité et robustesse des outils
  - Grand nombre d'outils annexe et d'interfaces graphiques
- Subversion
  - Successeur par conception de CVS
  - Base installée - Nombreux hébergements
  - Élégance de son interface - sa scriptabilité
  - Excellente documentation

Subversion n'est certainement pas mort !



L'avenir des VCS centralisés

- Subversion 1.5
  En plus de nombreux bugs corrigés, SVN 1.5 apporte un certain nombre de nouvelles fonctionnalités, dont certaines très attendues :
  - La gestion des copies et déplacements de fichiers ont été totalement revues et améliorées (déplacement direct du fichier dans l'espace de travail)
  - Optimisation complète de l'ensemble donnant une exécution beaucoup plus rapide
  - Apparition du '*sparse checkout*' : on peut ne récupérer qu'une partie de projet, et non nécessairement l'intégralité
  - Implémentation du '*merge tracking*' : la fusion de 2 branches devient beaucoup plus facile et ne demande plus de gestion manuelle des numéros de révision ...
    Voir '*svn mergeinfo*' et '*svn merge*'

L'avenir des VCS centralisés

- Subversion 2
  Il est évoqué par l'équipe de développement les évolutions suivantes, s'inspirant d'un certain nombre de caractéristiques apportées par les DVCS :
  - '*offline commits*' : pouvoir gérer la granularité de ses commits dans des branches locales
  - la gestion de branches locales synchronisées ultérieurement avec le dépôt central
  - la centralisation de la gestion des '*metadata*' en un seul endroit de la copie locale
  - tout en gardant une interface très simple

Les points forts respectifs (2)

- Le VCS décentralisé
  - Pas de serveur à gérer
  - Modèle de développement autorisé très souple
  - Travail déconnecté - accès aux dépôts sans réseau
  - Rapidité
  - Branches privées
  - Sauvegardes distribuées

Parmi les VCS décentralisés du moment - Git

- Linus Torvalds ...
- Écrit essentiellement en C, Perl
- Très rapide, robuste
- Documentation assez riche
- **svn2git** : bi-directionnel (du point de vue de Git)
- Nombreux projets (Kernel Linux, Android (Google), Ruby on Rails, VLC, ...)

- Interface compliquée et moins élégante que celle de Svn
- Design et implémentation peu documentés
- Pas de version Windows native

Parmi les VCS décentralisés du moment - Mercurial

- Écrit en Python
- Interface simple et élégante
- Bonnes perfomances
- Bonne documentation
- Choisi par OpenSolaris, Mozilla

D'autres candidats ...

- Bazaar
- Darcs
- Monotone

Pourquoi et comment choisir

Le modèle centralisé est séduisant :

- on le connaît, on le pratique
- Subversion évolue en intégrant l'expérience acquise avec les DVCS
- il est pratiquement fourni par défaut

Le modèle distribué est séduisant :

- il étend le modèle du traditionnel dépôt centralisé utilisé avec CVS et Subversion
- pas de serveur à installer et à maintenir
- il s'affranchit largement des contraintes du réseau
- il permet, tout au moins dans les environnements Open Source, de démarrer plus facilement, plus souplement une contribution
- une fusion performante entre branches facilite le travail collaboratif et la remontée de correctifs ou d'améliorations

Pourquoi et comment choisir

Les craintes face au nouveau paradigme :

- La facilité à créer des branches peut faciliter, voire encourager, le développement 'obscure', sans échanges avec l'équipe et rendant le '*code review*' difficile sinon impossible ou encourageant la divergence en '*fork*'.
  En contre partie, on pourrait considérer que la fusion de branche sous Svn, peut être tellement complexe qu'on renonce à la fusion et qu'on crée ainsi par défaut un '*fork*' !
- Dans le système distribué, chaque développeur démarre de façon tout à fait légitime un '*fork*' du projet. En fait, le dépôt 'officiel' est purement conventionnel; par défaut tout le travail du développeur est privé et caché; la publication ne peut être que volontaire et nécessite un certain effort !
- Dans le système centralisé, il n'y a qu'un seul dépôt; créer un 'fork' est techniquement difficile. Par défaut, tout le travail est public : il est publié dans le seul dépôt partagé. Il est difficile de travailler de façon privée : cela signifie gérer un gros 'patch', sans historique.

Mon sentiment

- L'utilisation d'un VCS ne peut pas remplacer la communication entre développeurs ...

  et ce, quelque soit l'outil, CVS, Svn ou Git ...

- Il n'y a pas de solution technique à des problèmes de communication sociale

Fin !