TD 5: Développement collaboratif centralisé

ED6 | Licence 3 | Université Paris Diderot

Séances du 3 et 10 avril 2012

L'objectif de ces travaux dirigés est de manipuler deux gestionnaires de versions : Subversion et Git, des outils qui s'appuient sur les principes de diff et patch, pour permettre la sauvegarde de l'historique d'un projet et le travail collaboratif.

A l'issu de ce TD, vous devrez savoir :

- mettre en place un dépôt personnel et y sauvegarder l'historique de votre travail ;
- utiliser un gestionnaire de versions centralisé pour travailler en équipe sur une même branche de développement ;
- travailler sur une branche parallèle au développement principal et fusionner deux branches ;
- effectuer les mêmes opérations dans un modèle de collaboration distribuée ;
- publier des patchs.

Sauvegarder son histoire

Subversion est un logiciel de contrôle de versions développé par Karl Fogel. Il s'inscrit dans un modèle de développement centralisé, c'est-à-dire qu'il maintient une version de référence, appelée trunk, d'un arbre de développement ainsi que les différentes évolutions de cet arbre. Ces données sont entreposées dans un dépôt[1] (repository en anglais).

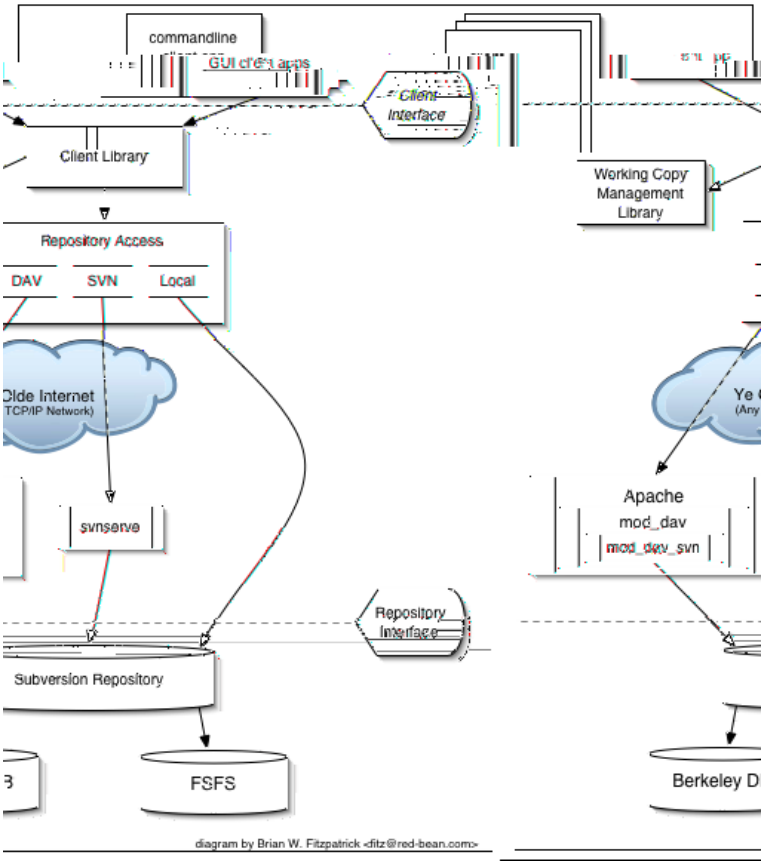

Figure 1 { Architecture de Subversion [1]

Question 1.3. checkout svn,

Question 1.4. svn diff .

Question 1.5. svn commit. $EDITOR

Question 1.6. svn revert ?

Question 1.7. svn status ?

Question 1.8. LISEZMO LISEZMO README svn status ?

Question 1.9. README ?

Question 1.10. svn log ? svn annotate ?

2 Developpement collaboratif

Comme l'a montre notre etude de l'outil diff3 , le travail collaboratif pose le probleme de la modi cation concurrente de chier source.

Dans cette partie, vous allez jouer plusieurs scenarii de developpement collaboratif en binôme (ou si vous êtes seuls, en jouant deux rôles a la fois). Pour cela, chaque groupe de travail doit mettre en place un depôt Subversion partage. Nous utiliserons un acces local car il est le plus simple a mettre en place : mais il faut que tous les utilisateurs du depôt aient acces (en lecture/ecriture) au depôt.

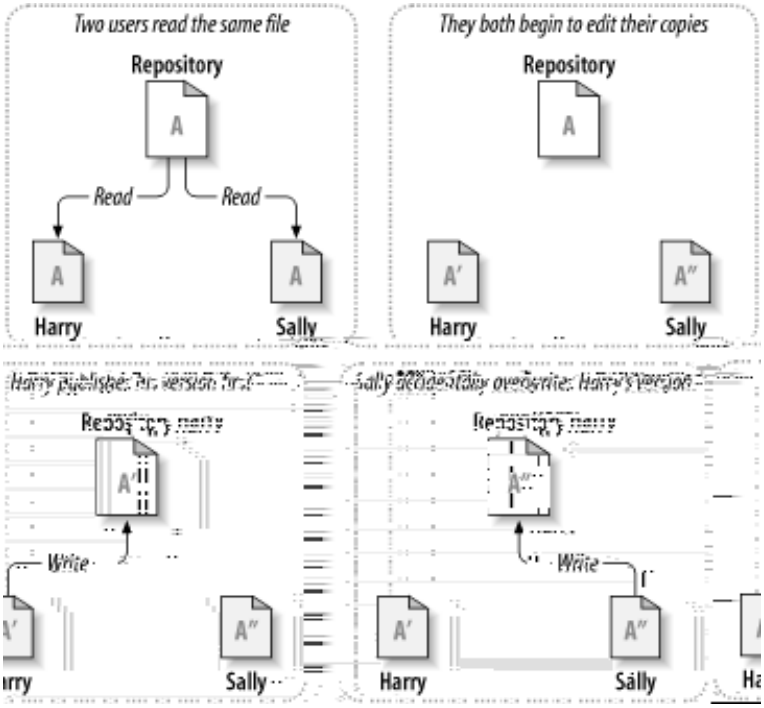

Figure 2 { Situation collaborative non-productive [1]

/info/nouveaux est un syst eme de chier accessible en ecriture depuis tous les postes, ou se trouvent tous les , c'est ici que nous placerons les d epôts.

Question 2.1. [illegible] repertoire personnel [illegible]

[2] chmod -R a+rwX /info/nouveaux/HOME/NOTREDEPOT

Question 2.2 [illegible] R

Question 2.3 (A faire par le binôme 1). [illegible]

[illegible]

Question 2.4 (A faire par le binôme 2). [illegible]

Question 2.5. [illegible]

Deux methodes de travail collaboratif

Pour collaborer correctement, il faut pouvoir travailler de manière concurrente sur un arbre de sources sans introduire de con its. Dit autrement, il est contre-productif de perdre le travail d'un membre de l'equipe parce que le travail concurrent d'un autre membre l'a ecras e.

La gure 2 r esume la situation a eviter. Harry et Sally font des modi cations concurrentes sur un même chier A. Harry publie ses changements. Sally a la possibilit e d'ecraser le travail de Harry sans l'avoir pris en compte.

Deux solutions a ce probl eme sont possibles :

[2] [illegible] chmod [illegible]
{ [illegible] -R [illegible]
{ [illegible]
{ +rwx [illegible] +) [illegible] r [illegible] w) [illegible] X {[illegible]
[illegible]

Lock-Modify-Unlock Harry indique qu'il va travailler sur le chier A en posant un verrou sur A. Sally peut lire le travail de Harry mais ne peut pas modi er le chier tant que Harry n'a pas relâche le verrou.

Copy-Modify-Merge Harry et Sally font evoluer leur copie de travail. Harry publie ses modi cations. Lorsque Sally essaie de publier ses modi cations, le syste me rejette la publication de Sally : elle doit d'abord prendre en compte les modi cations de Harry. Une fusion des modi cations inde pendantes est e ectue e. Si un con it existe, Sally doit d'abord le re soudre pour être autorise e a publier ses changements.

Question 2.6.

Question 2.7.

Question 2.8.

Question 2.9.

Il est parfois ne cessaire de ne pas se synchroniser pendant un moment avec le de veloppement du reste de l'equipe. Un exemple d'une telle situation est la coexistence de deux versions de de veloppement d'un logiciel : une version stable en voie d'être publie e et dans laquelle seules les modi cations de correction d'erreurs mineures sont admises et une version instable dans laquelle est de veloppe e une fonctionnalite supple mentaire. Ces deux versions de de veloppement sont appele es des branches .

Les interactions entre les branches sont de deux types :

Transfert de modi cations particulie res d'une branche a l'autre Tant que le de veloppement de la nouvelle fonctionnalite n'est pas mature, les modi cations qui la concernent sont publie es uniquement dans la branche instable. Durant ce de veloppement, il se peut que des corrections d'erreurs soient e ectue es dans la branche stable : il doit être possible d'importer ces modi cations dans la branche instable.

Synchronisation totale entre deux branches Une fois que le de veloppement de la nouvelle fonctionnalite est termine, il est temps de l'importer dans la branche stable. Pour cela, il ne faut importer que les modi cations qui ont trait a la nouvelle fonctionnalite et non pas les modi cations de correction d'erreurs qui existent de ja dans la branche stable.

Les branches

Question 2.10. scripts .

Question 2.11. svn-publish.sh . svn update svn commit.

Question 2.12. svn merge.

Question 2.13.

Question 2.14. svn log .

Question 2.15. svn merge --reintegrate .

Re férences

[1] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. . 2004.