INF2015 – Développement de logiciels dans un environnement Agile

Gestionnaires de sources

Jacques Berger

Objectifs

Démontrer l'utilité d'un gestionnaire de source

Prérequis

Programmation

Gestionnaire de sources

Logiciel qui gère le code source

- Modifications
- Versions
- Modifications concurrentes
- Imputabilité des changements
- Historique des modifications

Utilité

Le développement sans gestionnaire de sources :

- Copies manuelles (backup et versions)
- Impossible de retracer les changements
- Écraser les modifications des autres
- Historique manuel dans le fichier

Utilité

La gestion des sources est un prérequis à l'agilité

Fonctionnalités

Élémentaires :

- Historique des modifications
- Mises à jour des sources (commit / update)
- Retour en arrière (fichier, version, modification)
- Consulter les anciennes versions d'un fichier

Fonctionnalités

Élémentaires :

- Branches
- Appliquer une modification dans plusieurs branches
- Un dépôt centralisé

Fonctionnalités

Essentielles :

- Accessible par ligne de commande
- Interface visuelle (GUI)
- Intégration à l'IDE
- diff

Fonctionnalités

Utiles :

- Dépôt local
- Interface web
- Déplacer un fichier et conserver son historique

Modifications concurrentes

Réservation de fichiers VS fusion des modifications

Gestionnaires de sources connus

Gratuits :

- CVS
- Subversion (SVN)
- Mercurial
- Git

Chers :

- Team Foundation Server (Microsoft)
- ClearCase

Et plusieurs autres...

Qualités recherchées

Simplicité d'utilisation
- Productivité du développeur

Flexibilité
- Ex. SVN et la création d'un fichier fraîchement supprimé

Rapidité

Inclure

Ce qu'on inclut dans le gestionnaire de sources :

- Code source (évidemment)
- Prototypes jetables
- Tout ce qu'on tappe (tests, documentation, etc.)

Format texte autant que possible

- Les formats binaires ne passent pas l'épreuve du temps

Exclure

Le code généré
- Par un générateur de code
- Par l'IDE

Les librairies

Documents d'analyse

Diagrammes UML

Discussions

Doit-on garder du code en commentaire ou classe inutilisée pour référence ultérieure ou un besoin probable dans le futur?

Discussions

Est-ce que chaque développeur devrait faire ses commits dans sa propre branche et intégrer son travail à la branche principale plus tard ou tous travaillent dans la branche principale?

Centralisé vs décentralisé

2 types de gestionnaires de sources :
- Centralisé
- Décentralisé

SVN est centralisé
Git est décentralisé

Commandes Git

init : Création d'un dépôt local pour un nouveau projet

clone : Création d'un dépôt local à partir d'un projet déjà existant

Commandes Git

add : ajoute des fichiers au prochain commit

status : indique l'état des fichiers (modifié, à jour, nouveau)

diff : montre les différences entre 2 versions d'un fichier

commit : appliquer les modifications

Commandes Git

reset : Revenir en arrière

rm : Supprimer des fichiers de la liste à commettre (qui ont été ajoutés avec le add)

branch : gère les branches (créer, supprimer, lister, etc)

checkout : navigation dans les branches

Commandes Git

merge : fusionner une branche avec une autre

log : affiche l'historique d'une branche

remote : gérer les aliases de dépôts distants

fetch : télécharge une branche

Commandes Git

pull : fetch + merge

push : envoyer la branche sur le dépôt distant

Conclusion

Outil indispensable pour un développeur
- Seul ou en équipe
- Peu importe la taille de l'équipe
- Peu importe la taille du projet