Environnement de Programmation

Philippe Collet

Licence 3 Informatique
2007-2008

Génie logiciel : organisation générale

Génie Logiciel et supports de Programmation
Design Patterns, Réflexivité, Tests OO...

Spécifications Spec. OO

Analyse et Conception Orientée objets

Environnement de Programmation (outils pour le Génie Logiciel)

M1

L3

Organisation (suite)

- Envt de Prog. : 6 cours / 6 TP machine
  - Introduction
  - Gestion de versions et de configurations
  - Construction de programmes, Makefile
  - Création et utilisation de bibliothèques
  - Qualités du logiciel, Documentation
  - Analyse de performances
  - Principes de V&V
  - Tests (principes, organisation, tests unitaires, couverture)
  - Débogage, profilage mémoire et trace
  - Conclusion
- Évaluation
  - 1 TP à rendre (30 %)
  - Examen final (70 %)

Introduction

- Pourquoi ?
- Génie logiciel : définition(s)
- Pourquoi c'est difficile ?

Pourquoi le Génie logiciel ?

- pour passer du développement logiciel *ad hoc* et *imprévisible*

à

- un développement logiciel systématique et réfléchi

Génie logiciel : historique

- Réponse à la crise du logiciel, il y a 30 ans
- Conférence OTAN 1968

La crise du logiciel

- Grosses erreurs :
  - Les sondes perdues (Vénus dans les années 60, Mars en 99)
  - La fausse attaque de missiles (1979)
  - Les missiles Patriotes (1991)
  - 1er vol d'Ariane 5 (1996)
  - L'aéroport de Denver (1994-96)
  - L'an 2000
  - Un incident tous les mois...
    - Les régulateurs de vitesse
    - La bourse de Tokyo
- Les projets logiciels
  - ne livrent pas le produit dans les temps
  - coûtent beaucoup plus chers que prévu.
  - délivrent un produit de qualité très faible
  - échouent dans la majorité des cas !!!
  - Étude américaine de 1995 : 81 milliard $ / an en échec

Pourquoi ne construit-on pas les logiciels comme on construit des ponts ?

- Génie civil
  - Échecs moins nombreux
  - L 'écroulement est grave et met en danger l 'utilisateur
  - On ne répare pas un pont « buggé », on reconstruit un pont qui s'écroule.
  - On inspecte tous les ponts construits sur le même modèle
  - Les ponts résistent à toutes les conditions (à 99 %...)
- Génie logiciel
  - Échecs très nombreux
  - *Crash* système pas considéré comme inhabituel
  - Cause du bug pas directement identifiable
  - Dommages mineurs
  - A part dans les systèmes critiques, on considère que le logiciel ne peut anticiper **TOUTES** les situations

☞ ***Différence d 'approche face à l 'échec, face aux pannes***

Pourquoi ne construit-on pas les logiciels comme on construit des ponts ?

- Génie civil
  - Plusieurs milliers d 'années d 'expérience dans la construction des ponts
  - Les ponts sont des systèmes continus et analogiques
  - On repeint un pont, on change l 'enrobée de la route...
  - On ne reconstruit pas la moitié d 'un pont
- Génie logiciel
  - Les systèmes informatiques se complexifient trop vite
  - Les logiciels passent pas des états discrets, dont certains ne sont pas prévus
  - Ajouts, changements de fonctionnalités, de plates-formes...

☞ ***Différence dans la complexité et dans la maintenance***

Liste (non-exhaustive) des problèmes

- Productivité
  - Coûts et délais
- Simplicité
  - Uniformité, orthogonalité, unicité, normalisation
- Communication H/M
  - Ergonomie, interactivité, multimédia, simplicité, rapidité, documentation (contextuelle)
- Fonctionnels
  - Étendue et pertinence des services, fiabilité (correction, robustesse)

Liste des problèmes (suite)

- Matériau
  - Logiciel, structure, langage, modularité...
- Organisation
  - Gestion de projet visibilité, protections, contrôles
- Réalisme
  - Adéquation aux besoins, évolutivité
- Économique
  - Réutilisabilité, transportabilité
- Diversité
  - BD, IA, Calcul, Parallélisme, Réseau, Internet, intranet
- Divers
  - Juridique, psychologique

Génie logiciel : définition (ou presque)

- Discipline (= méthodes, techniques et outils)
  - basée sur le savoir (théorique)
  - le savoir-faire (pragmatique)
  - et le faire savoir (communication)
  - pour produire (développement)
  - de façon industrielle (taille, diffusion)
  - des logiciels (= les produits)
  - *de qualité au meilleur prix...*

Génie logiciel : les besoins

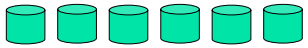
- Langages *pour décrire* — Elts de GL
- Outils *pour manipuler* — Envt de Prog
- Méthodes *pour décider* — Elts de GL
- Théories *pour démontrer*
- Professionnels *pour réaliser*
- Logistique *pour supporter* — Envt de Prog

Gestion de version et de configuration

- Introduction à SVN

Motivations

- Quand on modifie des sources :
  - Des bugs apparaissent parfois (souvent !)
- On pourrait sauver chaque version de chaque fichier modifié...



  - Ou ne stocker que les différences !
- Et quand on est plusieurs à modifier

  - ☞ Savoir qui modifie quoi
  - ☞ Ne rien écraser
  - ☞ Fusionner si on modifie à plusieurs

Principe de la différenciation

- Outil diff
- Différences entre 2 fichiers d'après
  - Ligne de début/de fin
  - Insertion/Suppression d'une ou plusieurs lignes
- Facilité de détection et de construction d'un patch
- Pas de détection des lignes modifiées
  - Traitées comme suppression + insertion

Historique

- SCCS (livré avec Unix dès Vx, Bell labs programmer workbench, fusionné en 1983)
  - Delta descendant
- RCS (W. Tichy 1985)
  - Delta ascendant
- CVS (B. Berliner 1989)
  - WinCVS
  - Support dans beaucoup d'environnements...

Historique (suite)

- Subversion (subversion.tigris.org)
  - La relève de cvs
  - Bonne gestion des modifications de l'arborescence des répertoires
  - Installation et maintenance simplifiée
  - Nombreuses interfaces graphiques ou intégrations dans les IDE
- Visual Source Safe
  - The Microsoft Way
- ClearCase (Rational)
  - L'usine de gestion de traçabilité

Ce que SVN n'est pas...

- Un système de construction (makefile, ant...)
- Un système de gestion de projet (Ms-project)
- Un substitut à la communication entre développeurs (ex: conflit *sémantique)*
- Un système de **contrôle** du changement (bug-tracking, ChangeLog)
- Un système de tests automatisés
- Un système fondé sur un processus particulier

Commande(s) SVN

- `svn subcommand [switches] [cmd_args]`
  - **Commande de base coté client**
  - subcommand : obligatoire
  - switches : options spécifiques à la sous-commande
  - cmd_args : arguments de la sous-commande

```
svn checkout http://svn.c.net/rep/svn/trunk subv
```

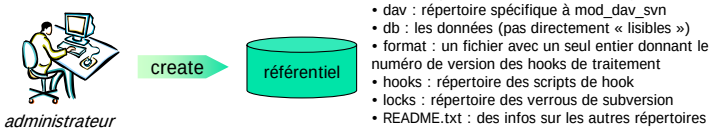
- `svnadmin subcommand [switches] [cmd_args]`
  - **Administration de la base**

Base (ou dépôt) svn

- Locale (accéder directement par le client) :
  - **`file://`**
- Accédée à travers Apache 2 (WebDAV)
  - **`http://`**
  - **`https://`** **(SSL encryption)**
- Accédée par le protocole spécifique « svn » (possibilité de passer par ssh)
  - **`svn://`** **(nécessitée d'avoir un serveur svnserve)**
  - **`svn+ssh://`** **identique à `svn://`, mais *tunneling ssh* (et pas de serveur)**

Administrer une base SVN

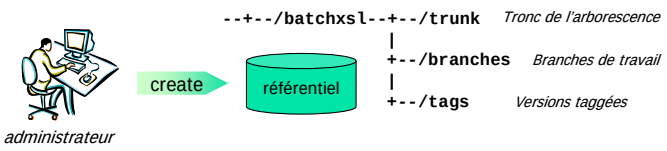
- Créer une base SVN
  - **`svnadmin create /chemin/vers/referentiel`**
  - Par défaut format de stockage FSFS (autre format Berkeley-DB moins performant, conservé pour compatibilité)



Administrer une base SVN

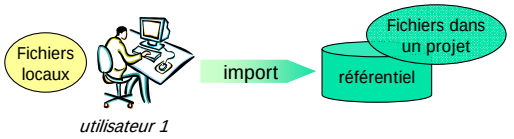
- Au sein d'une base se trouvent un ou plusieurs projets.
- À chaque projet correspond en général un répertoire situé à la racine du dépôt et qui contient lui-même les fichiers et dossiers du projet.
  - Organiser les répertoires :

`svn copy trunk branches/my-branch`



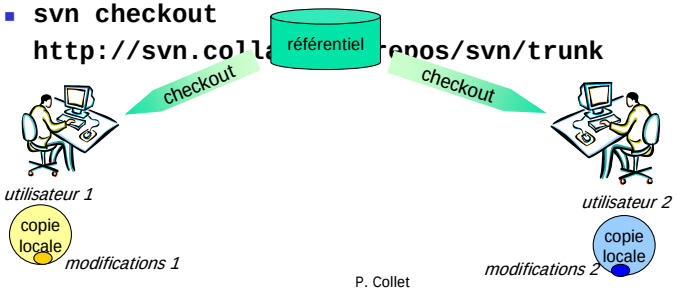
Importer des sources

- `Importer des sources`
  - **`svn import rep_local /chemin/vers/referentiel [options]`**
  - **`svn import myTree file:///usr/local/svn/newrepos/batchxsl/trunk -m "Initial import"`**



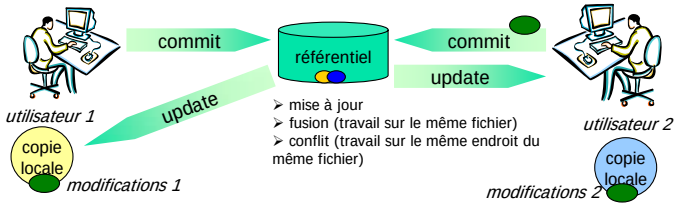
Récupérer une copie locale des sources

- **`svn checkout chemin/vers/referentiel/et/projet [options]`**
- **`svn checkout http://svn.coll…repos/svn/trunk`**



Propager ses changements
Mettre à jour par rapport à la base

- Propagation de vos changements
  - **`svn commit`**
- Récupération de nouvelles mises à jour
  - **`svn update`**



Signification des sorties SVN pour *update* et *checkout*

- **`U file`** : votre répertoire a été mis à jour
- **`A file`** : fichier ajouté à votre copie privée, sera propagé après commit
- **`D file`** : fichier effacé… définitivement après commit
- **`C file`** : conflit détecté lors de fusion
- **`G file`** : fusion effectuée (car pas de conflit)

Quelques commandes et options

- Ajouter un fichier/répertoire : **`svn add`**
  - + commit
- Retirer un fichier/répertoire : **`svn delete`**
  - + commit
- Copie des fichiers/répertoires : **`svn copy`**
  - + commit
- Déplacer des fichiers/répertoires : **`svn move`**
  - + commit

Quelques commandes et options (suite)

- Liste des répertoires dans le référentiel :
  - `svn list`
- Affichage des messages de commit :
  - `svn log`
- Mes modifications locales (pas de connection au référentiel) :
  - `svn status`
- Visualiser les différences :
  - `svn diff`
- Revenir en arrière (undo) :
  - `svn revert`
- Indiquer qu'un conflit est résolu sur un fichier :
  - `svn resolved sandwich.txt`

Plein d'autres choses...

- Cf. refcard
- Les outils graphiques aident à rendre tout cela *vivable*
  - *RapidSVN (multi-plateformes)*
  - *TortoiseSVN (très bonne intégration dans Windows)*
  - *Les supports des environnements de développement (Eclipse : très bon support)*

Références

- The SVN Book (en téléchargement, licence GPL)
  - http://svnbook.red-bean.com/
- Source d'information
  - http://subversion.tigris.org/