

Gestion de version et de configuration : de svn à git

Philippe Collet

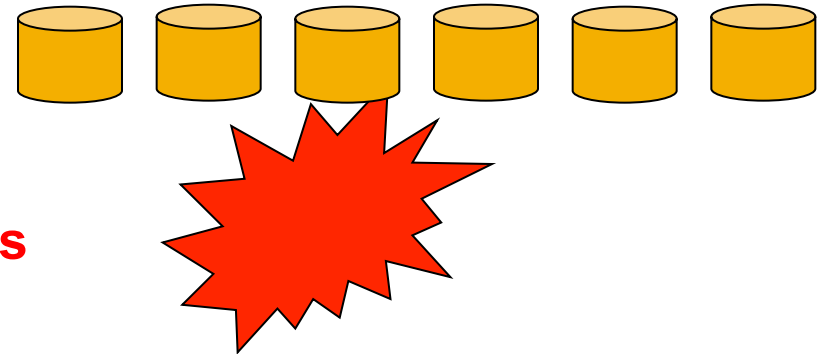
Licence 3 MIAGE – S6

2014-2015

http://miageprojet2.unice.fr/User:PhilippeCollet/Projet_de_d%c3%a9veloppement_2014-2015

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**
 - ➡ **Revenir en arrière (un bug...)**
 - ➡ **Gérer des développements en //**



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

Différenciation : illustration

WinMerge - [DiffContext.cpp - DiffContext.cpp]

File Edit View Merge Tools Plugins Window Help

Src\ - Src\ DiffContext.cpp - DiffContext.cpp

Location Pa x G:\WinMerge\WinMerge_SVN\Src\DiffContext.cpp G:\WinMerge\WinMerge_26Branch\Src\DiffContext.cpp

```
*/
CDiffContext::CDiffContext(LPCTSTR pszLeft /*=NU
: m_piFilterGlobal(NULL)
, m_piPluginInfos(NULL)
, m_nCompMethod(-1)
, m_bIgnoreSmallTimeDiff(FALSE)
, m_pCompareStats(NULL)
, m_pList(&m_dirlist)
, m_piAbortable(NULL)
, m_bStopAfterFirstDiff(FALSE)
, m_pFilterList(NULL)
, m_pCompareOptions(NULL)
, m_pOptions(NULL)
{
    m_paths.SetLeft(pszLeft);
    m_paths.SetRight(pszRight);
    InitializeCriticalSection(&m_criticalSect);
}

/**
 * @brief Construct copy of existing CDiffContex
 *
*/
```

Ln: 71 Col: 1/22 Ch: 1/22 1252 Win

```
*/
CDiffContext::CDiffContext(LPCTSTR pszLeft /*=NU
: m_piFilterGlobal(NULL)
, m_piPluginInfos(NULL)
, m_hDirFrame(NULL)
, m_nCompMethod(-1)
, m_bIgnoreSmallTimeDiff(FALSE)
, m_pCompareStats(NULL)
, m_pList(&m_dirlist)
, m_piAbortable(NULL)
, m_bStopAfterFirstDiff(FALSE)
{
    m_paths.SetLeft(pszLeft);
    m_paths.SetRight(pszRight);
}

/**
 * @brief Construct copy of existing CDiffContex
 *
*/
```

Line: 68-69 1252 Win

Ready Difference 6 of 34

Historique

- ❑ **SCCS** (livré avec Unix dès Vx, Bell labs programmer workbench, fusionné en 1983)
- ❑ **RCS** (W. Tichy 1985)
- ❑ **CVS** (B. Berliner 1989)
 - Support dans beaucoup d'environnements...
- ❑ **Subversion** (subversion.tigris.org)
 - Bonne gestion des modifications de l'arborescence des répertoires
 - Installation et maintenance simplifiée
- ❑ **Visual Source Safe** : The Microsoft Way
- ❑ **ClearCase** (Rational) : L'usine de gestion de traçabilité
- ❑ **Git, Mercurial...**

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://` (nécessité d'avoir un serveur svnserv)
- `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)**



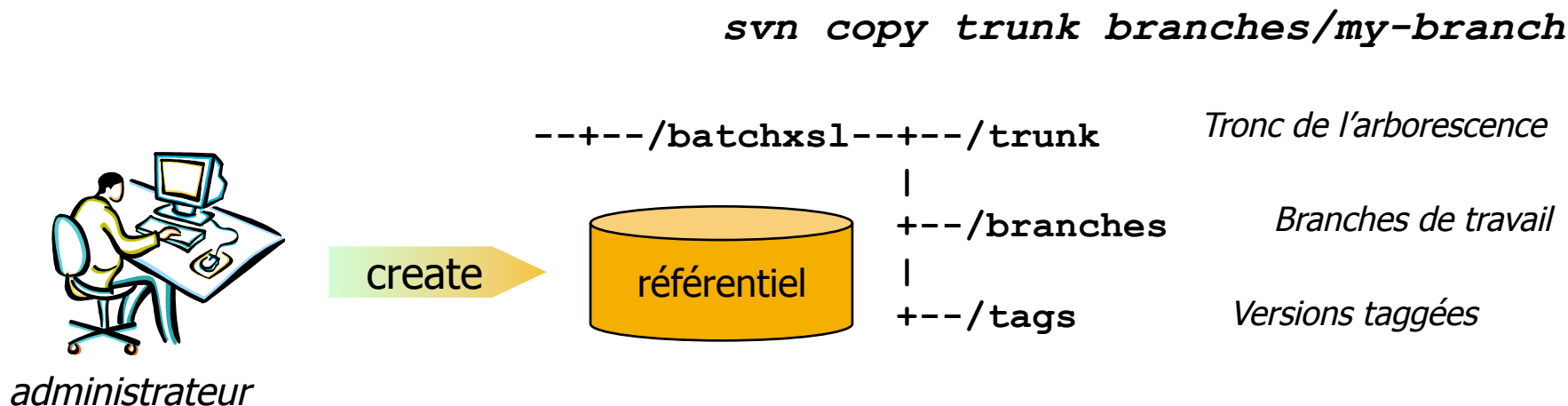
create



- conf : répertoire des fichiers de config
- dav : répertoire spécifique à mod_dav_svn
- db : les données (pas directement « lisibles »)
- format : un fichier avec un seul entier donnant le numéro de version des hooks de traitement
- hooks : répertoire des scripts de hook
- locks : répertoire des verrous de subversion
- README.txt : des infos sur les autres répertoires

Administrer une base SVN

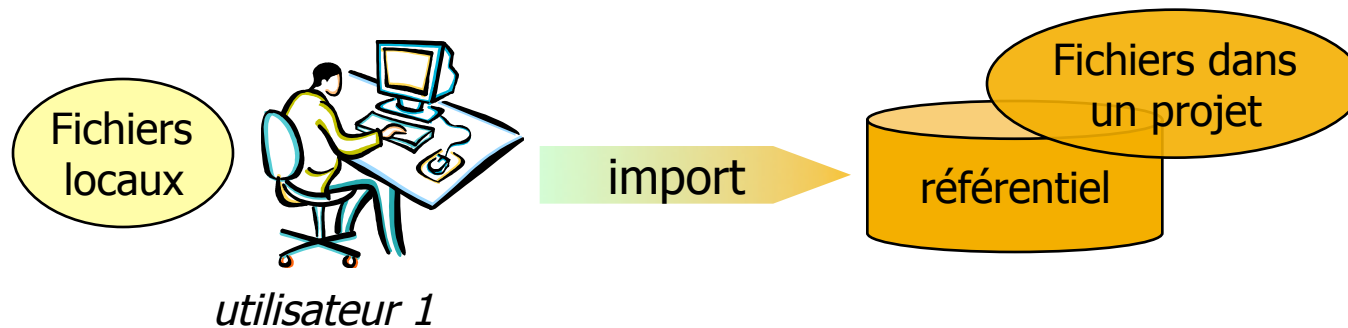
- ❑ Au sein d'une base se trouvent un ou plusieurs projets.
- ❑ À chaque projet correspond en général un répertoire situé à la racine du dépôt et qui contient lui-même les fichiers et dossiers du projet.
 - Organiser les répertoires :



Importer des sources

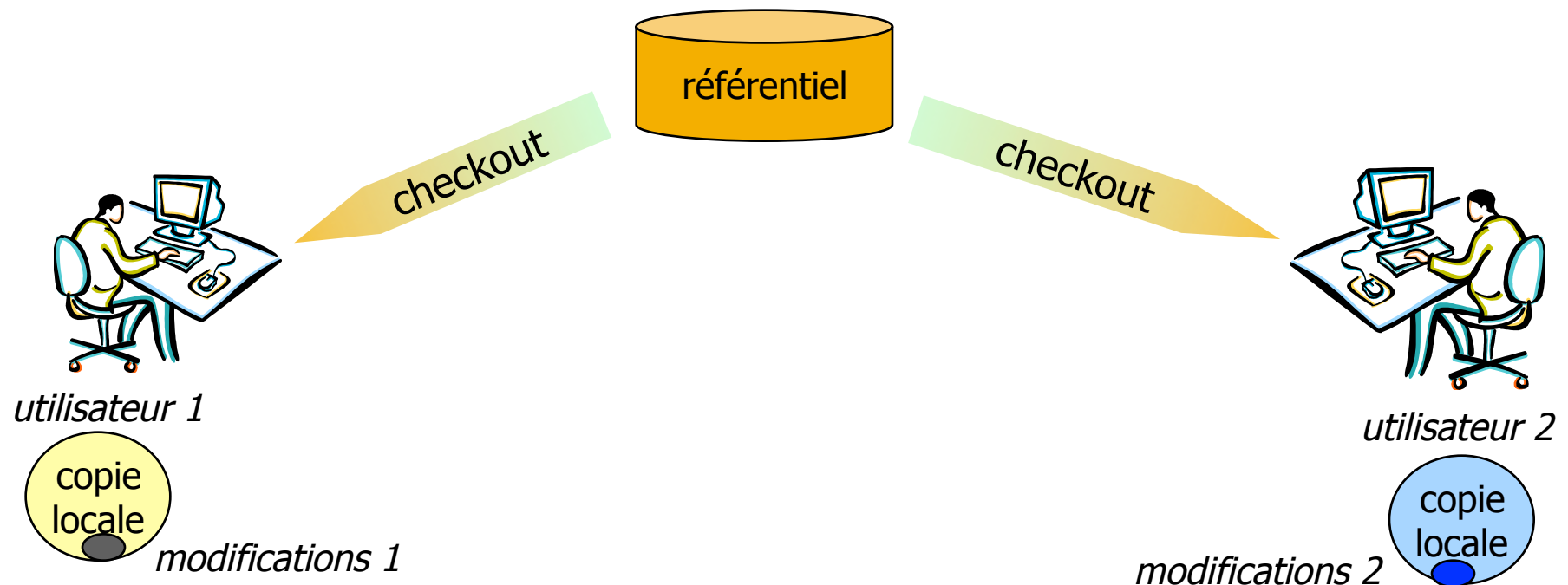
❑ Importer des sources

- `svn import rep_local /chemin/vers/referentiel [options]`
- `svn import myTree file:///usr/local/svn/newrepos/batchxml/trunk -m "Initial import"`



Récupérer une copie locale des sources

- ❑ `svn checkout chemin/vers/referentiel/et/projet [options]`
- ❑ `svn checkout http://svn.collab.net/repos/svn/trunk`



Propager ses changements

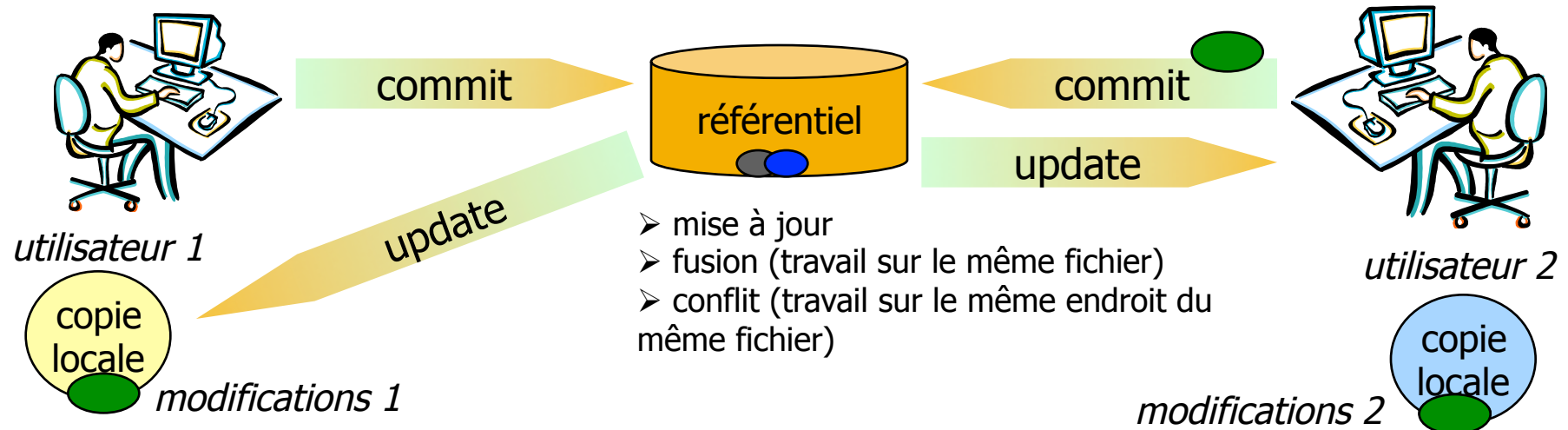
Mettre à jour par rapport à la base

□ 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`

Illustration : TortoiseSVN

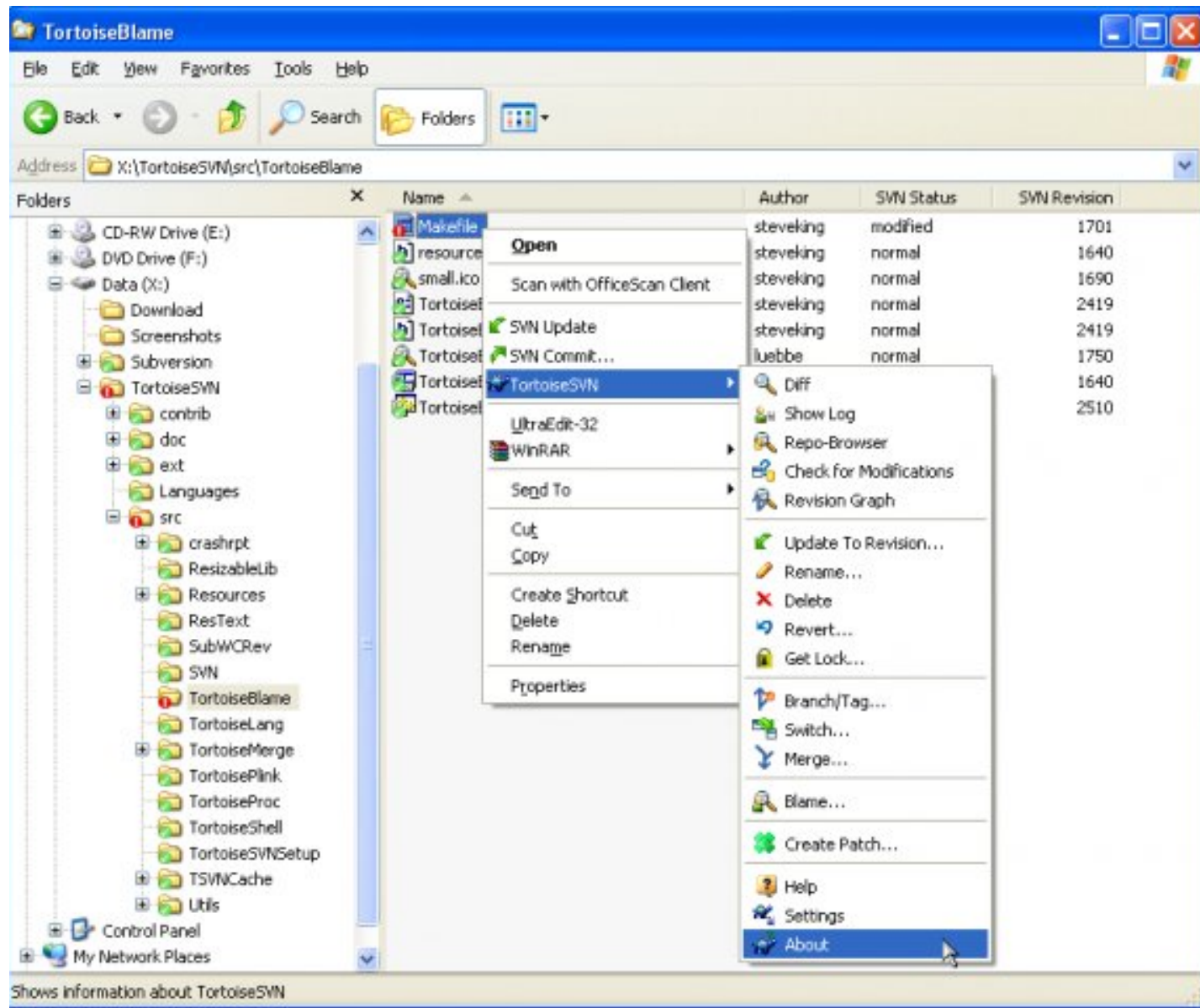


Illustration : plugin subclipse dans Eclipse

The screenshot shows the Eclipse IDE interface with the following components:

- Left Panel:** SVN (/core) tree view showing the project structure under 'core'.
- Editor:** SVNClientManager.java with a diff view comparing 'Local File' and 'Remote File (1500)'. The diff shows changes in a try-catch block.
- Bottom Panel:** SVN Resource History table for SVNClientManager.java.

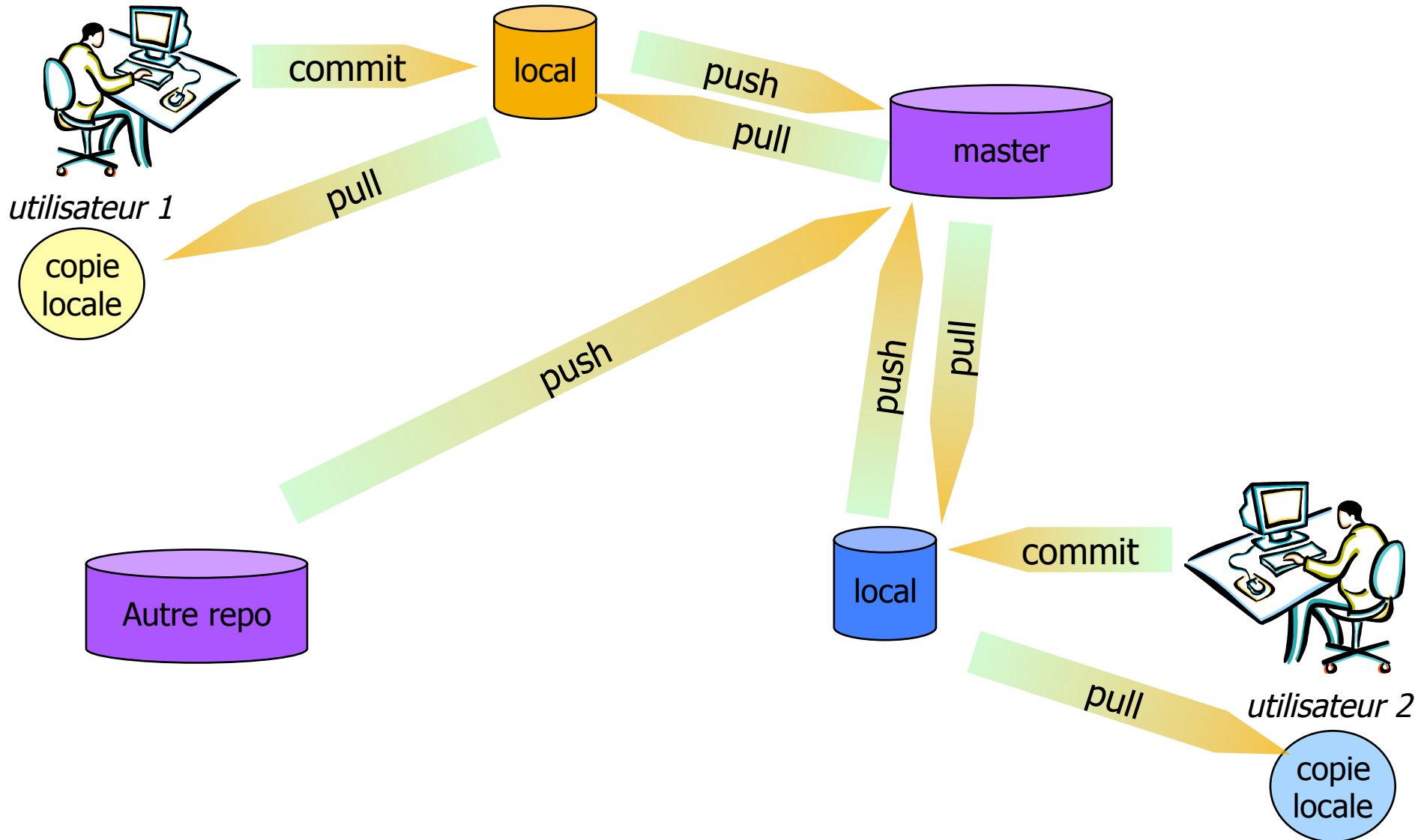
Revision	Date	Author	Comment
1493	8/3/05 11:...	markhip	Added support for JavaSVN as an adapter choi...
*1420	6/28/05 8...	markhip	Setting svn:eol-style on files that are mis...

- Synchronize with Repository
- Commit...
- Update
- Create Patch...
- Apply Patch...
- Show in Resource History
- Show Properties
- Set Property...
- Add Keywords...
- Add to Version Control
- Add to svn:ignore
- Branch/Tag...
- Switch...
- Merge...
- Configure Branches/Tags
- Copy...
- Export...
- Edit conflicts
- Revert...
- Mark as Deleted
- Mark Resolved
- Show pending operations
- Cleanup
- Disconnect...
- Share Project...

Centralisation vs. Distribution

- ❑ **SVN: pas d'accès à une base partagée sans connexion réseau**
- ❑ **GIT (et Mercurial, Bazaar):**
 - **« Distributed » Revision control**
 - ◆ Plus de serveur central (si le serveur tombe, le service est indisponible)
 - ◆ Chacun à une base locale, il existe une ou plusieurs bases distribuées sur des serveurs (une base est la base maître)
 - **On peut donc :**
 - ◆ commiter en local, sans que les autres développeurs soient tenus informés
 - ◆ puis « pousser » quand le moment est opportun vers la base maître
 - ◆ Updater depuis la base maître
 - **Avantages / inconvénients :**
 - ◆ Si la base maître tombe, on clone sa base locale sur une autre base
 - ◆ Complexe à utiliser : parfois on oublie de « pousser » vers la base...

Git : illustration



- ❑ <http://git-scm.com/>
- ❑ **Gestion de version distribuée open-source**
 - gestion des branches et des merges
- ❑ **Chaque copie de repository distribué est autonome**
 - gère son propre historique et ses versions
 - peut-être utilisé hors connexion indépendamment du repository original
 - Les sources sont publiables d'un repository vers un autre (push)
 - Le partage et la réintégration des sources (pull) est très simple

Principales commandes Git

- ❑ ***git init*** : crée un nouveau dépôt
- ❑ ***git clone*** : clone un dépôt distant
- ❑ ***git add*** : ajoute les nouveaux objets depuis le dernier commit. Les objets précédents restent inchangés
- ❑ ***git commit*** : marquer les changements en local
- ❑ ***git branch*** : crée une nouvelle branche de développement
- ❑ ***git merge*** : fusionne plusieurs branches de développement
- ❑ ***git push*** : publie le commit vers un repository distant
- ❑ ***git pull*** : récupère depuis un repository distant

Tags en git

- Pour tagger la tête de votre repository local avec un tag (nommé ESSAI ici) :
 - `git tag ESSAI`
 - `git push origin ESSAI`

- Vous pouvez aussi exécuter "`git push origin --tags`" pour pusher tous les tags existants à la fois.

- Si vous souhaitez déplacer le tag FINAL, vous pouvez d'abord l'effacer :
 - `git tag -d ESSAI`
 - `git push origin :refs/tags/ESSAI`

- Puis retagger comme vu précédemment.

- Pour information, je clonerai votre projet par une commande "`git clone`" classique, puis je récupérerai votre code par :
 - `git checkout FINAL`

- Si vous n'êtes pas capable d'exécuter la dernière commande sur un autre clone de votre git, c'est que le tag est mal fait, ou pas pushé...

Client Git : SourceTree (Mac, Windows)

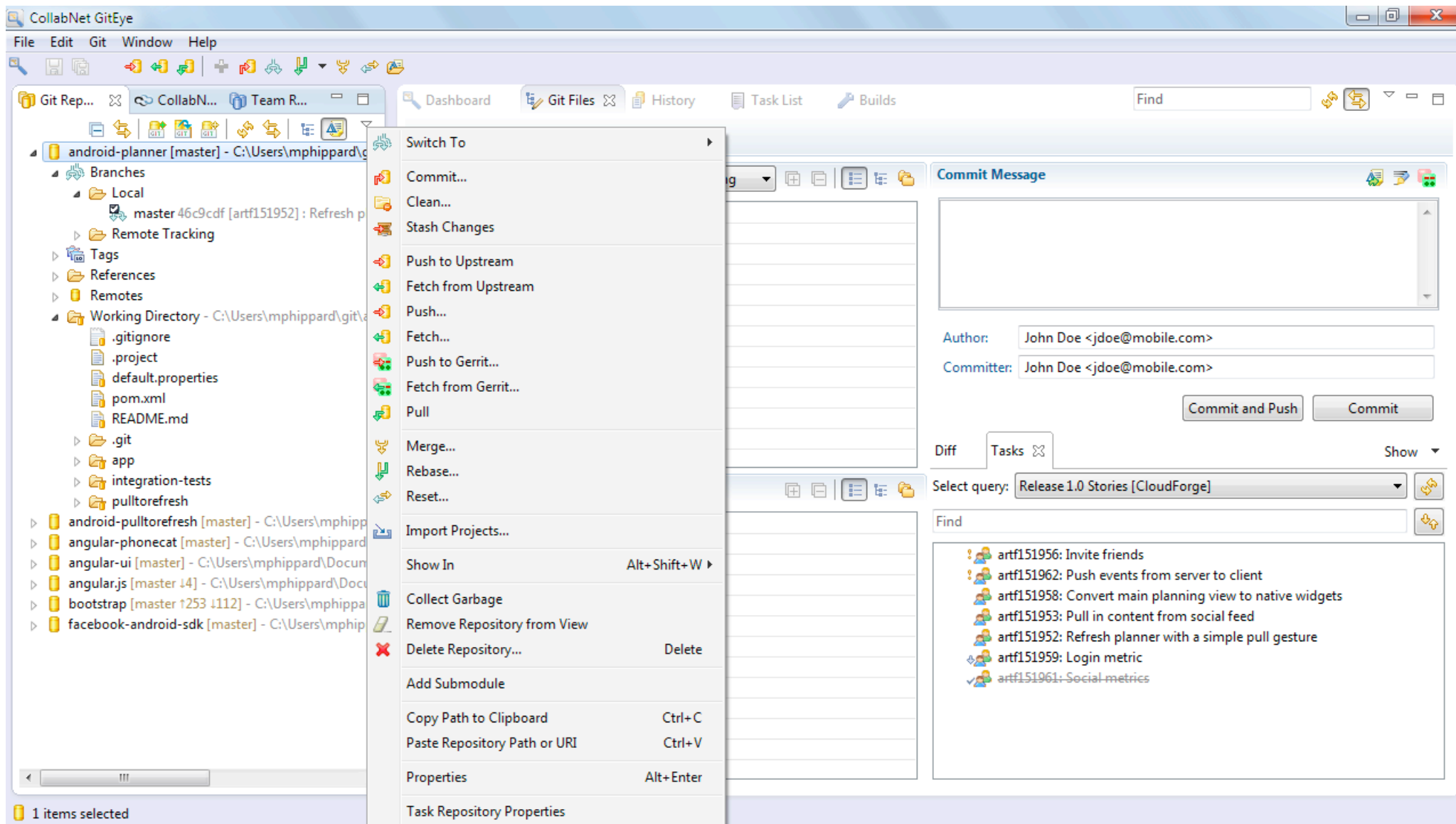
The screenshot displays the SourceTree Git client interface. The top toolbar includes icons for View, Commit, Checkout, Reset, Stash, Add, Remove, Add/Remove, Fetch, Pull, Push, Branch, Merge, Tag, Create Patch, Apply Patch, and Settings. The main window is divided into several sections:

- FILE STATUS:** Working Copy is checked.
- BRANCHES:** 7.x-1.x is the current branch.
- TAGS:** origin/8.x-1.x is selected.
- REMITES:** origin is expanded, showing branches like 4.7.x-1.x, 5.x-1.x, 6.x-1.x, 6.x-2.x, 7.x-1.x, 8.x-1.x, HEAD, and master.
- STASHES:** Empty.
- Commit Details:** Commit e774eadbbd690ac3a67748703fb197b8fcc33749 [e774ead] by Digidog, dated October 23, 2011 9:56:29 PM GMT+02:00. The commit message is "Digidog: added 8.x tag to module info (this is an early 8.x shot, please don't use this - for maintainers only)".
- Diff View:** Shows changes to link.info. Hunk 1 (Lines 1-6) includes:

```
1 1 name = Link
2 2 description = Defines simple link field types.
3 - core = 7.x
3 + core = 8.x
4 4 package = Fields
5 5
6 6 files[] = link.module
```

The bottom status bar shows the current branch (7.x-1.x), 1 Modified file, and 1 Not Tracked file. The Atlassian logo is visible in the bottom right corner.

Client Git : GitEye (Mac, Windows, Linux)



Attention !

❑ Ne pas utiliser EGit sous Eclipse

- Effectue des commandes dans votre dos
- Se plante
- Met le bazar dans tous les repositories



□ Git

- Pro-Git Book : <http://git-scm.com/book/fr>
- <http://www.cheat-sheets.org/saved-copy/git-cheat-sheet.pdf>
- http://www.git-tower.com/files/cheatsheet/Git_Cheat_Sheet_grey.pdf
- <http://git-scm.com/>

□ SVN

- The SVN Book: <http://svnbook.red-bean.com/>
- <http://subversion.tigris.org/>