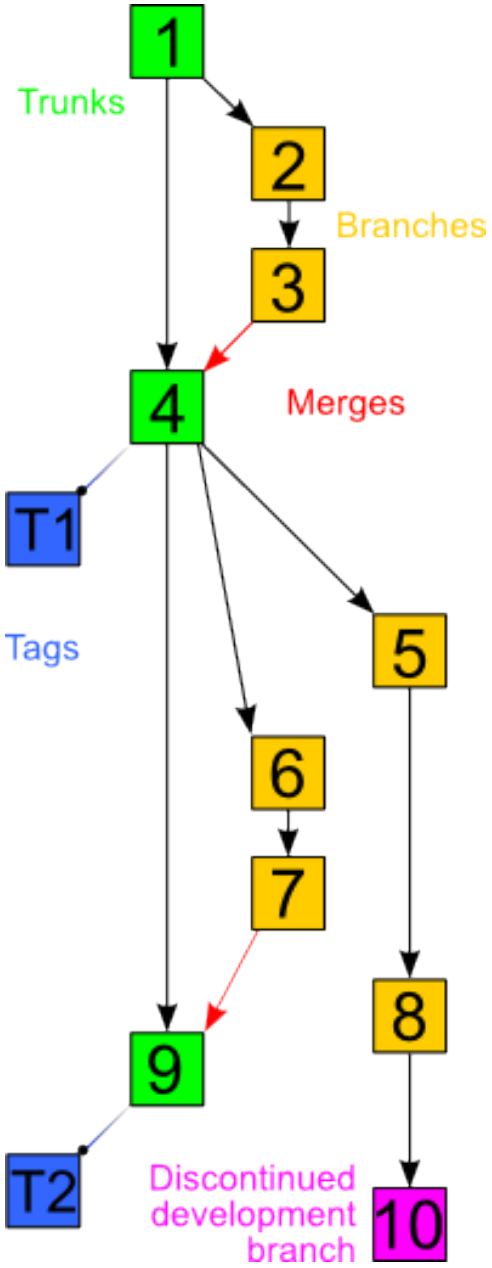
Systèmes de gestion de version


1
Trunks
2
Branches
3
4
Merges
T1
Tags
5
6
7
8
9
T2
Discontinued
development
branch
10

Problèmes

Comment conserver l'historique des modifications apportées chaque jour ... et revenir en arrière en cas de besoin ?

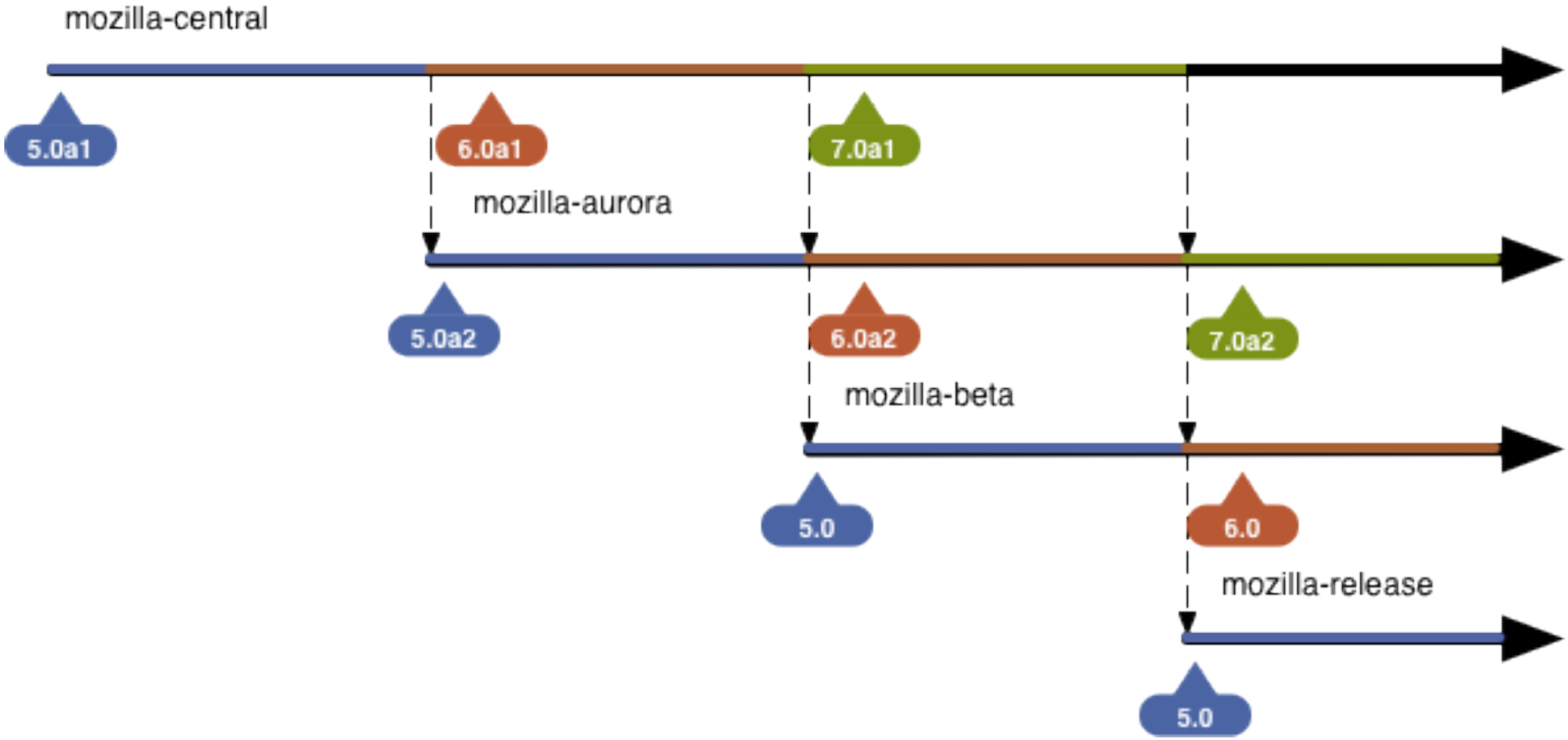
notions de régression et effets de bords

Problèmes

Comment partager le développement entre plusieurs personnes (écrasement, intégration) ?
- permettent un accès partagé au système de fichiers versionné

Problèmes

Comment gérer des distributions ?
- plusieurs versions peuvent coexister (branches de developpement, branches beta, branches production)


mozilla-central
5.0a1
6.0a1
7.0a1
mozilla-aurora
5.0a2
6.0a2
7.0a2
mozilla-beta
5.0
6.0
mozilla-release
5.0

Problèmes

Comment gérer les maintenances correctives & applicatives ?

Définitions

La gestion de versions consiste à maintenir l'ensemble des versions d'un ou plusieurs fichiers.

Essentiellement utilisée dans le domaine de la création de logiciels, elle concerne surtout la gestion des codes source.

Définitions

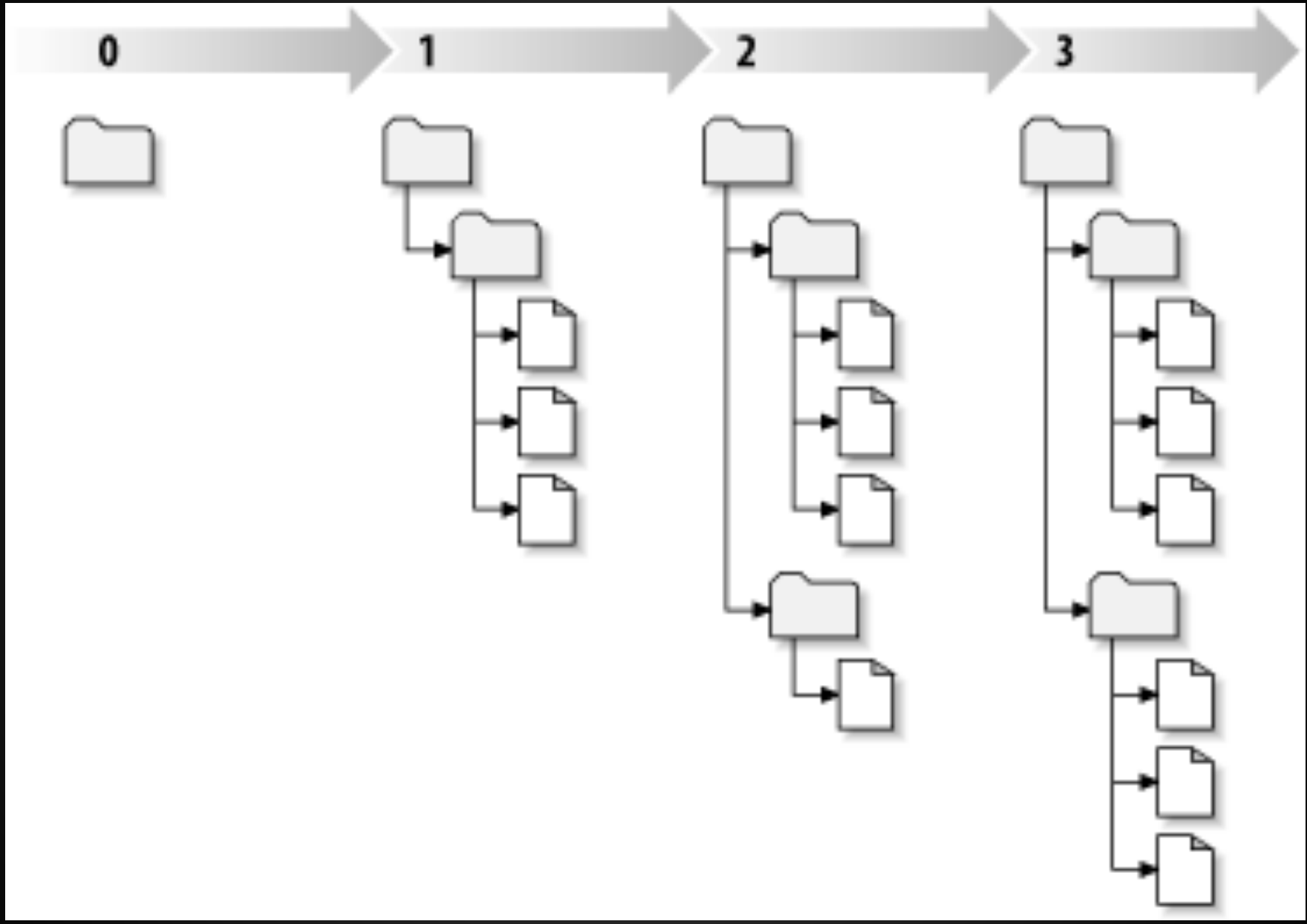
Un logiciel de gestion de versions agit sur une arborescence de fichiers.

Ce système permet de mutualiser un développement.

Un groupe de développeurs se servira de l'outil pour stocker toute évolution du code source.

Définitions

Un système de gestion de versions (VCS) va ajouter une nouvelle dimension au système de fichiers : **le temps**.


0
1
2
3

Définitions

Un système de fichiers traditionnel
est organisé de manière hiérarchique.

Le système de fichiers garde en outre une trace des méta-données, comme le nom du créateur du fichier, la date de création ...

Historique

Source Code Control System (SCCS)

Il a été développé en 1972 par Marc J. Rochkind au Laboratoires Bell pour les System/370 (en) d'IBM puis a été porté sur Unix pour être inclus dans la version standard ainsi que dans les spécifications de ce système d'exploitation.

Historique

CVS (Concurrent Versions System)

successeur de SCCS

originellement écrit par Dick Grune en 1986

Les sources convergent vers la même destination.

Historique

Subversion (SVN)

Il a été conçu pour remplacer CSV.

mêmes concepts que CSV (notamment sur le principe du dépôt centralisé et unique).

Réécriture, correction de bugs, ajouts de fonctionnalités

Historique

Bazaar 2005 : Ubuntu, mysql, emacs

Mercurial 2005 : Netbeans, Pyton, Vim

Git 2005 : Noyau Linux, Google, facebook, twitter

Historique

Bazaar 2005 : Ubuntu, mysql, emacs

Mercurial 2005 : Netbeans, Pyton, Vim

Git 2005 : Noyau Linux, Android, Gnome, twitter

Modèles client-serveur

il n'existe qu'un seul dépôt des versions qui fait référence.

Modèles client-serveur

il existe plusieurs dépôts pour un même logiciel.

chacun de travailler à son rythme
de façon désynchronisée des autres

Modèles client-serveur

il existe plusieurs dépôts pour un même logiciel.

chacun de travailler à son rythme
de façon désynchronisée des autres

Modèles client-serveur

AVANTAGES

permet de ne pas être dépendant d'une seule machine

permet aux contributeurs de travailler sans être connecté au gestionnaire de version ;

Modèles client-serveur

AVANTAGES

permet la participation à un projet sans nécessiter les permissions par un responsable du projet (les droits de commit/soumission peuvent donc être donnés après avoir démontré son travail et non pas avant) ;

Modèles client-serveur

AVANTAGES

permet toutefois de garder un dépôt de référence contenant les versions livrées d'un projet.

Notions communes

Dépôt

l'emplacement central où sont stockées toutes les données relatives aux projets gérés.

Notions communes

Dépôt

Le dépôt contient l'historique des versions des fichiers stockés, les logs enregistrés lors des modifications, les dates et auteurs de ces modifications, etc.

Notions communes

Dépôt

Un dépôt apparaît de l'extérieur comme un système de fichiers composé de répertoires au sein desquels on peut naviguer, lire et écrire selon les permissions accordées.

Notions communes

Copie de travail

La copie de travail est un répertoire situé en local sur le poste de l'utilisateur et qui contient une copie d'une révision donnée.

Elle contient les modifications qui seront ensuite importée vers le dépôt.

Notions communes

Trunk

la version centrale du programme, le développement principal.

C'està partir du tronc que l'on crée de nouvelle branche.

Notions communes

Branches

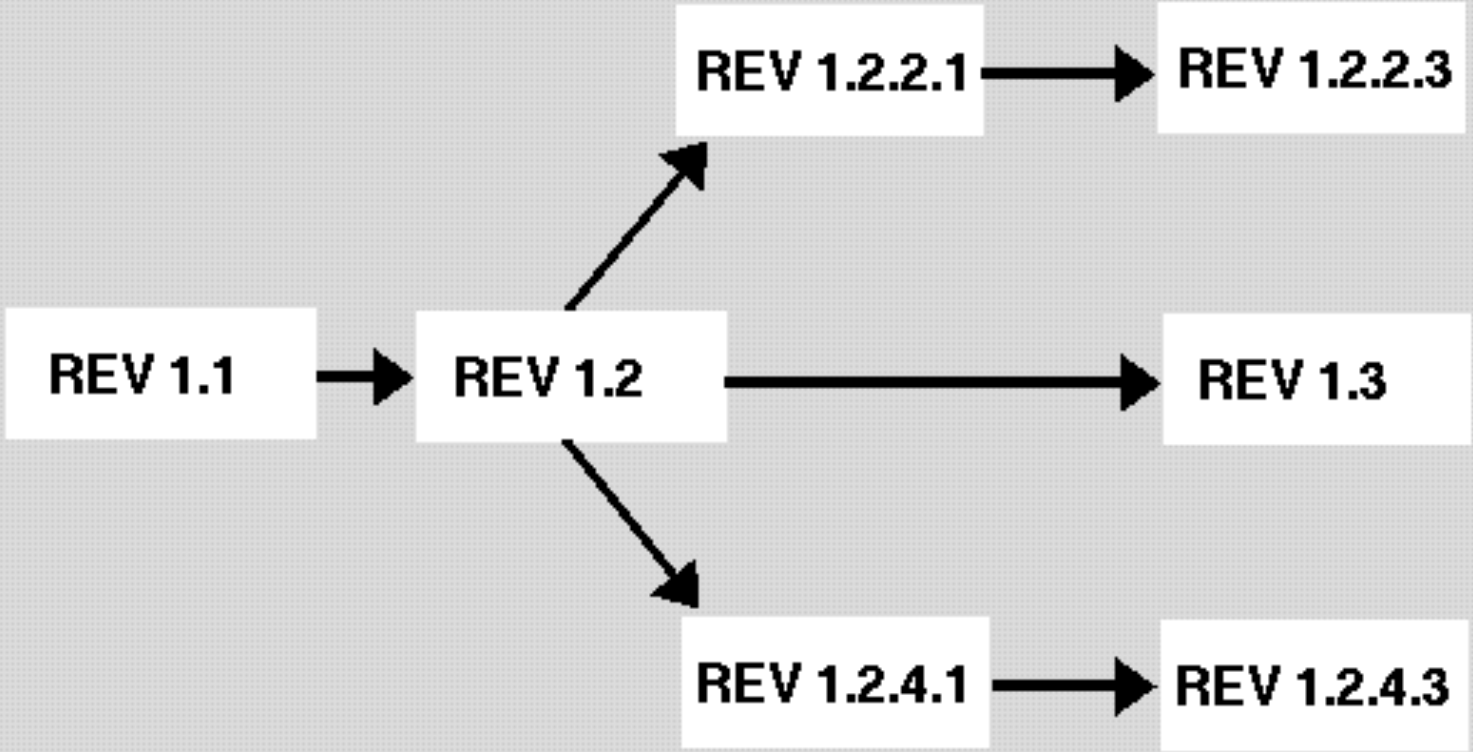
Une « branche » est créée lorsqu'un développement « secondaire » est mis en route (nouvelle fonctionnalité, nouvelle direction pour certains aspects du développement) .

Notions communes

Branches

Une branche peut, au bout d'un certain temps, soit être à nouveau fusionnée dans le « tronc », soit disparaître, soit donner lieu à un nouveau programme.

CVS Revision Numbers and Revision Branching



Notions communes

Checkout

Le *checkout* est l'opération qui consiste à récupérer pour la première fois les fichiers déjà existant au sein d'un projet du dépôt..

Le résultat est une copie de travail.

Notions communes

Commit

Mise à jour le dépôt à partir de la copie de travail locale. Une nouvelle révision est alors créée.
il faut que la copie de travail corresponde à la dernière version du dépôt.

Notions communes

Update

L'*update* consiste à synchroniser la copie de travail locale avec le dépôt en récupérant la dernière version des fichiers du dépôt.

C'est à cette occasion que des conflits de version peuvent apparaître :).

Notions communes

Tags

Permets de nommer un version

La notion de *tags* correspond en partie à celle de *release*, c'est à dire de marquage d'une certaine révision du projet comme composant une version du projet (1.0)

Notions communes

Révision

Chaque modification faite au dépôt constitue une révision.

Le numéro de révision est incrémenté à chaque opération.

Notions communes

Import

Elle consiste à placer dans le dépôt des fichiers locaux déjà existants pour y créer un nouveau projet.

Cette opération ne se fait en général qu'une fois par projet.

Notions communes

Change List

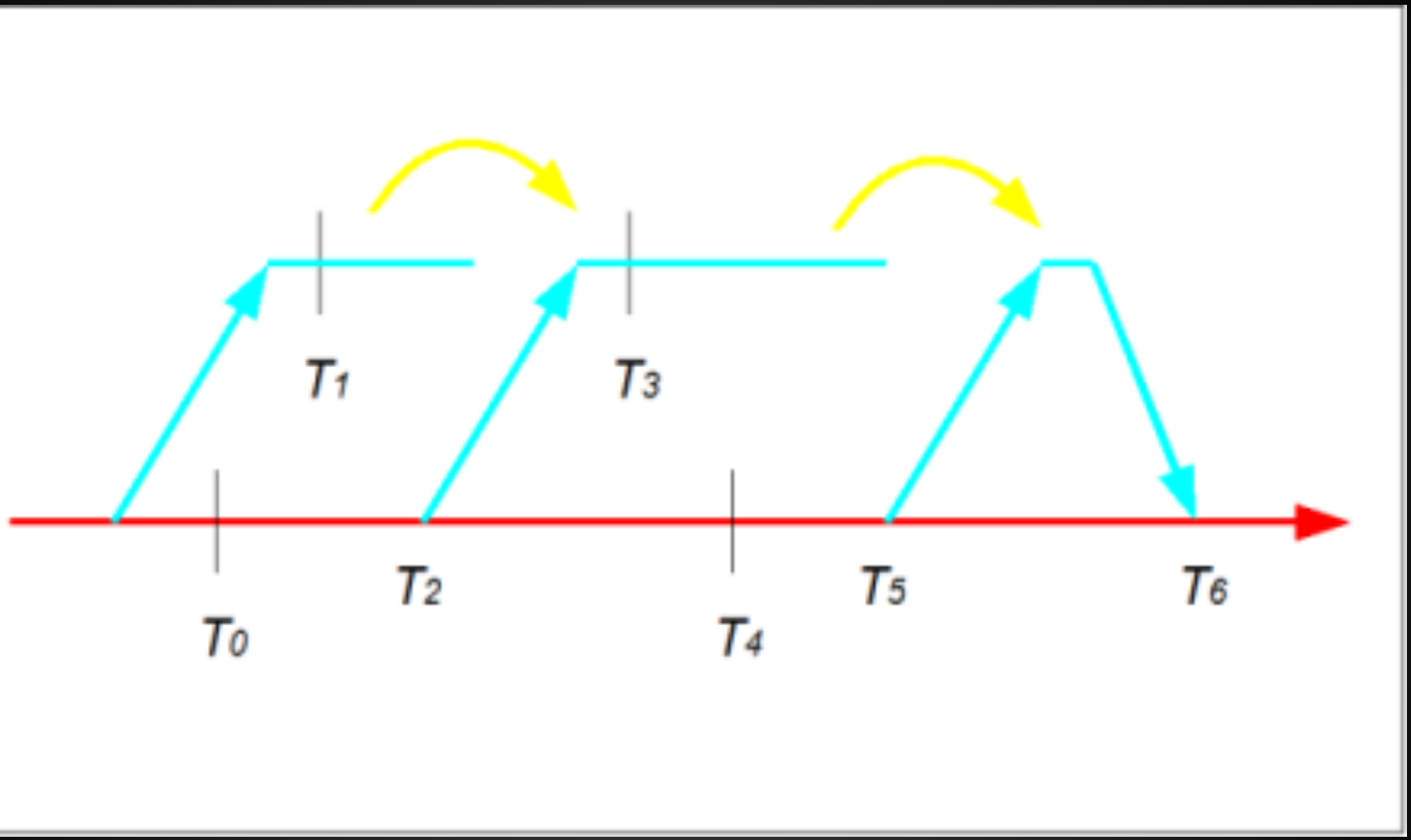
C'est la liste de l'ensemble des modifications effectuées pour une révision données.

Cette liste est consultable à tous moment ce qui permets de retrouver les modifications de chaque developpeur.

Notions communes

Merge

Permets de fusionner des branches entre elles ou de fusionner une branche avec la version principale (trunk).


T1
T3
T0
T2
T4
T5
T6

Notions communes

Conflit

certaines modifications peuvent être contradictoires (par exemple lorsque deux personnes ont apporté des modifications différentes à la même partie d'un fichier).

Notions communes

Contrôle de concurrence

Le contrôle de concurrence pessimiste impose à chaque utilisateur de demander un verrou avant de modifier une ressource

Notions communes

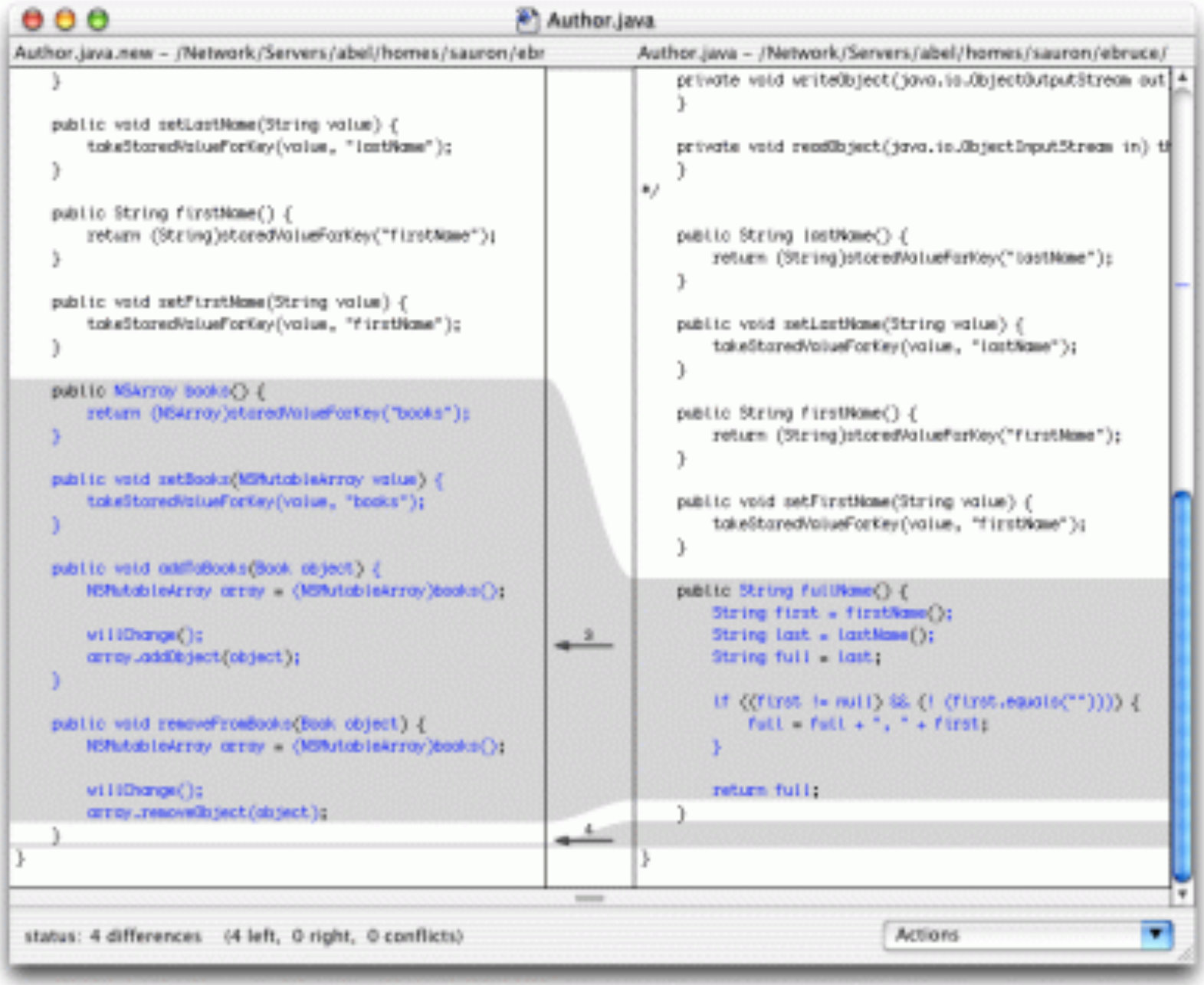
Contrôle de concurrence

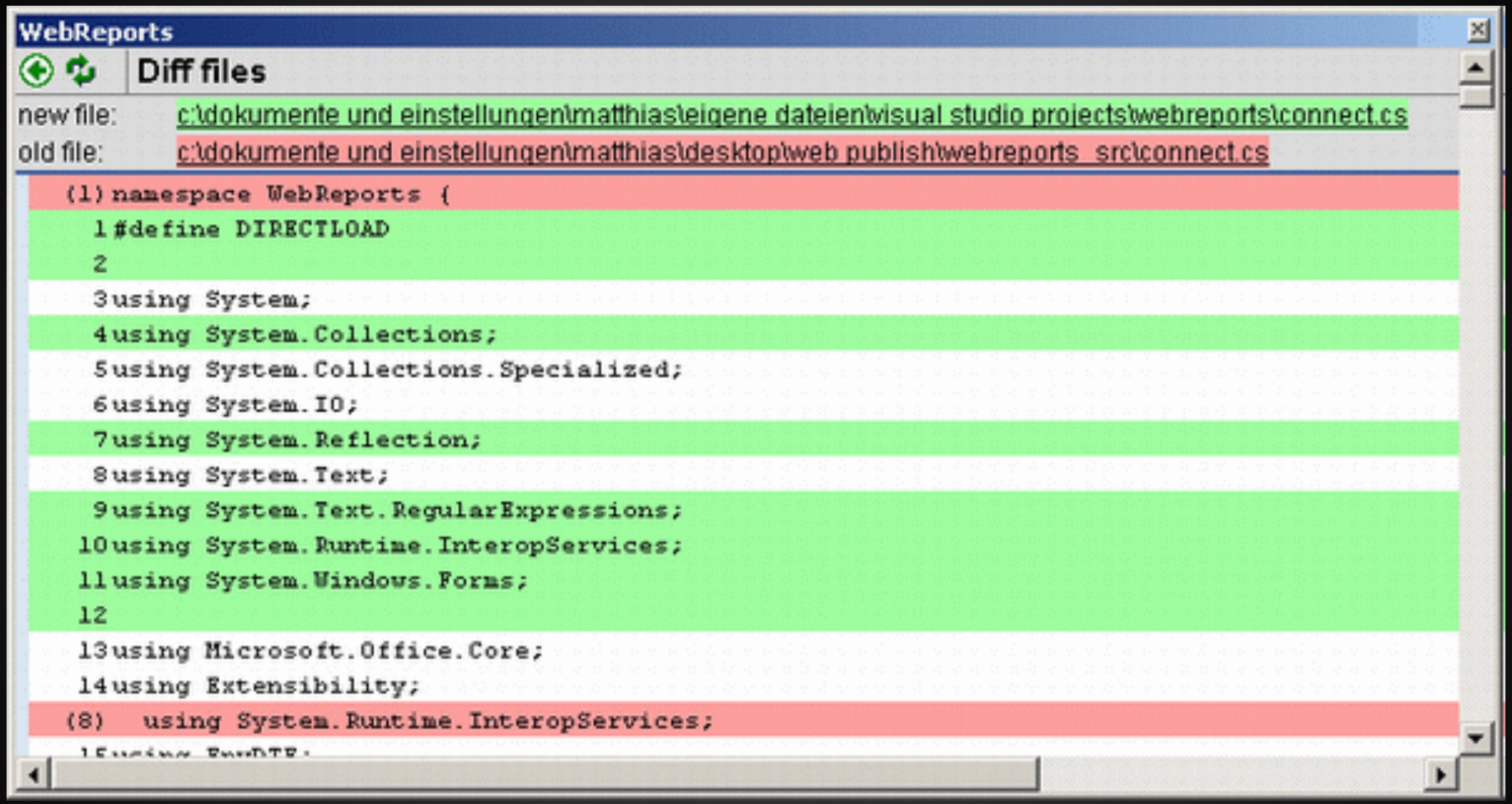
Le contrôle de concurrence optimiste permet à chaque utilisateur de modifier les données sans contrainte.

Notions communes

DIFF

Permets d'afficher les différence entre deux versions d'un fichier


Author.java
Author.java.new – /Network/Servers/abel/homes/sauron/ebr
Author.java – /Network/Servers/abel/homes/sauron/ebruce/
}
public void setLastName(String value) {
takeStoredValueForKey(value, "lastName");
}
public String firstName() {
return (String)storedValueForKey("firstName");
}
public void setFirstName(String value) {
takeStoredValueForKey(value, "firstName");
}
public NSArray books() {
return (NSArray)storedValueForKey("books");
}
public void setBooks(NSMutableArray value) {
takeStoredValueForKey(value, "books");
}
public void addToBooks(Book object) {
NSMutableArray array = (NSMutableArray)books();
willChange();
array.addObject(object);
}
public void removeFromBooks(Book object) {
NSMutableArray array = (NSMutableArray)books();
willChange();
array.removeObject(object);
}
}
private void writeObject(java.io.ObjectOutputStream out
}
private void readObject(java.io.ObjectInputStream in) th
}
*/
public String lastName() {
return (String)storedValueForKey("lastName");
}
public void setLastName(String value) {
takeStoredValueForKey(value, "lastName");
}
public String firstName() {
return (String)storedValueForKey("firstName");
}
public void setFirstName(String value) {
takeStoredValueForKey(value, "firstName");
}
public String fullName() {
String first = firstName();
String last = lastName();
String full = last;
if ((first != null) && (! (first.equals("")))) {
full = full + ", " + first;
}
return full;
}
}
3
4
status: 4 differences (4 left, 0 right, 0 conflicts)
Actions


WebReports
Diff files
new file: c:\dokumente und einstellungen\matthias\eigene dateien\visual studio projects\webreports\connect.cs
old file: c:\dokumente und einstellungen\matthias\desktop\web publish\webreports_src\connect.cs
(1) namespace WebReports {
1 #define DIRECTLOAD
2
3 using System;
4 using System.Collections;
5 using System.Collections.Specialized;
6 using System.IO;
7 using System.Reflection;
8 using System.Text;
9 using System.Text.RegularExpressions;
10 using System.Runtime.InteropServices;
11 using System.Windows.Forms;
12
13 using Microsoft.Office.Core;
14 using Extensibility;
(8) using System.Runtime.InteropServices;

Examples

Créer un nouveau dépôt

```
# Alice définit un nouveau dépot nommé "mon_depot" sur le serveur SVN

alice>svnadmin create mon_depot

# Alice crée une copie locale du nouveau dépot

alice>svn checkout file:///home/svn/mon_depot  mon_depot_local
```

Examples

Envoyer des modifications

```
# Alice a fini sa première version

alice> svn add article.db

alice> svn commit -m "Premiere version" article.db
```

Examples

Récuperer des modifications

```
# Qu'a fait Bob, exactement ?

alice> svn diff -r1.1 -r 1.2 article.db
```

Examples

Consulter les modifications

 # Bob récupère les changements

bob> svn update

 # Il effectue des modifications puis :

bob> svn commit -m "Meilleure explication des causes de panne" article.db

Examples

Revenir en arrière

 # Bob annulle les changements pour revenir à la version 1.1

bob> svn merge -rHEAD:1.1

Examples

Git : Créer un nouveau dépôt local

```
# Bob crée un nouveau dépôt

bob> cd /home/git/
bob> git init
```

Examples

Git : Récupérer un dépôt déjà existant

 # Bob récupère un dépôt

bob> git clone *git@github.com:user/test.git*

#bob ajoute à la copie de travail un nouveau fichier qu'il vient de créer

bob> git add bob_new_file.js

Examples

Git : commit

Bob envoie les données, il consulte auparavant la liste des modifications efffectuées.

bob> git status

création d'un nouveau commit (AU SEIN DE DEPOT LOCAL)
bob> git commit -a

Examples

Git : publier les modifications sur le référentiel

bob> **git push**

Alice récupère la dernière révision du dépôt contenant les dernières modifications de Bob:

alice> **git checkout** master

Il FAUT TOUJOURS connaitre le nom de la branche principale