Chapitre 4 : Gestion de configuration logicielle

Objectifs

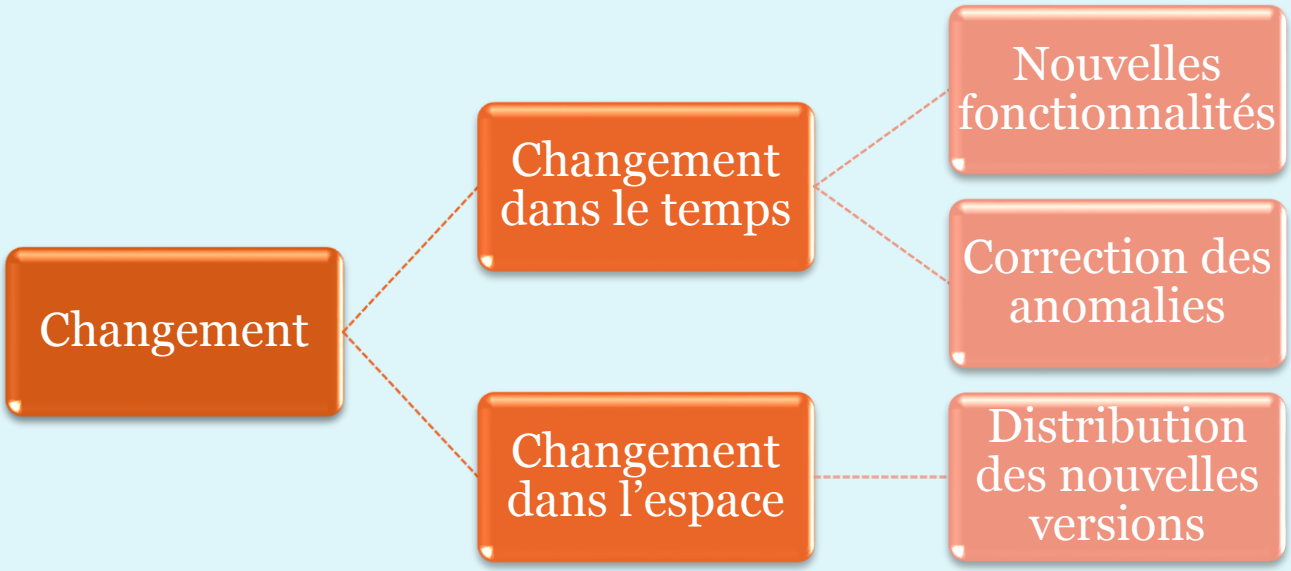
- Supporter la gestion de projet.
- Favoriser le travail collaboratif :
  - Plateforme adaptée.
  - Organisation du travail.
  - Meilleure synchronisation.

Plan du cours

- Gestion de configuration logicielle : Problématique.
- Processus de gestion de configuration.
- Architecture physique.
- Concepts de base:
  - Versionning .
  - Journalisation.
  - Gestion de conflits d'accès.

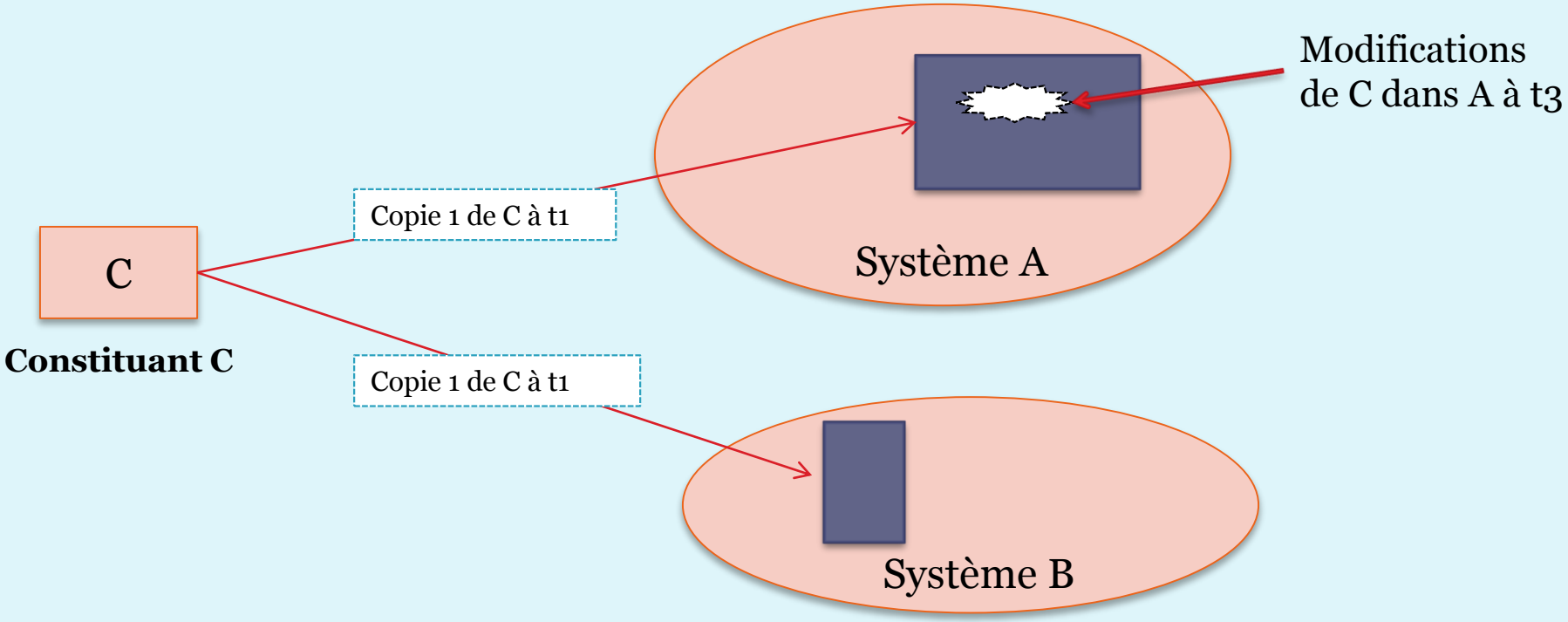
Gestion de configuration logicielle : Problématique(1/4)

- Evolution du logiciel à travers différents changements :



Gestion de configuration logicielle : Problématique(2/4)

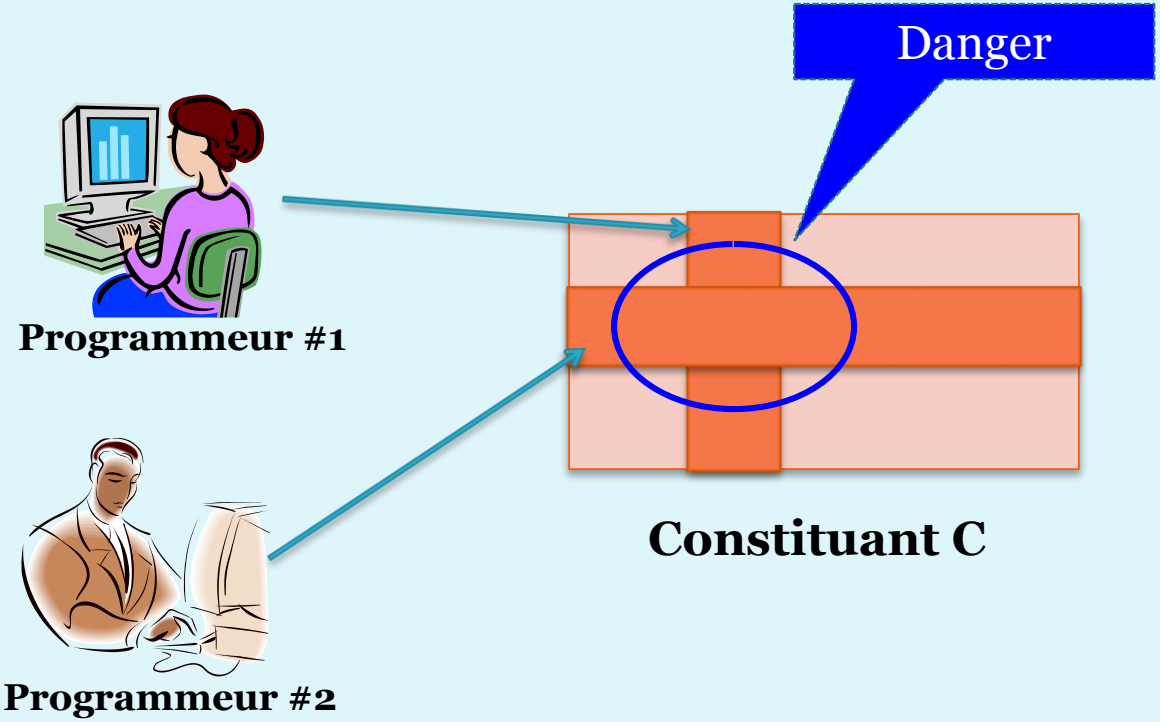
PB#1 : Problème de la double maintenance



Il faut minimiser les duplications car les copies multiples peuvent diverger.

Gestion de configuration logicielle : Problématique(3/4)

PB#2 : Problème du partage des données



Le programmeur #1 et #2 travaillent tous les deux sur le même constituant C

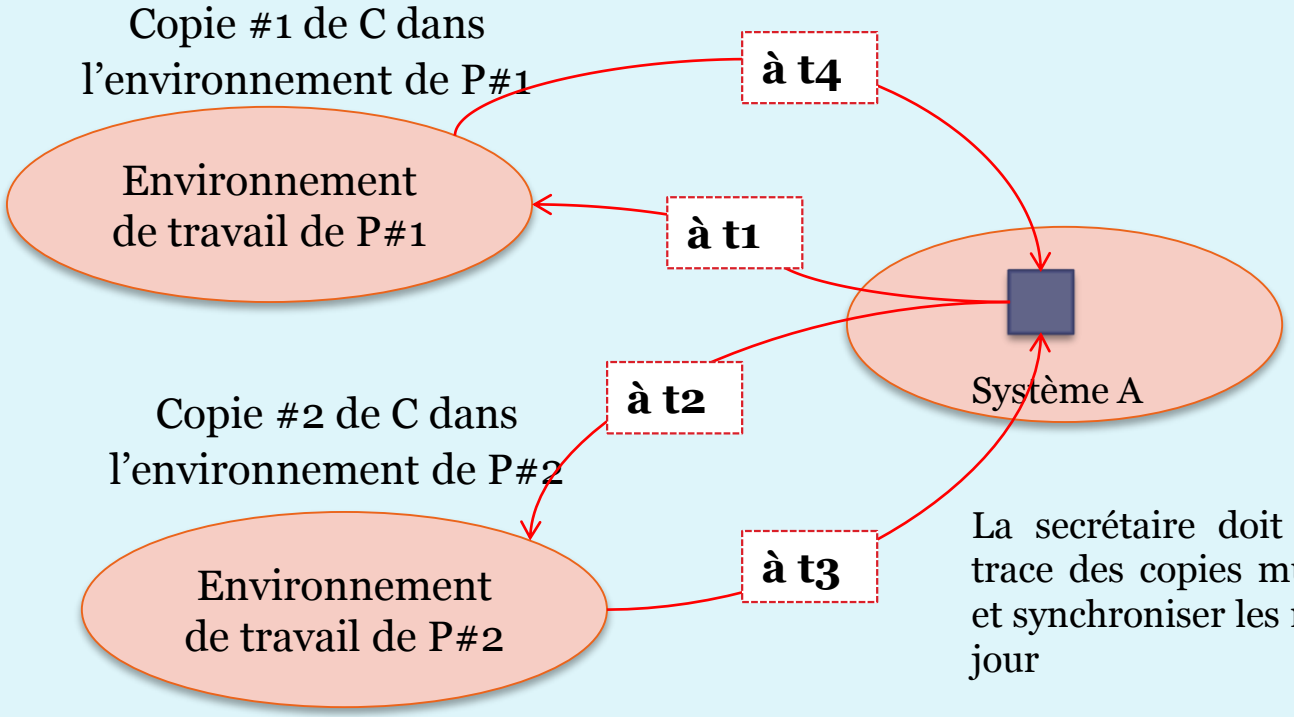
Les erreurs du programmeur #1 peuvent bloquer le programmeur #2.

Gestion de configuration logicielle : Problématique (4/4)

PB#3 : Problème des mises à jour simultanés

Programmeur P#1

Programmeur P#2



La secrétaire doit garder trace des copies multiples et synchroniser les mises à jour

Pour donner du confort à P#1 et à P#2, et éviter le problème PB#2, C a été dupliqué, ce qui nous ramène au problème PB#1 ⟹ Gérer le dilemme

Processus de gestion de configuration

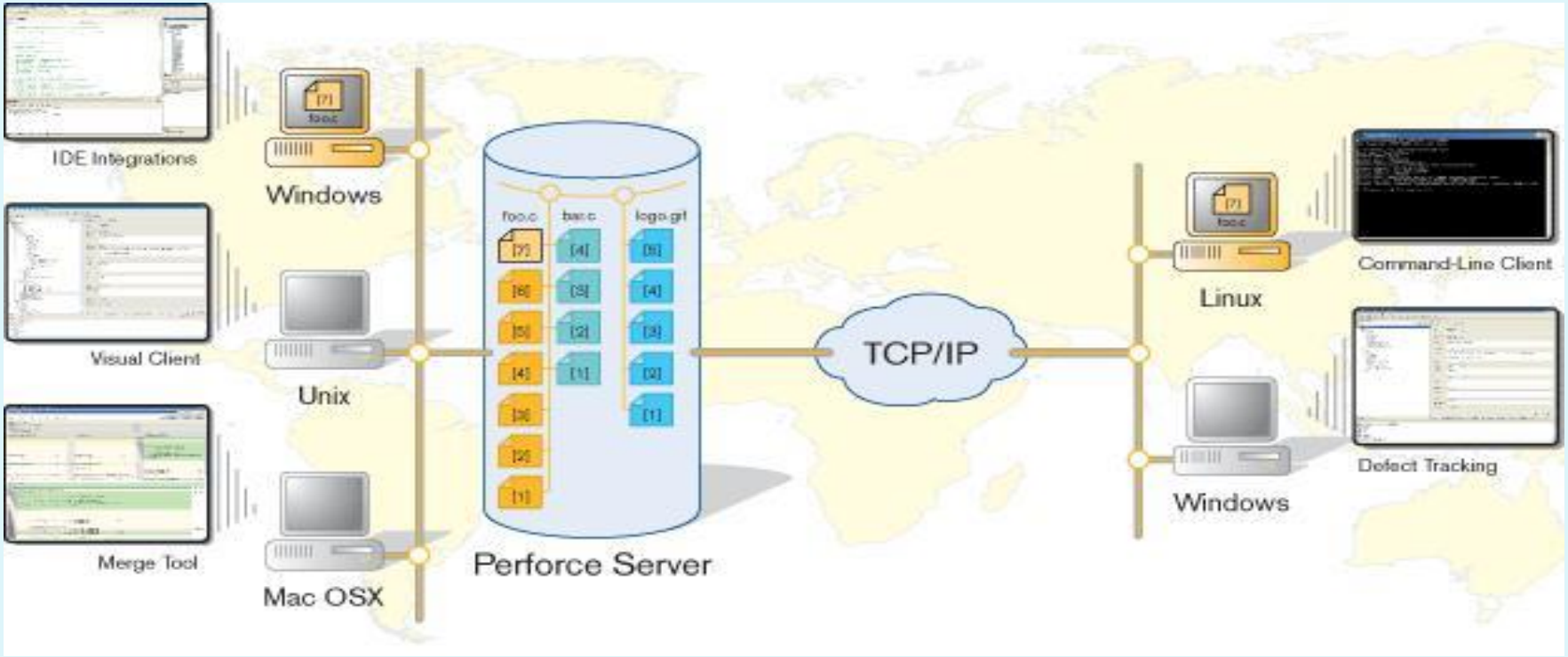
- Discipline de management de projet qui permet de définir, d'identifier, de gérer et de contrôler les articles de configuration tout au long du cycle de développement d'un logiciel. **(ISO 10007).**

- Article de configuration : Ensemble de matériels, de services ou un sous-ensemble défini de ceux-ci, retenu pour la gestion de configuration et traité comme une seule entité dans le processus de gestion de configuration. **(ref1:D.Jacquin).**

Architecture physique

Gestionnaire de configuration logicielle (GCL) centralisé	Gestionnaire de configuration logicielle (GCL) décentralisé
• Un seul dépôt des versions faisant référence. • Exemples : CVS, Subversion (SVN).	• Plusieurs dépôts pour un même logiciel. • Exemples : Git, Bitbucket.

Exemple d'architecture physique d'un systèmes de Gestion de configuration: Perforce



Perforce : un outil de gestion de configuration logicielle construit autour d'une architecture client/serveur

Concepts de base

- **Trois concepts de base :**

Versionning	• Suivre les évolutions dans le temps de la configuration.
Journalisation	• Archiver les états livrés successifs.
Gestion de conflit	• S'assurer que chacun des états livrés est cohérent et complet.

Concepts de base : Versionning(1/2)

- La numérotation à trois chiffres servant à identifier un ensemble cohérent de modules.
- Un label ou une étiquette de révision (Tag ou Flag) : définir une étiquette textuelle qui peut être associée à une révision spécifique d'un projet mis à jour.

Numéro de version. **Numéro de révision** . **Numéro de correction**

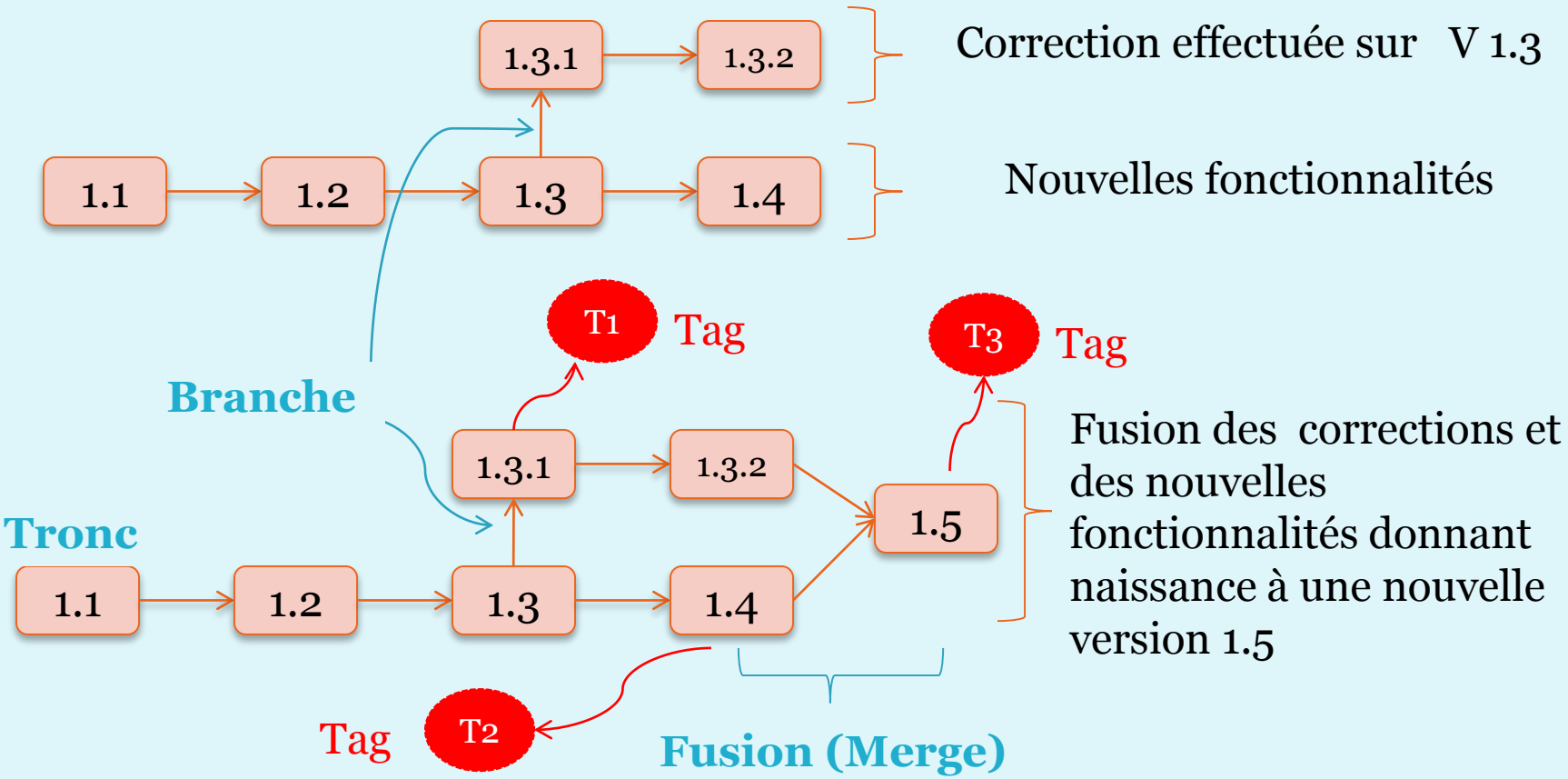
modification majeure

Ajout des nouvelles fonctionnalités

Correction des fautes

Concepts de base : Versionning(2/2)

- **Exemples d'évolutions :**

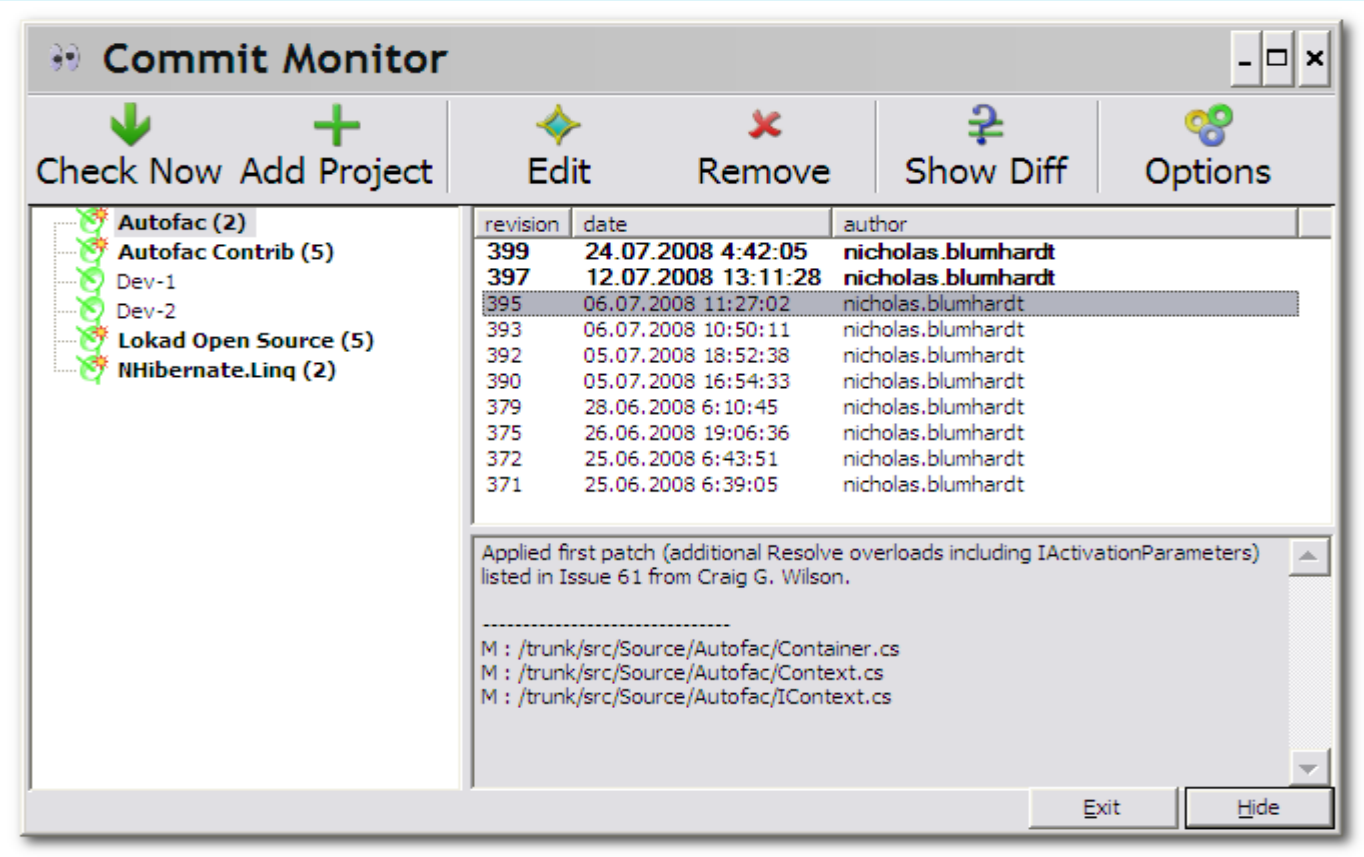


Concepts de base : Journalisation(1/2)

- Contrôler les modifications apportées à chaque fichier ou composant d'un logiciel :
  - **Historique d'accès :** Qui ? Quand? Nature (création /modification / suppression).
  - **Détail de la modification :** Par des outils intelligents de comparaison des sources.

Concepts de base : Journalisation(2/2)

- **Exemple :** Journalisation par Commit Monitor



Concepts de base : Gestion des conflits(1/2)

Développement en équipe

↓

Partage d'une base de travail commune

↓

Coordination entre les équipes de développement

↓

Problème : possibilité de modifications contradictoires

« CONFLIT »

↓

Solution 1 : Contrôle de concurrence pessimiste

Solution 2 : Contrôle de concurrence optimiste

Concepts de base : Gestion des conflits(2/2)

Solution pour la gestion des conflits	Description
Le contrôle de concurrence ***pessimiste***	impose à chaque utilisateur de demander un **verrou** avant de modifier une ressource ; ce verrou lui garantit qu'il sera le seul à modifier la ressource
Le contrôle de concurrence ***optimiste***	permet à chaque utilisateur de modifier les données sans contrainte. Au moment d'appliquer ces modifications le système vérifie si un autre utilisateur n'a pas déjà posté des modifications pour ces mêmes données