



Gestion des configurations



Objectifs et plan du cours

- Introduire les concepts de base de la gestion des configurations
- Introduction à l'utilisation (très basique) de l'outil cvs



A propos de changements...

- Un logiciel évolue en subissant une suite de changements
- Les changements sont rendus inévitables
 - ◆ par le système lui-même
 - ◆ par l'environnement du système (par exemple les erreurs)
- Un changement peut influencer
 - ◆ les délais de livraison
 - ◆ les coûts de la réalisation
 - ◆ la cohérence et la complétude du système
- Plusieurs acteurs sont concernés par un changement



Changements : nature et origines

■ Changements/modifications dans le temps

◆ Nouveaux besoins liés à la maintenance

- ☞ Évolution, adaptation du système à son environnement (nouvelles technologies)
- ☞ Corrections suite à des rapports d'anomalies signalées par les utilisateurs

◆ Acteurs concernés

- ☞ Essentiellement la maîtrise d'œuvre : équipes de maintenance (mais sous contrôle de la maîtrise d'ouvrage qui décide ce qui doit être fait)

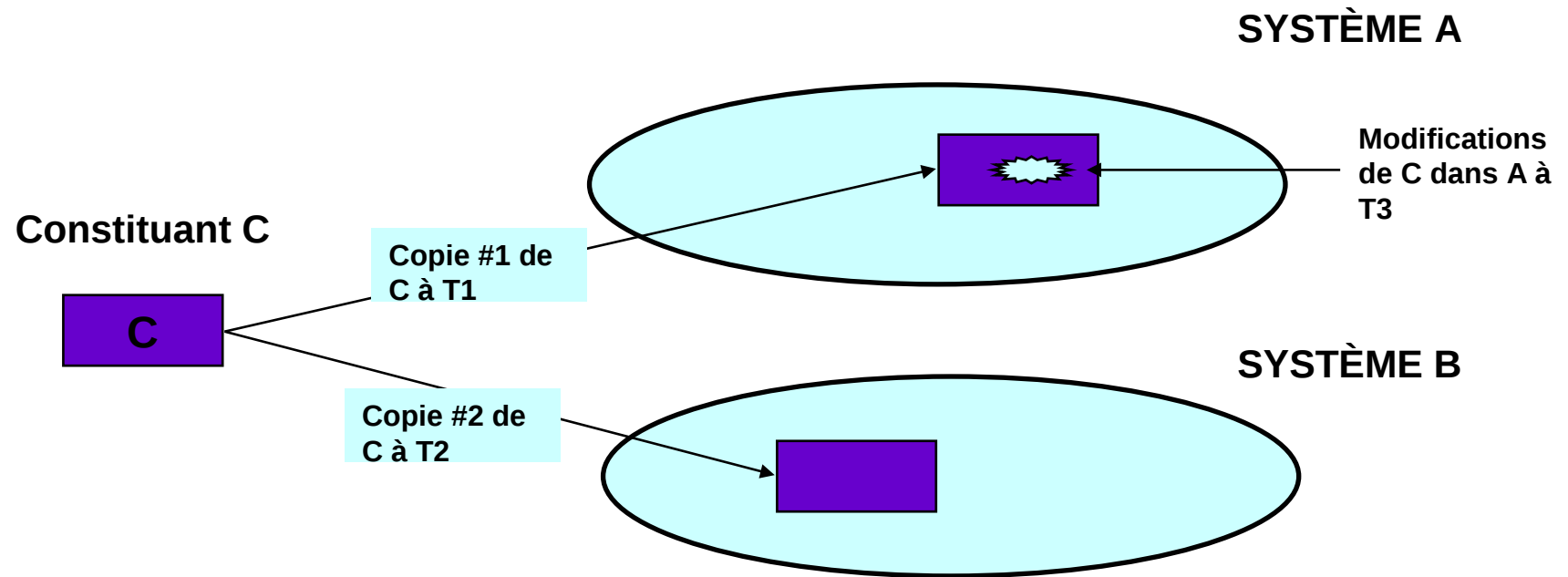
■ Changements/modifications dans l'espace

◆ Distribution de nouvelles versions de logiciel

◆ Acteurs concernés

- ☞ Essentiellement les équipes de support (« hot line ») et d'exploitation
- ☞ Les utilisateurs

Problème de la double maintenance



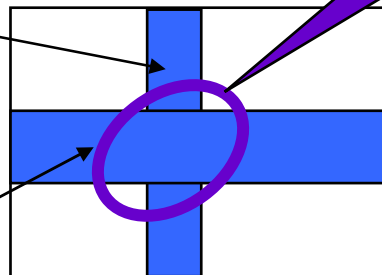
IL FAUT MINIMISER LES DUPLICATIONS CAR, *INÉVITABLEMENT*, LES COPIES MULTIPLES DIVERGENT ; L'AUGMENTATION DU COUT EST INÉLUCTABLE

Problème du partage des données

Programmeur #1



Programmeur #2



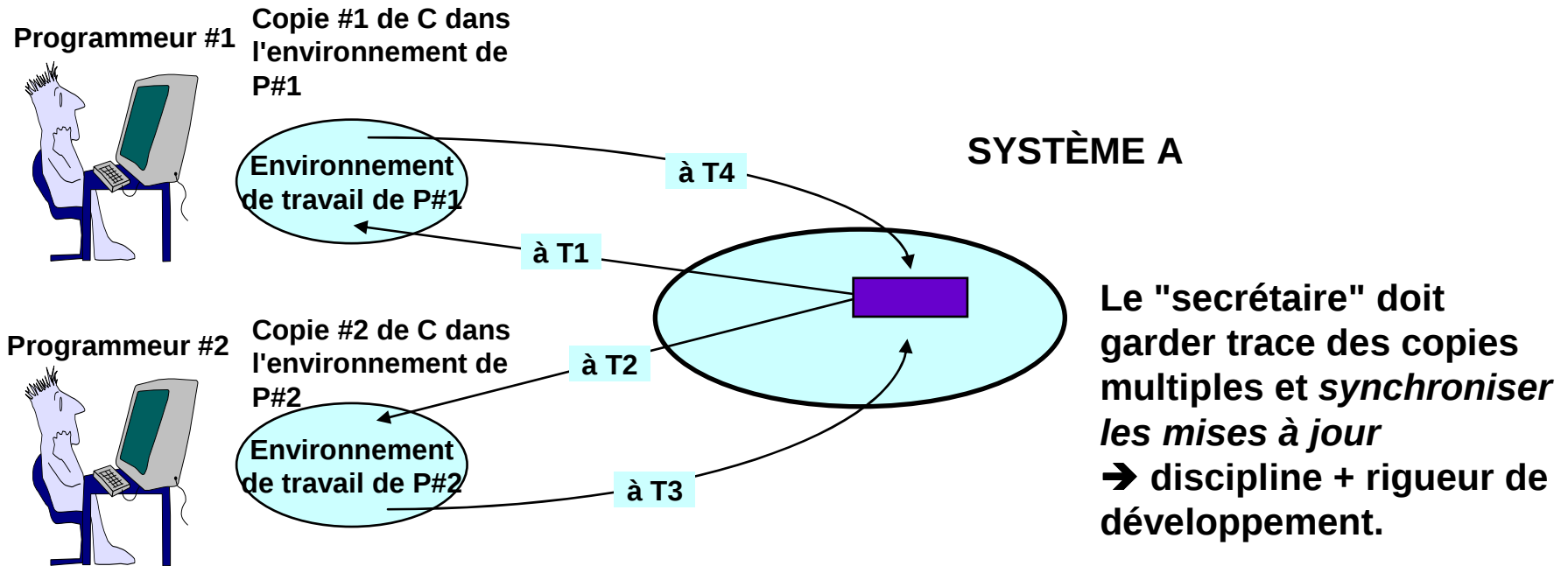
Constituant C

DANGER

LES PROGRAMMEURS P#1 ET P#2
TRAVAILLENT TOUS DEUX SUR LE
MÊME CONSTITUANT C

LES ERREURS DE P#1 PEUVENT BLOQUER P#2 ; LE RETARD EST INÉLUCTABLE

Problème des mises à jour simultanées



POUR DONNER DU CONFORT A P#1 ET A P#2, ET ÉVITER LE PB#2, C A ÉTÉ DUPLIQUÉ, CE QUI NOUS RAMÈNE AU PB#1 → GÉRER LE DILEMME !?



CONFIGURATION

- Ensemble cohérent de composants permettant, a un instant donné, d'éditer une version fonctionnelle complète du système.
 - ◆ C'est la garantie de l'intégrité du système.
 - ◆ La granularité de la configuration est un paramètre économique.
 - ☞ Valeur du produit (et de ses constituants), nature du risque, etc. ...
 - ☞ Les éléments « à risque » doivent être répertoriés



■ Difficultés de la gestion des configurations

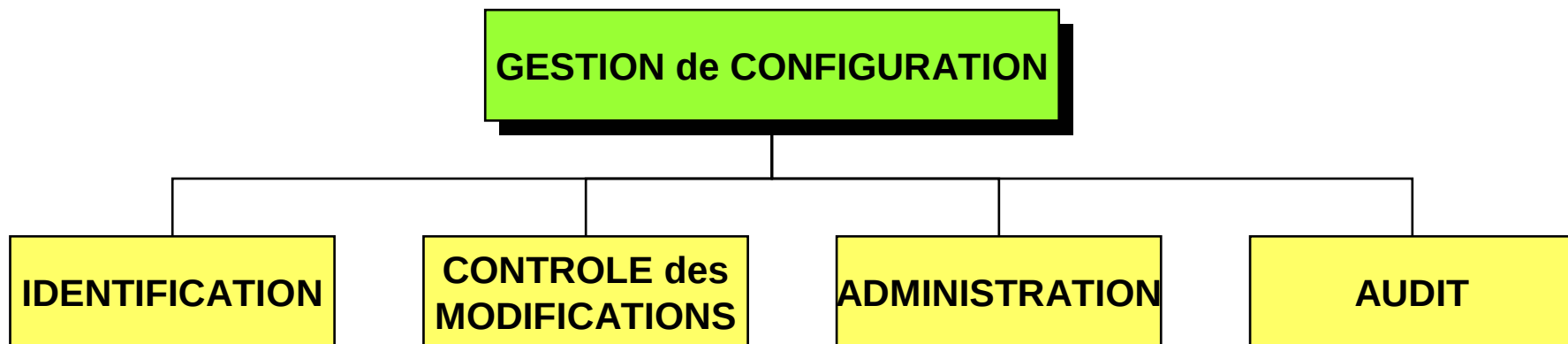
- ◆ Nombre d'objets à gérer → Dépend de la granularité et du type de nomenclature
- ◆ Variété des objets
- ◆ Variété des supports d'archivage et de stockage.
- ◆ Caractéristiques "molles" du logiciel → Dépendances fonctionnelles, canaux cachés, etc.
- ◆ Durée de vie des équipements, des outils, ...
- ◆ Organisation du développement (+ ou - normalisé)
- ◆ Les acquisitions en logiciel et matériel :
 - ↳ ce sont des boîtes noires dont il est souvent difficile de cerner les contours

Le processus de gestion de configuration

■ Activités :

- ◆ définir les composants de la configuration et toutes leurs relations
- ◆ suivre les évolutions dans le temps de la configuration
- ◆ archiver les états livrés successifs
- ◆ s'assurer que chacun des états livrés est cohérent et complet

■ 4 fonctions principales :





Le processus de gestion de configuration

- Ensemble de procédures a suivre
 - ◆ procédures automatiques qui s'appuient sur des outils :
 - ☞ au niveau du système d'exploitation :
 - Système de gestion de fichiers et bibliothécaire.
 - SGBD
 - Éditeurs de textes
 - Compilateurs, traducteurs de langages
 - Éditeurs de liens, relieurs.
 - ☞ Unix et/ou Windows :
 - SCCS, MAKE, RCS, CVS
 - ☞ Progiciels
 - Cf. outil ClearCase de RATIONAL
 - ☞ Dictionnaires de données et/ou référentiel
 - ◆ procédures manuelles
 - ☞ formation des équipes
 - ☞ discipline et rigueur individuelle, sens de l'équipe, sens du projet
- la bonne mise en oeuvre d'une gestion de configuration nécessite un bon niveau de maturité de l'organisation de développement
 - ◆ c'est un jeu collectif qui implique la coopération de nombreux acteurs



Identification

- Quels sont les éléments de la configuration?
 - ◆ Fichiers de code
 - ◆ Documents
 - ◆ Fichiers – programmes de test
 - ◆ ...

- Comment ces éléments sont identifiés?
 - ◆ Définir une *nomenclature*



Nomenclature

■ Exemple

◆ Identification à l'aide de trois identifiants

☞ Numéro de version

- Changement = modification majeure

☞ Numéro de révision

- Changement = amélioration performances, amélioration présentation, éventuellement nouvelle fonctionnalité

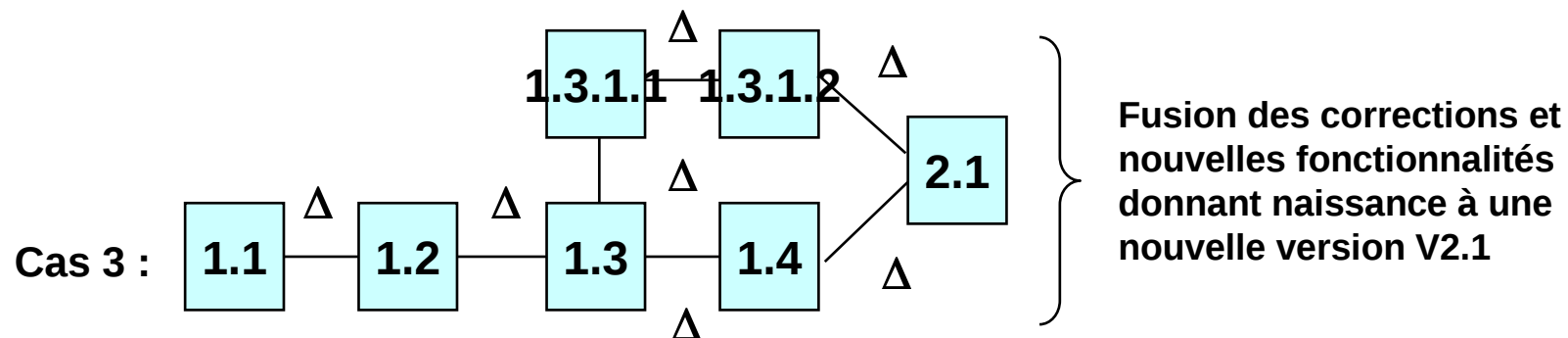
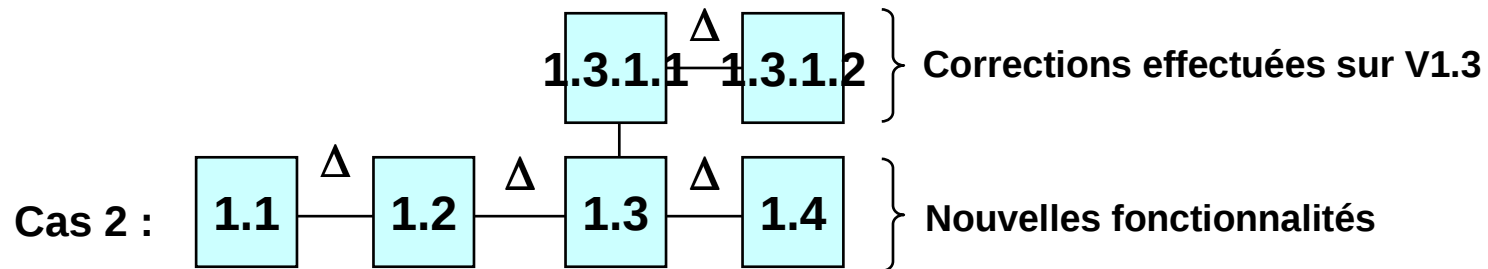
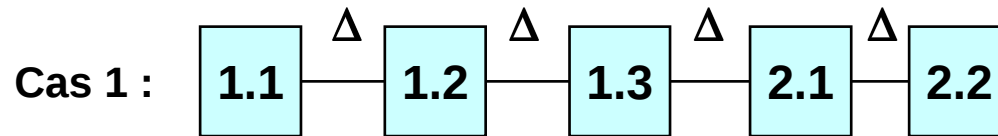
☞ Numéro de correction

- Changement = correction de fautes

■ Exemples d'évolution

◆ 1.0.0 – 1.0.1 – 1.0.2 – 1.1.0 – 1.2.0 – 1.2.1 – 2.0.0

◆ 1.0.0 – 1.1.0 – 1.1.1 – 1.1.2 – 2.0.0





Contrôle des modifications

- Qui décide du changement de nom d'un élément?
 - ◆ Nécessité d'un responsable de la gestion des configurations
 - ◆ Responsabilité du chef de projet
- Comment l'équipe du projet est au courant?
 - ◆ Procédure de diffusion – consultation de la configuration
- Est-on capable à tout moment de produire une configuration donnée à partir des éléments de la configuration?
 - ◆ Recompiler et livrer une ancienne version



Une brève introduction à CVS



L'outil CVS (introduction)

- Outil standard sous unix et linux (voire windows)
- Permet de gérer des configurations « concurrentes »
- Document cvs.ps disponible sous `~parissii/TAGL/Configurations-CVS`
- Principe du travail pratique demandé:
 - ◆ Les commandes ne sont pas décrites en détail
 - ◆ CONSULTER IMPERATIVEMENT LE MANUEL pour découvrir leur utilisation



Commandes de base

- Au démarrage du projet, création d'un « répertoire de référence » où sont stockés les éléments de configuration du projet
 - ◆ *init*
- Chaque membre de l'équipe
 - ◆ Crée son propre répertoire de travail
 - ☞ *Checkout*
 - ◆ Travaille en local
 - ◆ Est informé du statut des éléments qu'il manipule, par rapport au répertoire de référence
 - ☞ À jour
 - ☞ Pas à jour (la copie originale a été modifiée)
 - ☞ Modifié localement
 - *status*
 - ◆ Met à jour ses fichiers ou le répertoire de référence
 - ☞ *checkin, update*
 - ☞ Possibilité de définir des droits d'accès (*fichiers readers, writers*)



■ Nomenclature

- ◆ CVS attribue des identifiants (ex. 1.1.1.1) aux différents éléments suivant un principe interne (difficilement exploitable)
- ◆ Possibilité d'attribuer des identifiants définis par l'utilisateur
 - 🔑 *Tag*

■ Exporter une configuration

- ◆ Export différent de checkout (après un export, la copie effectuée n'est plus gérée par CVS)
- ◆ *export*



Manipulation demandée

- Créer un répertoire de référence
- Y ajouter un fichier.c de votre choix
- Créer deux copies locales C1 et C2
- Les modifier
- Mettre à jour le répertoire de référence à partir de C1 (utiliser tag)
- Observer l'état de C2
- Mettre à jour le répertoire de référence à partir de C2.
- Retrouver la version C1 à partir du répertoire de référence

- Autoriser/interdire l'accès à un autre utilisateur (votre binôme)
 - ◆ En lecture, écriture ou bien en lecture-écriture