Une architecture pour intégrer des composants de contrôle de la coopération dans un atelier distribué

Manuel MUNIER

LORIA - équipe ECOO

Plan

- Introduction
- Problématique
- Modèle de Transactions
- Mise en œuvre
- Conclusion et Perspectives

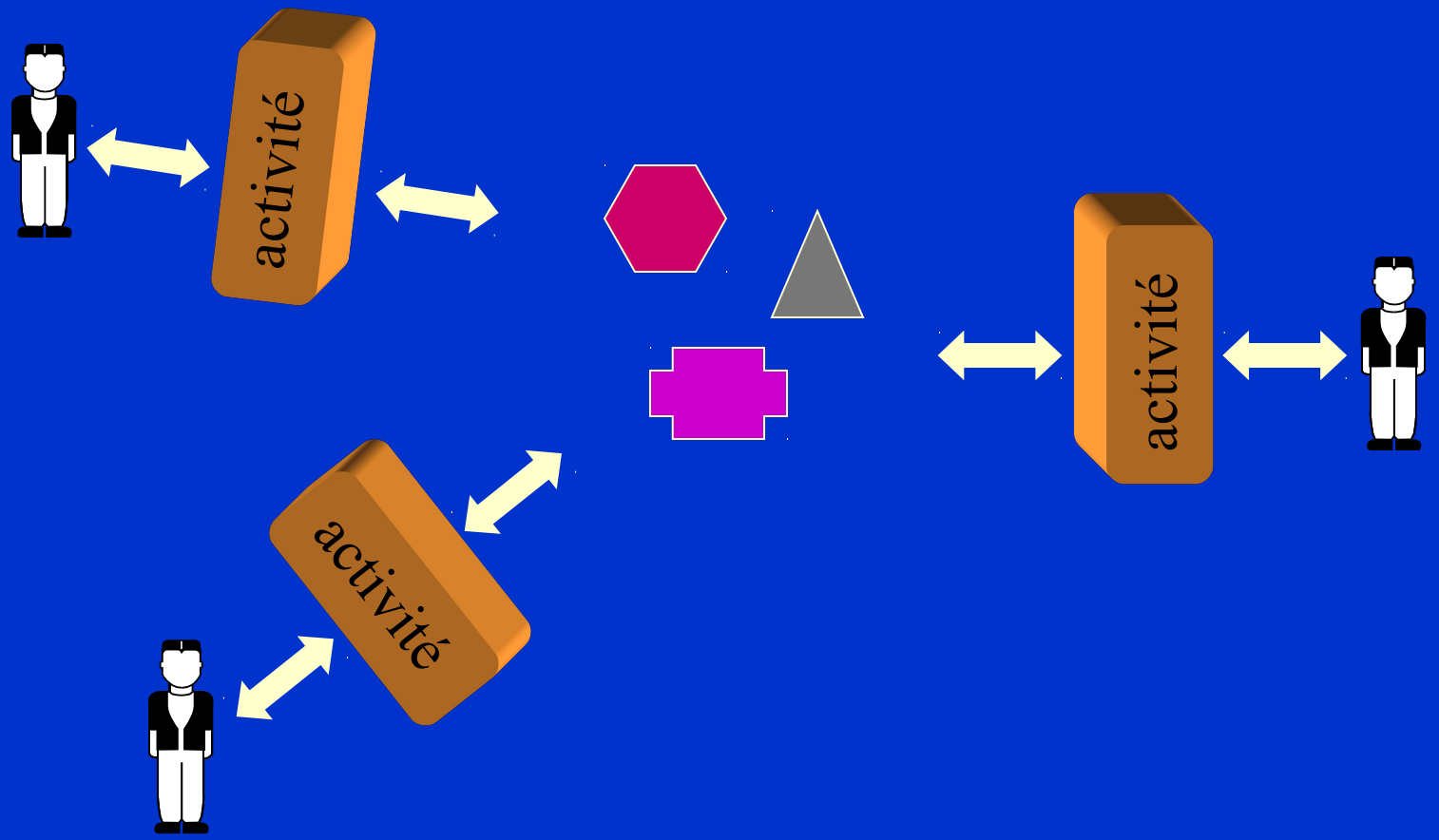
Introduction

- Réseau Internet : démocratisation des moyens de communication
- Idée : permettre la coopération entre des utilisateurs éloignés
- Question : comment coordonner ces différents utilisateurs ?

Coopération

- Coopération indirecte, i.e. au travers des objets de l'application
- Collaboration pour un objectif commun
  - *développement d'un logiciel*
  - *conception d'un bâtiment*
- Coordination des interactions
  - *synchronisation des accès concurrents*
  - *organisation des activités*

Coopération



Coopération

- Caractéristiques des activités :
  - interactives

    échanges d'informations en cours d'exécution
  - incertaines

    l'enchaînement des opérations n'est pas connu à priori
  - de longue durée

    elles peuvent durer plusieurs mois
  - distribuées

    elles s'exécutent sur des sites différents

Contraintes

- Besoins des utilisateurs :
  - autonomie
    travail en mode déconnecté
    gestion des données par les partenaires eux-mêmes
  - contrôle personnalisé
    utilisation de règles de coopération différentes
  - intégration
    production via les applications existantes
    faible impact sur les habitudes des utilisateurs

Objectifs

- Environnement support de la coopération
- Activités distribuées et autonomes
- Coordination des activités

Plan

- Introduction
- Problématique
- Modèle de Transactions
- Mise en œuvre
- Conclusion et Perspectives

Problématique

- Coordination d'activités coopératives

  $\Rightarrow$ problèmes de cohérence des objets partagés

- Correction des exécutions coopératives :
  - correction des interactions

    entrelacement des opérations des activités
  - correction individuelle

    contraintes d'intégrité sur les résultats

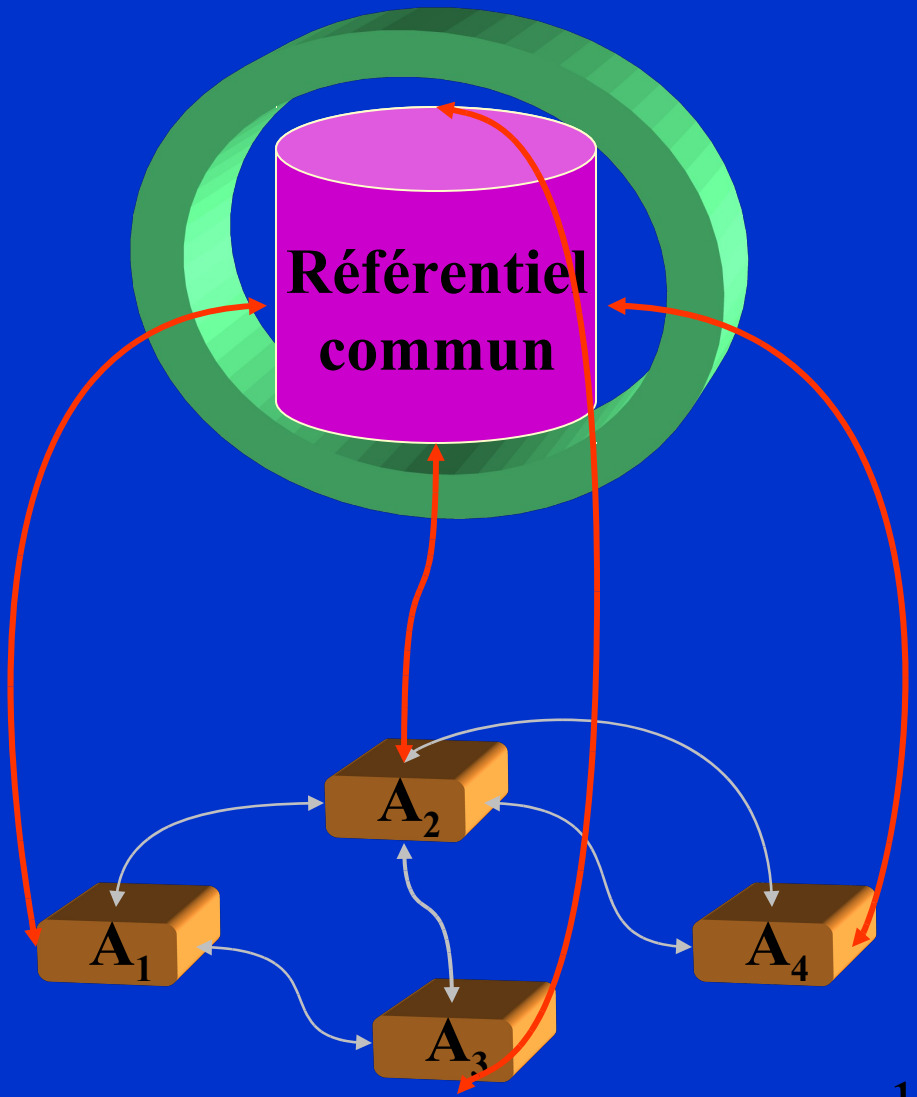
Solutions actuelles

- Gestionnaires de configurations
  - *graphe de versions des objets partagés*
- Gestion des procédés
  - *description des procédés exécutés*
- Systèmes transactionnels
  - *contrôle de la concurrence*
- Collecticiels
  - *communication, relations humaines*

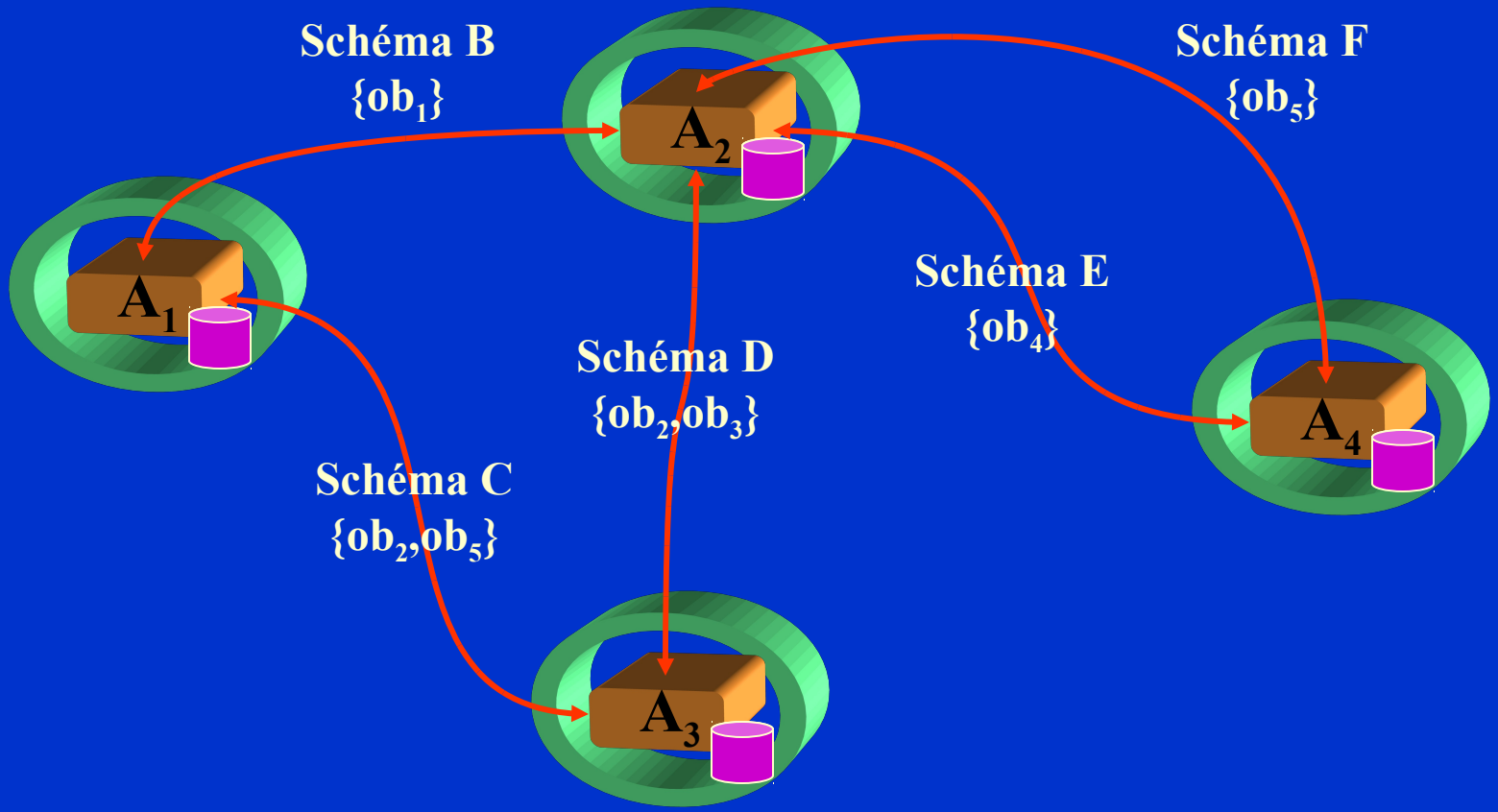
Approche COO

- Approche transactionnelle : cacher les problèmes de concurrence d'accès
- Activités encapsulées dans des COO-transactions (non isolées)
- Critère de correction des exécutions coopératives : la COO-sérialisabilité
  - *lectures de résultats intermédiaires*
  - *convergence d'un groupe vers un état final*

Organisation centralisée



Organisation distribuée



Plan

- Introduction
- Problématique
- Modèle de Transactions
- Mise en œuvre
- Conclusion et Perspectives

Modèle de Transactions

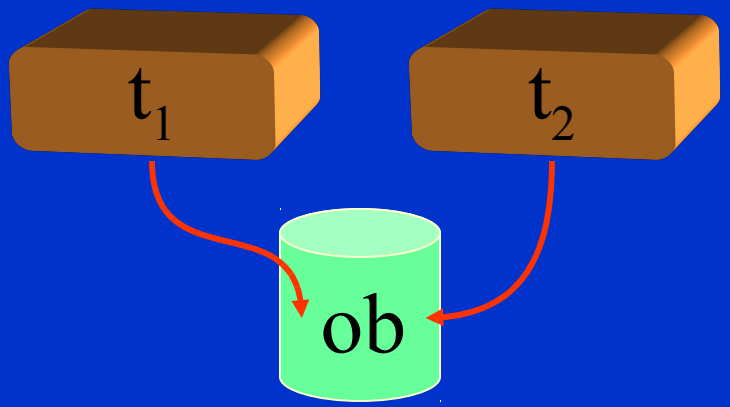
- Approche transactionnelle …
- … mais avec 3 différences notables :
  - *un référentiel local par transaction*
  - *échanges de données explicites*
  - *contrôle décentralisé*

⇒ Nouveau modèle de transactions avancé

Référentiels locaux

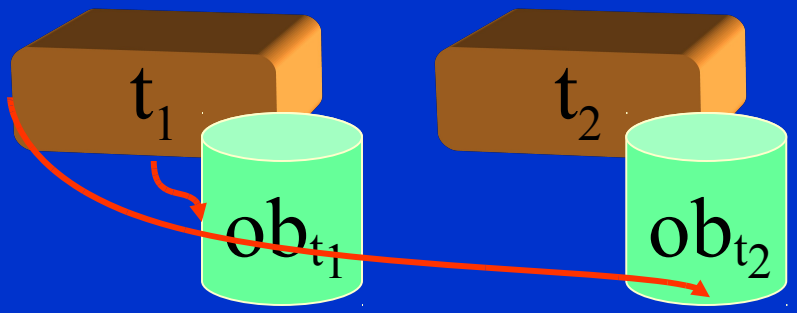
Référentiels locaux
⇒ plusieurs copies d'un même objet

référentiel unique



$op_{t_i}[ob]$

référentiels locaux



$op_{t_i}[ob_{t_j}]$

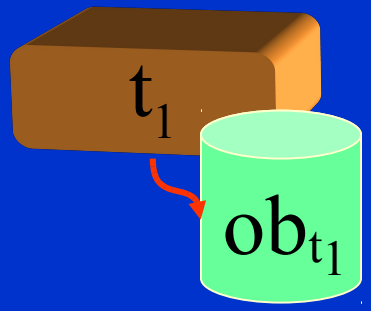
Histoires locales

- Informations pertinentes pour une transaction t_1 donnée :
  - *opérations qu'elle a invoquées*
  - *opérations invoquées par d'autres transactions sur son référentiel*

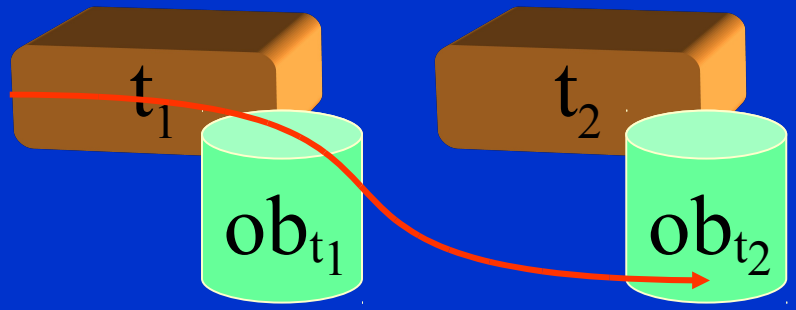
$$View_{t_1} = \{op_{t_1}[ob_{t_i}]\} \cup \{op_{t_j}[ob_{t_1}]\}$$

Histoires locales

- Opération $op_{t_1}[ob_{t_1}]$:
  - *journalisée par t_1 uniquement*

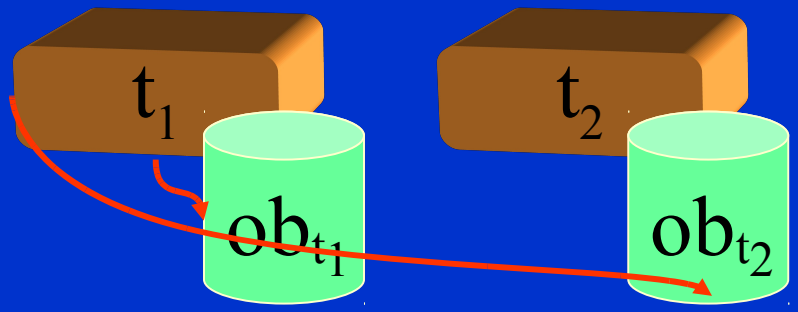


- Opération $op_{t_1}[ob_{t_2}]$:
  - *journalisée par t_1*
    (opération de t_1)
  - *journalisée par t_2*
    (objet de t_2)



Opérations de transfert

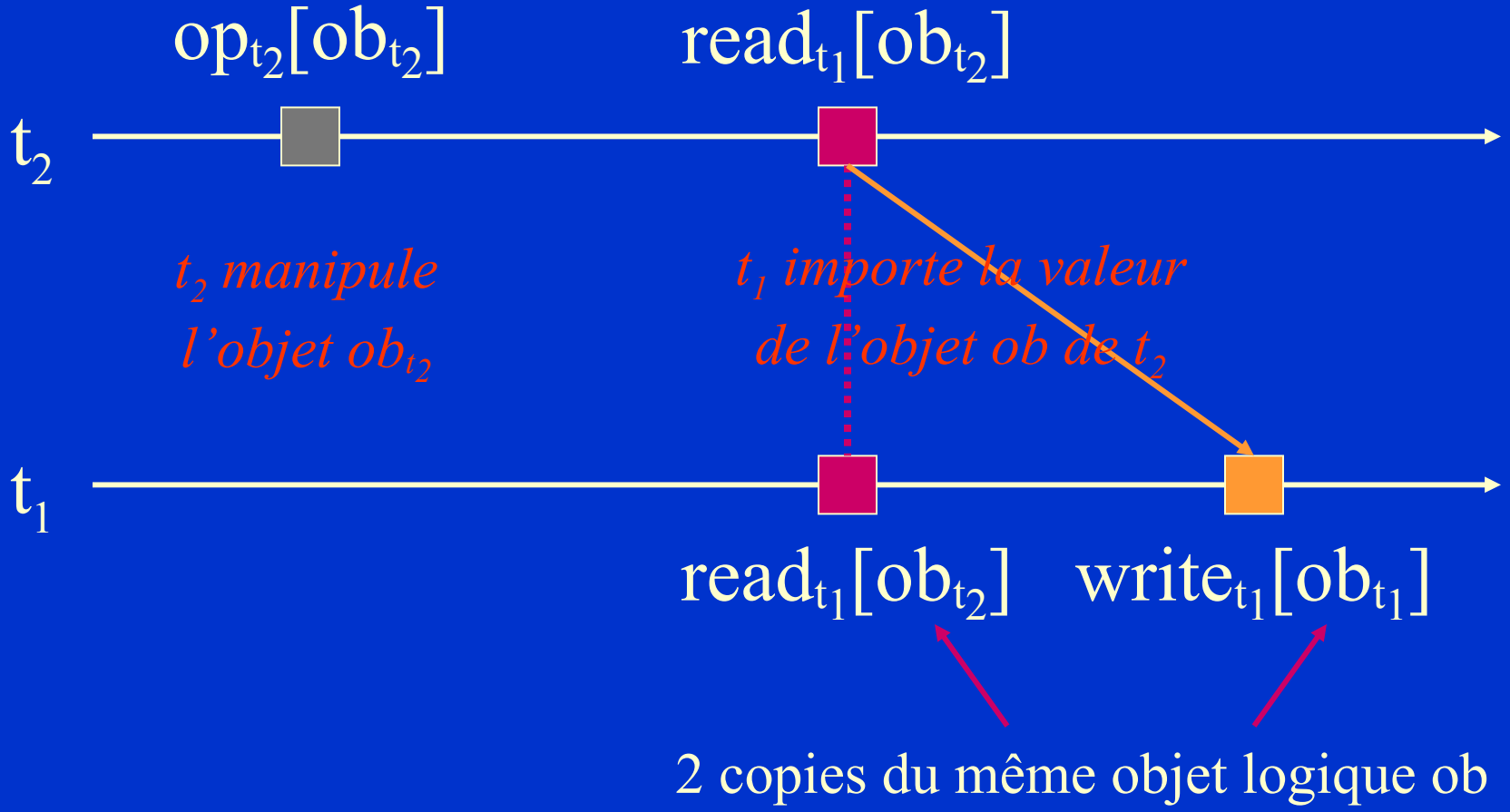
- Synchronisation des copies via des opérations de transfert :



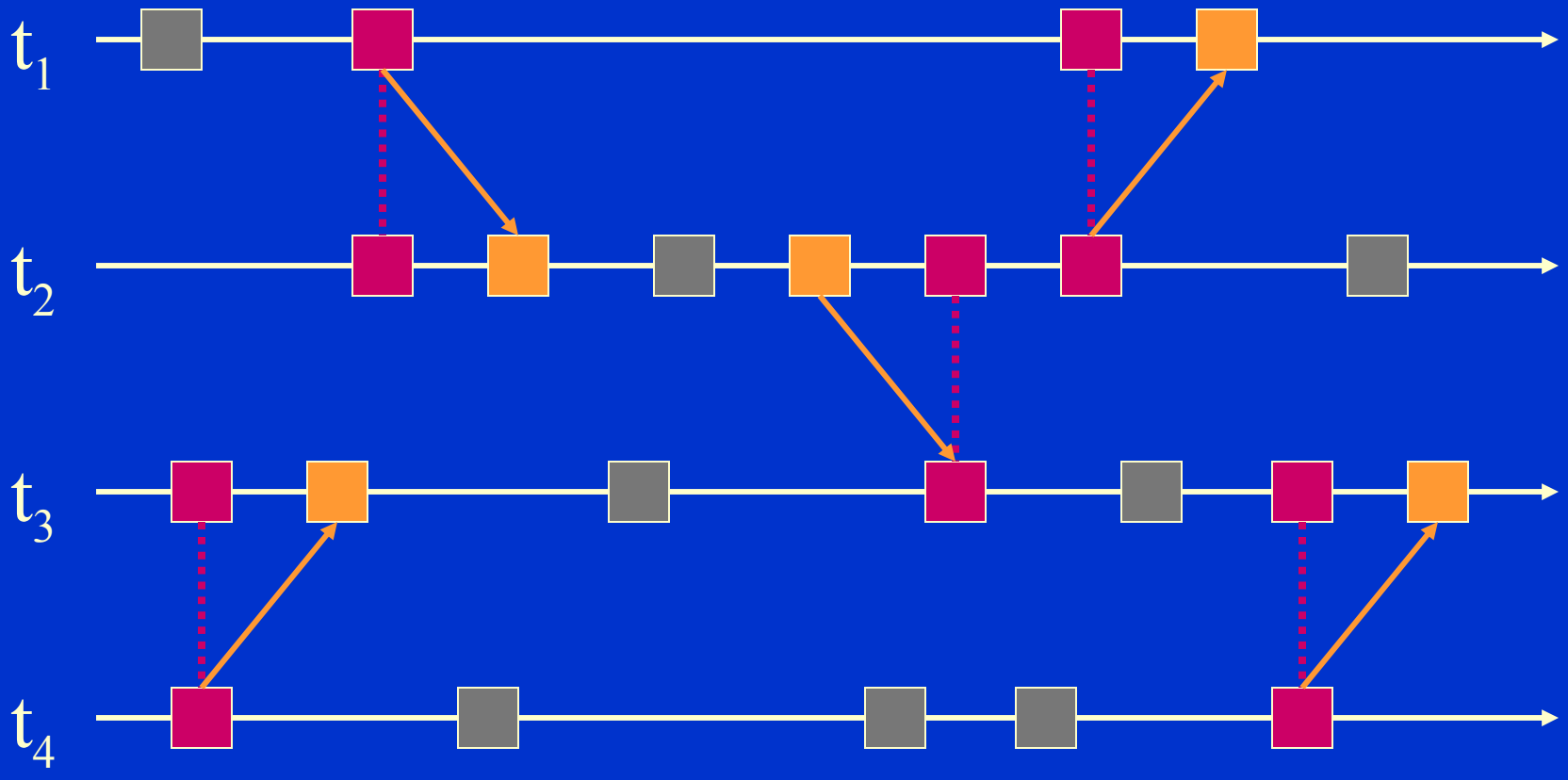
– *importation* : $read_{t_1}[ob_{t_2}] + write_{t_1}[ob_{t_1}]$

– *exportation* : $read_{t_1}[ob_{t_1}] + write_{t_1}[ob_{t_2}]$

Opérations de transfert



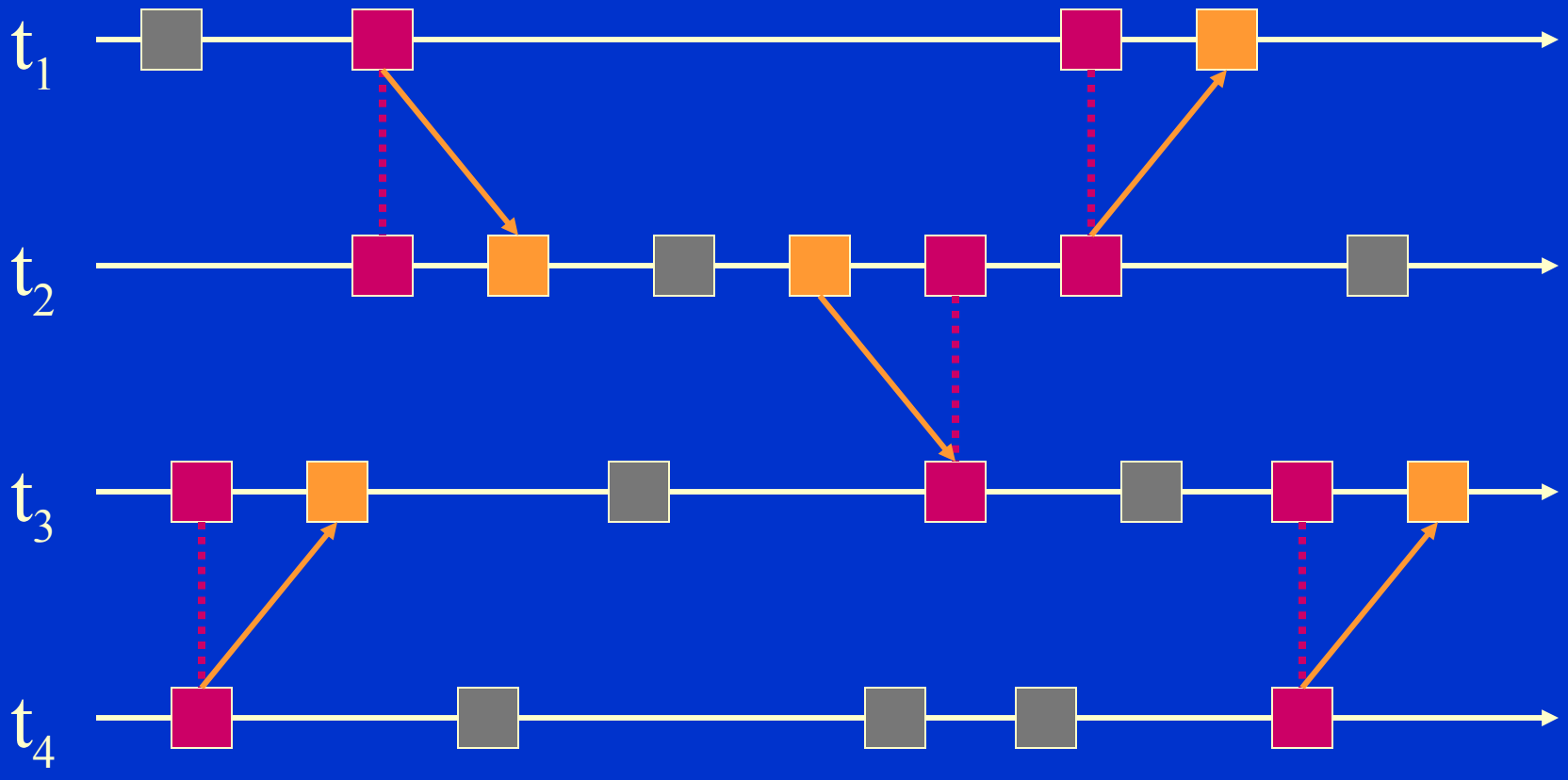
Ordre causal entre les opérations



Critères de correction distribués

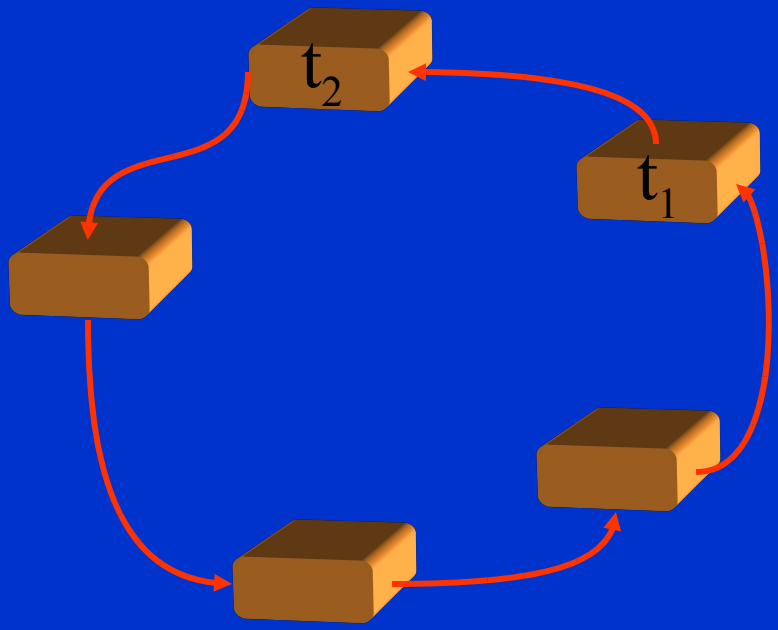
- Vérification de propriétés sur les histoires locales des transactions
- Mêmes propriétés globales qu'avec un critère de correction classique vérifié sur l'histoire globale

Critères de correction distribués



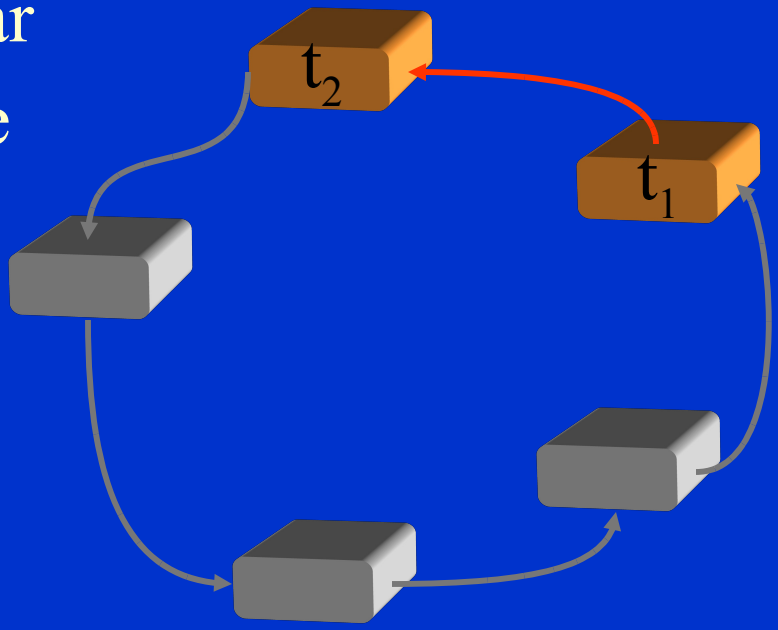
Critère de correction « centralisé »

- Critère centralisé :
  - notion de cycle de dépendances entre transactions
  - consensus sur les valeurs finales des objets partagés



Critère de correction « distribué »

- Critère distribué :
  - t_2 doit être « à jour » par rapport à t_1 avant d'être validée
  - t_2 ne peut être validée que si t_1 est elle-même validée



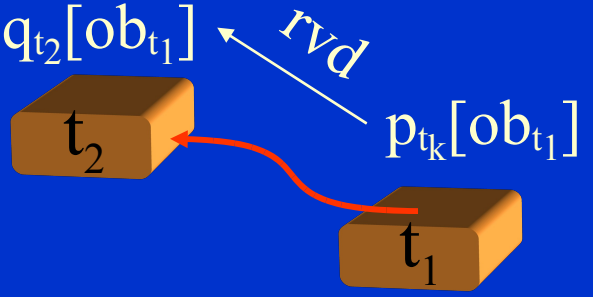
Modèle de Transactions

DisCOO-sérialisabilité

- Propriété locale « à jour » :

  t_2 doit être « à jour » par rapport à t_1 avant d'être validée

$$(\text{Commit}_{t_2} \in H_{t_1}) \Rightarrow \forall ob \;\; \text{up\_to\_date}(t_2,t_1,\{ob\})$$



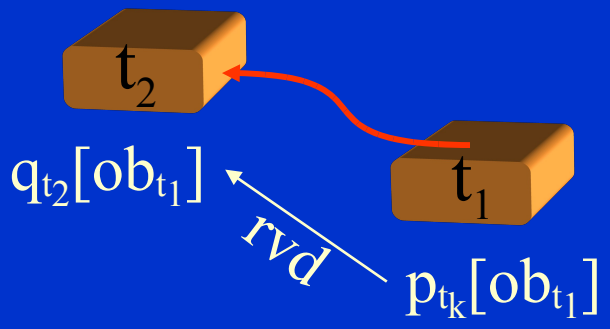
$$\text{up\_to\_date}(t_2,t_1,O) \equiv \forall ob \in O \;\; \forall Q_{t_2}\,[ob_{t_1}]$$
$$(\exists t_k \; \exists p \quad \text{rvd}(p_{t_k}[ob_{t_1}],\text{LastOcc}(Q_{t_2}\,[ob_{t_1}])$$
$$\wedge \;\; (p_{t_k}[ob_{t_1}] \rightarrow \text{LastOcc}(Q_{t_2}\,[ob_{t_1}]))$$
$$\Rightarrow$$
$$\neg\, (\exists p' \in \text{Sequence}(p_{t_k}[ob_{t_1}]) \;\; (\text{LastOcc}(Q_{t_2}\,[ob_{t_1}]) \rightarrow p'_{t_k}[ob_{t_1}]))$$

DisCOO-sérialisabilité

- Propriété locale « validation après » :

  t_2 ne peut être validée que si t_1 est elle-même validée

$$(\text{Commit}_{t_2} \in H_{t_1}) \Rightarrow$$
$$(\exists p,q \;\; \text{rvd}(p_{t_k}[ob_{t_1}], q_{t_2}[ob_{t_1}]) \wedge (p_{t_k}[ob_{t_1}] \rightarrow q_{t_2}[ob_{t_1}])) \Rightarrow$$
$$(\text{Commit}_{t_k} \in H_{t_1})$$

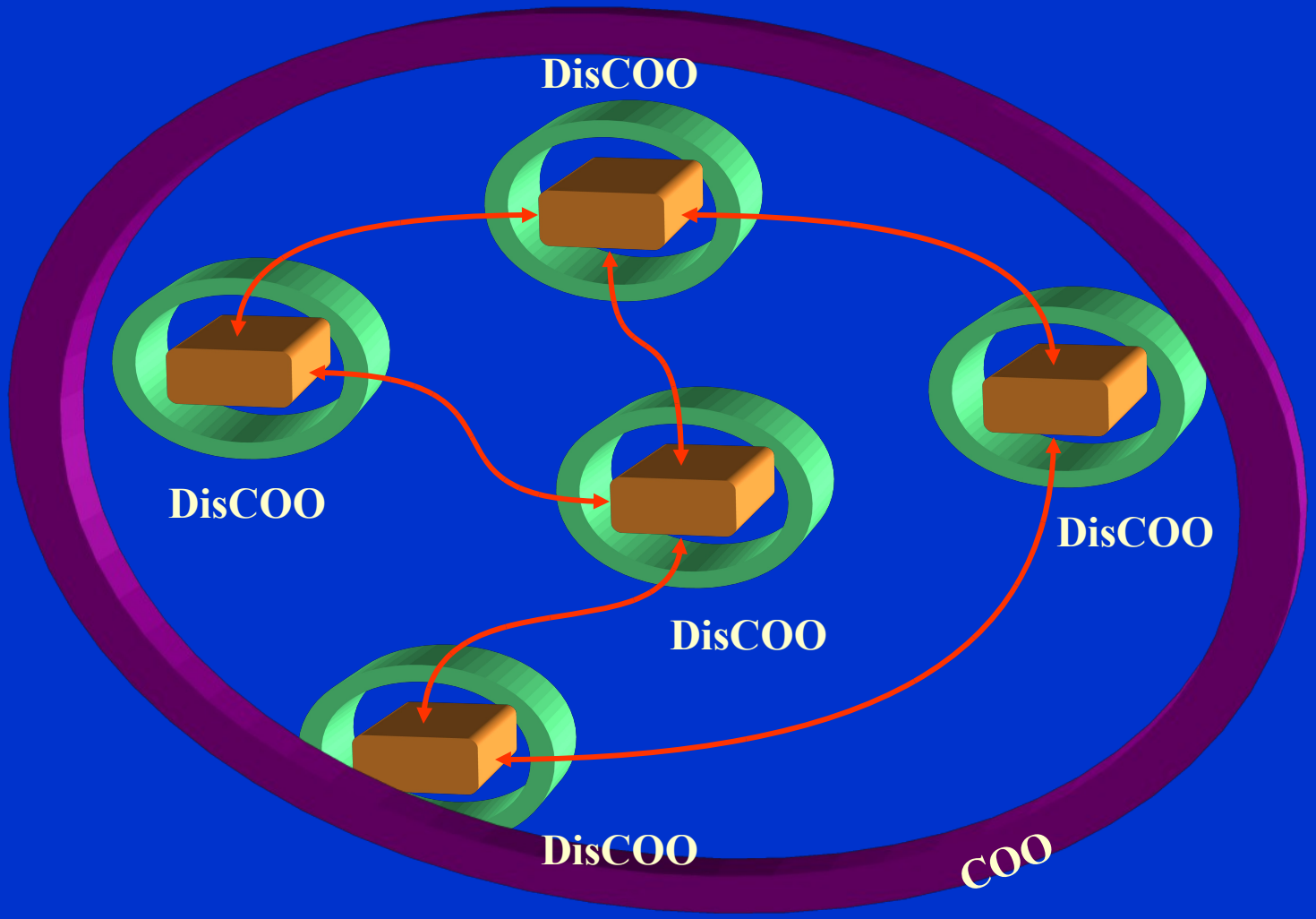


DisCOO-sérialisabilité

- 2 propriétés locales vérifiées sur l'histoire locale de chaque transaction :
  - « validation après »
  - « à jour »
- Propriétés globales garanties en cas de cycle:
  - « validation après » ⇒ *toutes les transactions du cycle doivent être validées en même temps*
  - « à jour » ⇒ *transactions à jour les unes par rapport aux autres*

Modèle de Transactions

Résultats



Schémas de coopération

- 4 schémas de coopération supportés par la DisCOO-sérialisabilité :
  - écriture coopérative
  - rédacteur/relecteur
  - client/serveur
  - exécution en isolation
- Contract[t_1,t_2,O,s] ≡ « *les transactions t_1 et t_2 respectent le schéma s pour partager les objets de l'ensemble O* »

Schémas de coopération

- 4 règles sur les événements Contract[…] :
  - contrat signé en même temps par les 2 transactions
  - opération $op_{t_1}[ob_{t_2}]$ acceptée si :
    - *il existe un contrat entre t_1 et t_2 portant sur ob*
    - *le schéma du contrat est respecté*
  - un seul contrat par objet entre 2 transactions
  - la validation (ou annulation) d'une transaction doit respecter les contrats signés

Conclusion

- Nouveau modèle de transactions avancé
  - *référentiel local, histoire locale*
  - *échanges de données explicites*
- Critères de correction distribués
  - *propriétés vérifiées sur les histoires locales*
  - *garantie implicite de propriétés globales*
- Intégration possible de nouveaux schémas de coopération

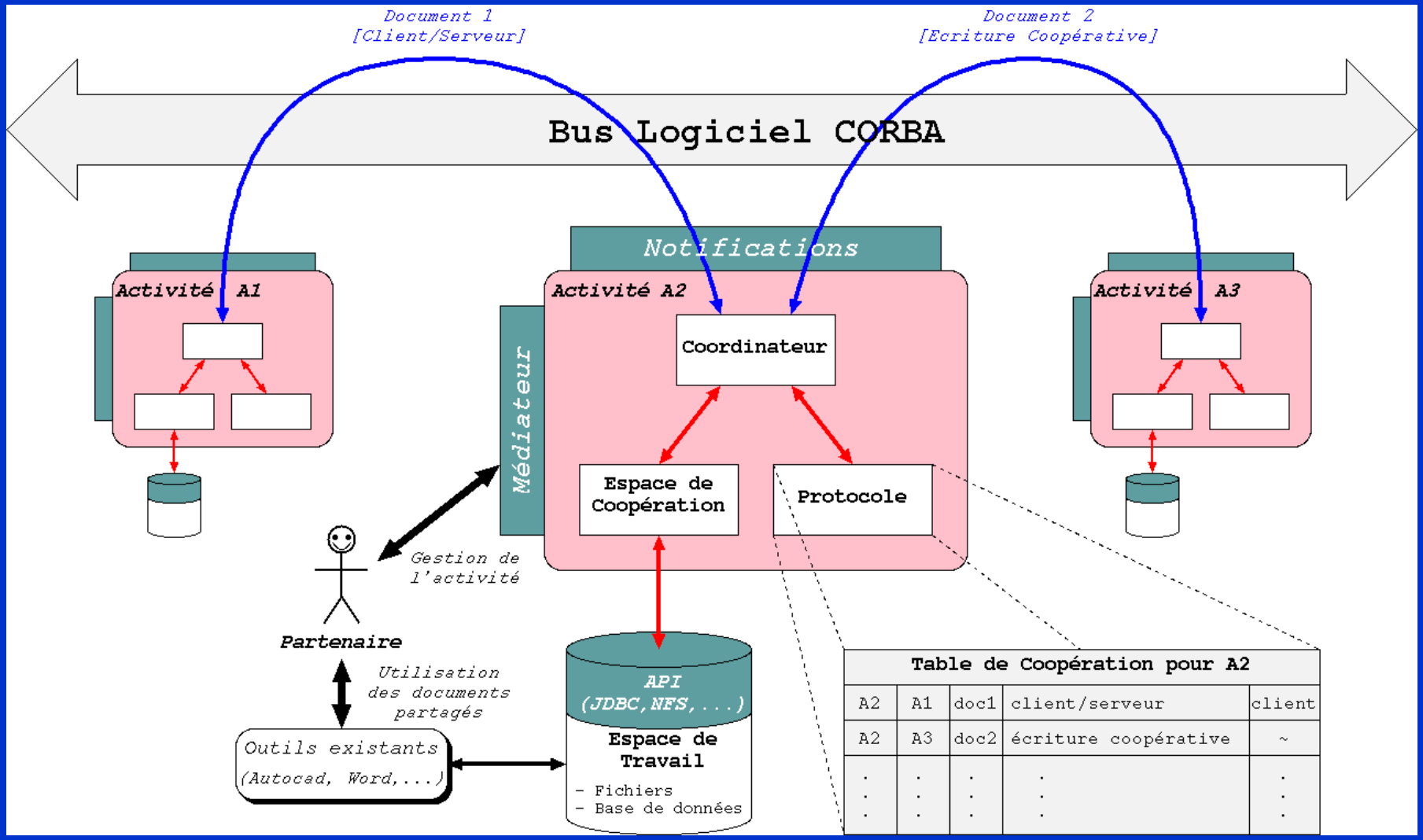
Plan

- Introduction
- Problématique
- Modèle de Transactions
- Mise en œuvre
- Conclusion et Perspectives

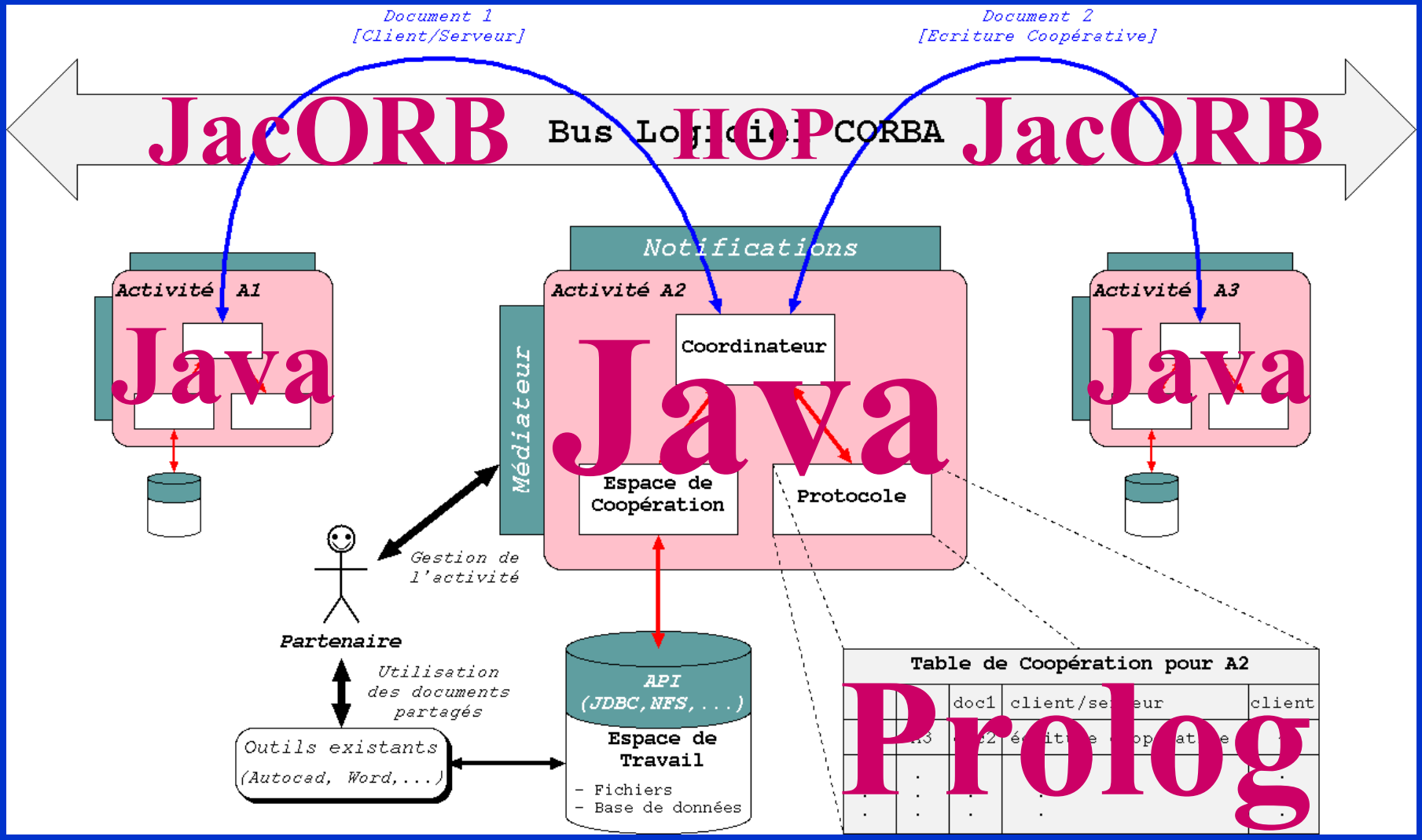
Mise en œuvre

- Environnement support de la coopération
  - *activités distribuées, d'égale à égale*
  - *autonomie, contrôles décentralisés*
  - *utilisation des applications existantes*

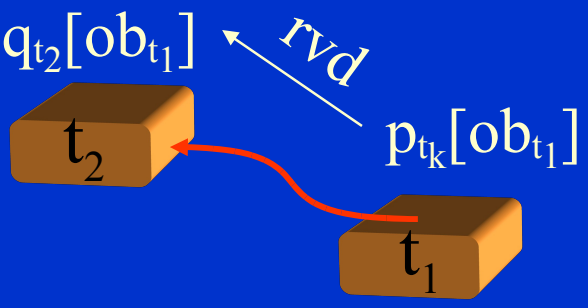
Architecture



Prototype


Document 1
[Client/Serveur]
Document 2
[Ecriture Coopérative]
JacORB
Bus Logiciel CORBA
HOP
JacORB
Activité A1
Java
Notifications
Activité A2
Médiateur
Java
Coordinateur
Espace de Coopération
Protocole
Activité A3
Java
Gestion de l'activité
Partenaire
Utilisation des documents partagés
Outils existants
(Autocad, Word,...)
API
(JDBC,NFS,...)
Espace de Travail
- Fichiers
- Base de données
Table de Coopération pour A2
doc1
client/serveur
client
Prolog

Mapping axiomes ACTA → clauses Prolog



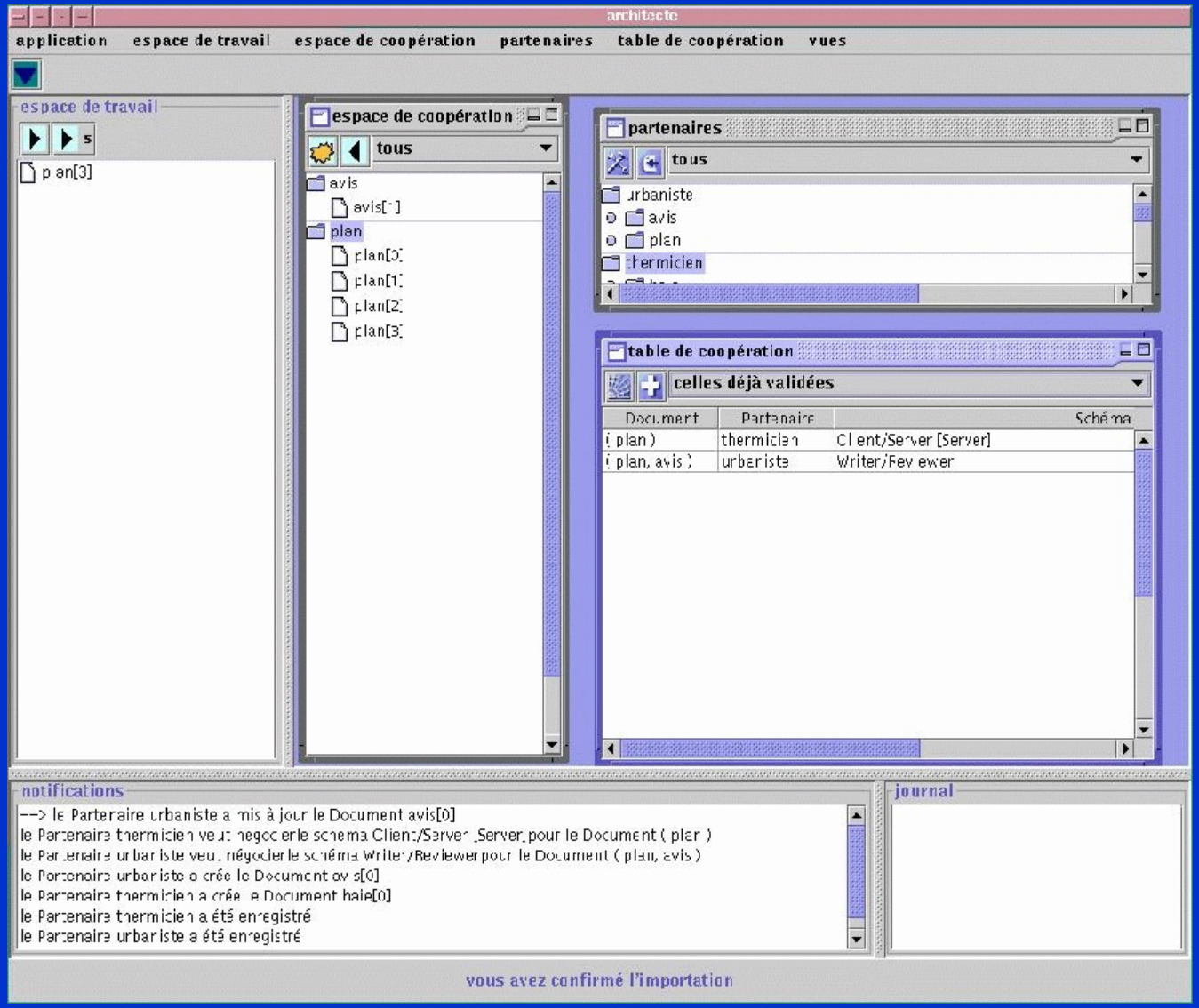
$$up\_to\_date(t_2,t_1,O) \equiv \forall ob \in O \;\; \forall Q_{t_2}[ob_{t_1}]$$
$$(\exists t_k \; \exists p \quad rvd(p_{t_k}[ob_{t_1}], LastOcc(Q_{t_2}[ob_{t_1}])$$
$$\wedge \;\; (p_{t_k}[ob_{t_1}] \rightarrow LastOcc(Q_{t_2}[ob_{t_1}]))$$
$$\Rightarrow$$
$$\neg (\exists p' \in Sequence(p_{t_k}[ob_{t_1}]) \;\; (LastOcc(Q_{t_2}[ob_{t_1}]) \rightarrow p'_{t_k}[ob_{t_1}]))$$

Mapping axiomes ACTA → clauses Prolog

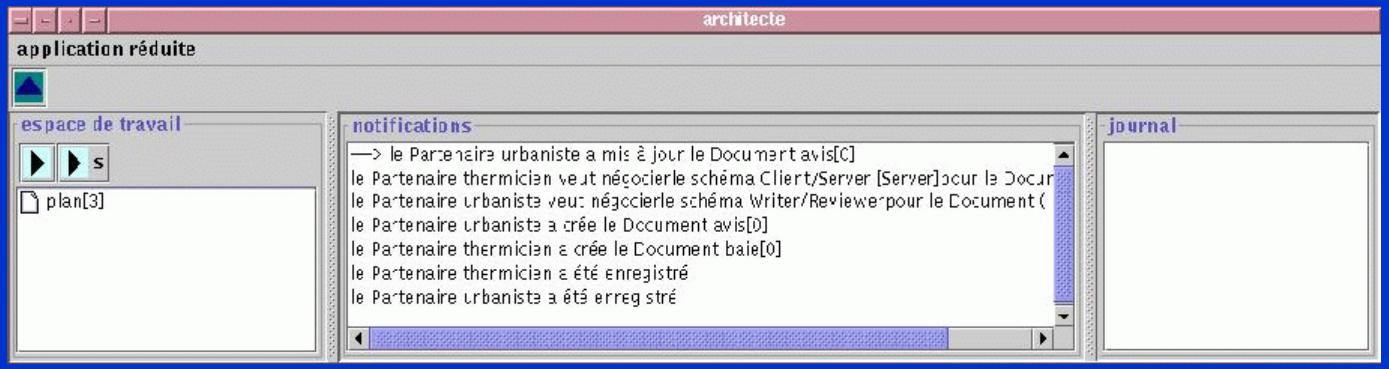
```
lostUpdate(A1, A2, X) :-
    me(Myself),
    trace(Myself, Tr),
    checkOverwrite(Tr, [], A1, A2, X),
    not(eq(A1, A2)).

checkOverwrite([write(Aj,Ob,Myself)|Queue], List, Ai, Aj, Ob) :-
    me(Myself),
    not(eq(Ai,Aj)),
    not(head(Queue,read(Aj,Ob,Ai))),
    not(find(read(Ai,Ob,Myself), List)),
    not(find(write(Myself,Ob,Ai), List)).

checkOverwrite([Op|Queue], List, Ai, Aj, Ob) :-
    checkOverwrite(Queue, [Op|List], Ai, Aj, Ob).
```

Mise en œuvre

Interface utilisateur pour coopérer



Interface utilisateur pour produire



Plan

- Introduction
- Problématique
- Modèle de Transactions
- Mise en œuvre
- Conclusion et Perspectives

Résultats

- Formalisation d'un nouveau modèle de transactions avancé
  - transactions distribuées
  - critères de correction distribués
    - *DisCOO-sérialisabilité*
    - *D-sérialisabilité*
  - schémas de coopération
- Prototype DisCOO
  - mise en œuvre et validation du modèle proposé

Perspectives

- Coopération directe :
  - notifications
  - négociation des schémas de coopération
  - renégociation dynamique
  - schémas multi-partites

projet de post-doc avec Xerox

- Nouveaux schémas de coopération
  - intégration aisée dans DisCOO

Merci de votre attention

Manuel Munier
LORIA - équipe ECOO
15 janvier 1999