

# Travail Coopératif et Nouvelles Technologies

Master 2 Informatique des Organisations  
Systèmes d'Information et Technologies Nouvelles

Joyce EL HADDAD  
*elhaddad@lamsade.dauphine.fr*

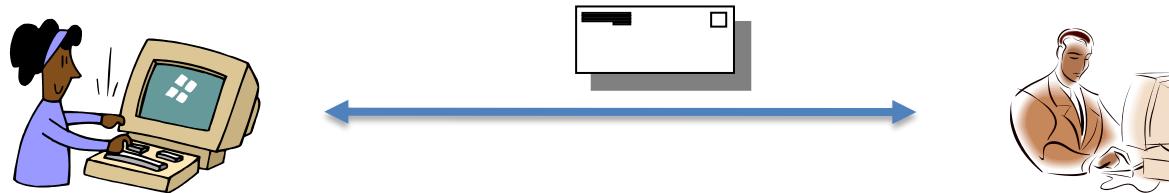
# Informations Administratives

- 24h de cours (8 séances de 3h)
- Contrôle de connaissance :  
Note finale = 0.5Examen + 0.5Projet
- Planning:  
S1, S2, S3 (Travail Collaboratif)  
S4, S5, S6 (Systèmes Collaboratifs Distribués)  
S7, S8 (TPs)  
S8 (Soutenances)

# Plan

- Introduction
  - Travail coopératif, applications coopératives, systèmes coopératifs
- Travail Collaboratif
  - Groupware, processus métier, workflow, web service composition
- Systèmes Collaboratifs Distribués
  - Problèmes, duplication, consistance

# Introduction



- Travail coopératif ou collaboratif
  - Communication par voie postale,...
- Travail Coopératif Assisté par Ordinateur
- L'étude des **technologies de la coopération** et des **systèmes coopératifs** passent par la compréhension du **travail coopératif**

# Introduction

- Historique :
  - Durant l'ère préindustrielle : activités sporadiquement coopératives. La logique fonctionnelle comme modèle organisationnel des entreprises : regrouper dans une même unité organisationnelle les personnes qui font la même chose, qui remplissent la même fonction, qui exercent les mêmes activités (*ex. une entreprise qui développe, fabrique et commercialise trois catégories de produits (A, B et C) s'organise selon la logique fonctionnelle si elle crée une direction du développement (réunie les personnes en charge du développement des produits A, B et C), une direction de fabrication et une direction commerciale*)

# Introduction

- Historique :
  - En 1990 : évolution des modèles organisationnels des entreprises à une organisation en division. Les personnes sont regroupées dans une même unité parce qu'elles contribuent toutes à produire un même résultat (*ex. une entreprise qui développe, fabrique et commercialise trois catégories de produits (A, B et C) s'organise selon la logique divisionnelle si elle crée une division A (dans laquelle sont réunies les personnes en charge du développement, de la fabrication et de la commercialisation des produits A), une division B et une division C.*)
  - Les tâches sont unifiées au sein de **processus métier cohérents** qui peuvent être conçus et analysés par des **workflows coopératifs**

# Introduction

- Pourquoi s'engager dans ce processus coûteux en termes de personnel, de ressources et de temps qui consiste à faire exécuter des tâches par de multiples personnes?
- Douglas Engelbart (Turing Award 1997)
  - « *La complexité et l'urgence des problèmes auxquels nous devons faire face croissent plus vite que notre capacité à les comprendre et à les résoudre. C'est un problème essentiel, nous pouvons prendre des mesures stratégiques, collectivement* »
  - L'approche d'Engelbart : Augmenter les capacités intellectuelles humaines en augmentant notre intelligence collective.
  - Les personnes seraient incapables individuellement de réaliser l'ensemble des tâches

# Introduction

- Travail coopératif dans une organisation
  - « *le travail coopératif est constitué de processus de travail liés par la nature de leur contenu, c'est-à-dire qui appartiennent à la production d'un produit particulier ou d'un type de produit ou de services* »
- Le travail coopératif n'implique pas forcément une communication directe entre les agents
  - Les agents peuvent coopérer via un espace informationnel plus ou moins commun sans communication directe et sans nécessairement connaître l'autre ou même sans connaissance de l'autre (*le site SourceForge de développement de logiciels libres*)

# Introduction

- Différentes dynamiques du travail coopératif :
  - **Coopération au sens étroit** : groupe bien établi, les rôles et les règles bien définis, chaque nouvelle conversation est liée aux précédentes
  - **Coopération au sens large** : groupe large, mal délimité et changeant dans le temps, les rôles sont mal ou pas définis, les interactions entre les individus sont liées par un sujet d'intérêt, les conversations ne sont pas nécessairement liées
- Différents types de coopération impliquant différent types de travail coopératif
  - La **coordination**
  - La **collaboration** (la codécision)

# Introduction

## □ La coordination

*« un processus dans lequel les individus ont besoin de coordonner leurs actions à celles des autres. Les actions d'un individu donnent un sens aux actions des autres, les actions des autres contribuent à l'action individuelle »*

- Les personnes coordonnent leurs activités pour réaliser le produit final
- Chaque participant agit d'une manière plus ou moins planifiée envers le processus (*selon sa compétence, son rôle, ...*)
- Problèmes :
  - La synchronisation des personnes et des actions
  - La cohérence des actions individuelles par rapport à l'ensemble du processus

# Introduction

## □ La collaboration

*« un processus dans lequel les individus ont besoin de travailler à l'exécution d'une certaine action pour produire un résultat final. A la fin du processus, il est impossible d'isoler la contribution unique des individus parce que le produit final est une entité incarnant le résultat unifié de toutes les contributions individuelles »*

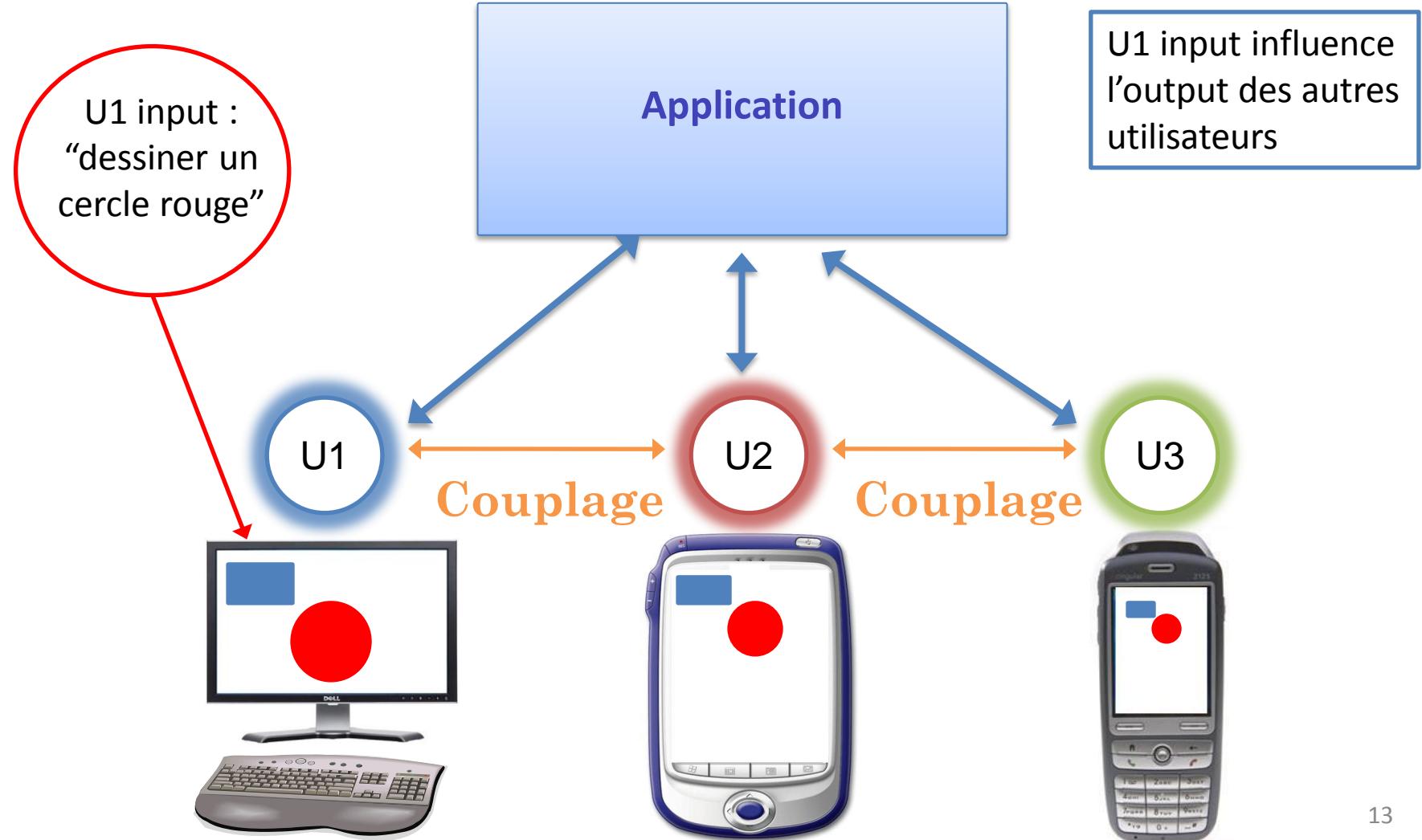
- Collaborer = travailler ensemble à l'exécution d'une tâche
- Problèmes :
  - Le succès dépend de la contribution des individus pour produire un savoir partagé
  - Le succès dépend de la compréhension commune de l'objectif et du processus

# Introduction

- Applications Collaboratives
  - Définition 1 : un éditeur permettant à plusieurs utilisateurs de rédiger un document ensemble
  - Définition 2 : une application permettant à plusieurs utilisateurs de réaliser un objectif commun
  - *Définition 3 : une application logicielle qui interagit avec plusieurs utilisateurs et qui les lie (output d'un utilisateur peut être influencer par un input d'un autre utilisateur)*

# Introduction

## ❑ Applications Collaboratives



# Introduction

- La coordination est améliorée si l'application permet de :
  - Distribuer et sélectionner facilement les messages (*filtre de communication intelligent*)
  - Lier des messages au sein d'une conversation (*l'unité de communication n'est pas un message mais une conversation comme dans gmail*)
  - Enregistrer et classer des messages-conversations avec leur statut en cours (*en phase de négociation, démarré par une requête, conversations closes,...*)
  - Modéliser le coordinateur
  - Classer et sélectionner les processus récurrents (*créer des catégories d'actions comme requêtes, approbations,...*)

# Introduction

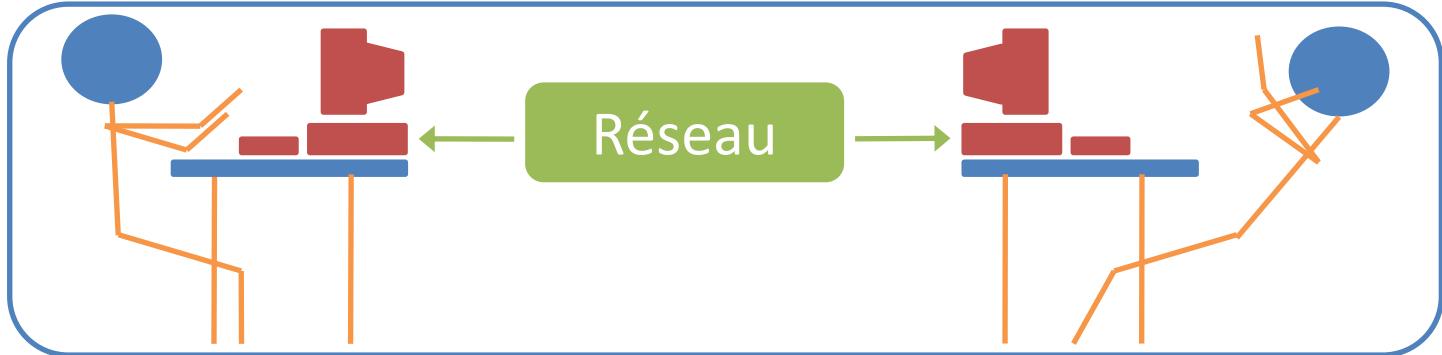
- Les individus qui collaborent à une tâche peuvent vouloir accéder à toutes les infos qui caractérisent cette tâche
- Développer des applications de support à la collaboration signifie introduire des outils qui permettent de :
  - Structurer l'information pour qu'elle reflète la manière dont elle a été créée
  - Accéder aux informations avec différents droits propres aux rôles que jouent les membres dans le processus
  - Assister les flux de questions-réponses sur la tâche en cours
  - Partager un espace commun avec la vision pour chacun de ce que font les autres

# Introduction

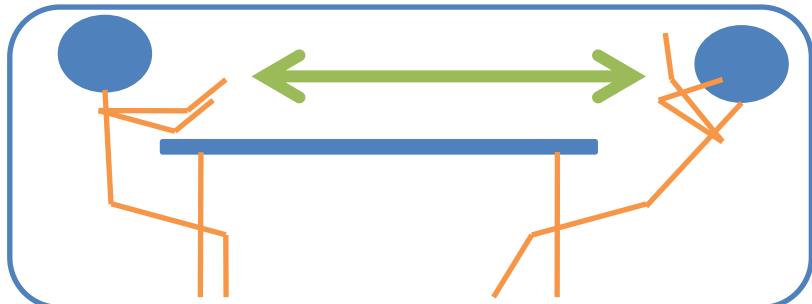
- Supports pour la codécision
  - Partager toute information utile pour prendre la décision
    - Toute info collectée et créée concernant le problème à traiter doit être accessible à tous les participants pendant toute la durée du processus de codécision
  - Partager les critères de décision
    - La convergence des opinions des participants vis-à-vis d'un groupe de décision s'améliore si les participants comprennent les critères de création de ces opinions
  - Partager les décisions déjà prises au cours du processus
    - Il faut que chaque membre sache ce qu'il a à faire avant la prochaine réunion
  - Gérer les conversations pour clarification et orientation
    - Les codécisions sont prises suite à un ensemble de conversations

# Introduction

- Applications Collaboratives
  - Les applications collaboratives permettent le Travail Coopératif Assisté par Ordinateur (TCAO)

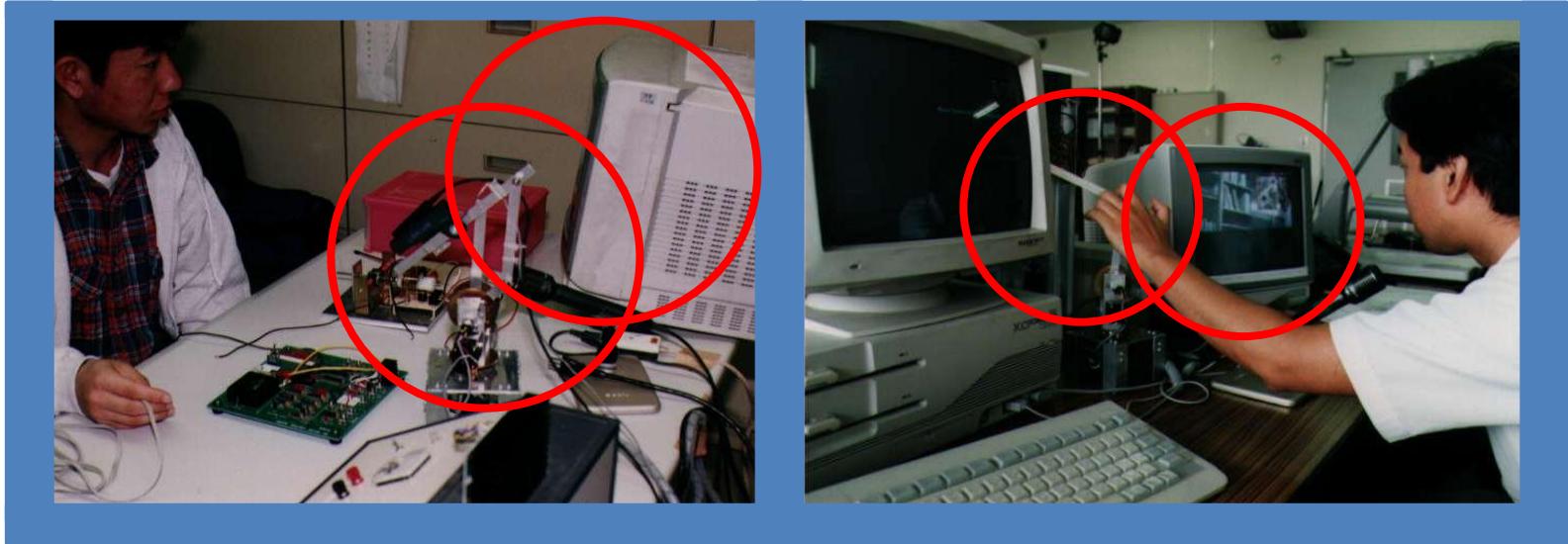


- Pourquoi avoir besoin de l'ordinateur comme support de collaboration alors qu'on le fait sans depuis des années ?



# Introduction

- Applications Collaboratives
  - Illusion d'être sur place



- Contrôle de la caméra dans une autre location
- Les deux utilisateurs voient sur leurs écrans la vue de la caméra
- “zoom in” pour pointer sur un élément
- Autre : LG Gesture Cam (Salon High-Tech CES 2012)

# Introduction

- Applications Collaboratives
  - Si uniquement illusion d'être sur place

Application Collaborative < Face à Face

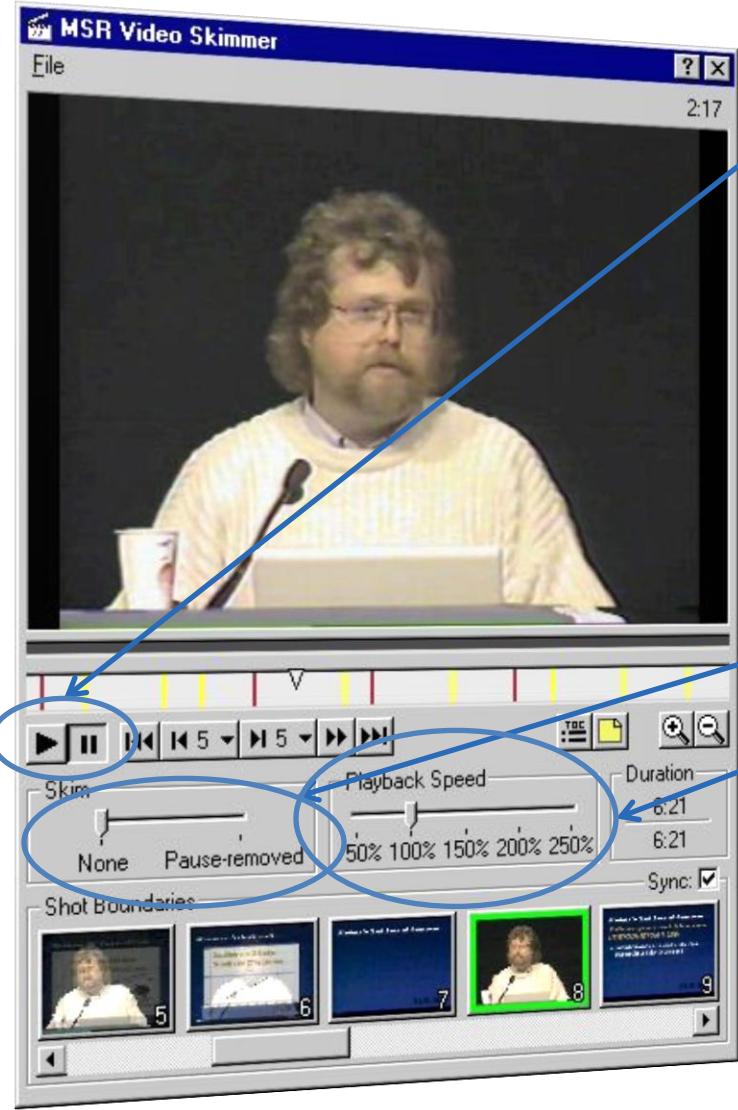
- Allez au-delà de l'illusion d'être sur place
  - Mécanisme similaire à la téléportation

Application Collaborative > Face à Face

- Exemple : VisoConférence

# Introduction

## □ Applications Collaboratives



Commandes de contrôle basiques:  
Play, Pause, etc.

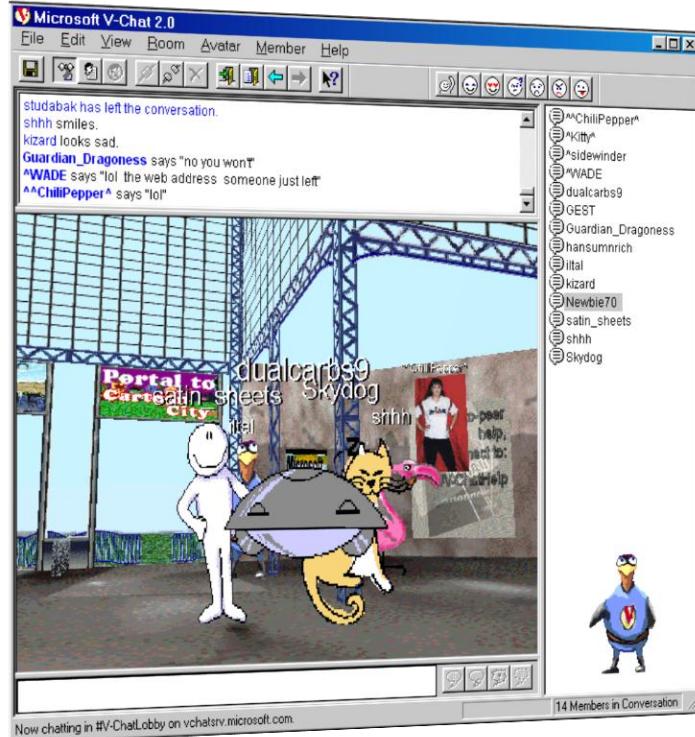
“Beyond Being There”

Caractéristiques additionnelles  
non disponible pour les  
personnes sur place :

- Pause
- Compression
- Génération automatique TOC
- Génération automatique  
résumé des slides
- ...

# Introduction

- Applications Collaboratives
  - Objectifs : donné l'illusion d'être sur place et au-delà
  - Exemple : 3D Chat room (combine les deux objectifs)



“Toward Being There”

- Sensation d'être transporter dans un monde virtuel

“Beyond Being There”

- Sélection d'avatar

# Introduction

## □ Systèmes Collaboratifs

- Tout système informatisé visant à assister un groupe d'utilisateurs qui travaillent ensemble et interagissent dans le but de réaliser une tâche commune
- Les utilisateurs interagissent via le système informatisé et ils ont une **conscience mutuelle** de leurs activités
- Leurs interactions peuvent prendre des formes très variées :
  - Courier électronique,
  - Système de partage de document,
  - Messagerie instantanée,
  - ...

# Introduction

## □ Systèmes Collaboratifs

- Le challenge : passer de l'ordinateur outil de productivité individuelle, à travers les applications interactives, à l'ordinateur **outil de productivité collective**
- Les raisons :
  - La diversification des structures des organisations (*la décentralisation et l'extension géographique génèrent des besoins de coopération*)
  - Le développement des réseaux et d'Internet (*passage d'un réseau comme média passif de publication d'informations à une plateforme de développement d'applications interactives puis à un espace de travail pour équipes distribuées comme SourceForge*)
  - Expansion des connections permanentes (*passage de formes asynchrones à des formes synchrones de coopération*)

# Introduction

- Systèmes Collaboratifs
  - Deux classifications pour les différents types de systèmes coopératifs
    - Du point de vue de l'espace et du temps
    - Du point de vue des fonctionnalités
  - Les caractéristiques temporelles et spatiales de la coopération sont à la base de la classification la plus courante

	Même moment	Moments différents
Même lieu	<u>Réunion face à face</u> Vidéoprojecteur, Tableau blanc,...	<u>Interactions asynchrones</u> Intranet, ...
Lieux différents	<u>Interactions distribuées synchrones</u> Réunion Virtuelle, Vidéo/Visio conférences,...	<u>Interactions distribuées asynchrones</u> Courrier électronique, Internet, Forum,...

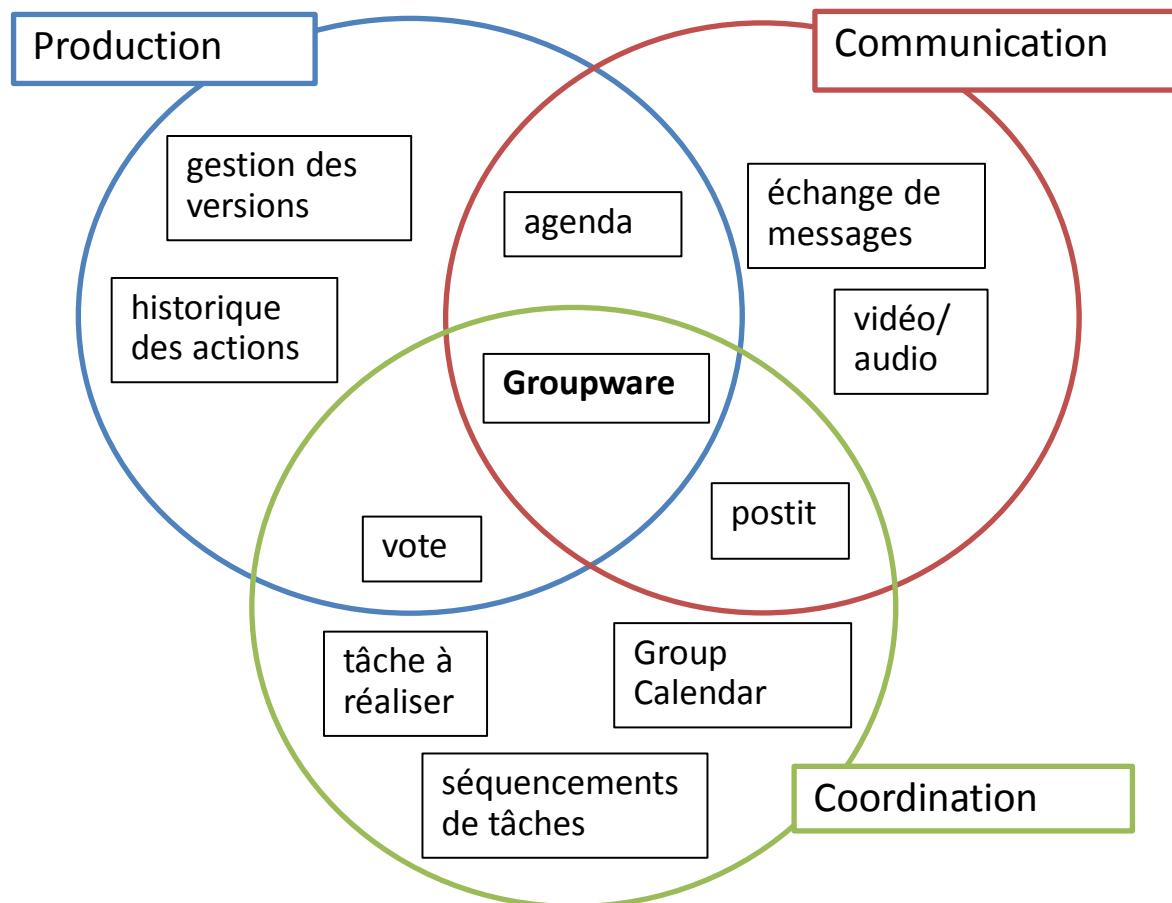
# Introduction

- Systèmes Collaboratifs
  - Les caractéristiques fonctionnelles d'un système coopératif :
    - La **communication** (*échange d'informations entre les acteurs*)
    - La **production** et le **partage** (*partage d'un espace d'objets commun, enrichi par des contributions individuelles et collectives*)
    - La **coordination** (*les règles d'interactions entre les acteurs et entre les acteurs et les objets partagés*)
  - Ces trois fonctions ne sont pas indépendantes ce que reflète le modèle du **trèfle fonctionnel**

# Introduction

## □ Systèmes Collaboratifs

- Les caractéristiques fonctionnelles d'un système coopératif :



# Introduction

- Systèmes Collaboratifs
  - Problèmes pour lesquels les systèmes collaboratifs apportent des solutions :
    - Préparation de documents, brochures, papiers...
    - Développement logiciel dans tout les phases (analyse, design, code, test)
    - Art & Science pour les collaborations entre artistes ou scientifiques
    - Education à travers le e-learning
    - Médecine (télémédecine)
    - Interaction social (trafic, température...)
    - ...

# Introduction

- Systèmes Collaboratifs
  - Au carrefour de plusieurs domaines de l'informatique et des sciences humaines et sociales :
    - Les **systèmes distribués** : la conception d'interfaces multiutilisateurs, la décentralisation du contrôle et des données, ...
    - Les **sciences humaines** : les fondements théoriques en psychologie, sociologie,...
  - Un système coopératif doit prendre en compte des aspects techniques : design et implémentation

# Introduction

- Systèmes Collaboratifs
  - Couplage de vues : les effets des actions de chacun sur les environnements des autres (quoi et quand reçoit un utilisateur suite à l'action d'un autre?)
  - Gestion de la conscience de groupe : rendre les utilisateurs conscients des autres et de leurs actions, les systèmes multiutilisateurs, ne doivent pas donner l'illusion à chaque utilisateur d'être seul, mais au contraire assister et encourager la propagation des activités entre utilisateurs
  - Gestion de la concurrence : protéger contre les actions génératrices d'incohérence

# Introduction

- Systèmes Collaboratifs
  - Assurer la **synchronisation** entre les contributions des différents utilisateurs : mettre en place la persistance des informations avec des fonctionnalités appropriées comme l'annulation, la fusion de versions,...
  - Mécanisme de **réPLICATION** : quels objets doivent être centralisés sur une machine et quels objets doivent être dupliqués sur toutes les machines
  - Assurer la **cohérence** entre les copies multiples : Les diverses copies d'une même donnée doivent être cohérentes
  - Présenter des propriétés ergonomiques

# Introduction

- Systèmes Collaboratifs
  - La nature des contextes coopératifs permet classiquement de distinguer deux grandes classes de systèmes :
    - La coopération s'organise autour de notions comme les *conférences* et les *sessions* qui permettent aux utilisateurs de se retrouver dans des  *(ou *) où ils collaborent en mode synchrone ou asynchrone (**produits groupware**)**
    - La coopération s'organise autour des notions de *processus* et de *tâche* : on parle alors de système de gestion de flux de travail (**système de workflow**)

# Les groupwares

- Le développement fulgurant ces dernières des réseaux et des moyens de communications a eu pour conséquence:
  - Globalisation, mondialisation
  - Constitution des équipes virtuelles
  - Travail à distance et mobilité
- ... de cette prise de conscience de la possibilité de travailler ensemble via une infrastructure en réseau...

# Les groupwares

- 1984 : Travail Coopératif Assisté par Ordinateur (TCAO) (en anglais, Computer Supported Cooperative Work )
- TCAO est le domaine qui étudie le travail coopératif et les différentes manières de l'informatiser
  - domaine qui cherche à comprendre la nature et les caractéristiques du travail coopératif tout en ayant comme objectif de concevoir des technologies appropriées pour assister ce type de travail en commun
- 16th ACM Conference on Computer Supported Cooperative Work (CSCW 2013)

# Les groupwares

- Le groupe représente le centre d'intérêt principal pour les approches TCAO
- En 1988, est né le terme **groupware** qui caractérise les logiciels conçus pour les groupes
- Groupware (en français Collecticiels) :
  - ensemble d'aides informatiques spécialisées conçues pour être utilisées par des groupes de travail coopératif formés autour d'un projet
- Exemples de groupware : Egroupware, Agora, Sharepoint, Lotus notes, ...

# Les groupwares

- EGroupware (<http://www.egroupware.org/>)
  - Un outil de travail collaboratif en ligne sous licence GNU GPL (gratuit)
  - Permet de gérer les contacts, les rendez-vous, les tâches,...
  - Conçu en PHP avec une interface Web qui permet d'accéder aux données de n'importe quel plateforme
  - Accessible à partir de n'importe quel navigateur Web
  - Fonctionnalités :
    - Carnet d'adresses, calendrier, client de messagerie
    - Gestionnaires de projets, gestionnaires des tâches,
    - Gestionnaires de documents, gestionnaires de ressources, feuilles de temps, Wiki,...

# Les groupwares

- EGroupware (<http://www.egroupware.org/>)

The screenshot shows the EGroupware interface for managing contacts. The left sidebar contains a navigation menu with various modules: Carnet d'adresses, Ajouter, Recherche avancée, importexport, Traqueur, Sondages, Wiki, Communiqués, Gestionnaire de fichiers, InfoLog, Base de connaissances, Calendrier, Ressources, Feuille de temps, phpfreechat, Signets, Messagerie, and Gestionnaire de projets. The main window is titled "Carnet d'adresses" and displays a form for editing a contact. The form includes fields for Société: Nom, Prénom, General tab (Nom, Titre, Rôle, organisation, Département, Rue, Ville, Pays), Categories tab (Carnet d'adresses dropdown), Details tab, Links tab, and several sections for phone numbers (Bureau, Téléphone portable, Privé, Fax) and email/internet (Url, E-mail). At the bottom, there are buttons for Enregistrer, Appliquer, and Annuler, along with a link to modify organization members.

Accueil Aide Rechercher Déconnexion User, demo - Lundi 17.12.2012 Europe / Berlin Ajouter ...

Carnet d'adresses X Carnet d'adresses X InfoLog X

EGroupware - EPL Demo [Carnet d'adresses] - Mozilla Firefox

https://demo-epl.egroupware.de/egw/index.php?menuaction=addressbook.addressbook\_ui.edit

Société: Nom, Prénom

Type contact

General Catégories Privé Détails Liens

Numéros de téléphone

Bureau Téléphone portable Privé Fax

Plus ...

Email & Internet

Url Privé E-mail Privé

Enregistrer Appliquer Annuler Modifier tous les membres de l'organisation

Motorisé par Stylite's EGroupware Version EPL 11.1

# Les groupwares

- EGroupware (<http://www.egroupware.org/>)

The screenshot shows the EGroupware interface with the following details:

- Header:** Accueil, Aide, Rechercher, Déconnexion, User, demo - Lundi 17.12.2012, Europe / Berlin, Ajouter ...
- Left Sidebar (Gestionnaire de fichiers):**
  - Favorites: Groupe Stylite, File a file
  - Gestionnaire de fichiers Menu: Listview (selected), Your home directory, Users and groups, Startfolder
  - Links: Traqueur, Carnet d'adresses, Sondages, Wiki, Communiqués, InfoLog, Base de connaissances, Calendrier, Ressources, Feuille de temps, phpfreechat, Signets, Messagerie.
- Central Area:**
  - Cabinet d'adresses X, Gestionnaire de fichiers X
  - Path: Chemin /home
  - File List: montre 1 - 3 de 3
  - Buttons: Current directory, Rechercher, Select action, Up arrow.
  - File List Table:

Type	Nom	Taille	Modifié	Créé	Permissions	Propriétaire	Groupe	Commentaire	Actions
Folder	demo	0	13.11.2008 16:47	13.11.2008 16:47	drwx-----	demo	root		<input type="checkbox"/>
Folder	Demo Guests	0	20.11.2008 16:14	20.11.2008 16:14	d---rwx---	root	Demo Guests		<input type="checkbox"/>
Folder	Stylite	0	27.07.2009 00:00	27.07.2009 00:00	d---rwx---	root	Stylite		<input type="checkbox"/>
- Footer:** Motorisé par Stylite's EGroupware Version EPL 11.1

# Les groupwares

- EGroupware (<http://www.egroupware.org/>)

The screenshot shows the EGroupware interface with the "Ressources" module selected. The left sidebar contains a navigation menu with various links such as Accueil, Aide, Rechercher, Déconnexion, User, demo - Lundi 17.12.2012, Europe / Berlin, Ajouter ..., Ressources, Ressources Menu (Liste des ressources, Ajouter une ressource), importexport, Traqueur, Carnet d'adresses, Sondages, Wiki, Communiqués, Gestionnaire de fichiers, InfoLog, Base de connaissances, Calendrier, Feuille de temps, phpfreechat, Signets, Messagerie, and Gestionnaire de projets.

The main content area displays a list of resources under the "Gestionnaire de fichiers" tab. The table has columns: Nom (Description courte), utilisable Quantité, Catégorie, Emplacement, Resource / Accessories, test, and Actions. The data is as follows:

Nom (Description courte)	utilisable Quantité	Catégorie	Emplacement	Resource / Accessories	test	Actions
Beamer	2	IT				
Beamer	2	Administrator, Stylite				
MAC	10	IT				
MacBookPro	10	Administrator, Stylite				
Meetingroom 1	2	Meeting rooms				
Meetingroom for 5-10 persons	2	Administrator, Stylite				
Meetingroom 2	3	Meeting rooms				
Meetingroom for 20-30 persons	3	Administrator, Stylite				

At the bottom of the table, there is an "Ajouter" button. The footer of the page states "Motorisé par Stylite's EGroupware Version EPL 11.1".

# Les groupwares

- EGroupware (<http://www.egroupware.org/>)

The screenshot shows the EGroupware Project Manager interface. The left sidebar contains a navigation menu with various modules: Gestionnaire de projets, Traqueur, Carnet d'adresses, Sondages, Wiki, Communiqués, Gestionnaire de fichiers, Base de connaissances, Calendrier, and Ressources. The main area displays a table of projects with the following data:

ID de projet	Titre	Priorité	Catégories	Propriétaire	Coordinateur	Comptabilité	Assistant	Membre de projet	Date de début prévu	Date de fin prévu	Budget prévu	Temps prévu re-planned	Etat	Dernière modification	Actions
P-2012-0001	Root Level Project	1		User, demo					18.12.2012 18.12.2012	27.12.2012			<span style="color: green;">OK</span>	16.12.2012 07:43 User, demo	<span style="color: green;">OK</span> <span style="color: green;">Edit</span> <span style="color: red;">Delete</span> <span style="color: green;">Details</span>
P-2012-0002	Pokus	1		User, demo					16.12.2012	15.02.2013			<span style="color: red;">OK</span>	16.12.2012 09:20 User, demo	<span style="color: red;">OK</span> <span style="color: red;">Edit</span> <span style="color: red;">Delete</span> <span style="color: red;">Details</span>

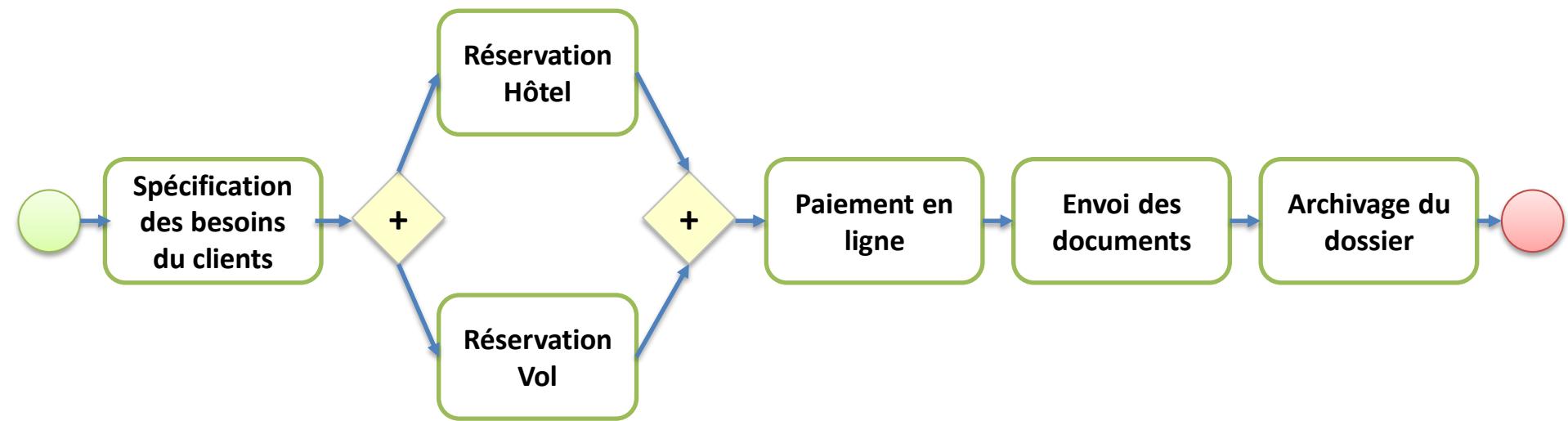
At the bottom, a footer note reads "Motorisé par Stylite's EGroupware Version EPL 11.1".

# Les workflows

- Qu'est ce qu'un processus métier?
  - *Un ensemble de procédures et d'activités plus ou moins liées qui réalisent collectivement un objectif métier, en général au sein d'une structure organisationnelle définissant des rôles et des relations fonctionnelles*
  - Un processus métier peut être entièrement inclus dans une organisation simple ou peut s'étendre sur plusieurs organisations
  - Un processus métier peut combiner des activités automatiques et des activités manuelles

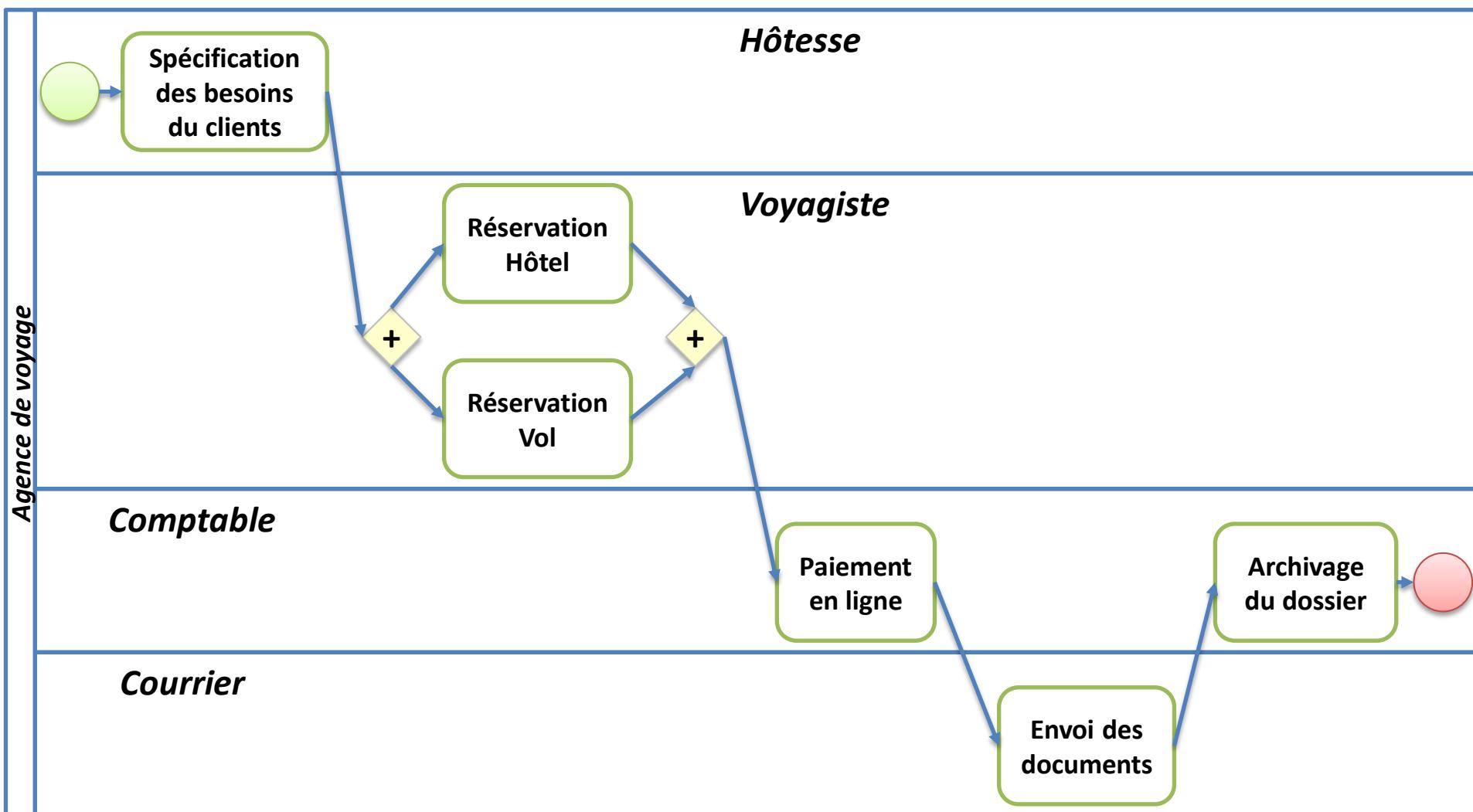
# Les workflows

- Exemple de modèle de processus : l'organisation d'un voyage



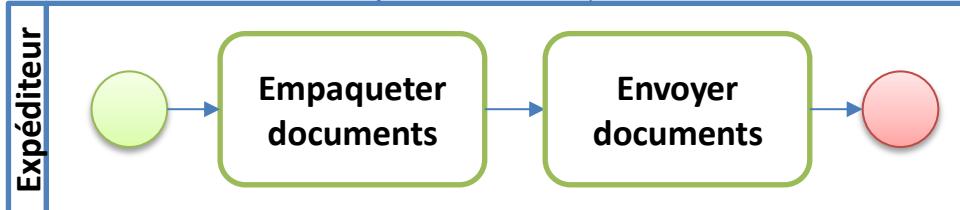
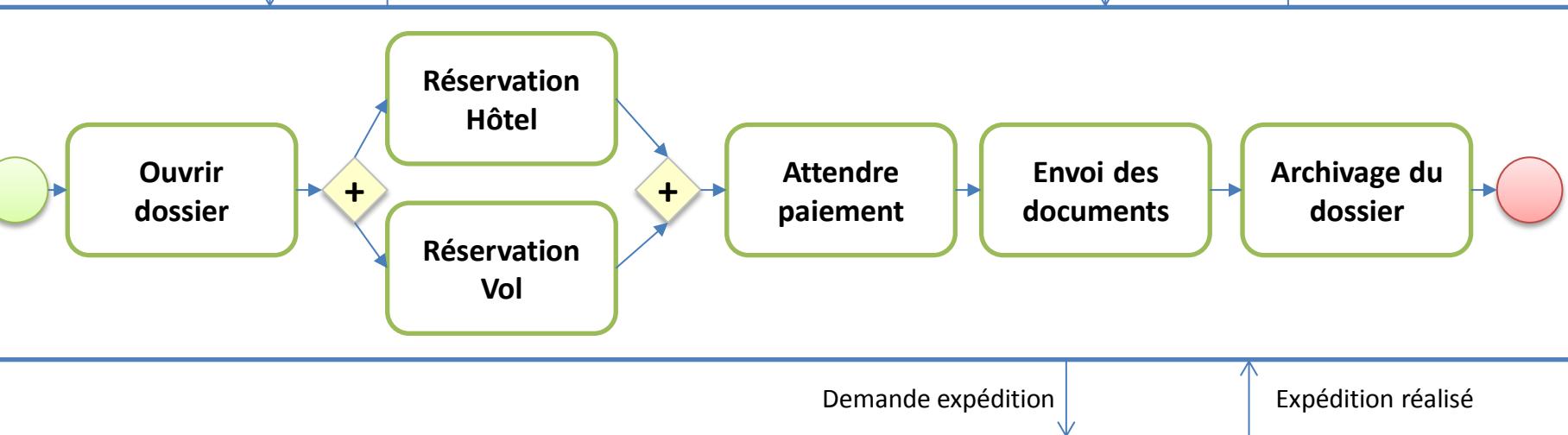
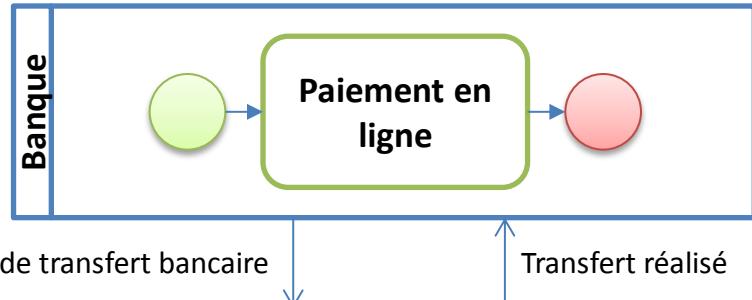
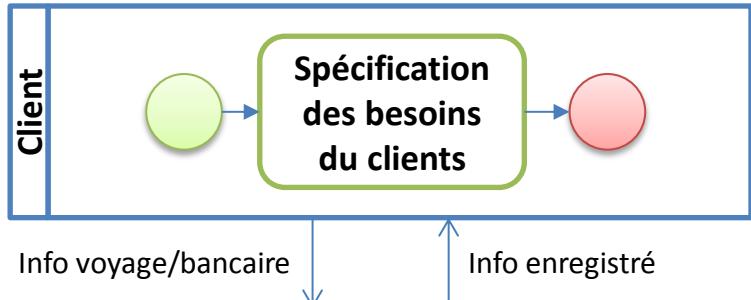
# Les workflows

## □ Processus intra-organisationnel



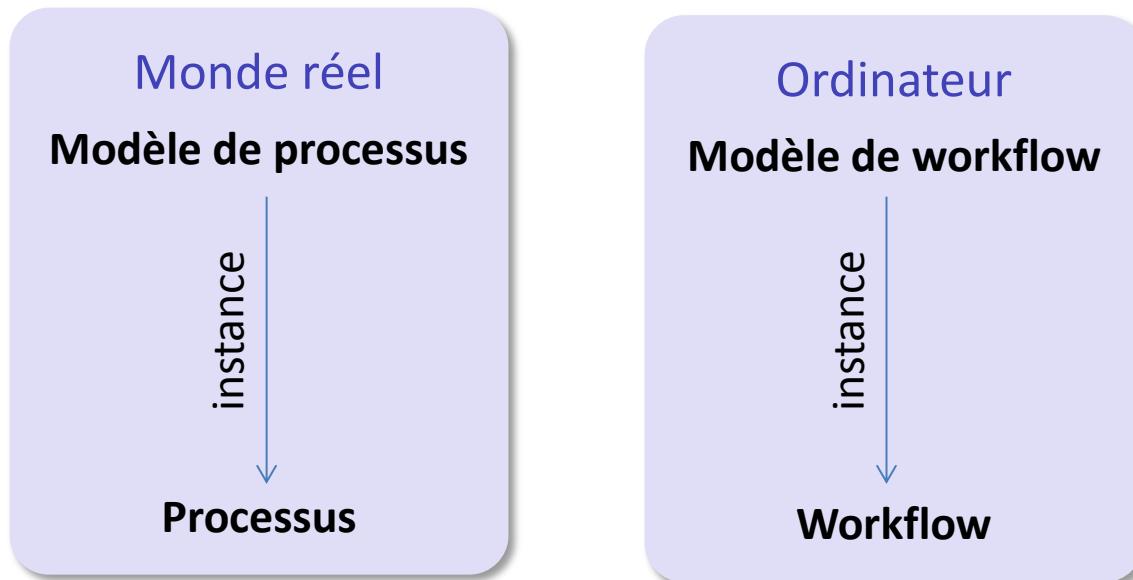
# Les workflows

## □ Processus inter-organisationnel



# Les workflows

- Qu'est ce qu'un Workflow?
  - Une **technologie TCAO** qui s'occupe de l'**automatisation** de **tout ou partie** d'un **processus métier** durant lequel des documents, des informations ou des tâches sont transférés d'un participant à un autre en vue d'actions réalisées conformément à des règles préétablies
- Processus et workflows

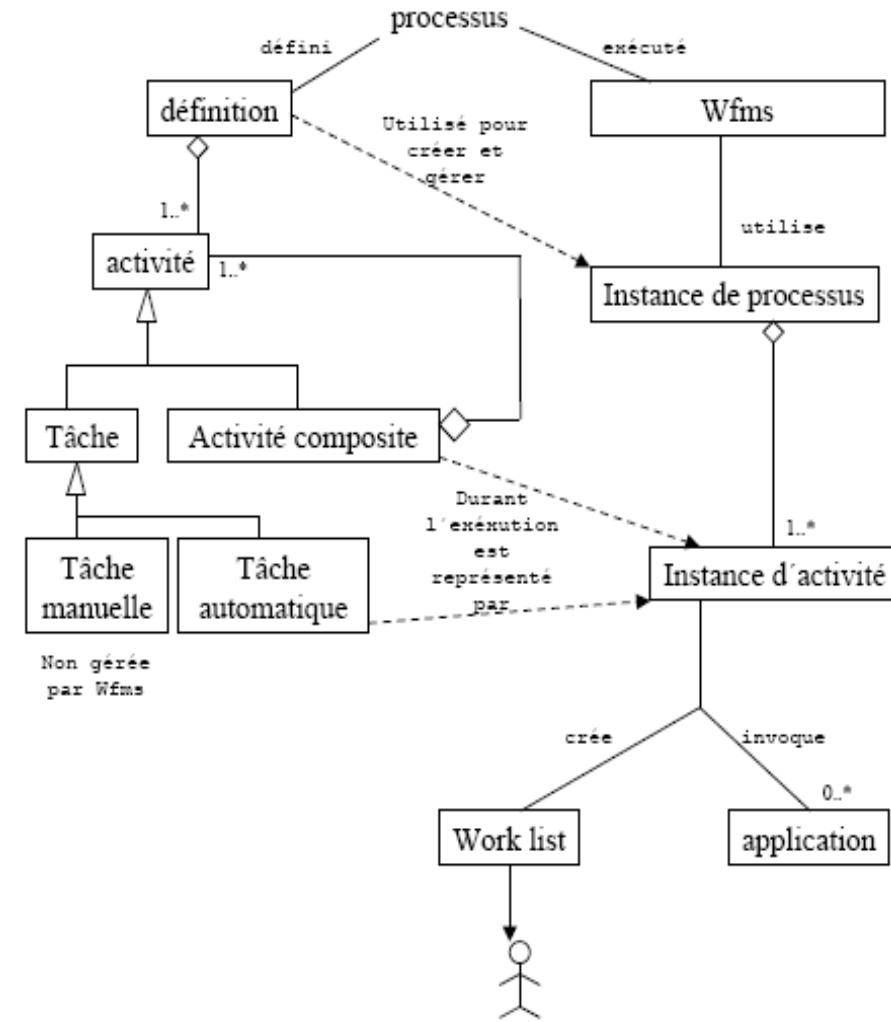


# Les workflows

- On appelle « workflow » les aspects opérationnels d'un processus :
  - La séquence des tâches et qui les réalisent, le flot de données qui supporte ces tâches, et les mécanismes qui permettent de mesurer, suivre et contrôler ces tâches
- Comment contrôler l'exécution des workflows ?
  - **Workflow Management System (WfMS)** : un système qui définit, gère et contrôle l'exécution des workflows
  - C'est un système capable d'interpréter la définition du processus, interagir avec les participants et si besoin invoquer des applications spécifiques

# Les workflows

- Un modèle de processus est *décomposable récursivement* en sous-processus jusqu'aux tâches ou *activités élémentaires* qui peuvent être manuelles ou automatiques
- Le WfMS maintient en interne une description des processus en cours d'exécution appelés instance de processus
- Les instances d'activités au sein des instances de processus correspondent à des invocations ou à des affectations d'activités à des participants humains

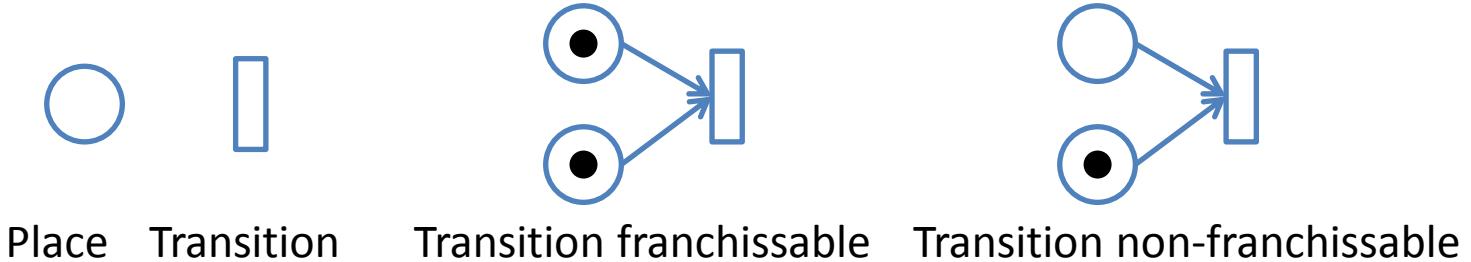


# Les workflows

- La **modélisation** d'un processus doit permettre de répondre aux questions suivantes :
  - **Dimension logique** : Quelles sont les activités à réaliser (quoi)? Quand faut-il les réaliser (quand)?
  - **Dimension organisationnelle** : Quelles sont les compétences nécessaires (qui)?
  - **Dimension informationnelle** : Quels sont les outils et les informations nécessaires (comment)?
- **Modélisation orientée activités** : la forme la plus classique de modélisation, elle définit les processus en termes d'activités et de transitions entre les activités

# Les workflows

- Plusieurs possibilités :
  - Modélisation par réseaux de Petri (simple, enrichi,...)
  - Modélisation par langages spécifiques (BPMN, BPEL, YAWL,...)
  - ...
- La modélisation de processus par Réseaux de Petri
  - Symboles et principe de base

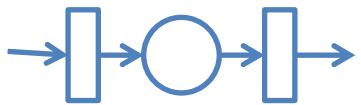


# Les workflows

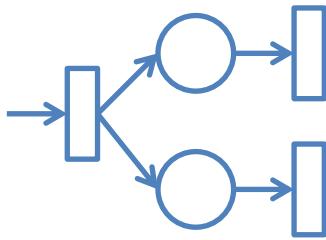
- Modélisation de processus par Réseaux de Petri
  - Les activités sont représentées par des transitions
  - Les flots de contrôle sont représentés par des (sous)-RdP
  - Un workflow bien structurés est un RdP qui a les propriétés suivantes :
    - Une seule place d'entrée et une seule place de sortie
    - Chaque place et chaque transition se trouve sur un chemin allant de la place initiale à la place finale
    - La consommation d'un jeton dans la place initiale produit un et un seul jeton dans la place finale

# Les workflows

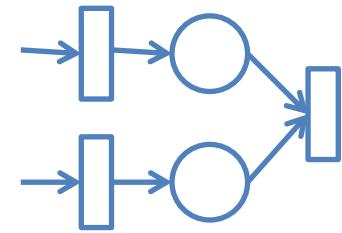
- Modélisation de processus par Réseaux de Petri
  - Les patrons de base pour flot de contrôle



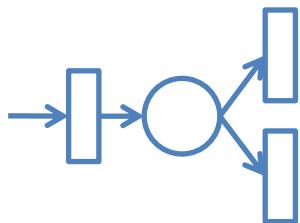
Séquence



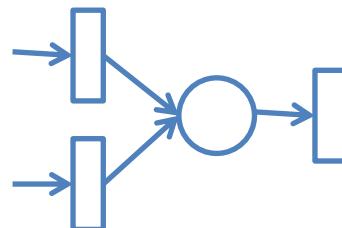
Branchement multiple  
(AND-split)



Synchronisation  
(AND-join)



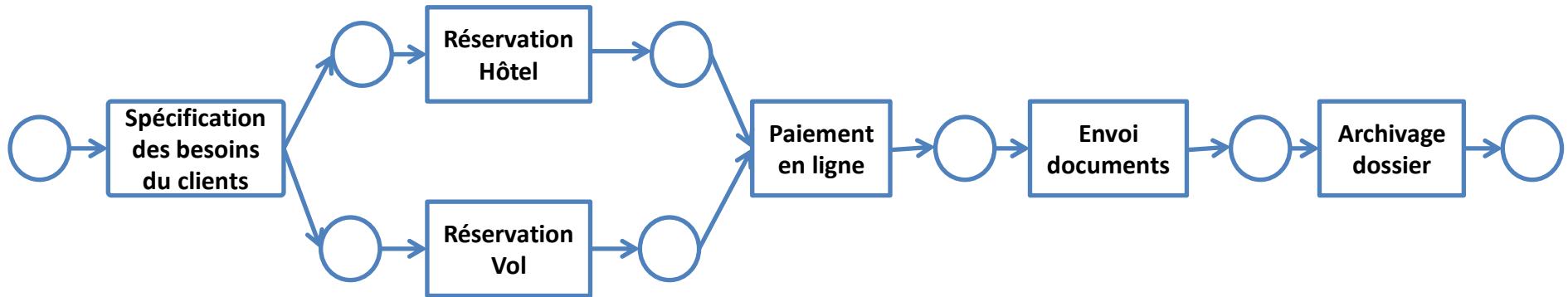
Choix simple  
(XOR-split)



Jonction  
(XOR-join)

# Les workflows

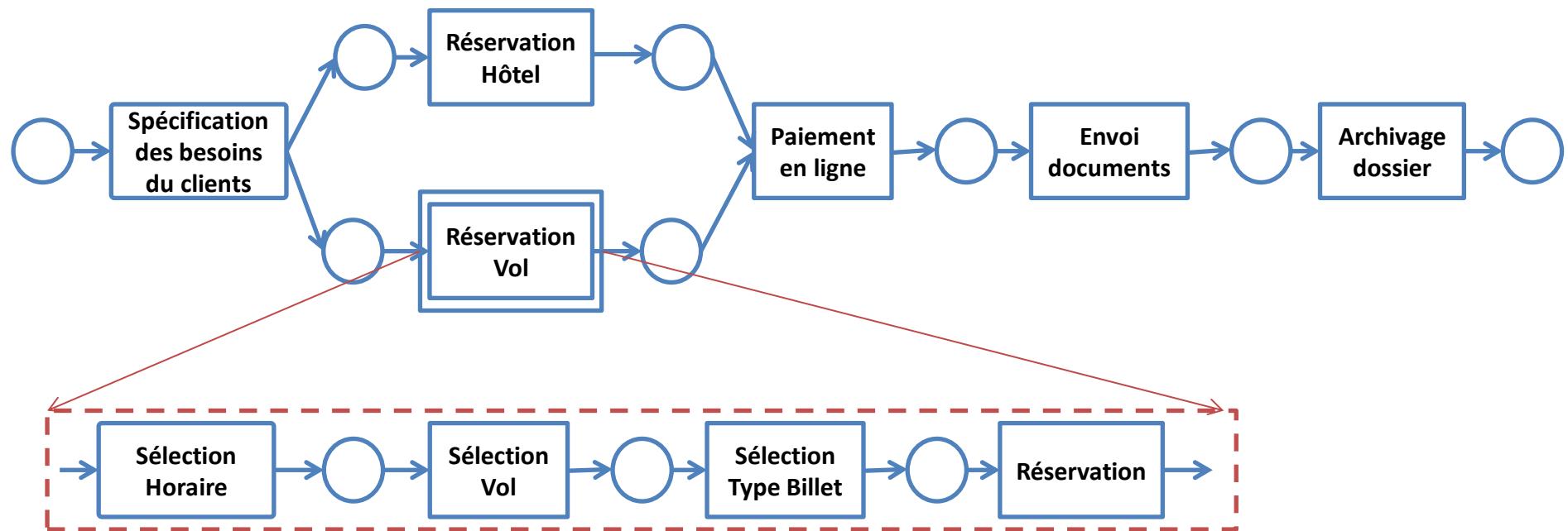
- Modélisation de processus par Réseaux de Petri
  - Exemple : modélisation du processus organisation d'un voyage par un RdP classique



- Beaucoup de modèles sont des enrichissements des RdP classiques : les RdP hiérarchisés, les RdP colorés, les RdP temporisés, ...

# Les workflows

- Modélisation de processus par Réseaux de Petri
  - Exemple : modélisation du processus organisation d'un voyage par des RdP hiérarchisés qui permettent de représenter le raffinement d'une activité composée en sous-activité



# Les workflows

- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - [www.yawl-system.com](http://www.yawl-system.com)
    - Langage de workflow basé sur les RdP, XML et les patrons
    - Patrons :
      - Projet *Workflow Patterns* débuté en 1999 Eindhoven University of Technology et Queensland University of Technology
      - Plus de 20 patrons
      - <http://www.workflowpatterns.com/>

# Les workflows

- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle de base : sequence, parallel split (AND-split), synchronization (AND-join), exclusive Choice (XOR-split), simple merge (XOR-join)
    - Patrons de flots de contrôle avancés : multi-Choice (OR-split), structured synchronizing merge (variante OR-join), multi-merge (variante XOR-join), structured discriminator

# Les workflows

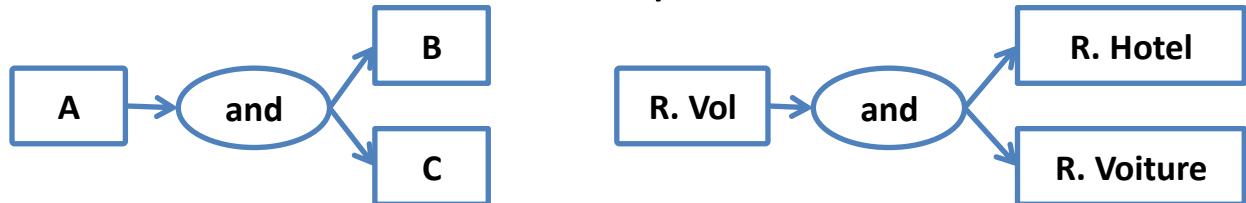
- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle de base : [Sequence](#)
      - On parle de séquence lorsqu'au cours d'un processus, les activités sont exécutées les unes à la suite des autres, et que c'est le seul itinéraire possible
      - Dans la séquence des deux activités A et B, A est activée d'abord puis à la fin de son exécution, B est activée



- $\text{depAct}(A,B)$  et  $\text{condAct}(B) = \text{terminé}(A)$
- Exécution acceptée : AB

# Les workflows

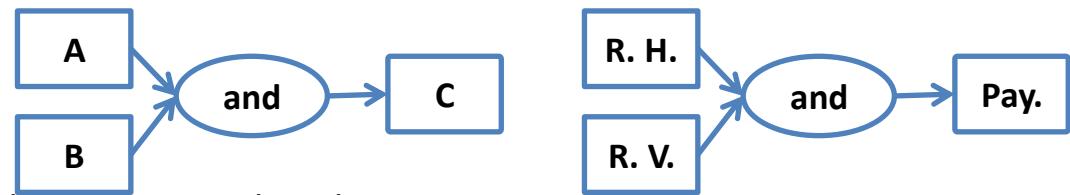
- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle de base : AND-split
      - Il y a branchement multiple lorsqu'un itinéraire unique se sépare en deux ou plusieurs itinéraires différents dans le but de réaliser deux ou plusieurs activités en parallèle
      - Après l'exécution de l'activité A, les activités B et C sont activées et exécutées en parallèle



- $\text{depAct}(A,B)$  et  $\text{depAct}(A,C)$  ,
  - $(\text{condAct}(B) = \text{terminé}(A))$  et  $(\text{condAct}(C) = \text{terminé}(A))$
  - Exécutions acceptées : ABC; ACB

# Les workflows

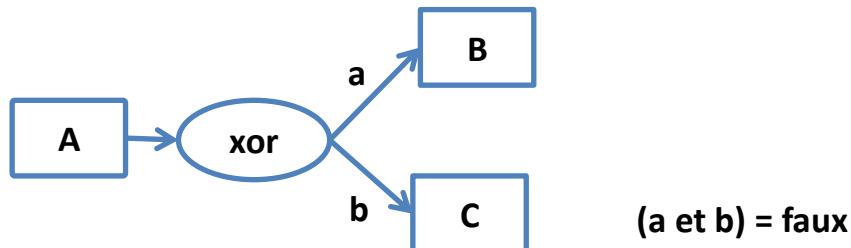
- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle de base : AND-join
      - Il y a synchronisation lorsque deux ou plusieurs itinéraires parallèles convergent vers un itinéraire unique et que l'on assure la synchronisation des itinéraires, c'est-à dire qu'on ne passe à l'activité suivante que lorsque toutes les activités parallèles sont terminées
      - L'activation de l'activité C est conditionnée par la fin de l'exécution des activités A et B



- $\text{depAct}(A,C)$  et  $\text{depAct}(B,C)$  et
- $\text{condAct}(C) = \text{terminé}(A) \text{ et } \text{terminé}(B)$
- Exécutions acceptées : ABC; BAC

# Les workflows

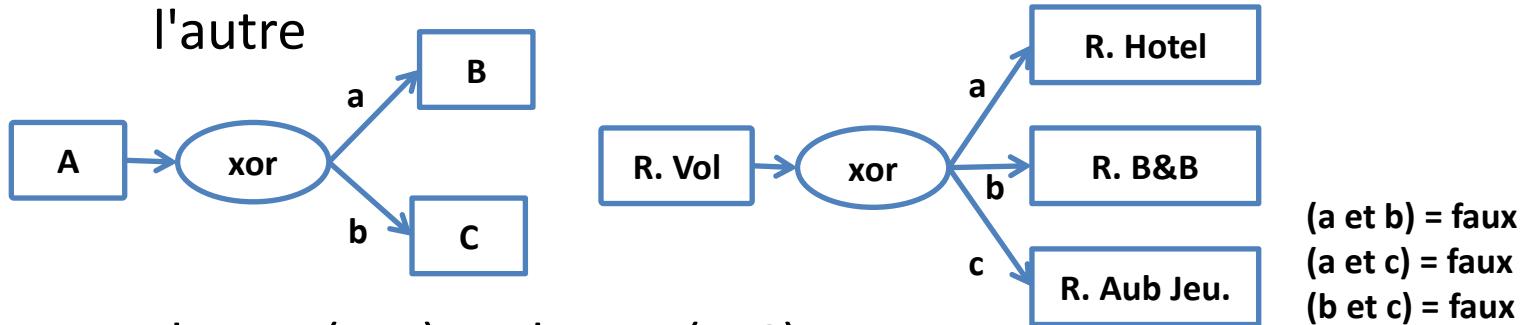
- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle de base : XOR-split
      - On parle de choix exclusif lorsqu'un itinéraire s'ouvre sur plusieurs itinéraires possibles et que le cas d'exécution suit un seul de ces itinéraires, selon les conditions de transition. Le choix se fait sur la base de l'évaluation de la condition associée aux transitions du patron, ces conditions étant exclusives l'une de l'autre
      - Après la fin d'exécution de l'activité A, soit l'activité B soit l'activité C est activée, à l'exclusion l'une de l'autre



# Les workflows

- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle de base : XOR-split

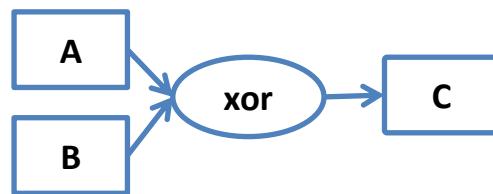
- Après la fin d'exécution de l'activité A, soit l'activité B soit l'activité C est activée, à l'exclusion l'une de l'autre



- $\text{depAct}(A,B)$  et  $\text{depAct}(A,C)$ , et
    - $\text{condAct}(B) = \text{terminé}(A) \text{ et } \text{cond}(B);$
    - $\text{condAct}(C) = \text{terminé}(A) \text{ et } \text{cond}(C) \text{ et }$
    - $\text{non}(\text{cond}(B) \text{ et } \text{cond}(C))$
    - Exécutions acceptées : AB; AC

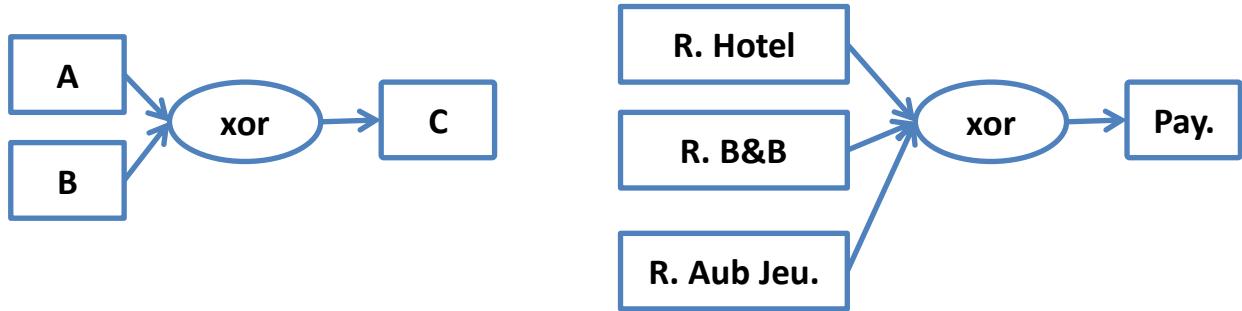
# Les workflows

- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle de base : XOR-join
      - Il y a jonction simple lorsque deux ou plusieurs itinéraires convergent vers une même activité.
      - Il ne s'agit pas de la synchronisation de plusieurs itinéraires, mais plutôt de la jonction de plusieurs itinéraires alternatifs
      - L'activation de l'activité C est conditionnée par la fin de l'exécution de l'activité B ou de l'activité A



# Les workflows

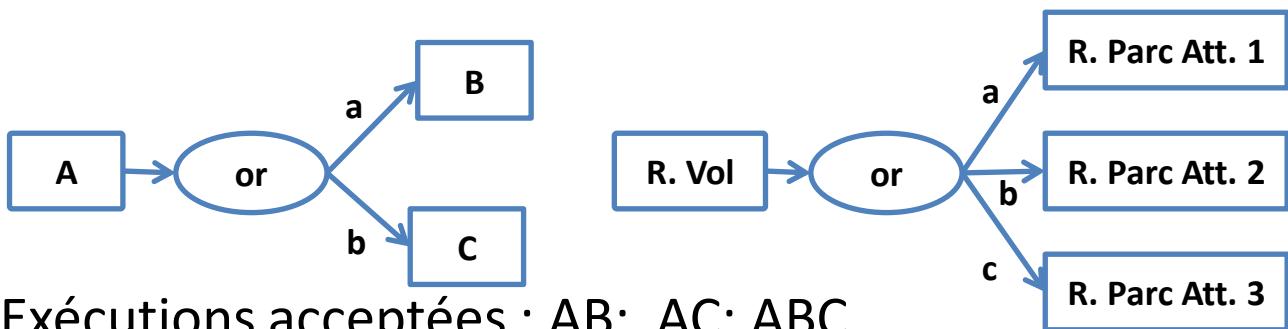
- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle de base : XOR-join
      - L'activation de l'activité C est conditionnée par la fin de l'exécution de l'activité B ou de l'activité A



- $\text{depAct}(A,C)$  et  $\text{depAct}(B,C)$ , et  
 $\text{condAct}(C) = \text{terminé}(A)$  ou  $\text{terminé}(B)$ , et  
 $\text{non}(\text{terminé}(A) \text{ et } \text{terminé}(B))$
- Exécutions acceptées : AC; BC

# Les workflows

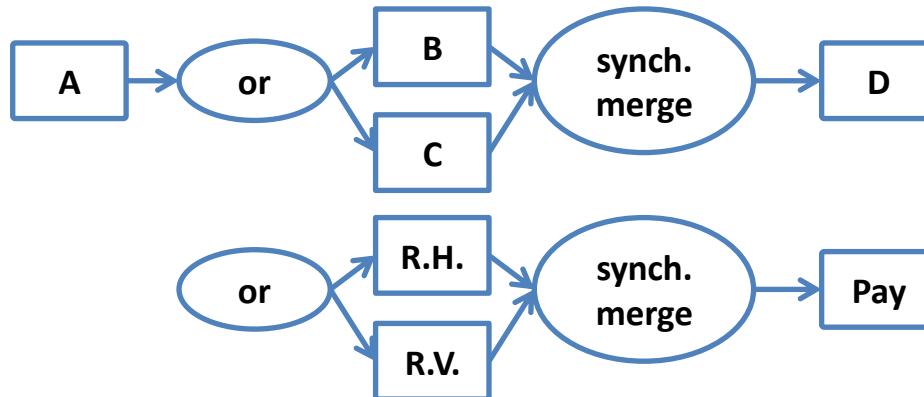
- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle avancés : OR-split
      - On parle de choix multiple lorsqu'un itinéraire s'ouvre sur plusieurs itinéraires possibles et que le cas d'exécution suit un ou plusieurs de ces itinéraires, selon les conditions de transition
      - Après la fin d'exécution de l'activité A, l'activité B, l'activité C ou les deux sont activées, en fonction des conditions associées aux transitions



- Exécutions acceptées : AB; AC; ABC

# Les workflows

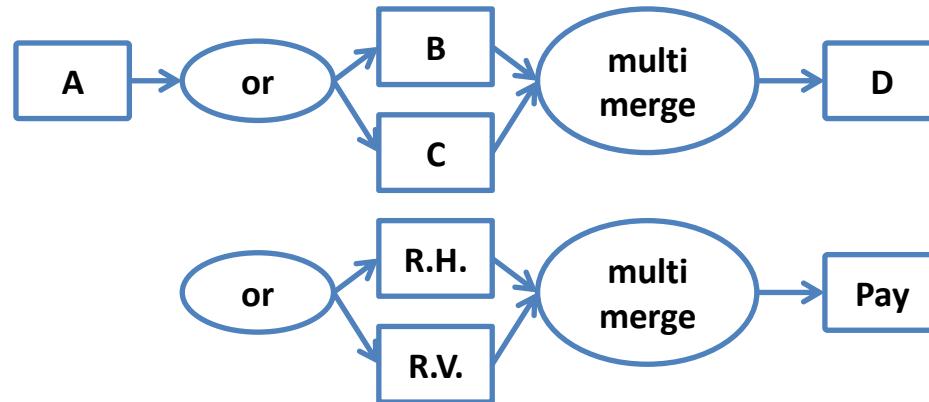
- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle avancés : var. OR-join
      - Il y a jonction Il y a jonction synchronisée (synchronizing merge) lorsque deux ou plusieurs itinéraires convergent vers un itinéraire unique et que l'on assure la synchronisation des itinéraires actifs



- Exécutions acceptées: ABD; ACD; ABCD; ACBD

# Les workflows

- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle avancés : var. OR-join
      - Il y a **jonction multiple** (multi-merge) lorsque deux ou plusieurs itinéraires convergent vers un itinéraire unique et que l'on assure l'activation de ce dernier autant de fois qu'il y a d'itinéraires actifs



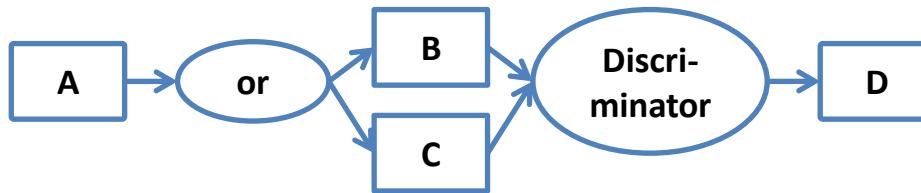
- Exécutions acceptées: ABD; ACD; ABCDD; ACBDD; ABDCD; ACDBD

# Les workflows

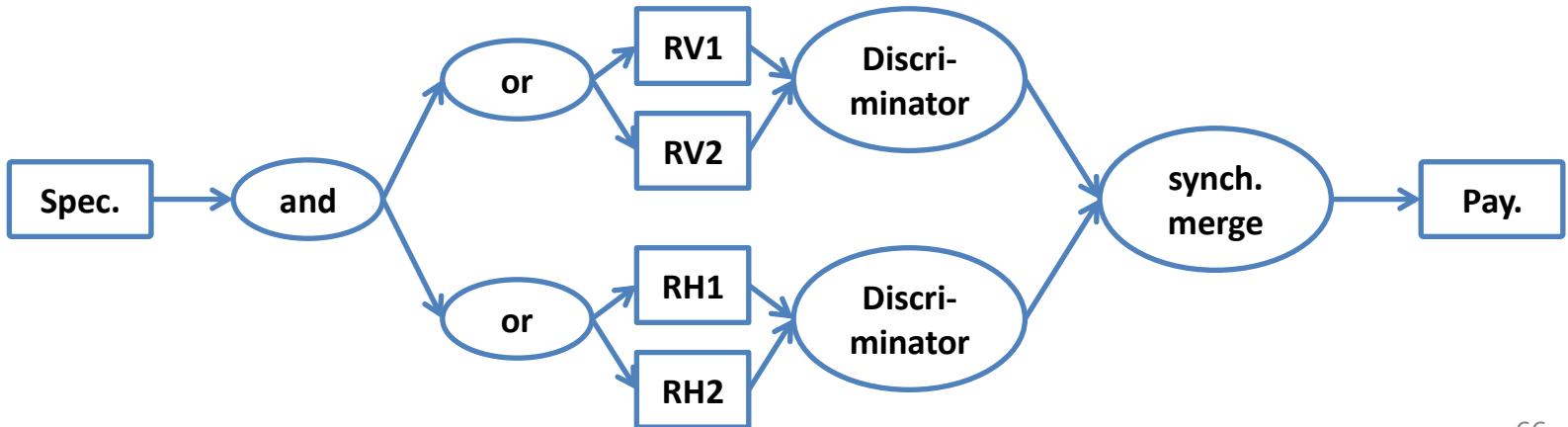
- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle avancés : [Discriminateur](#)
      - On utilise un discriminateur (structured discriminator) lorsque deux ou plusieurs itinéraires convergent vers un itinéraire unique dont on assure l'activation une seule fois
      - L'activation se fait après la terminaison d'un premier itinéraire
      - Les terminaisons des autres sont ignorées

# Les workflows

- Modélisation de processus par langage de workflow
  - YAWL (Yet Another Workflow Langague)
    - Patrons de flots de contrôle avancés : Discriminateur



- Exécutions acceptées: ABCD; ABDC; ACDB



# Les workflows

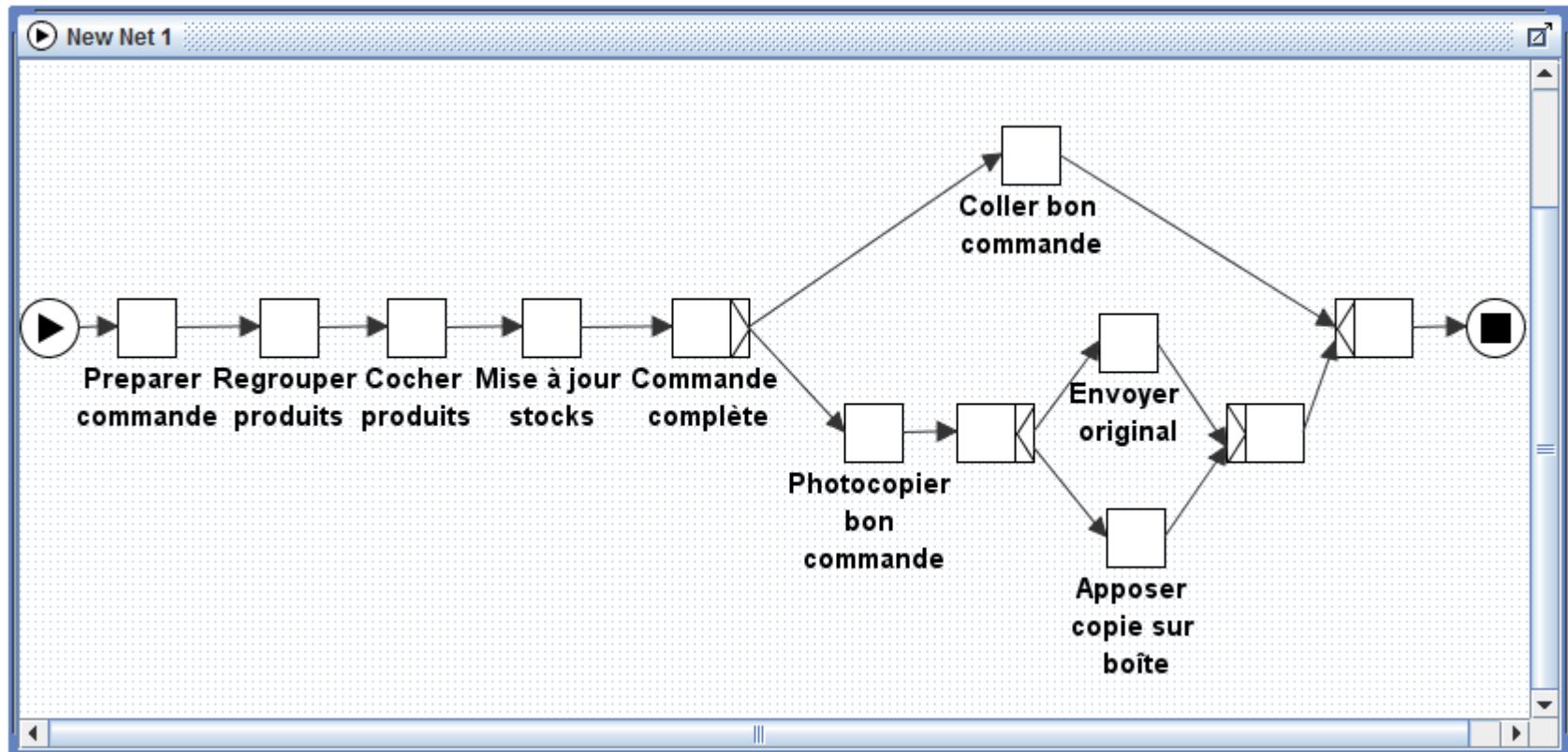
- Modélisation de processus par langage de workflow
  - Autres
    - Organisme de standardisation (consortium)
    - Workflow Management Coalition (WfMC) : fondé en 1993, organisation internationale à but non lucratif de vendeurs, utilisateurs, concepteurs et chercheurs de systèmes de workflow (<http://www.wfmc.org>)
    - Sa mission : établir des standards pour la terminologie et l'interopérabilité de produits de workflow
    - Parmi les outils : XPDL (XML Process Definition Language), langage de définition de processus métier à l'aide du langage XML, utilisé par un moteur de workflow. XPDL 1.0 en 2002 et XPDL 2.2 en 2010

# Les workflows

- Modélisation de processus
  - Exercice : modélisation d'un processus de préparation des commandes
    - Lorsqu'un représentant termine de préparer une commande, le responsable de l'entrepôt procède au regroupement des produits de la commande
    - Ensuite, il cochera les produits regroupés, sur son ordinateur, avant de mettre à jour le fichier des stocks
    - Si la commande est complète, alors il collera le bon de commande sur la boîte. Sinon, il photocopiera le bon de commande, enverra l'original du document au responsable des achats et apposera la copie sur la boîte.
    - Cette dernière étape débutera un autre processus pour la livraison

# Les workflows

- Modélisation de processus
  - Solution : modélisation dans le langage YAWL



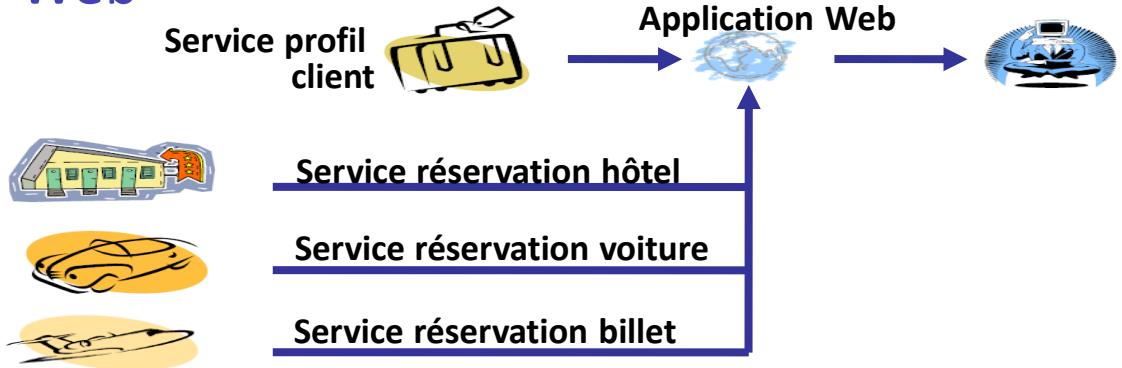
# Les workflows

- Interopérabilité dans un contexte métier **inter**-entreprise :  
Approche **Services Web**
  - La convergence entre des systèmes de gestion de processus et le Web suscite un intérêt grandissant depuis plusieurs années
  - Objectif : faire du Web l'infrastructure qui supportera des systèmes de gestion de workflow distribués et hétérogènes
  - Beaucoup de systèmes offrent seulement une interface Web et utilisent d'autres mécanismes pour la communication (sockets, RPC, CORBA). On parle alors de **systèmes interfacés sur le Web** par opposition aux **systèmes déployés sur le Web** pour lesquels le Web est la technologie de communication et distribution

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**

- Exemple :



- Les activités peuvent être :
  1. sous la responsabilité de participants différents,
  2. mises en œuvre par des solutions techniques différentes,
  3. dans des organisations différentes ou pas.
- Les applications réalisant ces tâches de façon complètement automatique et opaque s'appellent des **services Web**

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - Un service Web = une application modulaire basée sur les protocoles Internet, qui fournit un service spécifique, qui respecte le format d'échange de données XML
  - Objectif de l'architecture des services Web :
    - **Invoquer une fonction distante** (mêmes intentions que les architectures les plus anciennes en terme d'accès distant) sur un serveur Web distant avec le protocole HTTP
    - **Interopérabilité** : baser sur des standards ouverts, sans composant spécifique à un langage ou un système d'exploitation
    - **Faible couplage** : limiter au maximum les dépendances entre les différents éléments de l'application

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche Services Web
- Définition du World Wide Web Consortium (W3C)<sup>(1)</sup>

*« a Web service is a **software application** identified by a **URI**, whose **interfaces** and **binding** are capable of being **defined, described and discovered** by XML artifacts. A Web service supports direct interactions with other software agents using **XML based messages** exchanged via **Internet-based protocols** »*
- Un service Web est un **système logiciel** identifié par un **URI**, dont les **interfaces** et les **liaisons** sont définies et décrites en **XML**. Sa définition peut être **découverte** (dynamiquement) par d'autres systèmes logiciels. Ces autres systèmes peuvent ensuite interagir avec le service Web d'une façon décrite par sa définition, en utilisant des **messages XML** transportés par des **protocoles Internet**

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - **Système logiciel** : programme
  - **URI** : Uniform Resource Identifier, URL (<http://www.w3.org/>), mail (<mailto:toto@dauphine.fr>), FTP (<ftp://ftp.mido.dauphine.fr/docs/>)
  - **Interface** : une description des opérations proposées par le composant logiciel (même esprit que les interfaces Java)
  - **Liaison** (binding) : spécification du protocole et du format des données utilisés pour échanger des messages en vue de l'utilisation d'une interface
  - **Découverte** (dynamique) : obtention de la description d'un service Web
  - **XML** : eXtensible Markup Language, pré-requis indispensable pour faire des services Web
  - **Protocoles Internet** : bas niveau TCP/IP, haut niveau (HTTP)

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - Un service Web est donc :
    - un programme décrit en XML et identifié par un URI proposant diverses fonctionnalités que d'autres programmes peuvent
    - découvrir dynamiquement et utiliser grâce à des protocoles décrits en XML basés sur l'échange de messages
      - écrits en XML
      - transmis par HTTP, TCP, ...
  - Une vision plus simple
    - Un Web service est un programme accessible par internet par l'intermédiaire de messages XML transmis par HTTP

# Les workflows

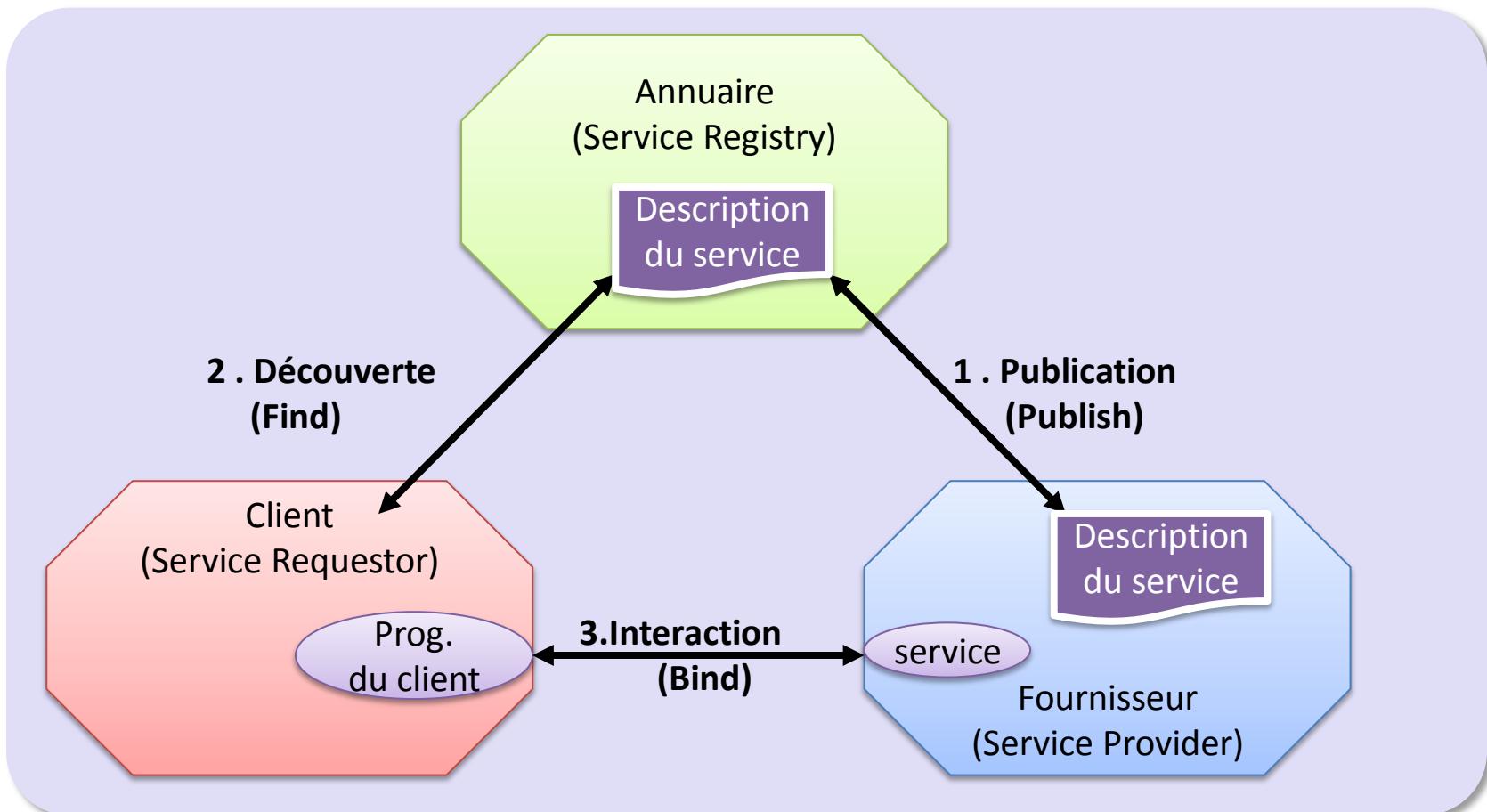
- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - Les principaux acteurs
    - **Le fournisseur de service (*service provider*)**
      - définit un service, publie sa description dans un annuaire, réalise les opérations
    - **L'annuaire (*discovery agency*)**
      - reçoit et enregistre les descriptions de services publiées par les fournisseurs
      - reçoit et répond aux recherches de services lancées par les clients
    - **Le client (*service requestor*)**
      - obtient la description du service grâce à l'annuaire, invoque (utilise) le service

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - Les principales technologies
    - **SOAP (*Simple Object Access Protocol*)**
      - protocole pour l'échange de données structurées au format XML
      - protocole de transport de données basé sur HTTP
    - **WSDL (*Web Service Description Language*)**
      - langage de description de l'interface du Web service
    - **UDDI (*Universal Description, Discovery and Integration*)**
      - annuaire permettant l'enregistrement et la recherche des descriptions de Web services

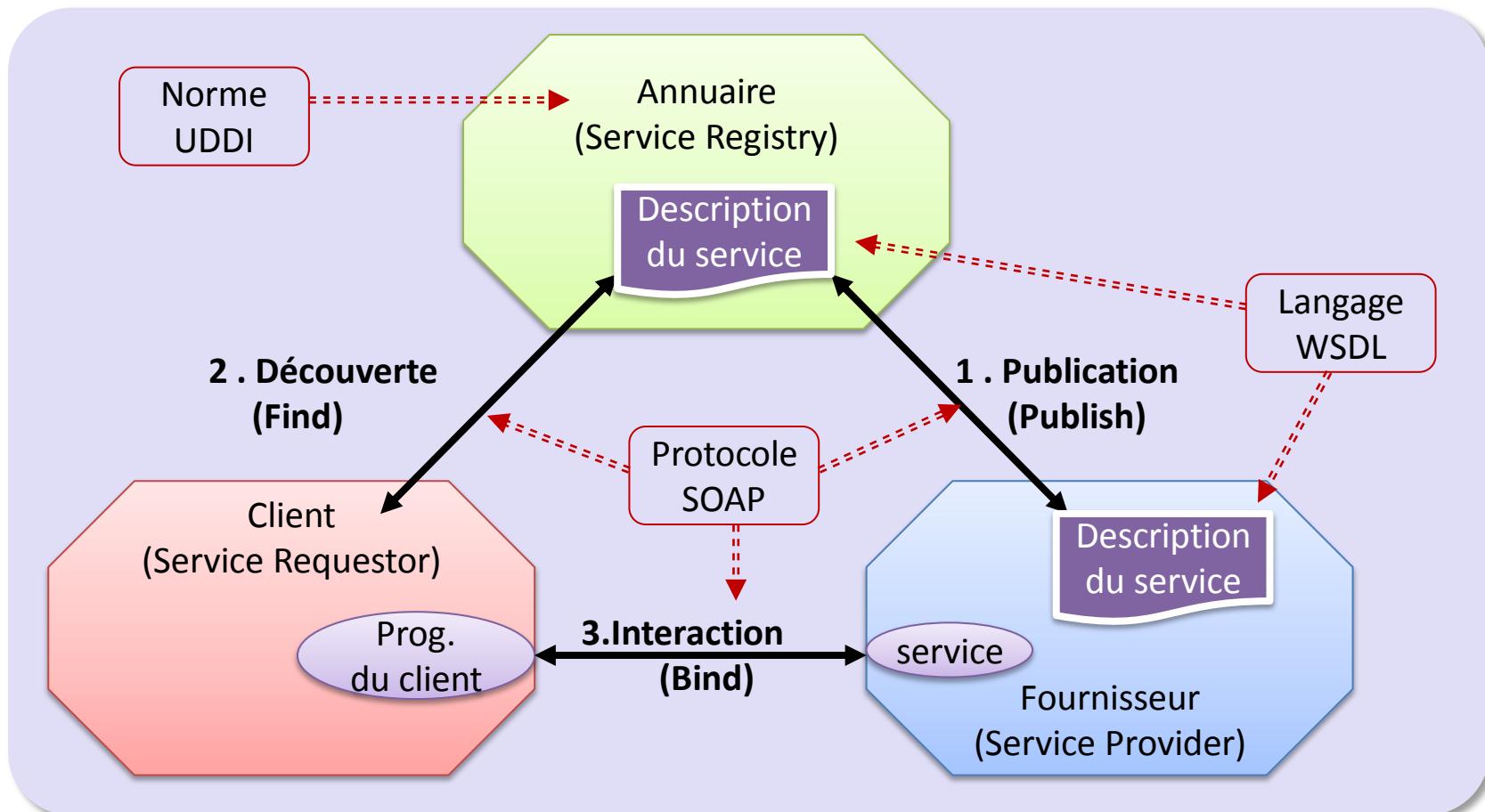
# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche *Services Web*



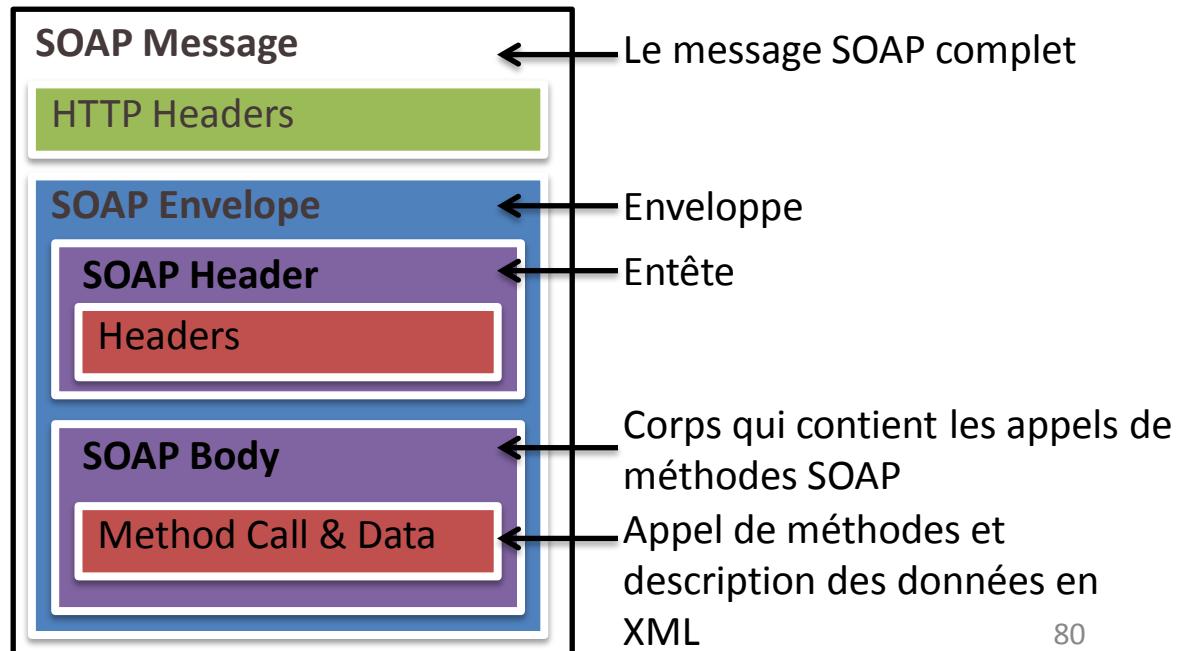
# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche *Services Web*



# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - SOAP
    - Standard W3C de définition d'un protocole assurant des appels de procédures à distance (comme RPC) mais s'appuyant principalement sur XML et HTTP
  - Le message



# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - SOAP
    - SOAP utilise une en-tête HTTP spécifique (**SOAPAction**) constitué d'un URI qui précise le but du message (par exemple l'opération appelée)
    - Exemple : *CurrencyConvertor* est un service qui permettent de connaître le taux de conversion entre deux devises
      - *ConversionRate* est l'opération qui permet de faire la conversion d'une devise à une autre
      - Résultat : échange de messages entre un client et un serveur pour savoir le taux de conversion entre EUR et USD

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche Services Web
  - SOAP
    - Exemple de requête SOAP-HTTP

```
POST /CurrencyConvertor.asmx HTTP/1.1
```

```
Host: www.webservicex.net
```

```
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: length
```

```
SOAPAction: "http://www.webserviceX.NET/ConversionRate"
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ConversionRate xmlns="http://www.webserviceX.NET/">
      <FromCurrency>EUR</FromCurrency>
      <ToCurrency>USD</ToCurrency>
    </ConversionRate>
  </soap:Body>
</soap:Envelope>
```

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche Services Web
  - SOAP
    - Exemple de réponse SOAP-HTTP

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
  
  <soap:Body>  
    <ConversionRateResponse xmlns="http://www.webserviceX.NET/">  
      <ConversionRateResult> 1.293 </ConversionRateResult>  
    </ConversionRateResponse>  
  </soap:Body>  
</soap:Envelope>
```

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - WSDL
    - Une interface qui cache le détail de l'implémentation du service permettant une utilisation indépendante
      - De la plate-forme utilisée
      - Du langage utilisé
    - Le fichier WSDL est au format XML et regroupe toutes les informations nécessaires pour interagir avec le service Web
      - Les méthodes, les paramètres et valeurs retournées
      - Le protocole de transport utilisé, et la localisation du service

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche Services Web
  - WSDL
    - Un document XML qui commence par la balise `<definitions>` et contient les balises suivantes
    - `<types>` : décrit la structure de donnée transmise dans un message et définit les types de données du service
    - `<messages>` : décrit l'ensemble des données transmises au cours de l'opération (ce peut être une requête ou une réponse), et fournit les détails des messages utilisés comme paramètres
    - `<portType>` : décrit l'ensemble des opérations abstraites applicables au service, et définit les noms des méthodes et leurs paramètres

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche Services Web
  - WSDL
    - Un document XML qui commence par la balise `<definitions>` et contient les balises suivantes
      - `<binding>` : décrit la façon dont un ensemble d'opérations abstraites (`portType`) est lié à un port selon un protocole réel, et définit le protocole à utiliser pour invoquer le service Web
      - `<port>` : spécifie un point d'entrée (`endpoint`) comme la combinaison d'un `<binding>` et d'une adresse réseau, spécifie l'emplacement actuel du service
      - `<service>` : décrit un ensemble de points finaux (`endpoints`) du réseau appelés ports

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - UDDI
    - Standard né sous l'initiation de Microsoft, IBM Sun, Oracle, Compaq, HP, Intel, SAP...
    - Standard pour faciliter le développement des collaboration entre partenaires dans le cadre d'échanges commerciaux (de type B2B)
    - Le cœur de UDDI est un annuaire qui contient des informations techniques et administratives sur les entreprises et les services qu'elles publient
      - publier des infos sur une entreprise et ses services
      - découvrir des infos sur une entreprise et ses services

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - UDDI
    - L'annuaire UDDI comportent plusieurs catégories de données :
      - **Pages blanches** : recensent les entreprises et contiennent des informations telles que : le nom de l'entreprise, ses coordonnées,...
      - **Pages jaunes** : contiennent, au format WSDL, la description des services déployés par l'entreprise. Les services sont répertoriés par catégorie.
      - **Pages vertes** : elles fournissent des informations techniques détaillées sur les services.

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - UDDI
    - Type de référentiel UDDI
      - Public : <http://www.webservicex.net/>,  
<http://www.xmethods.net>, ...
      - Privé ou d'entreprise : Microsoft([uddi.microsoft.com](http://uddi.microsoft.com)),  
IBM([www.ibm.com/services/uddi](http://www.ibm.com/services/uddi)), ...
    - Accès
      - JAXR (Java API for XML Registries) définit une API pour naviguer dans un référentiel UDDI
      - Naviguer dans la base de services (sélectionner un service, accéder au WSDL, essayer)

# Les workflows

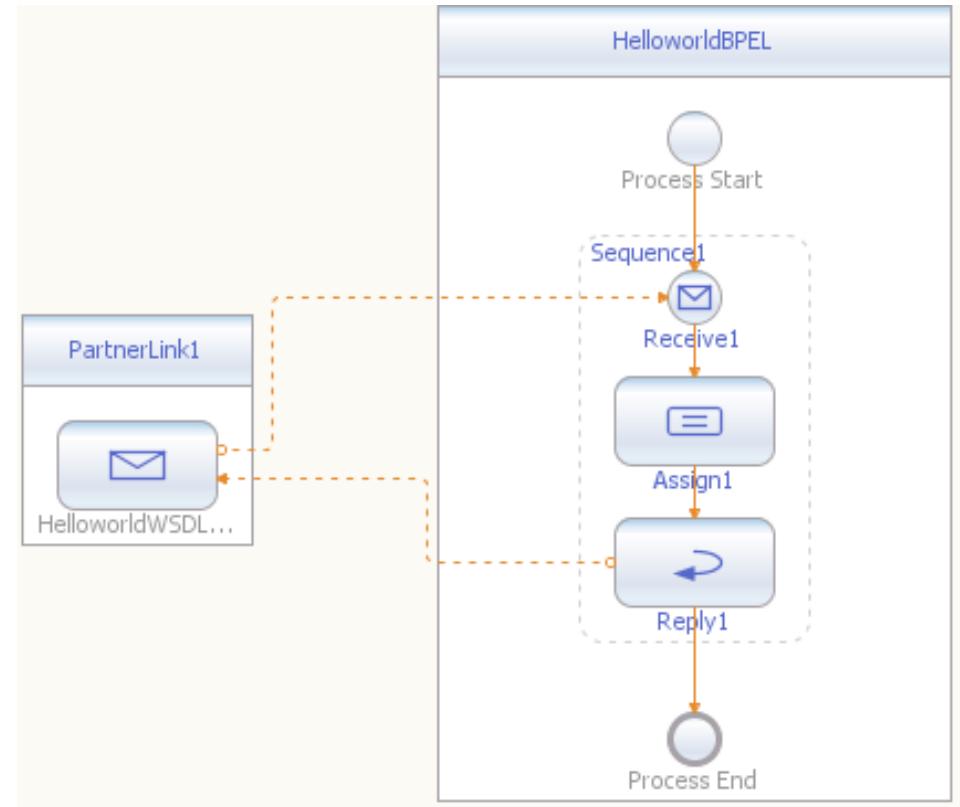
- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - **Orchestration de service :**
    - L'orchestration décrit l'interaction entre deux ou plusieurs services web au niveau :
      - de l'ordre d'exécution des interactions
      - des messages
  - **BPEL (Business Process Execution Language)**
    - Consiste en un langage XML dessiné pour définir et gérer les orchestrations de processus (des activités des processus métiers et les interactions entre des services web)
    - Définit la coordination des interactions entre l'instance du processus et ses partenaires (une couche d'abstraction de la logique d'orchestration pour les services Web impliqués)

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche **Services Web**
  - Les éléments du processus BPEL sont :
    - les liens de **partenaires** (`partnerLink`) : service avec lequel le procédé échange les informations, typés par `partnerLinkType`, définit le rôle de chacun des deux partenaires
    - les **activités** : le processus est constitué d'activités (`<invoke>`, `<receive>`, `<reply>`, `<wait>`, `<assign>`,...) liées par un flot de contrôle (`if`, `while`, `flow`, `pick`,...)
    - les **données** : le processus dans BPEL a un état maintenu par des variables contenant des données, par des conditions de transition ou de jointure, par un passage des variables par affectation

# Les workflows

- Interopérabilité dans un contexte métier inter-entreprise :  
Approche *Services Web*
  - Exemple de processus BPEL simple, *Helloworld* contenant :
    - un lien partenaire
    - trois activités



- ... TP et Projet

# Systèmes collaboratifs distribués

- La conception de systèmes collaboratifs
  - La plupart des systèmes CSCW ont été développées avec une architecture centralisée (wikis, SVN,...). La gestion de données est restée centralisée (les données sont conservées en un lieu unique)
  - Les collaborations ont évolué de petits groupes pour des millions d'utilisateurs sur le même groupware (Wikipedia, ...). Les SGF et SGBD classiques sont trop rigides et ne passent pas à l'échelle. Exploration des systèmes pair-à-pair
  - Exemple : système P2P d'édition collaborative

# Systèmes collaboratifs distribués

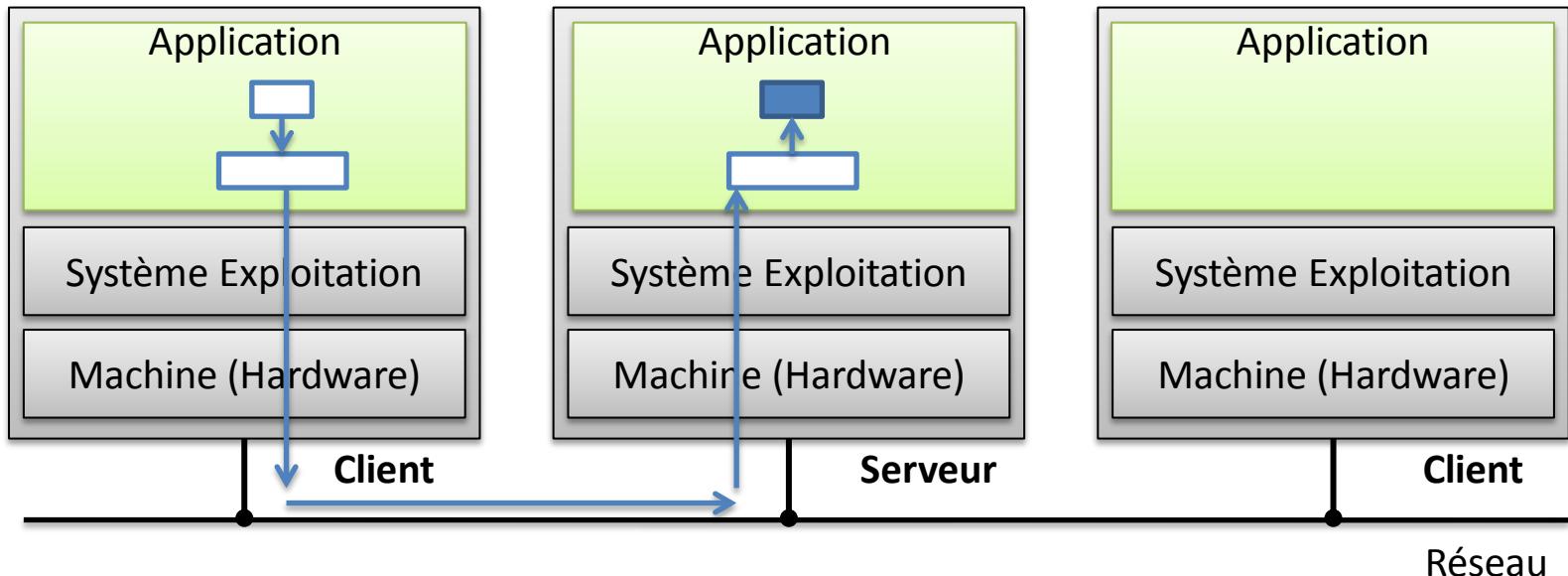
- La conception de systèmes collaboratifs
  - Un système P2P d'édition collaborative permet de distribuer le service, tolérer des pannes, améliorer performances, et de partager les coûts de l'infrastructure sous-jacente [Data Consistency for P2P Collaborative Editing. Gérald Oster, Pascal Urso, Pascal Molli and Abdessamad Imine. In Proceedings of the 2006 ACM Conference on Computer Supported Cooperative Work, CSCW 2006]
- Problèmes techniques :
  - Gestion des données : **duplication** et **cohérence**
  - Evolutivité (scalability),
  - Tolérance aux pannes,
  - Privacy,
  - ...

# Systèmes collaboratifs distribués

- Gestion des données
  - Pourquoi dupliquer les données ?
    - Disponibilité : permettre l'accès aux données même en cas de défaillance d'un support
    - Rapidité d'accès : en cas de passage à l'échelle, placer une copie des données "près" (en temps d'accès) de leur point d'utilisation
  - Cohérence : la contrepartie de la duplication
    - Les multiples copies d'une même donnée doivent être cohérentes (i.e. apparaître comme une copie unique)
    - Le maintien de la cohérence a un coût. Il faut faire un compromis entre le coût et la "qualité" de la cohérence

# Systèmes collaboratifs distribués

- Gestion des données
  - Duplication d'objet
    - Gestion des objets partagés



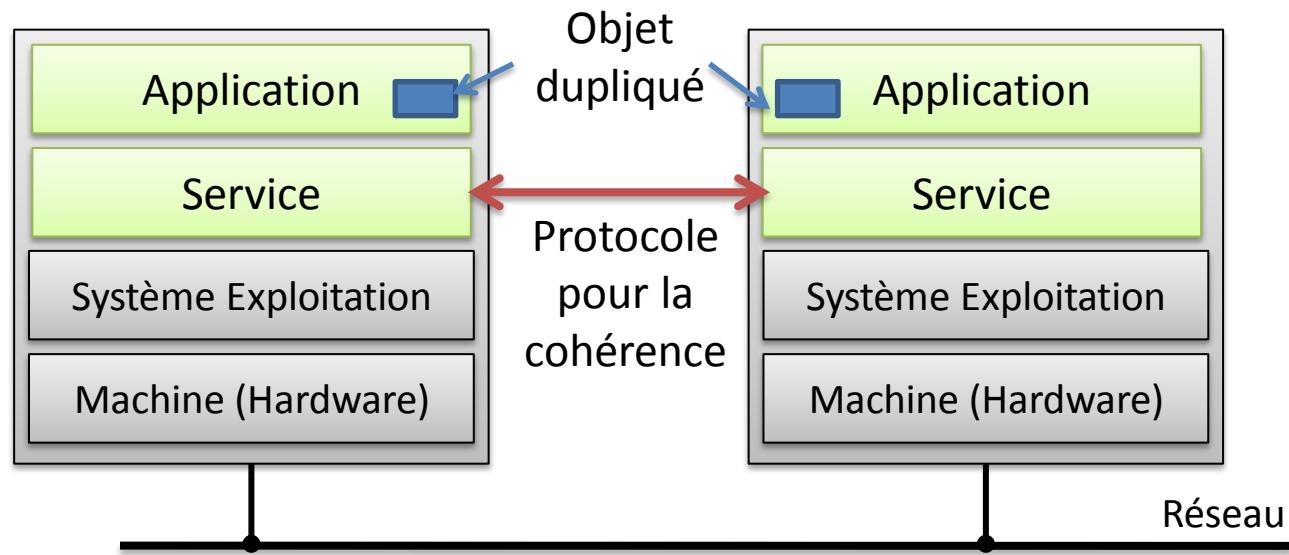
- Protection d'un objet des accès simultanés : synchronized (par l'objet), exclusion mutuelle (par le serveur)

# Systèmes collaboratifs distribués

- Gestion des données
  - Duplication d'objet
    - Gestion des objets dupliqués partagés par le systèmes distribués
    - Duplication des objets sans mécanismes de gestion de la concurrence implique des problèmes de cohérence
    - Les multiples copies ont besoin de synchronisation pour assurer que les accès concurrents s'effectue dans le bon ordre sur toutes les copies

# Systèmes collaboratifs distribués

- Gestion des données
  - Duplication d'objet
    - Gestion des objets dupliqués partagés par le systèmes distribués
    - Assurer la cohérence par le système distribué :  
Gestion des objets dupliqués partagés par la couche service (middleware)

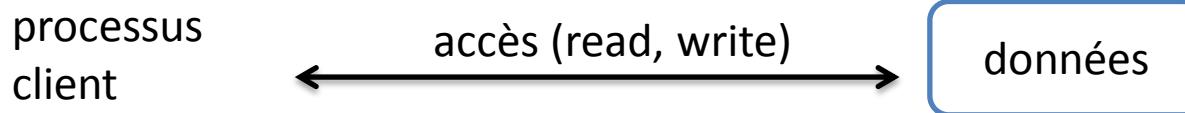


# Systèmes collaboratifs distribués

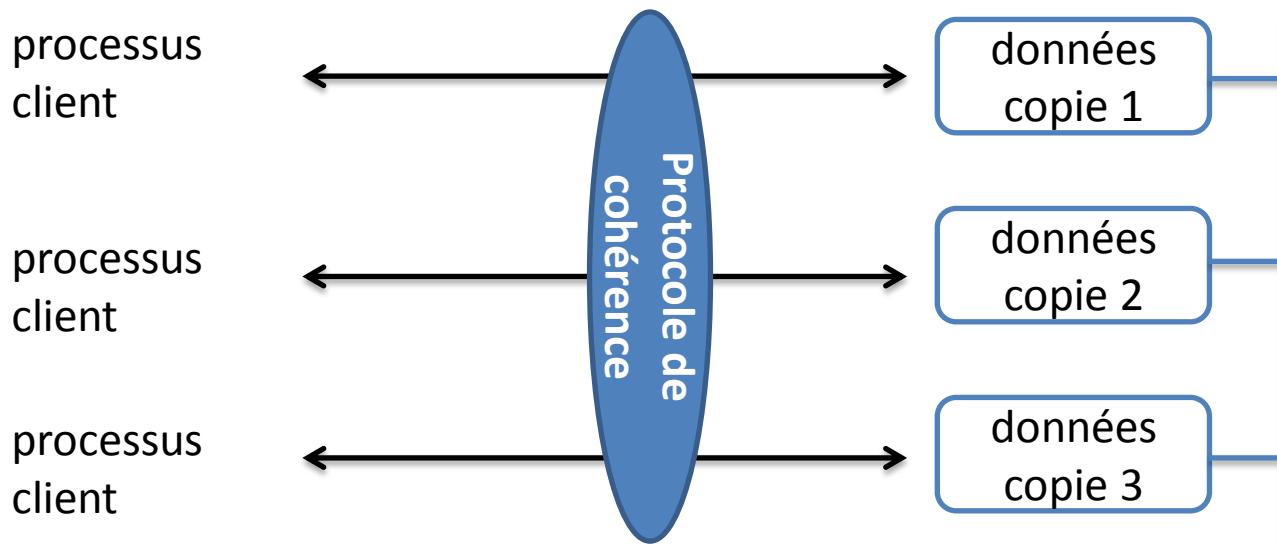
- Gestion des données
  - Duplication d'objet
    - Gestion des objets dupliqués partagés par le systèmes distribués
    - En particulier, le système distribué assure que les accès concurrents sont passés dans le *bon* ordre à l'ensemble des copies
  - Le problème de cohérence a été beaucoup étudier dans le contexte d'accès à des données partagées disponible à travers une mémoire virtuelle répartie
  - Accès aux données dupliquées : des opérations de lecture et d'écriture

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Accès aux données dupliquées



- **read(x)** : qui renvoie le contenu de l'objet x
  - **write(x,v)** : qui met à jour l'objet x avec la valeur v
- Une opération d'écriture doit être propagée à toutes les copies



# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Un **modèle de cohérence** pour des données dupliquées réparties spécifie un contrat entre un client et le système de gestion de données : si le processus client accepte de respecter certaines règles alors le système de gestion de données promet d'agir correctement
  - Une fois le modèle de cohérence défini, il faut construire un **protocole de cohérence** qui garantit que ce modèle est préservé
  - Le protocole de cohérence assure
    - l'exécution des opérations des clients, et
    - la mise en cohérence mutuelle des copies conformément à un comportement défini par un modèle de cohérence

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Normalement, un processus qui effectue une opération de lecture sur un objet  $x$  s'attend à obtenir la valeur de dernière opération d'écriture effectuée sur l'objet  $x$ .
  - Dans l'absence d'horloge globale, il est difficile de préciser qu'elle est la dernière opération d'écriture.
  - Alternatives : classification des modèles de cohérences
    - Cohérence stricte : le modèle le plus contraignant
    - Linéarisabilité
    - Cohérence séquentielle
    - Cohérence causal
    - Cohérence FIFO

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence stricte
    - Condition : *Toute lecture sur un élément x renvoie une valeur correspondant à l'écriture la plus récente sur x*
    - La réalisation d'un protocole réalisant la cohérence stricte dans un système réparti pose deux problèmes
      - définition de *l'événement le plus récent* : nécessite des horloges parfaitement synchronisées
      - réalisation *instantanée* des opérations : une opération qui nécessite un accès distant ne peut être instantanée (une opération locale lancée après une opération distante peut se terminer avant)

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - **Cohérence stricte**
    - La cohérence stricte est un modèle idéal (non réalisable), que l'on essaie d'approcher au moyen de modèles moins contraignants
  - Un point important dans la définition d'un modèle de cohérence est de définir exactement quel comportement est acceptable en présence de conflit entre des opérations

# Systèmes collaboratifs distribués

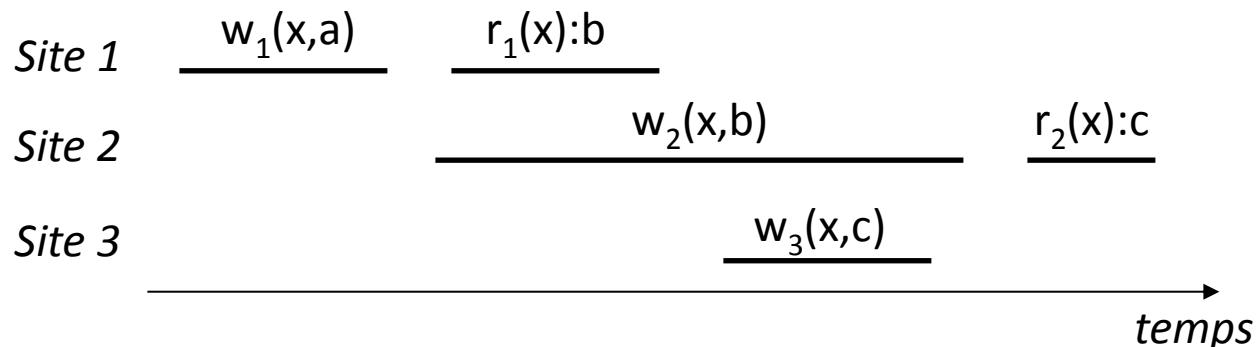
- Duplication et cohérence
  - Linéarisabilité
    - Afin d'étudier sous quelles conditions une exécution est acceptable, nous introduisons la définition de précédence temporelle  $<_t$  : soient deux opérations  $op_1$  et  $op_2$ ,  $op_1$  précède temporellement  $op_2$  si et seulement si la fin de l'opération  $op_1$  précède le début de l'opération  $op_2$ .
    - Deux opérations sont concurrentes si aucune d'entre elles ne précède temporellement l'autre : il y a concurrence dès qu'il y a un instant du temps où les deux opérations étaient simultanément en cours d'exécution
    - $<_t$  une relation d'ordre partiel

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Linéarisabilité

- Exemple 1 :



- $r_i(x):v$  une instance d'appel à  $\text{read}(x)$  qui a renvoyé la valeur  $v$ , et  $w_i(x,v)$  une instance d'appel à  $\text{write}(x,v)$
- $r_1(x):b <_t w_3(x,c)$  mais  $w_2(x,b)$  et  $w_3(x,c)$  sont concurrentes

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité
    - Une exécution est linéarisable si et seulement si il existe une relation d'ordre total sur les opérations notée  $<$  qui étend la précédence temporelle et qui vérifie :

Pour toute lecture  $r_i(x):v$ , il existe une écriture  $w_j(x,v) < r_i(x):v$ , et toute autre écriture  $w_k(x,v')$  avec  $v' \neq v$  vérifie :

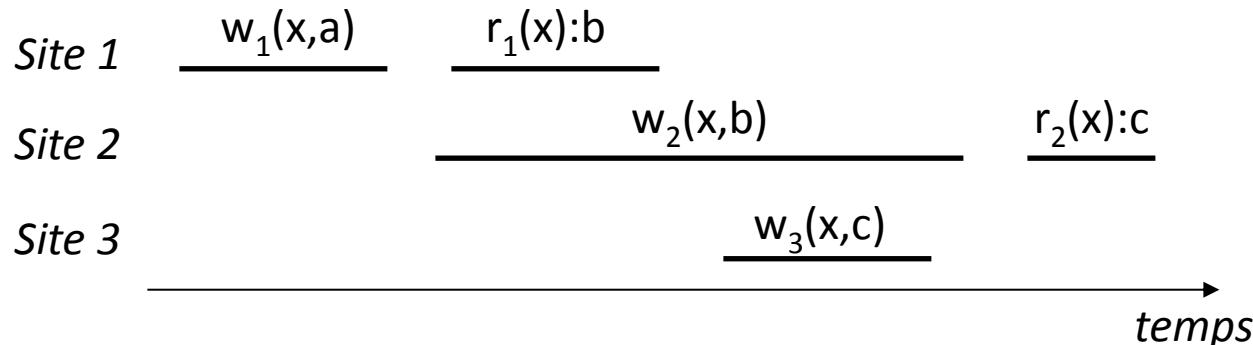
      - soit  $w_k(x,v') < w_j(x,v)$
      - soit  $r_i(x):v < w_k(x,v')$
    - Autrement dit, une exécution est linéarisable, si on peut ordonner l'ensemble des opérations en respectant la précédence temporelle et de telle sorte qu'une lecture renvoie la dernière valeur écrite selon cet ordre.

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Linéarisabilité

- Exemple 1 :



- L'exécution est linéarisable et équivalente à l'exécution séquentielle suivante :

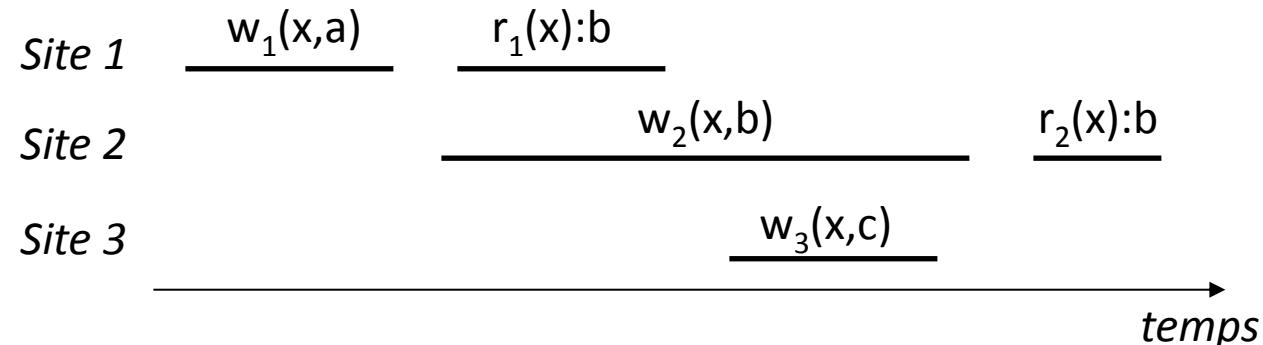
$w_1(x,a);w_2(x,b);r_1(x):b;w_3(x,c);r_2(x):c$

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Linéarisabilité

- Exemple 2 :



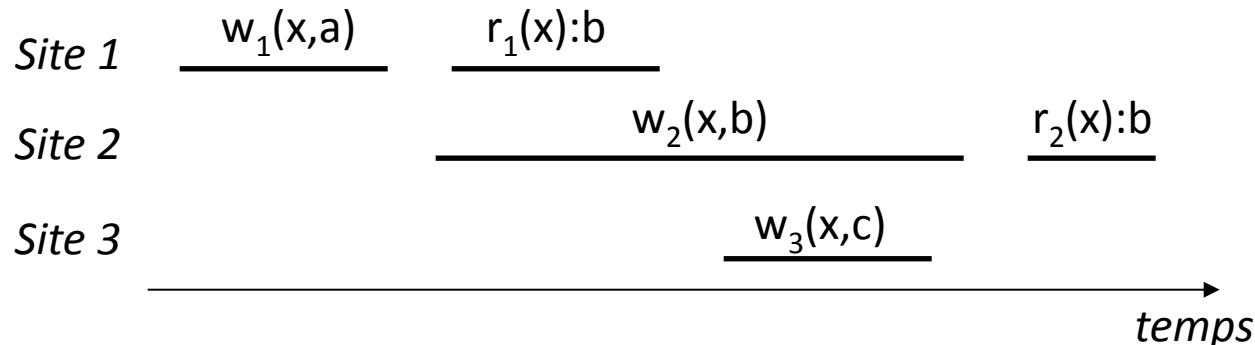
- Exécution linéarisable?

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Linéarisabilité

- Exemple 2 :



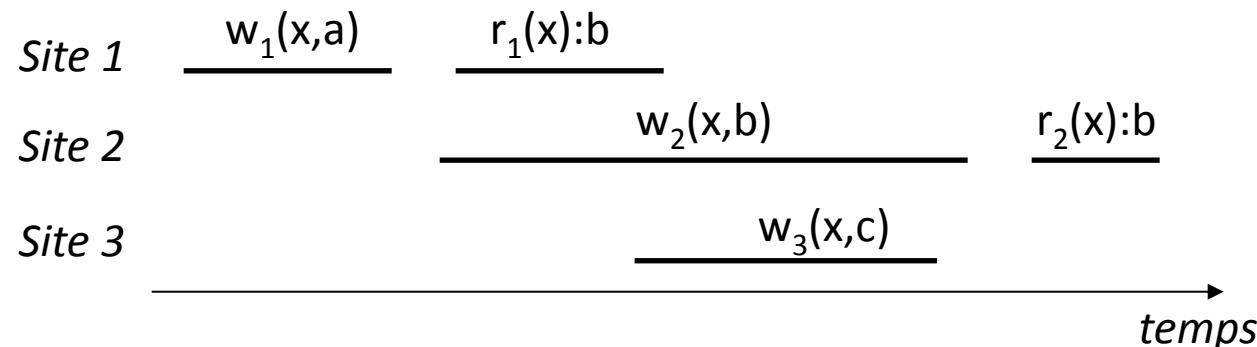
- Exécution linéarisable?
  - Non.  $w_2(x,b)$  peut être placé avant  $r_1(x):b$  et  $r_2(x):b$ , mais puisque  $r_1(x):b <_t w_3(x,c) <_t r_2(x):b$ , il n'y a pas de placement possible pour  $w_3(x,c)$ .

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Linéarisabilité

- Exemple 3 :



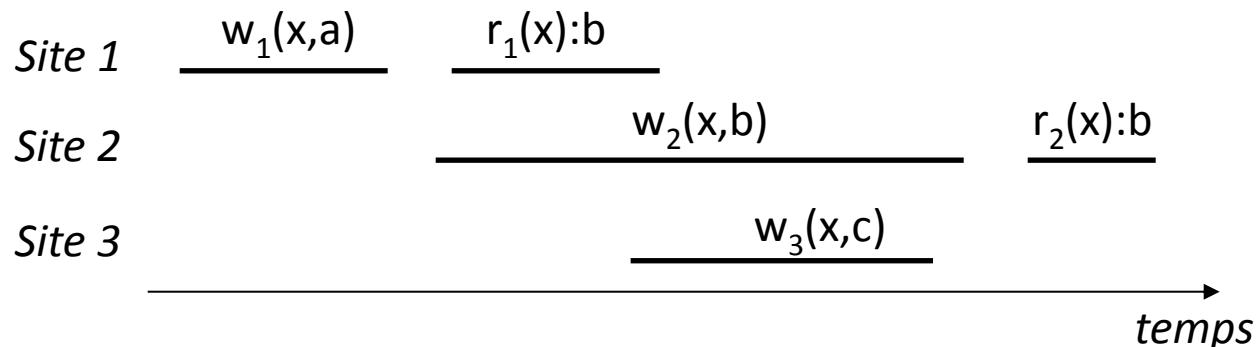
- Exécution linéarisable?

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Linéarisabilité

- Exemple 3 :



- Exécution linéarisable?
  - Oui. La deuxième exécution presque identique à la première n'implique pas la précédence temporelle  $r_1(x):b <_t w_3(x,c)$ . En conséquence elle est équivalente à l'exécution séquentielle suivante :

$w_1(x,a);w_3(x,c);w_2(x,b);r_1(x):b;r_2(x):b$

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité par exclusion mutuelle
    - Propriété de ce modèle : supposons que nous ayons un protocole qui garantit que, *relativement aux opérations sur chaque objet*, une linéarisation est possible, alors ce protocole garantit la linéarisation globale
    - Le premier algorithme garantit la linéarisabilité de manière triviale en assurant l'exclusion mutuelle entre les opérations sur un même objet
    - Ainsi la relation  $<_t$  est une relation d'ordre totale sur les opérations de chaque objet et la propriété précédente nous assure la linéarisabilité

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité par exclusion mutuelle
    - Nous organisons les sites en anneau. Un jeton par objet circule et contient son identité et sa valeur
    - Pour lire la valeur d'un objet, on attend le jeton et on renvoie la valeur après avoir envoyé le jeton au suivant sur l'anneau
    - Pour modifier un objet, on attend le jeton et on envoie le jeton au suivant sur l'anneau après avoir modifié sa valeur
    - A la réception d'un jeton, si l'objet associé n'était pas attendu, on envoie le jeton au suivant sur l'anneau sinon on indique que le jeton est arrivé et la valeur de cet objet

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité par exclusion mutuelle
    - Les variables d'un site  $S_i$ 
      - $\text{suivant}_i$  : constante contenant l'identité du site successeur de  $S_i$  sur l'anneau.
      - $\text{val}_i$  : variable contenant une valeur transmise par un jeton.
      - $\text{attendu}_i$  : variable contenant l'identité de l'objet dont le site attend le jeton. Cette variable est initialisée à NULL indiquant qu'on n'attend pas de jeton.

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité par exclusion mutuelle
    - Algorithme du site  $S_i$

**Read(x)**

Début

```
attendui = x;  
Attendre (attendui == NULL);  
Envoyer_à (suivanti, (x, vali));  
renvoyer(vali);
```

Fin

**Write(x,v)**

Début

```
attendui = x;  
Attendre (attendui == NULL);  
Envoyer_à (suivanti, (x, v));
```

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité par exclusion mutuelle
    - Algorithme du site  $S_i$

**Sur\_réception\_de ( j, (x, v))**

Début

Si ( attendu<sub>i</sub> != x ) Alors  
    Envoyer\_à ( suivant<sub>i</sub>, (x,v)) ;

Sinon

    val<sub>i</sub> = v;  
    attendu<sub>i</sub> = NULL;

Fsi

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité par lecture à la demande
    - Hypothèse : les objets ne sont modifiés que par un seul site appelé le propriétaire de l'objet
    - Solution simple pour assurer la linéarisabilité : lorsqu'un site appelle la primitive read() sur un objet distant, il envoie une demande de valeur au site propriétaire qui à la réception lui envoie la valeur demandée
    - A la lecture d'un objet, soit on est le propriétaire et on renvoie immédiatement la valeur, soit on envoie une demande au propriétaire et on renvoie la valeur fournie par le propriétaire
    - La modification d'un objet est purement locale

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité par lecture à la demande
    - A la réception d'une requête, on renvoie la valeur demandée
    - Les variables d'un site  $S_i$ 
      - $val_i$  : variable temporaire qui contient la valeur demandée lors d'un `read()` distant
      - $x_i$  : variable associée à l'objet partagé  $x$ . Un site  $S_i$  a autant de variables de ce type que d'objets dont il est propriétaire
      - $état_i$  : état du service à valeurs dans (repos, attente) initialisé à repos
      - $prop_i[objet]$  : constante qui contient l'identité du propriétaire de chaque objet

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité par lecture à la demande
    - Algorithme du site  $S_i$

**Read(x)**

Début

Si ( $prop_i[x]==i$ ) Alors  
    renvoyer( $x_i$ );  
Sinon  
    état<sub>i</sub>=attente;  
    Envoyer\_à( $prop_i[x]$ ,(req, x));  
    Attendre(état<sub>i</sub>==repos);  
    renvoyer( $val_i$ );  
Finsi

Fin

**Write(x,v)**

Début

$x_i = v;$

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité par lecture à la demande
    - Algorithme du site  $S_i$

**Sur\_réception\_de ( j, (req, x))**

Début

Envoyer\_à (j, (acq,  $x_i$ ));

Fin

**Sur\_réception\_de ( j, (acq, v))**

Début

$val_i = v;$

$état_i = repos;$

Fin

# Systèmes collaboratifs distribués

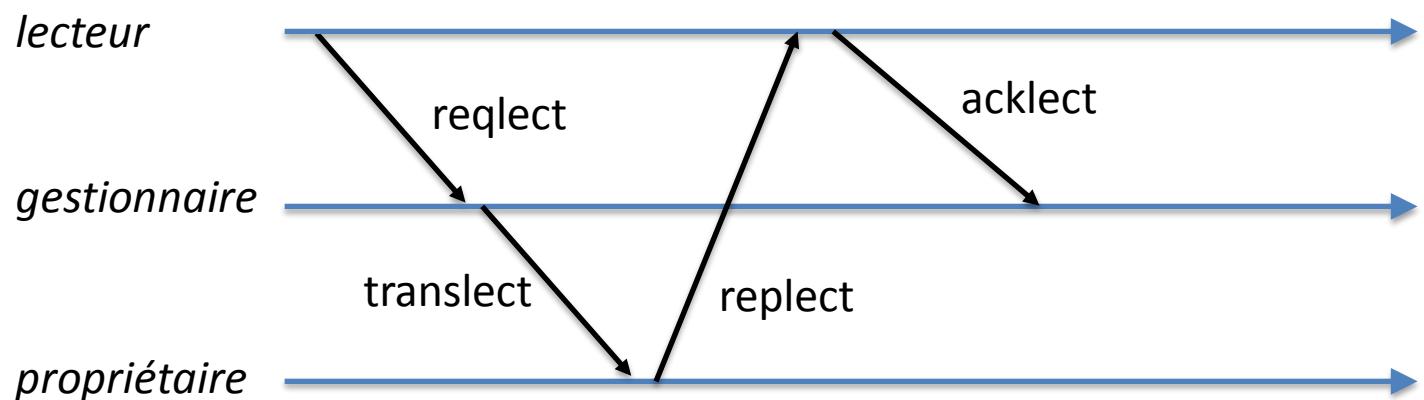
- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Les objets sont mobiles et éventuellement dupliqués
    - Chaque objet est associé un gestionnaire statique connu de tous. Chaque site dispose d'une table qui lui indique pour chaque objet son gestionnaire ( $gest_i$ )
    - Les informations qu'un gestionnaire possède sur un objet  $x$  sont contenues dans la cellule  $infos_i[x]$ 
      - L'identité du site propriétaire ( $infos_i[x].prop$ )
      - L'ensemble des autres sites qui ont une copie de l'objet ( $infos_i[x].ontcopie$ )

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Puisque des requêtes concurrentes peuvent parvenir au gestionnaire, celui-ci dispose d'une file qui contient les requêtes à traiter dont la requête en cours ( $\text{infos}_i[x].\text{à traiter}$ )
    - Chaque site dispose d'une mémoire virtuelle locale qui peut potentiellement contenir l'ensemble des objets du système ( $\text{mem}_i[x].\text{val}$ )
    - Chaque site doit connaître le statut relatif à chaque objet : objet invalide, accès en lecture seule ou accès en lecture-écriture ( $\text{mem}_i[x].\text{statut}$ )

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - La linéarisabilité est assurée par le recours au gestionnaire (il suffit que les accès à chaque objet soient linéarisables indépendamment des accès aux autres objets)
    - Le flux des messages engendrés par demande de lecture

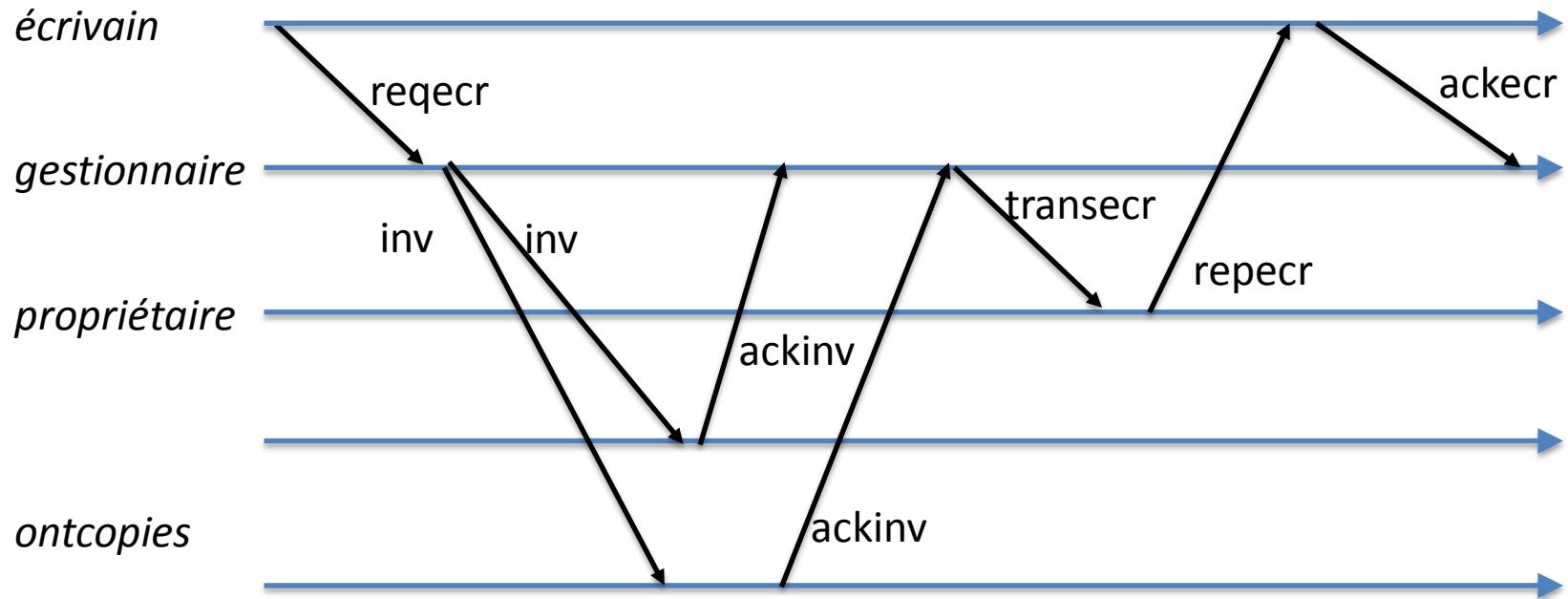


# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Un site qui désire obtenir le droit de lire envoie une requête de lecture (**reqlect**) au gestionnaire
    - Si celui-ci ne traite pas une autre requête sur cet objet, il transmet la requête au propriétaire qui modifie son accès à l'objet (lecture seule) et renvoie au lecteur l'objet (**relect**)
    - A la réception de l'objet, celui-ci indique au gestionnaire qu'il dispose de la copie (**acklect**)
    - Après mise à jour des informations, le gestionnaire peut éventuellement traiter une autre requête sur cet objet

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Le flux des messages engendrés par demande de écriture



# Systèmes collaboratifs distribués

- ❑ Duplication et cohérence
  - ❑ Linéarisabilité avec gestionnaires statiques
    - ❑ Un site qui désire obtenir le droit d'écrire envoie une requête d'écriture (`reqecr`) au gestionnaire
    - ❑ Si le gestionnaire ne traite pas une autre requête sur cet objet, il envoie des invalidations aux sites qui possèdent la copie (`inv`) et attend leur acquittement (`ackinv`). Puis, il transmet la requête au propriétaire qui invalide son accès à l'objet et renvoie à l'écrivain le droit d'accès (`repecr`)
    - ❑ A la réception de celui-ci, le demandeur indique au gestionnaire qu'il dispose du droit d'écrire (`ackecr`)
    - ❑ Après mise à jour des informations, le gestionnaire peut traiter une autre requête sur cet objet

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Les variables d'un site  $S_i$ 
      - $gest_i[Objets]$  : constante qui contient l'identité du gérant de chaque objet
      - $Objets_i$  : ensemble constant des objets gérés par  $S_i$
      - $infos_i[Objets_i]$  de
        - $prop$  : propriétaire courant de l'objet, initialisé à i
        - $ontcopie$  : ensembles des autres sites qui ont une copie de l'objet, initialement vide
        - $\grave{a}trajiter$  : file de couples  $<site,requête>$  initialement vide (primitives associées:  $enfiler()$ , $défiler()$ , $t\grave{e}te()$ )

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Les variables d'un site  $S_i$ 
      - memi[Objets] de
        - val : valeur de l'objet initialisé à la valeur initiale de l'objet s'il est affecté à  $S_i$
        - statut : statut de l'objet dans l'ensemble ([invalidé](#), [lect](#), [ecr](#)) , ecr pour un objet affecté, invalide pour un objet non affecté.
      - temp<sub>i</sub> : variable temporaire d'identité de site
      - req<sub>i</sub> : variable temporaire de type de requête

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture  
En cas de lecture d'un objet invalide, on émet une requête au gestionnaire de l'objet et on attend que l'objet soit disponible en lecture

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture  
En cas de lecture d'un objet invalide, on émet une requête au gestionnaire de l'objet et on attend que l'objet soit disponible en lecture

**Read(x)**

Début

Si ( $mem_i[x].statut == invalide$ ) Alors  
envoyer\_à( $gest_i[x]$ ,(reqlect,x));  
Attendre( $mem_i[x].statut == lect$ );

Finsi

renvoyer( $mem_i[x].val$ );

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture

A la réception d'une requête de lecture, si une requête n'est pas en cours on transfère la requête au propriétaire courant de l'objet. Dans tous les cas, on met à jour la file des requêtes à traiter

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture

A la réception d'une requête de lecture, si une requête n'est pas en cours on transfère la requête au propriétaire courant de l'objet. Dans tous les cas, on met à jour la file des requêtes à traiter

**Sur\_réception\_de ( j, (reqlect, x))**

Début

Si ( $\text{infos}_i[x].\text{àtraiter} == \text{vide}$ ) Alors

Envoyer \_à( $\text{infos}_i[x].\text{prop}, (\text{translect}, j, x)$ );

Finsi

enfiler( $\text{infos}_i[x].\text{àtraiter}$ , <j, reqlect>);

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture

A la réception d'un transfert d'une requête de lecture, le propriétaire de l'objet envoie une réponse au demandeur et met à jour le statut en lecture

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture  
A la réception d'un transfert d'une requête de lecture, le propriétaire de l'objet envoie une réponse au demandeur et met à jour le statut en lecture

**Sur\_réception\_de ( j, (translect, k, x))**

Début

```
memi[x].statut=lect ;  
envoyer_à(k, (relect, x, memi[x].val));
```

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture  
A la réception de la réponse de lecture, la valeur et le statut de cet objet sont mis à jour et un acquittement est envoyé au gestionnaire

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture  
A la réception de la réponse de lecture, la valeur et le statut de cet objet sont mis à jour et un acquittement est envoyé au gestionnaire

**Sur\_réception\_de ( j, (relect, x, v))**

Début

```
memi[x].val=v ;  
memi[x].statut=lect ;  
envoyer_à(gesti[x], (acklect, x));
```

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture  
A la réception de l'acquittement de lecture le nouveau lecteur est ajouté aux possesseurs d'une copie. Puis après avoir défilé la requête courante on traite la requête suivante s'il y en a une à l'aide de la primitive finrequete

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture  
A la réception de l'acquittement de lecture le nouveau lecteur est ajouté aux possesseurs d'une copie. Puis après avoir défilé la requête courante on traite la requête suivante s'il y en a une à l'aide de la primitive finrequete

**Sur\_réception\_de ( j, (acklect, k, x))**

Début

$\text{infos}_i[x].\text{ontcopie} = \text{infos}_i[x].\text{ontcopie} \cup \{j\};$

    Finrequete(x);

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande de lecture

**Finrequete(x)**

Début

défiler( $\text{infos}_i[x].\text{à traiter}$ );

Si ( $\text{infos}_i[x].\text{à traiter} \neq \text{vide}$ ) Alors

$\langle \text{temp}_i, \text{req}_i \rangle = \text{tête}(\text{infos}_i[x].\text{à traiter})$ ;

Si ( $\text{req}_i = \text{reqlect}$ ) Alors

envoyer \_à( $\text{infos}_i[x].\text{prop}$ , (translect,  $\text{temp}_i$ ,  $x$ ));

Sinon

**Traiterécriture**( $\text{temp}_i$ ,  $x$ );

Finsi

Finsi

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture

Pour modifier un objet dont on ne possède pas le droit en écriture, on envoie une requête d'écriture et on attend le droit d'écriture. Finalement on écrit la valeur prévue.

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture

Pour modifier un objet dont on ne possède pas le droit en écriture, on envoie une requête d'écriture et on attend le droit d'écriture. Finalement on écrit la valeur prévue.

**Write (x, v)**

Début

Si ( $mem_i[x].statut \neq ecr$ ) Alors  
    envoyer\_à( $gest_i[x]$ , (reqecr, x));  
    Attendre( $mem_i[x].statut == ecr$ );

Finsi

$mem_i[x].val = v$ ;

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception d'une requête d'écriture, si une requête n'est pas en cours on appelle la primitive `traiterécriture()`. Dans tous les cas, on met à jour la file des requêtes à traiter.

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception d'une requête d'écriture, si une requête n'est pas en cours on appelle la primitive traiterécriture().  
Dans tous les cas, on met à jour la file des requêtes à traiter.

**Sur\_réception\_de ( j, (reqecr, x))**

Début

Si ( $\text{infos}_i[x].\text{à traiter} == \text{vide}$ ) Alors

    Traiterecriture(j,x);

Finsi

    enfiler( $\text{infos}_i[x].\text{à traiter}$ , <j, reqecr>);

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture

Le traitement d'une écriture consiste d'abord à envoyer à un message d'invalidation à tous les possesseurs d'une copie excepté le demandeur. S'il n'y en a pas, on transfère directement la requête d'écriture au propriétaire

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture

**Traiterecriture( j, x)**

Début

Si ( $\text{infos}_i[x].\text{ontcopie} \setminus \{j\} \neq \emptyset$ ) Alors

Pour  $\text{temp}_i \in \text{infos}_i[x].\text{ontcopie} \setminus \{j\}$  faire

envoyer\_à( $\text{temp}_i$ , (inv, x));

Finpour

Sinon

envoyer\_à( $\text{infos}_i[x].\text{prop}$ , (transecr, j, x));

Finsi

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception d'une invalidation, l'objet est invalidé et un acquittement est envoyé au gestionnaire

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception d'une invalidation, l'objet est invalidé et un acquittement est envoyé au gestionnaire

**Sur\_réception\_de ( j, (inv, x))**

Début

```
memi[x].statut=invalidé ;  
envoyer_à(j, (ackinv, x));
```

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception d'un acquittement d'invalidation l'émetteur est extrait des possesseurs de copies. S'il n'y a plus de possesseurs autre qu'éventuellement le demandeur, la requête est transférée au propriétaire

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception d'un acquittement d'invalidation l'émetteur est extrait des possesseurs de copies. S'il n'y a plus de possesseurs autre qu'éventuellement le demandeur, la requête est transférée au propriétaire

**Sur\_réception\_de ( j, (ackinv, x))**

Début

```
<tempi,reqj> = tête(infosi[x].àtriter);  
infosi [x].ontcopie = infosi [x].ontcopie \ {j};  
Si (infosi[x].ontcopie \{tempi} == Ø ) Alors  
    envoyer _à(infosi[x].prop, (transecr, tempi, x));
```

Finsi

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception d'un transfert d'une requête d'écriture, le propriétaire de l'objet envoie une réponse au demandeur et met à jour le statut à invalide

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception d'un transfert d'une requête d'écriture, le propriétaire de l'objet envoie une réponse au demandeur et met à jour le statut à invalide

**Sur\_réception\_de ( j, (transecr, k, x))**

Début

```
memi[x].statut=invalide ;  
envoyer_à(k, (repecr, x));
```

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception de la réponse d'écriture, le statut de cet objet est mis à jour et un acquittement est envoyé au gestionnaire

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception de la réponse d'écriture, le statut de cet objet est mis à jour et un acquittement est envoyé au gestionnaire

```
Sur_réception_de ( j, (repecr, x))
```

Début

```
    memi[x].statut=ecr;  
    envoyer_à(gesti[x], (ackecr, x));
```

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception de l'acquittement d'écriture, le demandeur est le nouveau propriétaire de l'objet et personne n'en possède une copie. Puis après avoir défilé la requête courante, on traite la requête suivante s'il y en a une.

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Linéarisabilité avec gestionnaires statiques
    - Algorithme du site  $S_i$ : demande d'écriture  
A la réception de l'acquittement d'écriture, le demandeur est le nouveau propriétaire de l'objet et personne n'en possède une copie. Puis après avoir défilé la requête courante, on traite la requête suivante s'il y en a une.

**Sur\_réception\_de ( j, (ackecr, x))**

Début

```
infosi[x].ontcopie = Ø ;  
infosi[x].prop = j;  
Finrequete(x);
```

Fin

# Systèmes collaboratifs distribués

- ❑ Duplication et cohérence
  - ❑ Cohérence séquentielle
    - ❑ Lamport 1979 dans le contexte de mémoire partagée
    - ❑ Le résultat de l'exécution d'un ensemble de processus clients est identique à celui d'une exécution dans laquelle :
      - ❑ Toutes les opérations sur les données sont exécutées selon une certaine séquence S
      - ❑ Les opérations exécutées par tout processus p figurent dans S **dans le même ordre que** dans p
    - ❑ Pas de contraintes sur l'ordre relatif des opérations dans des processus différents (tous les processus voient le même séquencement des opérations)

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle
    - Nous restreignons d'abord la précédence temporelle aux opérations exécutées sur un même site
    - Définition de la précédence locale  $<_l$  : soient deux opérations  $op_1$  et  $op_2$ ,  $op_1$  précède localement  $op_2$  si et seulement les deux opérations sont exécutés sur le même site et  $op_1 <_t op_2$

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle
    - Une exécution est séquentiellement consistante si et seulement si il existe une relation d'ordre total sur les opérations notée  $<$  qui étend la précédence locale et qui vérifie :

Pour toute lecture  $r_i(x):v$ , il existe une écriture  $w_j(x,v) < r_i(x):v$  tel que  $\text{NOT } r_i(x):v <_t w_j(x,v)$  , et toute autre écriture  $w_k(x,v')$  avec  $v' \neq v$  vérifie :

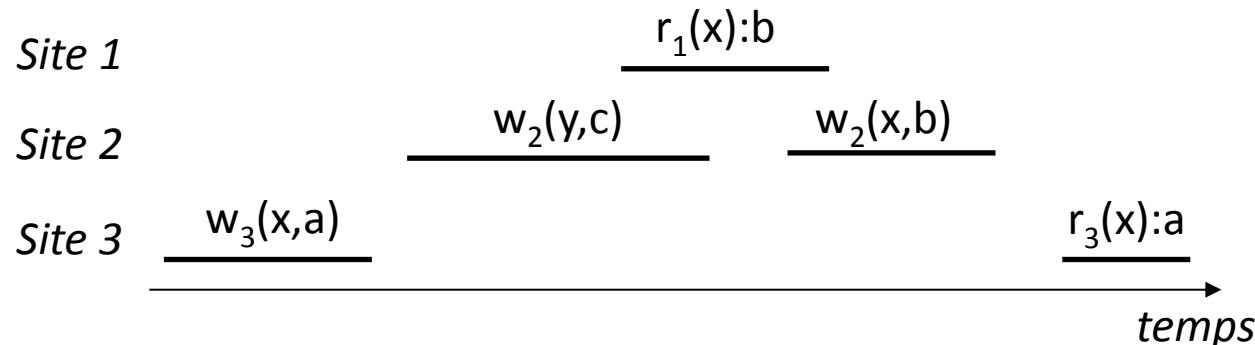
      - soit  $w_k(x,v') < w_j(x,v)$
      - soit  $r_i(x):v < w_k(x,v')$
    - Une exécution est séquentiellement consistante, si on peut ordonner l'ensemble des opérations en respectant la précédence locale et de telle sorte qu'une lecture renvoie la dernière valeur écrite selon cet ordre. De plus, cette écriture ne peut suivre temporellement la lecture.

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Cohérence séquentielle

- Exemple 4 :



- L'exécution ne peut être linéarisée car :

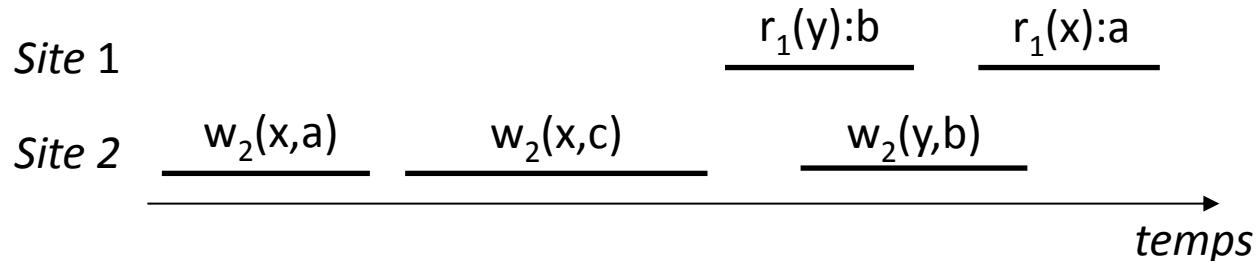
$$w_3(x,a) <_t w_2(x,b) <_t r_3(x):a$$

- L'exécution est séquentiellement consistante car on peut ordonner les opérations en préservant la précédence locale de la manière suivante :

$$w_3(x,a); r_3(x):a; w_2(y,c); w_2(x,b); r_1(x):b$$

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle
    - Exemple 5 :



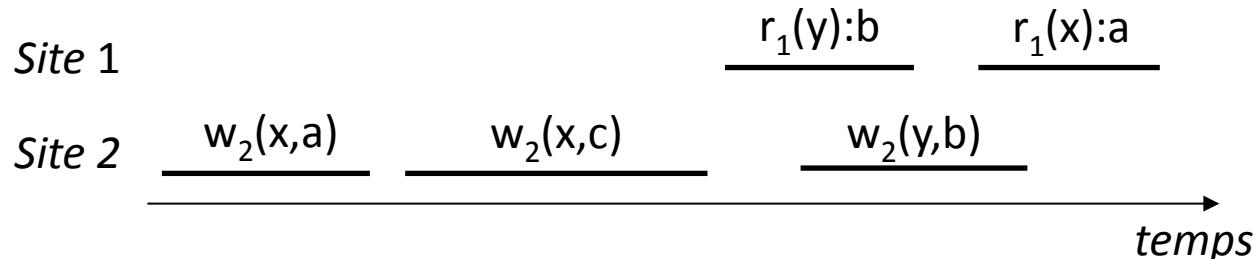
- Exécution linéarisable?
- Exécution séquentiellement consistante?

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Cohérence séquentielle

- Exemple 5 :



- Exécution linéarisable? Non, car

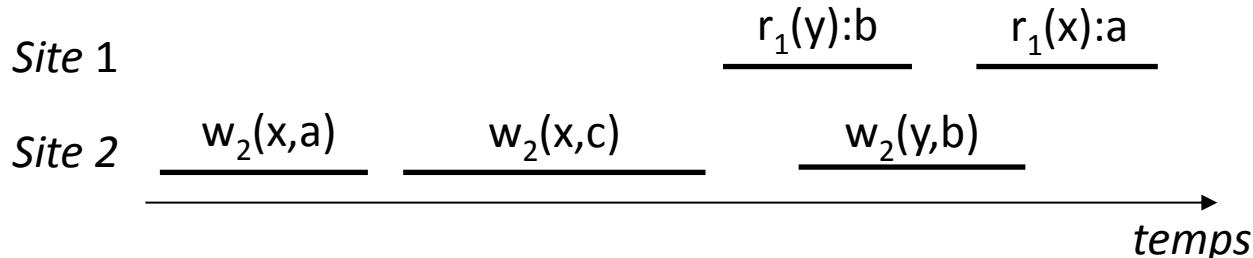
$$w_2(x,a) <_t w_2(x,c) <_t r_1(x):a$$

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Cohérence séquentielle

- Exemple 5 :



- Exécution séquentiellement consistante? Non, car une exécution séquentielle équivalente devrait vérifier:

$w_2(x,c) < w_2(y,b)$  (précédence locale)

$w_2(y,b) < r_1(y):b$  (consistance des lectures)

$r_1(y):b < r_1(x):a$  (précédence locale)

D'où par transitivité :  $w_2(x,c) < r_1(x):a$

Mais en raison de la consistance des lectures on doit avoir:  
 $r_1(x):a < w_2(x,c)$ . Ce qui est contradictoire.

# Systèmes collaboratifs distribués

- ❑ Duplication et cohérence
  - ❑ Cohérence séquentielle par propagation des écritures
    - ❑ Nous supposons à nouveau que les objets ne sont modifiés que par leur propriétaire
    - ❑ Nous supposons de plus que chaque site commence son exécution par une écriture (*au besoin la couche service peut ajouter une écriture sans effet au premier appel à read() si aucune écriture n'a été effectuée*)
    - ❑ Objectif de l'algorithme : anticiper les lectures distantes par diffusion de chaque modification d'un objet partagé par le propriétaire

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle par propagation des écritures
    - Chaque site gère une horloge logique pour dater les écritures sur les objets partagés
    - Conformément au fonctionnement de ces horloges, celles-ci s'incrémentent à chaque nouvelle écriture et leur gestion garantit qu'une écriture distante dont un site est au courant a une heure plus petite que les futures écritures de ce site

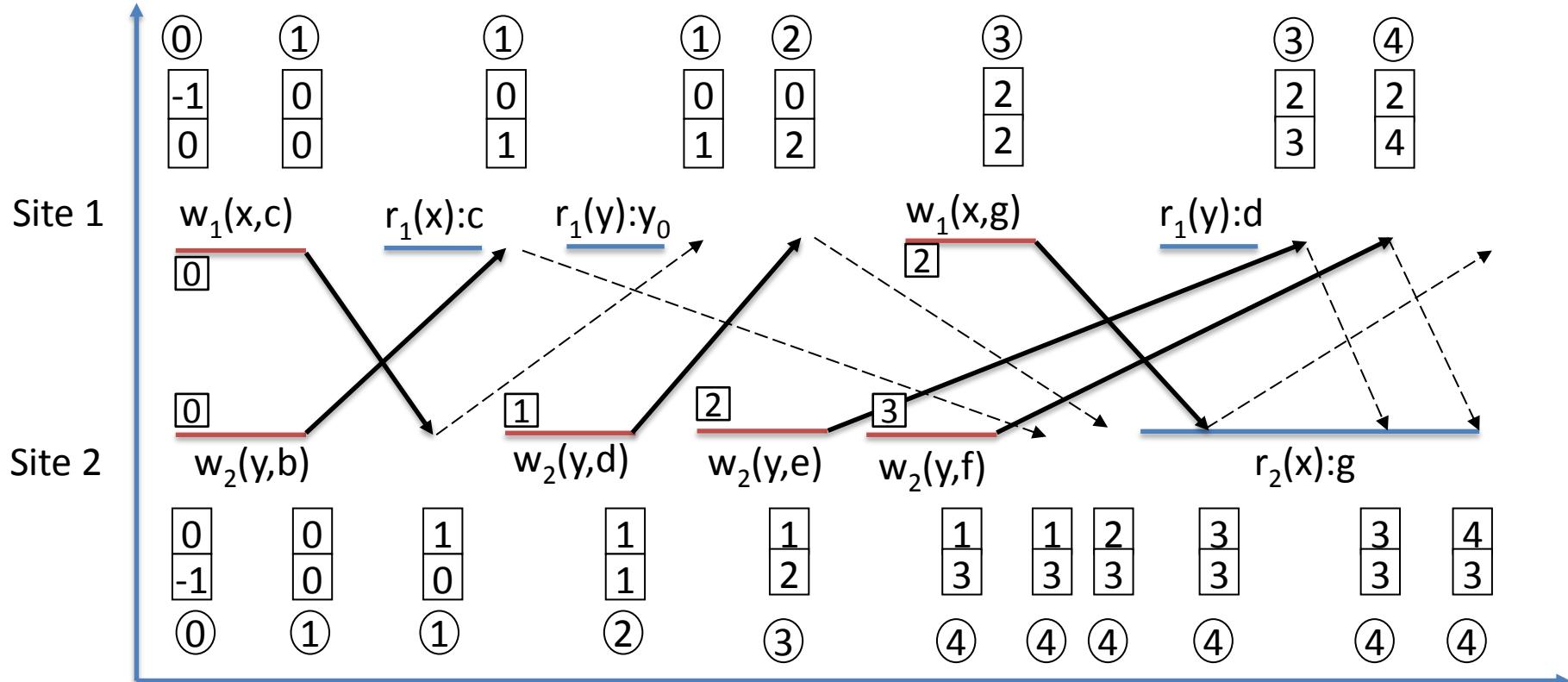
# Systèmes collaboratifs distribués

- Duplication et cohérence
  - **Cohérence séquentielle par propagation des écritures**
    - La séquentialisation repose sur un ordre total entre les écritures obtenu en comparant lexicographiquement les doublons (heure, identité du site) associés aux écritures
    - Les lectures sur un site qui s'intercalent entre deux écritures locales sont "placées" immédiatement après l'occurrence de la première écriture

# Systèmes collaboratifs distribués

## □ Duplication et cohérence

### □ Cohérence séquentielle par propagation des écritures

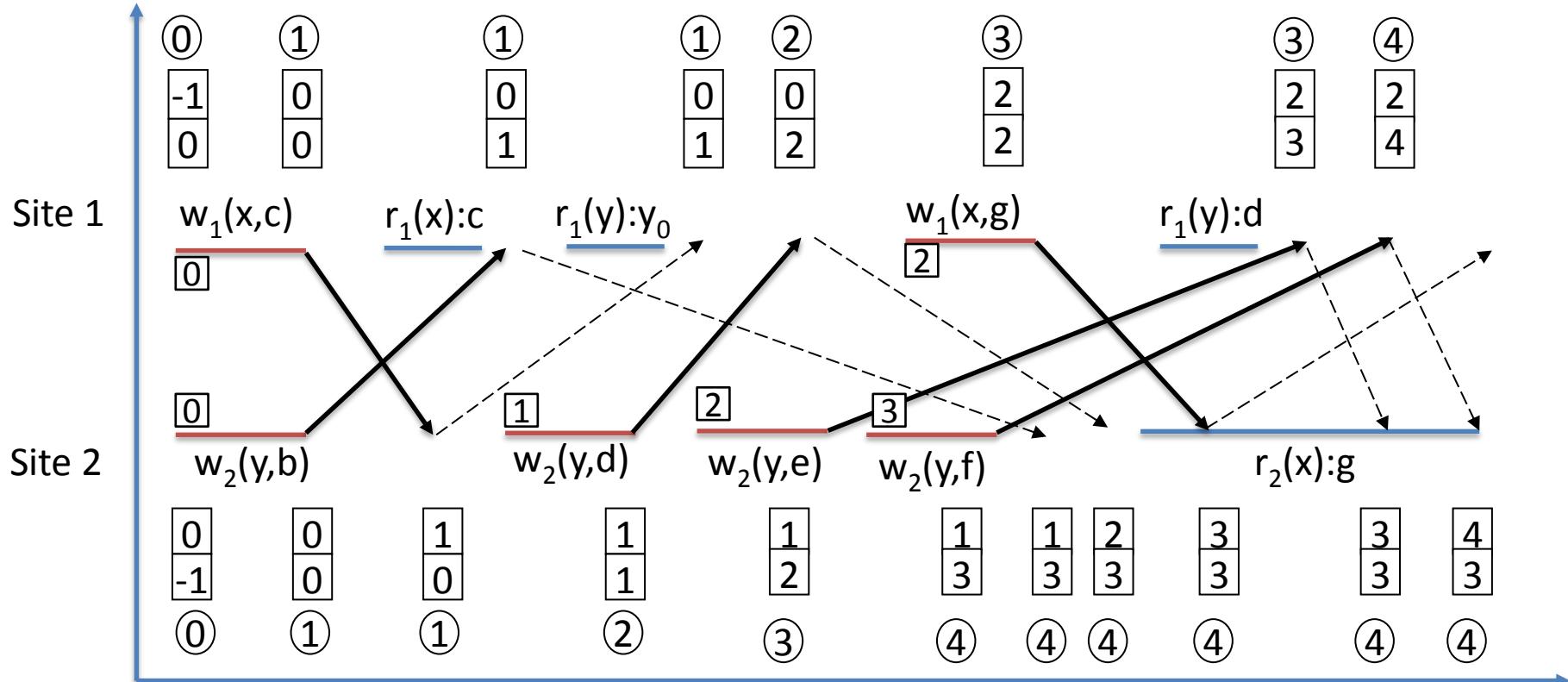


- L'ordre des écritures est le suivant :
- $w_1(x, c); w_2(y, b); w_2(y, d); w_1(x, g); w_2(y, e); w_2(y, f)$

# Systèmes collaboratifs distribués

## □ Duplication et cohérence

### □ Cohérence séquentielle par propagation des écritures



□ En intercalant les lectures, on obtient :  $w_1(x,c); r_1(x):c; r_1(y):y_0 ; w_2(y,b); w_2(y,d); w_1(x,g); r_1(y):d; w_2(y,e); w_2(y,f); r_2(x):g$

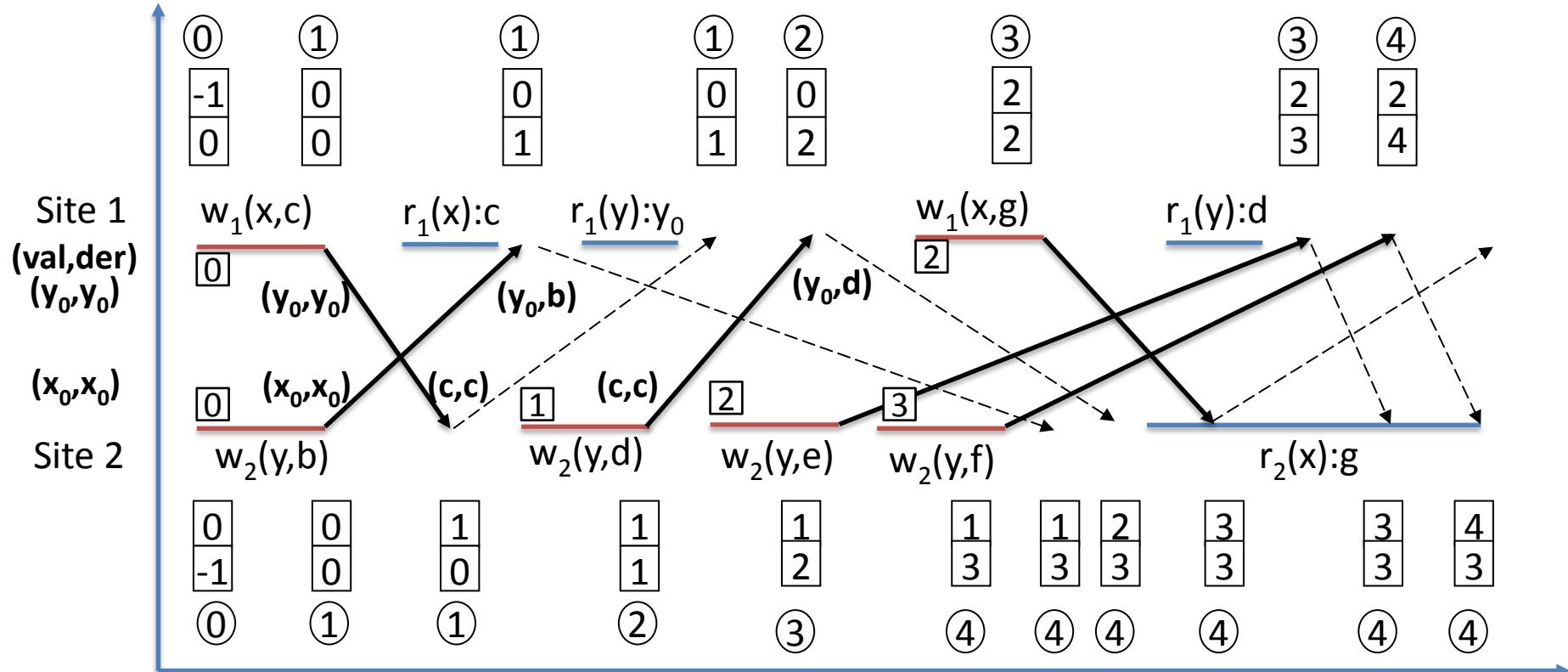
# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle par propagation des écritures
    - Il convient donc d'assurer la consistante des lectures
      - Une lecture ne renvoie la valeur d'un objet distant que si elle sait que les écritures sur le site distant "suivront" la dernière écriture locale
    - La gestion des valeurs reçues lors de la propagation des écritures. A la réception d'une valeur, à l'aide des horloges logiques on sait si la valeur peut être renvoyée comme résultat. A cet effet, deux variables sont utilisées par objet distant: l'une contient la dernière valeur reçue et l'autre la dernière valeur reçue qui peut être renvoyé comme résultat d'une lecture

# Systèmes collaboratifs distribués

## □ Duplication et cohérence

### □ Cohérence séquentielle par propagation des écritures

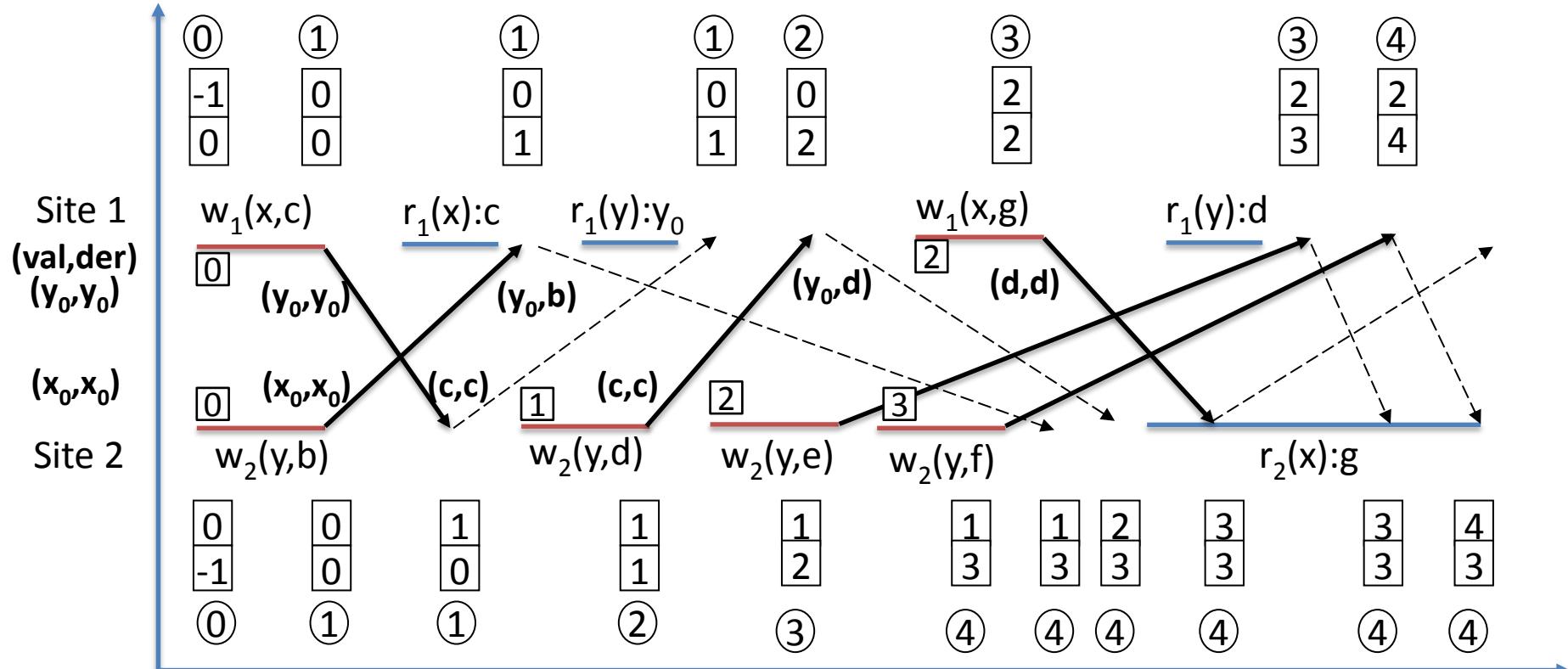


- $w_2(y, b)$  puis  $w_2(y, d)$  ne sont pas considérées comme des valeurs acceptables à leur réception sur le site 1 car leur heure est trop grande mais elles sont stockées successivement comme dernière valeur reçue

# Systèmes collaboratifs distribués

## □ Duplication et cohérence

### □ Cohérence séquentielle par propagation des écritures

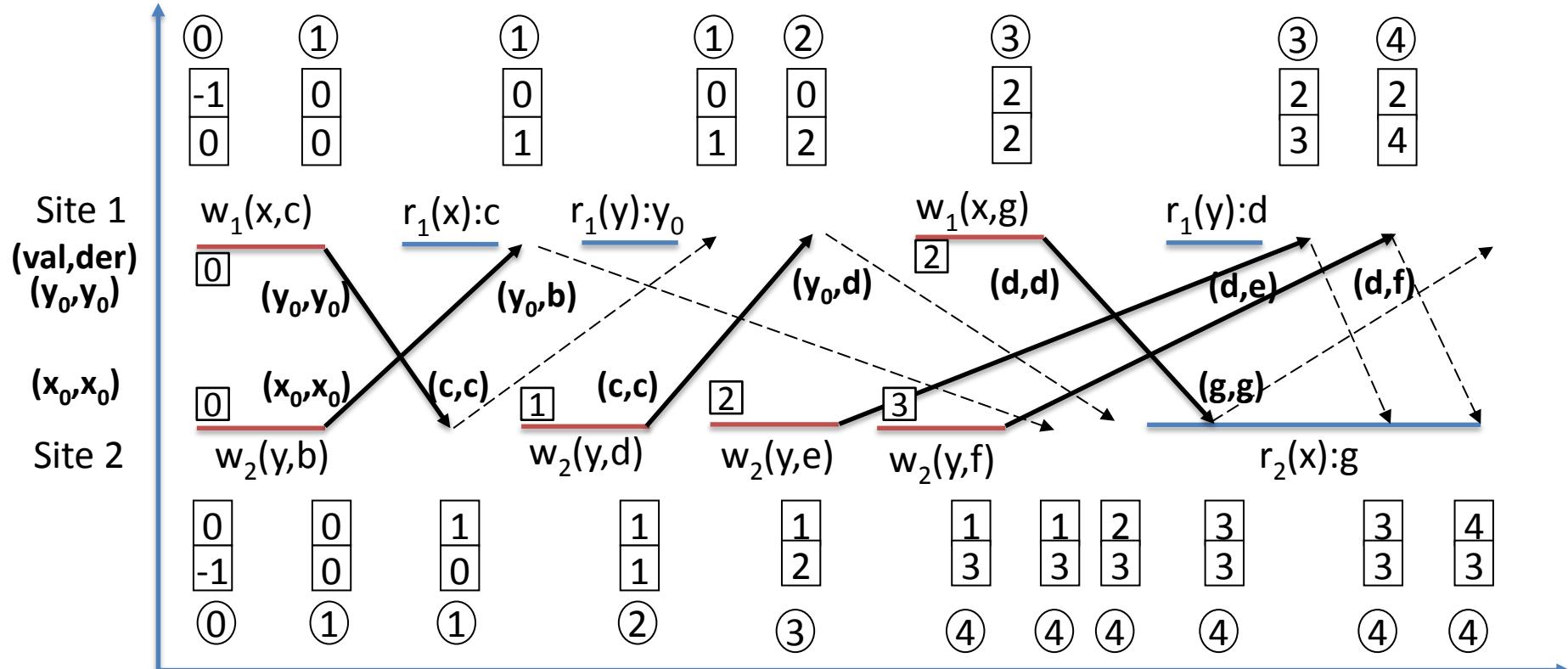


- A chaque nouvelle écriture les dernières valeurs reçues deviennent acceptables:  $w_2(y,d)$  devient une valeur acceptable après l'écriture  $w_1(x,g)$

# Systèmes collaboratifs distribués

## Duplication et cohérence

### Cohérence séquentielle par propagation des écritures



- $r_2(x)$  ne renvoie  $g$  qu'à la réception de l'acquittement de la modification  $w_2(y, f)$  car le site 2 sait à cet instant que la prochaine écriture sur le site 1 sera datée d'au moins  $(4, 1)$

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle par propagation des écritures
    - A la lecture d'un objet (read), soit on est le propriétaire et on renvoie immédiatement la valeur, soit on attend d'être sûr que les prochaines écritures du propriétaire de l'objet "suivent" la dernière écriture sur i
    - A la modification d'un objet (write), on met aussi à jour les variables dont on n'est pas propriétaire au vu des dernières valeurs reçues. On enregistre l'heure de l'écriture, on informe les autres sites de la modification et on met à jour son horloge logique

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle par propagation des écritures
    - A la réception d'une modification, on mémorise la valeur et on vérifie qu'elle peut être renvoyée en cas de lecture. Puis on met à jour le tableau des heures d'écritures
    - L'acquittement sert uniquement à mettre à jour le tableau des heures des opérations.

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle par propagation des écritures
    - Les variables d'un site  $S_i$ 
      - $\text{val}_i[\text{Objets}]$  : tableau de valeurs indicé par les objets. Ces valeurs serviront comme résultat d'un appel à `read()`. Chaque cellule est initialisée à une valeur par défaut commune à tous les sites mais qui est spécifique à chaque objet.
      - $\text{der}_i[\text{Objets}]$  : tableau de valeurs indicé par les objets. Ces valeurs sont les dernières valeurs connues de l'objet correspondant. Ce tableau est initialisé de la même façon que le tableau précédent. Il ne sert en fait qu'aux objets "distants".
      - $h_i$  : heure locale de  $i$  initialisée à 0.

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle par propagation des écritures
    - Les variables d'un site  $S_i$ 
      - $hor_i[1..N]$  : tableau d'heures indicé par les sites, initialisé à  $hor_i[j]=0$  pour  $j \neq i$  et  $hor_i[i]=-1$ . Dans le cas d'un site différent de  $i$  la cellule fournit un minorant de l'heure de la prochaine écriture. Dans le cas de  $i$ , la cellule fournit l'heure de la dernière écriture.
      - $prop_i[Objets]$  : constante qui contient l'identité du propriétaire de chaque objet.

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle par propagation des écritures
    - Algorithme du site  $S_i$

**Read(x)**

Début

Si ( $prop_i[x] \neq i$ ) Alors

Attendre(  $(hor_i[prop_i[x]], prop_i[x]) > (hor_i[i], i)$  );

Finsi

renvoyer( $val_i[x]$ );

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle par propagation des écritures
    - Algorithme du site  $S_i$

**Write(x,v)**

Début

$\text{val}_i[x] = v;$

Pour tout  $y$  tel que  $\text{prop}_i[y] \neq i$  faire

$\text{val}_i[y] = \text{der}_i[y];$

Fin pour

$\text{hor}_i[i] = h_i;$

$\text{diffuser}(\text{maj}, h_i, x, v);$

$h_i++;$

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence séquentielle par propagation des écritures
    - Algorithme du site  $S_i$

**Sur\_réception\_de ( j, (maj, h, x, v))**

Début

$der_i[x] = v;$

Si  $(h,j) < (hor_i[i],i)$  ) Alors

$val_i[x] = v;$

Finsi

$hor_i[j] = h+1;$

$h_i = \max(h_i, h+1);$

$envoyer\_à(j, (acq, h_i));$

Fin

**Sur\_réception\_de ( j, (acq, h))**

Début

$hor_i[j] = h;$

Fin

# Systèmes collaboratifs distribués

- Duplication et cohérence
  - Cohérence Causale
    - Modèle plus faible que la cohérence séquentielle, car on ne considère que des événements reliés par une relation de causalité
    - Si deux opérations  $op_1$  et  $op_2$  sont tels que  $op_1$  précède causalement  $op_2$  ( $op_1 <_c op_2$ ), alors tout processus doit « voir »  $op_1$  avant  $op_2$
    - Soient deux opérations  $op_1$  et  $op_2$ .  $op_1 <_c op_2$  si et seulement si
      - $op_1:w(x)$  et  $op_2:r(x)$ , ou
      - $op_1:r(x)$  et  $op_2: w(y)$  (car la valeur écrite peut dépendre d'un calcul fait à partir de la valeur lue)

# Systèmes collaboratifs distribués

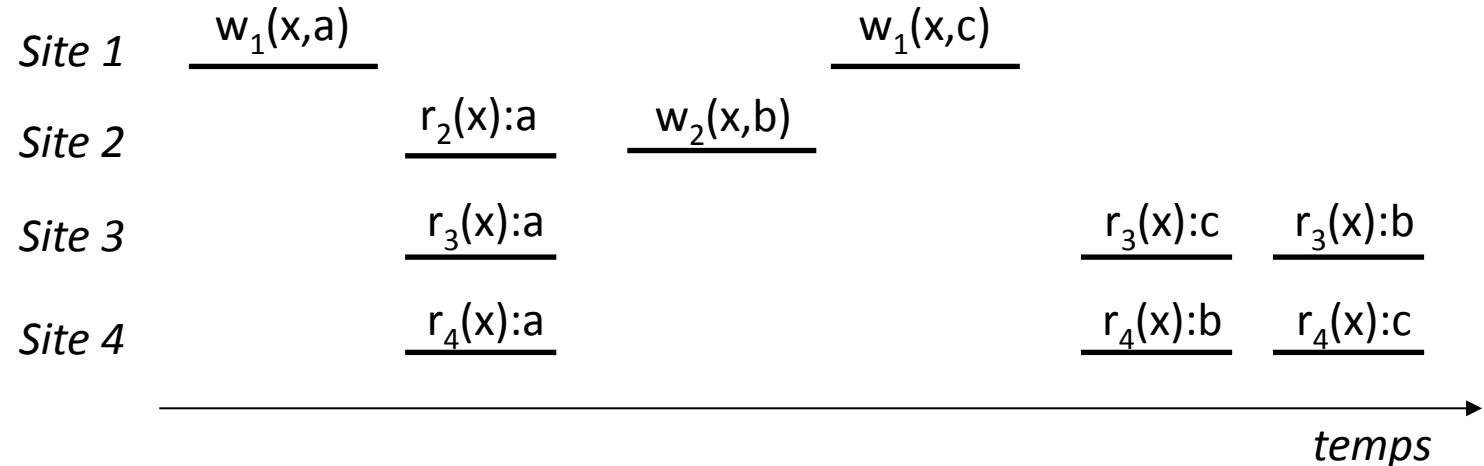
- Duplication et cohérence
  - **Cohérence Causale**
    - Deux opérations sont concurrentes si  $op1:w(x)$  et  $op2:w(y)$  (de manière indépendante)
    - La condition sur écriture et lecture est déjà incluse dans la définition de précédence
    - Reste à spécifier la condition sur les écritures : des écritures causalement liées doivent être vues par tous les processus dans leur ordre causal. Les écritures causalement indépendantes peuvent être vues dans un ordre différent

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Cohérence Causale

- Exemple 6 :



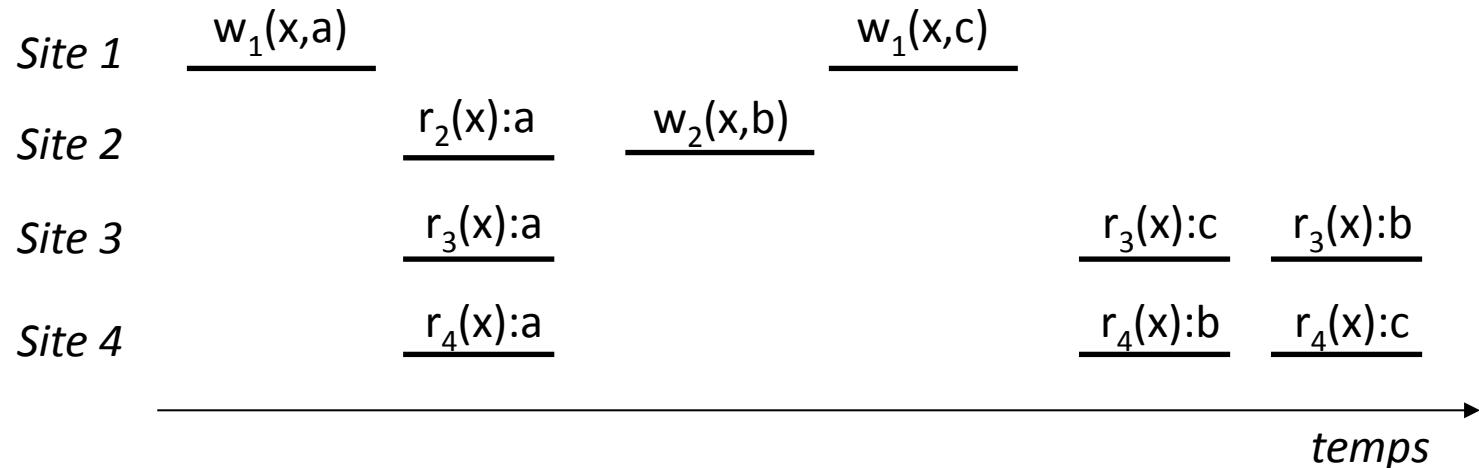
- Exécution séquentiellement consistante?
    - Exécution causalement consistante?

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Cohérence Causale

- Exemple 6 :



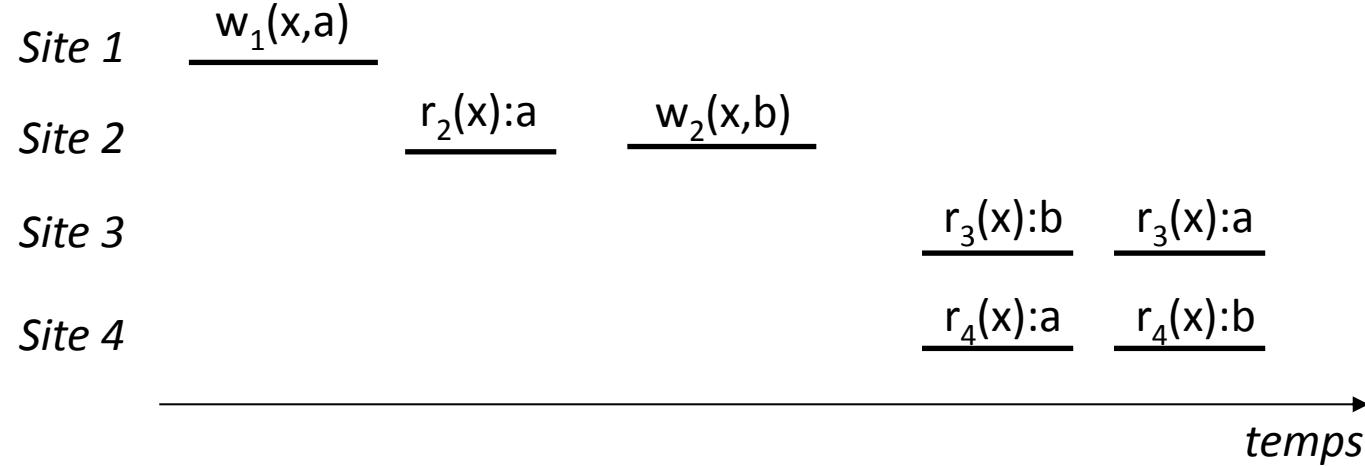
- Exécution séquentiellement consistante? Non
    - Exécution causalement consistante? Oui, les lectures des sites 3 et 4 renvoient b et c dans des ordres différents. C'est possible car les écritures sont causalement indépendantes

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Cohérence Causale

- Exemple 7 :



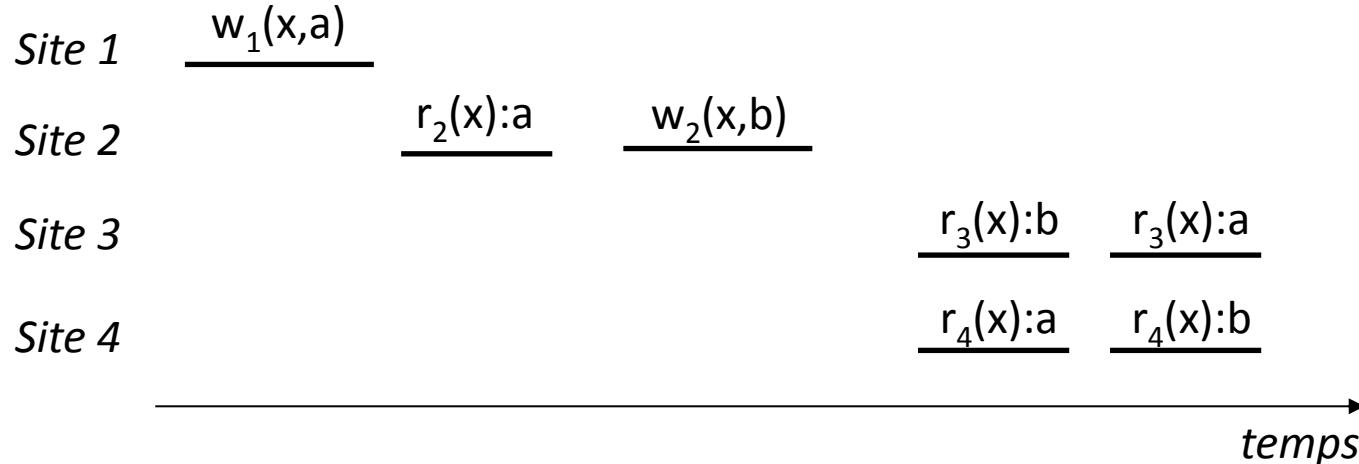
- Exécution causalement consistante?

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Cohérence Causale

- Exemple 7 :



- Exécution causalement consistante? Non, car précédence causale entre  $w_1(x,a)$  et  $w_2(x,b)$  alors que les lectures des sites 3 et 4 renvoient a et b dans des ordres différents.

# Systèmes collaboratifs distribués

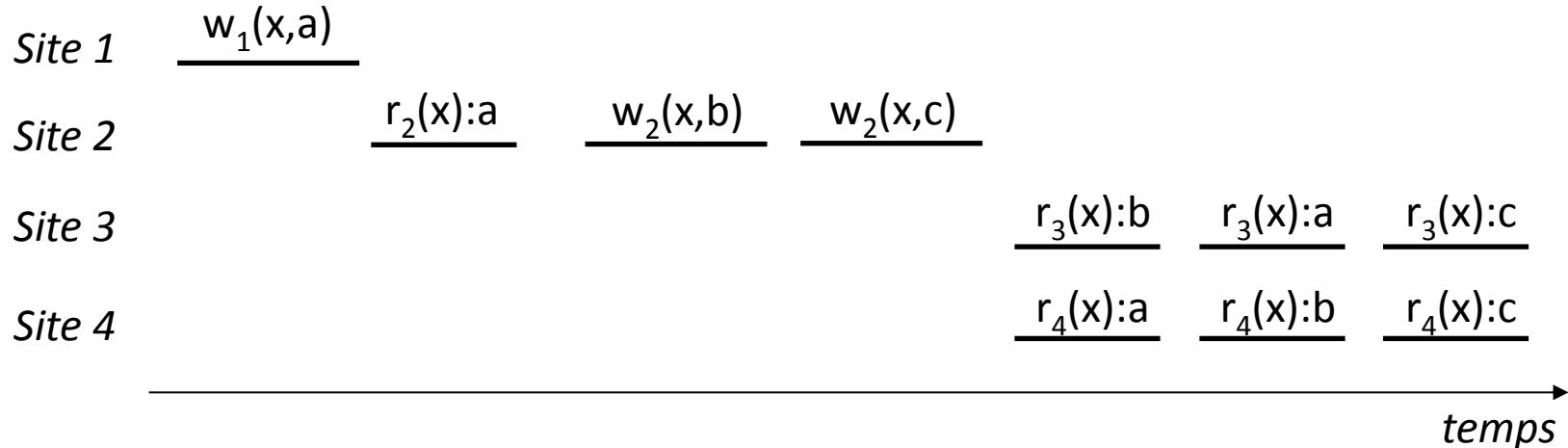
- Duplication et cohérence
  - Cohérence FIFO
    - La cohérence causale peut encore être affaiblie, si on ne considère la causalité qu'à l'intérieur d'un seul processus, non entre processus différents. Dans un processus unique, la causalité se réduit à l'ordre FIFO
    - Des écritures réalisées par un même processus doivent être vues par tous les processus dans leur ordre de réalisation. Des écritures réalisées par des processus différents peuvent être vues dans un ordre différent

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Cohérence Causale

- Exemple 8 :



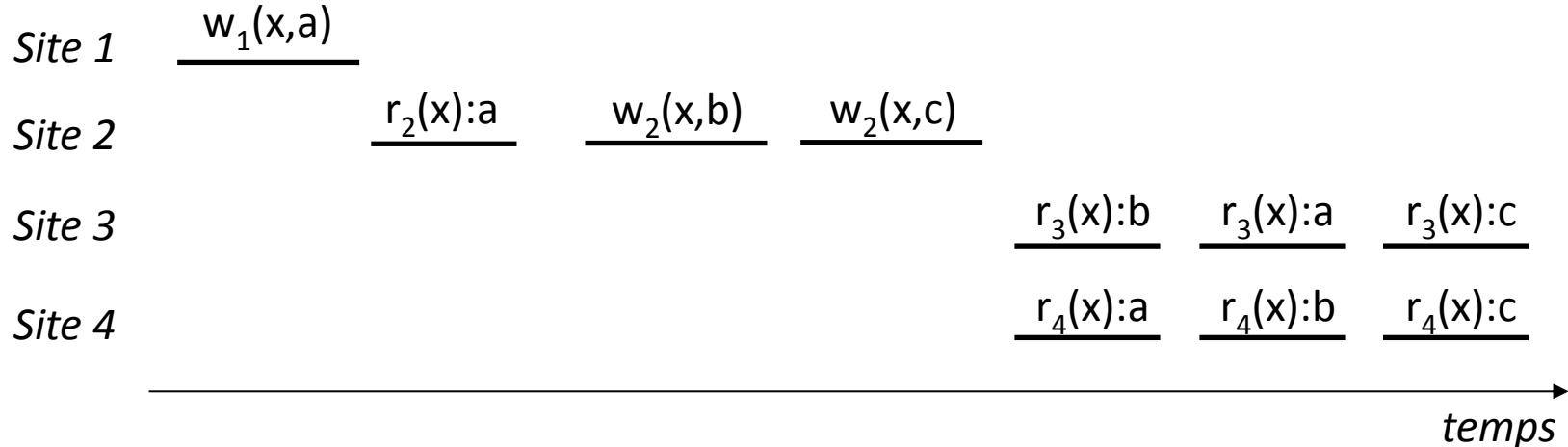
- Exécution causalement consistante?
    - Exécution FIFO consistante?

# Systèmes collaboratifs distribués

- Duplication et cohérence

- Cohérence Causale

- Exemple 8 :



- Exécution causalement consistante? Non
    - Exécution FIFO consistante? Oui, r(b) précède r(c) partout ; r(a) est indépendant

# Systèmes collaboratifs distribués

- ... Soutenances le 27/02 de 10h15 à 11h45 et de 13h45 à 20h30
  
- Examen le 17/04 de 14h15 à 16h15

# Références

- J. Lonchamp, Le travail coopératif et ses technologies , Hermes science, Lavoisier, ISBN : 2-7462-0668-4, 2003
- A. S. Tanenbaum et M. van Steen, Distributed Systems Principles and Paradigms, Pearson Prentice, Hall, 2007
- P. Molli, Systèmes Distribués Collaboratifs, <http://pagesperso.lina.univ-nantes.fr/~molli-p/pmwiki/pmwiki.php/Main/Polytech11>
- S. Krakowiak, Algorithmique et techniques de base des systèmes répartis, <http://pagesperso.lina.univ-nantes.fr/~molli-p/pmwiki/uploads/Main/krakodup.pdf>
- C. Godart, O. Perrin, Les processus métiers : concepts, modèles et systèmes, Hermes, 2009
- A. H. M. ter Hofstede, W. M.P. van der Aalst, M. Adams, N. Russell, Modern Business Process Automation: YAWL and its Support Environment , ISBN 978-3-642-03120-5, 2010
- M. Diaz, Les Réseaux de Petri – Modèles fondamentaux, Hermes Science Publications, 2001
- BPEL, Web Services Business Process Execution Language Version 2.0, [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsbpel](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel)
- Ngoc Chan Nguyen, [http://www-inf.it-sudparis.eu/~nguyen\\_n/teaching\\_assistant](http://www-inf.it-sudparis.eu/~nguyen_n/teaching_assistant)