

# Les Architectures Orientées Services

SOA

**Démarche SOA : Présentation & Applications Composites**

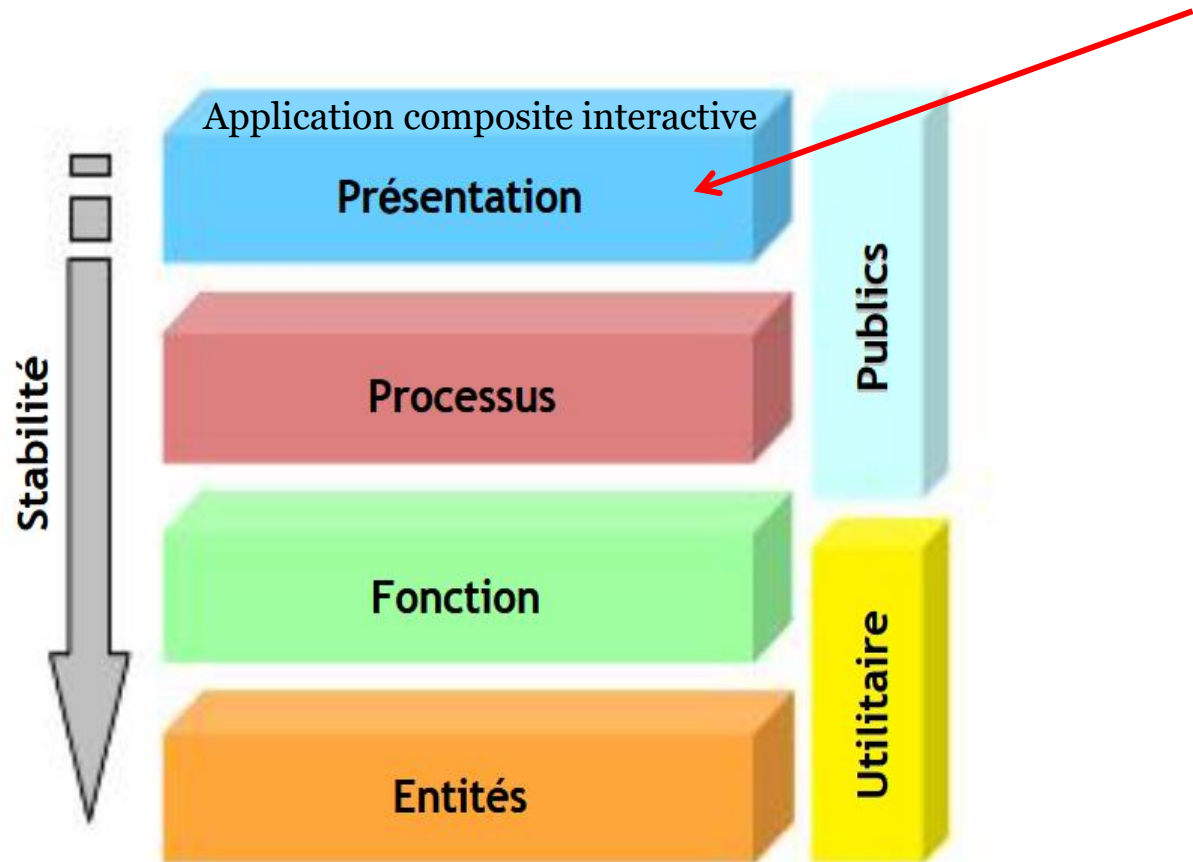
# Plan

1. Composant Présentation
2. Applications Composites
3. Pattern MVC - Classique
4. Pattern MVC revisité - SOA
5. Session et Contexte utilisateur

## Rappel - Hiérarchie de services

**Typologie de services – SEA** - 4 couches logiques + Utilitaire et Public

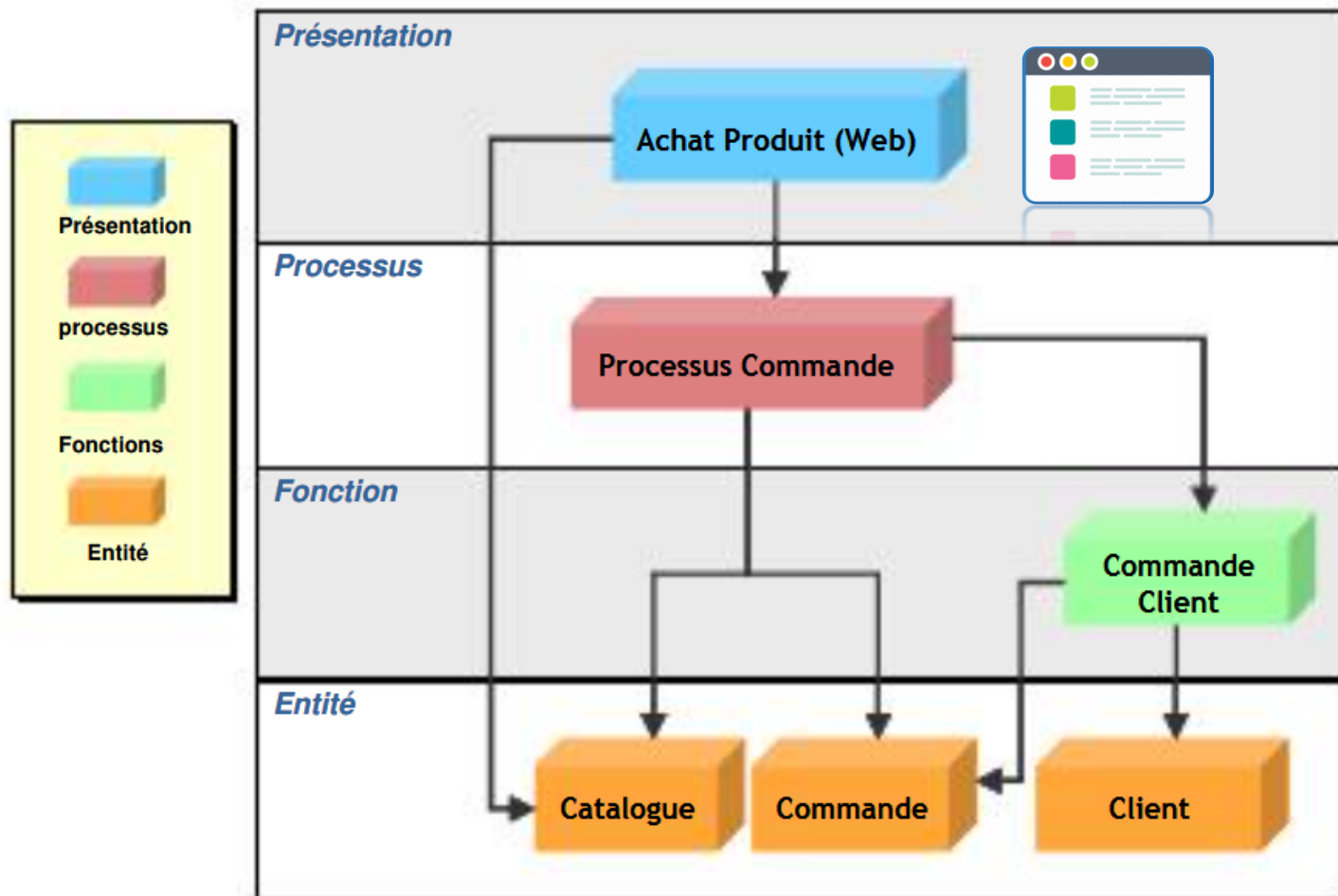
Un composant ne peut pas utiliser un composant d'une couche d'un niveau supérieur




## Rappel - Hiérarchie de services

### Typologie de services – **SEA** - 4 couches logiques

Exemple:



# Composant Présentation



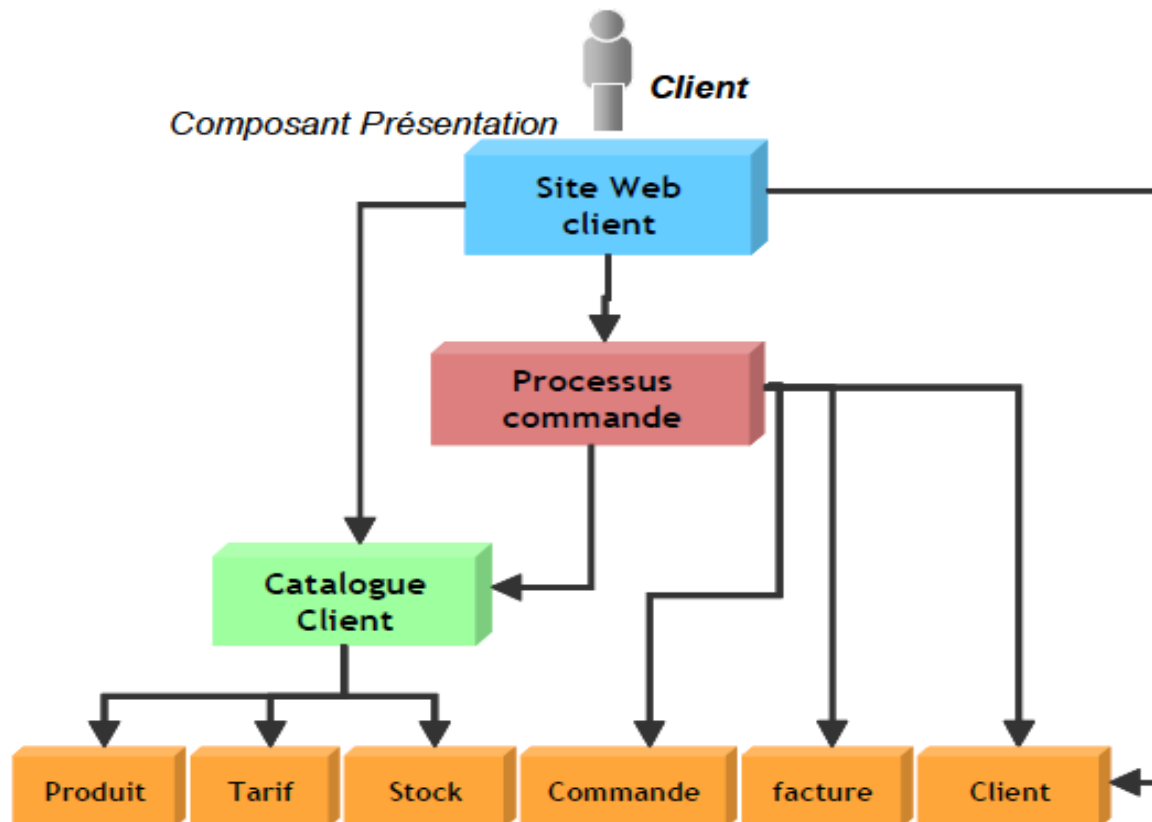
Présentation

- Présentation, ou Service applicatif, ou Use Case
- Permet de mettre en œuvre la logique applicative d'une application telle qu'elle a été identifiée par les cas d'utilisation ou les processus métiers.
- Il est fortement lié à la logique de l'application qui a nécessité sa création ; en général, il n'est donc pas réutilisable, hors du contexte de l'application.
- Le service applicatif active des règles de gestion qui vont conduire, dans le contexte de l'application, à la modification de quelques objets métier.
- Il gère le dialogue entre le système et les acteurs externes. Il gère également l'évolution des données et leurs modifications au cours d'une session.
- Il assure la gestion des interfaces homme machine et la maintenance du contexte session de l'utilisateur.

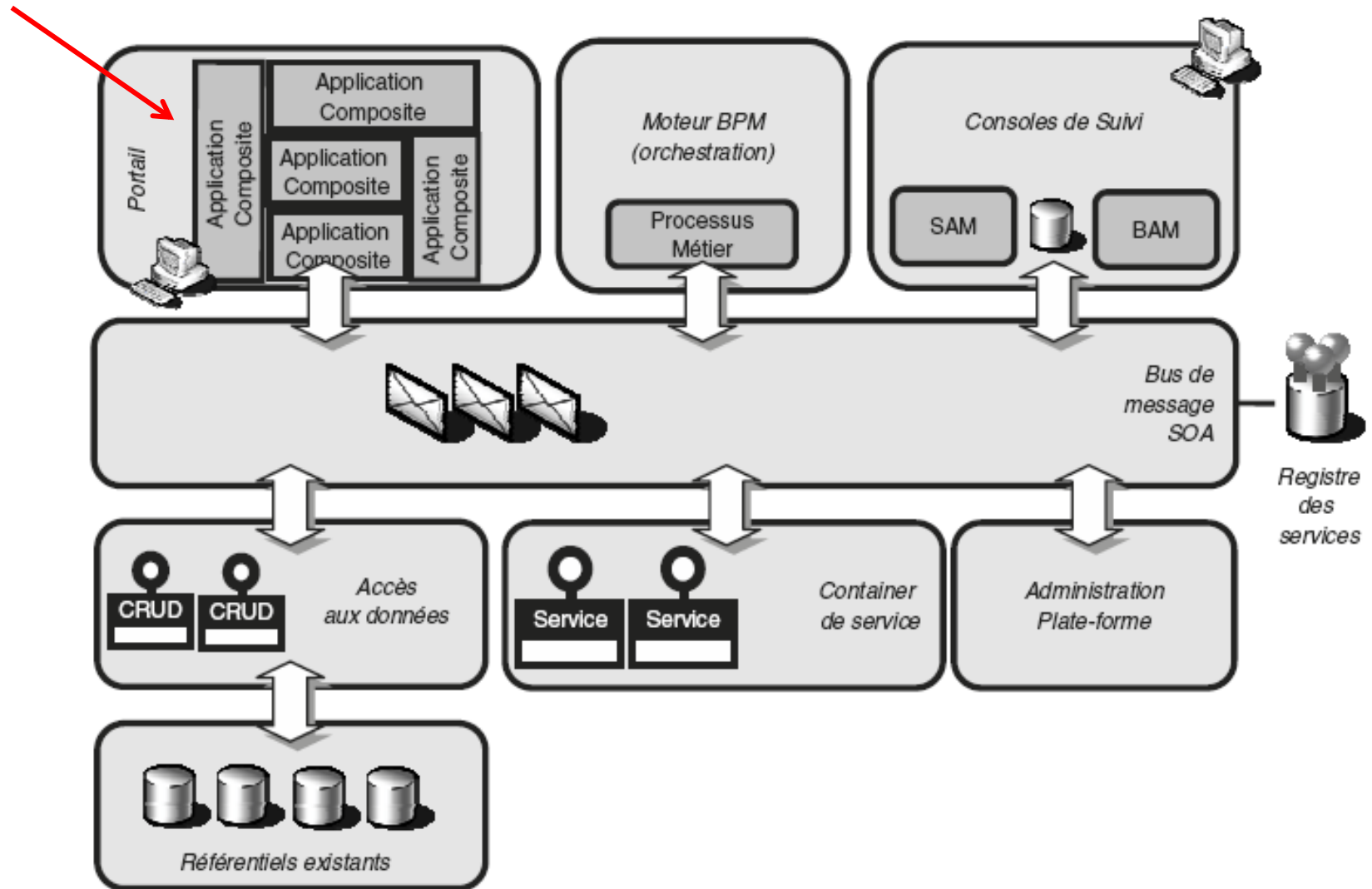
## Composant Présentation

Présentation

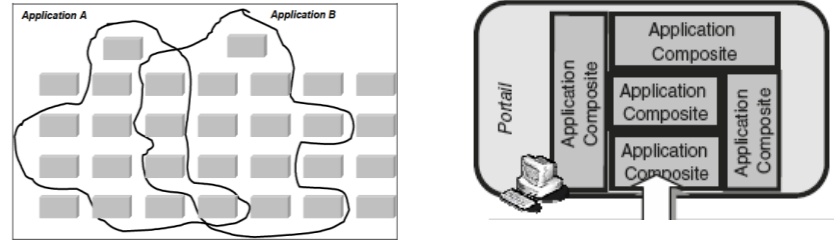
- Les **composants Présentation** ne sont pas des composants de service proprement dit : ils ne fournissent pas de services, sauf à l'utilisateur.
- Ils sont **consommateurs de service** pour tout autre type de composant.



# Plateforme SOA



# Applications Composites

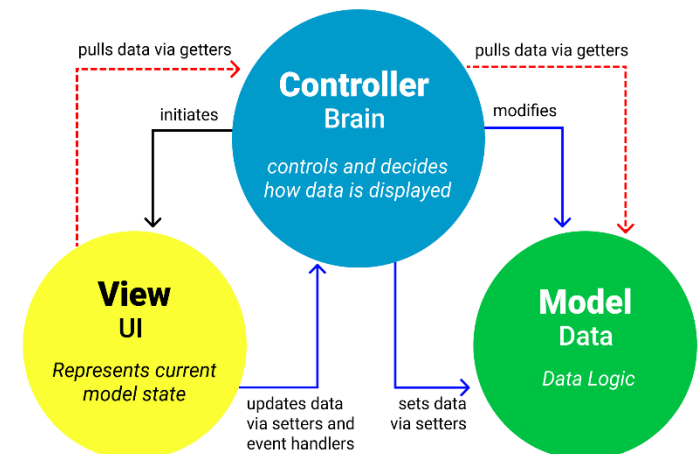


- Les applications sont les éléments tangibles du point de vue des utilisateurs du système.
- Application dans **SOA** = Application Composite Interactive.
- Une **application composite** est constituée par un ensemble de composants qui concourent pour répondre aux besoins dédiés à une ligne métier ou une utilisation spécifique du système.
- Typiquement on va trouver dans une application composite un composant Présentation (IHM, session utilisateur) qui s'appuie sur des composants de services de diverses natures (Processus, Fonction, Entité, etc).
- Dans une architecture SOA, les projets applicatifs vont potentiellement **réutiliser et partager des composants de service**, qui n'appartiennent pas à une application particulière.



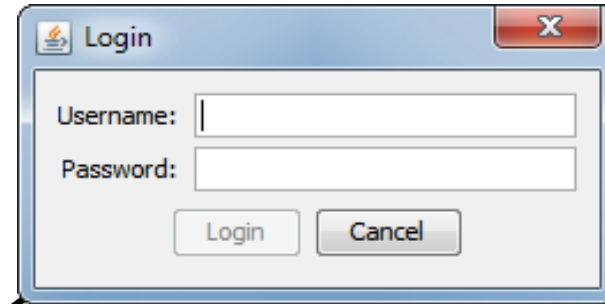
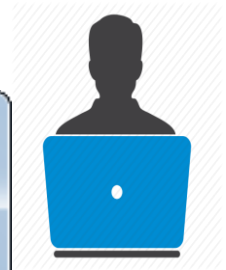
## Pattern MVC - Classique

- Le **pattern MVC** est le modèle architectural logiciel de référence pour concevoir des applications graphiques interactives.
- MVC : 3 types de composants = Modèle – Vue – Contrôleur
- La **Vue** affiche les informations attendues par l'utilisateur final (IHM).
- Le **Contrôleur** reçoit les demandes (actions) de l'utilisateur et contrôle le comportement de l'application interactive pour répondre à ces demandes;
- Le **Modèle** contient la logique et les informations métier nécessaires pour répondre aux demandes de l'utilisateur;



## Pattern MVC - Classique

Exemple :

A screenshot of a classic Windows-style login dialog box. The title bar says "Login" with a close button (X) on the right. Inside the dialog, there are two input fields: "Username:" and "Password:". Below the fields are two buttons: "Login" and "Cancel".

Clic Login

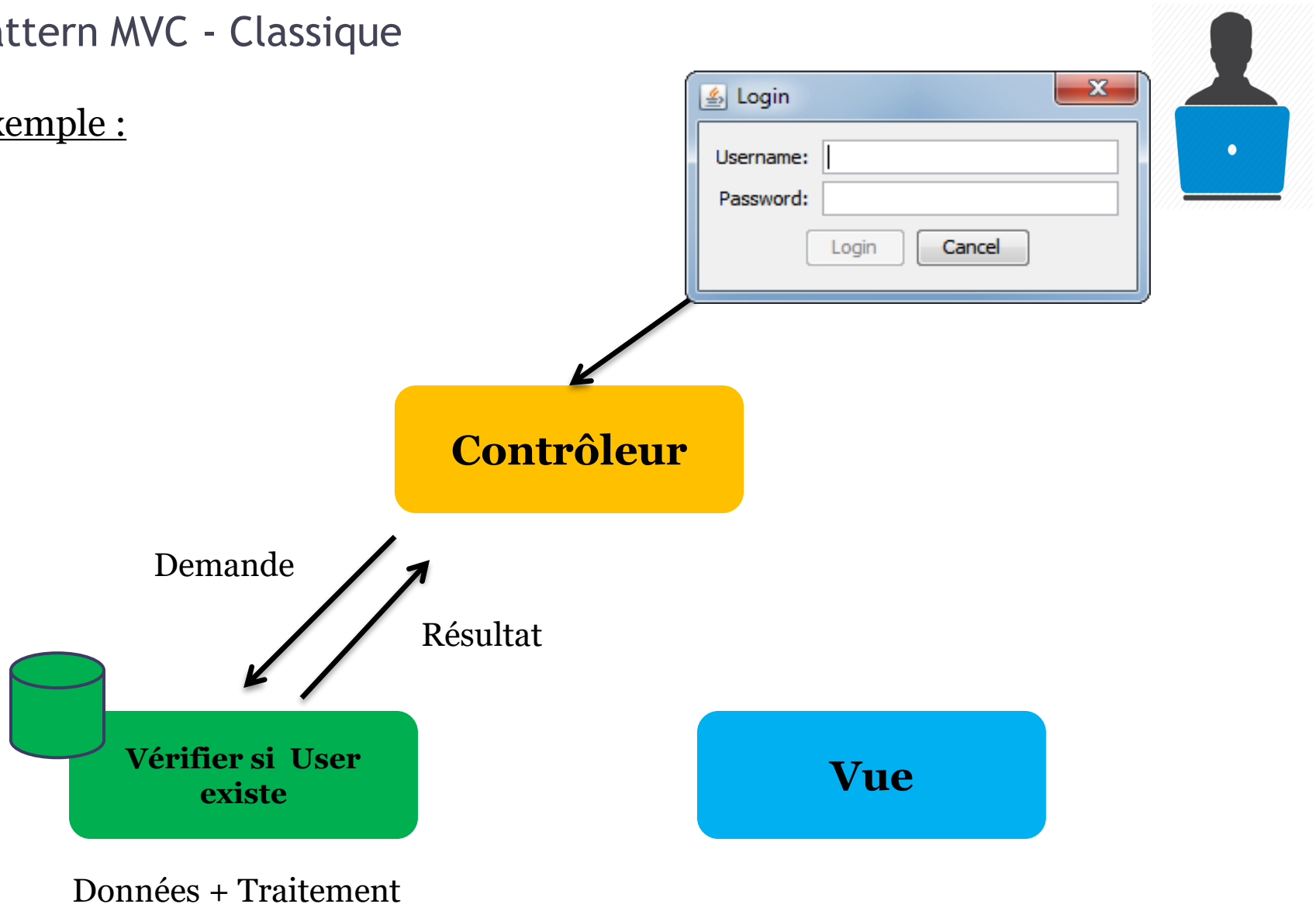
**Contrôleur**

**Modèle**

**Vue**

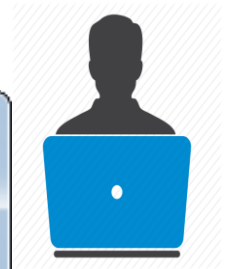
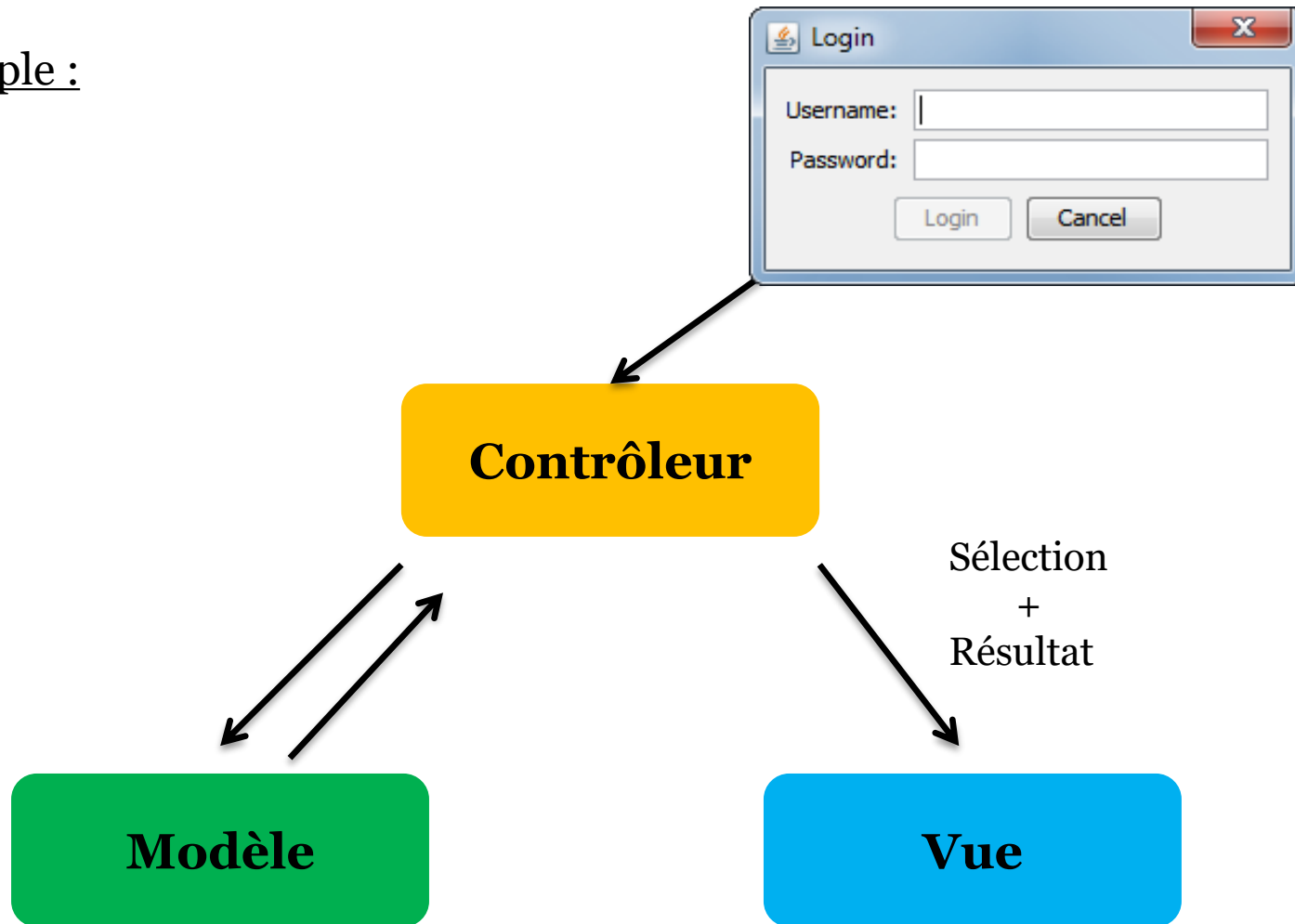
# Pattern MVC - Classique

Exemple :



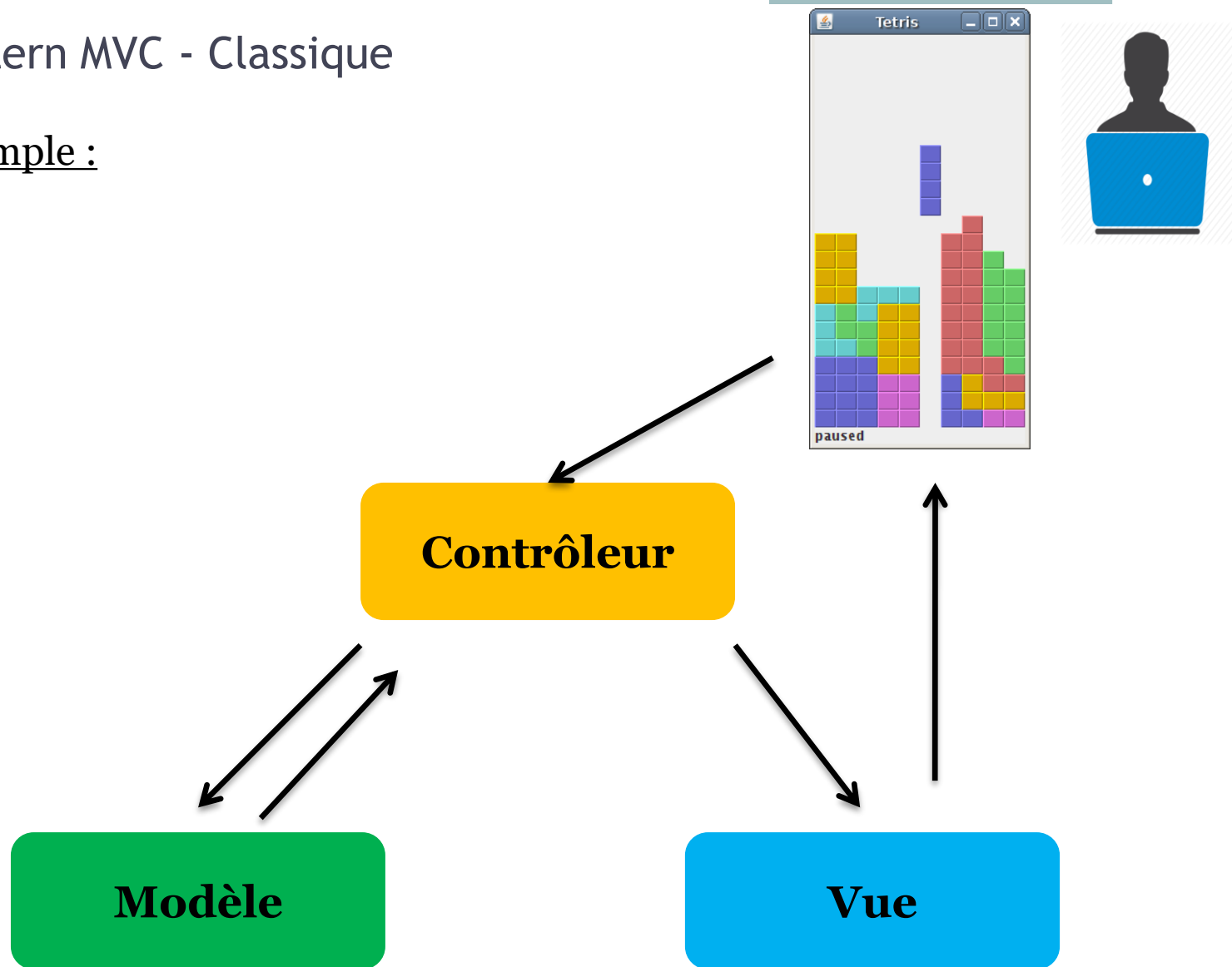
# Pattern MVC - Classique

Exemple :



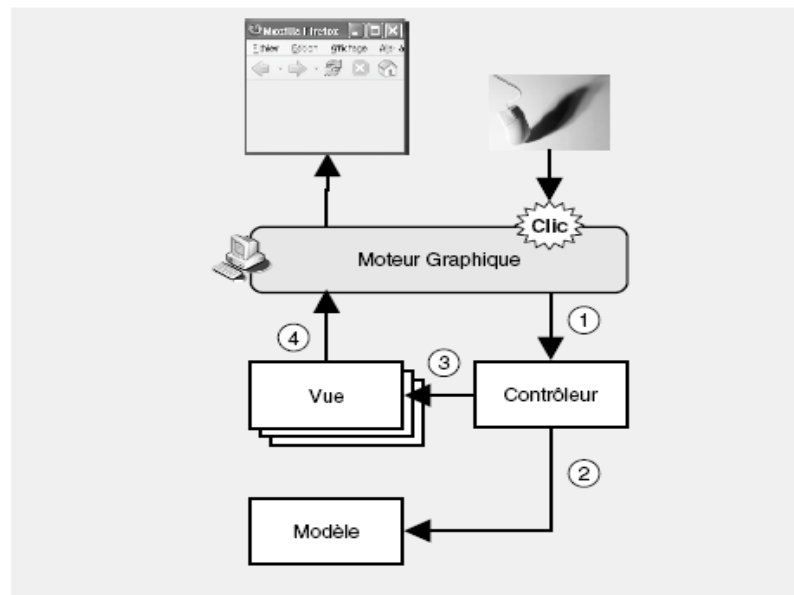
# Pattern MVC - Classique

Exemple :



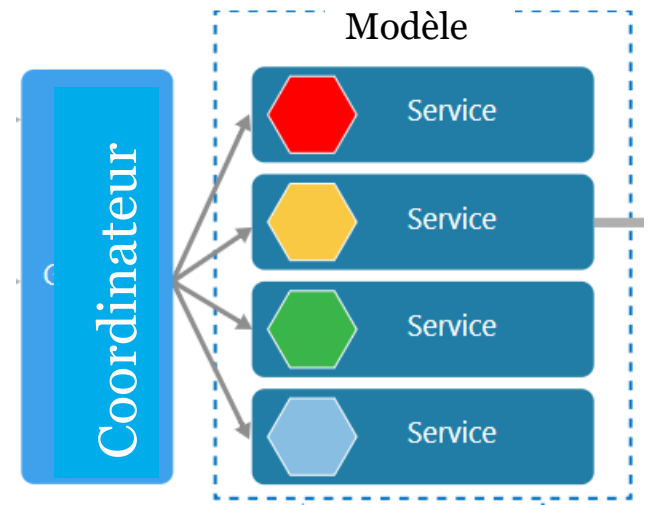
## Pattern MVC - Classique

- Le **Modèle** contient :
  - Les données
  - Les traitements de l'application
- La **Vue** :
  - Présente les informations du modèle
  - Permet à l'utilisateur d'agir sur le modèle
- Le **Contrôleur** gère la synchronisation entre la vue et le modèle

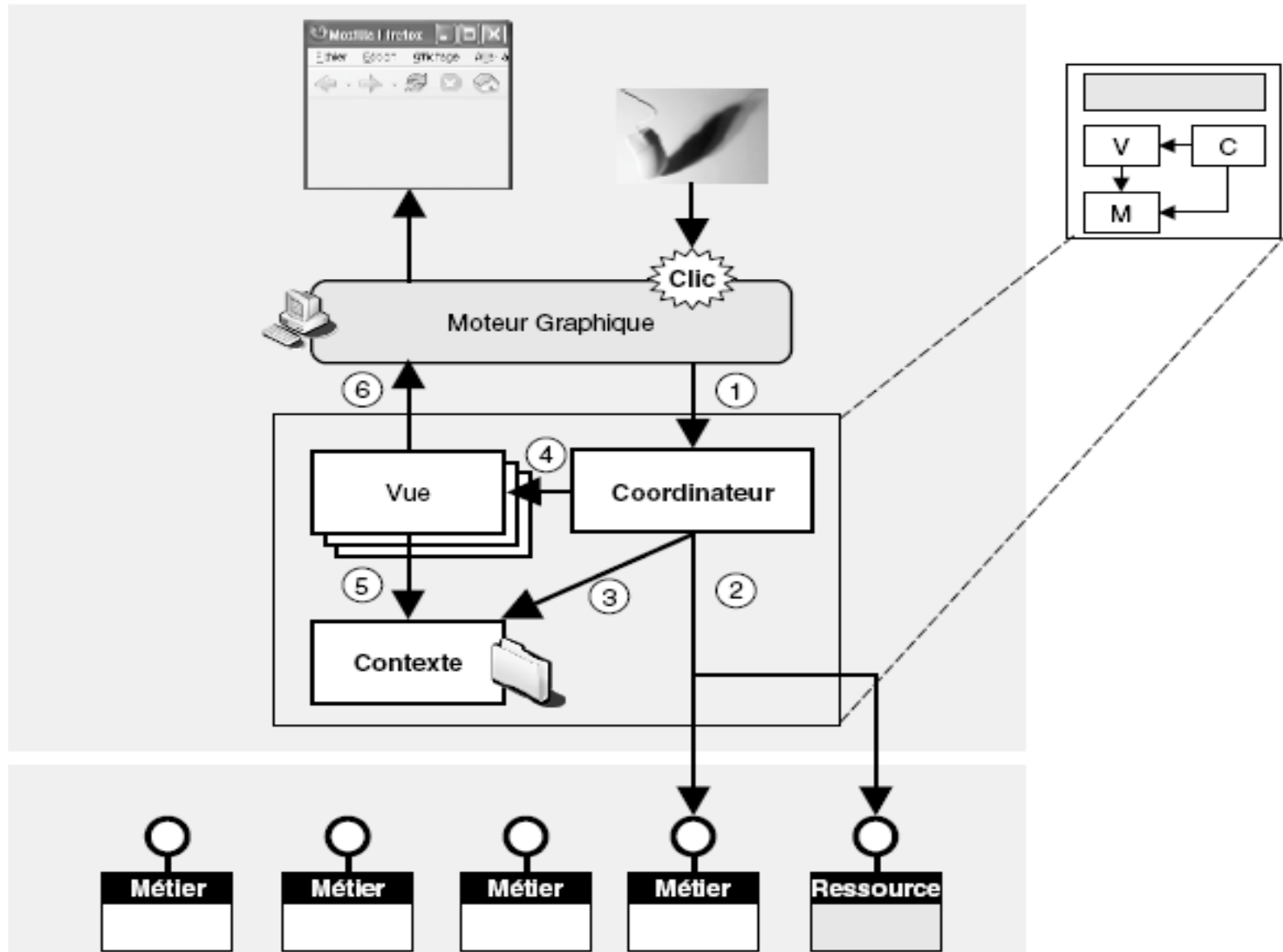


## Pattern MVC revisité pour SOA

- Le pattern MVC a été revisité pour les architectures SOA.
- La différence la plus évidente concerne le Modèle.
- **Modèle = ensembles de services.**
- Aussi, le Contrôleur n'active plus directement le Modèle, mais fait appel à un ou plusieurs Services.
- Le **Contrôleur = Coordinateur d'appel de services.**

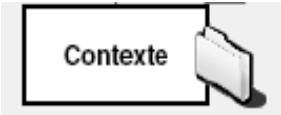


# Pattern MVC revisité pour SOA

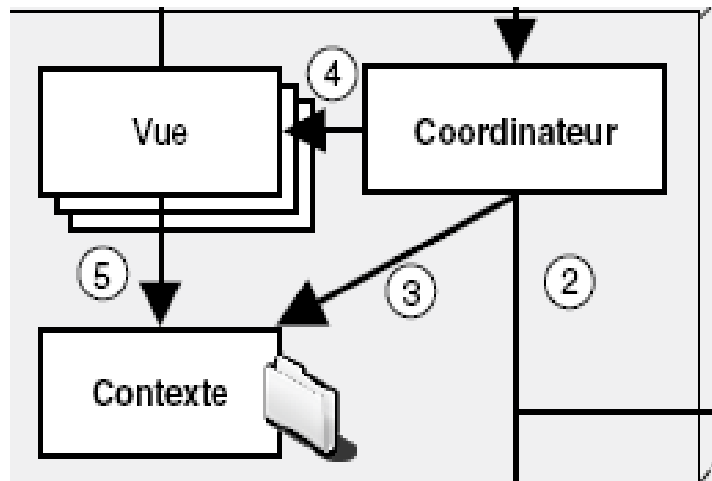




## Session et Contexte Utilisateur



- Lancement d'une application composite interactive par un utilisateur = **Session de travail**.
- L'utilisateur accède à un ensemble d'objets métier (graphe).
- Le **Contexte** est le composant dans lequel l'application composite va stocker pendant la durée de la session de travail les objets métier dont elle a besoin pour répondre aux demandes de l'utilisateur associé à cette session.
- Une instance de Contexte par Session de travail.



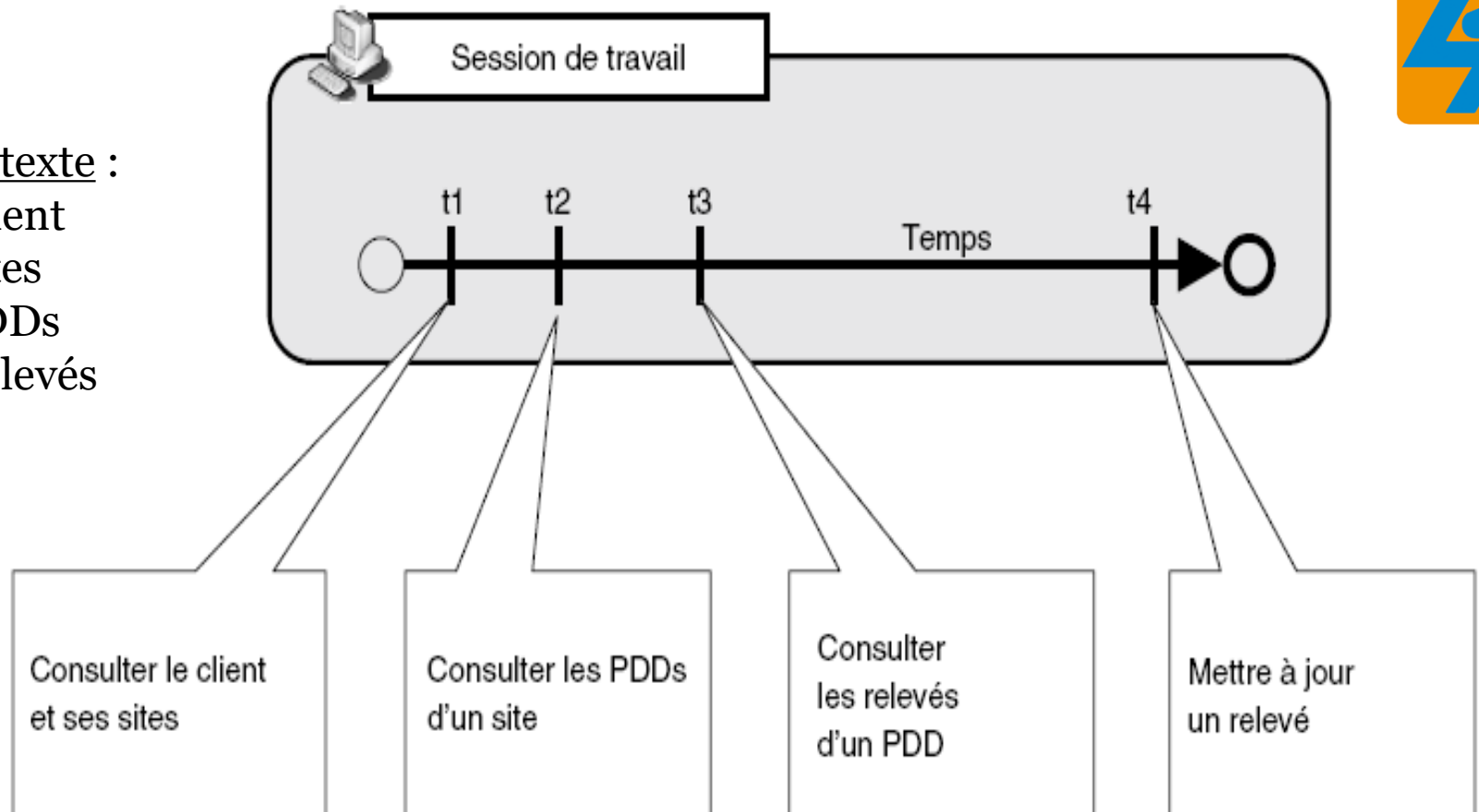
## Session et Contexte Utilisateur

Contexte

**Exemple** d'une Session de travail après le lancement de l'application composite « consulter et rectifier les Relevés de Consommation Electrique »

Contexte :

- Client
- Sites
- PDDs
- Relevés



## Session et Contexte Utilisateur

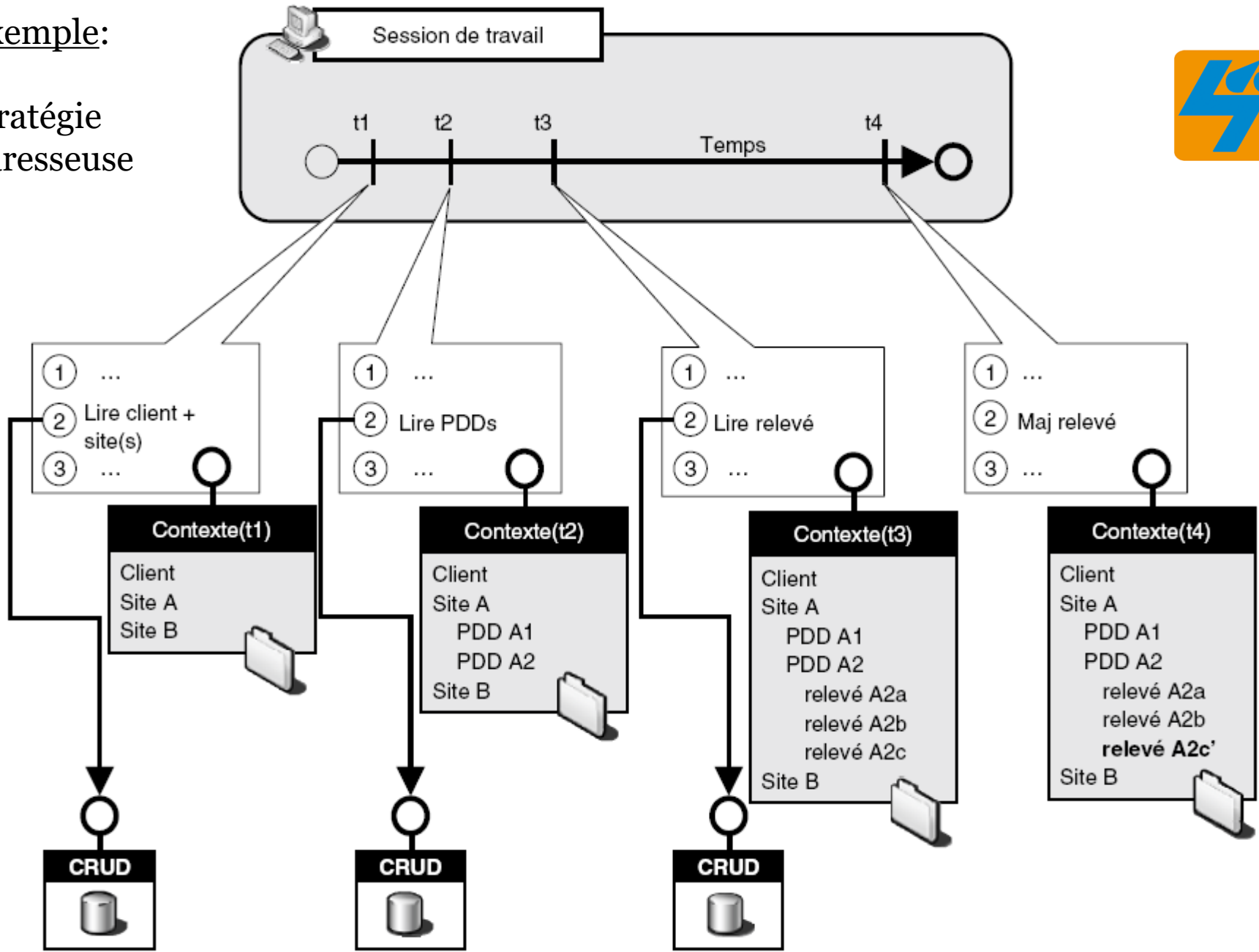


Contexte

- Deux stratégies pour **charger**/remplir un **Contexte** :
  - **Stratégie par anticipation** : l'application charge en une seule fois l'ensemble des informations métier nécessaires à partir des services CRUD appropriés.
  - **Stratégie paresseuse** : l'application charge une information dans le contexte, uniquement quand l'utilisateur demande cette information. Cela revient à charger les informations une par une.
- Le **choix de la stratégie** à adopter dépend des besoins métier.
  - La stratégie par anticipation autorise une navigation fluide entre les informations, puisque ces informations sont déjà contenues dans le contexte. Inconvénient : certaines informations seront inutiles.
  - Stratégie paresseuse : chargement individuel plus performant mais navigation moins fluide (services CRUD).

Exemple:

Stratégie  
Paresseuse



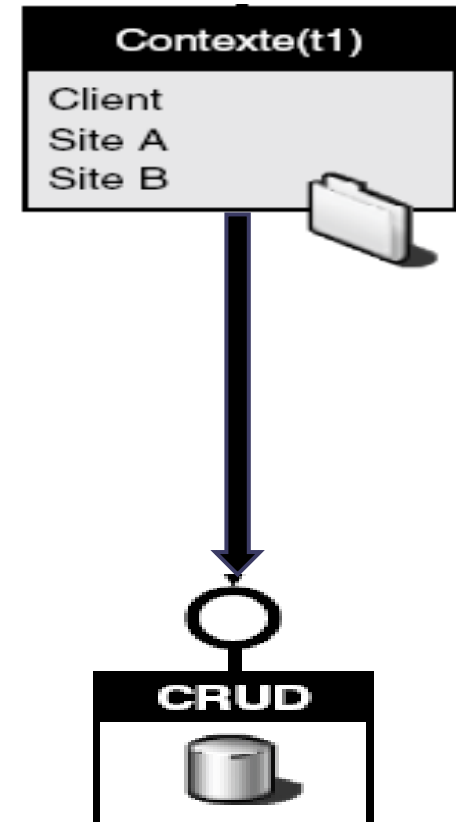
### ➤ Sauvegarde du Contexte

- Lors d'une session de travail, l'utilisateur **consulte** les informations métier mais aussi les **modifie** et en **crée** de nouvelles.
- L'utilisateur doit avoir un « **droit à l'erreur** », c'est-à-dire la possibilité de revenir en arrière ou de modifier plusieurs fois la même information dans la même session de travail.
- Les modification qu'il effectue ne seront pas immédiatement répercutées vers le ou les référentiels concernés.
- L'application devra **attendre la fin de session pour sauvegarder** ces mises à jour ou création.

## Session et Contexte Utilisateur

### ➤ Sauvegarde du Contexte

- La sauvegarde du Contexte ne doit pas se faire « en bloc ».
- Le mécanisme de sauvegarde devra **sélectionné** que l'objet qui a été modifié, ou celui qui vient d'être créé.
- Pourquoi ?

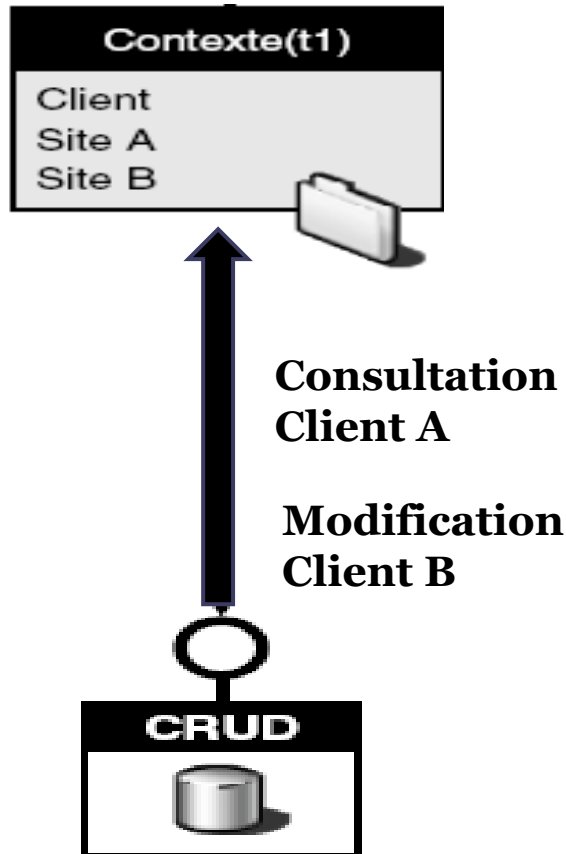


## Session et Contexte Utilisateur

### Sauvegarde du Contexte - **Problèmes**

Chargement de Contexte

**User 1**

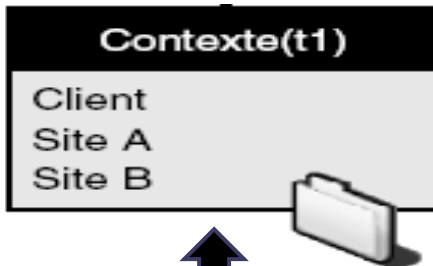


## Session et Contexte Utilisateur

### Sauvegarde du Contexte - **Problèmes**

Chargement de Contexte

**User 1**

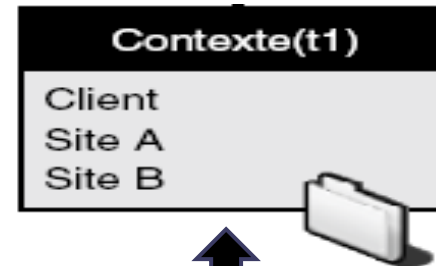


**Consultation  
Client A**

**Modification  
Client B**



**User 2**



**Modification  
Client A**



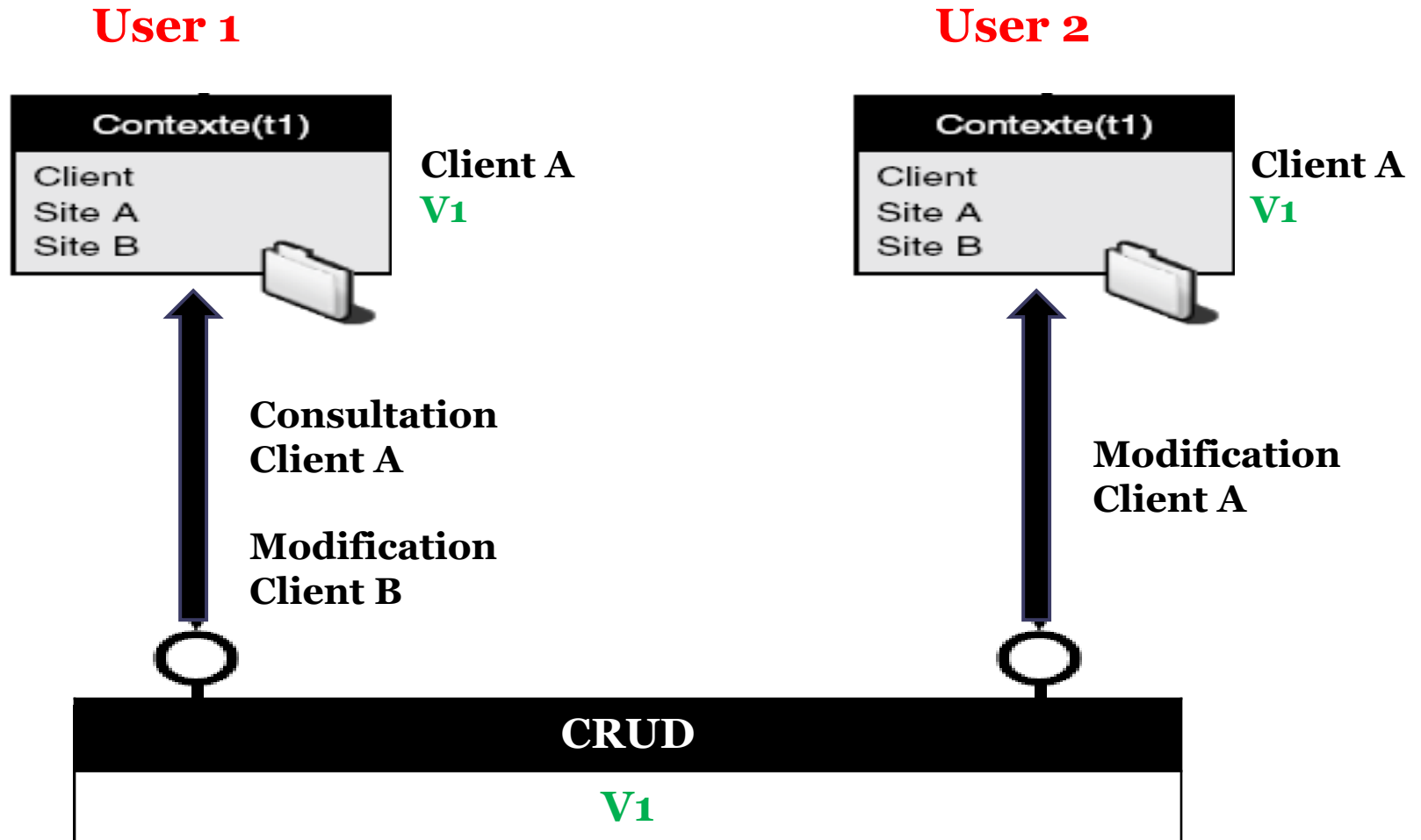
**CRUD**





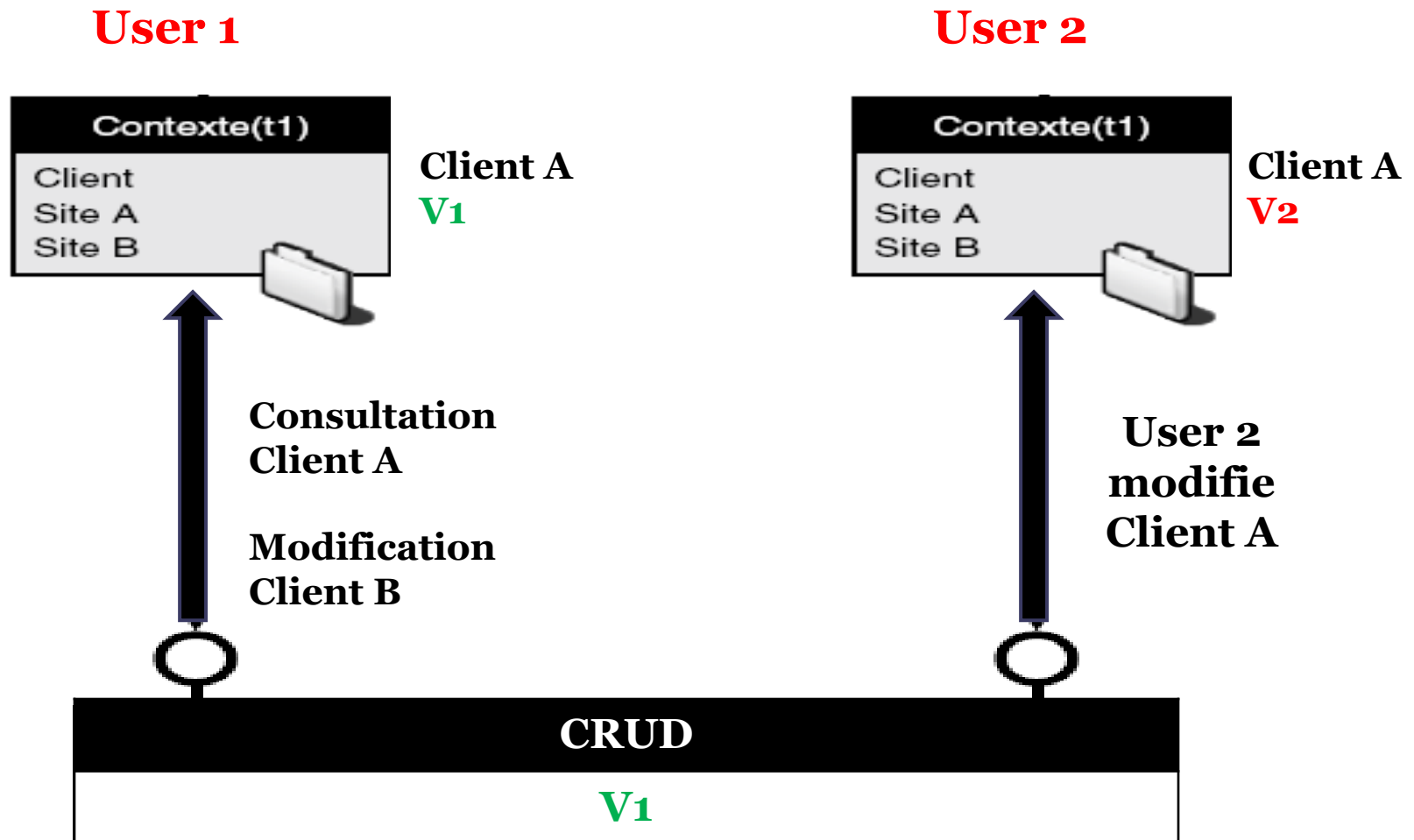
## Session et Contexte Utilisateur

### Sauvegarde du Contexte - **Problèmes**



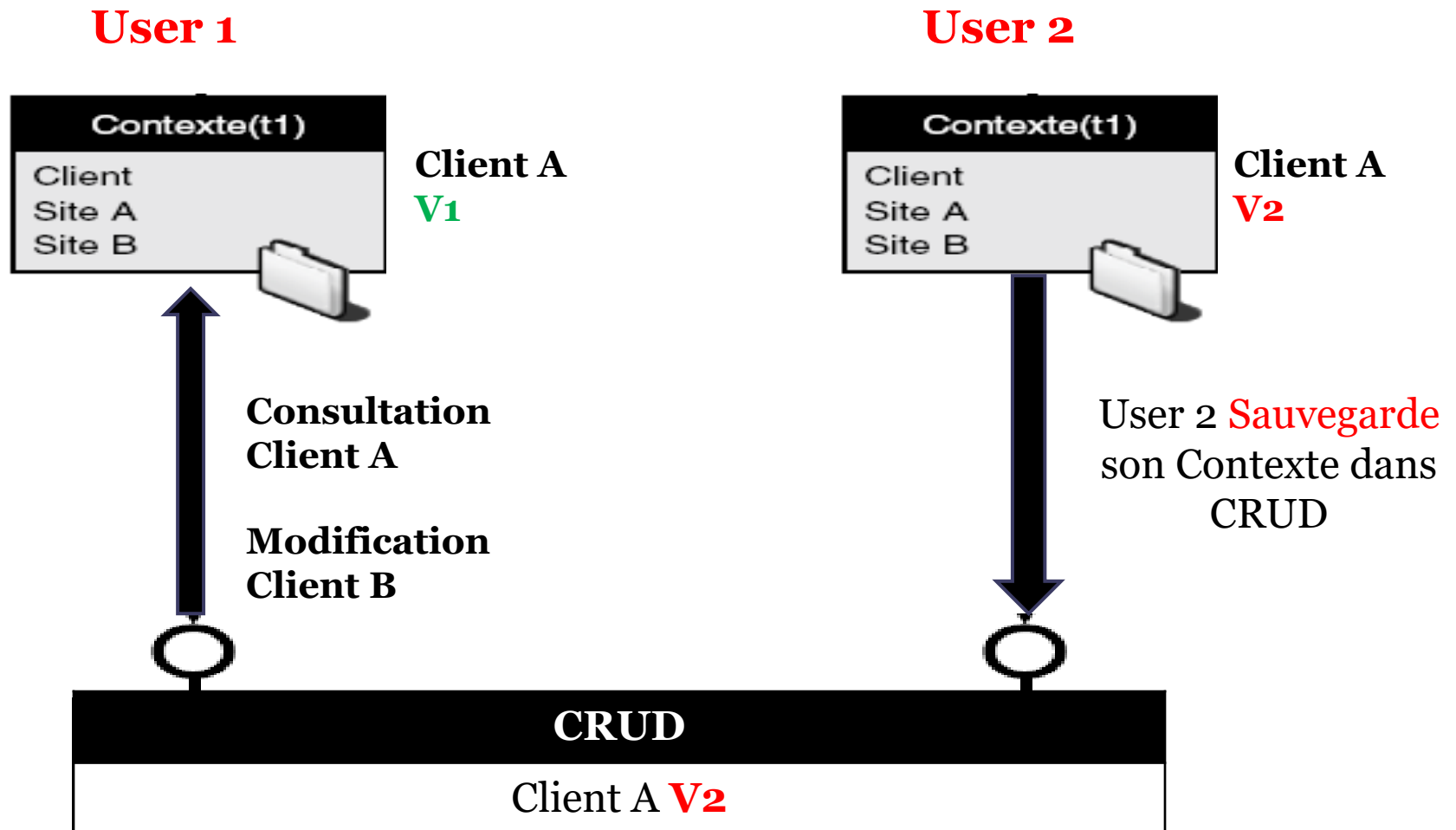
## Session et Contexte Utilisateur

### Sauvegarde du Contexte - **Problèmes**



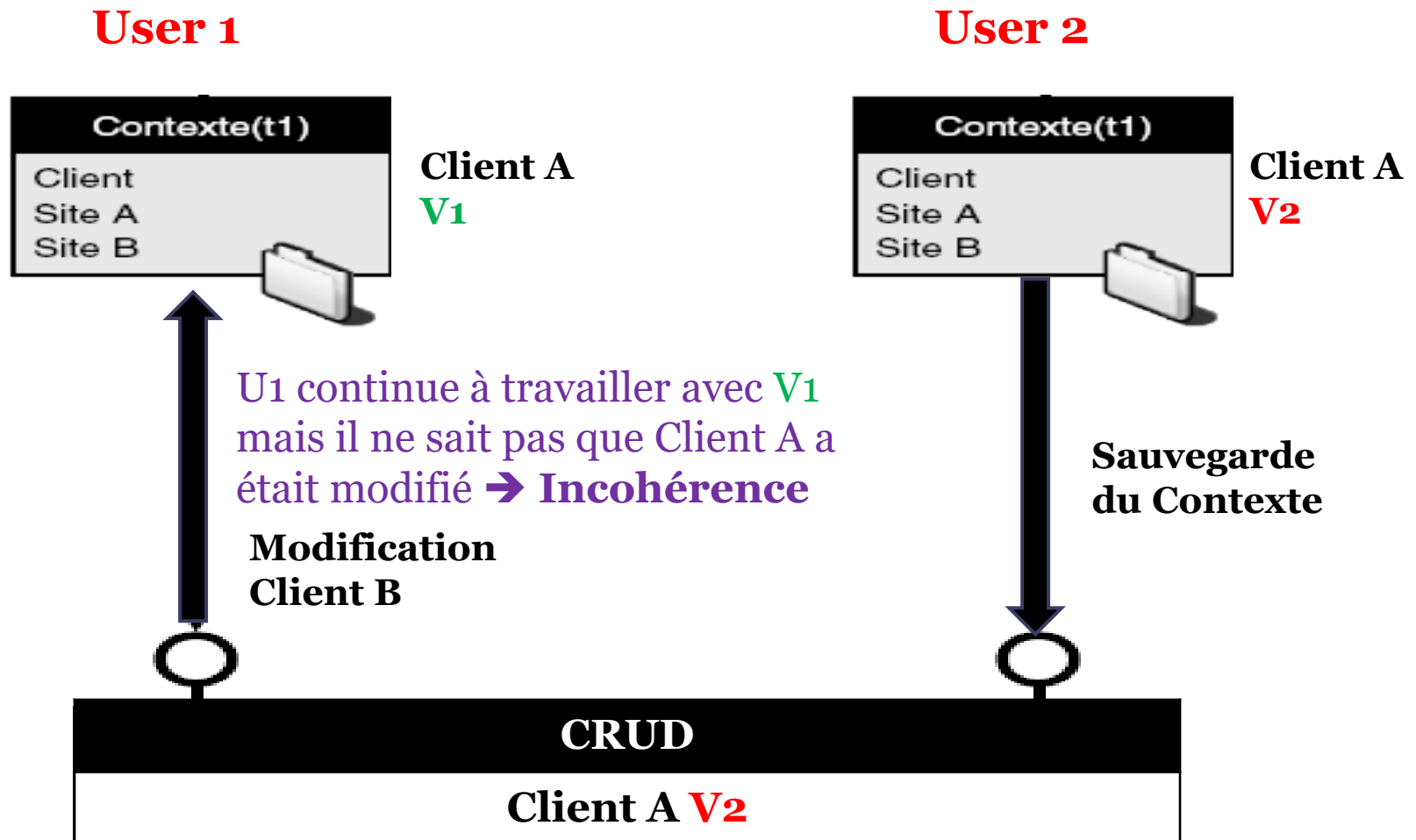
## Session et Contexte Utilisateur

### Sauvegarde du Contexte - **Problèmes**



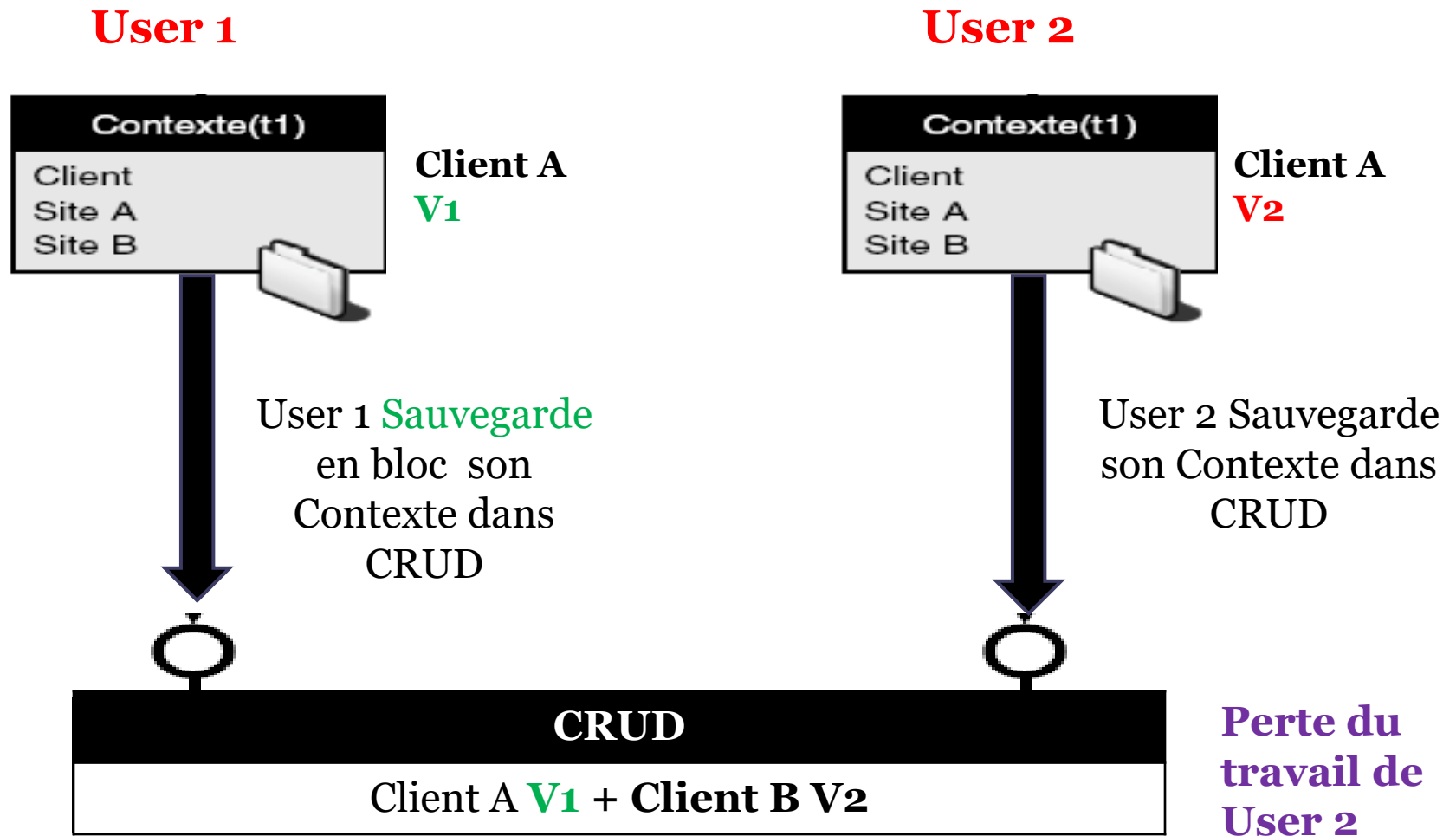
## Session et Contexte Utilisateur

### Sauvegarde du Contexte - **Problème 1**



## Session et Contexte Utilisateur

### Sauvegarde du Contexte - **Problème 2**



## Session et Contexte Utilisateur

### Sauvegarde du Contexte - Solutions

→ d'où le **Service de Gestion de Contexte**

Ce service offre dans son contrat les **opérations** suivantes :

1. **Créer** un nouveau Contexte;
2. **Ecrire** un objet dans un Contexte – cette opération détecte et évite les doublons;
3. **Récupérer** un objet présent dans un Contexte;
4. **Sauvegarder** un Contexte – cette opération ne sauvegarde dans le ou les référentiels concernés que les objets modifiés ou nouvellement créés dans ce Contexte;
5. **Supprimer** un Contexte.

## Session et Contexte Utilisateur

### Sauvegarde du Contexte - Solutions

- ➔ d'où le **Mécanisme de verrouillage métier**.
- Objectif : permettre à une application composite de prévenir un de ses utilisateurs qu'un autre utilisateur est déjà en train de travailler (en lecture ou en écriture) sur un objet métier.
  - Une application composite, lorsqu'elle va charger dans un Contexte un Objet Métier, **pose un verrou « métier » sur cet objet métier**.
  - Ce verrou peut être un verrou « **pour Consultation** » (l'application ne modifiera pas cet objet métier), ou un verrou « **pour Modification** ».



## Session et Contexte Utilisateur

### Sauvegarde du Contexte - Solutions

- ➔ d'où le **Mécanisme de verrouillage métier**.
- Une autre session de travail, souhaitant également charger le même Objet Métier, vérifiera d'abord systématiquement la présence ou non d'un verrou sur cet objet.
- S'il n'y a pas de verrou, alors le chargement peut avoir lieu.
- S'il y a un verrou, soit la session est bloquée soit l'utilisateur est alerté seulement.
- ➔ **Service de gestion des verrous**





### Sauvegarde du Contexte - Solutions

#### ➔ **Service de gestion des verrous**

➤ Ce service offre dans son contrat les opérations suivantes :

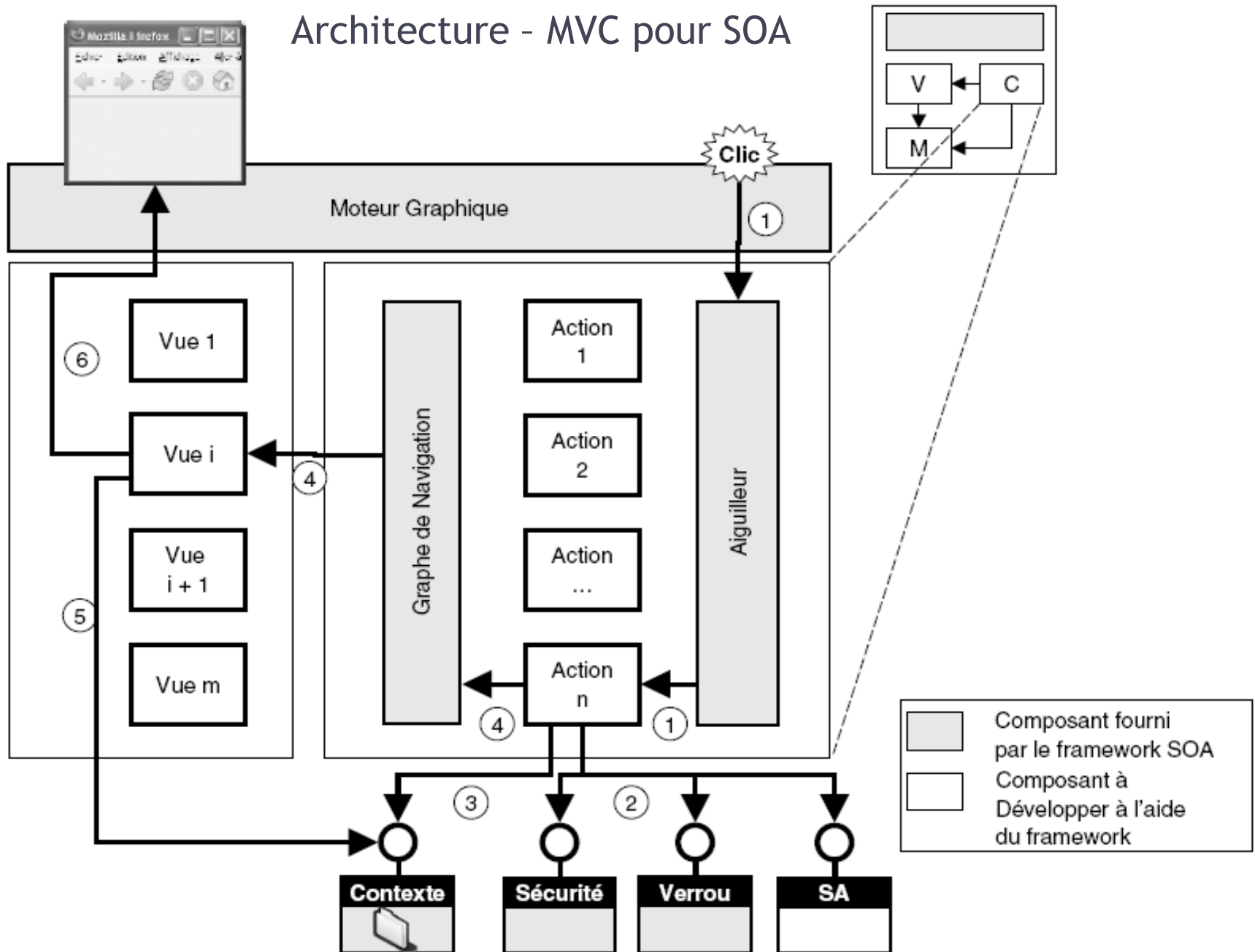
1. **isLocked**(businessObject) : cette opération permet de savoir si un verrou est déjà posé sur l'objet métier.
2. **getLocks**(businessObject) : cette opération permet de récupérer l'ensemble des verrous posés sur un objet métier.
3. **removeLock**(user, businessObject, lockMode) : cette opération permet de supprimer un verrou.
4. **setLock**(user, businessObject, lockMode) : cette opération permet de poser un verrou du type défini sur l'objet métier.

## Session et Contexte Utilisateur

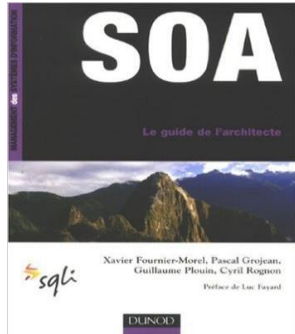
### Service de Gestion de Contexte – Services Web

- Cette vision du Contexte en tant que service à part entière a permis une standardisation autour des normes **WS-CAF** (*Composite Application Framework*), et plus particulièrement de la norme **WS-Context**.
- Ces normes permettent à l'application composite de déléguer la gestion de son Contexte.
- Le concepteur/développeur de toute Application Composite n'a plus à se préoccuper de l'appel des services CRUD et du remplissage du Contexte; il se contente d'utiliser les services offerts par les normes.

## Architecture - MVC pour SOA



# Ressources



## **Le guide de l'architecte du SI**

- ✓ Auteur : Xavier Fournier-Morel, Pascal Grosjean, ...
- ✓ Éditeur : Dunod
- ✓ Edition : Octobre 2006 - 302 pages - ISBN : 2100499726



## **SOA Principles of Service Design**

- ✓ Auteur : Thomas Erl
- ✓ Éditeur : Prentice Hall Ptr
- ✓ Edition : Juillet 2007 - 608 pages - ISBN : 0132344823



## **SOA : Architecture Logique : Principes, structures et bonnes pratiques**

- ✓ Auteur : Gilbert Raymond
- ✓ Éditeur : Softeam
- ✓ Edition : Livre Blanc

# Ressources



## **URBANISATION & ARCHITECTURE ORIENTÉE SERVICE (SOA) Quelques bonnes pratiques pour leur mise en oeuvre**

- ✓ Auteur : Cyril Devaux
- ✓ Éditeur : Aubay Management
- ✓ Edition : 2008, Livre Blanc



## **SOA Principles of Service Design**

- ✓ Auteur : Thomas Erl
- ✓ Éditeur : Prentice Hall Ptr
- ✓ Edition : Juillet 2007 - 608 pages - ISBN : 0132344823



## **SOA : Architecture Logique : Principes, structures et bonnes pratiques**

- ✓ Auteur : Gilbert Raymond
- ✓ Éditeur : Softeam
- ✓ Edition : Livre Blanc