


Plan

- 
- Diagrammes fonctionnels
 - Diagrammes statiques
 - *Diagrammes dynamiques*
 - *D'états*
 - *D'activité*
 - *De séquence*
 - *De communication*
 - *Vue générale d'interaction*
 - *De temps*
 - Diagramme d'implémentation

Diagrammes dynamiques

Définition des aspects dynamiques d'une application, plusieurs points de vue

➤ Diagrammes d'états

- Description du comportement ou du fonctionnement d'un objet
- Extension des diagrammes de Harel

➤ Diagrammes d'activité

- Diagrammes de flot de données

➤ Définition des interactions entre des objets

- Description de la coopération d'un ensemble d'objets
- 2 types de diagrammes d'interaction
 - Diagrammes de **séquence** : mise en avant de l'évolution et de l'enchaînement temporel des messages échangés
 - Diagrammes de **communication** : mise en avant des liens entre les objets et les messages échangés au travers de ces liens

Diagramme d'états

Diagrammes d'états : comportement interne d'un objet

- La définition de tous les états possibles d'un objet
- La définition de tous les changements d'états via des transitions
- Associé à un objet à une opération

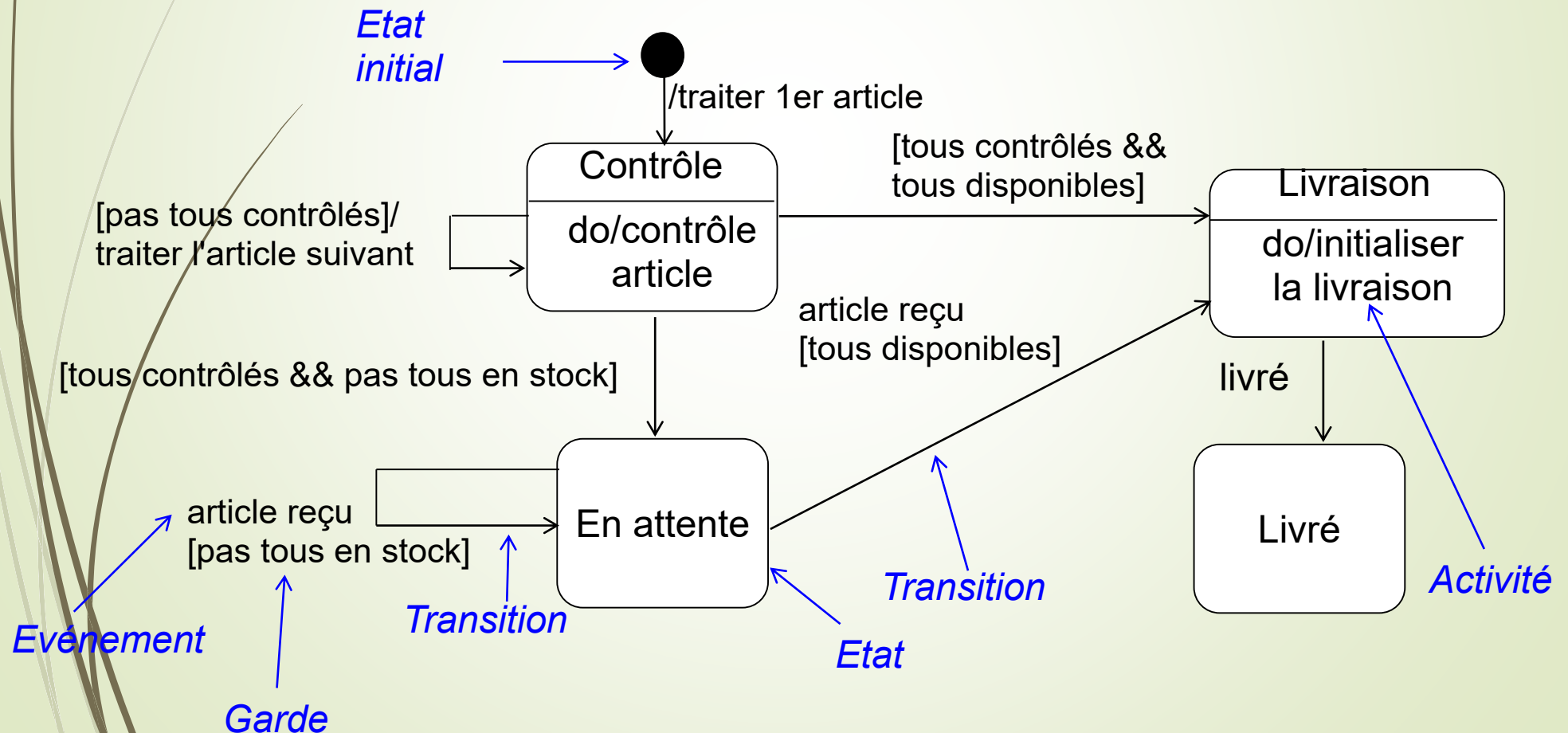


Diagramme d'états

■ Diagrammes d'états – syntaxe

■ Syntaxe d'une transition

- événement [garde] / action
- Chaque partie est optionnelle
- La transition est suivie si l'événement a été généré et que la garde est valide
 - Exécute alors l'action avant de rentrer dans l'état ciblé par la transition

■ Syntaxe des activités que l'on peut associer à un état

- do / action : action exécutée dans l'état
- entry / action : action exécutée à l'entrée dans l'état
- exit / action : action exécutée à la sortie de l'état
- evt / action : transition interne pour l'occurrence de l'événement evt

■ Lien avec l'objet associé au diagramme d'états

- Les actions peuvent être les méthodes de la classe de l'objet
- Peut utiliser les attributs de l'objet, par exemple dans les gardes des transitions

Diagramme d'états

- Diagrammes d'états : notion d'état composite
- Permet de structurer de manière hiérarchique les états et les transitions
- Exemple d'une commande annulée sans super état

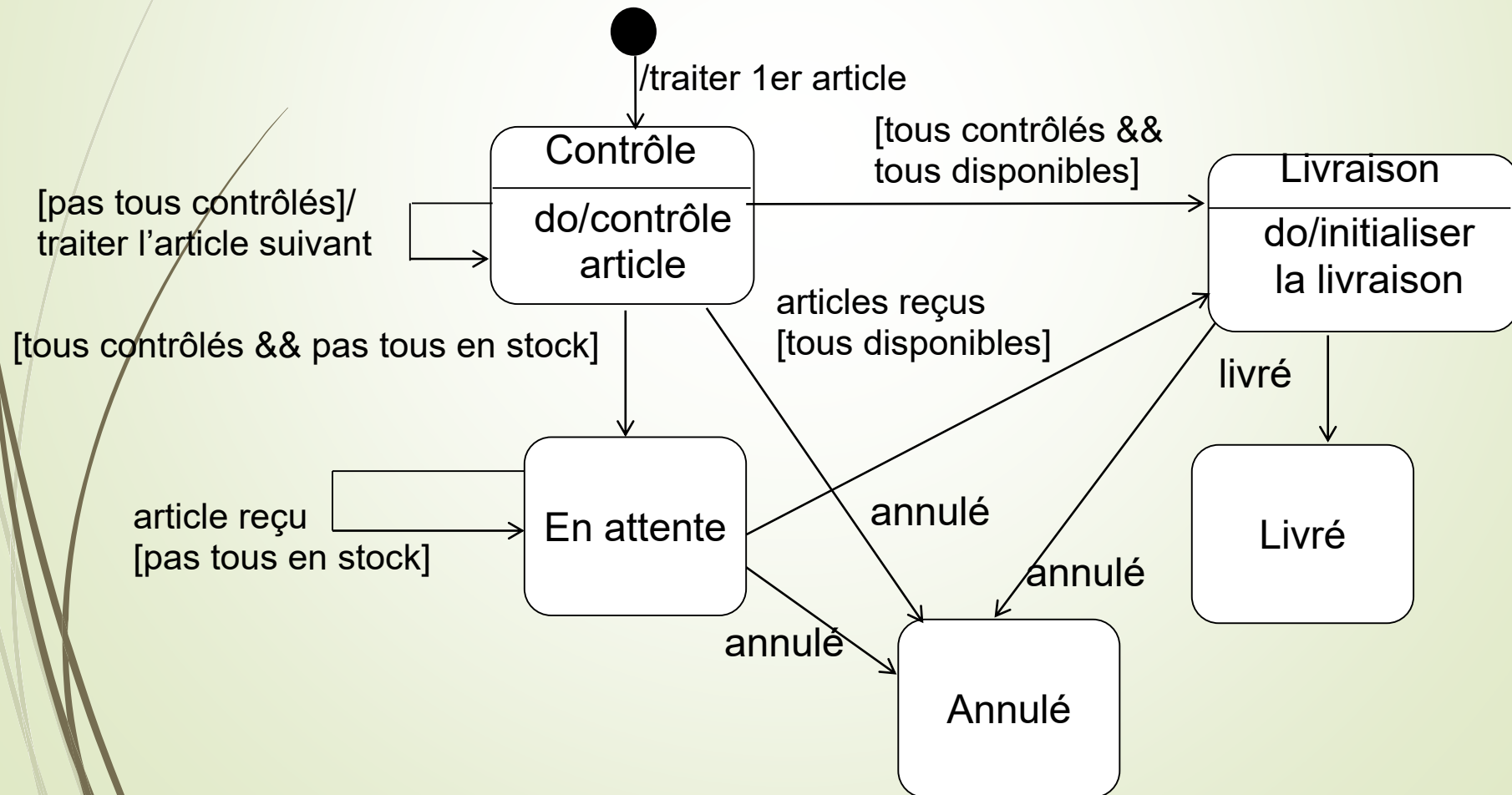


Diagramme d'états

Diagramme d'états : notion d'état composite

- Exemple d'une commande annulée avec super état
- Permet de factoriser la transition associée à l'événement Annuler et de définir 3 états principaux (Actif, Livré, Annulé)

Nom du super état

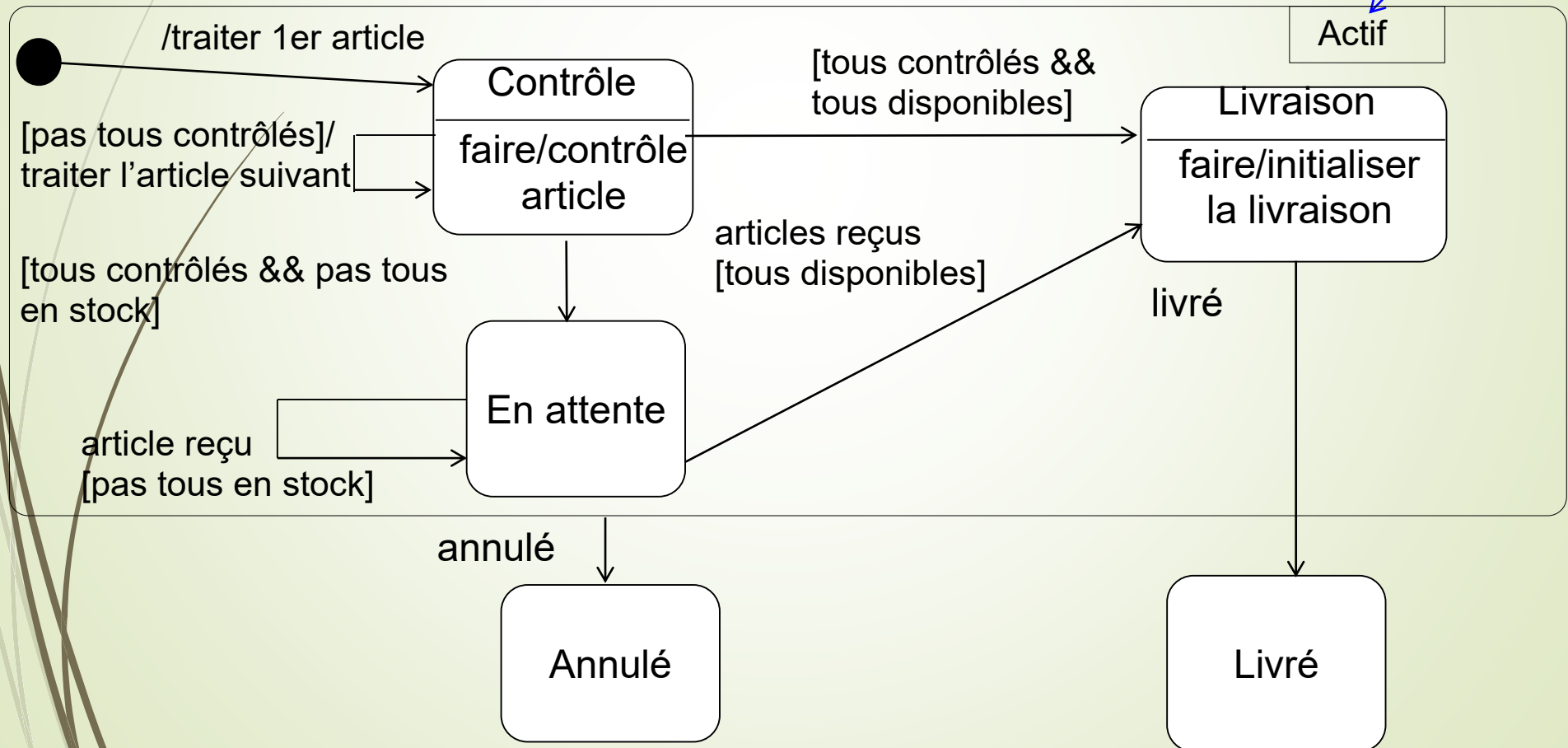


Diagramme d'états

Diagramme d'états concurrents

- Plusieurs sous-parties parallèles au sein d'un composite
- Possibilité d'ajouter des éléments de synchronisation entre les sous-parties

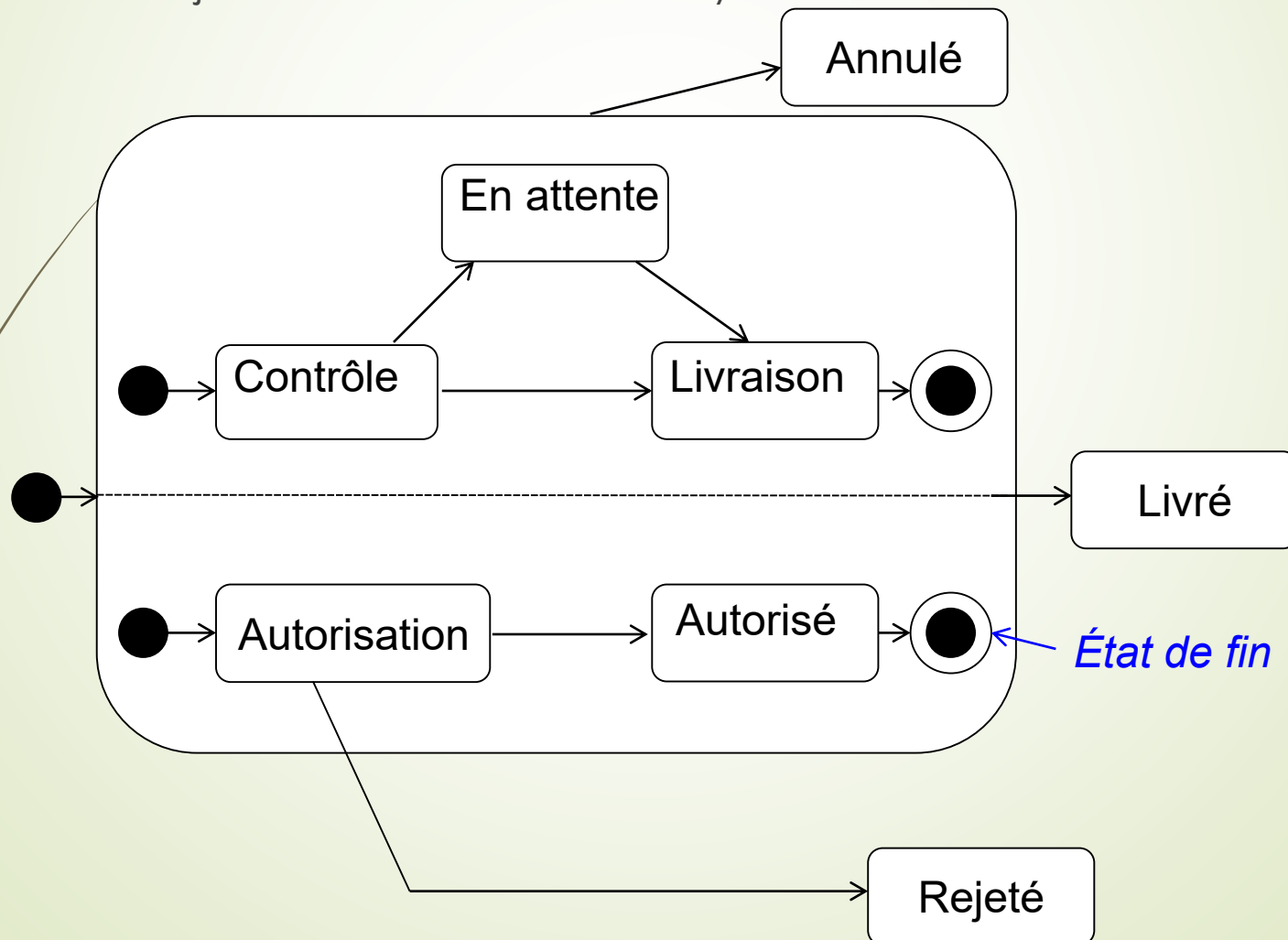
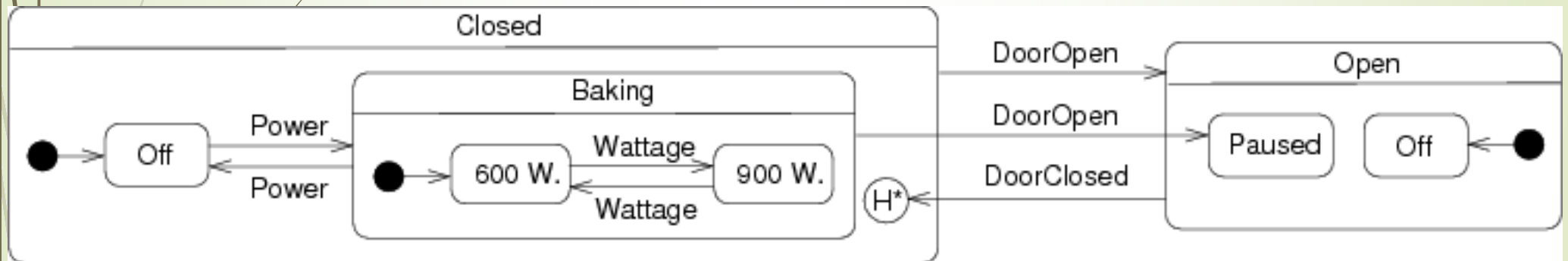


Diagramme d'états

États historiques

- Dans un état composite, permet de revenir dans l'état interne qui était celui qu'on a quitté en dernier
- Deep history (H^*) : si dernier état est un composite, réactive également son dernier état interne et ainsi de suite jusqu'au bout de la hiérarchie
- Shallow history (H) : ne réactive que le « premier » niveau (donc si dernier état est un composite, prend son état initial)



➤ Exemple

- Hiérarchie initiale d'états actifs : Closed / Baking / 900W
- Puis événements DoorOpen et DoorClosed
- Si deep history (comme sur le diag.) : retrouve Closed / Baking / 900W
- Si shallow history : Closed / Baking / 600W

Diagramme d'activités

Diagrammes d'activités

- A utiliser pour:
 - analyser un cas d'utilisation
 - comprendre un flot de données traversant plusieurs cas d'utilisation
- Description des comportements parallèles
 - Modélisation de flot de données (workflow)
 - Dérivé de diagrammes d'événements, de réseaux de Petri, de SDL
- Selon le niveau de modélisation, une activité correspond à
 - Conception : une tâche qui est exécutée soit par un humain ou par un ordinateur
 - Spécification/implémentation : une méthode ou le comportement d'une classe

Diagramme d'activités

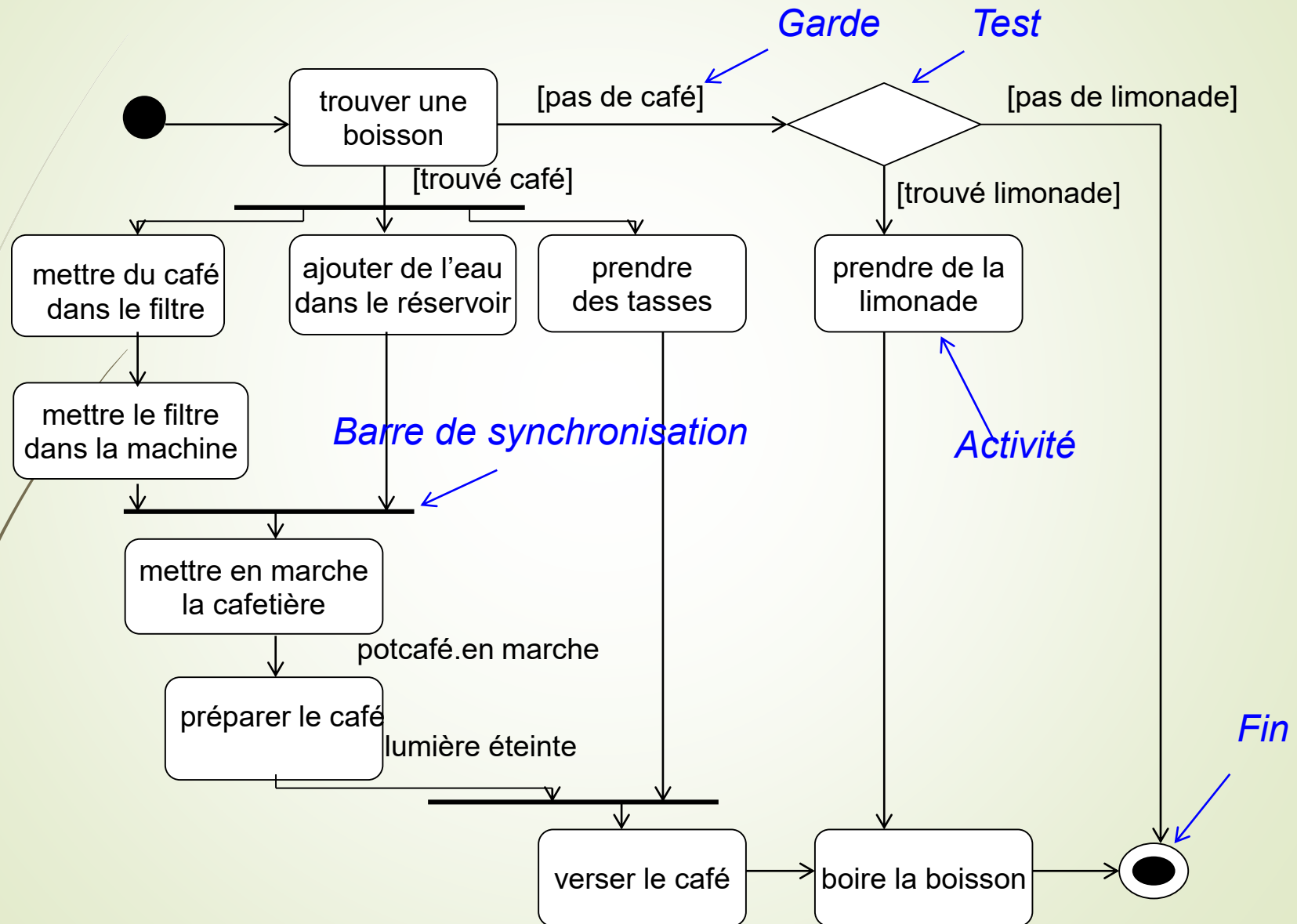


Diagramme d'activités

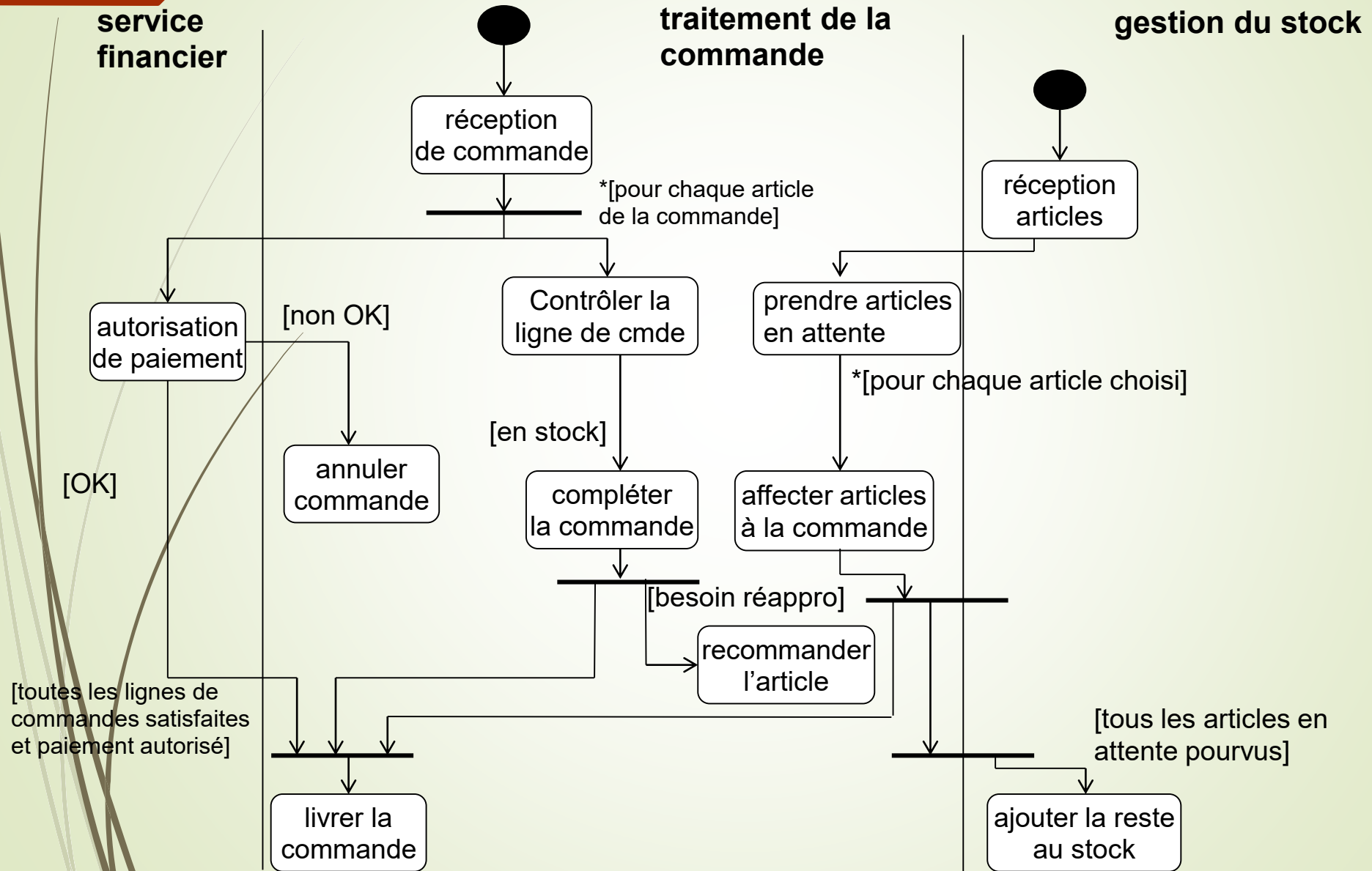


Diagramme de séquence

- Interaction entre objets
 - Chaque objet est représentée par une ligne verticale
 - Temps s'écoule de haut en bas
 - Précision des messages échangés entre les objets
 - Message = appel de méthode
- Permet de spécifier l'ordonnancement temporel des interactions entre les objets
 - Enchaînement / imbrication des appels de méthodes
- Nouveauté UML 2 : ajout de cadres pour définir des boucles, des alternatives ...
 - Mais peut vite devenir assez peu lisible en pratique

Diagramme de séquence

Diagramme de séquence

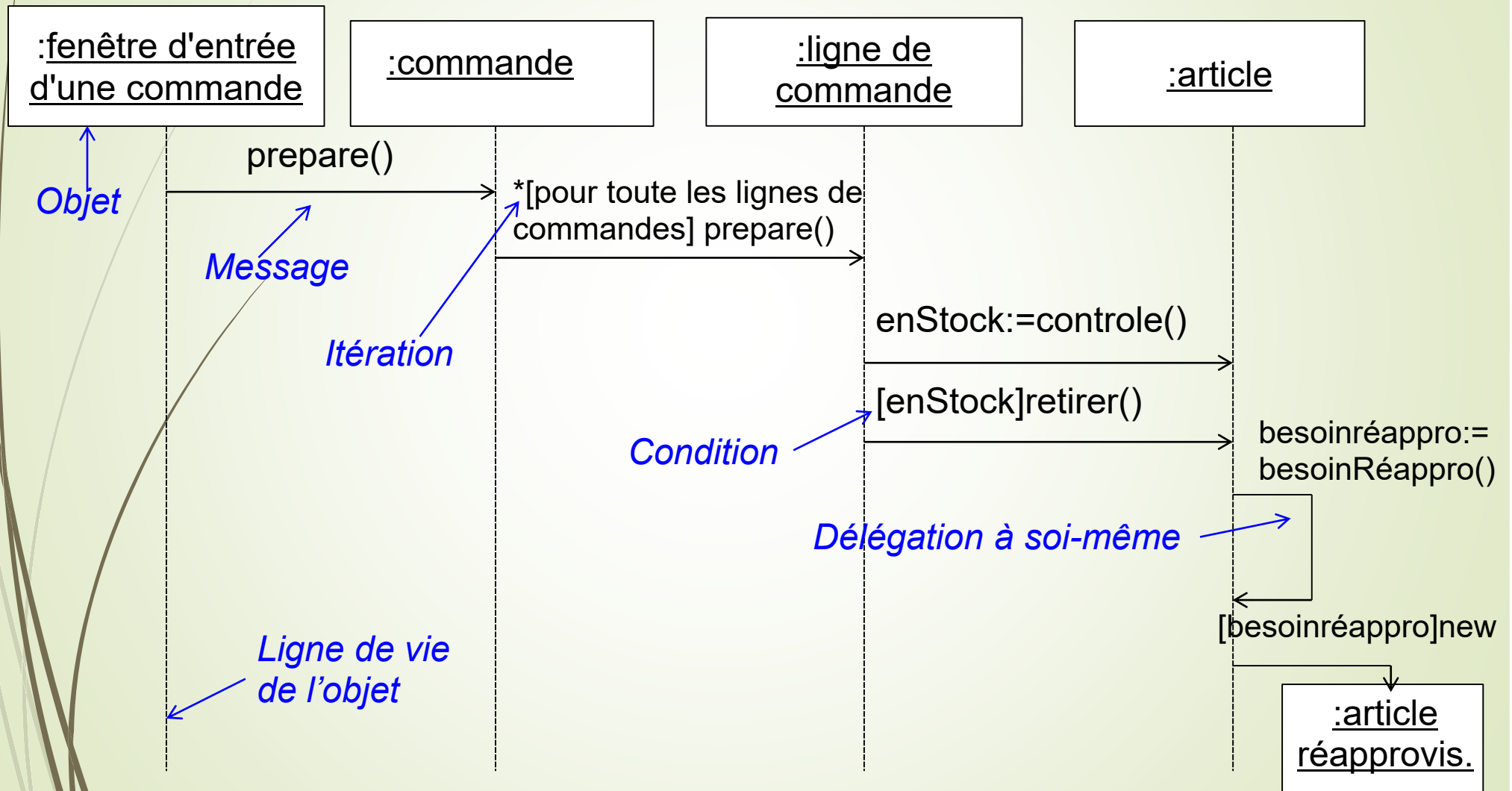


Diagramme de séquence

Diagramme de séquence (suite)

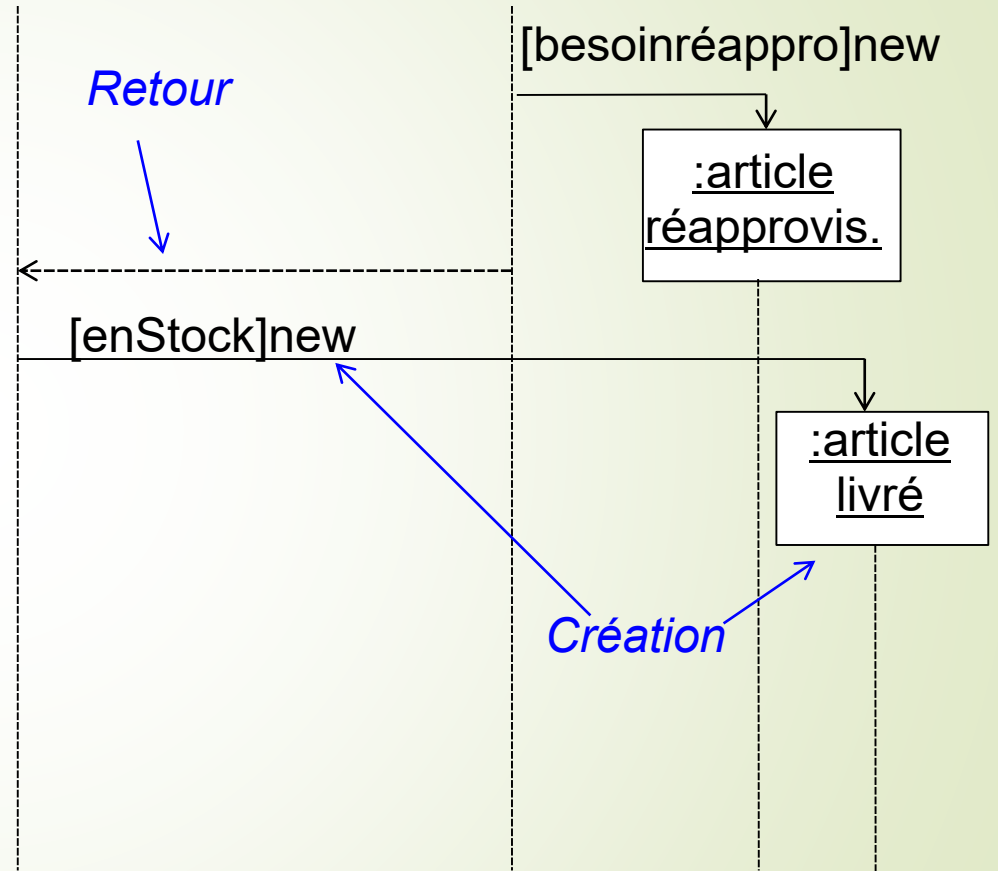
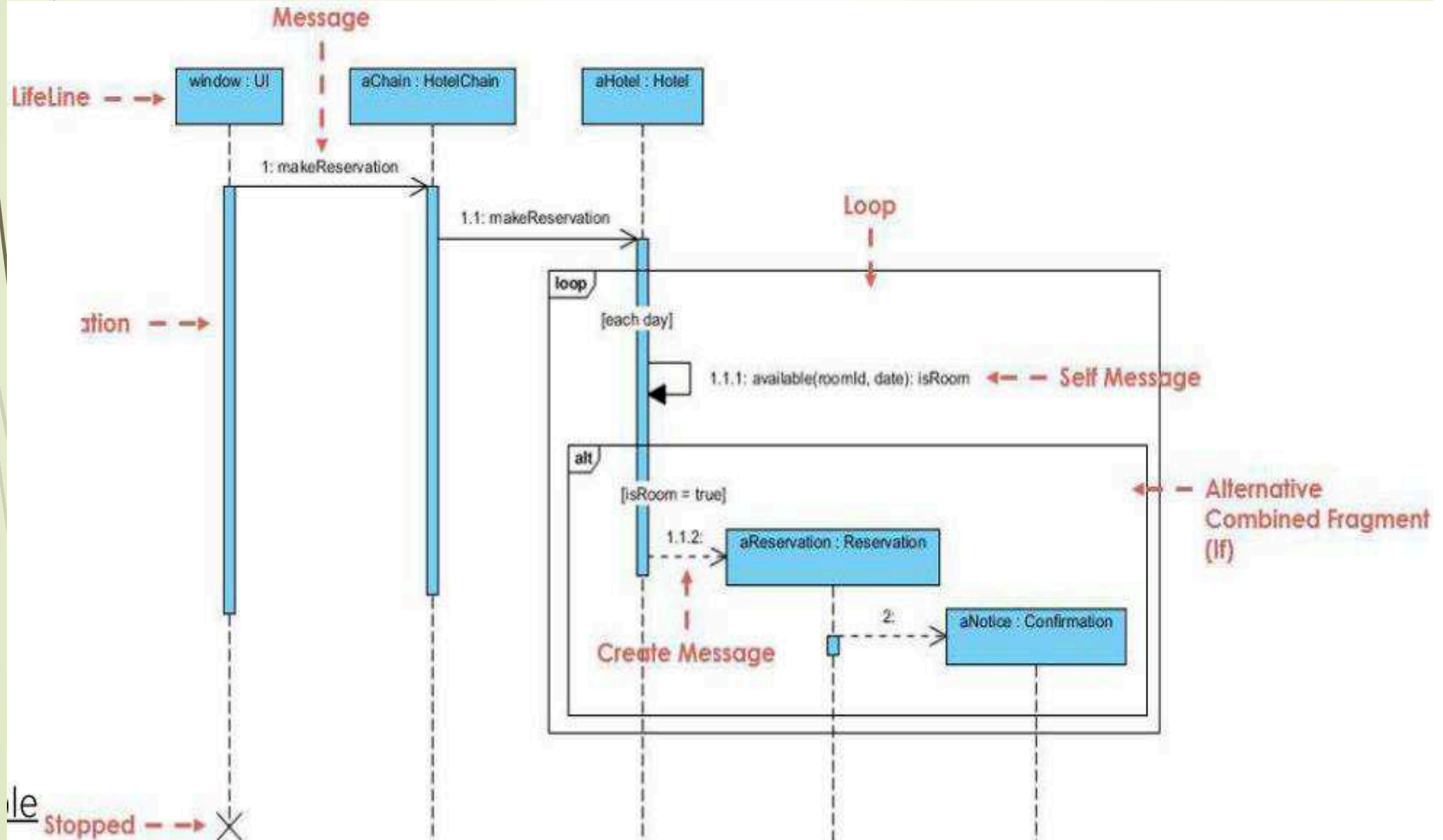


Diagramme de séquence

Cadre d'interaction

- Cadre qui englobe une partie du diagramme de séquence (un fragment) pour définir un fonctionnement non séquentiel
- Types de cadres
 - Alt
 - Alternative (if-then-else) entre deux parties selon une garde
 - Loop
 - Boucle
 - Opt
 - Partie optionnelle (if-then) selon une garde
 - Par
 - Deux parties en parallèle
 - Region
 - Partie en exécution mutuelle (processus / thread)

Diagramme de séquence



Diagrammes de collaboration


- 
- Diagramme de collaboration « équivalent » au diagramme de séquence
 - Met en avant la vue structurelle au lieu de temporelle
 - Notion de rôle : un élément a une fonction particulière
 - Deux niveaux / étapes
 - Définition du diagramme de collaboration qui représente une interaction
 - L'utilisation d'une collaboration pour montrer l'interaction d'éléments dans un diagramme de classes ou d'objets
 - Ces éléments sont liés à un rôle de la collaboration

Diagramme de communication


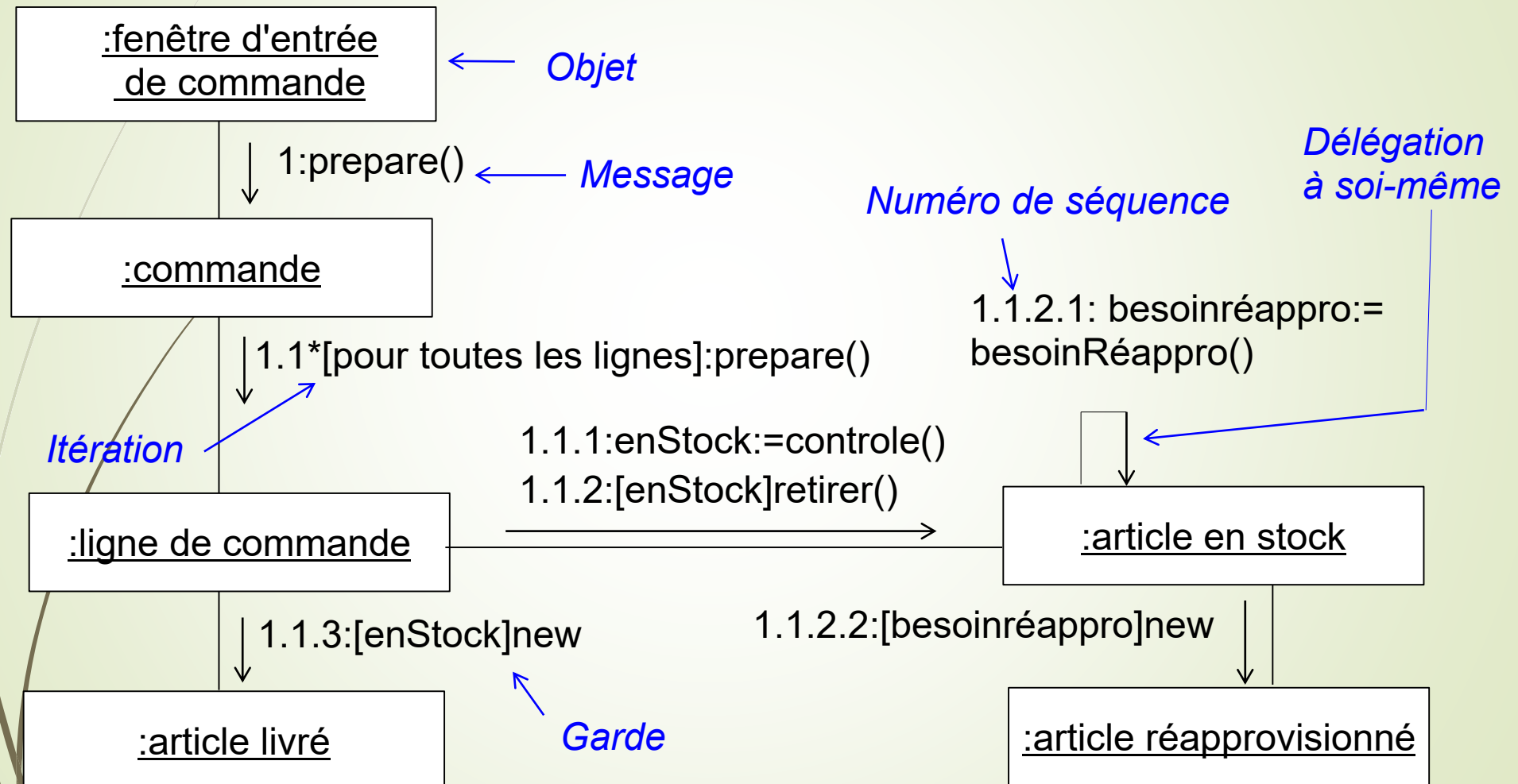
- 
- Diagramme de communication
 - Nouveau nom du diagramme de collaboration en UML 2
 - Diagramme de collaboration au niveau instance = diagramme de communication
 - Diagramme de séquence vs collaboration
 - Le diagramme de séquence n'existe qu'au niveau instance

Diagramme de collaboration (instance)

Diagramme de collaboration au niveau instance



Diagrammes dynamiques – conclusion


- 
- Diagrammes d'interaction (séquence ou collaboration)
 - Pour comprendre la coopération entre les objets
 - Diagrammes d'états
 - Pour comprendre le comportement interne d'un objet
 - Diagrammes d'activités
 - Pour analyser un cas d'utilisation
 - Pour comprendre un flot de données traversant plusieurs cas d'utilisation
 - Pour comprendre les applications multi-activités

Diagramme de vue globale d'interaction

- Sorte de « mélange » d'un diagramme de séquence et d'un diagramme d'activité

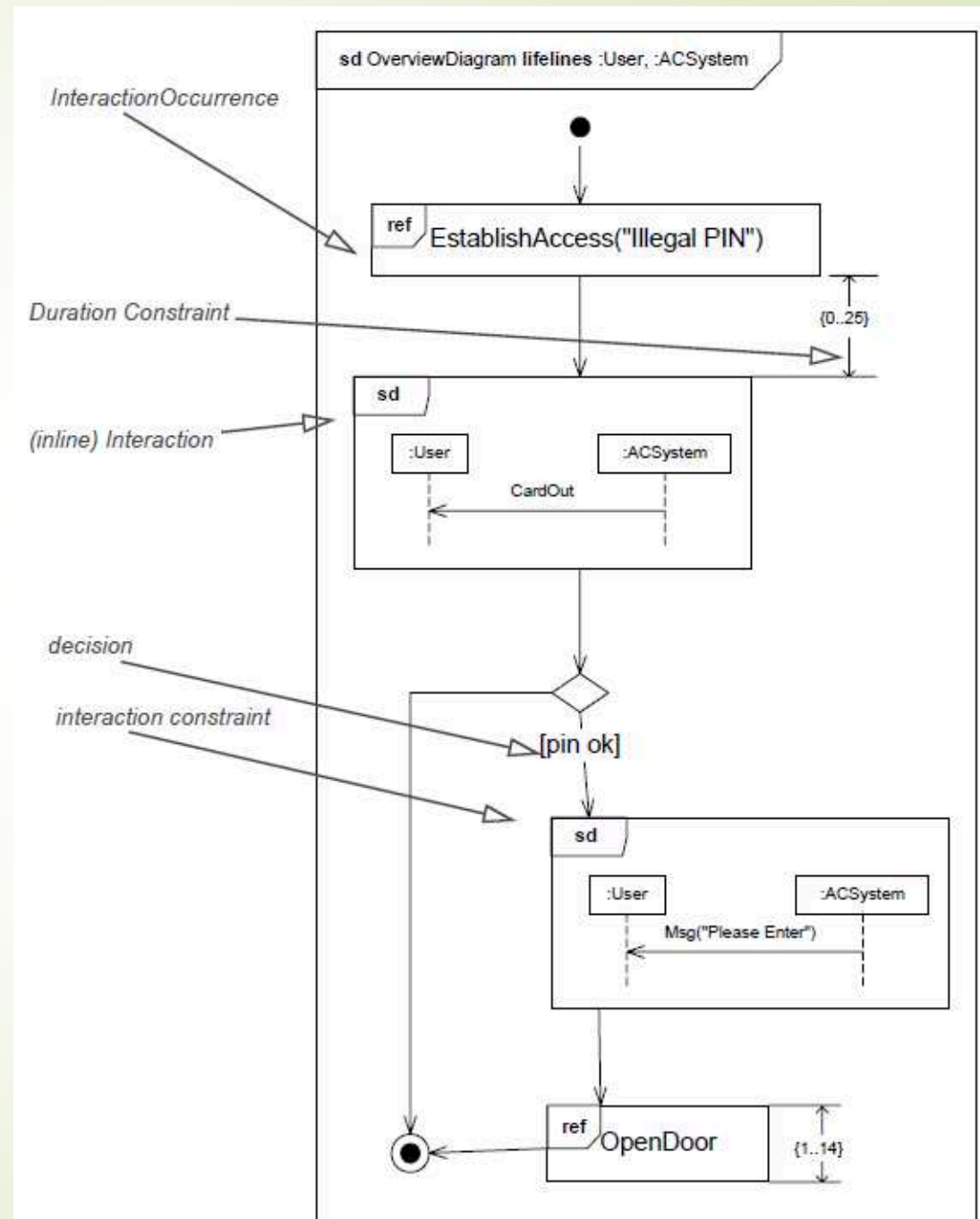
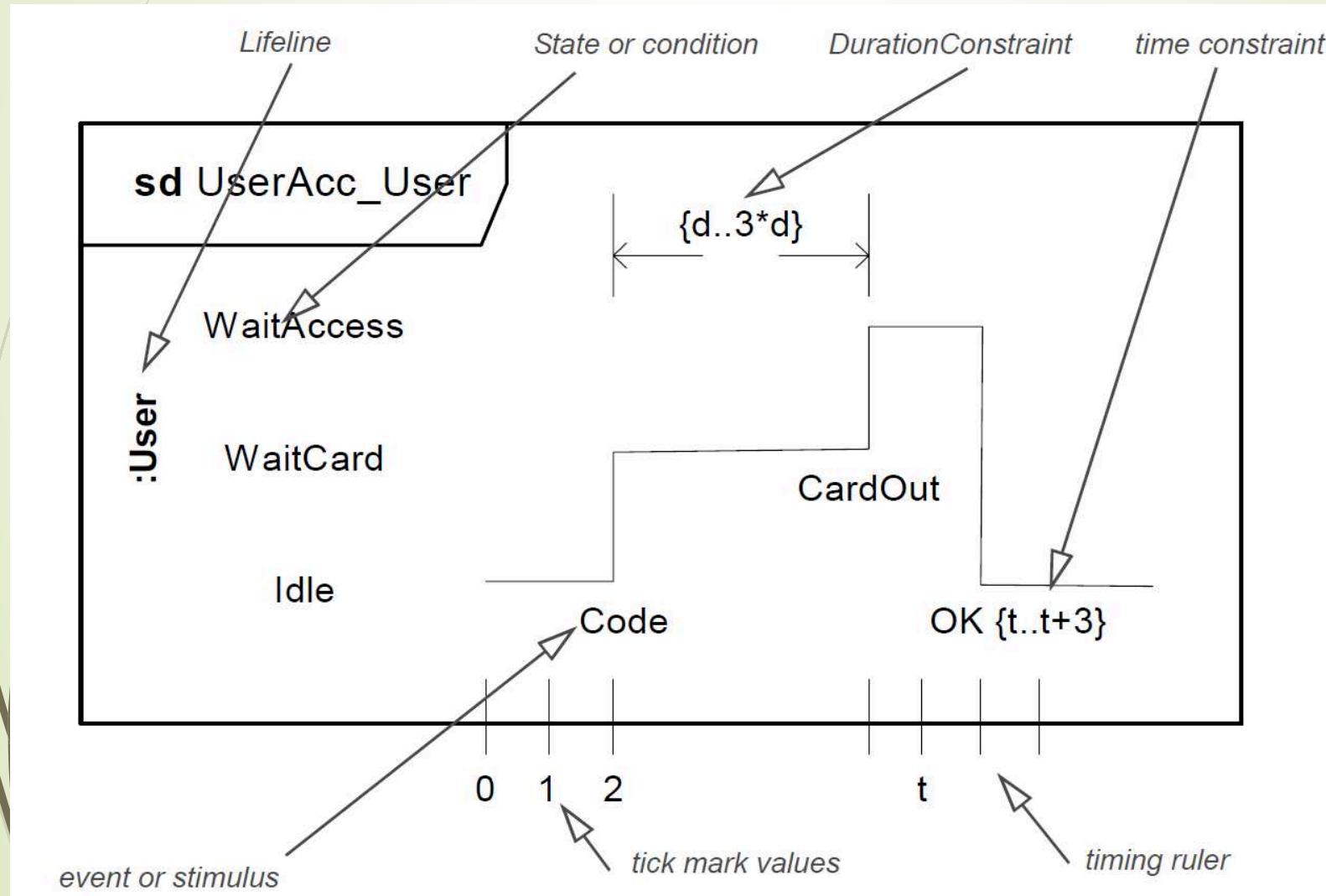


Diagramme de temps

- Evolution de l'état du système selon un point de vue principalement temporel



Plan

- ❖ Diagrammes fonctionnels
- ❖ Diagrammes statiques
- ❖ Diagrammes dynamiques
- ❖ *Diagrammes d'implémentation*
 - ❖ *De paquetages*
 - ❖ *De déploiement*

Diagrammes d'implémentation


- 
- Mise en place de l'application sur un environnement
 - Diagramme de paquetages
 - Description de l'organisation du code des applications
 - Utile au programmeur
 - Diagramme de déploiement
 - Description du déploiement sur un réseau
 - Aspects liés à la topologie, à l'intégration des systèmes et aux communications

Diagramme de paquetages


- 
- Regrouper les classes dans des “packages”
 - Disposer d'heuristiques pour regrouper les classes
 - Heuristique la plus utilisée : la dépendance entre les classes
 - Une dépendance existe entre 2 éléments si le changement de définition d'un élément peut modifier un changement dans l'autre élément
 - Dépendances entre classes
 - Envoi d'un message (appel de méthode)
 - Une classe fait partie des données d'une autre classe
 - Une classe mentionne une autre classe comme un paramètre d'une opération
 - Idéalement, seules les modifications de l'interface de la classe affectent les autres classes

Diagramme de paquetages

- Exemple de diagramme de paquetages
 - Note : les classes contenues dans les packages ne sont pas représentées ici

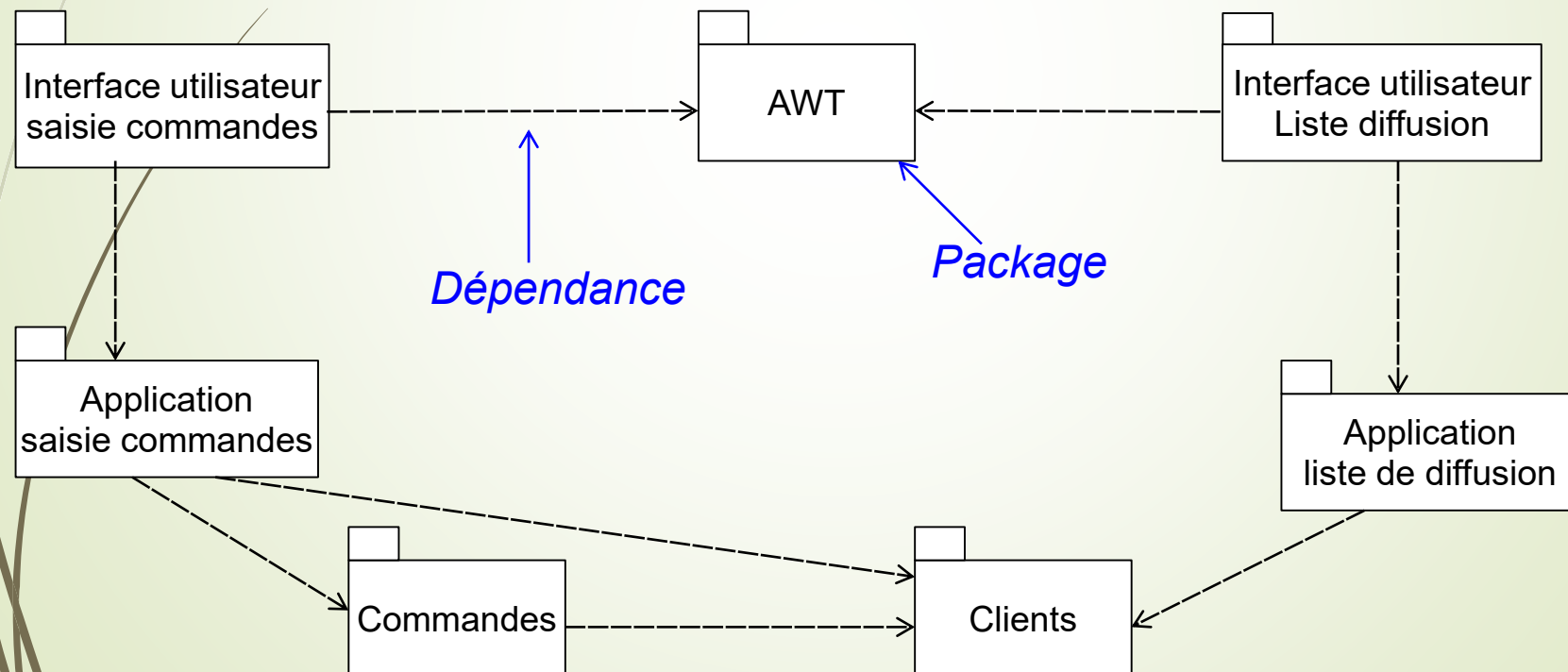
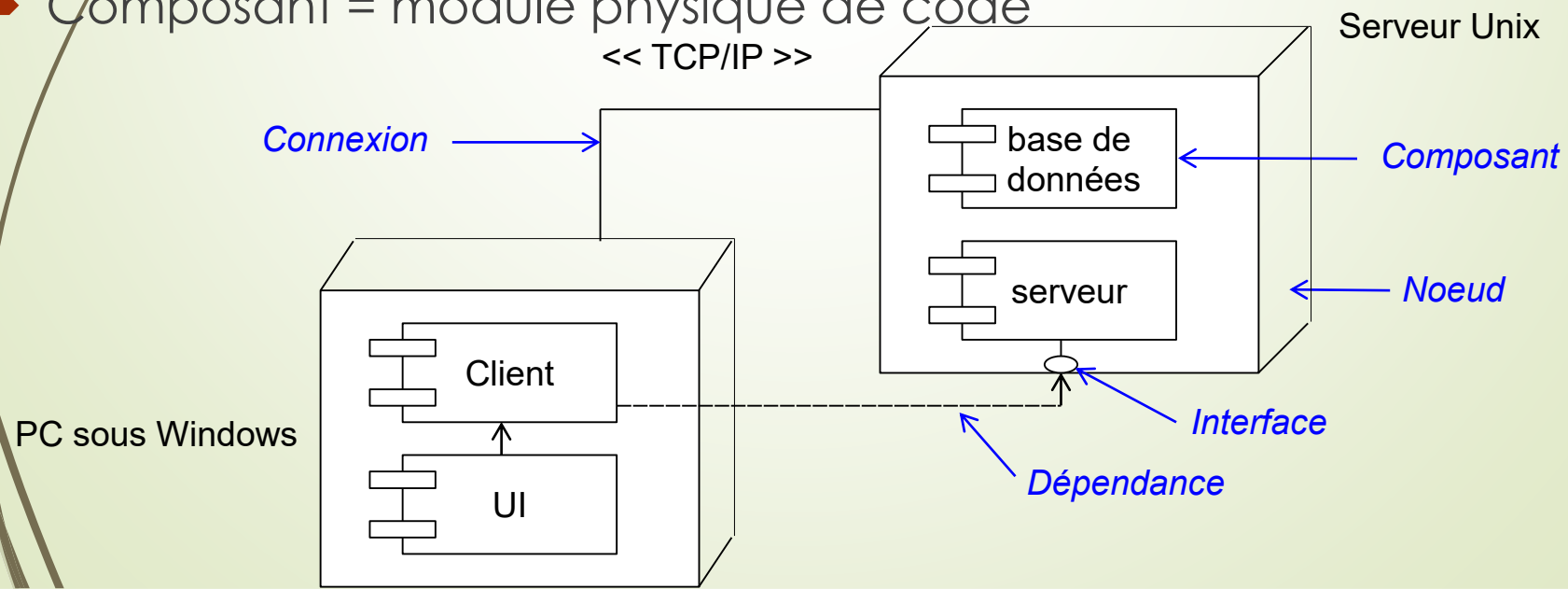


Diagramme de déploiement

- Diagramme de déploiement
- Relation entre le logiciel et le matériel
- Placement des composants et objets dans le système réparti
- Nœud = unité informatique (périphérique, capteur, mainframe, PC,...)
- Connexion
- Composant = module physique de code



Conclusion sur UML

Avantages d'UML

- Un certain consensus autour de l'utilisation d'UML : standard de fait dans l'industrie
- Notation avec une syntaxe très riche
- Intégration dans des ateliers de génie logiciel avec production de squelettes de codes et autres transformations automatiques des modèles
- Langage de contraintes OCL pour spécifications précises à utiliser en complément

➤ Inconvénients d'UML

- Notation majoritairement graphique pouvant se révéler insuffisante ou trop chargée d'un point de vue expressivité
- Sémantique floue ou mal définie pour certains types de diagrammes
- Lien parfois difficile entre les vues et diagrammes d'une même application