UML
Aspects dynamiques
Compléments pour la conception

Emmanuel Pichon
2013

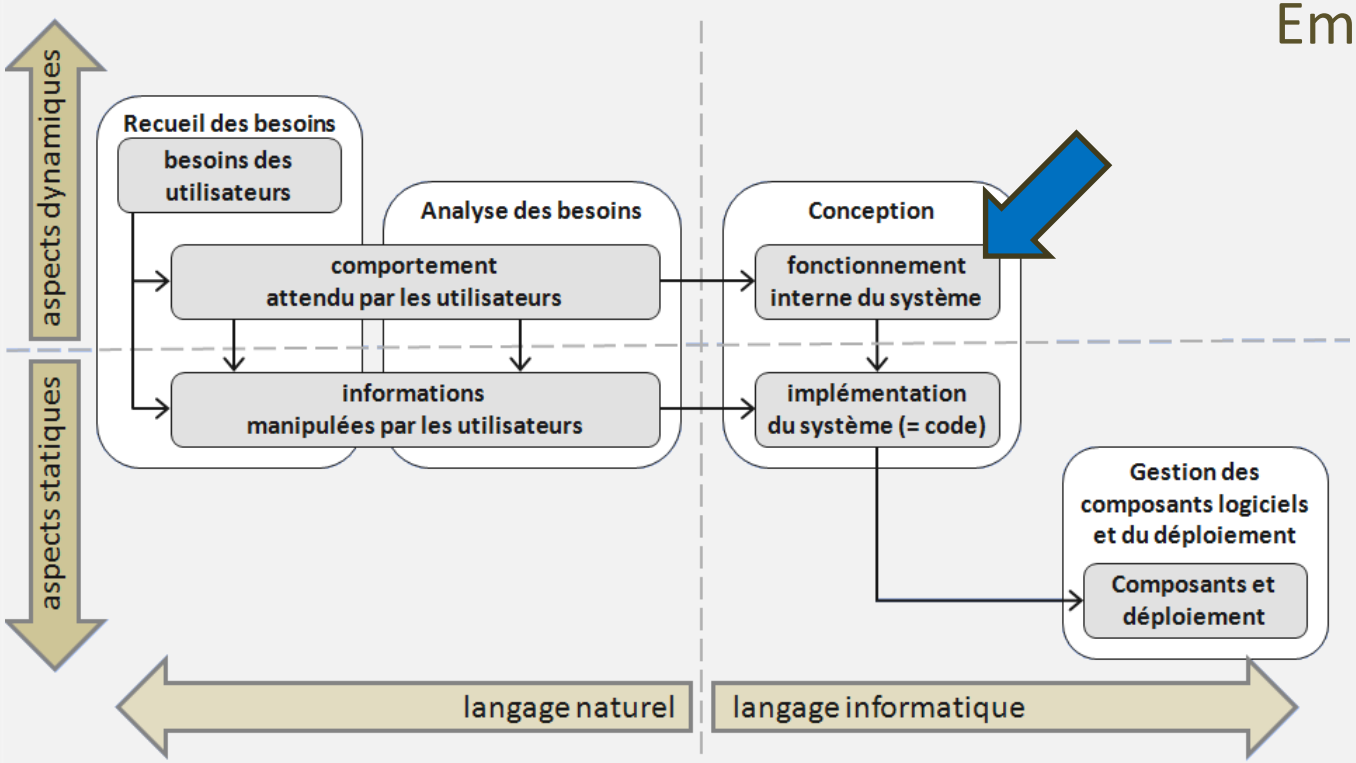


Diagramme de séquence
(sequence diagram)

Types de messages

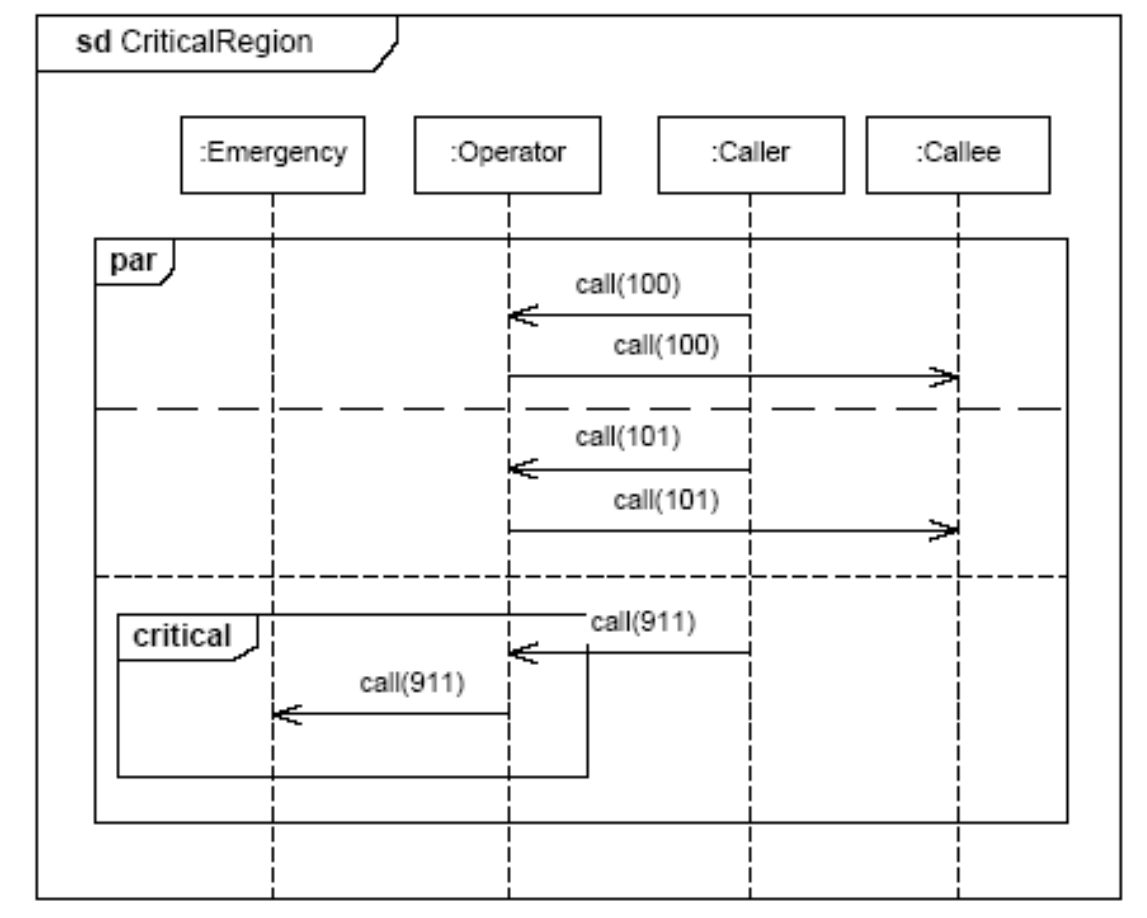
- Message synchrone (par défaut des les langages objet)
  - Sens : l'émetteur attend le retour du message
  - Notation UML : une flèche pleine 
  - Usage : fonctionnement par défaut des langages objet

- Message asynchrone
  - Sens : l'émetteur n'attend aucun retour direct du message 
  - Notation UML : une flèche ouverte
  - Usage : par exemple, envoi de fichiers entre systèmes
  - Peut apparaître en biais pour indiquer qu'il prend un certain temps pour circuler

Compléments sur les fragments

- Spécification du parallélisme
  - **par** : permet la représentation de plusieurs séquences qui vont s'exécuter dans un ordre quelconque
  - **seq** (séquencement faible) : permet de garantir la conservation de l'ordre des événements relativement à une ligne de vie
  - **strict** : permet de garantir l'ordre d'exécution au premier niveau, sans contrainte sur les fragments inclus
  - **critical** (région critique) : la région est considérée comme atomique et ne peux être interrompue
  - **coregion** : définit une zone sur une ligne de vie à l'intérieur de laquelle la réception des messages se fait dans un ordre quelconque

Compléments sur les fragments
Illustration



Compléments sur les fragments

- Autres types de fragments possibles
  - **break** : le fragment se substitue à la suite du bloc englobant
  - **neg/assert**
    - **neg** : le fragment spécifie une séquence interdite
    - **assert** : le fragment spécifie la seule séquence valide
  - **consider/ignore {message1, message2, …}**
    - **consider** : les messages pour lesquels la séquence est décrite (le traitement des autres messages est quelconque)
    - **ignore** : les messages ne sont pas traités dans le fragment
    - Ils peuvent être combinés avec les autres types de fragments, par exemple : assert consider {message1, …}

Diagrammes de séquence : contraintes temporelles

- { …} spécifie des contraintes (temporelles dans ce cas)
  - De durée entre la réception ou l'émission de message sur une même ligne de vie
  - De temps entre un événement particulier et la réception d'un message

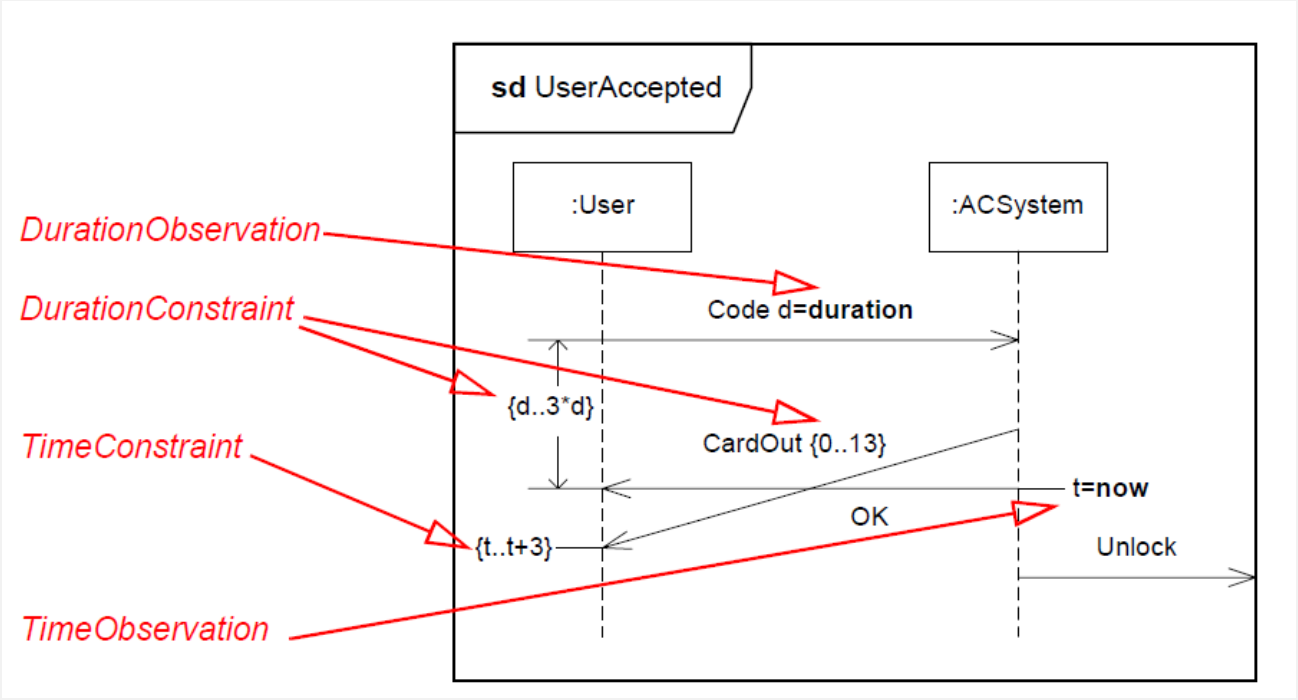


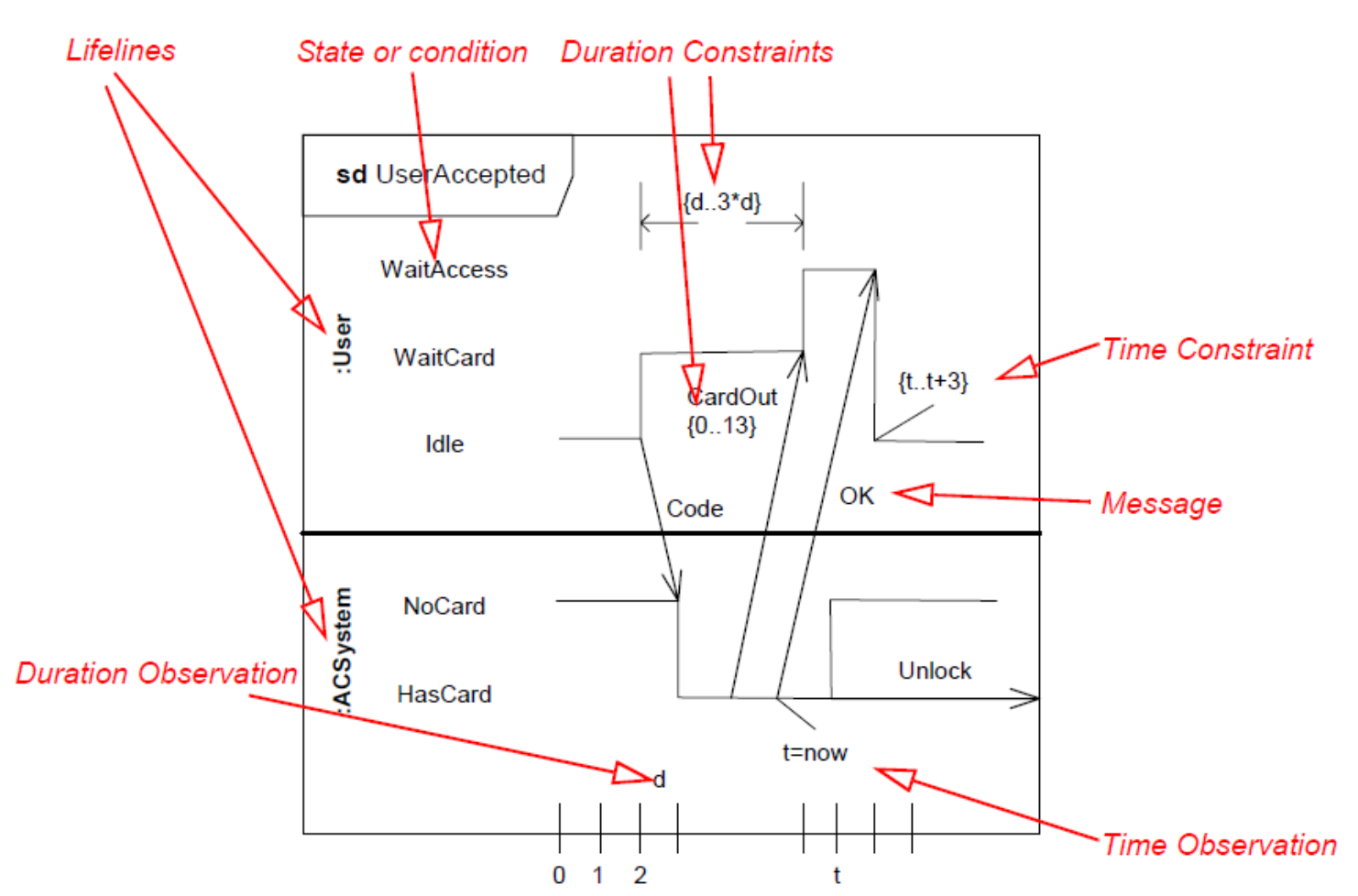
Diagramme de temps
(timing diagram)

Diagramme de temps (*timing diagram*)

- Objectif
  - Similaire à celui des diagrammes de séquence
  - En plus, possibilité d'indiquer des états ou des conditions et leur évolution dans le temps

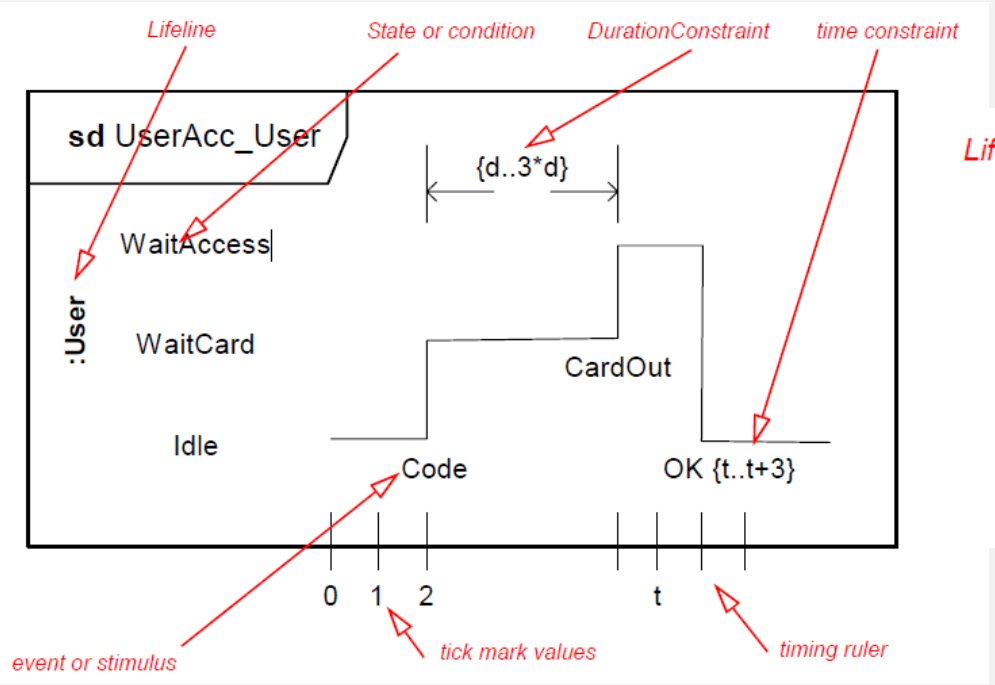
- Notation UML
  - Présentation dérivée du diagramme de séquence avec les spécificités suivantes
    - Présentation à l'horizontal
    - Les lignes de vie sont représentées par des boîtes contenant un enchaînement d'états ou de conditions

Exemple de diagramme de temps



Deux niveaux de détails possibles

- Présentation détaillée



- Présentation compacte

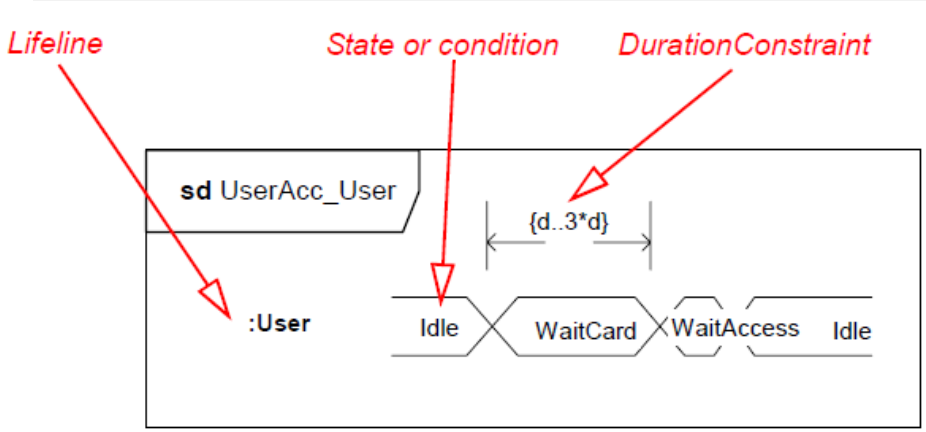


Diagramme d’activité
(activity diagram)

Compléments pour la conception

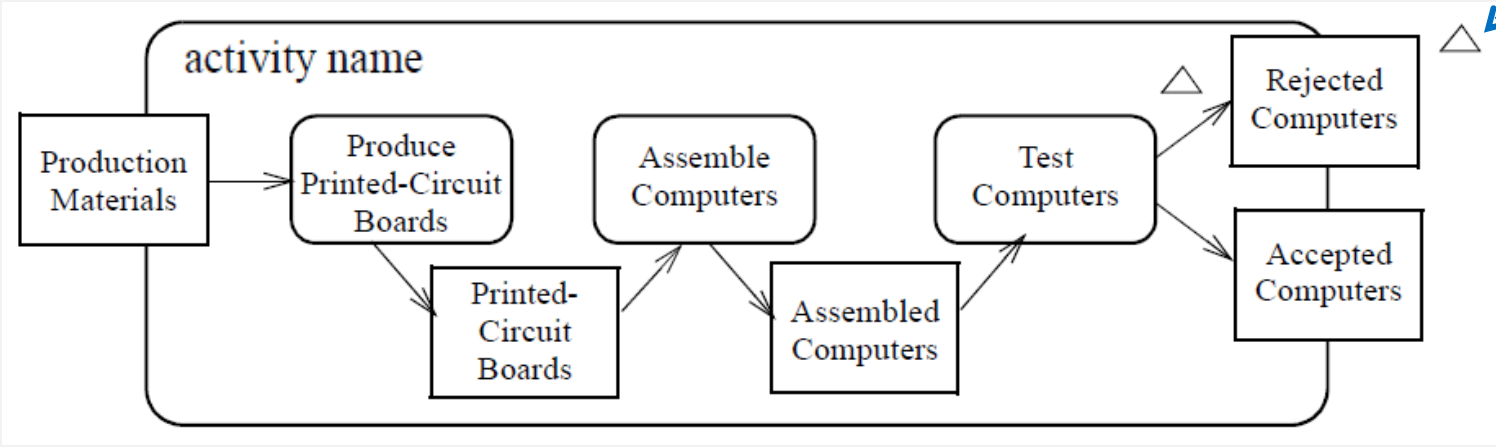
- Distinction entre activité et action
  - Activité = unité de comportement décomposable en actions
  - Action = unité de comportement non décomposable

- Les actions peuvent
  - Déclencher des opérations sur des objets (≈ message synchrone)
  - Envoyer des signaux (≈ message asynchrone)
  - Appeler d'autres activités

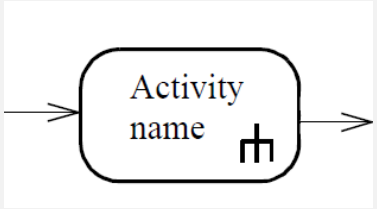
- Notion de contexte
  - Une activité peut être définie dans une classe
  - Dans ce cas, les actions ont une portée limitée à la classe (et aux classes imbriquées ou liées par une composition)

Activités

- Les objets peuvent être utilisés pour identifier les paramètres d'entrée ou de sortie d'une activité

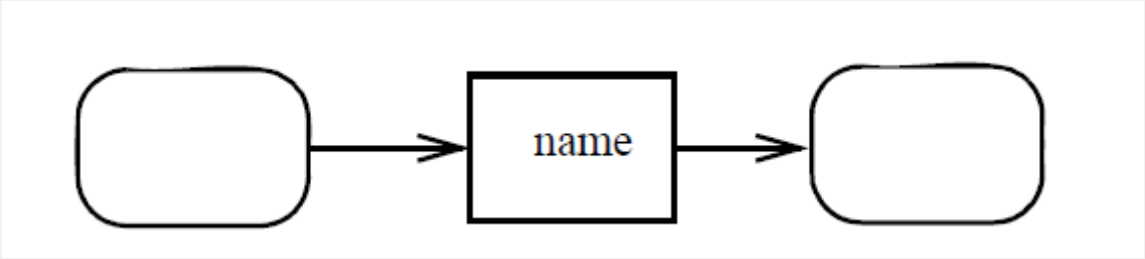


- Dans un autre diagramme, on peut faire référence à cette activité sans faire apparaitre l'intérieur de celle-ci (le « râteau » indique un contenu non décrit dans le diagramme)

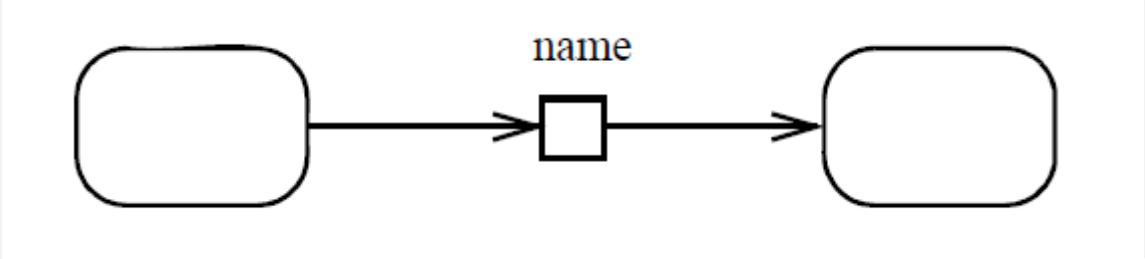


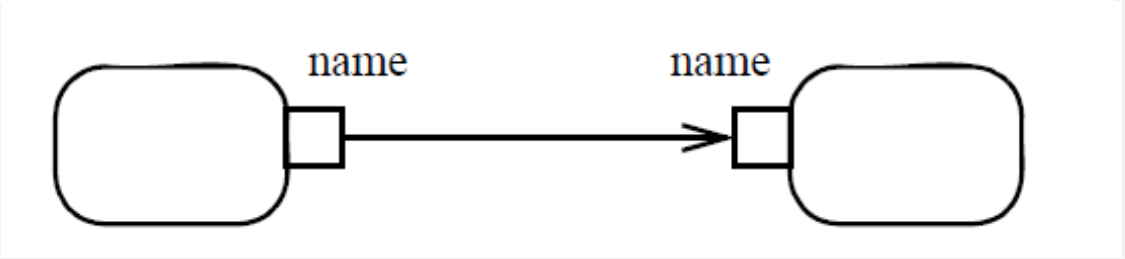
Trois notations équivalentes pour les flots d'objets

- Notation déjà présentée en analyse



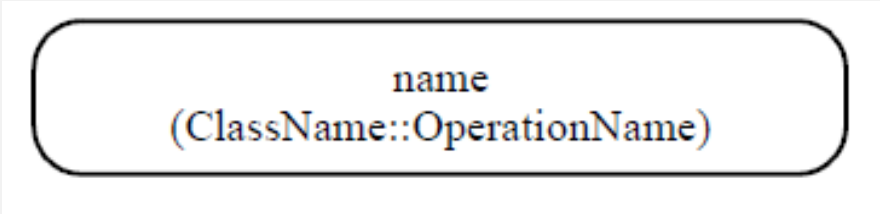
- Deux autres possibilités



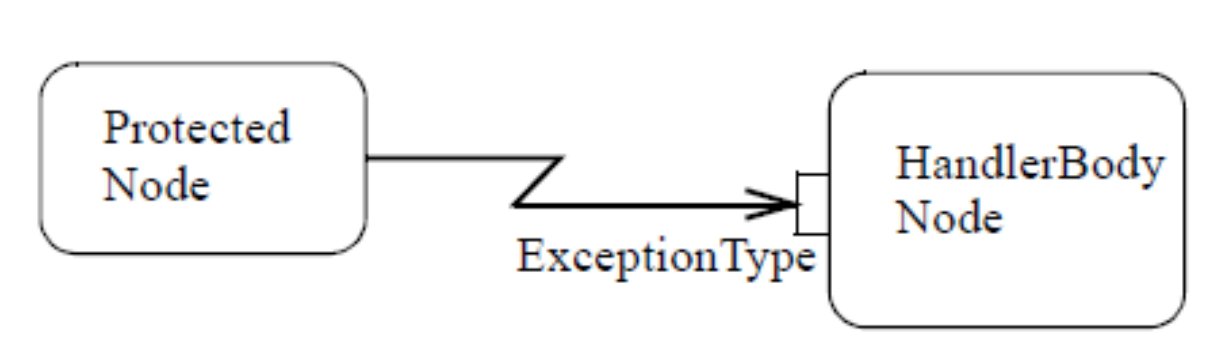


Notations complémentaires

- Action invoquant une opération définie dans une classe
  (si le nom de l'action ne correspond pas à celui de l'opération)



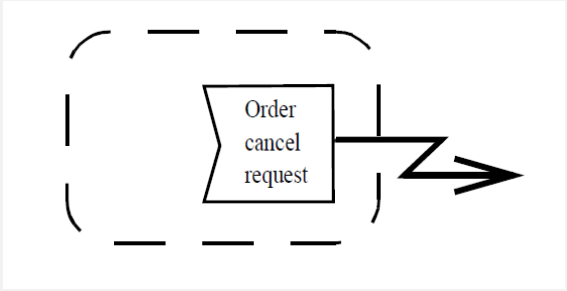
- Gestion d'une exception



Région

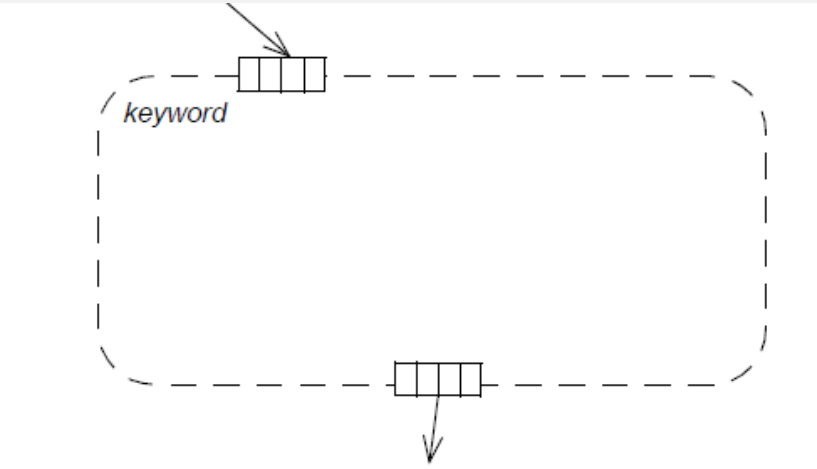
- Sens
  - Regroupement d'actions
- Notation UML

- Exemple : région interruptible
  - Région = regroupement d'actions
  - Exemple : réception d'un événement

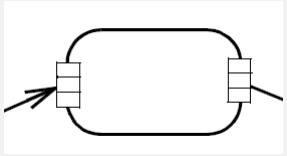


Région d'expansion *(expansion region)*

- Sens
  - Regroupement d'actions traitant des listes d'éléments
- Notation UML



- Notation possible si une seule action dans la région



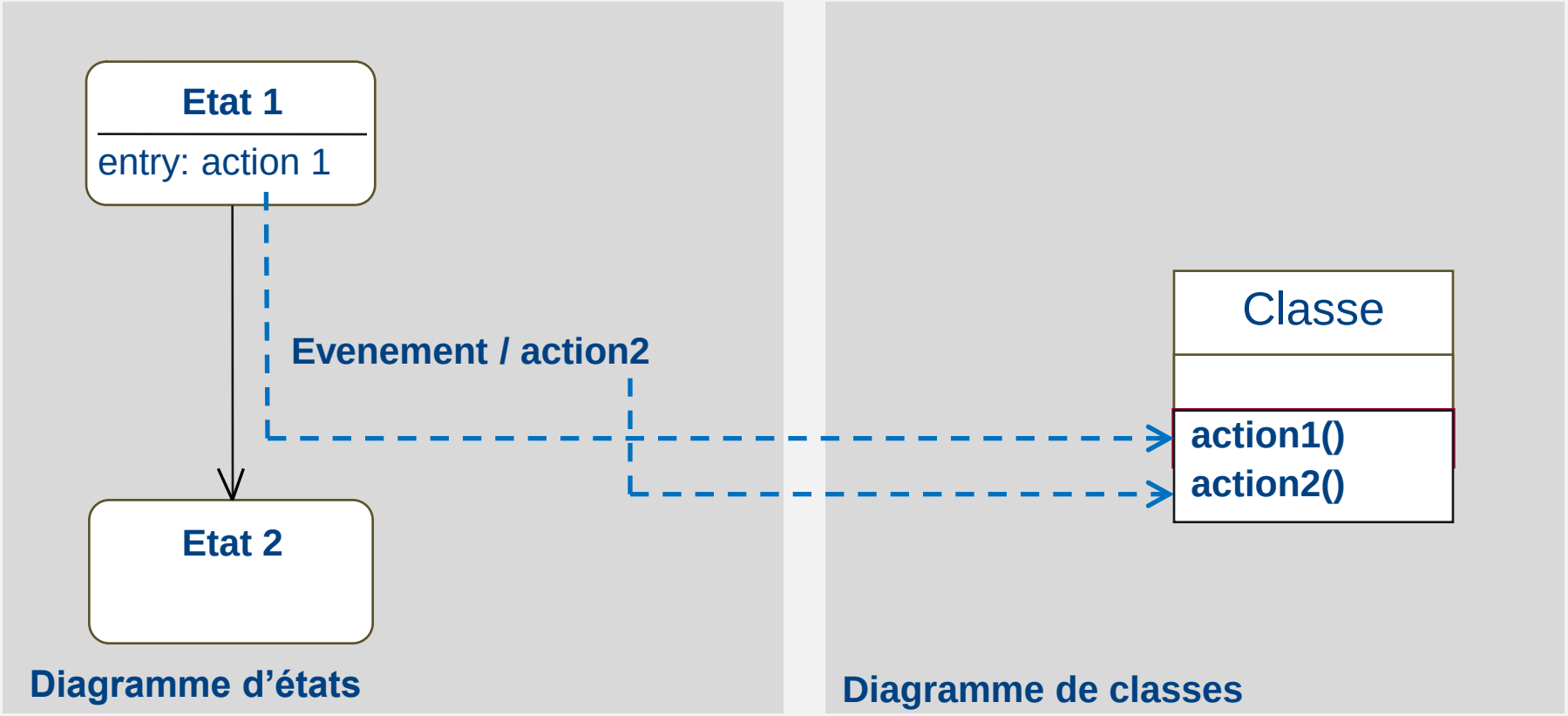
Région d'expansion *(expansion region)*

- Un mot-clé permet d'indiquer le mode de traitement
  - iterative (valeur par défaut)
    - Chaque exécution traite un élément en entrée
    - Les exécutions sont exécutées une par une, dans l'ordre de la liste en entrée
  - parallel
    - Chaque exécution traite un élément en entrée
    - Les exécutions peuvent être exécutées simultanément
  - stream
    - Une exécution unique permet de traiter un flux d'éléments en entrée, dans l'ordre de la liste en entrée

Diagramme d’états
(state machine diagram)

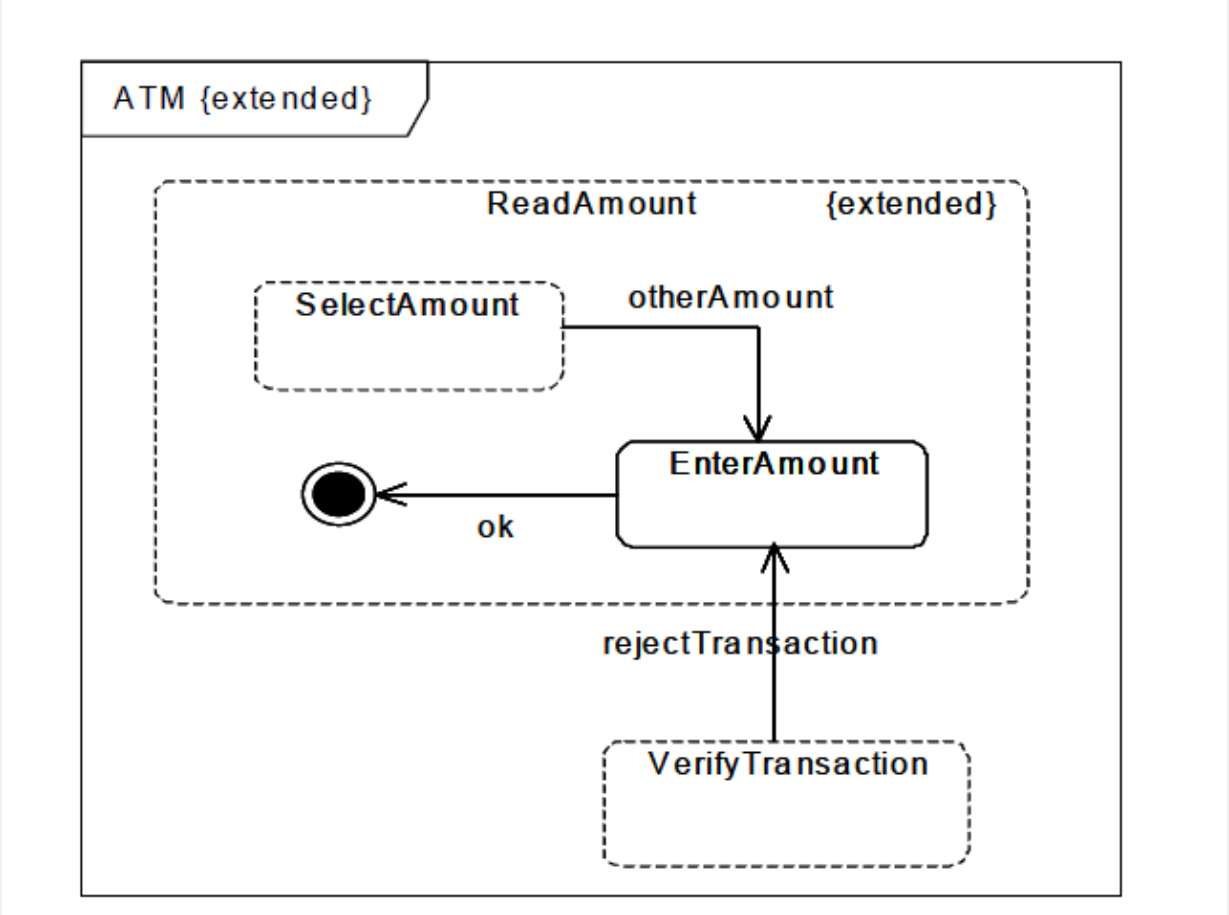
Compléments pour la conception

- Un diagramme d'état peut être défini dans une classe
- Une action peut déclencher l'exécution d'une opération



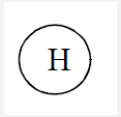
Héritage

- Les états hérités apparaissent en pointillés

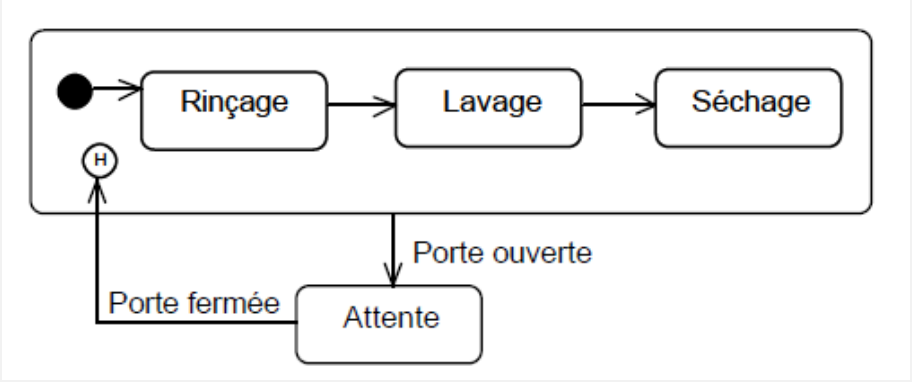


États composites, sous-états, H et H*

- Les états peuvent contenir des sous-états
- Pseudo-états H et H* = dernier sous-état actif dans un état
  - H est limité aux sous-états directement défini dans l'état (premier niveau de décomposition)
- Notation UML

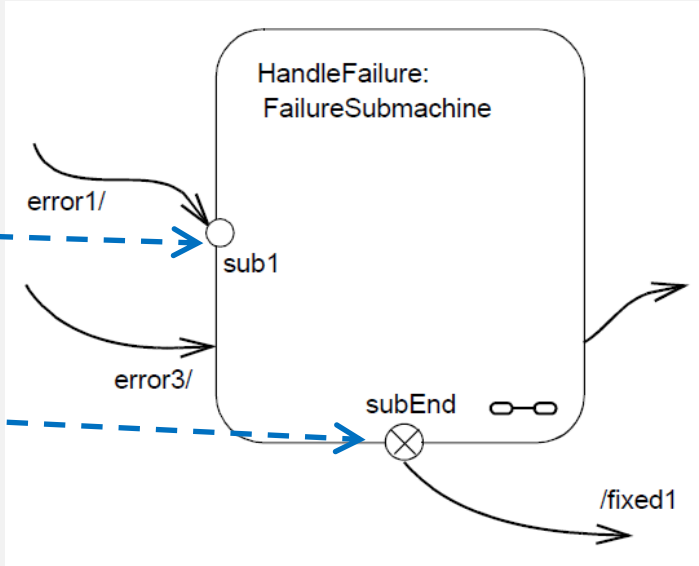


- Exemple



État composite

- On peut définir
  - Des points d'entrée
  - Des points de sortie



- Le symbole suivant indique que le contenu de l'état composite n'est pas visible dans ce diagramme

Exemple de point de sortie

- Le point de sortie fait le lien avec une prise en charge spécifique