# *Diagramme de Cas d'Utilisation*

**Objectif du diagramme** : description de **haut niveau** des **fonctionnalités du système**.

**Se concentrer sur ce que le système doit faire**
~~Comment le système pourra le faire~~ **NON**

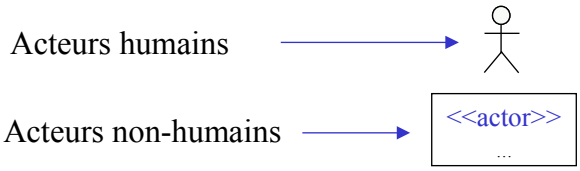
**But de ce diagramme** :
- Effectuer une bonne délimitation du système (*les acteurs représentent cette frontière*)
- Améliorer la compréhension de son fonctionnement

**Cas d'utilisation** :
- Fonction essentielle du système
- S'occupe d'un objectif élémentaire de l'utilisateur

**Démarche :**
**1- Identifier les acteurs**
**2- Identifier les CU en se posant la question pour chaque acteur :**
**Quelles sont ses intentions métier ?**
**3- Ajouter les relations entre CU**
**4- Structurer les CU en paquetages**

**Acteur** : **rôle** joué par une entité externe (*humain, logiciel, matériel, robot et automate*) qui interagit **directement** avec le système.
Vérifier que les acteurs se trouvent bien **à l'extérieur** du système.

**Acteur principal** : pour qui le CU produit la plus-value métier. Souvent déclencheur du CU.
**Acteur secondaire** : sollicité pour des informations complémentaires.




**Cas d'utilisation** : verbe à l'infinitif + complément.
Point de vue de l'acteur et non pas celui du système.

## Pas de notion temporelle dans un diagramme de CU

**Cas d'utilisation inclus**
Un CU inclus sert à **factoriser des enchaînements**, ce qui évite de les décrire plusieurs fois.
Un CU inclus peut intervenir à tout moment et pas uniquement au début d'un CU.
Incorporation **obligatoire**.

## Inclusion : pas de décomposition fonctionnelle

## Cas d'utilisation étendu

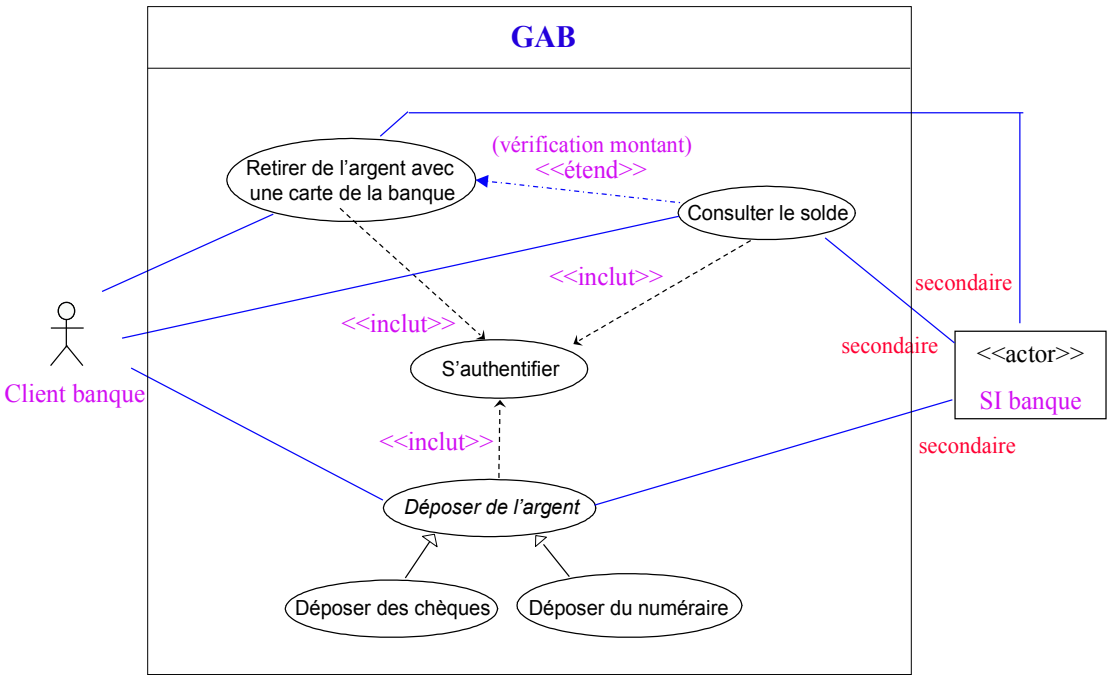
Incorporation **optionnelle**.

## Cas d'utilisation spécialisé

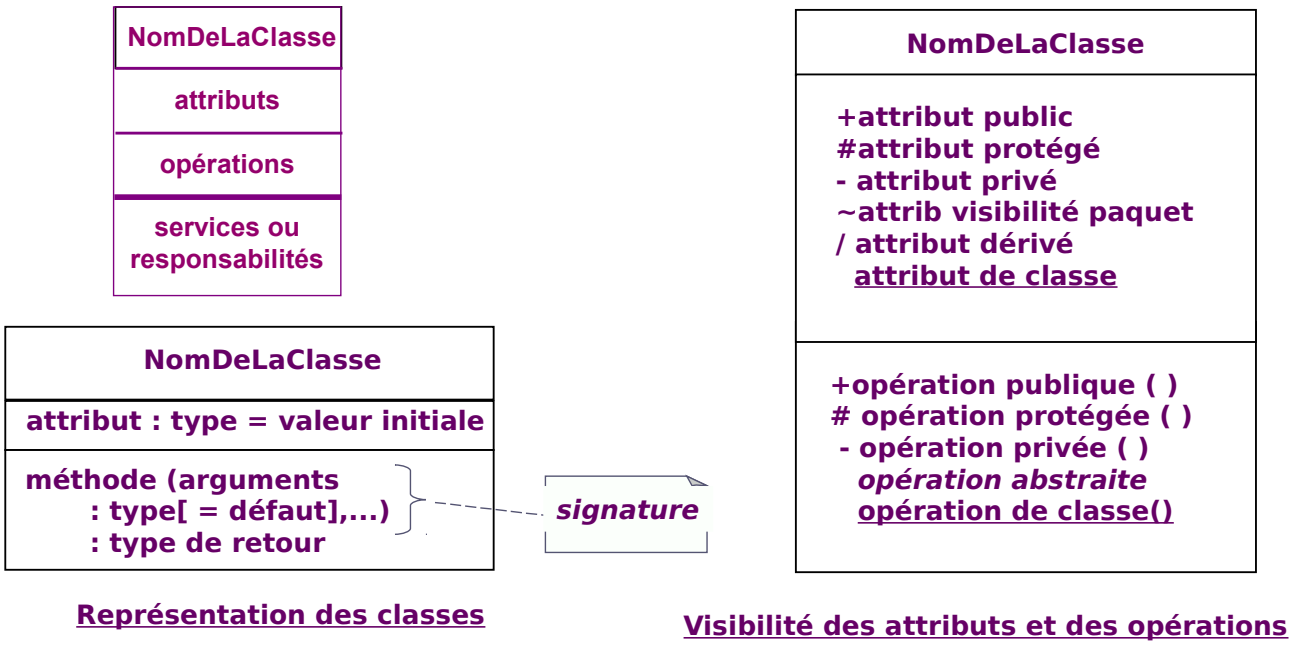
Il hérite de la sémantique du parent.
Il peut avoir des interactions spécifiques supplémentaires et/ou modifier les interactions héritées.

## Structuration en paquetages

Plusieurs stratégies possibles :
- regroupement par acteur
- regroupement par domaine fonctionnel



# *Diagramme de Classes*




**Représentation des classes**

**Visibilité des attributs et des opérations**

**attribut dérivé** : attribut dont la valeur peut être déduite d'autres informations disponibles dans le modèle.

**attribut de classe** : attribut dont la valeur est commune pour toutes les instances de la classe.

**{unique}** : les valeurs de l'attribut n'ont pas de doublons.

**{frozen}** : attribut non modifiable une fois initialisé.

**{readOnly}** : attribut modifiable à l'intérieur de la classe mais ne peut être changé de l'extérieur de la classe.

**{addOnly}** : on peut ajouter de nouveaux liens mais pas en supprimer.

**{ordered}** : objets ordonnés. On n'indique pas comment.

**{subset}** : un ensemble est inclus dans un autre.

**{xor}** : pour un objet donné, il n'y a qu'une seule association parmi un groupe d'associations..

**classe abstraite** : classe non instanciée.

**/ association dérivée** : associations redondantes mais considérées comme pertinentes par l'expert du domaine.

**agrégation** : si une des classes (*conteneur ou agrégat*) joue le rôle d'ensemble composé d'instances de l'autre classe (*agrégé*).

**composition**: agrégation particulière où
- l'objet composant (*partie*) ne peut appartenir qu'à un seul composite ou composé (*tout*) et
- le cycle de vie est imbriqué.

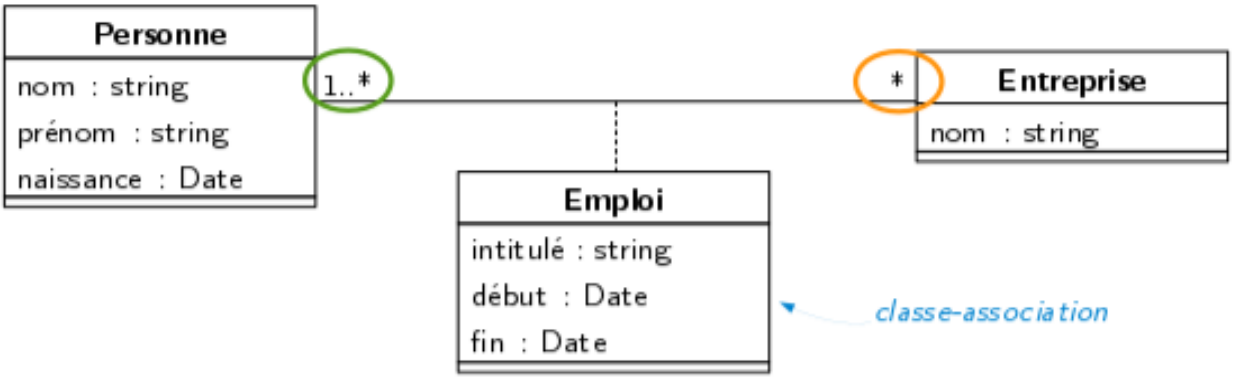
**1 - composition pas partageable : un composant ne peut appartenir qu'à un seul composite à la fois.**
**2 - cycle de vie imbriqué : la destruction du composite entraîne la destruction de ses composants.**

**héritage**: 

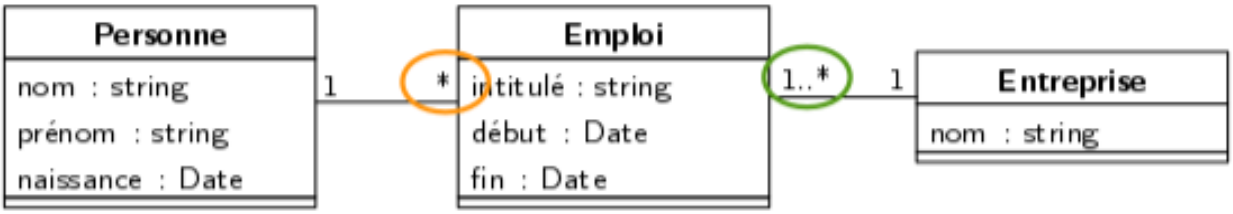
**contrainte entre sous-classes** : couverture + disjonction

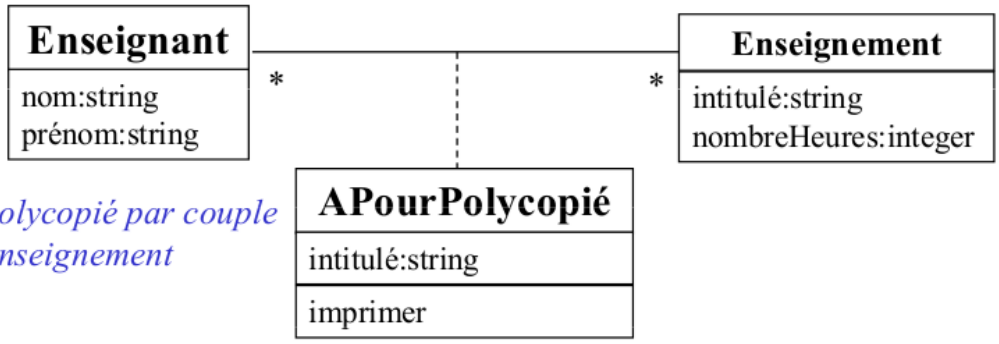
**contrainte de couverture {complete}** : tout objet de la super-classe doit appartenir à au moins l'une des sous-classes (*liste exhaustive de classes*).

**contrainte de disjonction {disjoint}** : tout objet de la super-classe doit appartenir à une seule sous-classe (*les sous-classes sont mutuellement exclusives*).

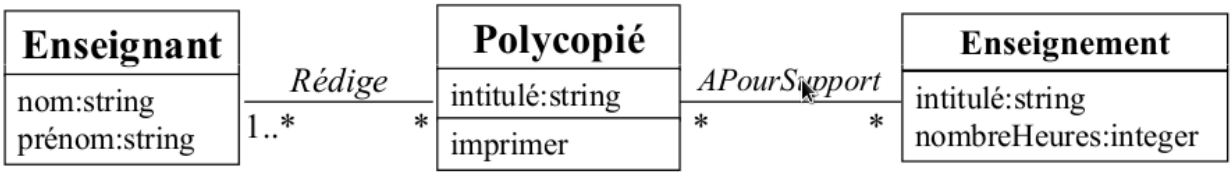



## Équivalence en termes de classes et d'associations

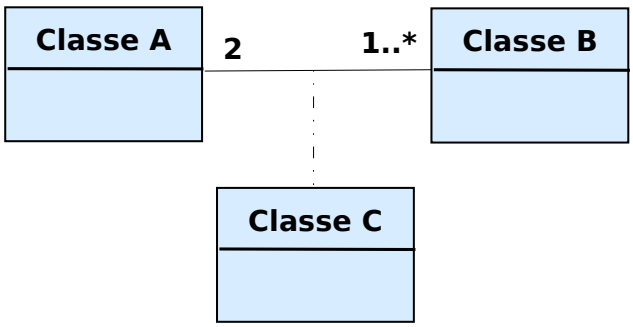


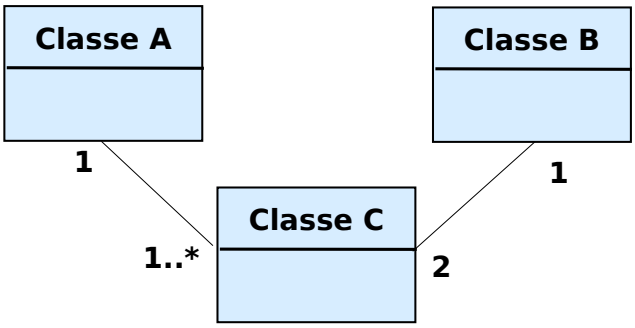

*Il y a un seul polycopié par couple Enseignant / Enseignement*



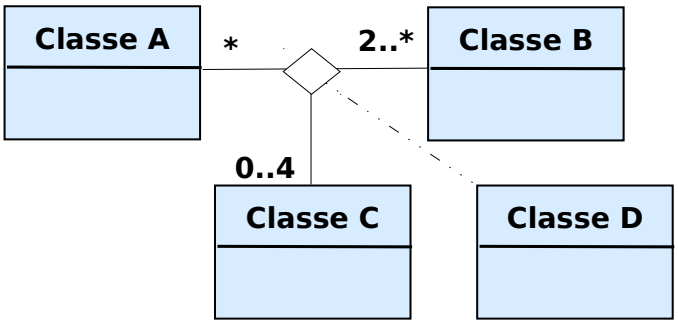

*Un nombre quelconque d'occurrences de Polycopié pour chaque Enseignant et chaque Enseignement*



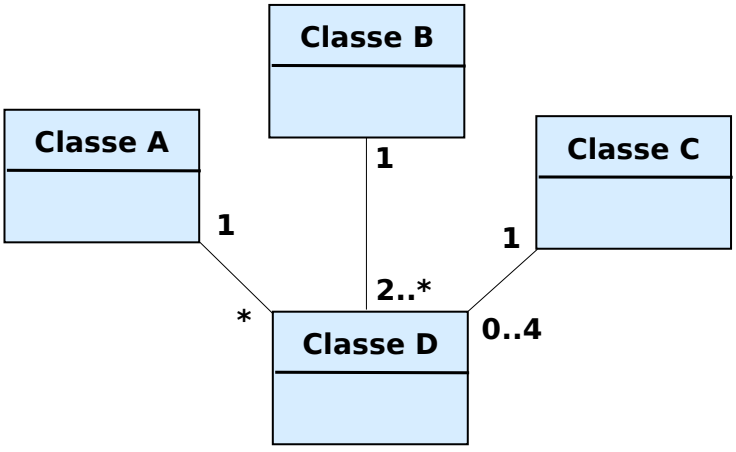

est équivalent à



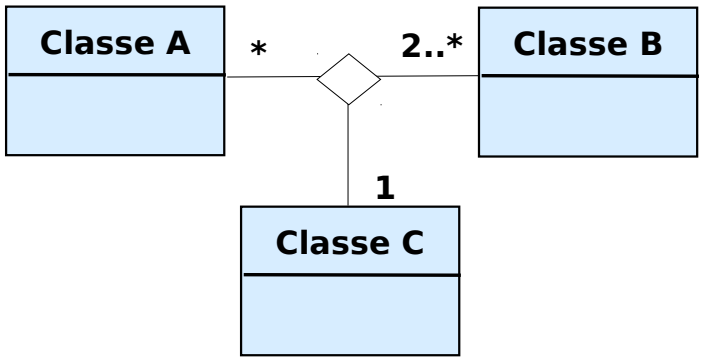

---



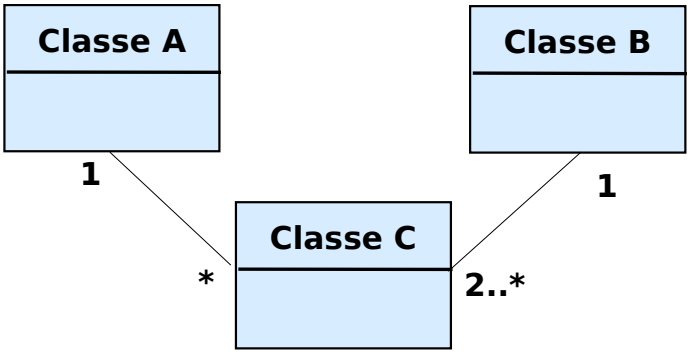

est équivalent à



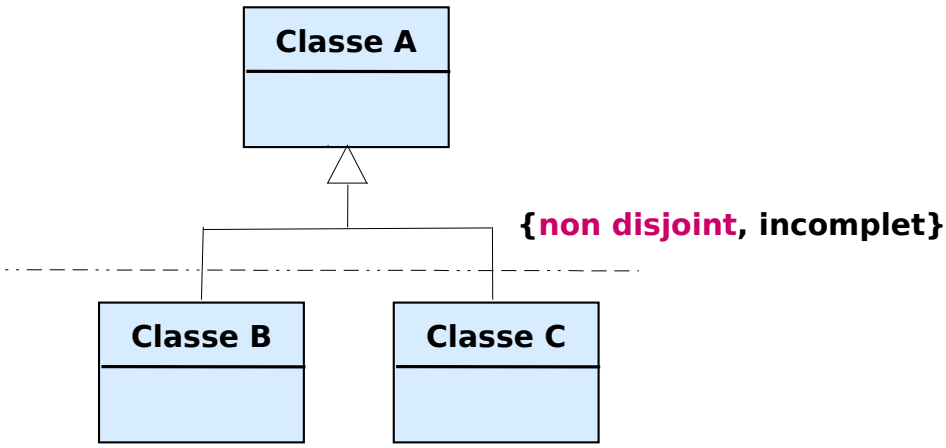
Modélisation à éviter (*cardinalité maximale à 1 dans une ternaire*)



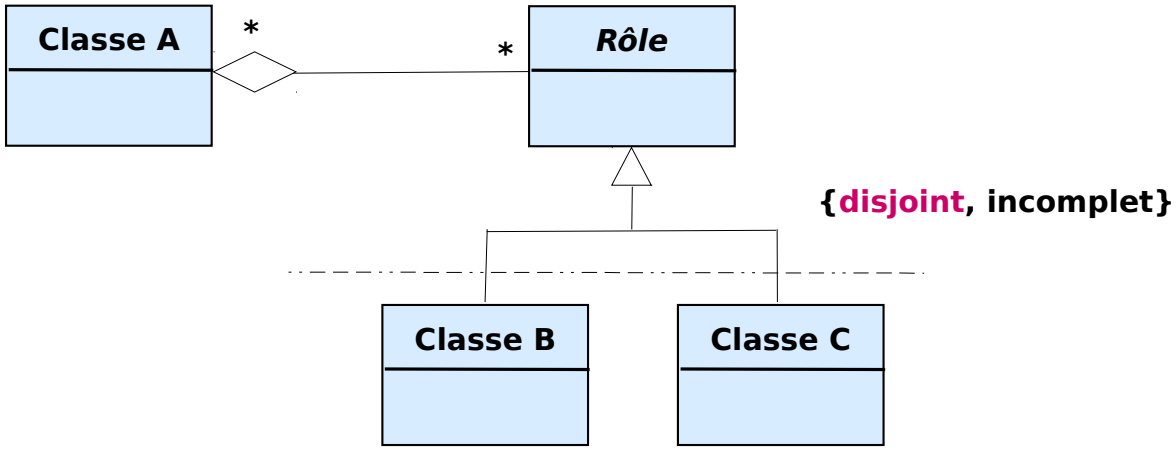

Il est préférable de faire deux binaires



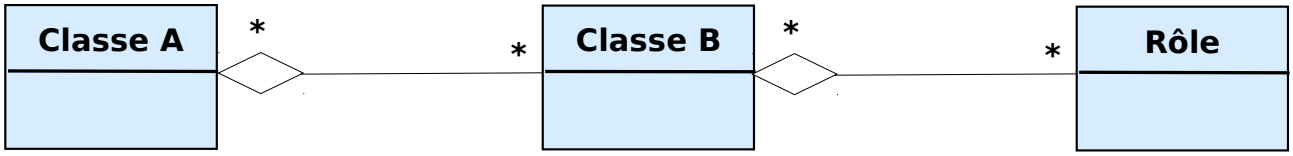

---



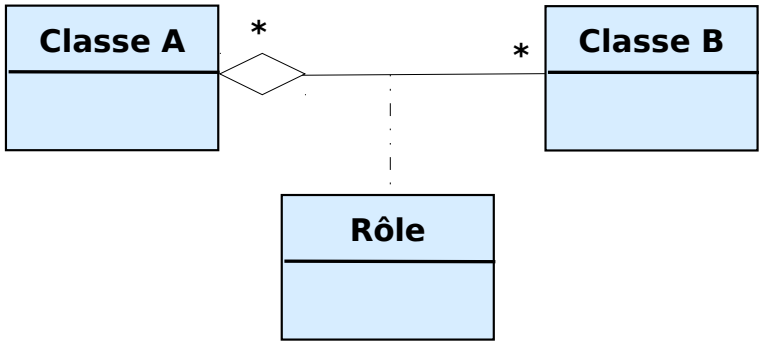

En programmation objet, il est préférable de modéliser de cette façon :




---

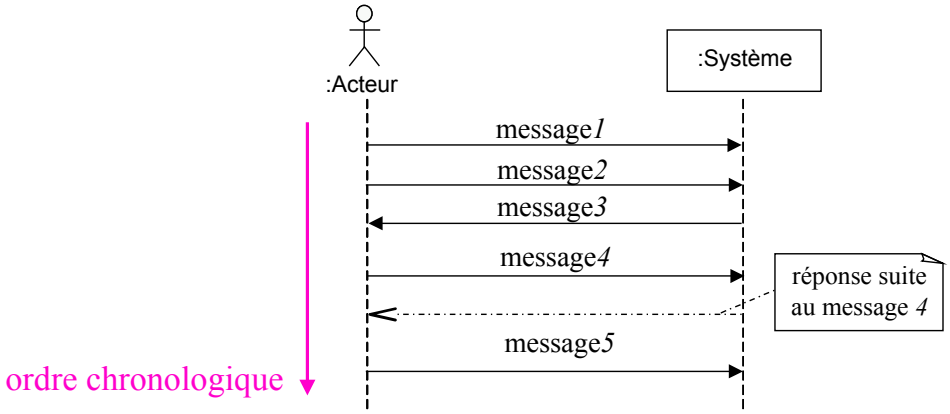



Si on veut connaître le rôle joué par un objet B pour un objet A, il faut modéliser de cette façon (*illustration voir exercice Quidditch*)

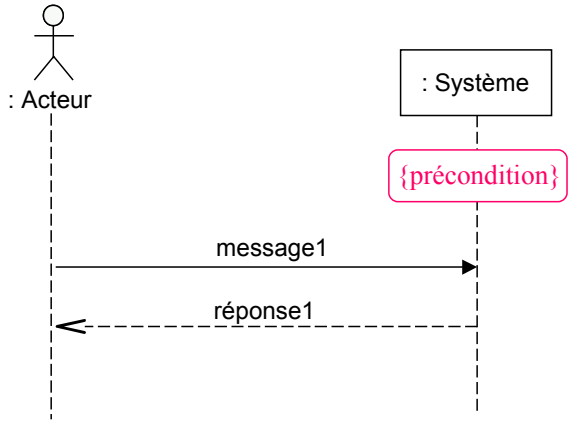


# *Diagramme de Séquence système*

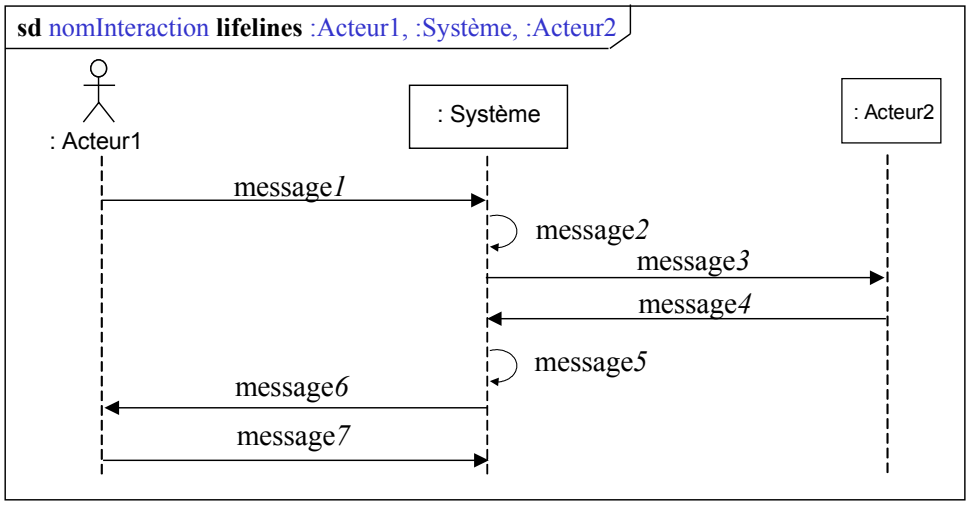
Le système informatique est vu comme une **boite noire**. Description **vu de l'extérieur**, sans préjuger de ***comment*** il sera réalisé. Interactions des acteurs avec le système.



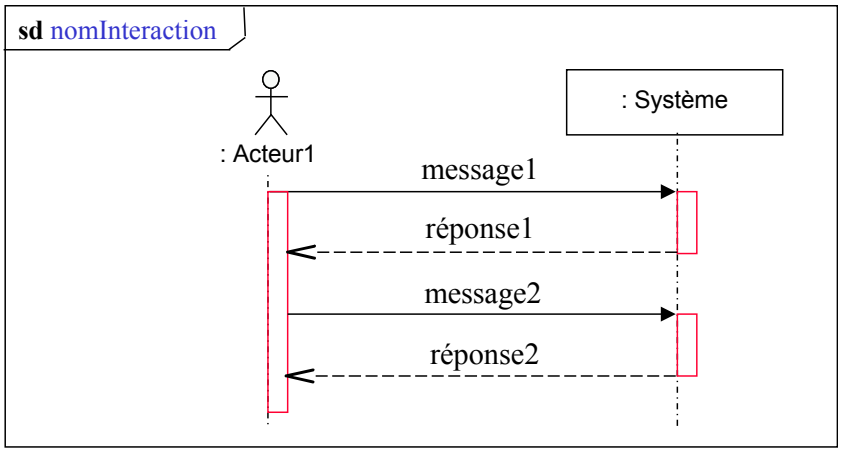

## Pré-condition

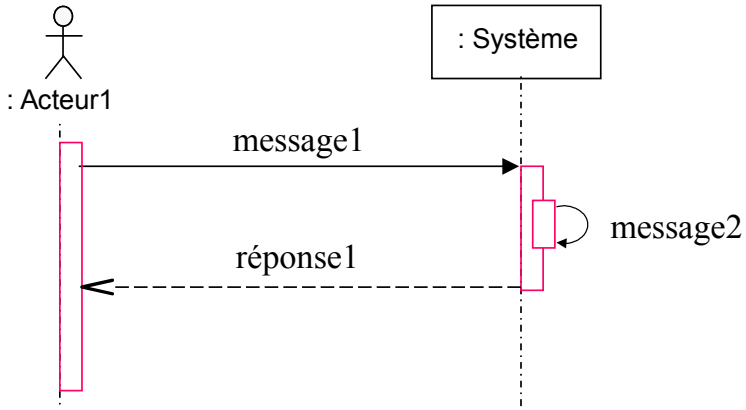



## Diagramme avec plus d'un acteur

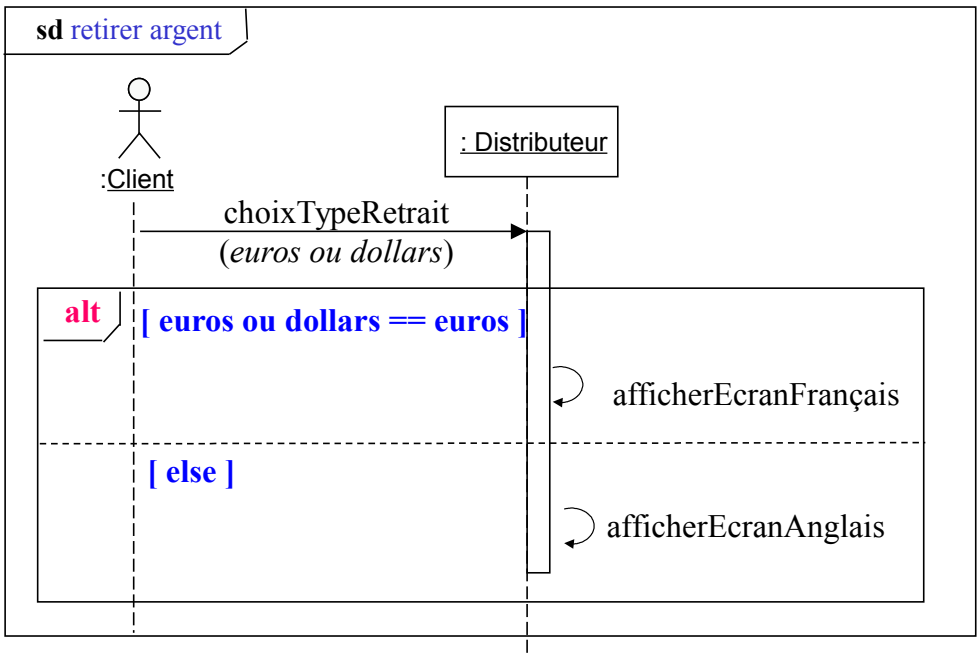


# Activation

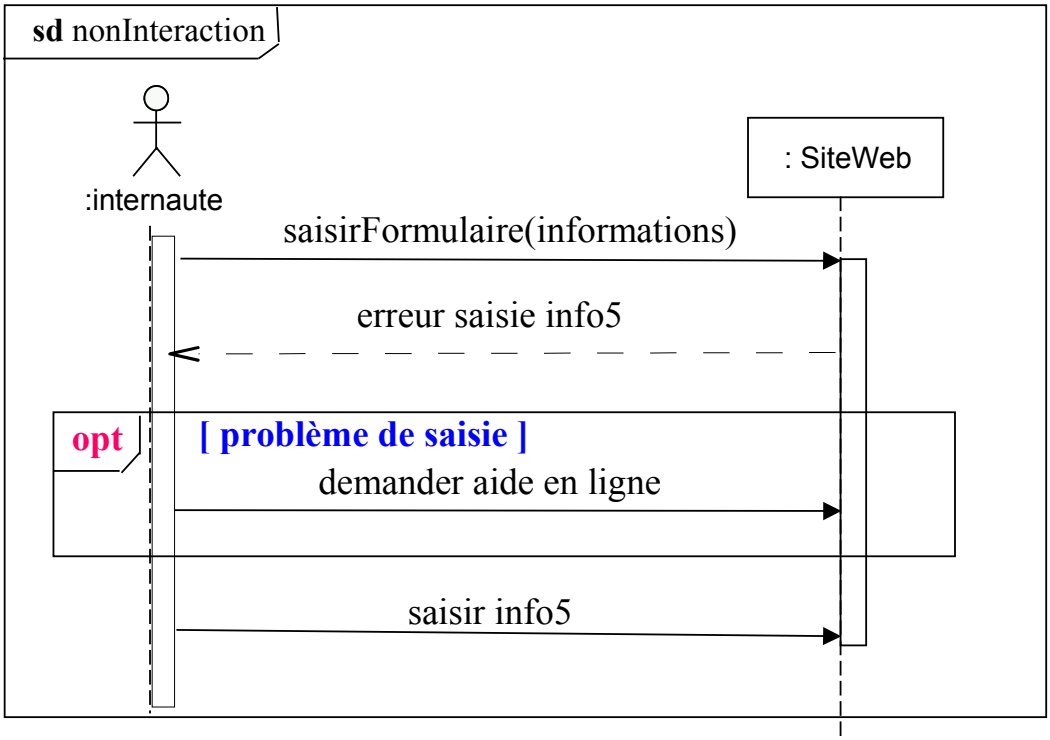


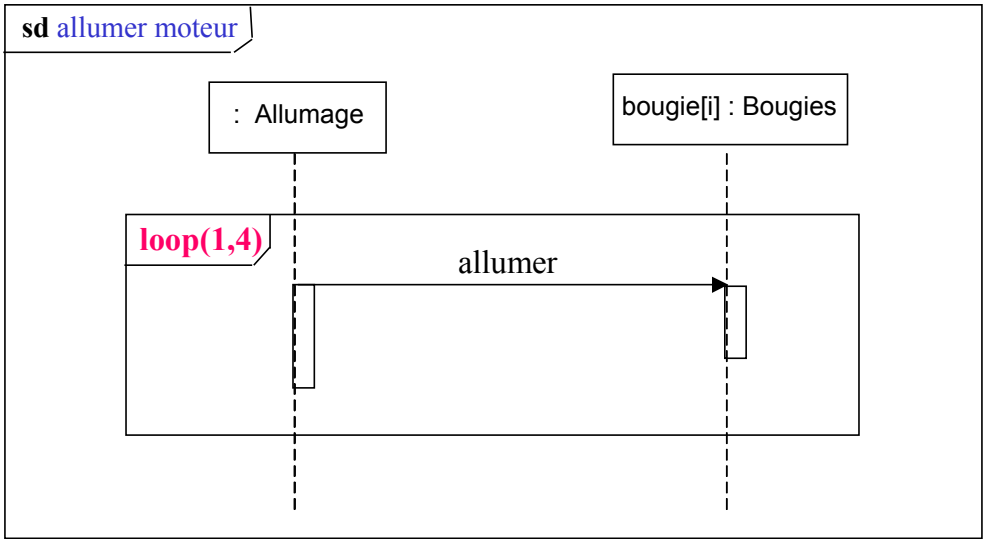

# Alternative

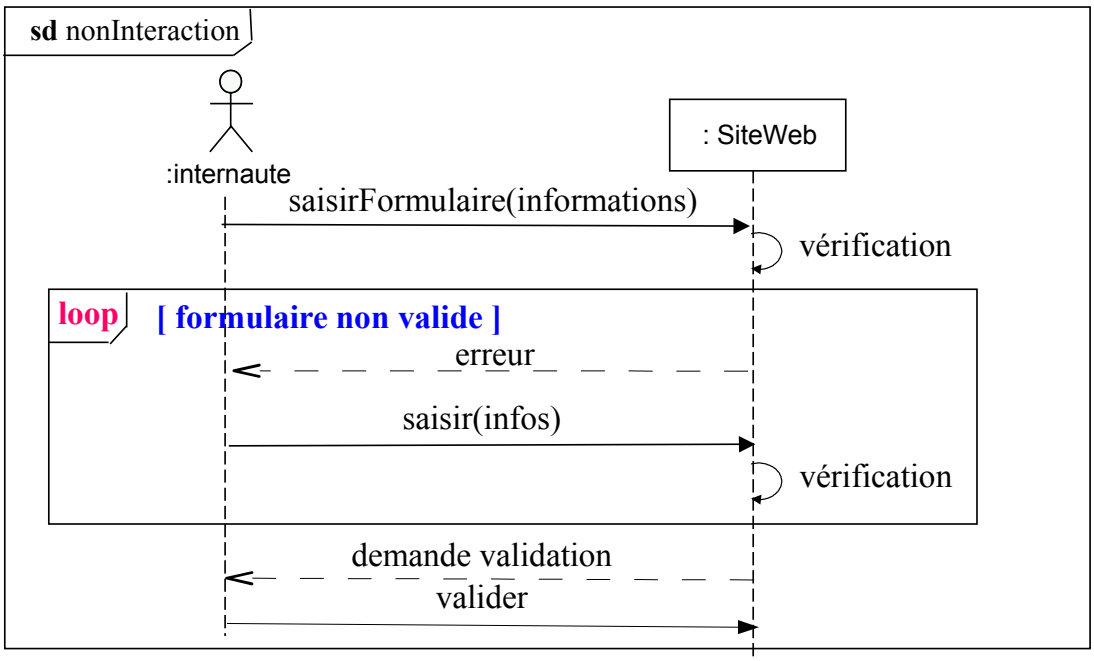


## Option

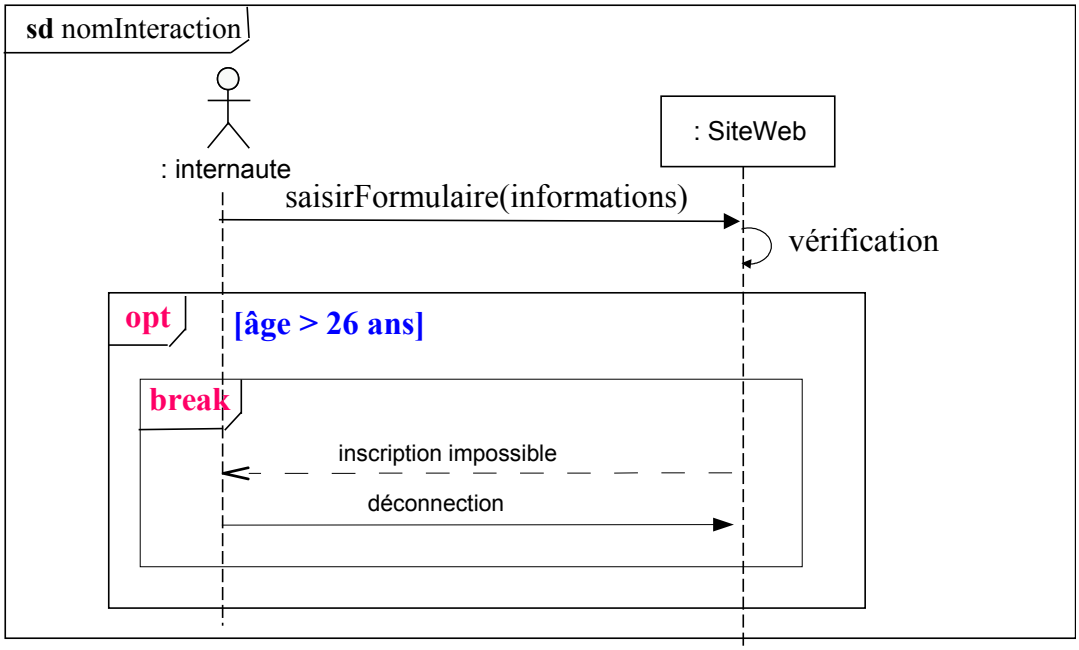



## Boucle



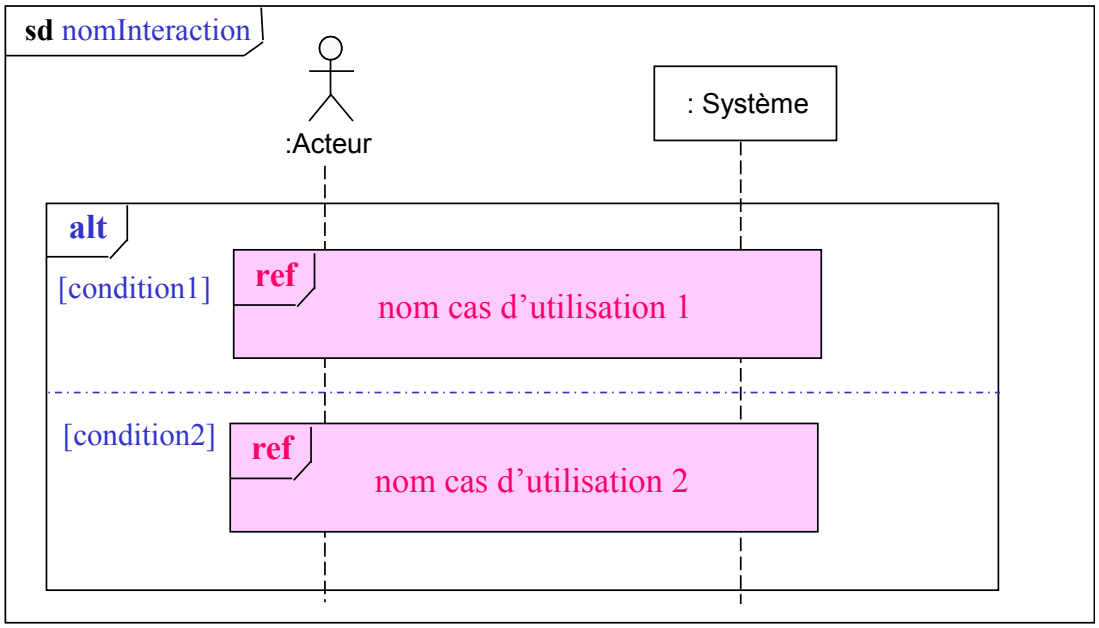



## Break




Le fragment est exécuté à la place de l'interaction englobante.

# Renvoi vers d'autres cas d'utilisation

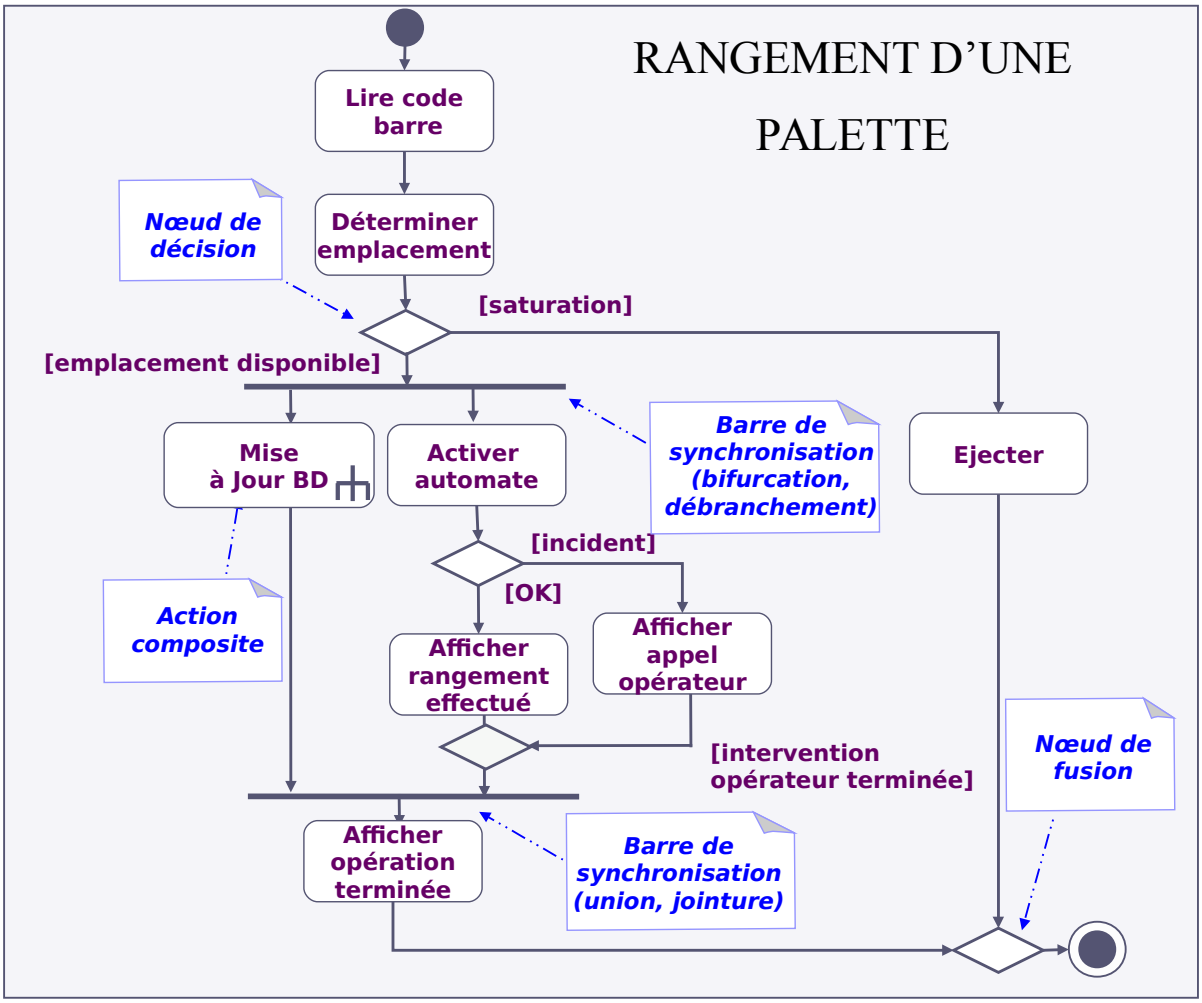


# *Diagramme d'activité*

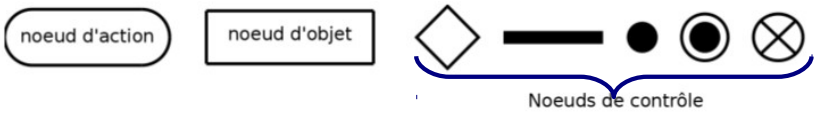
Diagramme comportemental d'UML. Proche de l'organigramme.

Permet de modéliser :

le comportement d'une méthode ou
le déroulement d'un cas d'utilisation.



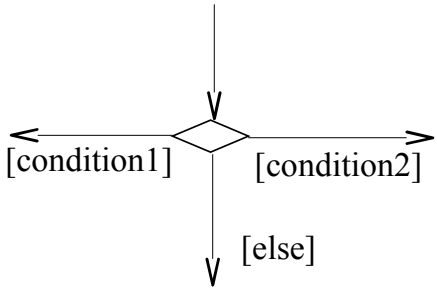
## Nœuds d'activité



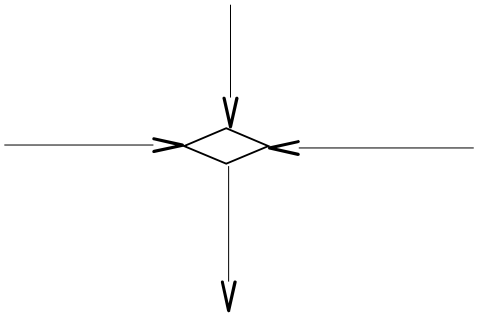

*Nœuds de contrôle :*

- nœud de décision ou de fusion
- nœud de bifurcation ou d'union
- nœud initial
- nœud final
- nœud final de flot

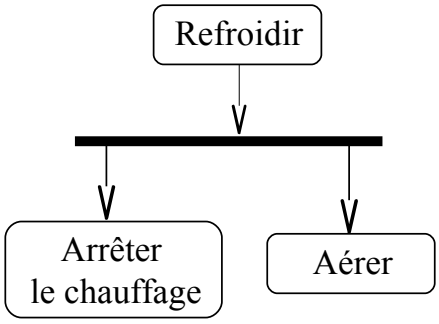
## Nœud de décision



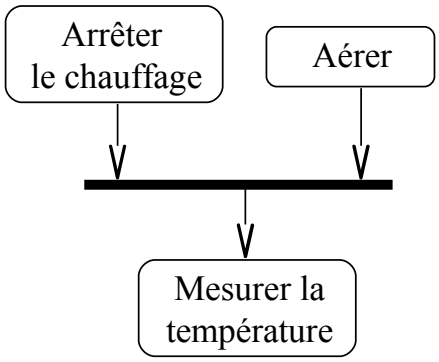

## Nœud de fusion



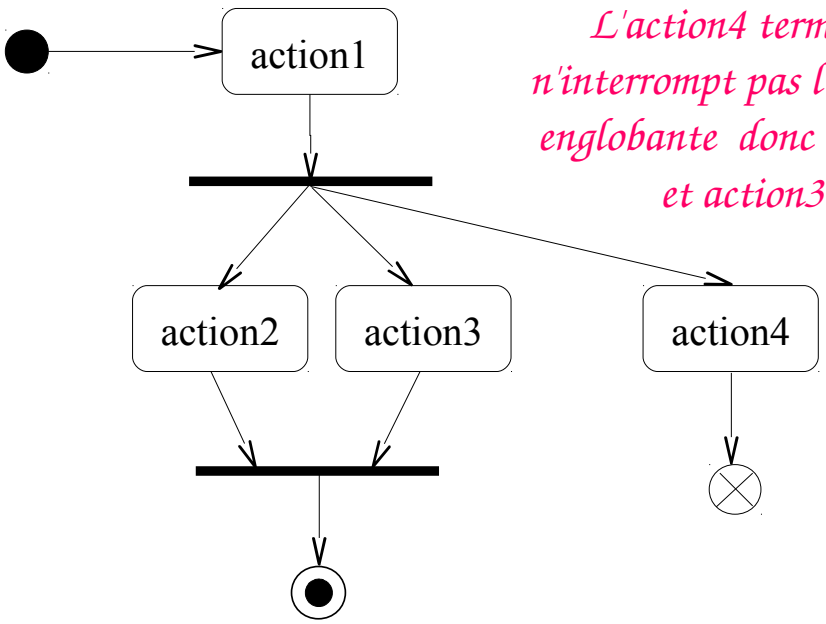
## Nœud de bifurcation




## Nœud d'union



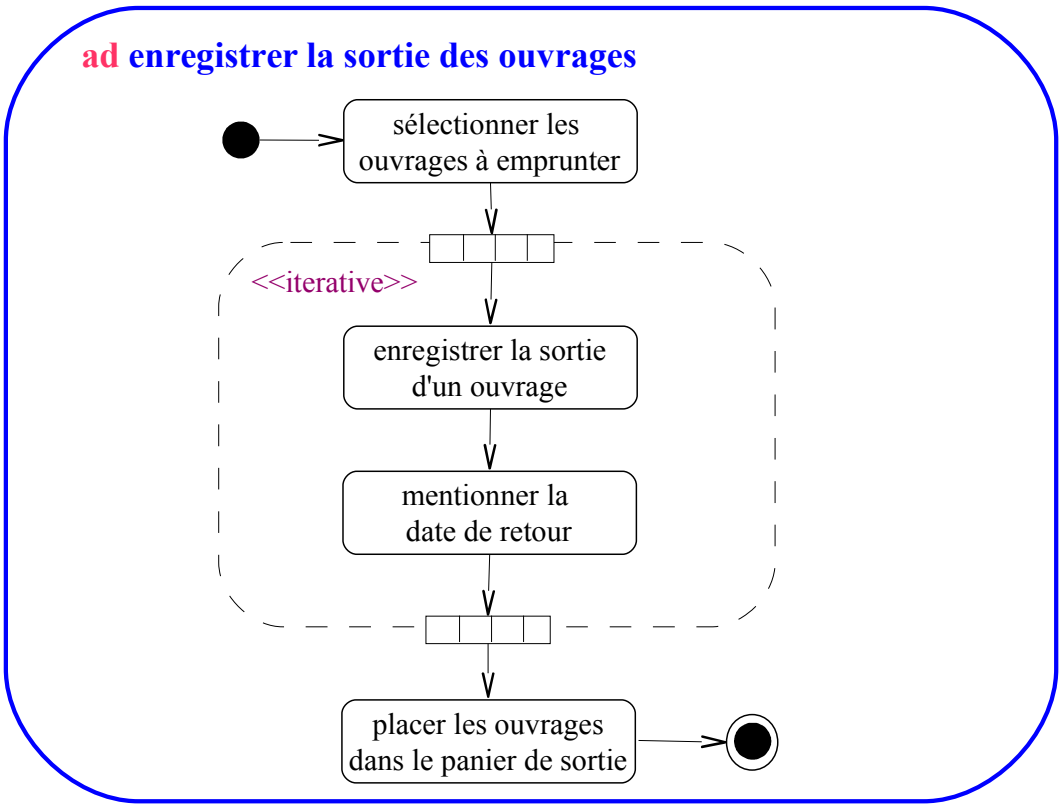

## Nœud de fin de flot

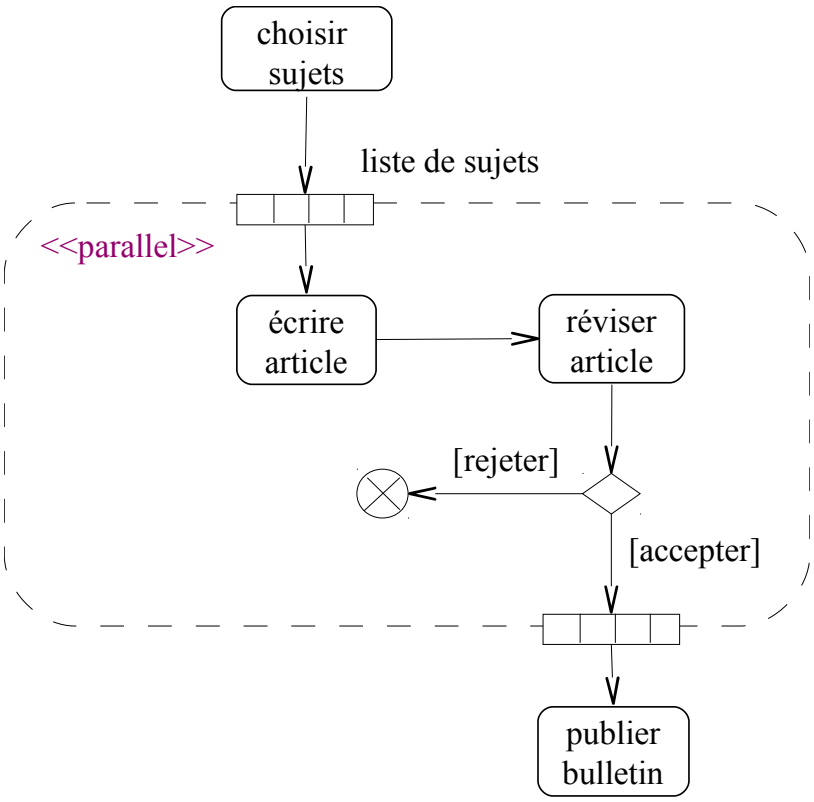



*L'action4 terminée n'interrompt pas l'activité englobante donc action2 et action3*

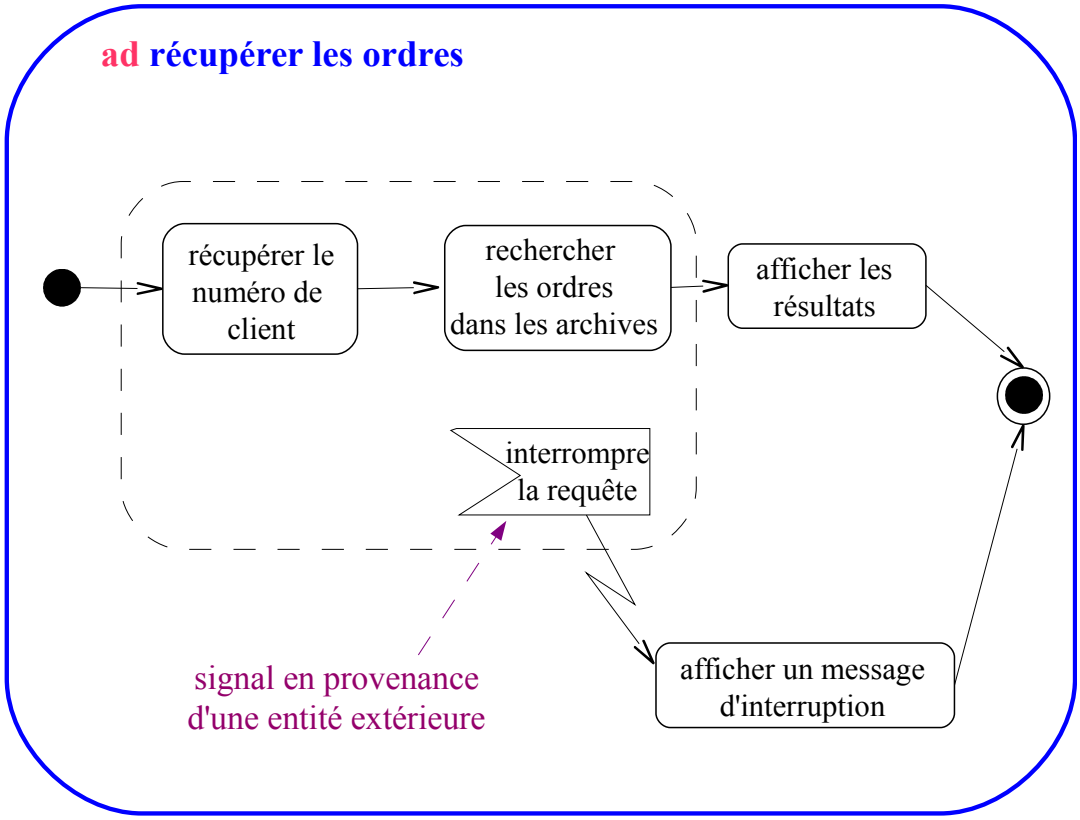
## Zone d'expansion

Nœud dans une activité qui accepte un ensemble d'objets en entrée, les traite individuellement et finalement les retourne en sortie tous traités.

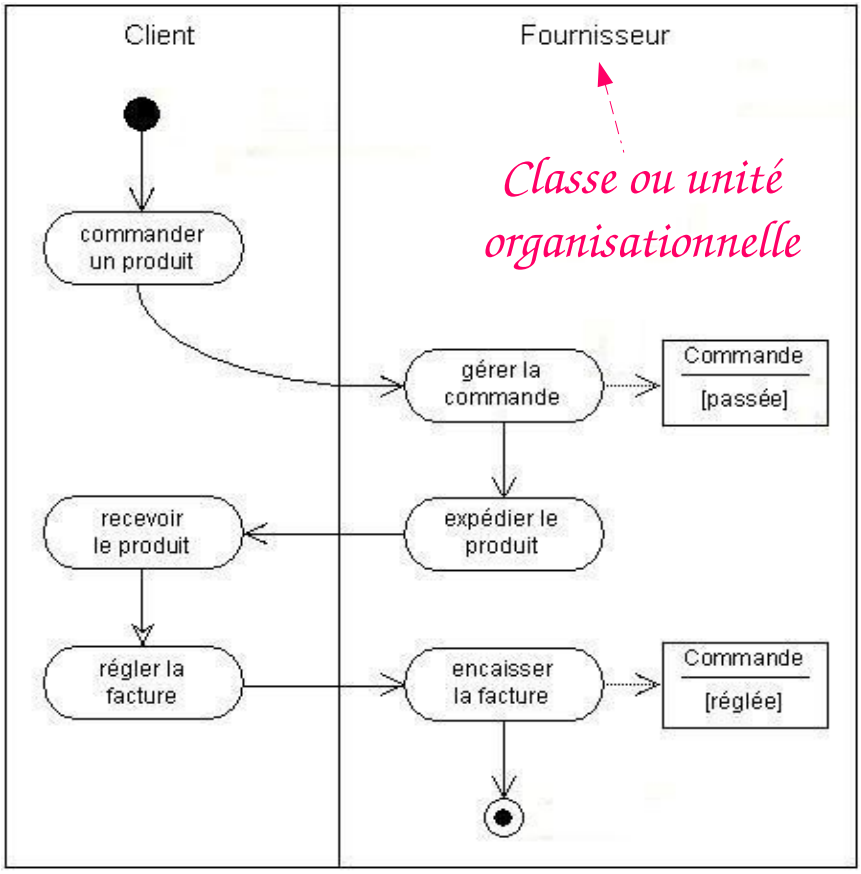


## Région interruptible



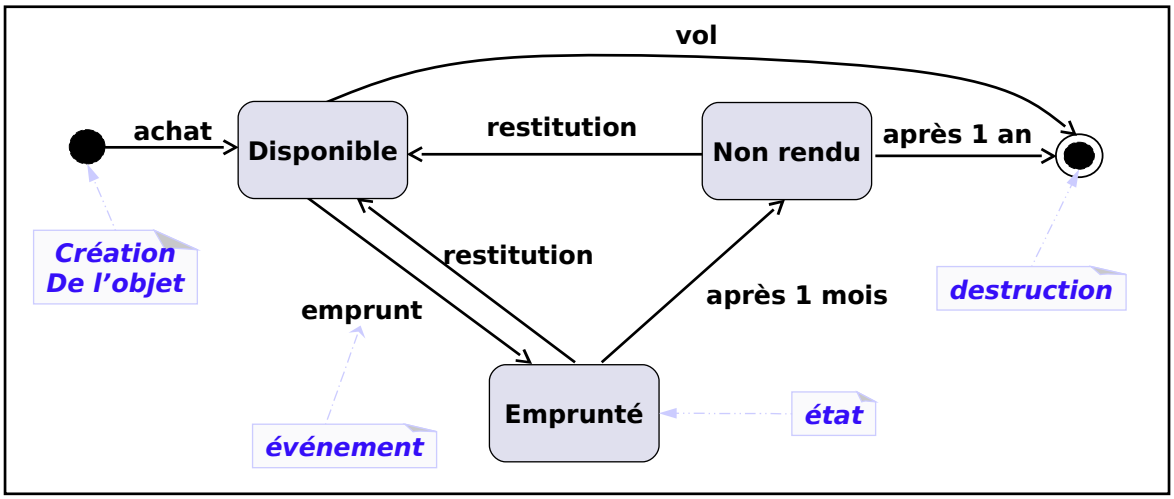

## Partition



# *Diagramme d'états - transitions*

Comportement d'un objet au cours du temps pour **une classe**.

## *Bibliothèque – objet exemplaire*




Un diagramme d'états n'a d'utilité que si l'objet passe par **au moins trois états**.

Le passage d'un état à un autre est **instantané** car le système doit toujours être dans un état connu.

## Construction

1- Représenter tout d'abord la séquence d'états qui décrit le comportement nominal d'un objet, avec les transitions qui y sont associées.

2- Ajouter progressivement les transitions qui correspondent aux comportements alternatifs ou d'erreur.

3- Compléter les actions sur les transitions et les activités dans les états.

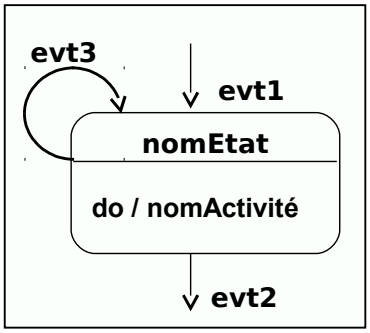
4- Structurer le diagramme en sous-états s'il devient trop complexe.

## Événements

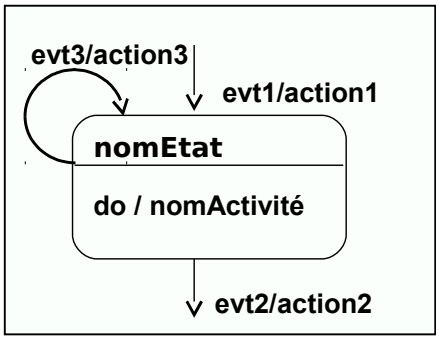
- Appel opération
- Passage du temps : *after(durée)*
- Changement dans la satisfaction d'une condition : *when(condition)*
- Fin d'une activité : *do/*

# Opérations
- activités
- actions

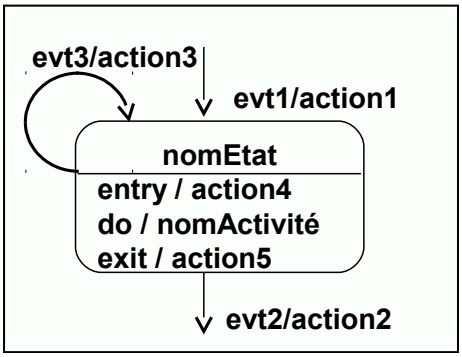
# Activité
Une activité est une opération qui nécessite du temps pour s'exécuter.
Une activité est associée à un état.
Une activité est placée à l'intérieur d'une boite d'état après un do : *do / nomActivité*




# Action
Une action est une opération instantanée.
Une action est associée à un événement.
Une action est placée après l'événement séparée par une barre oblique : *nomEvt / nomAction*.

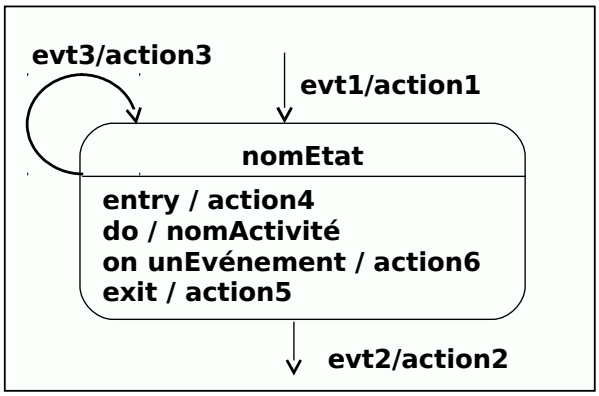



# Action Entrée / Sortie
Une action en entrée (*sortie*) est exécutée chaque fois que l'on rentre (*quitte*) dans l'état.
L'action est indiquée après les mots clés "*entry*" ou "*exit*" suivis du caractère "/".

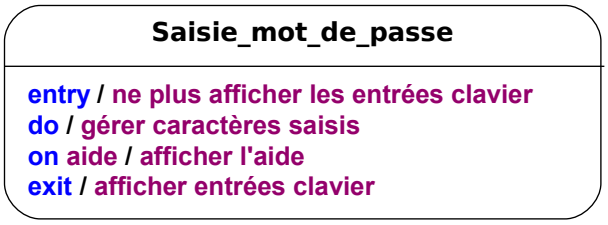


## Action dans état

Un état peut contenir une autre action exécutée lors de l'occurrence d'un événement pendant que l'objet est dans l'état : c'est un événement interne.
L'action est indiquée après le mot clé : *on unEvénement / .*

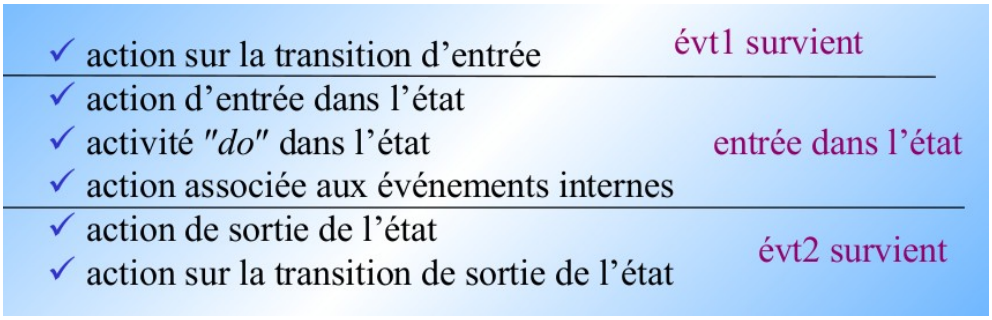



**Exemple**

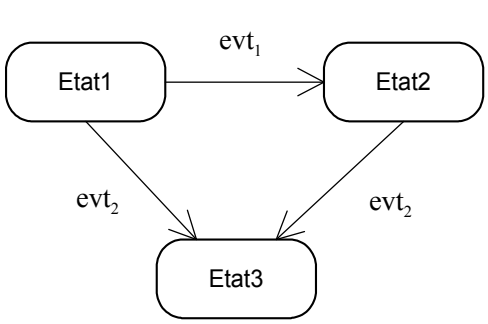


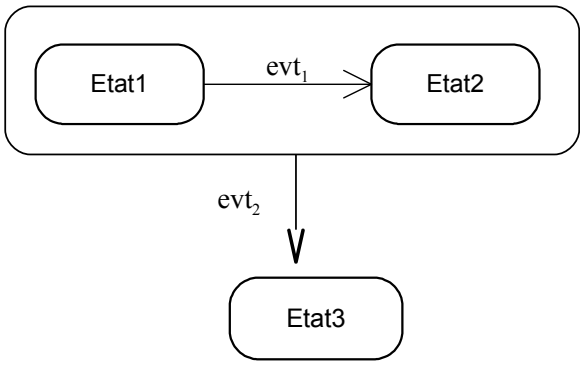

## Exécution opérations

Si plusieurs opérations sont spécifiées dans un état, elles sont exécutées dans l'ordre suivant :

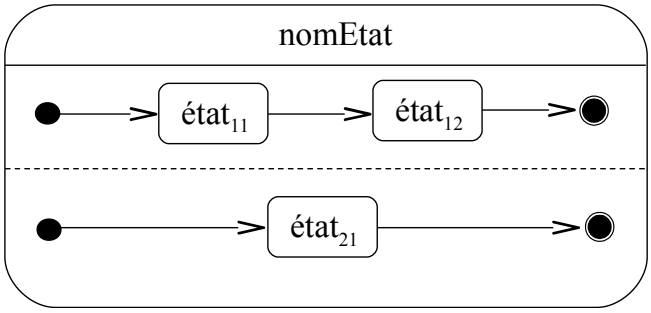



## Généralisation d'états



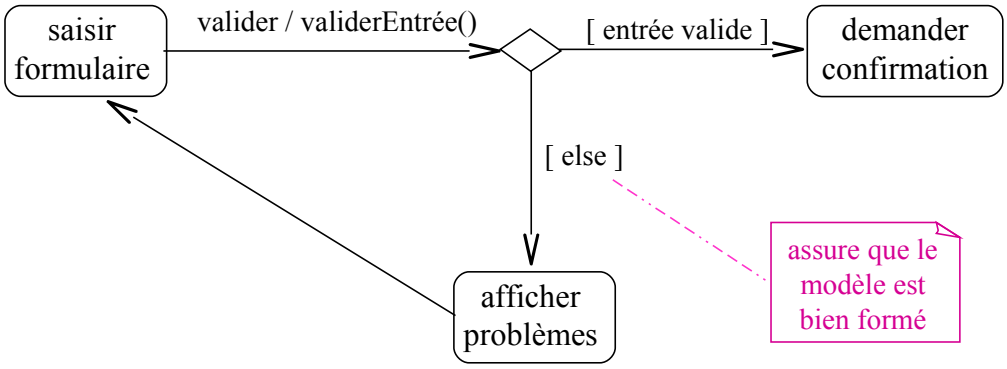
## Agrégation d'états




**La généralisation simplifie par factorisation,**
**L'agrégation simplifie par segmentation de l'espace des états.**

## Point de choix

Un point de choix ou décision possède une entrée et au moins deux sorties.




## Historique

Une transition ayant pour cible le pseudo-état historique, noté par un cercle contenant un H, est équivalente à une transition qui a pour cible le dernier état visité dans la région contenant le H.


*historique de surface*

Une transition ayant pour cible le pseudo-état historique profond, noté par un cercle contenant un H*, permet d'atteindre le dernier état visité dans la région, quel que soit son niveau d'imbrication, alors que le pseudo-état H limite l'accès aux états de son imbrication dans la région.


*historique profond*