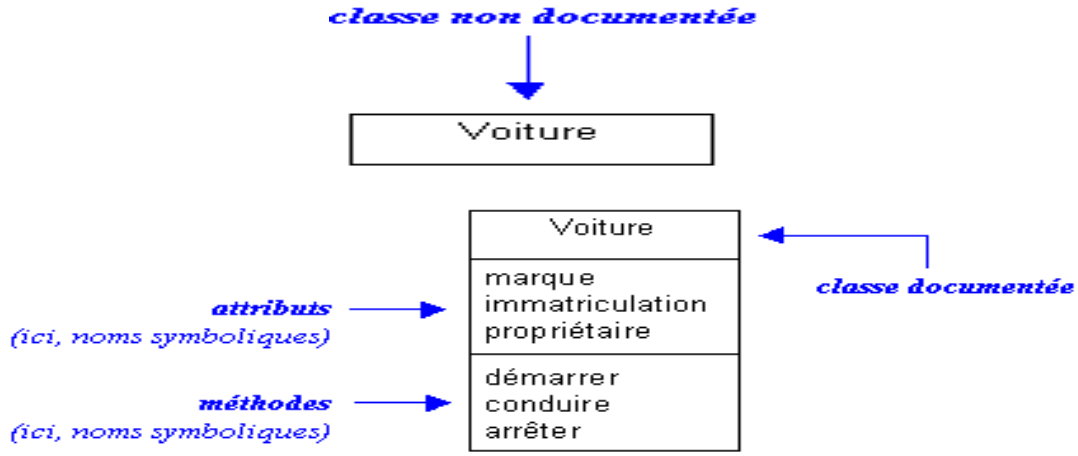
Cours UML (suite)

1) Diagramme des classes

Ce diagramme montre la structure du système sous la forme d'un ensemble de classes et de relations entre ces classes. Le modèle représenté par ce diagramme constitue le pilier des étapes de conception et de réalisation du système. Il spécifie la structure et les liens entre les objets qui composent le système

1.1. Objet et classe

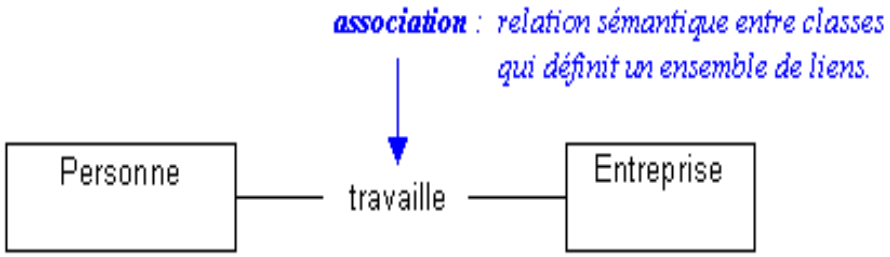
- Un **objet** est un concept, une abstraction ou une chose ayant un sens précis dans le contexte du problème.
- Une **classe** décrit un groupe d'objets ayant : des propriétés similaires (attributs), un comportement commun (opérations communes), des relations communes avec les autres objets, une même sémantique.
- Un **attribut** est une valeur de donnée. Son existence dépend d'un objet
- Une **opération** est une fonction applicable aux objets d'une classe : Toute opération a un objet cible.



1.2 - Liens et Associations

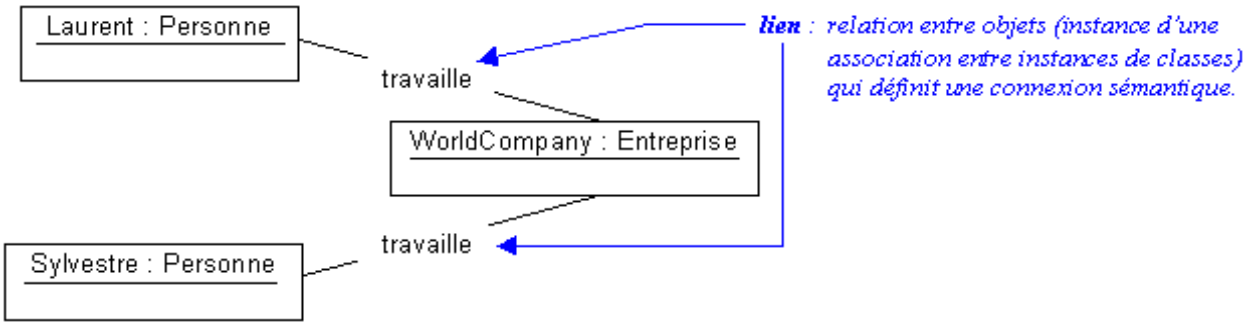
- Un **lien** est une relation physique ou conceptuelle entre des objets. C'est une instance d'une association.
- Une **association** est un groupe de liens ayant une structure et une sémantique communes. Elle est fondamentalement bidirectionnelle et peut être binaire, ternaire,...

Toute relation entre les classes doit être modélisée en terme d'association et non pas par un attribut de type pointeur. Bien que l'association est généralement bidirectionnelle, on peut donner un sens de lecture facilitant ainsi la compréhension du modèle, comme le montre l'exemple suivant.

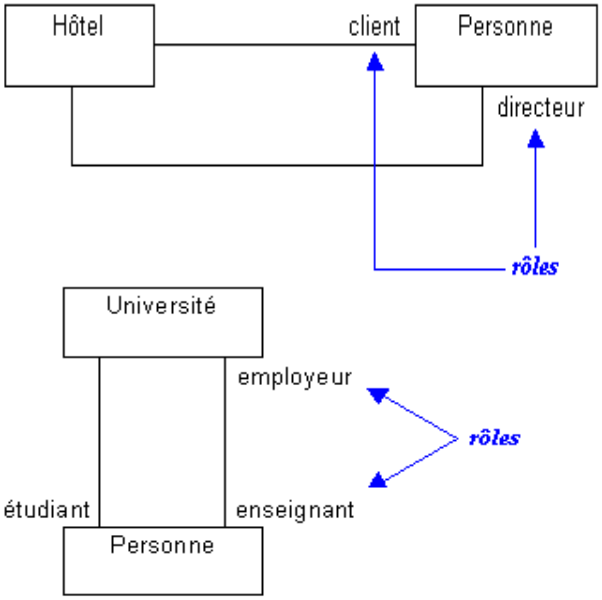


Exemple d'une association entre 2 classes

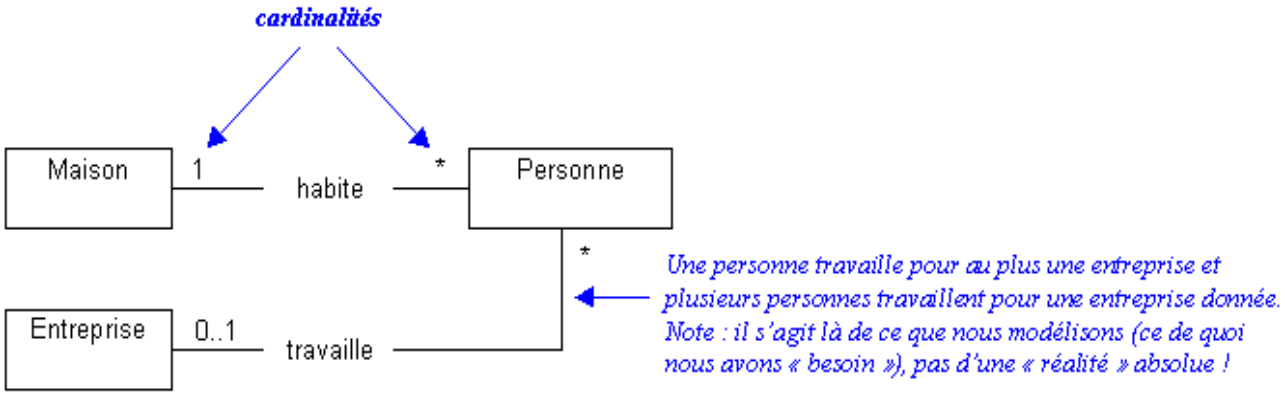
On peut instancier cette association sous la forme d'un lien entre un objet de la classe Personne (Laurent) et un objet de la classe Entreprise (WorldCompany), comme le montre l'exemple suivant :



Pour mieux documenter les associations entre les classes, on peut employer la notion de rôle. **Un Rôle** identifie de façon unique une extrémité d'une association lorsque la même classe participe plusieurs fois à la même association.



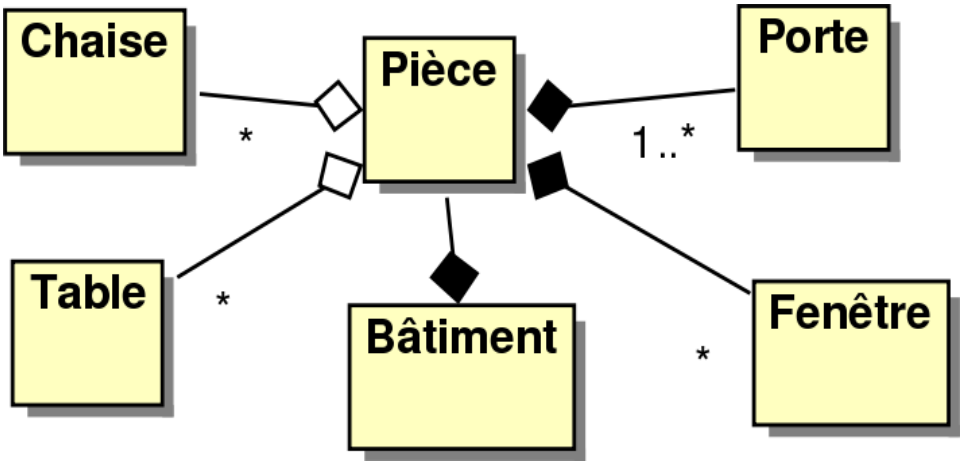
Comme dans le cadre du modèle entité/association, les **Cardinalités** peuvent être ajoutées pour préciser le nombre d'instances qui participent à une association.



Expression des cardinalités d'une relation en UML :
-n : exactement "n" , n entier naturel > 0
-n..m : de "n" à "m" , entiers naturels ou variables, m ≥ n
-* : plusieurs , équivalent à "0..n" et "0..*"
-n..* : "n" ou plus , n entier naturel ou variable

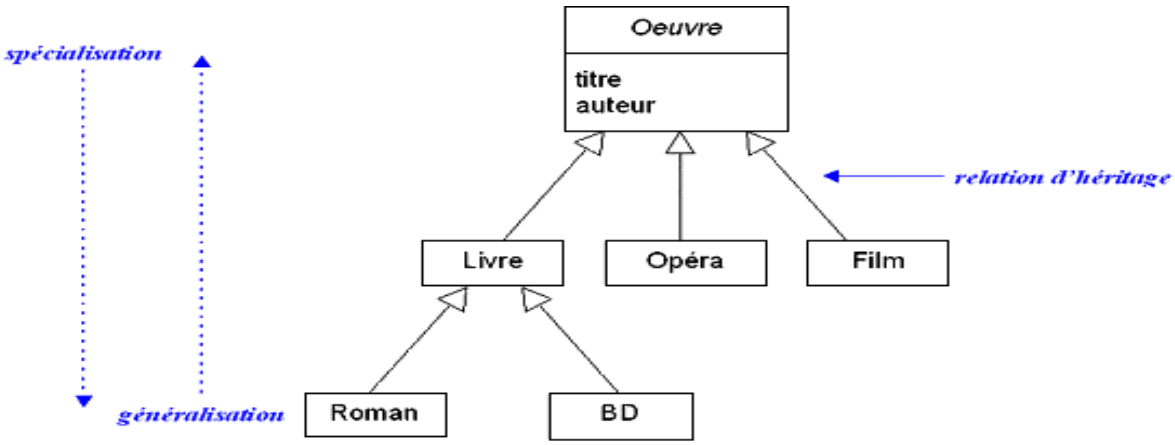
1.3 - L'agrégation

C'est une association de type "composé-composant" ou "partie de". Elle exprime une relation entre un ensemble et ses parties. On distingue deux types d'agrégation:
-l'agrégation forte ou composition (losange plein): Un composant n'a pas d'existence indépendante de l'objet agrégat. La destruction de l'objet composite (l'ensemble) implique la destruction de ses composants (les parties)
-l'agrégation simple (losange vide): les composantes peuvent exister malgré la destruction de l'objet composite.



1.4 - Généralisation et héritage

L'héritage est un mécanisme d'abstraction puissant permettant aux classes de partager leurs points communs tout en préservant leurs différences. Il exprime la possibilité pour une classe (dite sous-classe) d'hériter les attributs et les opérations d'une autre classe (dite sa super-classe). Bien évidemment, la sous-classe peut introduire d'autres attributs et opérations en plus de ceux hérités de sa super-classe. Une instance de sous-classe est aussi une instance de toutes les classes ancêtres.
La généralisation est la relation entre une classe (super-classe) et une ou plusieurs versions affinées (sous-classes) de cette classe.

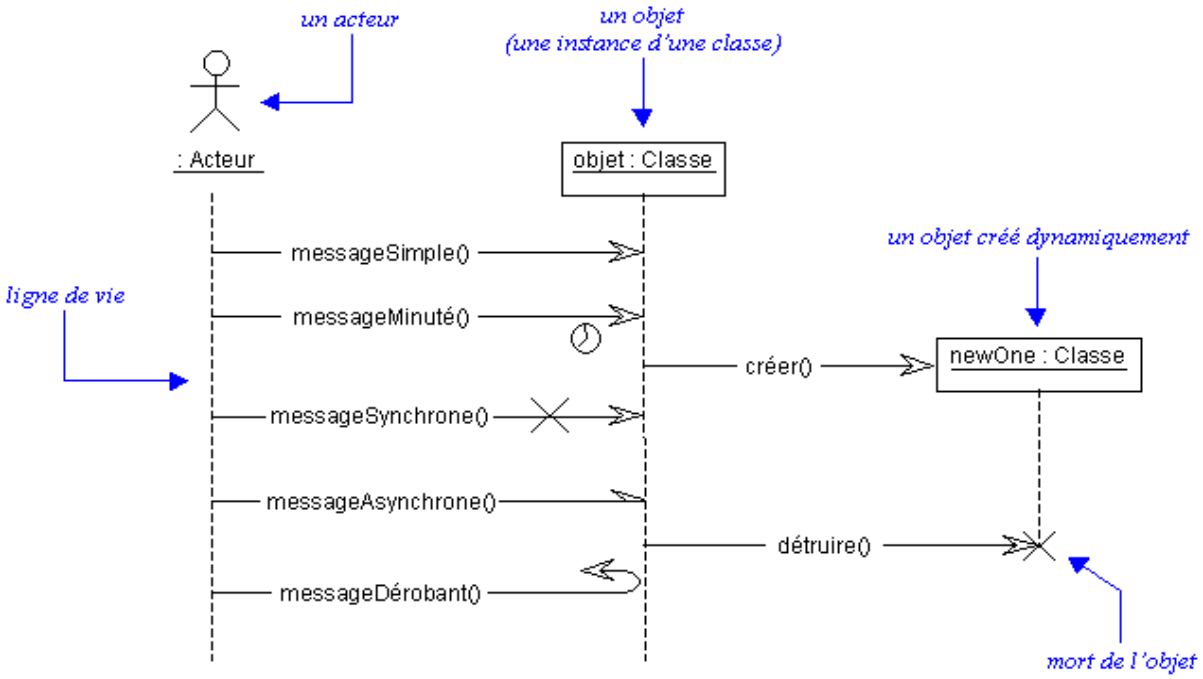


2. Les diagrammes de séquence et de collaboration

2.1 Diagramme des séquences (d'interaction)

Il spécifie les interactions entre les objets selon un point de vue temporel. Les diagrammes de séquences peuvent servir à documenter les cas d'utilisation en montrant les interactions qu'ils

déclenchent entre les objets participant à leur réalisation. La figure suivante résume quelques éléments des diagrammes de séquence.

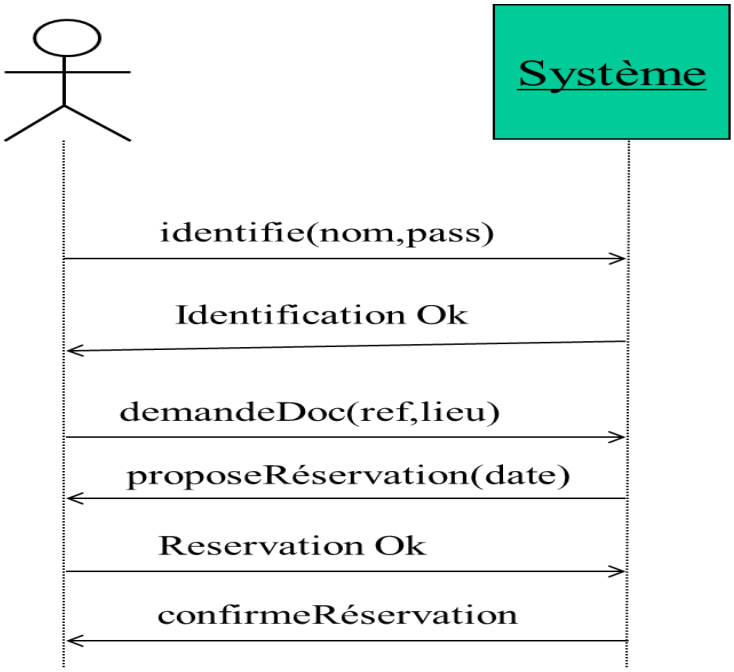


Ainsi La description d'un cas d'utilisation en termes de diagrammes de séquence se fait par des scénarios qui définissent la suite logique des interactions:

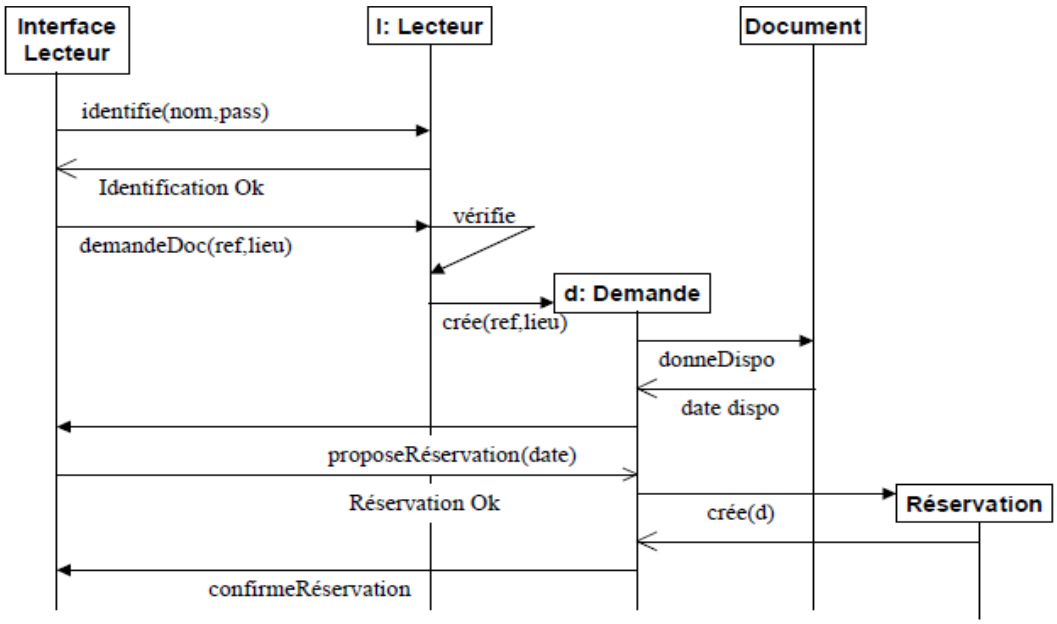
• Entre l'acteur et le système dans son ensemble: Description de haut niveau
• Entre les objets qui participent à la réalisation du cas d'utilisation: Description détaillée

Pour illustrer ces deux descriptions, considérons l'exemple d'un scénario correspondant au cas d'utilisation « Demande d'un document », de l'application de gestion d'une bibliothèque. Le scénario choisi concerne le cas où la demande ne peut être satisfaite, cependant le système propose la possibilité de réserver le document : On peut résumer ce scénario en langage naturel ainsi :
• le lecteur s'identifie
• Après accord, il effectue sa demande de document
• Le système propose une réservation
• Le lecteur accepte la réservation
• Le système confirme la réservation par la délivrance d'une fiche

Un premier diagramme d'interaction (description de haut niveau) peut être construit et utilisé pour concevoir une 1ère version de l'IHM (des écrans de dialogue pour l'exécution d'un Cas d'Utilisation). Il met l'accent sur les messages échangés entre l'acteur déclenchant le cas d'utilisation et le système, comme le montre la figure suivante :



Dans un second temps, le digramme précédent peut être détaillé pour exprimer ce qui se passe à l'intérieur du système : les échanges de messages entre les objets qui participent à la réalisation du cas d'utilisation considéré. On obtient alors le deuxième diagramme suivant (description détaillée du scénario) :

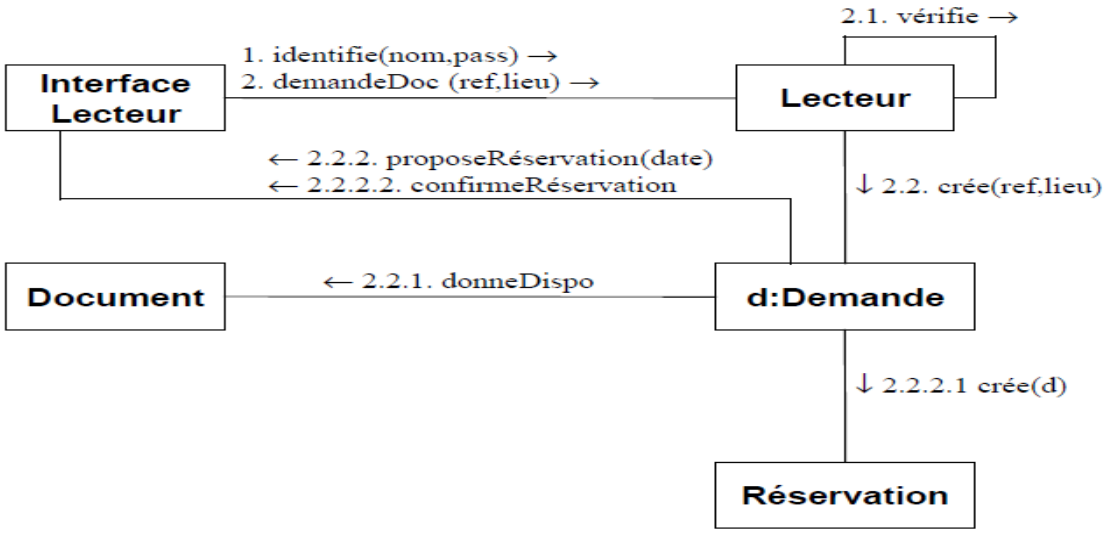


2.2 Diagramme de collaboration

Il a le même objectif que le diagramme de séquence, mais en faisant abstraction de l'aspect temporel des interactions:

- Le temps n'est pas représenté explicitement
- Les différents messages sont numérotés pour indiquer l'ordre des envois

Pour illustrer ce type de diagramme, reprenons le même exemple que précédemment. La figure suivante illustre le diagramme de collaboration correspondant à la description détaillée du scénario précédent.



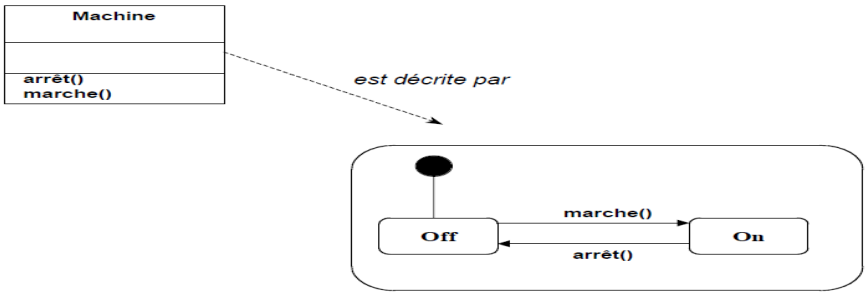
3. DIAGRAMME D'ETATS-TRANSITIONS

3.1 Définition

Il permet de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets ou composants ou avec des acteurs. Le formalisme utilisé pour décrire le diagramme d'état est celui des automates à états finis : graphes dont les nœuds sont les états, reliés par des arcs orientés qui décrivent les transitions. Voici la forme générale :

Qui peut se lire ainsi : « Si le système ou l'objet est dans l'état A et que l'événement E survient alors le système passe (transite vers) à l'état B »

A titre d'exemple, considérons la classe Machine, à qui on peut associer un comportement simple, spécifié par l'automate à deux états suivant :



On notera que l'évènement marche () (ou arrêt()) correspond à l'appel de l'opération marche de la classe machine.

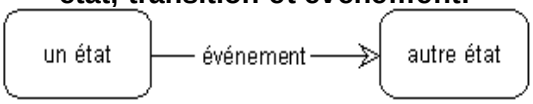
UML utilise une version des automates appelée les statecharts : des automates introduisant la notion d'hiérarchie entre les états (un état peut se décomposer en sous-états), l'orthogonalité (la possibilité de représenter des comportements parallèles) et la diffusion.

3.2 Les Concepts de base : Etat, Evénement, Transition

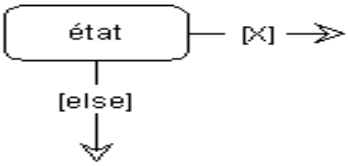
- **Etat** : se caractérise par sa durée, sa stabilité, et représente une conjonction instantanée des valeurs des attributs d'un objet.
- **Evénement :** représente un phénomène sans durée, survenant à un instant donné et déclenche une opération, ou signale la fin d'une opération. A chaque objet, on peut associer au moins un événement de création et un événement de fin de vie.
- **Transition** : représente le passage instantané d'un état vers un autre.

- est déclenchée par un événement : c'est l'arrivée d'un événement qui conditionne la transition.
- peut aussi être automatique, lorsqu'on ne spécifie pas l'événement qui la déclenche.
- peut être conditionnée à l'aide de "gardes" : expressions booléennes, exprimées en langage naturel et encadrées de crochets.

état, transition et événement:



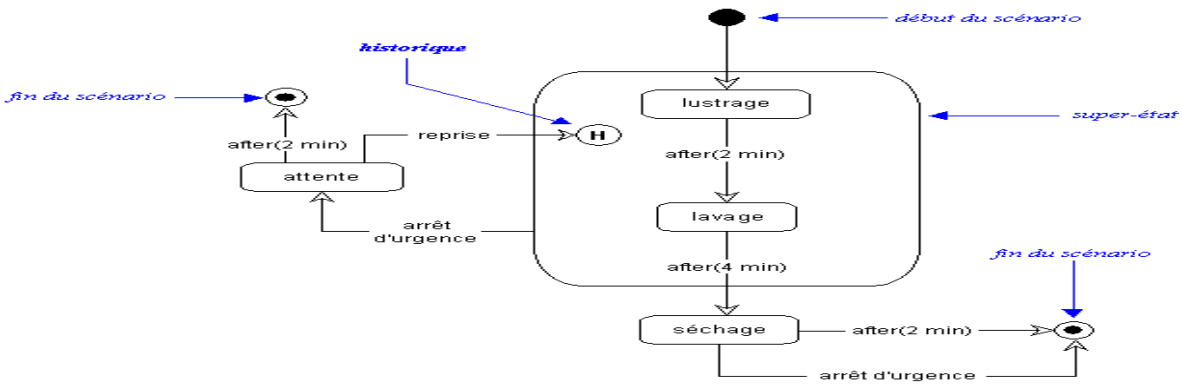
Transition conditionnelle :



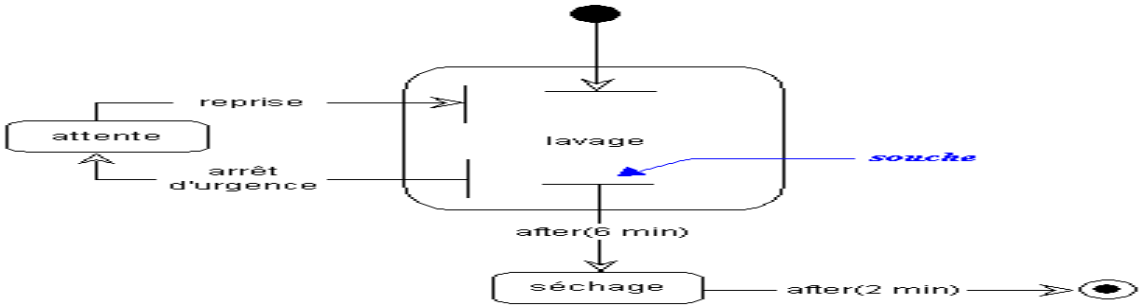
Nous allons maintenant introduire la notion d'hiérarchie entre les états et l'orthogonalité.

3.3 Notion d'hiérarchie entre les états : Super-Etat

C'est un élément de structuration des diagrammes d'états-transitions : un état (dit super état) peut englober d'autres états et transitions. Comme dans l'exemple de l'automate ci-dessous : les deux états « lustrage » et « lavage » sont regroupés dans un seul super état.
Le symbole de modélisation " H ou historique", mémorise le dernier sous-état actif d'un super-état, pour y revenir directement ultérieurement. L'état « lustrage » est l'état initial de l'automate.

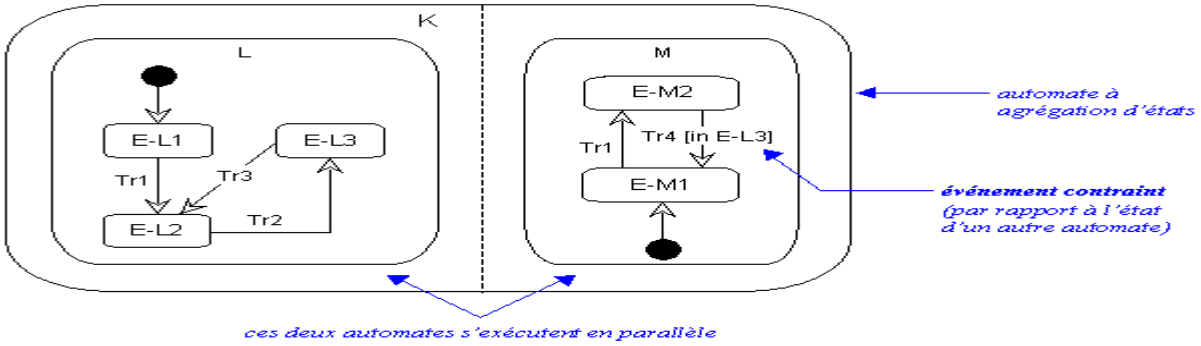


On peut introduire plus d'abstraction dans un diagramme d'états-transitions complexe grâce à la notion de souches : réduire la charge d'information, tout en matérialisant la présence de sous-états, à l'aide de souches.



3.4 Orthogonalité : Etats concurrents

Permet de modéliser le comportement d'objets concurrents sur un même diagramme d'états-transitions.



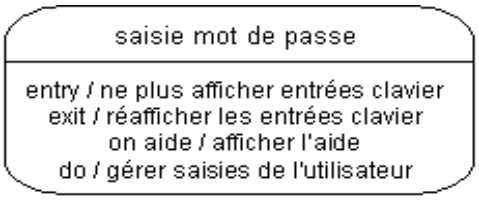
L'automate **K** est composé des sous-automates **L** et **M** qui s'activent simultanément et évoluent en parallèle. E-L1 (respectivement E-M1) est l'état initial du sous-automate L (respectivement M). Par conséquent, l'état initial de l'automate global K est le couple (E-L1,E-M1).

3.5 D'autres extensions

Actions dans un état

Une action correspond à une opération disponible dans l'objet dont on représente les états. Les actions propres à un état peuvent aussi être documentées directement à l'intérieur de l'état.

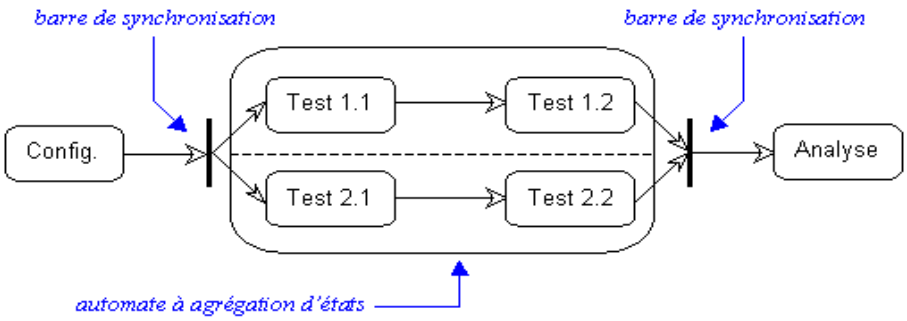
- entry / action : action exécutée à l'entrée de l'état
- exit / action : action exécutée à la sortie de l'état
- on événement / action : action exécutée à chaque fois que l'événement cité survient
- do / action : action récurrente ou significative, exécutée dans l'état



On peut associer une action à l'événement qui déclenche une transition : `événement / action`

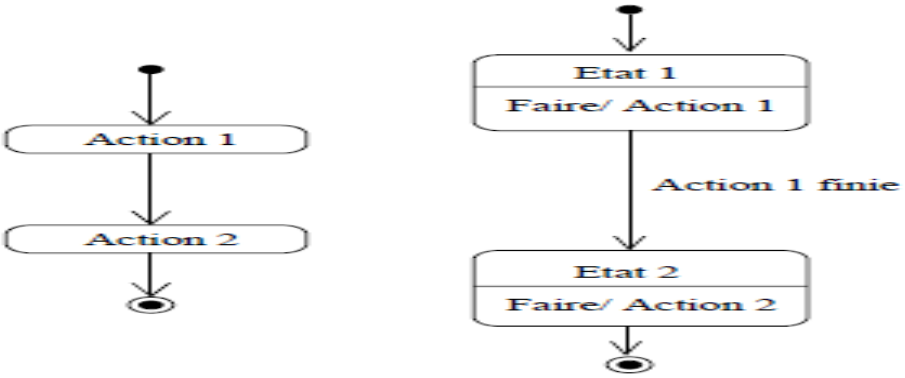
Barre de synchronisation

Permet de représenter graphiquement des points de synchronisation. Les transitions automatiques qui partent d'une barre de synchronisation ont lieu en même temps. On ne franchit une barre de synchronisation qu'après la réalisation de toutes les transitions qui s'y rattachent.

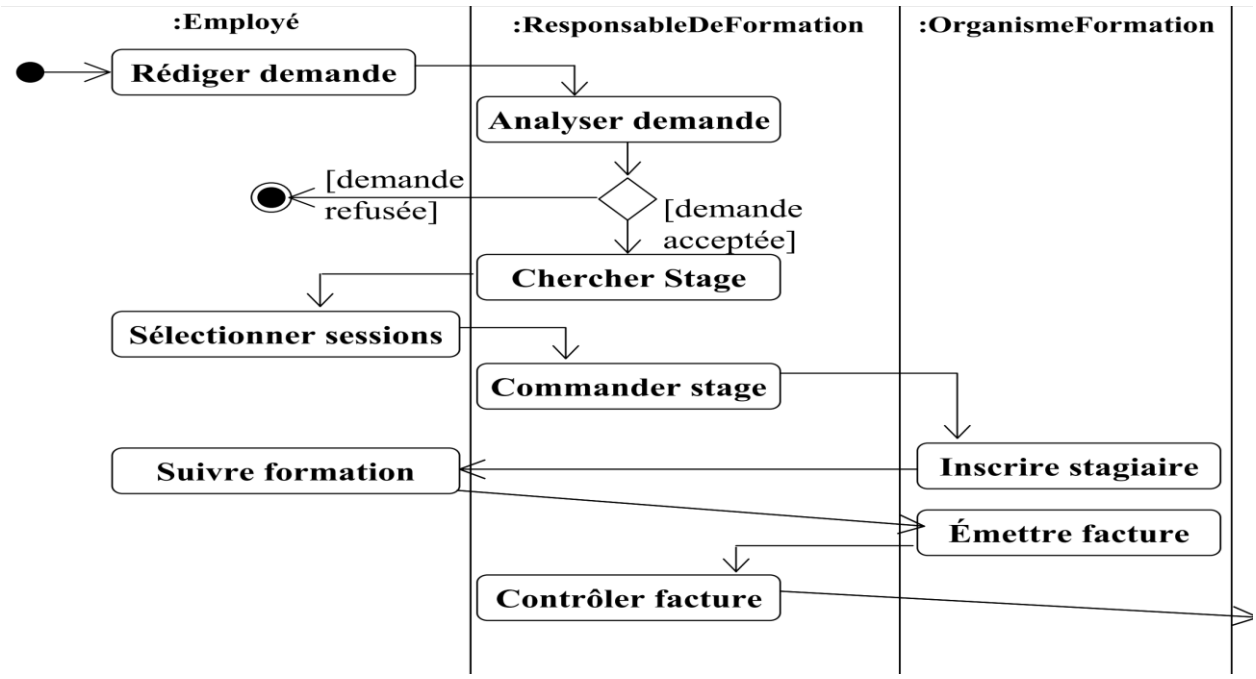


4. Diagramme d'Activités

Il permet de représenter le comportement interne d'une méthode, d'une opération ou d'un cas d'utilisation. Il correspond à la traditionnelle décomposition fonctionnelle.
Les activités sont reliées par des transitions automatiques qui peuvent être gardées par des conditions booléennes mutuellement exclusives. La figure suivante montre le lien entre un diagramme d'état (à droite) et un diagramme d'activités.



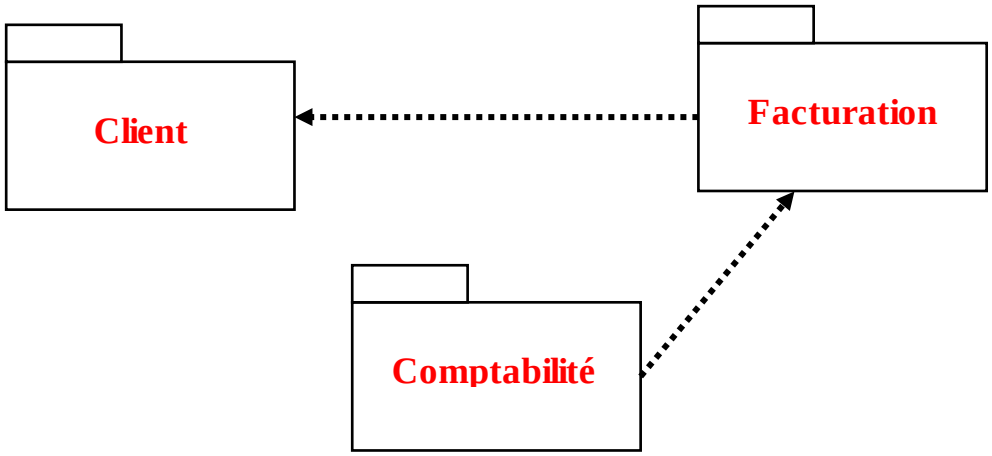
Voici un exemple d'un digramme d'activités qui modélise tout le processus ou la séquence d'actions déclenchée par un cas d'utilisation « demander une formation », initié par l'acteur employé et faisant intervenir deux autres acteurs (responsable de formation et organisme dispensant la formation).



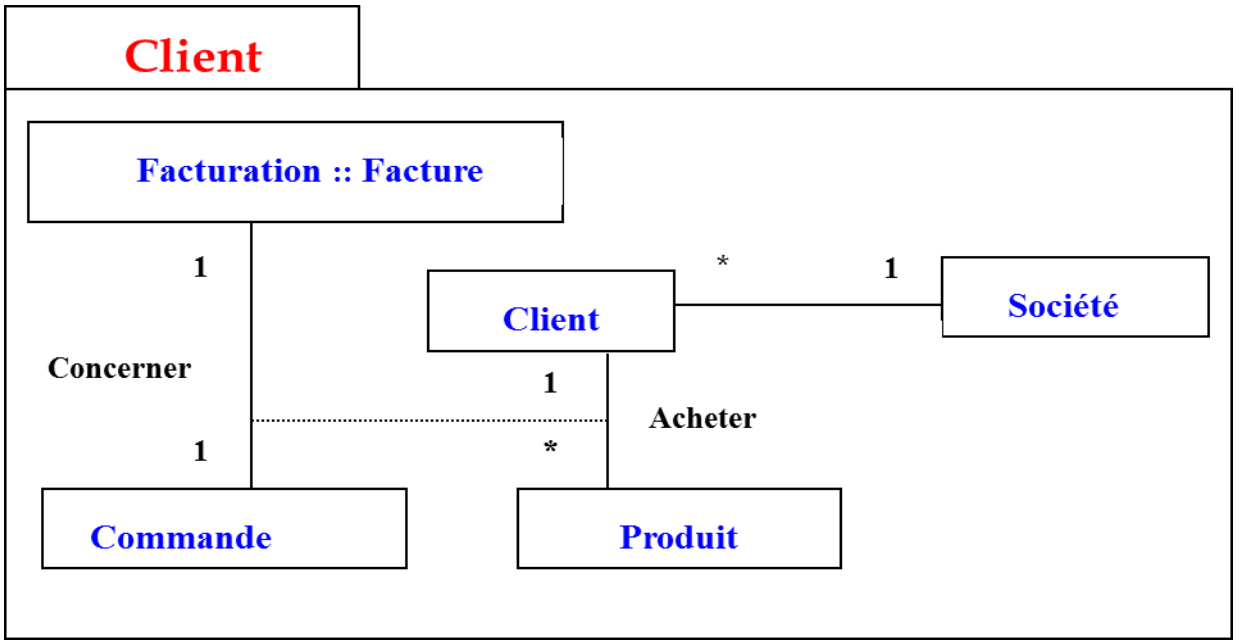
5. Notion de paquetage

Un paquetage permet de regrouper un ensemble de classes, d'associations, et éventuellement d'autres paquetages pour des raisons logiques. L'objectif est d'organiser les modèles de la même manière que les répertoires organisent les fichiers. Certains langages comme ADA, PL-SQL introduisent aussi le paquetage comme outil de modularité des programmes.

La figure suivante montre l'organisation d'un modèle de classe d'un système de gestion de commandes en 3 paquetages. Le lien entre 2 paquetages montre la relation de dépendance : un paquetage peut faire référence à une entité appartenant à un autre paquetage.



Une classe peut apparaître dans différents paquetages (avec le même nom). On peut aussi trouver des classes qui n'appartiennent pas au paquetage (comme Facture dans le paquetage Client) mais qui sont référencées par les classes propres. On désigner une classe d'un package par: nomPackage :: nomClasse. Attention, le mot Client est utilisé ici pour des raisons de simplicité pour nommer la classe Client, et aussi le paquetage qui l'encapsule.



Bibliographie

Alexi Drogoul : Conception Orientée Objet, Paris 6
Pierre-Alain Muller, Nathalie Gaertner: Modélisation avec UML, Eyrolles
Robert Ogor, Modélisation avec UML, ENST Bretagne, 2003.
Pascal Roques, UML 2 par la pratique, Eyrolles, 2008.