

La notation UML

RAZAFIMAHATRATRA Hajarisena
Docteur en Informatique

Année : 2024

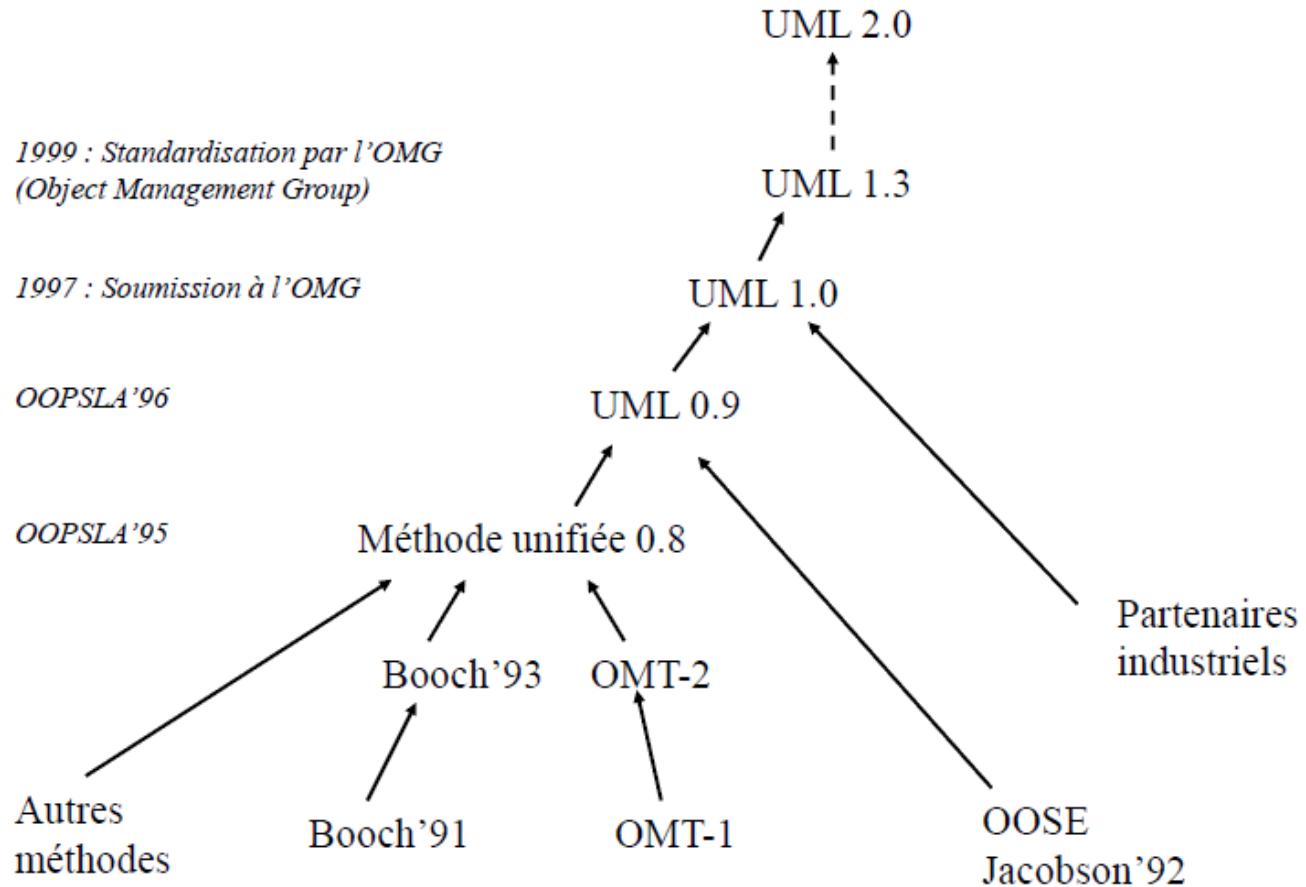
Qu'est-ce qu'UML ?

- UML = Unified Modeling Language ;
- UML = Langage uniifié pour la modélisation objet ;
- UML est un langage universel de modélisation objet ;
- UML est une notation, un outil de communication visuelle (diagrammes) ;
- UML est un langage de modélisation des applications construites à l'aide d'objets ;
- UML n'est pas une méthode ;
- UML n'est pas un langage de programmation ;
- UML n'est pas un processus de développement ;
- UML est une norme maintenue par l'OMG.

Historique d'UML (1/2)

- Issu en 1996 de la pratique industrielle et de la modélisation des systèmes logiciels ;
- Unification des méthodes objets dont les auteurs sont :
 - ✓ Ivar Jacobson (OOSE),
 - ✓ Grady Booch (BOOCH'93),
 - ✓ James Rumbaugh(OMT).
- Normalisation OMG en 1997 ;
- En 2005 : UML 2.0 ;
- En 2007 : UML 2.1.1 ;
- En 2009 : UML 2.2 ;
- En 2010 : UML 2.3 ;
- En 2011 : UML 2.4.

Historique d'UML (2/2)



Objectifs d'UML

- Proposer un langage visuel de modélisation :
 - ✓ utilisable par toutes les méthodes ;
 - ✓ adapté à toutes les phases du développement ;
 - ✓ compatible avec toutes les techniques de réalisation.
- Proposer des mécanismes d'extension et de spécialisation pour pouvoir étendre les concepts de base ;
- Être indépendant des langages particuliers de programmation ;
- Proposer une base formelle pour comprendre le langage de modélisation.

UML est un langage pour...

- Visualiser
 - ✓ chaque symbole graphique a une sémantique ;
- Spécifier
 - ✓ de manière précise et complète, sans ambiguïté ;
- Construire
 - ✓ les classes, les relations, ...
- Documenter
 - ✓ les diagrammes, notes, contraintes, exigences.

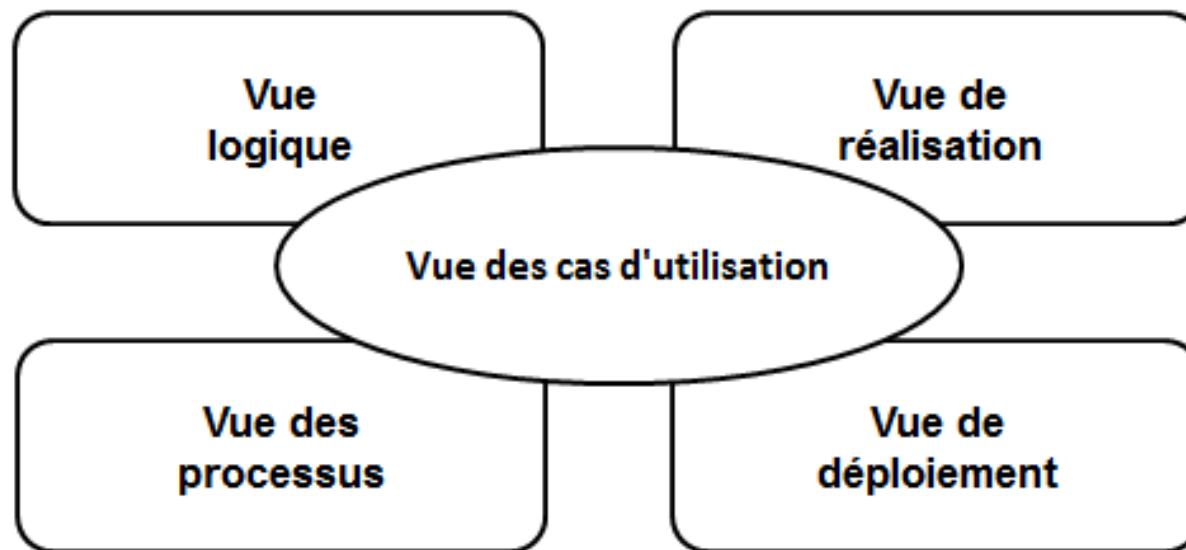
Les auteurs d'UML préconisent ...

Trois types de démarches :

- Itérative et incrémentale ;
- Guidée par les besoins des utilisateurs du système ;
- Centrée sur l'architecture logicielle.

Les « 4 + 1 » vues (1/6)

- Modèle d'architecture de Philippe Kruchten :



Les « 4 + 1 » vues (2/6)

Vue des cas d'utilisation :

- Guide toutes les autres ;
- Permet :
 - ✓ de trouver le bon modèle ;
 - ✓ d'expliquer et de justifier les choix qui ont guidé sa conception et son fonctionnement pour pouvoir le construire, le maintenir et le tester.

Les « 4 + 1 » vues (3/6)

Vue logique :

- Concerne l'intégrité de conception ;
- Se concentre sur l'abstraction et l'encapsulation ;
- Modélise les éléments et mécanismes principaux du système ;
- Identifie les éléments du domaine, les relations, et les interactions entre ces éléments ;
- Organise les éléments du domaine en catégories.

Les « 4 + 1 » vues (4/6)

Vue de réalisation :

- Concerne l'intégrité de gestion ;
- Exprime la perspective physique de l'organisation du code en termes de modules, de composants, et des concepts du langage ou de l'environnement d'implémentation.

Les « 4 + 1 » vues (5/6)

Vue des processus :

- Concerne l'intégrité d'exécution ;
- Très importante dans les environnements multitâches ;
- Exprime la perspective sur les activités concurrentes et parallèles.

Les « 4 + 1 » vues (6/6)

Vue de déploiement :

- Concerne l'intégrité de performance ;
- Exprime la répartition du système à travers un réseau des calculateurs et des nœuds logiques de traitements ;
- Utile pour décrire la distribution d'un système réparti.

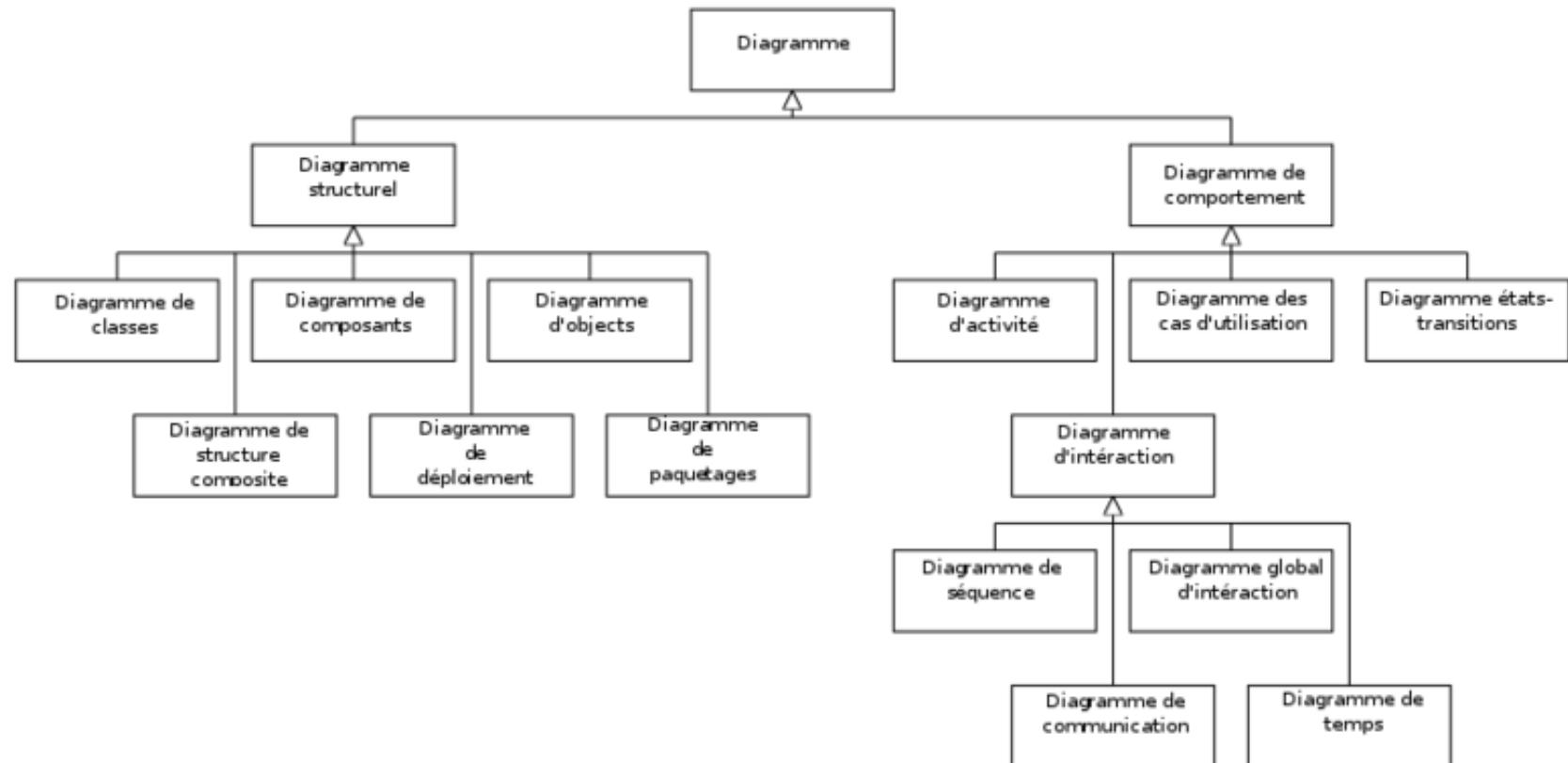
Briques de base d'UML

- Les éléments :
 - ✓ ce sont les abstractions essentielles à un modèle ;
- Les relations :
 - ✓ elles constituent des liens entre ces éléments ;
- Les diagrammes :
 - ✓ ils regroupent des éléments et des liens au sein de divers ensembles.

Diagrams d'UML (I/2)

- Diagrammes structurels / statiques (UML Structure) :
 - ✓ diagramme de classes (Class diagram)
 - ✓ diagramme d'objets (Object diagram)
 - ✓ diagramme de composants (Component diagram)
 - ✓ diagramme de déploiement (Deployment diagram)
 - ✓ diagramme de paquetages (Package diagram)
 - ✓ diagramme de structures composites (Composite structure diagram)
- Diagrammes comportementaux / dynamiques (UML Behavior) :
 - ✓ diagramme des cas d'utilisation (Use case diagram)
 - ✓ diagramme d'activités (Activity diagram)
 - ✓ diagramme d'états-transitions (State machine diagram)
 - ✓ diagrammes d'interaction (Interaction diagram):
 - diagramme de séquence (Sequence diagram)
 - diagramme de communication (Communication diagram)
 - diagramme global d'interaction (Interaction overview diagram)
 - diagramme de temps (Timing diagram)

Diagrammes d'UML (2/2)





DIAGRAMMES COMPORTEMENTAUX



DIAGRAMMES COMPORTEMENTAUX

DIAGRAMME DES CAS D'UTILISATION

Diagramme des cas d'utilisation...?

- Décrit les fonctionnalités d'un système d'un point de vue utilisateur, sous la forme d'actions et de réactions ;
- Permet de définir les limites du système et ses relations avec l'environnement ;
- Comprendre les besoins de l'utilisateur ;
- Sert à modéliser les aspects dynamiques d'un système ;
- Fait ressortir les acteurs et les fonctions offertes par le système.

Système

- Permet de délimiter le sujet de l'étude ;
- Possède :
 - ✓ les cas d'utilisation,
 - ✓ mais pas les acteurs.

Nom du système

Acteur (1/3)

- Un rôle joué par une entité extérieure au système ;
- Interactions directes avec le système ;
- Humain, matériel externe, autre système ;

- Comment identifier les acteurs ?
- ✓ Qui ont besoins le système pour réaliser le travail ?
- ✓ Qui vont exécuter les fonctions principales du système (maintenance et administration) ?
- ✓ Est-ce que le système interagit avec le matériel ou autre système ?
- ✓ ...



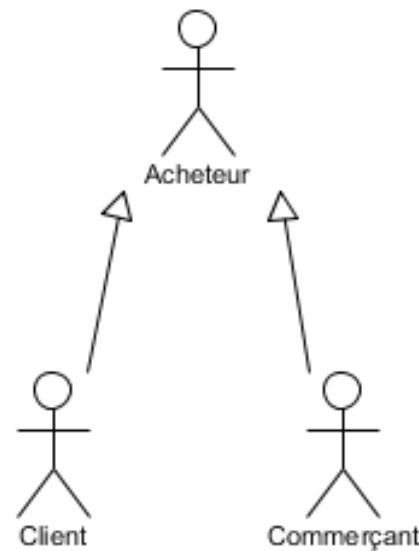
<<actor>>
Acteur non humain

Acteur (2/3)

- Trois types d'acteurs :
 - ✓ Humain : utilisateur du système à travers son interface graphique (client, guichetier, vendeur..) ;
 - ✓ Matériel : entité matérielle qui exploite les données du système ou qui est piloté par le système (imprimante, robots, automates,...) ;
 - ✓ Logiciel : entité logicielle existante et fonctionnelle qui communique avec le système grâce à une interface logicielle (application de gestion, base de données,...).
- Du point de vue système :
 - ✓ Les acteurs principaux : pour lesquels l'objectif du cas d'utilisation est essentiel ;
 - ✓ Les acteurs secondaires : interagissent avec le cas d'utilisation mais dont l'objectif n'est pas essentiel.

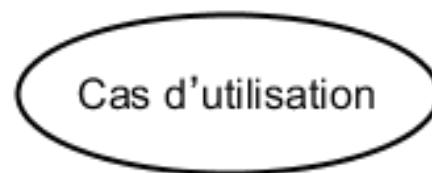
Acteur (3/3)

- Relation entre acteurs : généralisation/specialisation
- ✓ Dans le cas qu'un acteur peut être une spécialisation d'un autre acteur déjà défini.

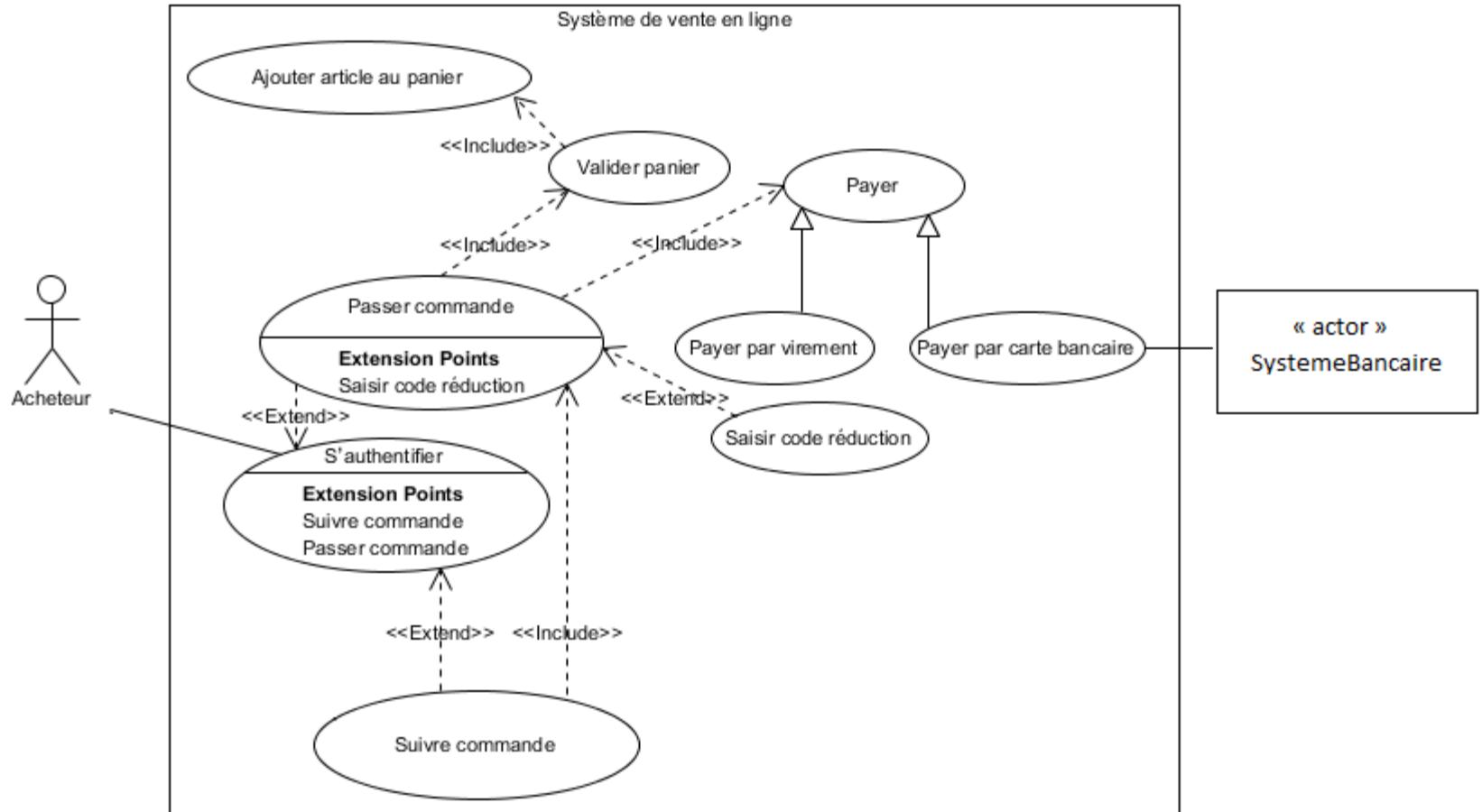


Cas d'utilisation

- Un moyen de représenter les différentes possibilités d'utiliser un système ;
- Exprime toujours une suite d'interactions entre un acteur et l'application ;
- Définit une fonctionnalité utilisable par un acteur ;
- Décrit par une forme verbale (Verbe à l'infinitif) ;
- Comment identifier les cas d'utilisation ?
 - ✓ Qu'est ce que l'acteur attend de l'application ?
 - ✓ Quelles informations l'acteur doit-il créer, sauvegarder, consulter, modifier, détruire ?
 - ✓ Y-a-t-il des événements externes que l'application doit connaître ? Quels acteurs l'en informent ?
 - ✓ ...



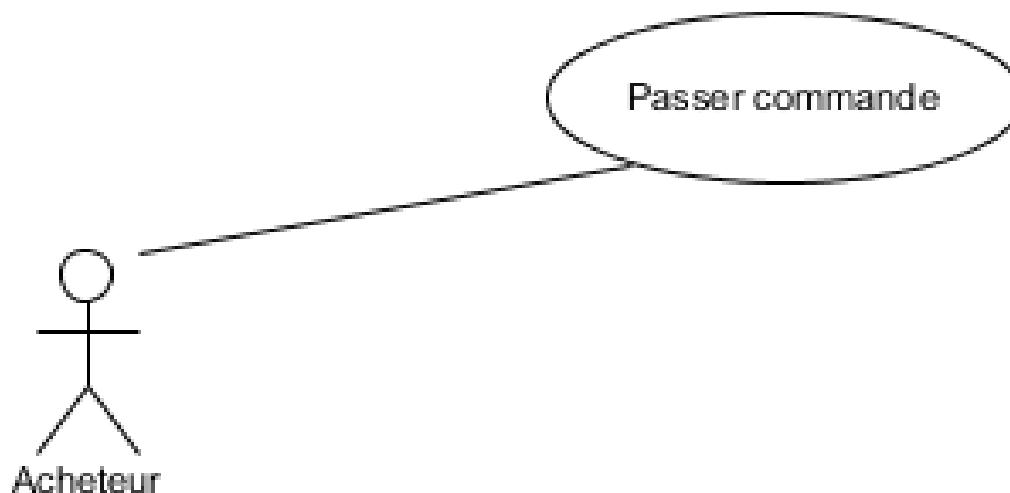
Exemple d'un diagramme des cas d'utilisation



Relation entre acteur et cas d'utilisation

Relation d'association :

- Représente l'interaction du système avec l'extérieur.

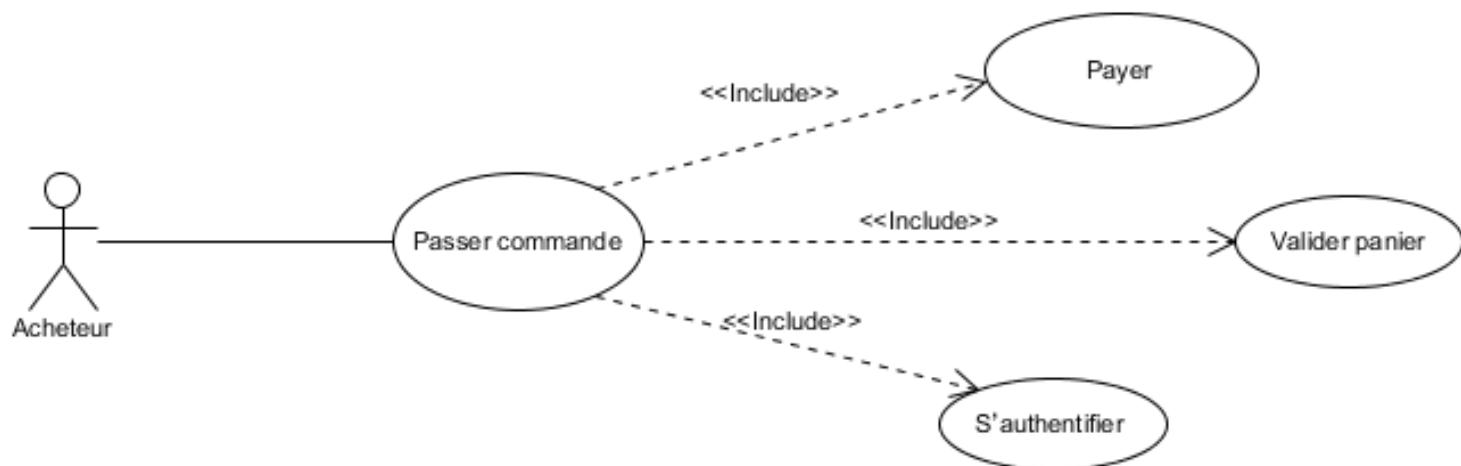


Relations entre cas d'utilisation

(1/4)

Relation d'inclusion (include) :

- Le cas d'utilisation de base en incorpore explicitement un autre, de façon obligatoire.

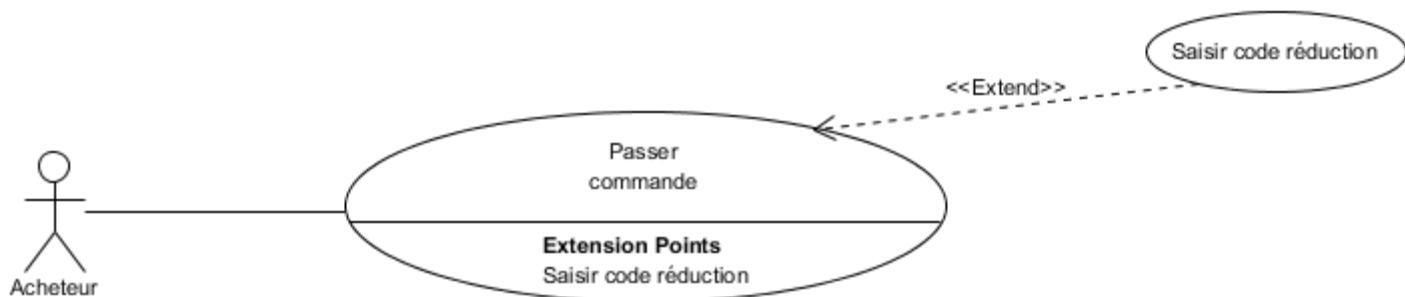


Relations entre cas d'utilisation

(2/4)

Relation d'extension (extend) :

- Le cas d'utilisation de base en incorpore implicitement un autre, de façon optionnelle.

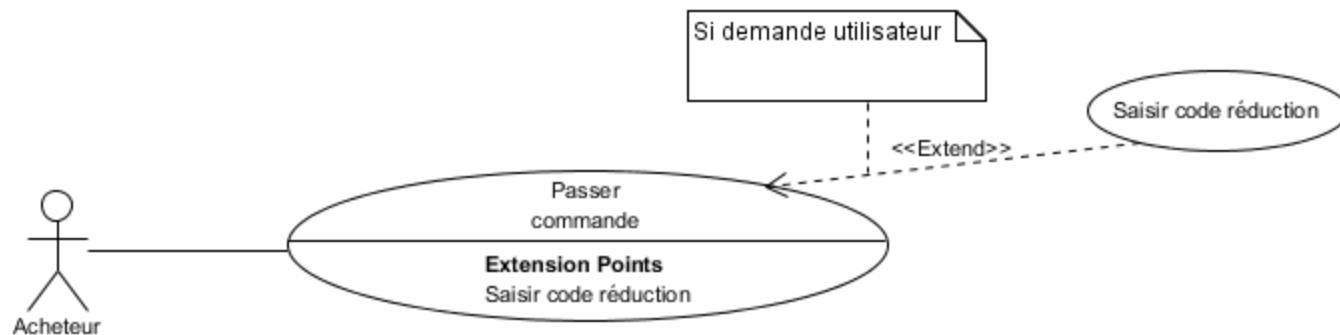


Relations entre cas d'utilisation

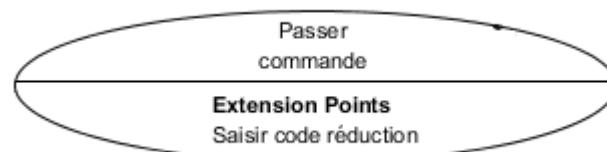
(3/4)

Relation d'extension (extend) :(suite)

- Condition d'extension nécessaire ;



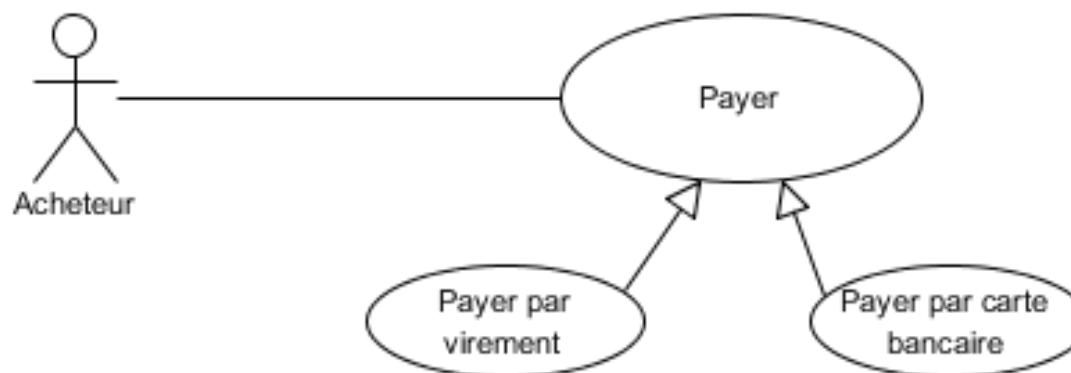
- Liste éventuelle des points d'extension pour préciser le moment auquel intervient l'extension.



Relations entre cas d'utilisation (4/4)

Relation de généralisation/spécialisation :

- Pour un cas particulier, la relation se traduit soit par la généralisation/spécialisation.



Description textuelle d'un cas d'utilisation (1/4)

- Permet de :
 - ✓ clarifier le déroulement de la fonctionnalité ;
 - ✓ décrire la chronologie des actions qui devront être réalisées.
- Pas normalisée par UML ;
- Mais beaucoup d'informations à structurer...;
- Plusieurs propositions dans la littérature ;
- Chaque cas d'utilisation doit être décrit en détail ;
- Commencer par les cas d'utilisation prioritaires.

Description textuelle d'un cas d'utilisation (2/4)

Les informations à décrire :

- Précondition : quand et comment le cas d'utilisation débute ;
- Postcondition : quand et comment le cas d'utilisation se termine ;
- Scénario nominal : description du fonctionnement du cas d'utilisation dans le cas du scenario le plus probable ;
- Scénario alternatif : description du fonctionnement du cas d'utilisation pour une partie de scenario différente du scenario le plus probable ;
- Scénario d'exception : description du fonctionnement du cas d'utilisation pour une partie du scenario supplémentaire et optionnelle ;
- Les variantes possibles et les cas d'erreurs ;
- Les exigences non fonctionnelles.

Description textuelle d'un cas d'utilisation (3/4)

Exemple pour le cas d'utilisation «Payer par carte bancaire»

- Acteurs principaux : Acheteur (Client, Commerçant)
- Acteurs secondaires : Système Bancaire
- Précondition : L'acheteur a validé sa commande.
- Début : L'acheteur arrive sur la page de paiement.
- Postcondition : La commande est validée, le compte de l'entreprise est crédité.
- Fin : ...
- Scénario nominal :
 1. L'acheteur choisit de faire le mode paiement par la carte bancaire.
 2. Le système demande à l'acheteur les informations sur sa carte bancaire.
 3. L'acheteur saisit les informations sur sa carte bancaire.
 4. Le système vérifie que le numéro de carte bancaire est correct.
 5. Le système demande l'autorisation de prélèvement au système bancaire.
 6. Le système bancaire valide la transaction.
 7. Le système indique à l'acheteur que la commande est confirmée.

Description textuelle d'un cas d'utilisation (4/4)

- Scénario alternatif :
 - ✓ A1: Le numéro de la carte bancaire est incorrect.
 - ✓ Démarrage à l'étape 3 du scénario nominal.
 1. L'acheteur recommence de saisir les informations sur sa carte bancaire.
 2. ...
 - ✓ A2 : ...
 - ✓ ...
-
- Scénario d'exception :
 - ✓ E1 : L'acheteur annule le paiement par carte bancaire.
-
- Contraintes non fonctionnelles :
 - ✓ Sécurité et confidentialité des échanges ;
 - ✓ ...

Conseils (1/3)

- Un diagramme de cas d'utilisation doit toujours rester simple;
- Donner un nom parlant (verbe à l'infinitif) à chaque cas d'utilisation;
- Compléter la spécification du cas d'utilisation:
 - ✓ Par une description textuelle;
 - ✓ Eventuellement à l'aide du diagramme d'activité.
- Ne pas modéliser à un niveau de détail trop précis:

~~Entrer le nom du client~~

~~Entrer le prénom du client~~

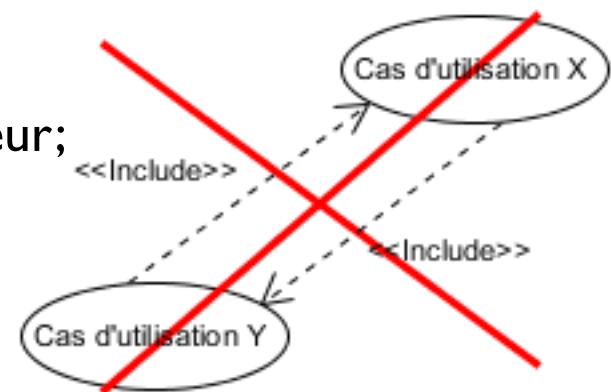
- Ne pas modéliser en dehors de l'application:

~~Choix des articles commandés~~

...

Conseils (2/3)

- Fragmenter les cas d'utilisation si:
 - ✓ Interactions trop complexes;
 - ✓ Isolation de parties indépendantes possible.
- Il doit toujours y avoir au moins un acteur par cas d'utilisation:
 - ✓ Explicite pour les cas d'utilisation de plus haut niveau;
 - ✓ Via les relations d'extension et d'inclusion pour les autres.
- Présentation des cas d'utilisation:
 - ✓ Par séquence de déclenchement / acteur;
 - ✓ Par hiérarchisation;
 - ✓ Par domaine d'activité.
- Eviter:
 - ✓ De décomposer les cas d'utilisation / classes;
 - ✓ Les cycles.



Conseils (3/3)

- Représentation du comportement normal de l'application mais aussi:
 - ✓ Des variations possibles;
 - ✓ Des situations exceptionnelles.
- Les cas d'utilisation doivent générer un résultat.
- Les cas d'utilisation doivent traduire ce que fait l'application:
 - ✓ Vision claire;
 - ✓ Organiser les cas d'utilisations et les diagrammes.
- Vision complète:
 - ✓ Représenter un maximum de choses;
 - ✓ En respectant et en utilisant la notation choisie.



DIAGRAMMES COMPORTEMENTAUX

DIAGRAMME DE SÉQUENCE

Diagramme de séquence...?

- Représente les interactions entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages ;
- Documente un ou plusieurs scénarios d'un cas d'utilisation ;
- Montre :
 - ✓ les réactions du système aux actions des utilisateurs ;
 - ✓ la création et la manipulation des objets.
- Utile en phase d'analyse, de conception et de test.

Objet

- Représenté par un rectangle contenant le nom de l'objet.

Syntaxe d'un objet :

[<nomObjet>] : [<NomClasse>]

Nommant format	Notation
Un objet d'une classe non spécifiée.	objet : objet
Un objet nommé d'une classe spécifiée.	objetX : Classe
Un objet sans nom d'une classe spécifiée.	: Classe

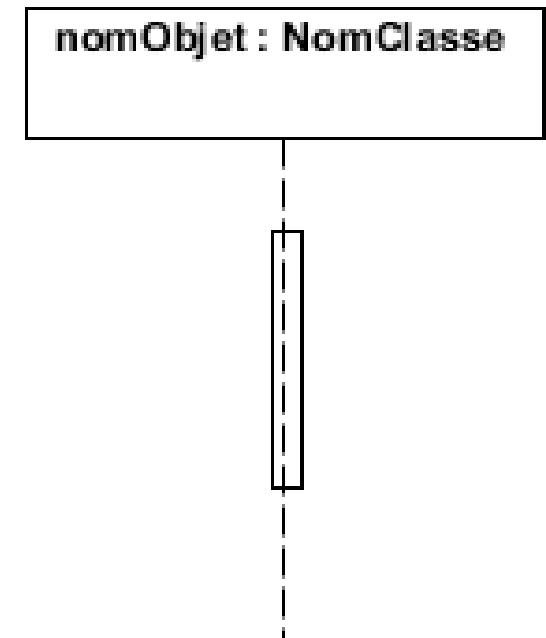
Ligne de vie

- Représente l'ensemble des opérations exécutées par un objet ;
- Ligne verticale pointillée dirigée vers le bas à partir de chaque objet ;
- Symboliser une durée qui dépend du scénario et du comportement modélisé.

nomObjet : NomClasse

Barre d'activation

- Un rectangle qui remplace la ligne de vie ;
- Traduire l'activation de l'objet.



Message (1/5)

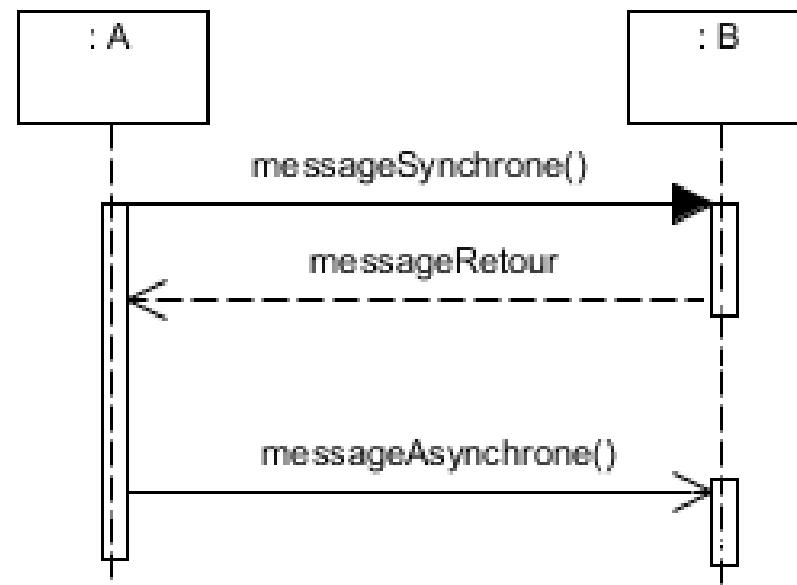
- Généralement un appel, un signal ou une réponse ;
- Représentés par des flèches horizontales reliant la ligne de vie de l'objet émetteur à la ligne de vie de l'objet récepteur ;
- Syntaxe d'un message :
[' 'Condition' ' [séquence] [* [| |] ' ' itération' '] :] [résultat:=] message ([paramètres])

Message (2/5)

- Message synchrone : l'objet émetteur reste bloqué le temps lorsque l'objet récepteur traite le message envoyé ;
- Message asynchrone : l'objet émetteur n'est pas bloqué lorsque l'objet récepteur traite le message envoyé ;
- Message de retour de l'invocation d'une méthode n'est pas systématique, toutes les méthodes ne retournant pas un résultat ;

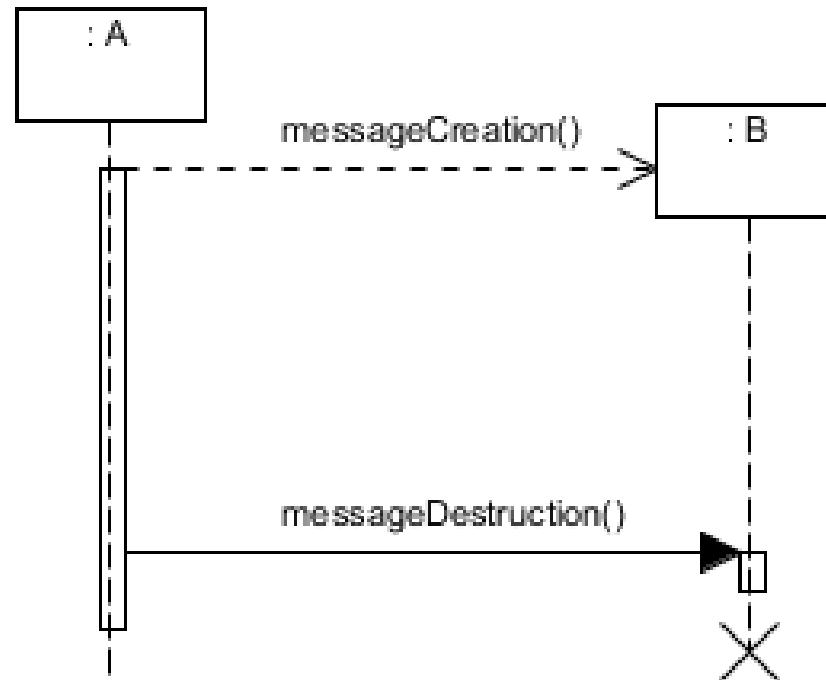
Syntaxe d'un message retour :

[<attribut> =] message [: <valeur_de_retour>]



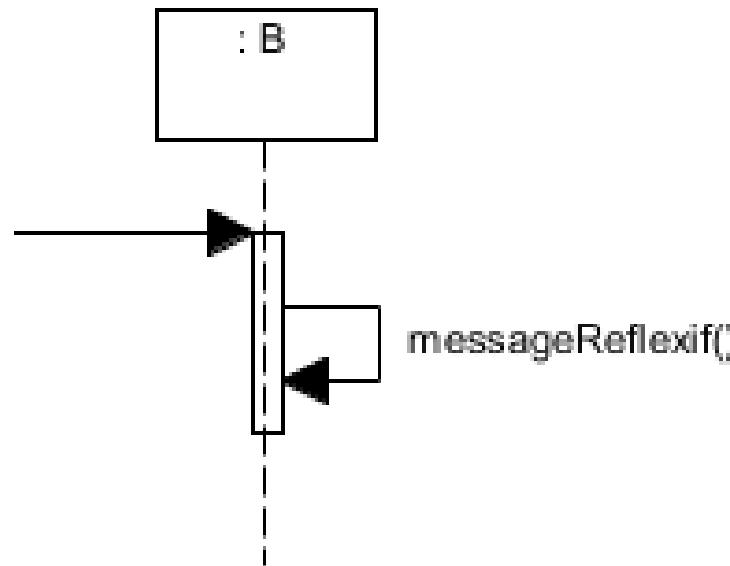
Message (3/5)

- Message de création : un message spécifique qui donne lieu au début de la ligne de vie du nouvel objet ;
- Message de destruction : un message envoyé à un objet existant et qui donne lieu à la fin de sa ligne de vie.



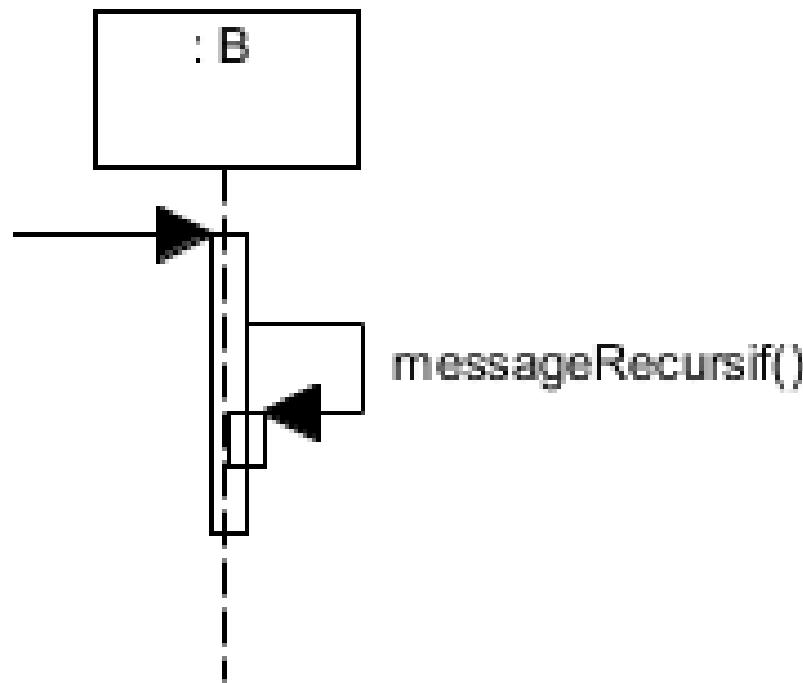
Message (4/5)

- Message réflexif : un objet peut envoyer un message à lui-même.



Message (5/5)

- Message récursif se représente par un dédoublement de la barre d'activation.

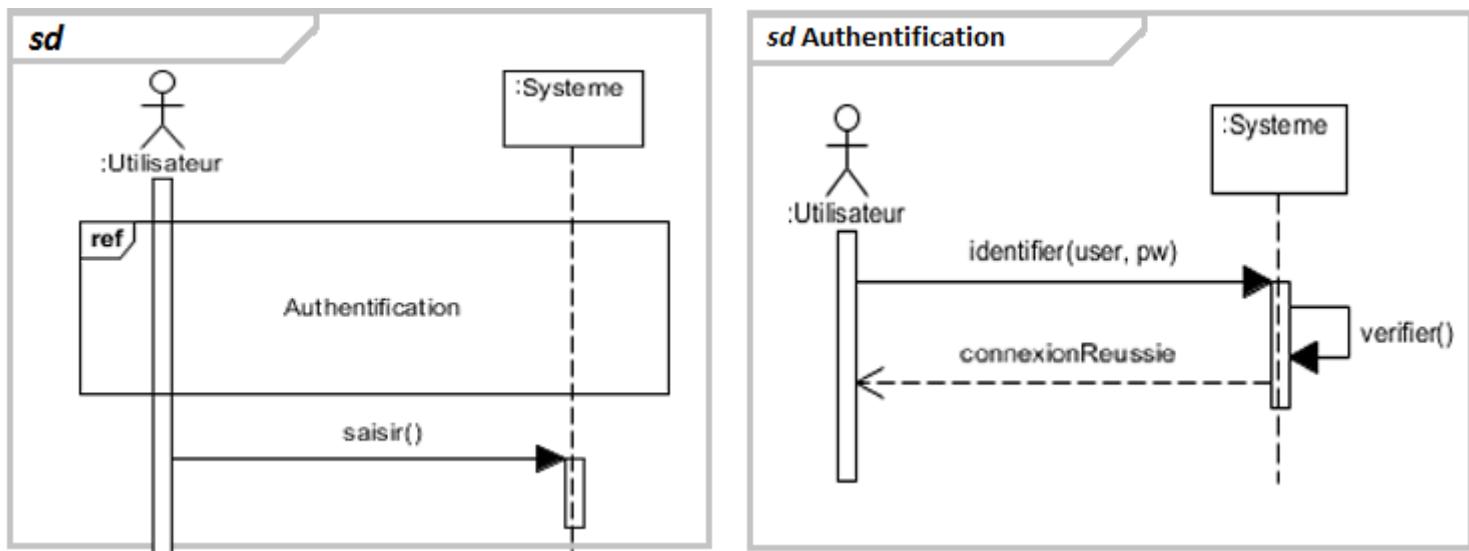


Fragments combinés (1/15)

- Regroupements logiques représentés par un rectangle et contenant les structures conditionnelles qui affectent le flux de messages ;
- Pour décrire les diagrammes de séquence de manière compacte ;
- Définis par un opérateur et des opérandes ;
- L'opérateur conditionne la signification du fragment combiné ;
- Les diagrammes peuvent se référencer les uns les autres :
 - ✓ Pointeur / raccourci vers un diagramme de séquence complètement décrit ;
 - ✓ Pour factoriser les parties de comportements à plusieurs endroits.



Fragments combinés (2/15)



Fragments combinés (3/15)

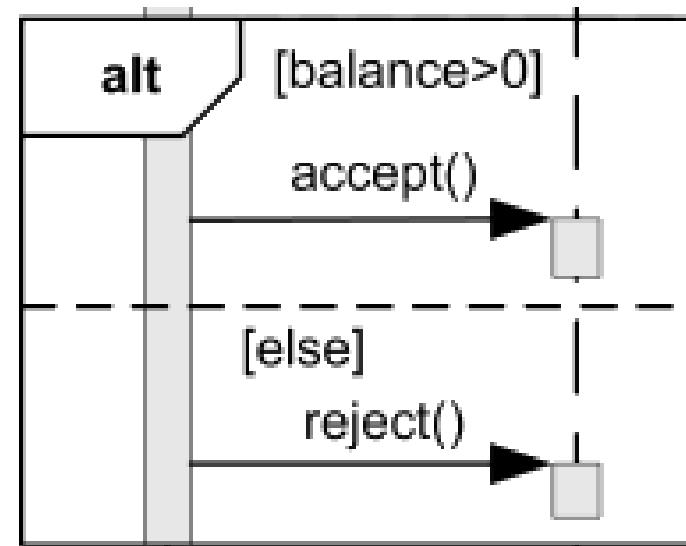
Différents fragments combinés d'opérateur d'interaction :

- Les opérateurs de base :
 - ✓ **alt**
 - ✓ **opt**
 - ✓ **loop**
 - ✓ **par**
 - ✓ **seq**
 - ✓ **strict**
- Les opérateurs pour les situations exceptionnelles :
 - ✓ **break**
- Les opérateurs pour les tests:
 - ✓ **critical**
 - ✓ **ignore**
 - ✓ **consider**
 - ✓ **negative**
 - ✓ **assert**

Fragments combinés (4/15)

Les opérateurs de base

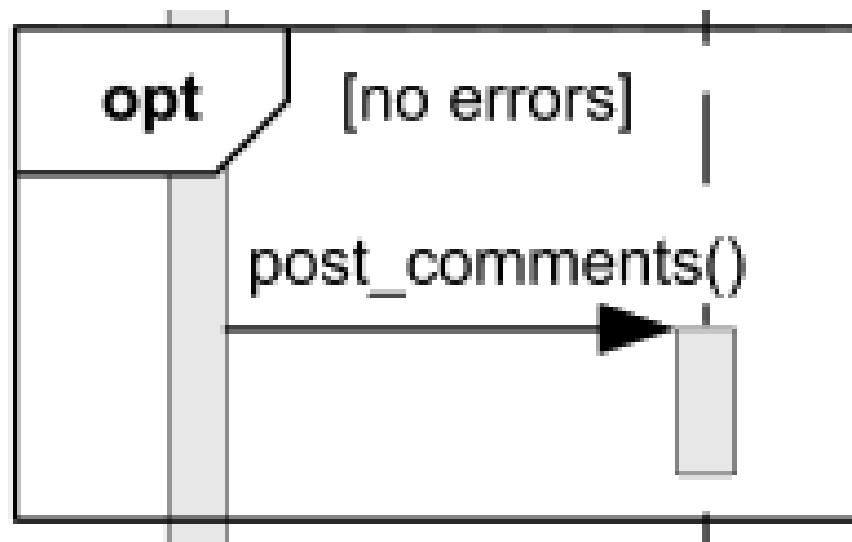
- Operateur « Alternative » ou alt:
 - ✓ Exprime la possibilité de choisir différents comportements ;
 - ✓ Réalisé par le choix d'un unique opérande d'interaction en fonction des contraintes d'interaction.



Fragments combinés (5/15)

Les opérateurs de base (suite)

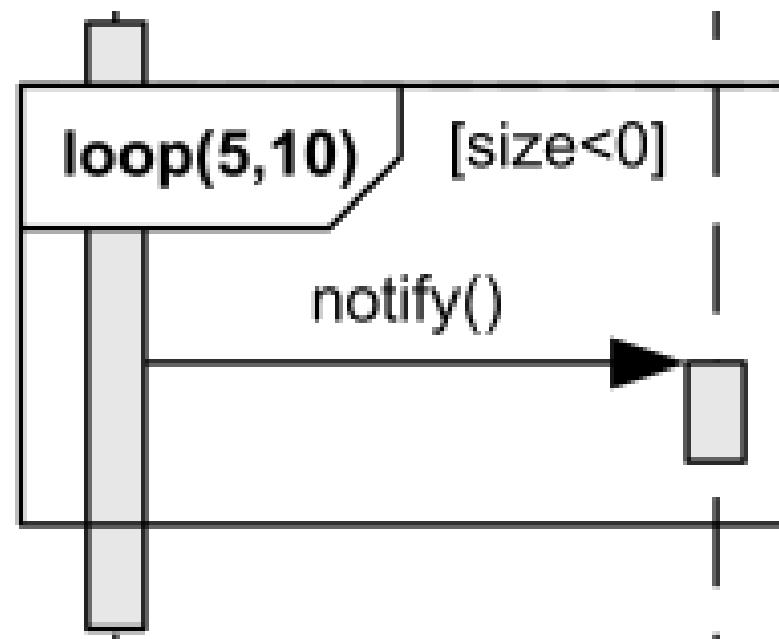
- Operateur « Option » ou opt:
 - ✓ Représente un choix dans le comportement où soit seul l'opérande d'interaction contenu dans le fragment combiné s'exécute, soit rien ne se produit ;
 - ✓ Correspond à un **alt** sans **else**.



Fragments combinés (6/15)

Les opérateurs de base (suite)

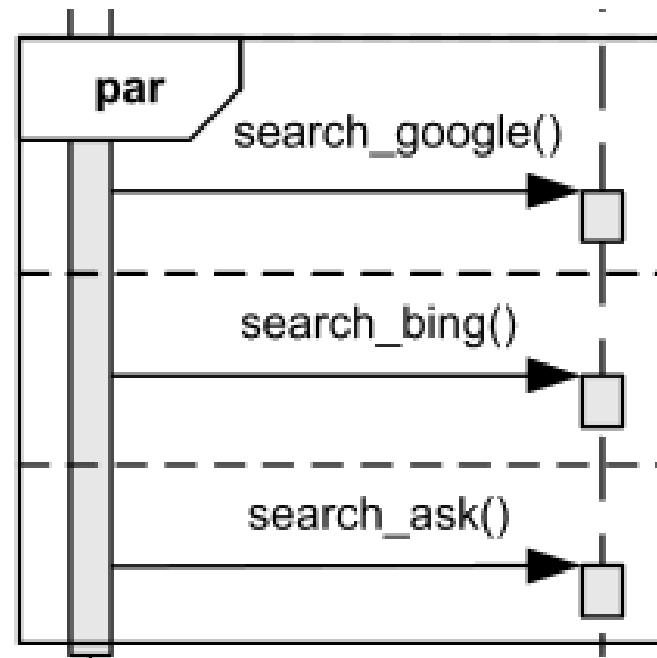
- Operateur loop:
 - ✓ Représente une boucle ;
 - ✓ L'opérande d'interaction sera répété un certain nombre de fois.



Fragments combinés (7/15)

Les opérateurs de base (suite)

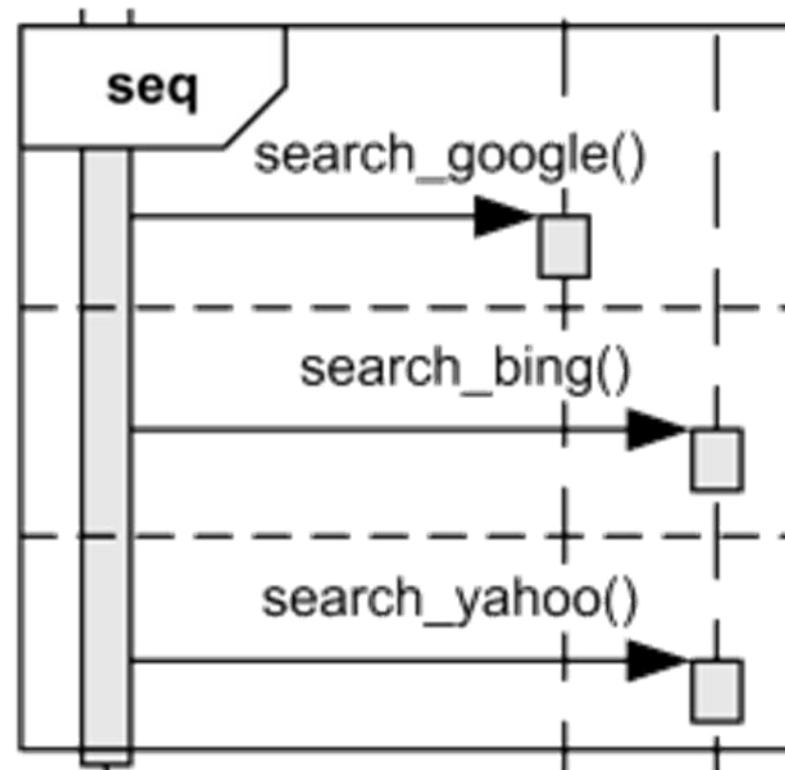
- Operateur « Parallel » ou `par`:
- ✓ Désigne un interclassement (ou entrelacement) entre les comportements des opérandes ;
- ✓ Les occurrences des évènements des divers opérandes d'interaction peuvent être entrelacées de toutes les façons tant que l'ordre impose par chaque opérande est préservé.



Fragments combinés (8/15)

Les opérateurs de base (suite)

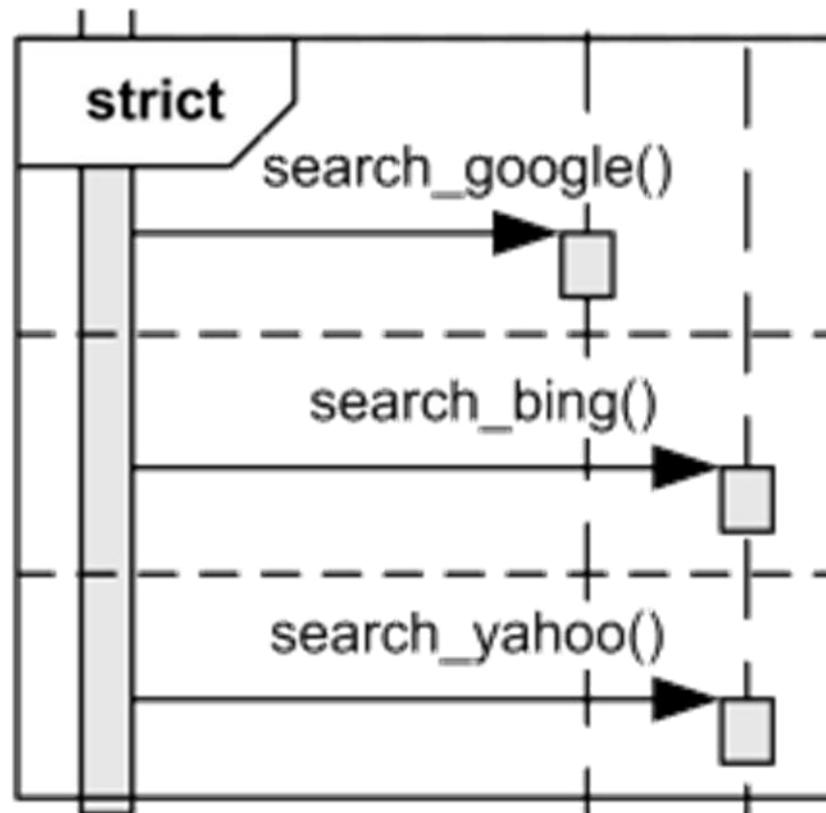
- Operateur seq :
 - ✓ Désigne un entrelacement faible entre les comportements des opérandes.



Fragments combinés (9/15)

Les opérateurs de base (suite)

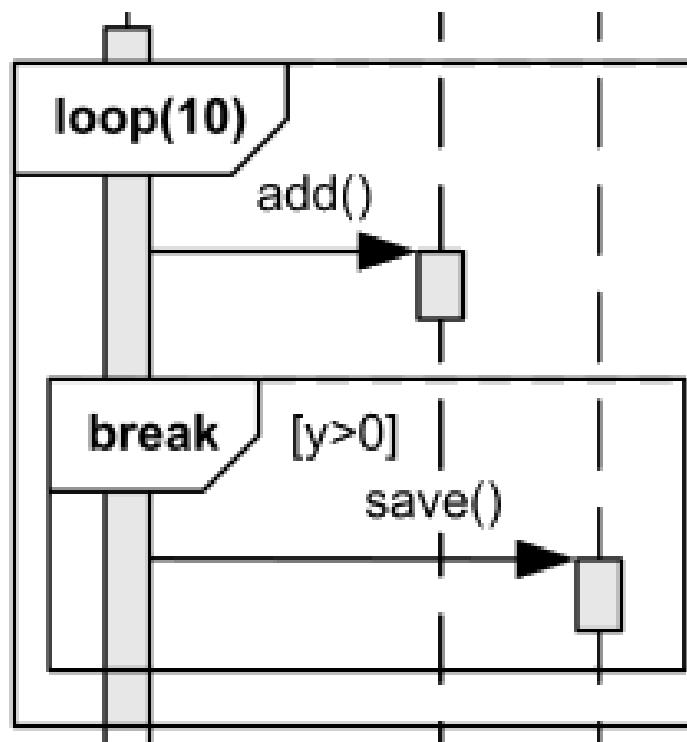
- Operateur strict :
 - ✓ Définit un séquencement strict entre les comportements des opérandes.



Fragments combinés (10/15)

Les opérateurs pour les situations exceptionnelles

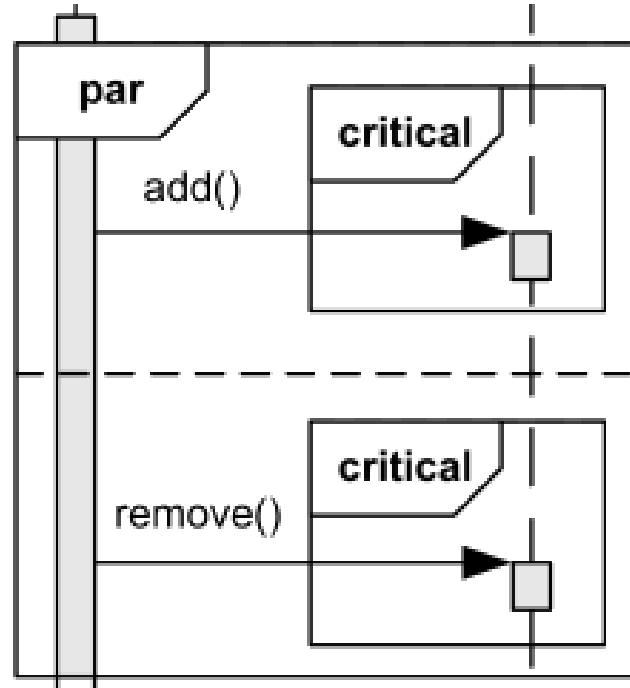
- Operateur break :
 - ✓ Représente un scénario d'arrêt qui est exécuté à la place du reste du fragment d'interaction englobant.



Fragments combinés (11/15)

Les opérateurs pour les tests

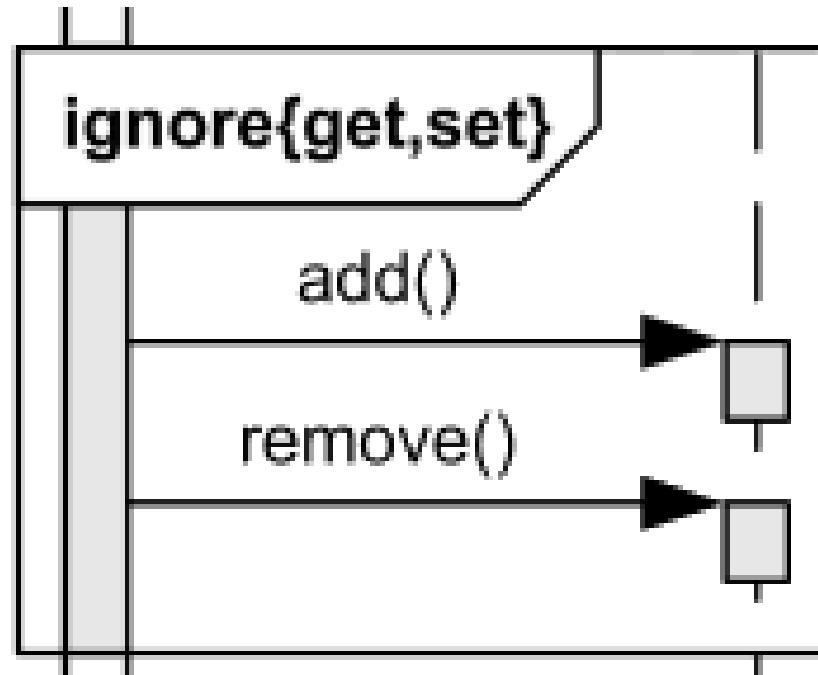
- Opérateur critical :
 - ✓ Représente une région critique ;
 - ✓ Les occurrences d'évènement ne peuvent pas être entrelacées avec les traces de la région critique ;
 - ✓ La région doit être traitée de manière atomique.



Fragments combinés (12/15)

Les opérateurs pour les tests (suite)

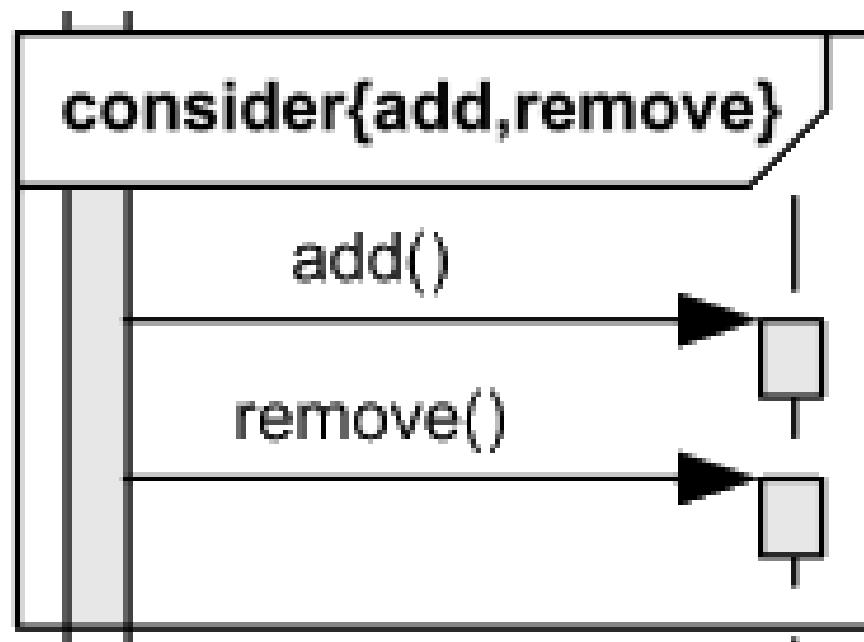
- Operateur ignore :
 - ✓ Indique que les types de certains messages sont ignorés dans le fragment combiné ;
 - ✓ Les messages ignorés peuvent apparaître à n'importe quel endroit de la trace.



Fragments combinés (13/15)

Les opérateurs pour les tests (suite)

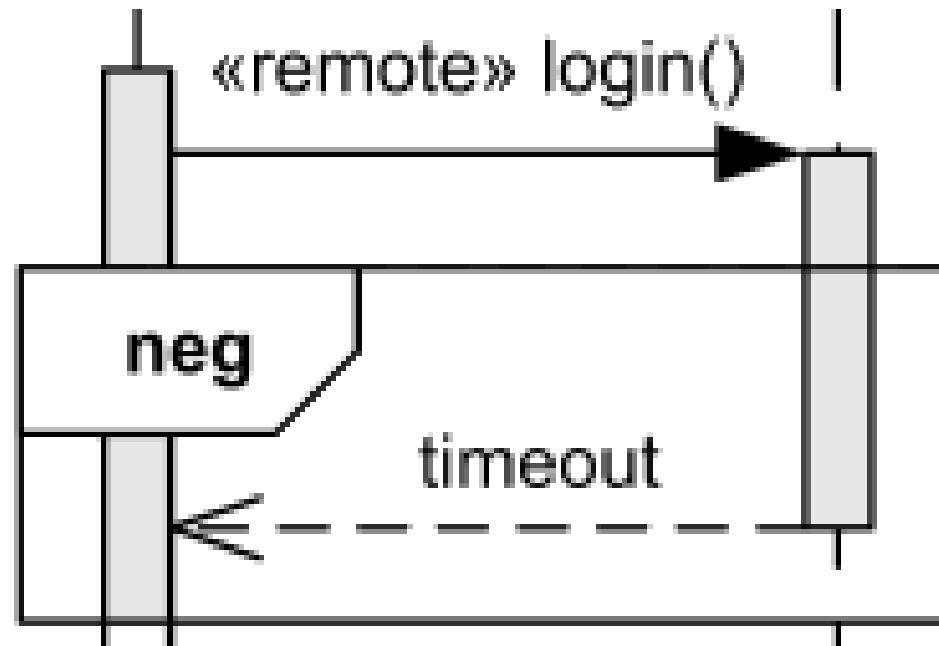
- Operateur consider :
 - ✓ Seuls certains messages vont être considérés à l'intérieur du fragment combiné.



Fragments combinés (14/15)

Les opérateurs pour les tests (suite)

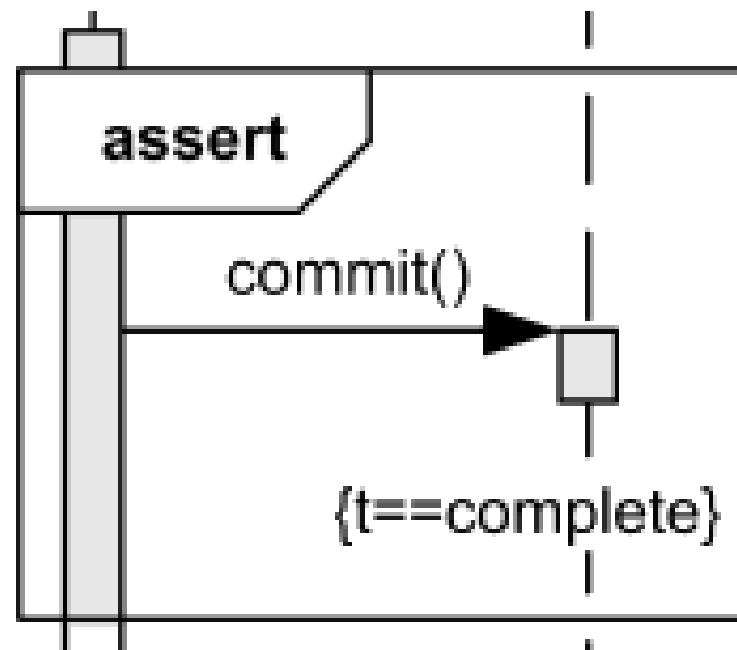
- Operateur « Negative » ou neg :
 - ✓ Définit des traces invalides.



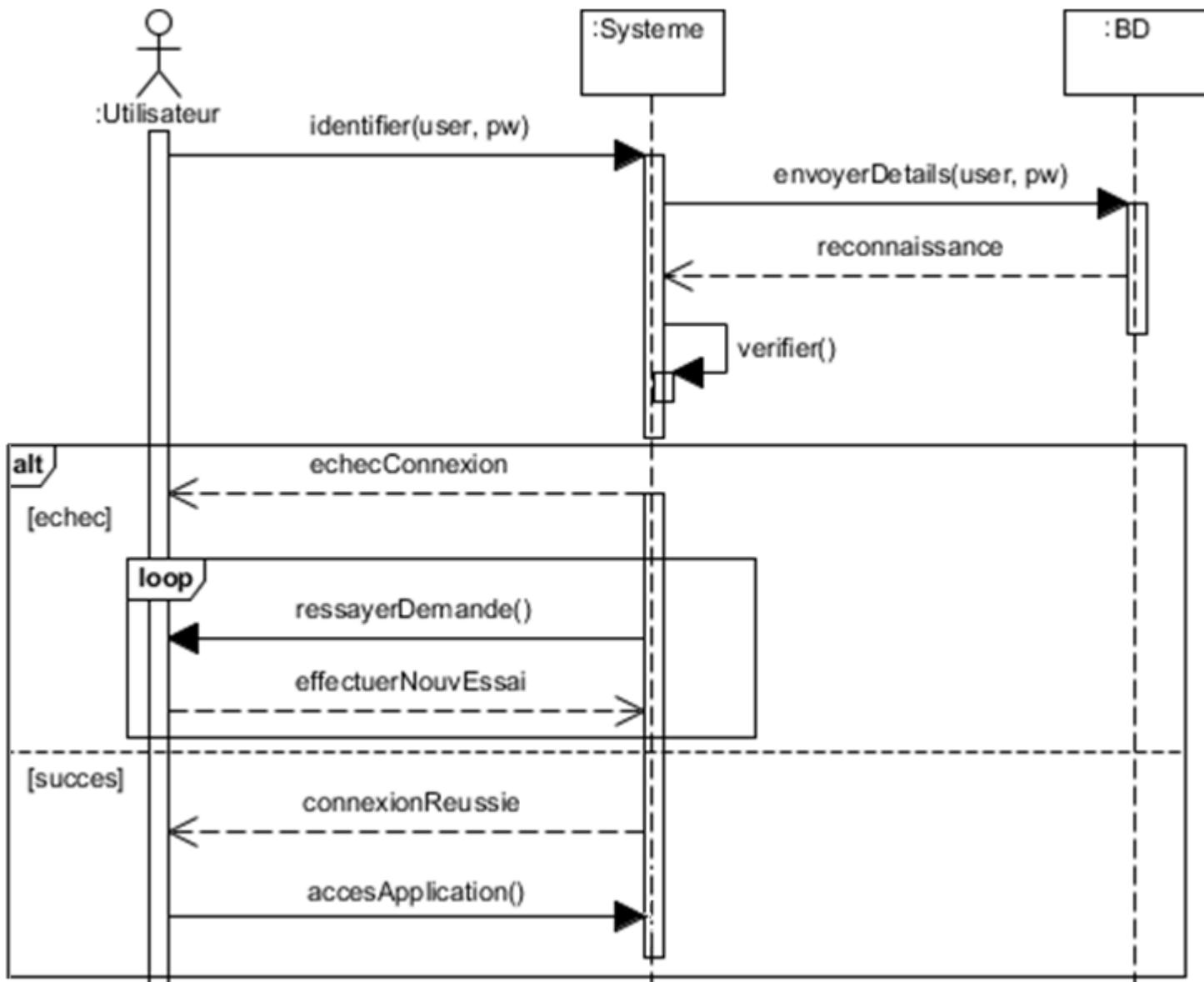
Fragments combinés (15/15)

Les opérateurs pour les tests (suite)

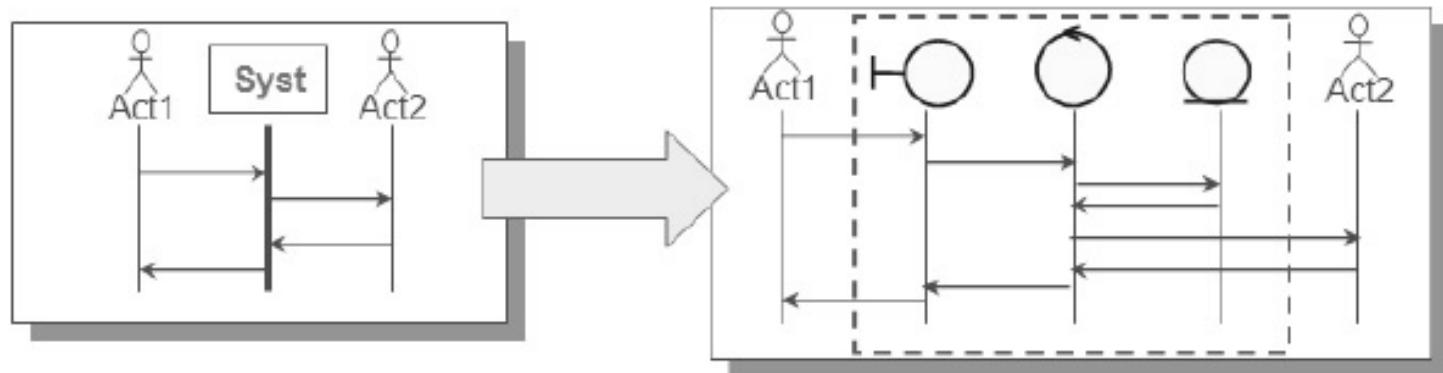
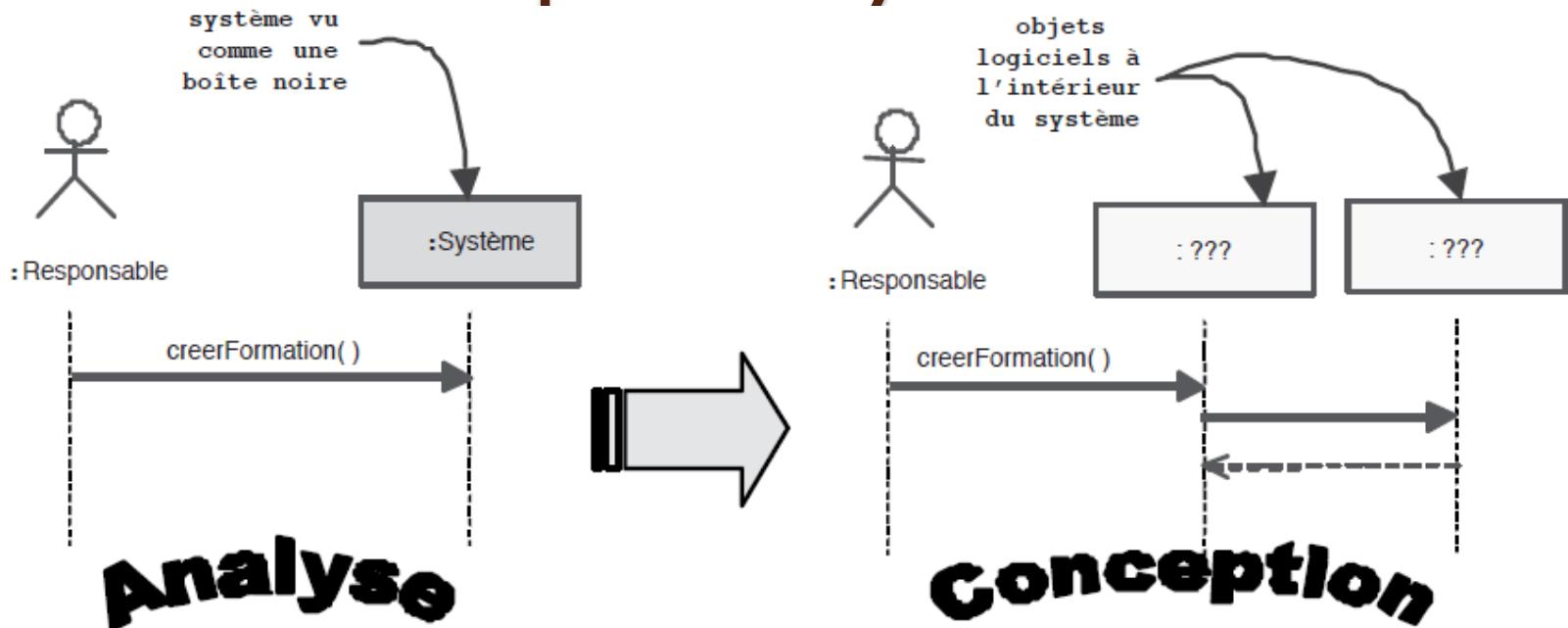
- Operateur « Assertion » ou assert :
 - ✓ Représente une assertion ;
 - ✓ La séquence de trace de l'opérande décrit par l'assertion est la seule trace valide.



Exemple d'un diagramme de séquence du système

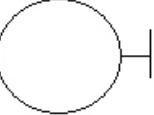
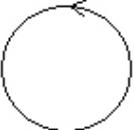
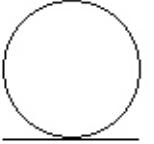


Changement de niveau d'abstraction par rapport au diagramme de séquence système



Stéréotypes de Jacobson (I/2)

- À l'intérieur du système, Jacobson distingue les trois stéréotypes suivants :

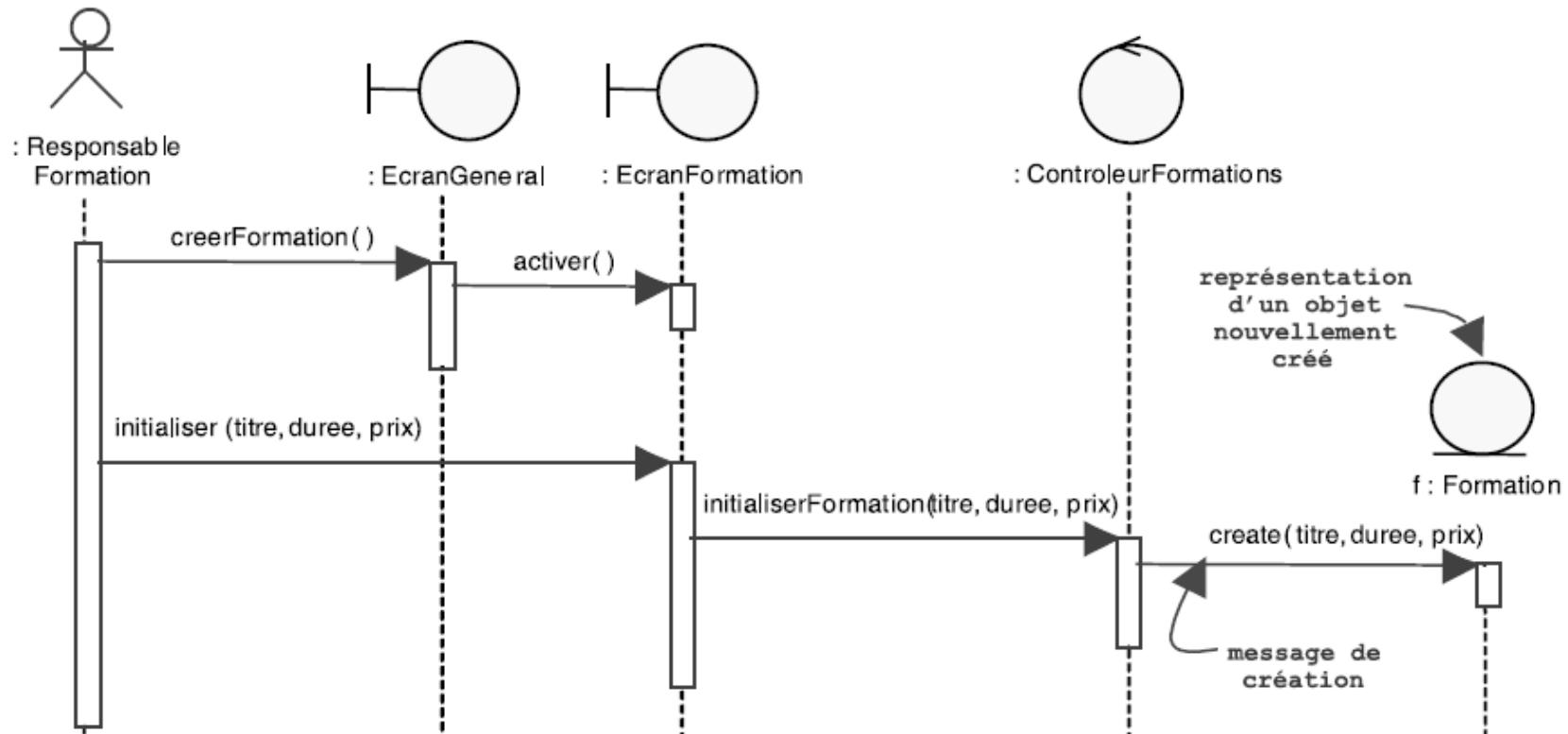
Stéréotype	Symbole	Description
<<boundary>>		Ce sont les classes qui servent à modéliser les interactions entre le système et ses acteurs.
<<control>>		Ce sont les classes utilisées pour représenter la coordination, l'enchaînement et le contrôle d'autres objets – elles sont en général reliées à un cas d'utilisation particulier.
<<entity>>		Ce sont les classes qui servent à modéliser des informations durables et souvent persistantes

Stéréotypes de Jacobson (2/2)

Il existe des règles précises sur les interactions possibles entre instances de ces trois types de classes:

- Les acteurs ne peuvent interagir (envoyer des messages) qu'avec les dialogues (Boundary);
- Les dialogues (Boundary) peuvent interagir avec les contrôles;
- Les contrôles (Controller) peuvent interagir avec les dialogues, les entités, ou d'autres contrôles;
- Les entités (Entity) ne peuvent interagir qu'entre elles.

Exemple d'un diagramme de séquence de conception



Conseils

- Toujours donner le contexte du diagramme du cas d'utilisation;
- Indiquer précisément le but du scénario;
- Bien préciser:
 - ✓ l'acteur qui déclenche le scénario;
 - ✓ le résultat observable de l'exécution du cas d'utilisation.
- Faire apparaître un objet interface/application entre l'acteur et les objets du système;
- Soigner la succession des appels.

NB:

- ✓ Ne pas confondre le cas d'utilisation / diagramme de séquence;
- ✓ Cohérence diagramme de classe / diagramme de séquence.



DIAGRAMMES COMPORTEMENTAUX

DIAGRAMME DE COMMUNICATION

Diagramme de communication...?

- Aussi appelé diagramme de collaboration en UML I.x ;
- Représentation des scénarios ;
- Interaction entre les objets ;
- Définit une communication particulière entre les objets ;
- Représentation simplifiée d'un diagramme de séquence :
 - ✓ Pas de ligne de vie ;
 - ✓ Même syntaxe d'objet au diagramme de séquence.
- Messages séquentiels numérotés.

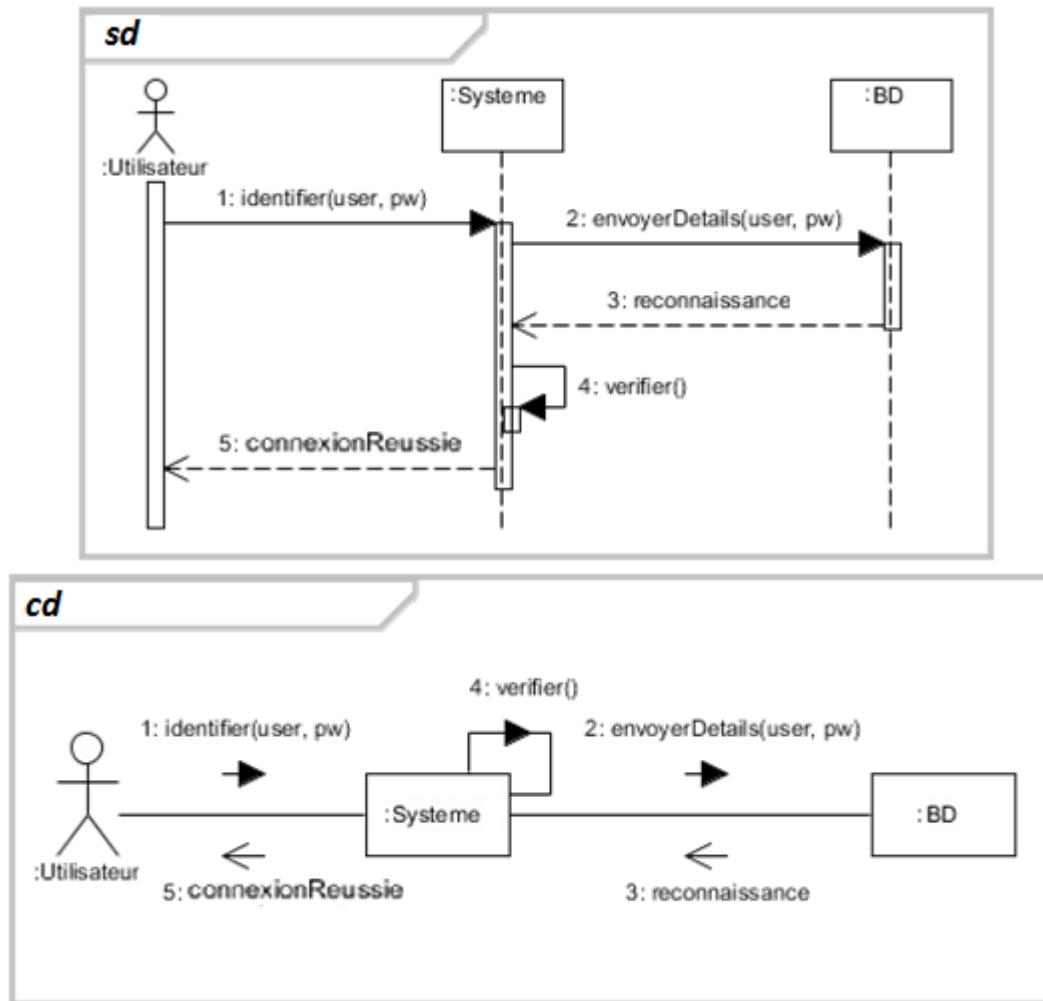
Diagramme de séquence et Diagramme de communication (1/3)

- Deux vues du même phénomène : Il existe des outils supportant UML qui permet de créer automatiquement le diagramme de communication (ou collaboration) à partir du diagramme de séquence.

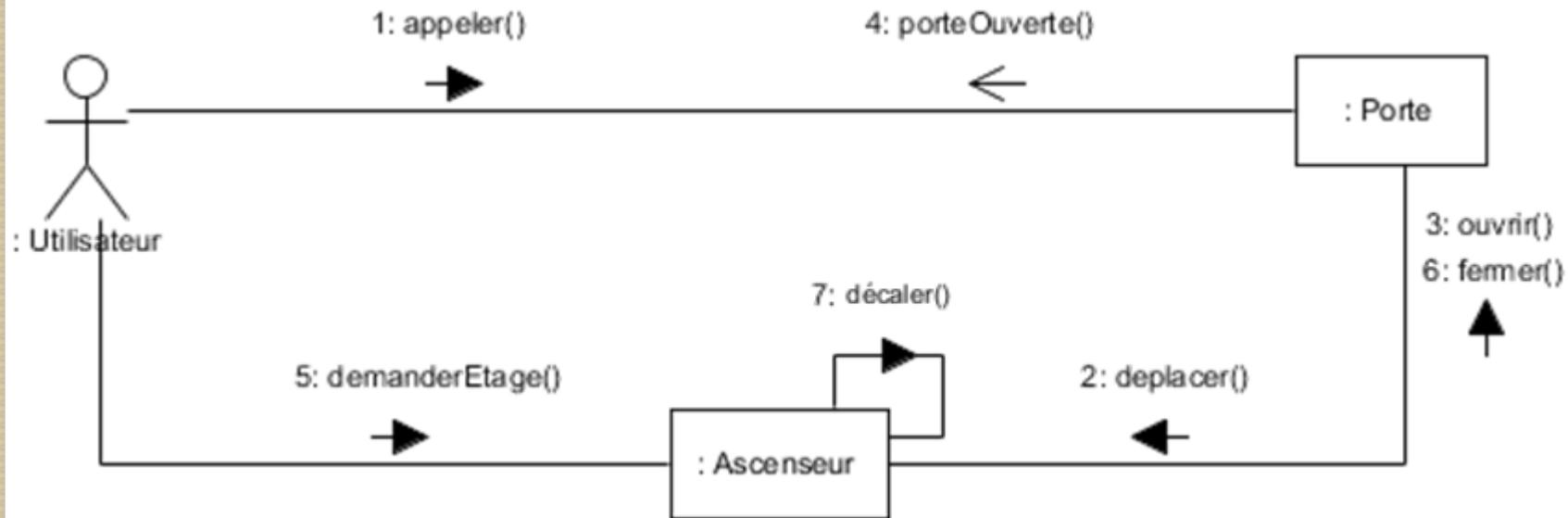
Diagramme de séquence et Diagramme de communication (2/3)

- Diagramme séquence :
 - ✓ Se focalise sur les aspects temporels ;
 - ✓ Importance des messages ;
 - ✓ Modélise la création/destruction d'objets ;
 - ✓ Montre l'activation des objets ;
 - ✓ Décrit les scénarios complexes ;
 - ✓ Permet la modularité de la représentation.
- Diagramme de communication :
 - ✓ Se focalise sur la représentation spatiale ;
 - ✓ Adapté pour montrer les liens ;
 - ✓ Donne la priorité à figuration des interactions ;
 - ✓ Pas de modularité de la représentation.

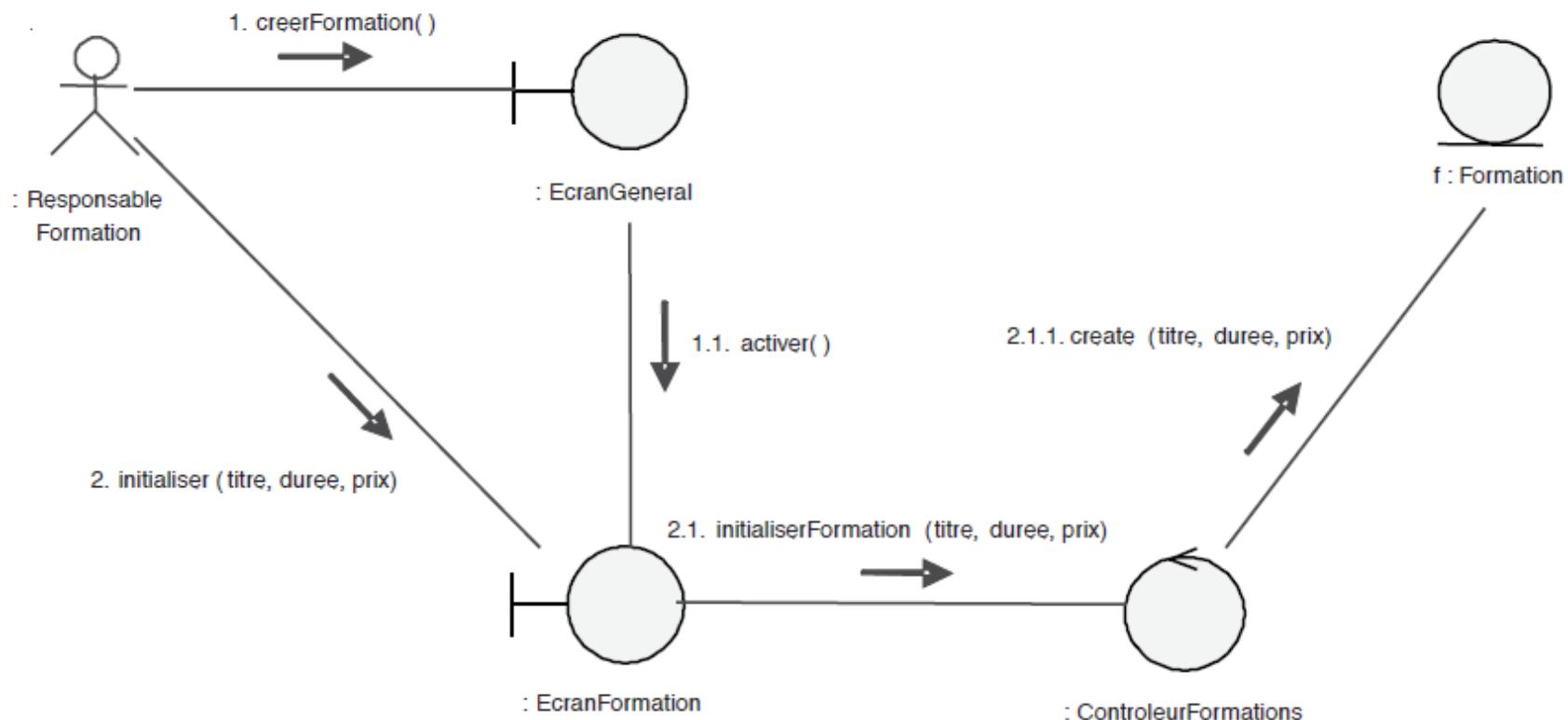
Diagramme de séquence et Diagramme de communication (3/3)



Exemple d'un diagramme de communication du système



Exemple d'un diagramme de communication de conception





LES DIAGRAMMES COMPORTEMENTAUX

DIAGRAMME D'ÉTATS – TRANSITIONS

Diagramme d'états-transitions...?

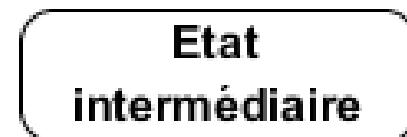
- Aussi appelé diagramme de machine d'état ;
- Décrit le comportement des objets d'une classe au moyen d'un automate d'états associé à la classe ;
- ✓ Le comportement est modélisé par un graphe :
 - Nœuds = états possibles des objets ;
 - Arcs = transitions d'état à état.
- Montre les états remarquables des objets du système ;
- Permet d'autoriser/interdire les traitements sur les objets ;
- Pas de diagramme si la classe n'a pas d'états différents ;
- Un diagramme est attaché à une seule classe.

État

- Ensemble des valeurs qui impliquent la même réponse à l'arrivée d'un événement ;
- Situation durant la vie d'un objet pendant laquelle :
 - ✓ il satisfait une certaine condition ;
 - ✓ il exécute une certaine activité ;
 - ✓ ou il attend un certain évènement.
- Chaque objets possède à un instant donné un état particulier ;
- Un état est stable et durable ;
- Chaque état est identifié par un nom ;
- Il est possible de n'avoir aucun état final : un système qui ne s'arrête jamais.



Etat initial



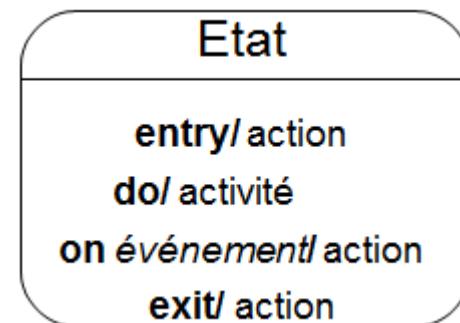
Etat final

Action et activité (1/2)

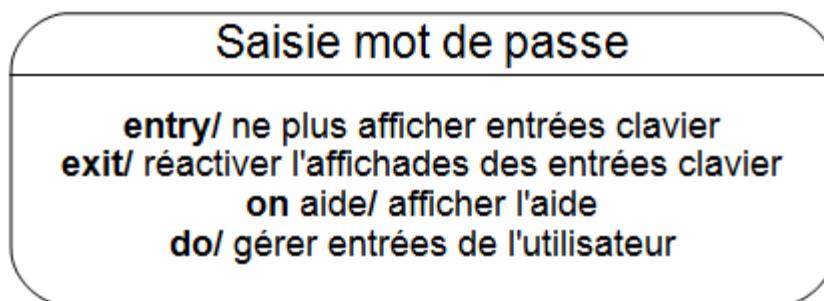
- Action :
 - ✓ opération instantanée qui ne peut être interrompue ;
 - ✓ être associée à une transition ;
 - ✓ peut être exécutées :
 - lors d'une transition (ex : action4) ;
 - en entrant dans un état (ex : action1) ;
 - en sortant d'un état (ex : action3).
- Activité :
 - ✓ une opération d'une certaine durée qui peut être interrompue ;
 - ✓ être associée à un état d'un objet ;
 - ✓ peut être exécutées dans un état (ex : activité2).



Action et activité (2/2)

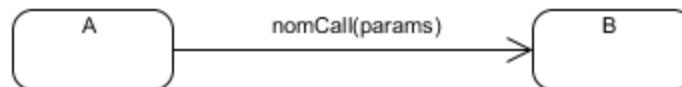


Exemple :

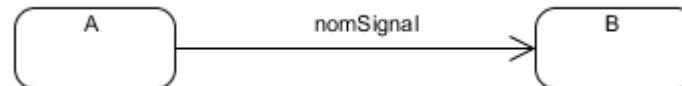


Événement

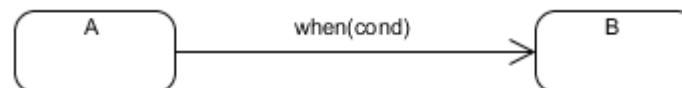
- Un fait survenu qui déclenche une transition ;
- Types d'événements :
 - ✓ Type appel de méthode (call) – C'est le type d'événement qui représente la réception de l'appel d'une opération par un objet. Les événements d'appel sont des méthodes déclarées au niveau du diagramme de classes ;



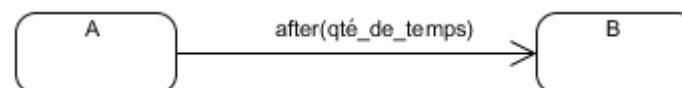
- ✓ Type signal – C'est la réception d'un signal asynchrone, explicitement émis par un autre objet ;



- ✓ Type changement de valeur (vrai/faux) – C'est le cas de l'évaluation d'une expression booléenne ;

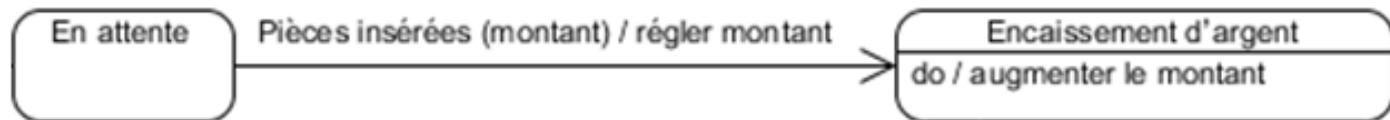


- ✓ Type écoulement du temps – C'est un événement lié à une condition de type after (durée) ou when (date).



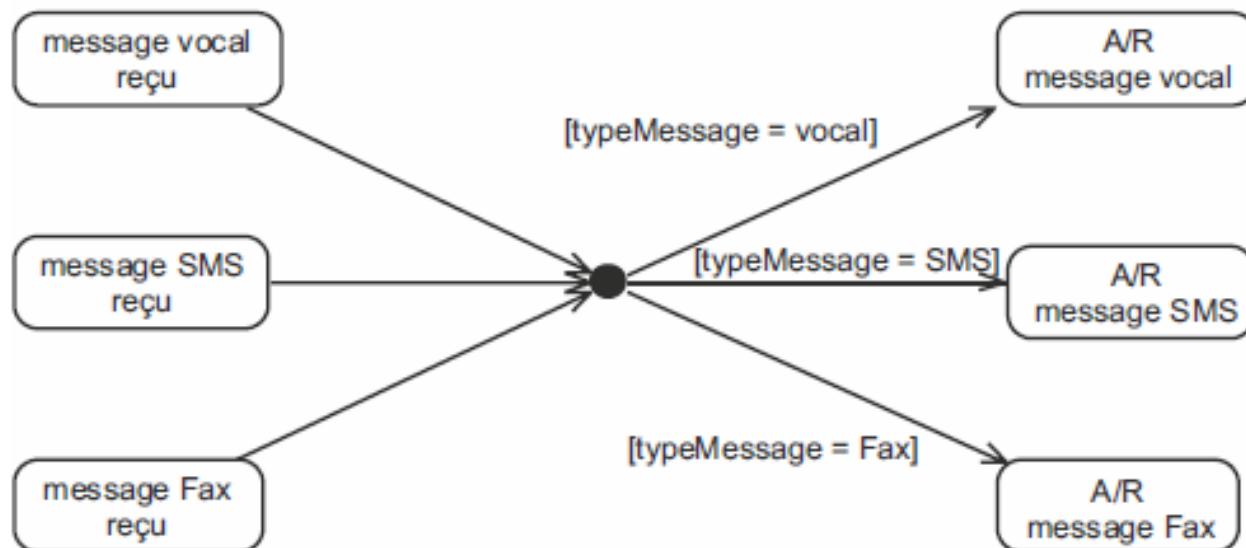
Transition

- Réponse d'un objet à l'occurrence d'un évènement ;
- Une relation entre deux états ;
- Syntaxe d'une transition :
événement [garde] / action



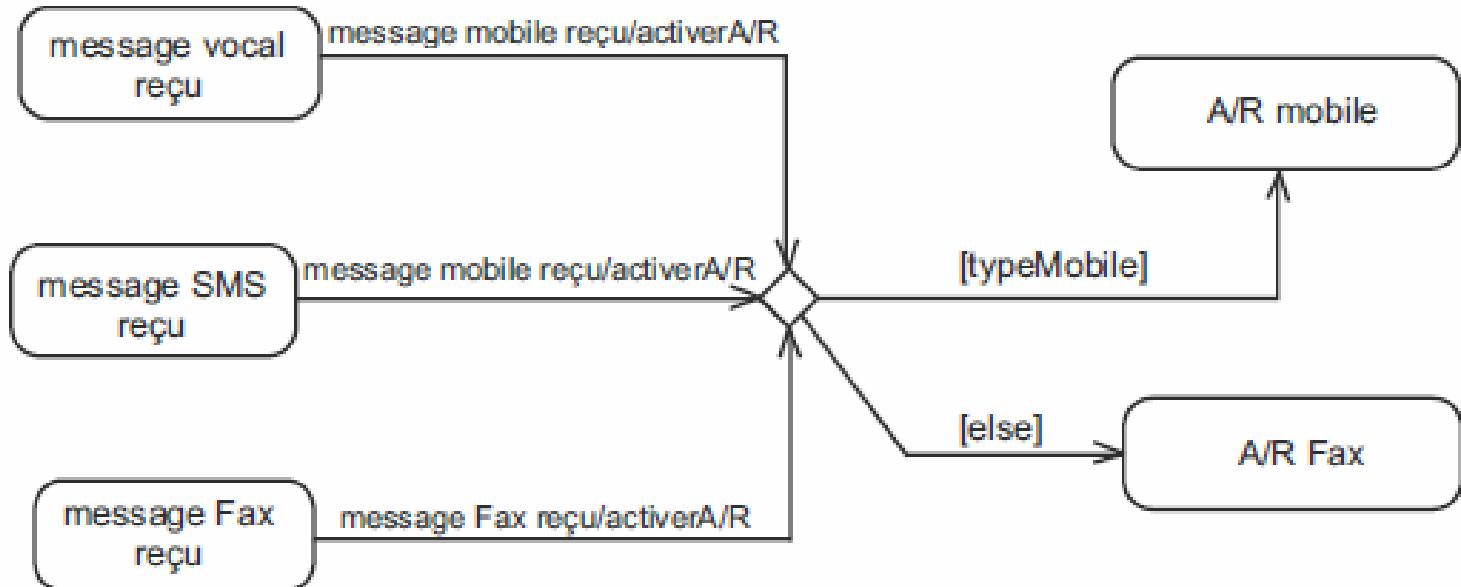
Point de jonction

- Permet de décomposer une transition en deux parties en indiquant si nécessaire les gardes propres à chaque segment de la transition.



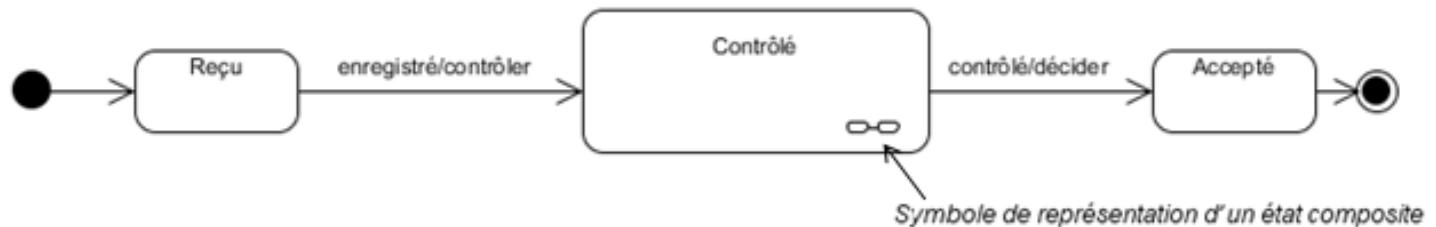
Point de choix

- Se comporte comme un test de type : si condition faire action1 sinon faire action2.

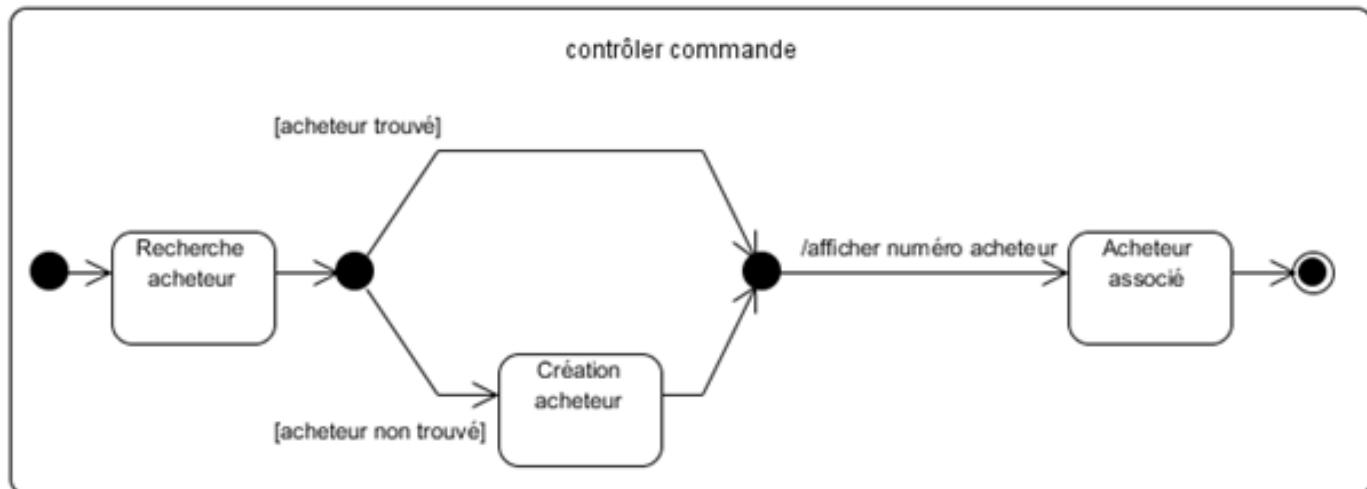


Etat composite

- Permet de structurer de manière hiérarchique les états et les transitions.

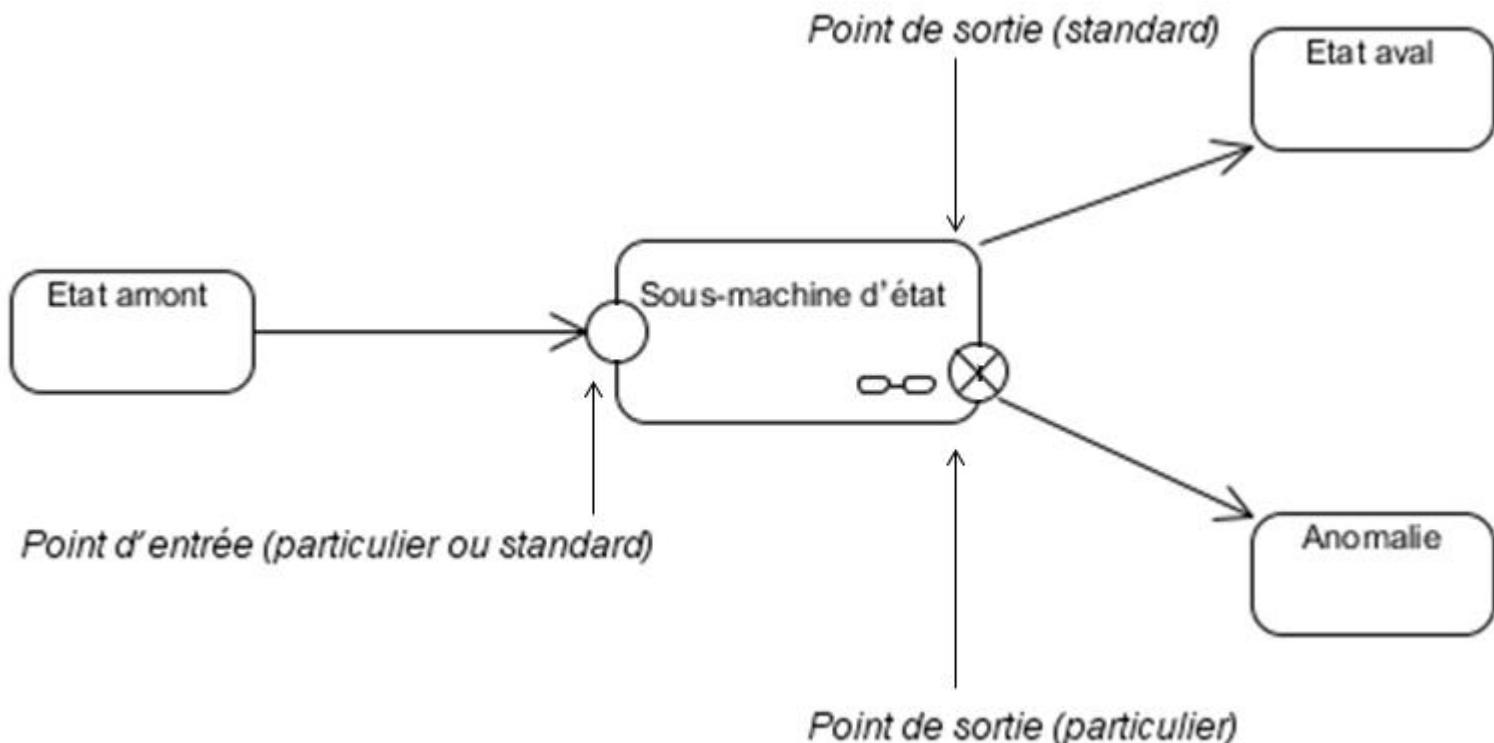


Détail de l'état composite



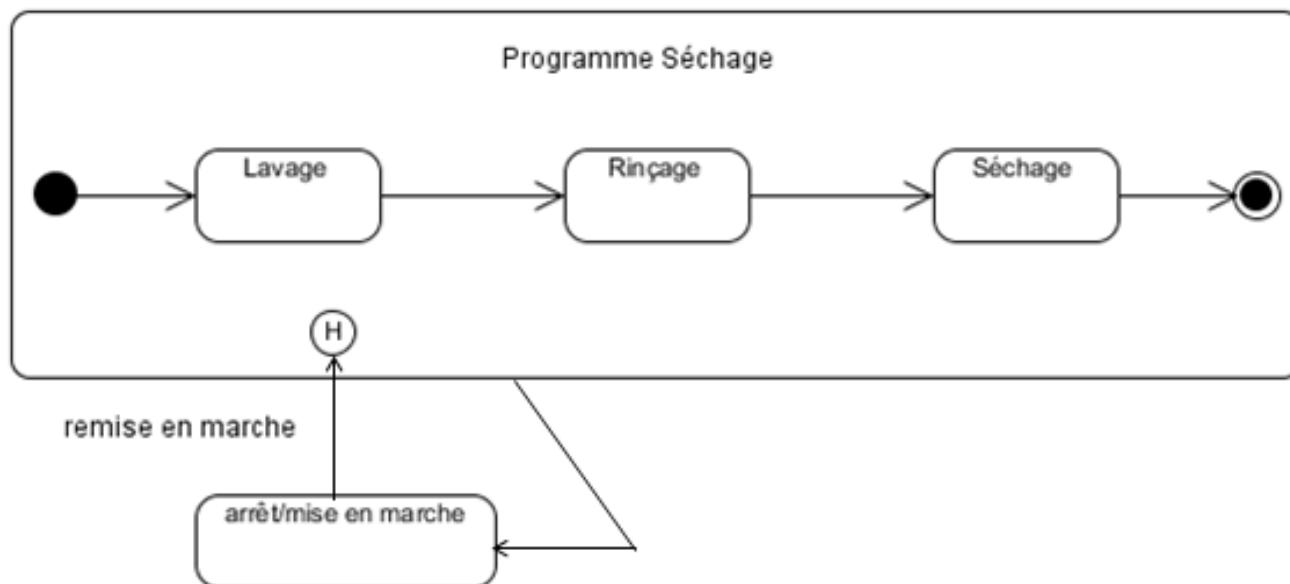
Point d'entrée et de sortie

- Repérer un point d'entrée et un point de sortie particuliers sur une sous-machine d'état.

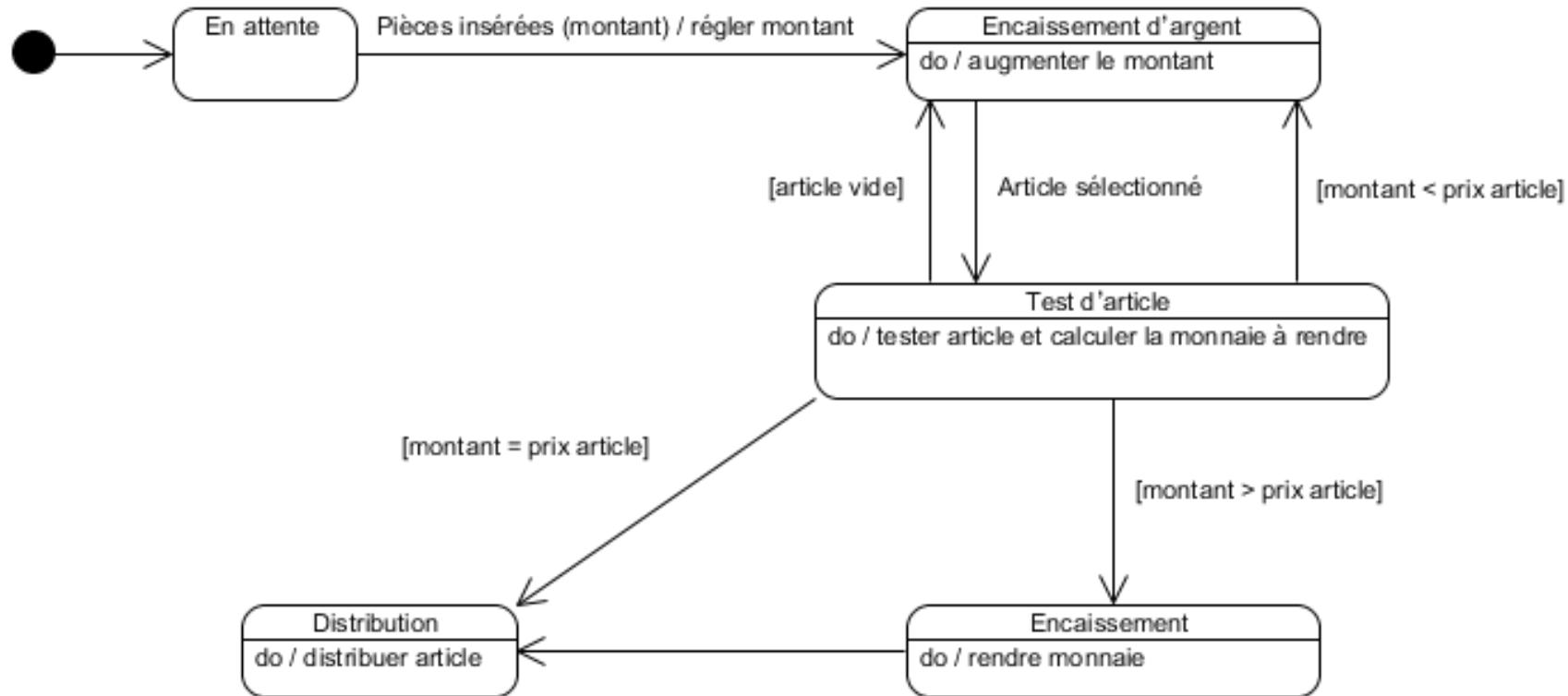


État historique

- Permet de pouvoir indiquer la réutilisation du dernier état historisé en cas de besoin.



Exemple d'un diagramme d'états-transitions



Conseil

- Cohérence entre la classe et le diagramme d'états-transitions.

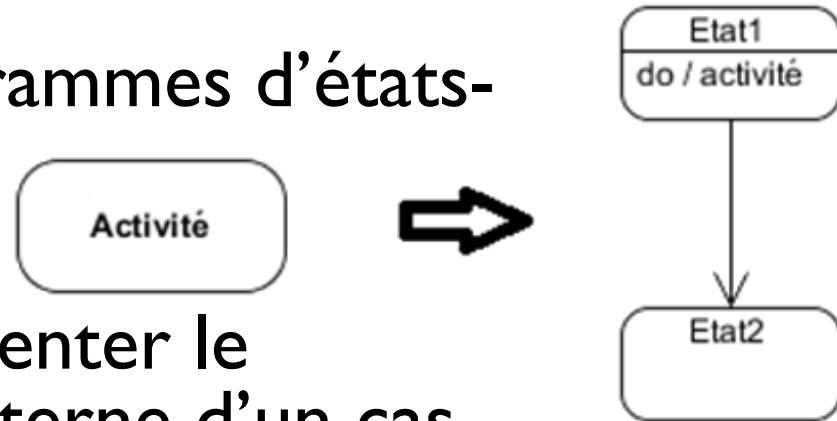


DIAGRAMMES COMPORTEMENTAUX

DIAGRAMME D'ACTIVITÉS

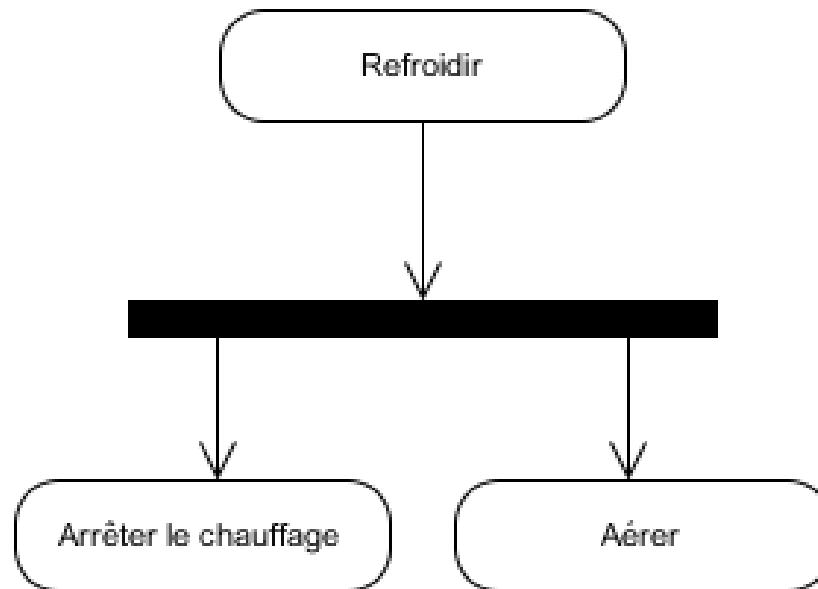
Diagramme d'activités...?

- Variante des diagrammes d'états-transitions ;
- Permet de représenter le comportement interne d'un cas d'utilisation ou processus ;
- Représente le déroulement des traitements en les regroupant dans des étapes appelées « Activité » ;
- Prend en charge les comportements parallèles.



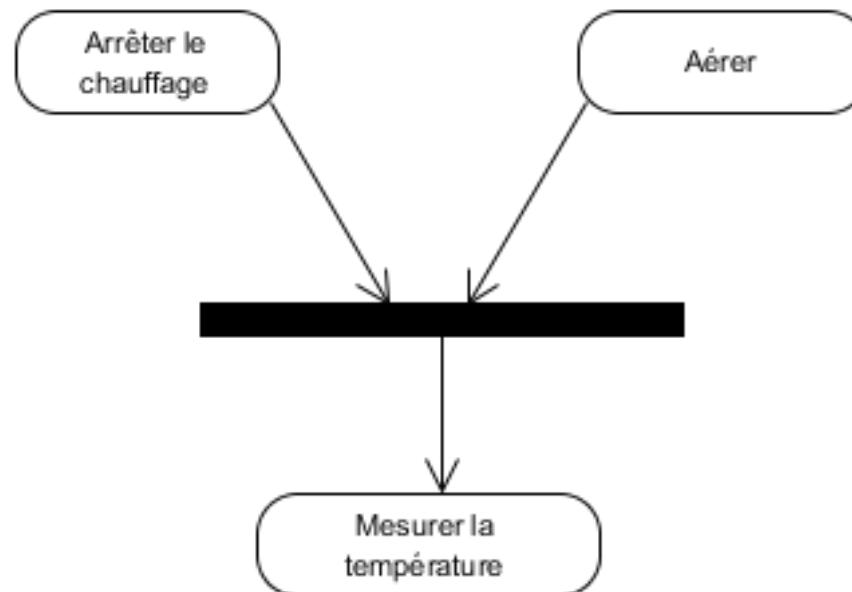
Nœud de bifurcation (fourche)

- Création des plusieurs flots concurrents en sortie de la barre de synchronisation à partir d'un flot unique entrant.



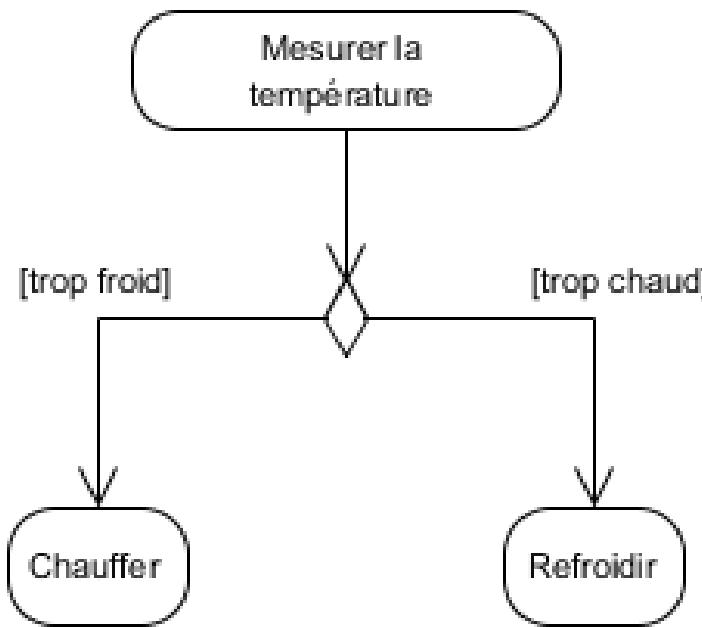
Nœud de jonction (synchronisation)

- Production d'un flot unique sortant à partir de plusieurs flots concurrents en entrée de la synchronisation ;
- Symétrique du nœud de bifurcation.



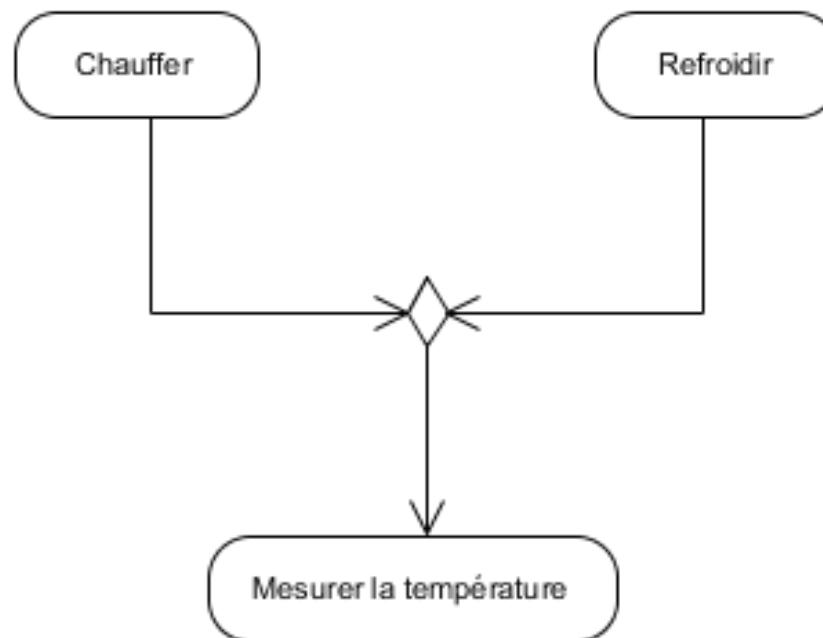
Nœud de test-décision

- Permet de faire un choix entre plusieurs flots sortants en fonction des conditions de garde de chaque flot ;
- N'a qu'un seul flot en entrée ;
- On peut aussi utiliser seulement deux flots de sortie : le premier correspondant à la condition vérifiée et l'autre traitant le cas sinon.



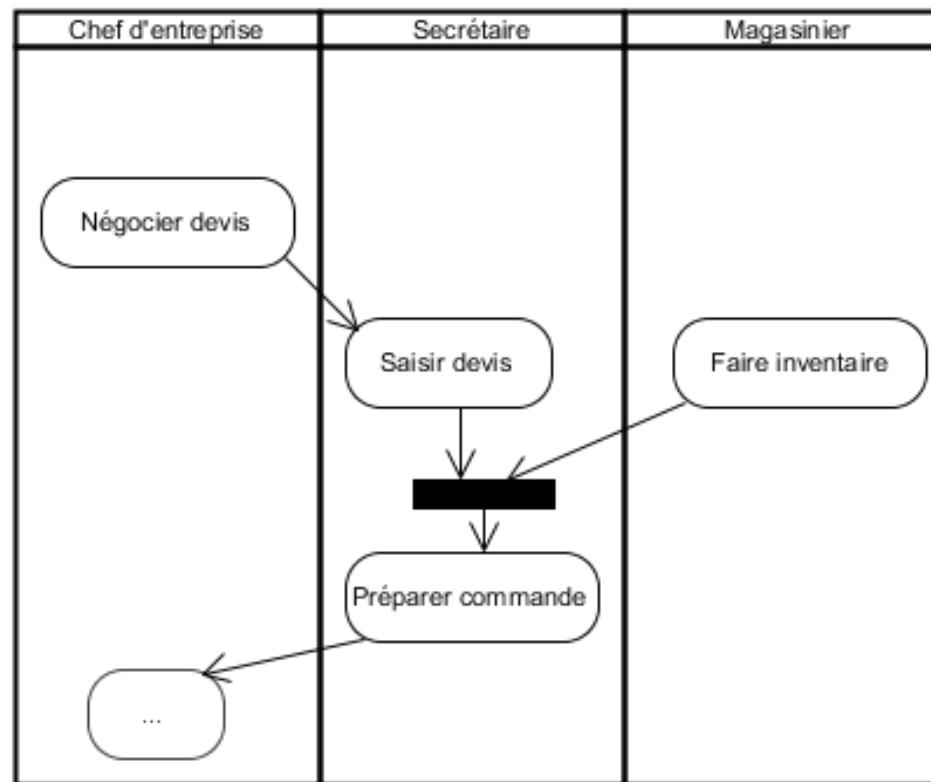
Nœud de fusion-test

- Permet d'avoir plusieurs flots entrants possibles et un seul flot sortant ;
- Le flot sortant est donc exécuté dès qu'un des flots entrants est activé.



Couloir d'activités (Responsabilité)

- Préciser la responsabilité de différents acteurs ;
- Chaque activité est allouée à un couloir correspondant à la ressource concernée.



Noeud d'objet

- Souvent, différentes activités manipulent un même objet qui change alors d'état selon le degré d'avancement du mécanisme ;
- Un résultat de l'activité (lié par une flèche) et repris comme événement pour l'activité suivante.



Autres notations (1/2)



- Signal reçu : un événement pour le processus étudié ;
- ✓ On le représente par un pentagone concave.



- Signal envoyé : un résultat émis par le processus étudié ;
- ✓ On le représente par un pentagone convexe.



- Evénement temporel : une date ou un délai ;
- ✓ On le représente par deux triangles isocèles inversés (en tête bêche).

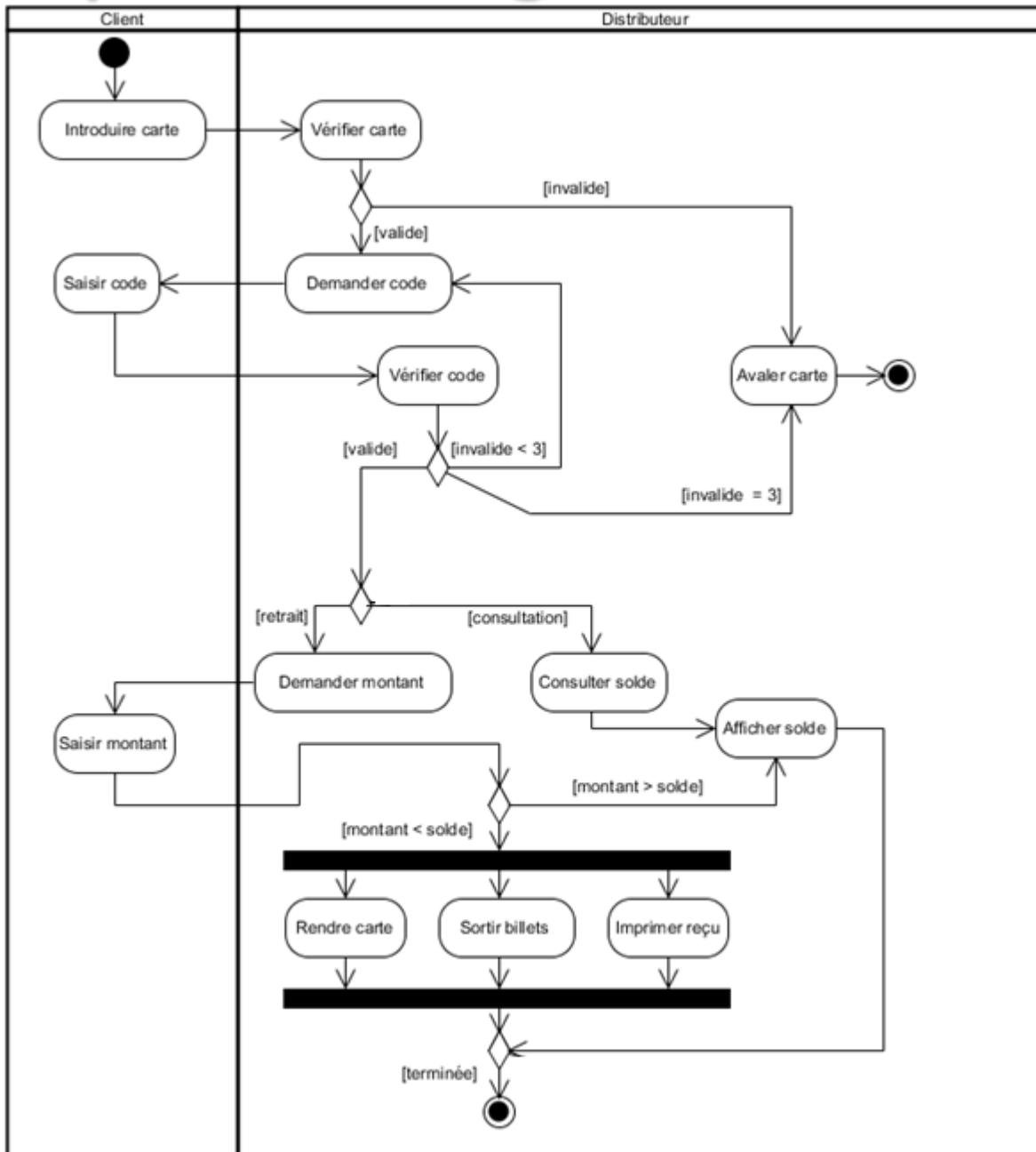


- Nœud de départ du diagramme ;
- ✓ On le représente par un petit cercle plein.



- Nœud de fin du diagramme ;
- ✓ On le représente par un cercle vide contenant un petit cercle plein.

Exemple d'un diagramme d'activités



Conseils (1/2)

- A utiliser pour documenter les cas d'utilisation:
 - ✓ Les cas d'utilisation en extension doivent apparaître dans le cadre d'un branchement / fusion qui explicite la condition d'extension;
 - ✓ Distinguer les cas d'utilisation des tâches qui les constituent:
 - Actions / Activités;
 - Stéréotypes.

Conseils (2/2)

- A utiliser à bon escient;
- Peuvent devenir trop complexes.



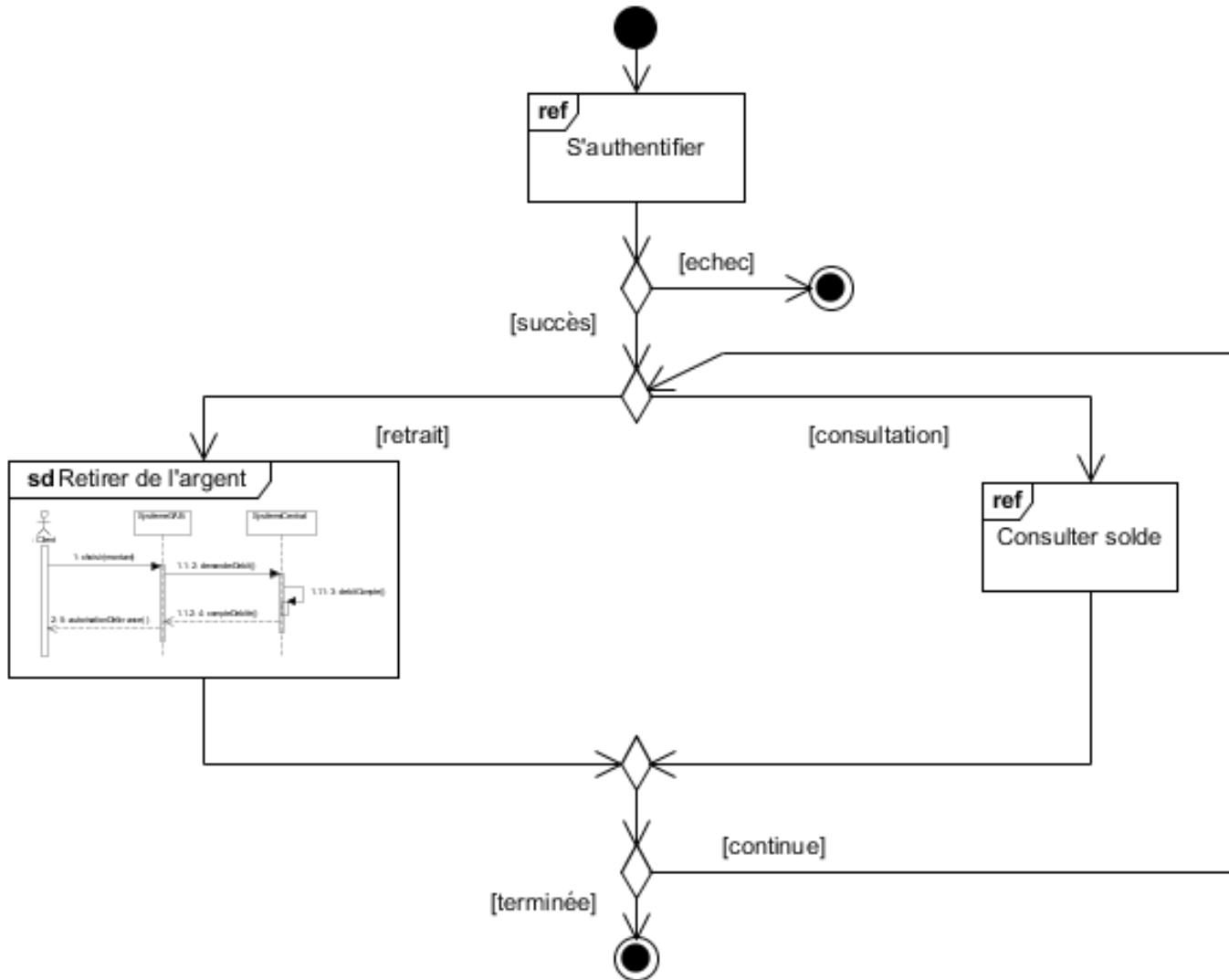
DIAGRAMMES COMPORTEMENTAUX

DIAGRAMME GLOBAL D'INTERACTION

Diagramme global d'interaction...?

- Nouveau diagramme d'UML 2.x ;
- Mélange du diagramme d'activités et du diagramme de séquence.

Exemple d'un diagramme global d'interaction





DIAGRAMMES COMPORTEMENTAUX

DIAGRAMME DE TEMPS

Diagramme de temps...?

- Nouveau diagramme d'UML 2.x ;
- Permet de représenter les changements d'états et des interactions entre objets liés à des contraintes de temps ;
- A faire après le diagramme d'états-transitions ;
- Vient du domaine du hardware ;
- Plutôt pour les systèmes temps réel.

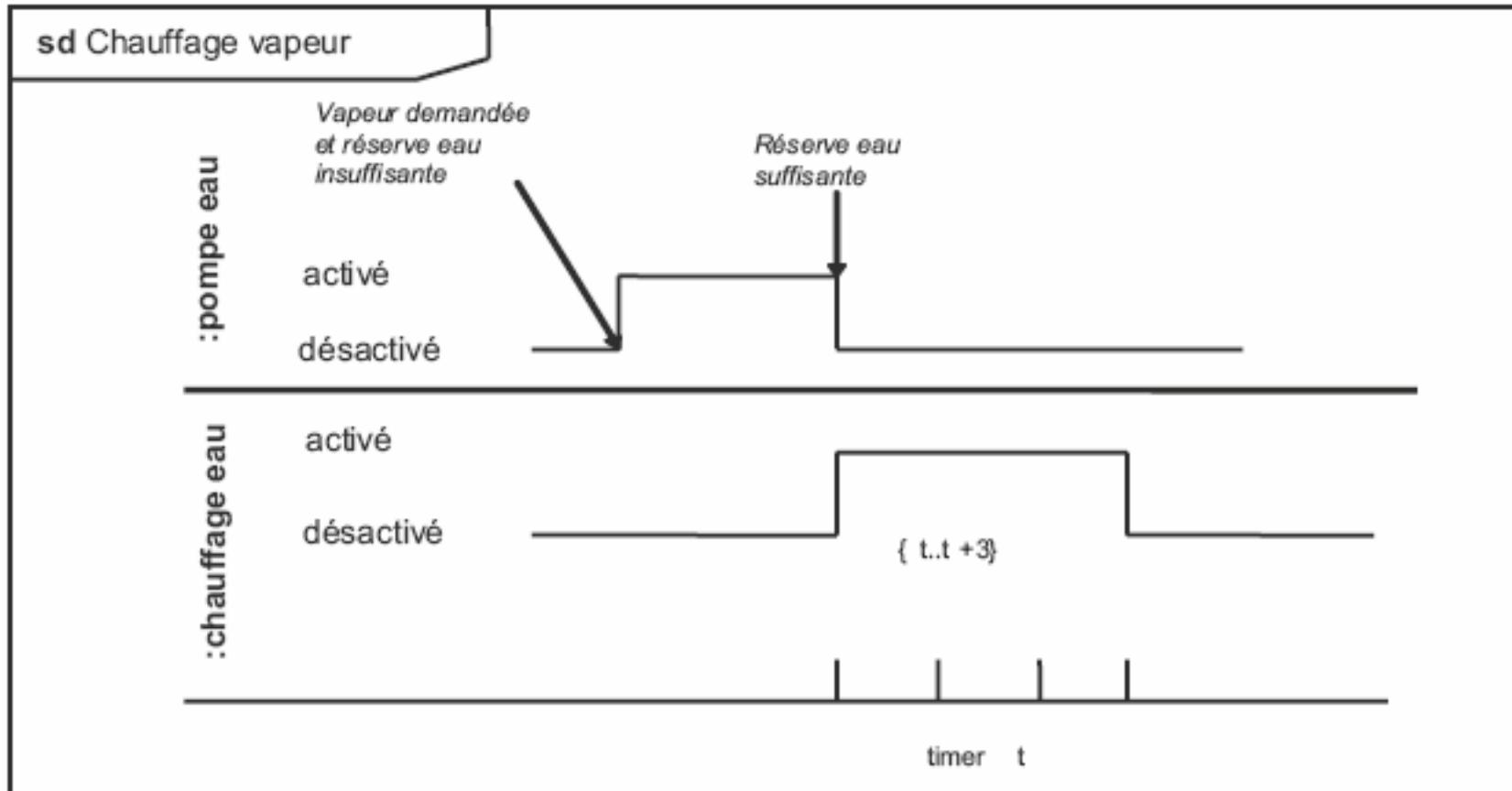
Représentations

- Deux représentations graphiques possibles :
 - ✓ représentation « en escalier » ;
 - ✓ représentation linéaire.

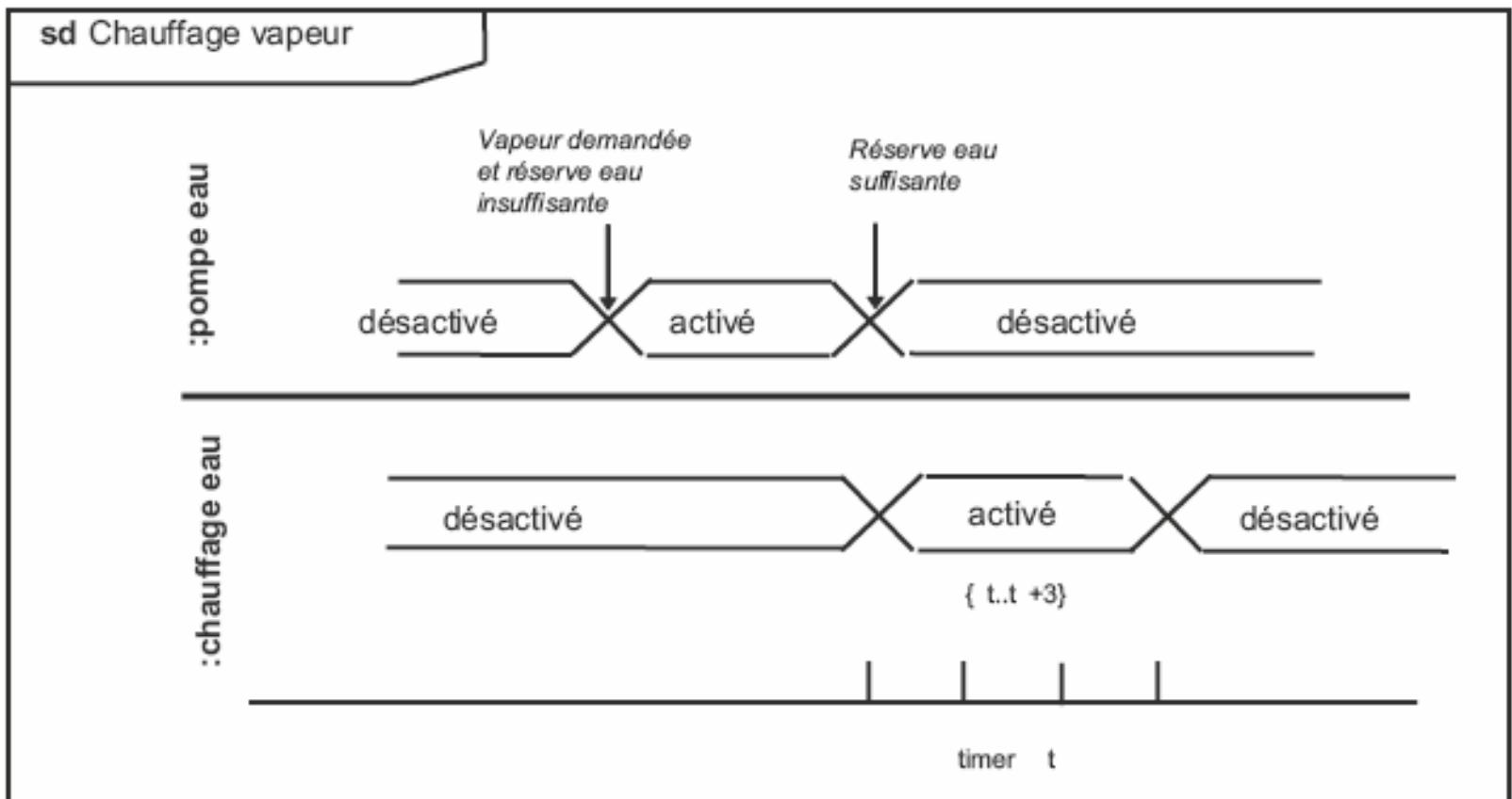
Concepts

- Concepts représentés :
 - ✓ la ligne de vie ;
 - ✓ l'évènement ;
 - ✓ l'état ;
 - ✓ la contrainte de durée.

Exemple d'un diagramme de temps avec représentation « en escalier »



Exemple d'un diagramme de temps avec représentation linéaire





DIAGRAMMES STRUCTURELS



DIAGRAMMES STRUCTURELS

DIAGRAMME DE CLASSES

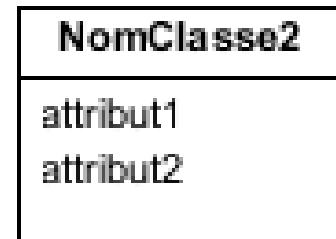
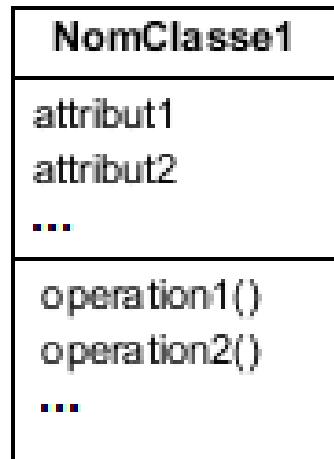
Diagramme de classes...?

- Diagramme pivot de l'ensemble de la modélisation d'un système ;
- Permet de donner la représentation statique du système à développer;
- Fondé sur :
 - ✓ le concept d'objet,
 - ✓ le concept de classe comprenant les attributs et les opérations,
 - ✓ les différents types d'association entre classes.

Classe, attribut et opération (1/6)

Classe :

- Décrire un groupe d'objets ;
- Rectangle comportant plusieurs compartiments :
 - ✓ Compartiment 1 : nom de la classe,
 - ✓ Compartiment 2 : attributs,
 - ✓ Compartiment 3 : opérations,
 - ✓ Possibilité d'ajouter les compartiments des responsabilités et/ou exception.
- Différents niveaux de détail possibles : possibilité d'omettre des attributs et/ou des opérations.



Classe, attribut et opération (2/6)

Attribut :

- Propriété élémentaire d'une classe.

Ordinateur
nom : texte
marque : texte
puissance : entier
tailleMémoire : entier
os : texte

Classe, attribut et opération (3/6)

Attribut : (suite)

- Syntaxe :

visibilité/nomAttribut : type [= valeurInitiale {propriétés}]

- ✓ visibilité : + (public), - (privé), # (protégé), ~ (package)
- ✓ nomAttribut : nom unique dans sa classe
- ✓ type : type primitif (Entier, Chaîne, ...) ou classe
- ✓ valeurInitiale : valeur initiale à la création de l'objet
- ✓ propriétés : valeurs marquées facultatives ({gelé}, {variable}, {ajoutUniquement}, ...)

- Attributs de classe (statiques) soulignés ;
- Attributs dérivés précédés de "/".

Classe, attribut et opération (4/6)

Opération :

- Fonction applicable aux objets d'une classe ;
- Décrire le comportement d'un objet.

Ordinateur
nom : texte
marque : texte
puissance : entier
tailleMemoire : entier
os : texte
allumer()
eteindre()

Classe, attribut et opération (5/6)

Opération : (suite)

- Syntaxe :

```
visibilité nomOpération (paramètres)
[: [typeRésultat] {propriétés}]
```

- ✓ visibilité: + (public), - (privé), # (protégé), ~ (package)
- ✓ nomOpération: utiliser un verbe représentant l'action à réaliser
- ✓ paramètres: liste des paramètres selon le format
- ✓ typeRésultat: type de la valeur renvoyée (type primitif ou classe)
- ✓ propriétés: valeurs facultatives applicables ({query}, ...)
- Opérations abstraites/virtuelles (non implémentées) en italique ;
- Opérations de classe (statiques) soulignées ;
- Possibilité d'annoter avec des contraintes stéréotypées «*precondition*» et «*postcondition*» : programmation par contrats ;
- Possibilité de surcharger une opération : même nom, mais paramètres différents ;

Classe, attribut et opération (6/6)

Visibilité des attributs et opérations :

- Public (+) : Attribut ou opération visible par tous
- Privé (-) : Attribut ou opération seulement visible à l'intérieur de la classe.
- Protégé (#) : Attribut ou opération visible seulement à l'intérieur de la classe et pour toutes les sous-classes de la classe
- Paquetage (~) : Attribut ou opération ou classe seulement visible à l'intérieur du paquetage où se trouve la classe.

Ordinateur
-nom : texte
-marque : texte
-puissance : entier
-tailleMemoire : entier
-os : texte
+allumer()
#eteindre()

Association, multiplicité, navigabilité et contraintes (1/6)

Association:

- Représenter les liens existant entre les instances des classes ;
- Porter un nom (signification).



Association, multiplicité, navigabilité et contraintes (2/6)

Association: (suite) rôle d' association

- Précise le rôle d'une classe au sein de l'association ;
- Explicite le nom de l'association si besoin.



Association, multiplicité, navigabilité et contraintes (3/6)

Multiplicité :

- Indique un domaine de valeurs pour préciser le nombre d'instance d'une classe vis-à-vis d'une autre classe pour une association donnée.

1	un et un seul
0 .. 1	zéro ou un
M .. N	de M à N (entiers naturels)
*	de zéro à plusieurs
0 .. *	
1 .. *	de un à plusieurs
N	exactement N (entier naturel)

Association, multiplicité, navigabilité et contraintes (4/6)

Navigabilité :

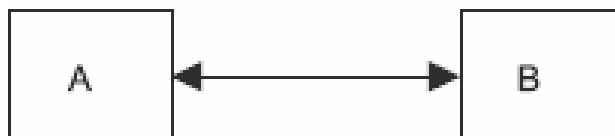
- Indique si l'association fonctionne de manière unidirectionnelle ou bidirectionnelle.



- Navigabilité unidirectionnelle de A vers B.
- Pas de navigabilité de B vers A (non définie explicitement).



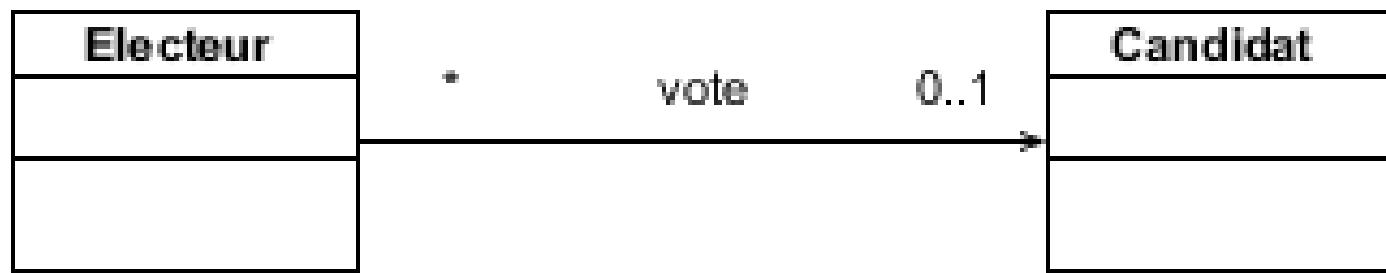
- Navigabilité unidirectionnelle de B vers A.
- Navigabilité de A vers B (non définie explicitement).



- Navigabilité bidirectionnelle entre A et B. Habituellement représentée sans flèche.

Association, multiplicité, navigabilité et contraintes (5/6)

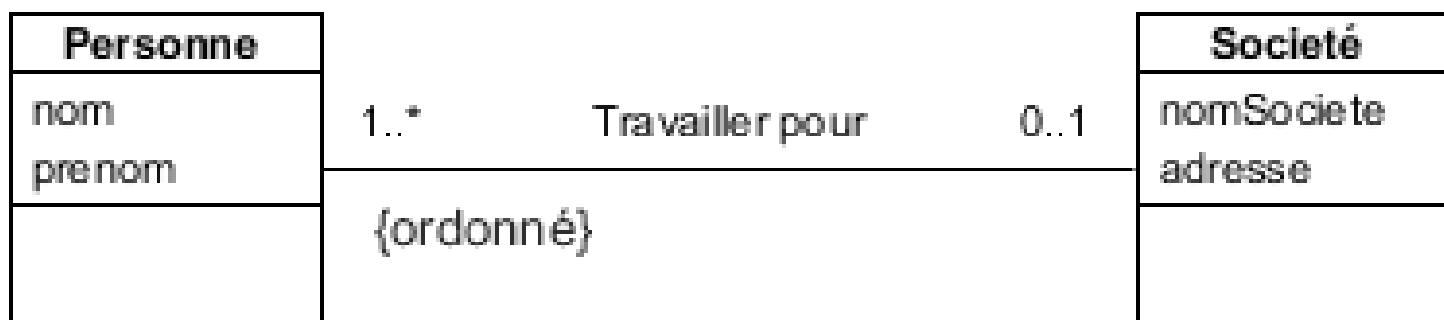
Navigabilité : (suite)



Association, multiplicité, navigabilité et contraintes (6/6)

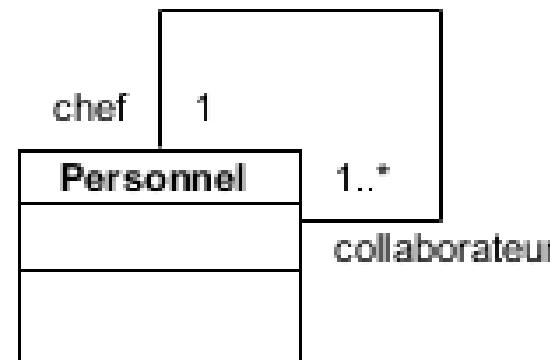
Contraintes :

- Ordre de tri (multiplicité supérieure à 1) :
 - ✓ non ordonnés (valeur par défaut),
 - ✓ ordonnés ou triés lorsque l'on est au niveau de l'implémentation (tri sur une valeur interne).



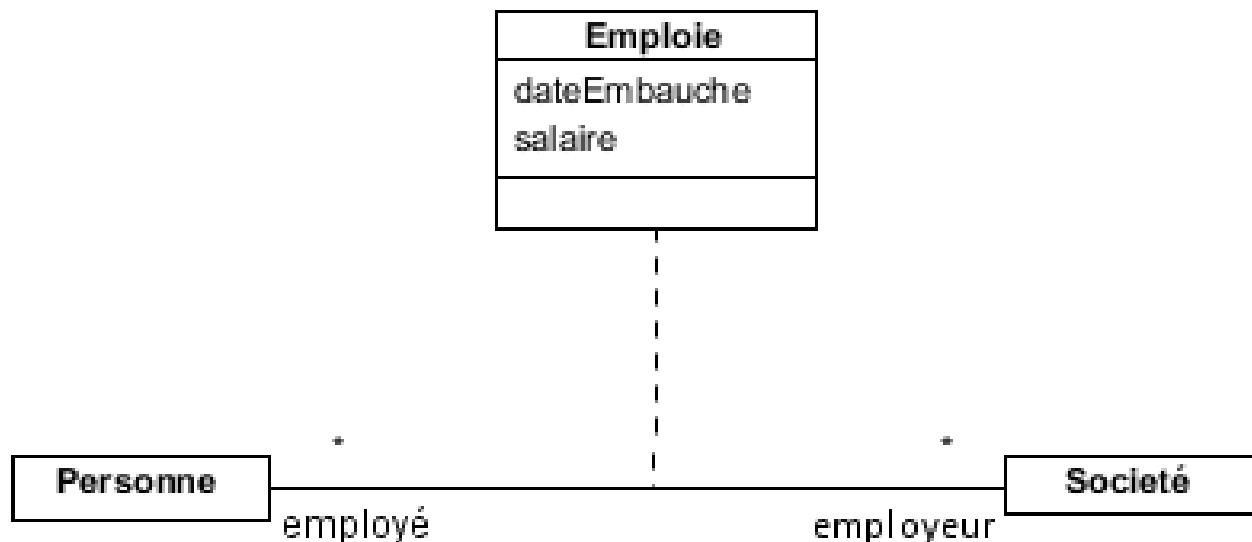
Association réflexive

- Définit les liens entre des objets d'une même classe.



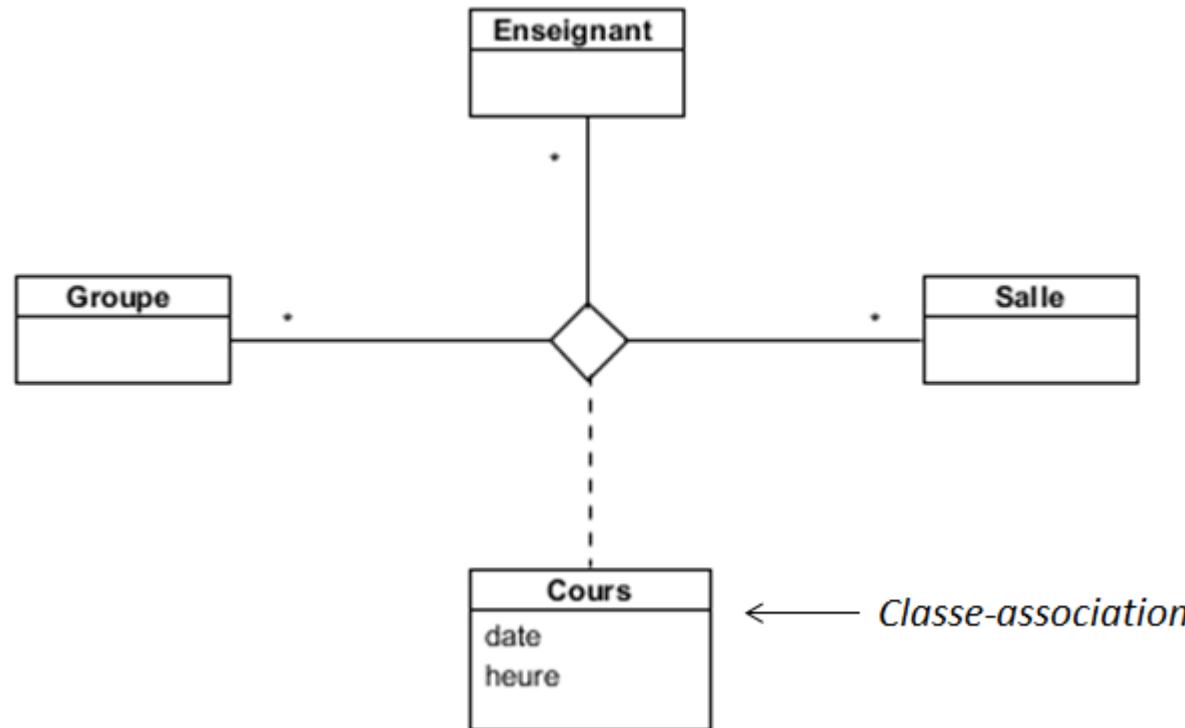
Classe-association

- Permet de décrire soit des attributs soit des opérations propres à l'association.



Association n-aire

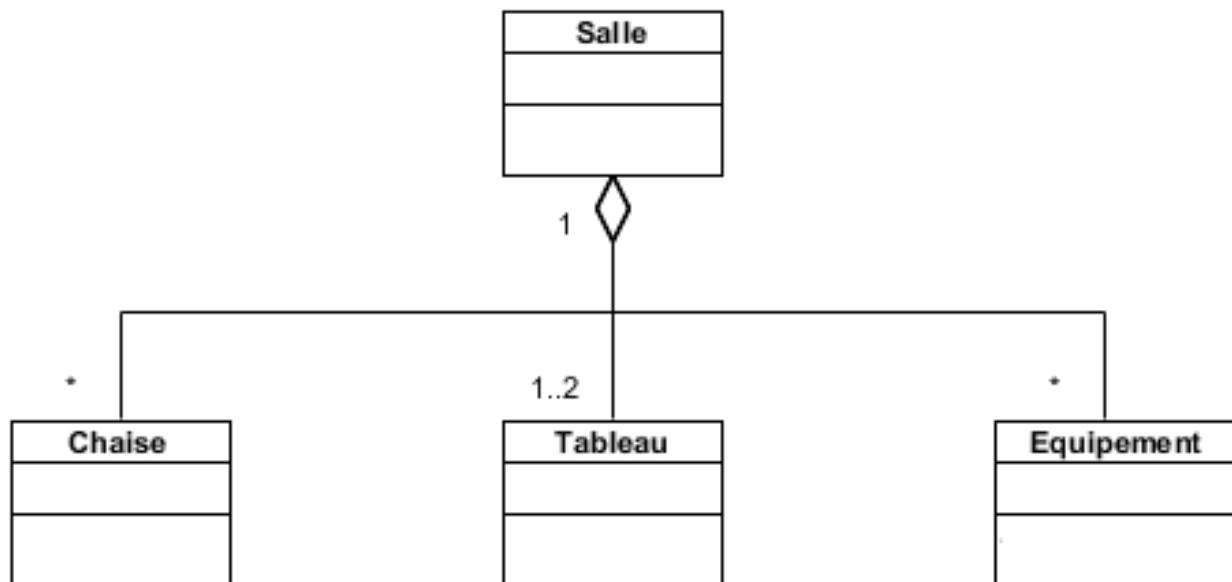
- Association reliant plus de deux classes.



Agrégation et composition (1/2)

Agrégation:

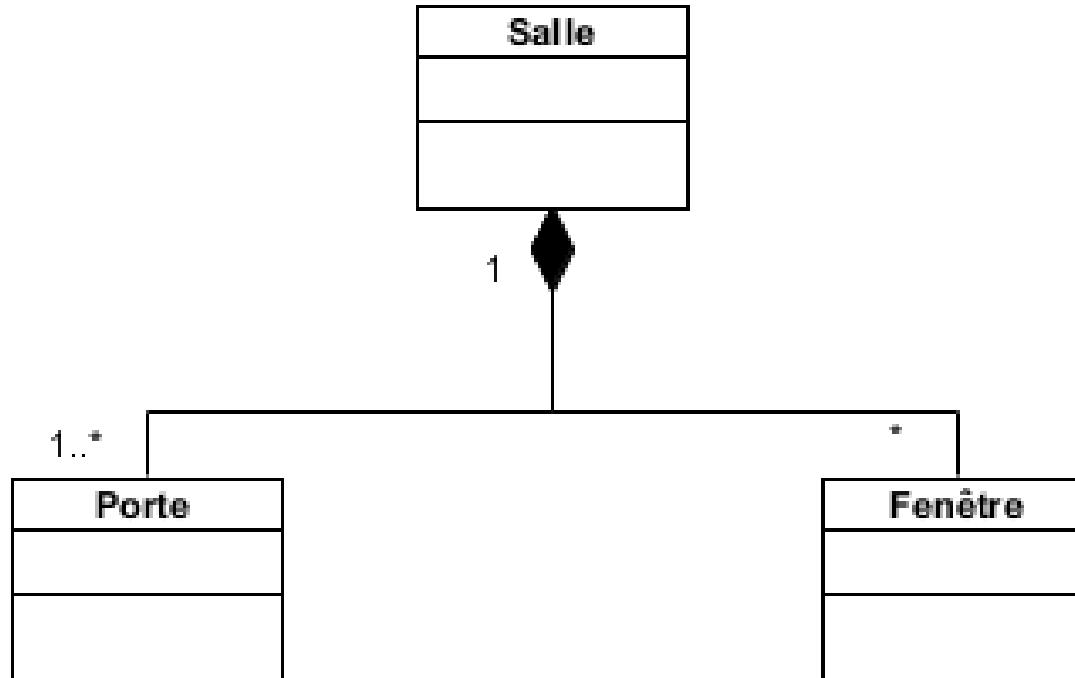
- Pas besoin d'être nommées : « contient », « est composée de » ;
- Durées de vie indépendantes des objets.



Agrégation et composition (2/2)

Composition :

- Besoin d'être nommées : « contient », « est composée de »;
- Durées de vie liées composants/composé.

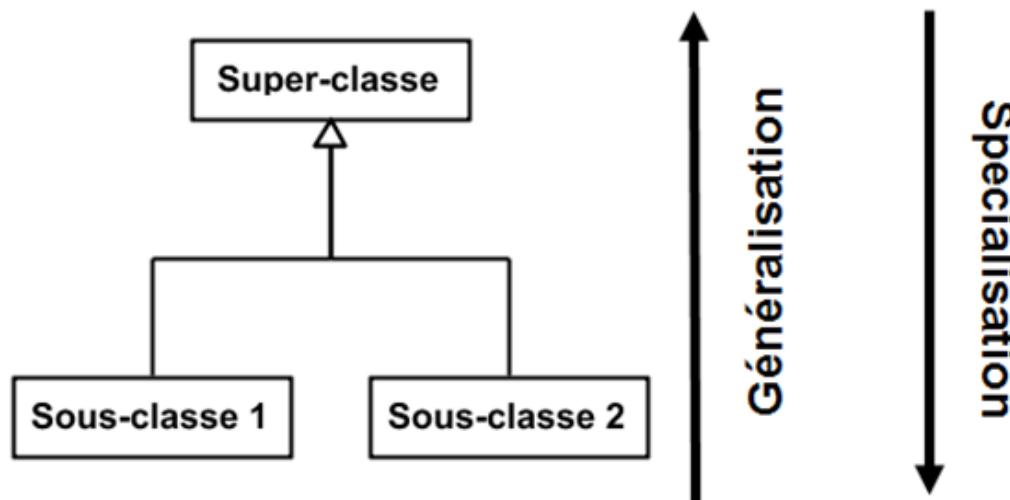


Généralisation et spécialisation (1/4)

Généralisation et spécialisation :

Elles sont des points de vue portés sur les hiérarchies de classes.

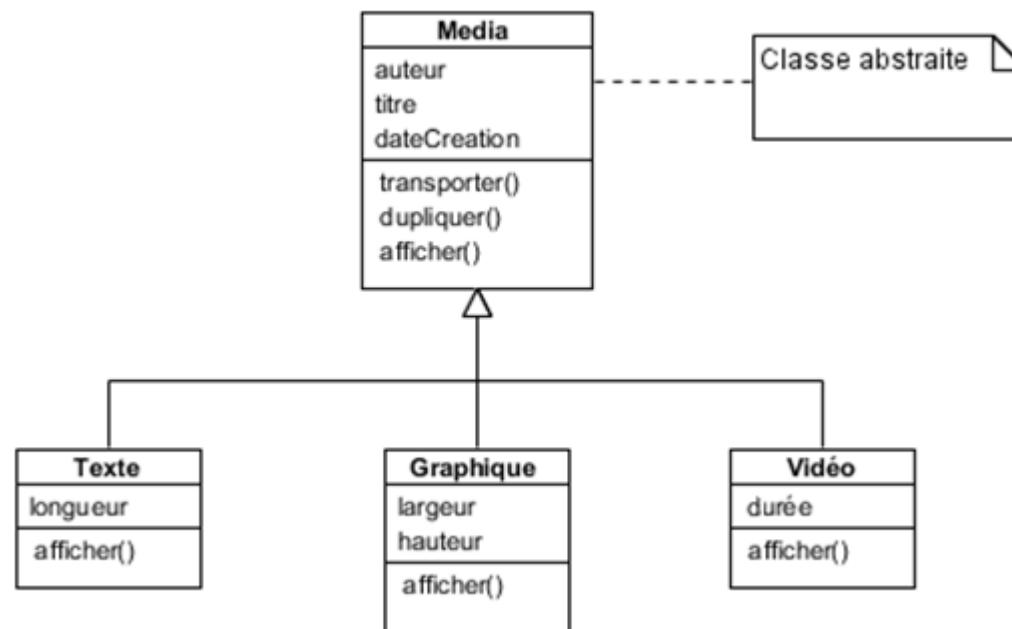
- Généralisation :
 - ✓ consiste à factoriser dans une classe, appelée superclasse, les attributs et/ou opérations des classes considérées ;
 - ✓ permet de réaliser une hiérarchie des classes.
- Spécialisation :
 - ✓ représente la démarche inverse de la généralisation ;
 - ✓ consiste à créer à partir d'une classe, plusieurs classes spécialisées ;
 - ✓ Chaque nouvelle classe créée est dite spécialisée puisqu'elle comporte en plus des attributs ou opérations de la super-classe (disponibles par héritage) des attributs ou opérations qui lui sont propres.



Généralisation et spécialisation (2/4)

Classe abstraite :

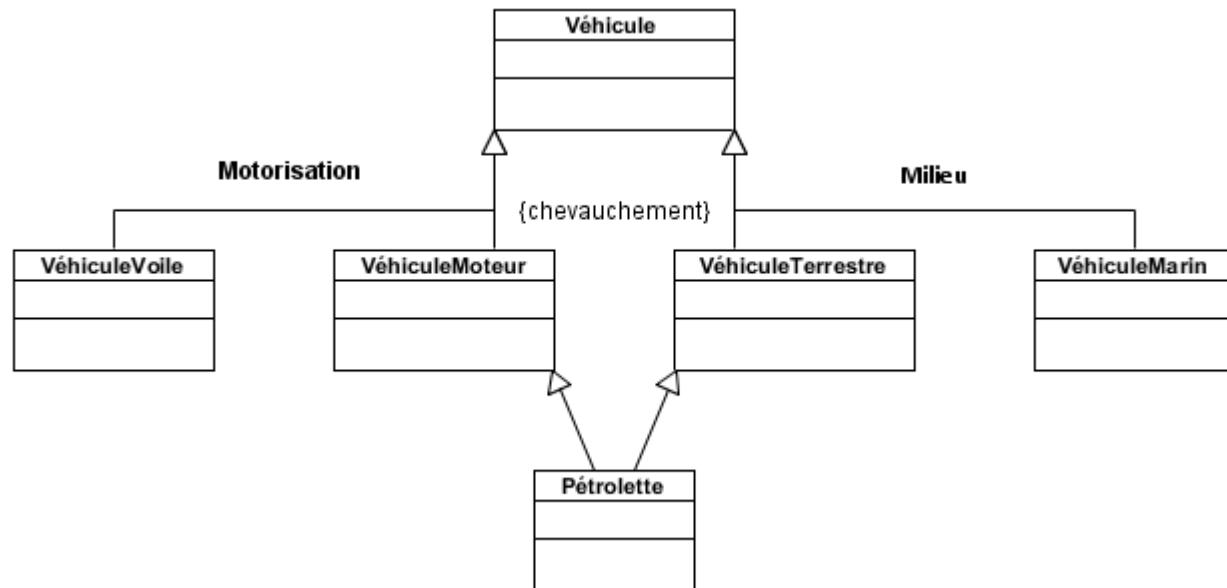
- Pas d'instance directe ;
- Classes descendantes ayant des instances ;
- Indiquée en *italique* de manière générale.



Généralisation et spécialisation (3/4)

Héritage avec recouvrement :

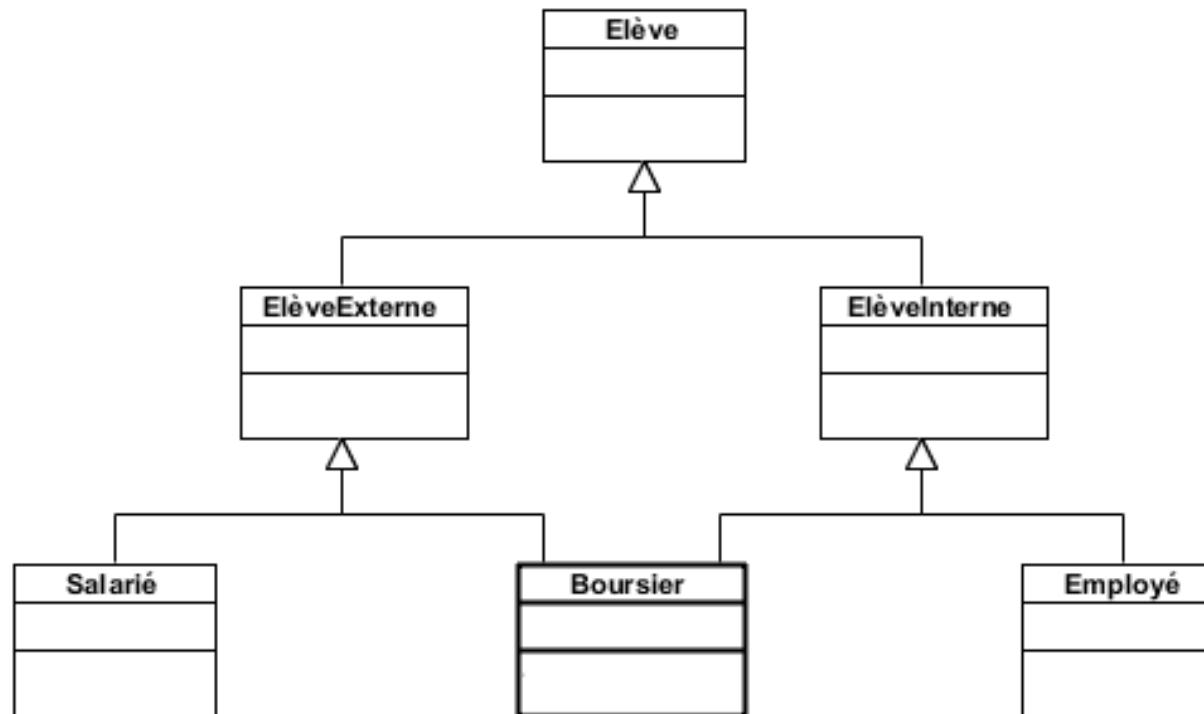
- { chevauchement } (ou { inclusif }) : deux sous-classes peuvent avoir, parmi leurs instances, des instances identiques ;
- { disjoint } (ou { exclusif }) : les instances d'une sous-classe ne peuvent être incluses dans une autre sous-classe de la même classe ;
- { complète } : la généralisation ne peut pas être étendue ;
- { incomplète } : la généralisation peut être étendue.



Généralisation et spécialisation (4/4)

Héritage multiple :

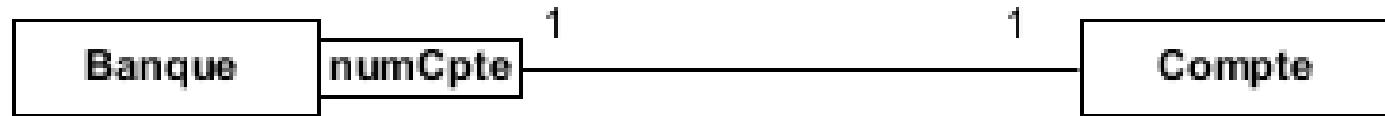
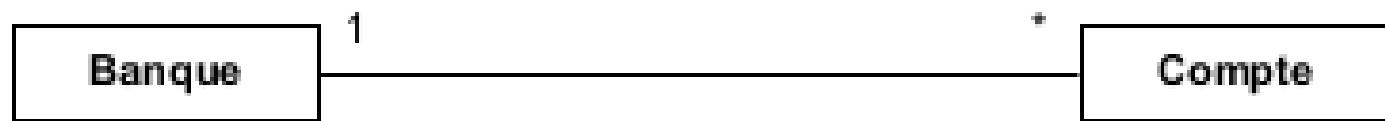
- Une classe peut hériter des plusieurs super-classes.



Association qualifiée, dépendance et classe d'interface (1/3)

Qualification :

- Relation entre deux classes permettant de préciser la sémantique de l'association et de qualifier de manière restrictive les liens entre les instances.



Association qualifiée, dépendance et classe d'interface (2/3)

Dépendance :

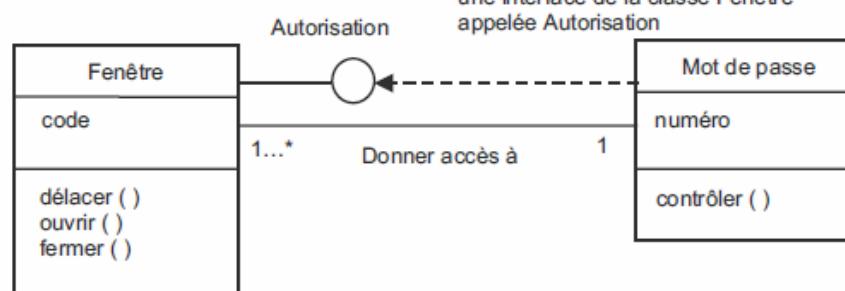
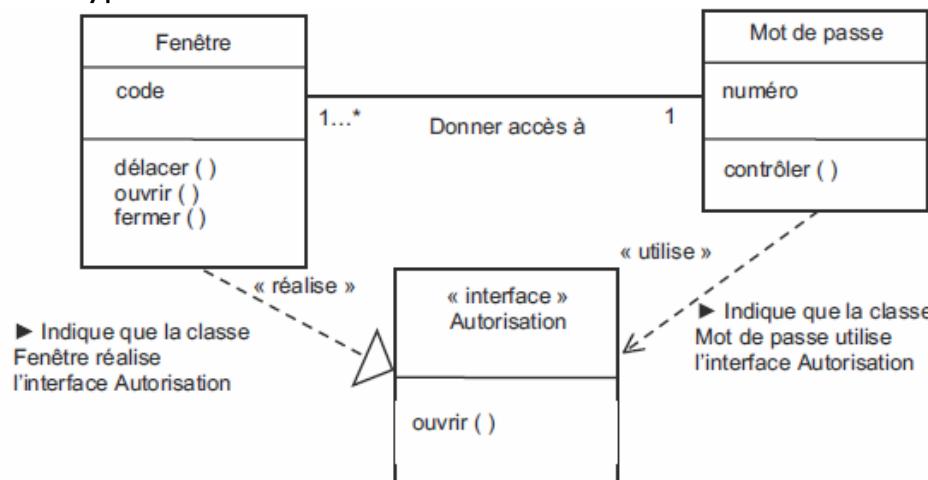
- Permet de représenter l'existence d'un lien sémantique ;
- Une classe B est en dépendance de la classe A si des éléments de la classe A sont nécessaires pour construire la classe B.



Association qualifiée, dépendance et classe d'interface (3/3)

Classe d'interface :

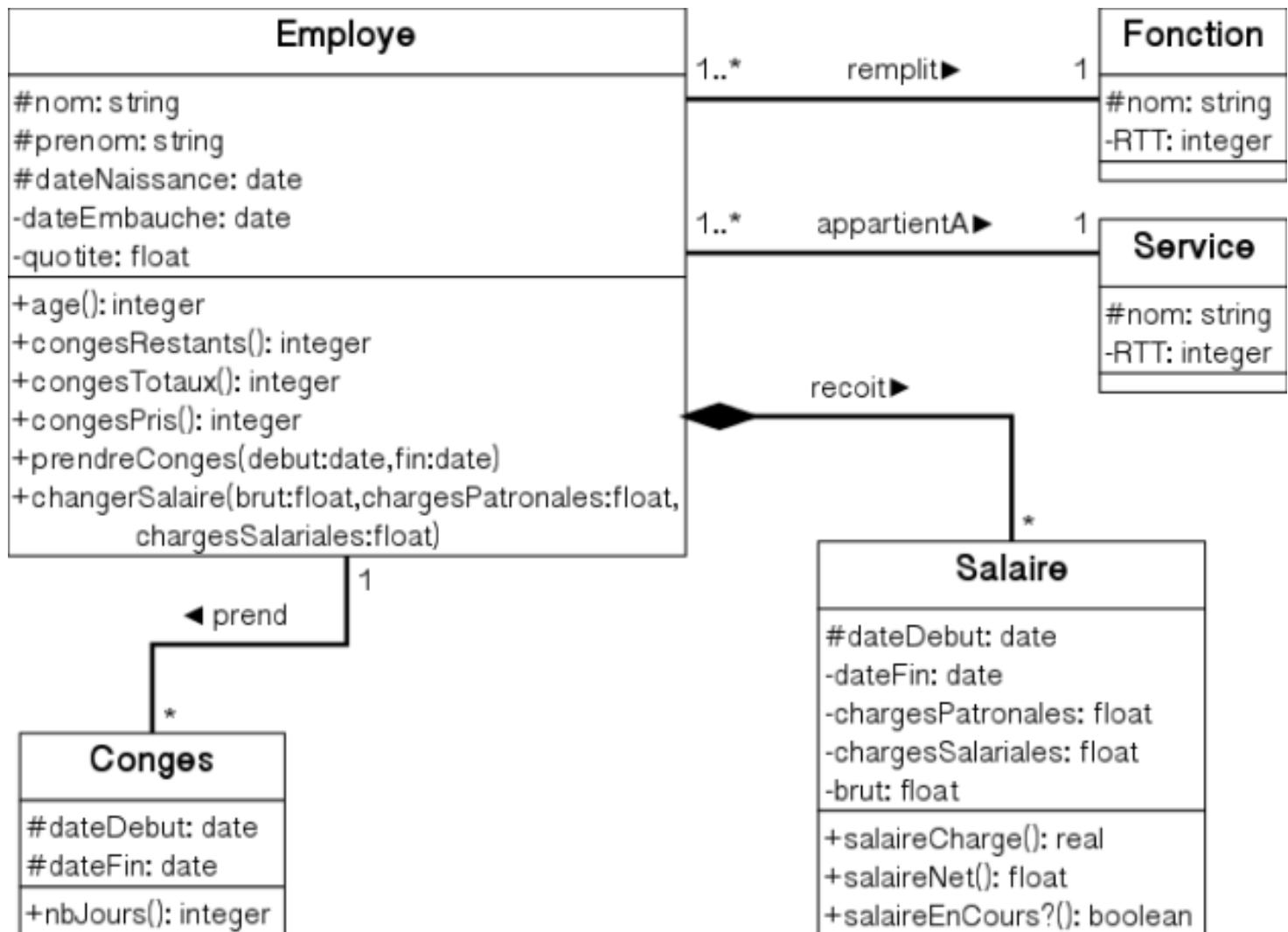
- Décrire la vue externe d'une classe ;
- Classe sans compartiment des attributs ;
- Comporte la liste des opérations accessibles par les autres classes ;
- Se note avec le stéréotype <<interface>> ou avec un rond.



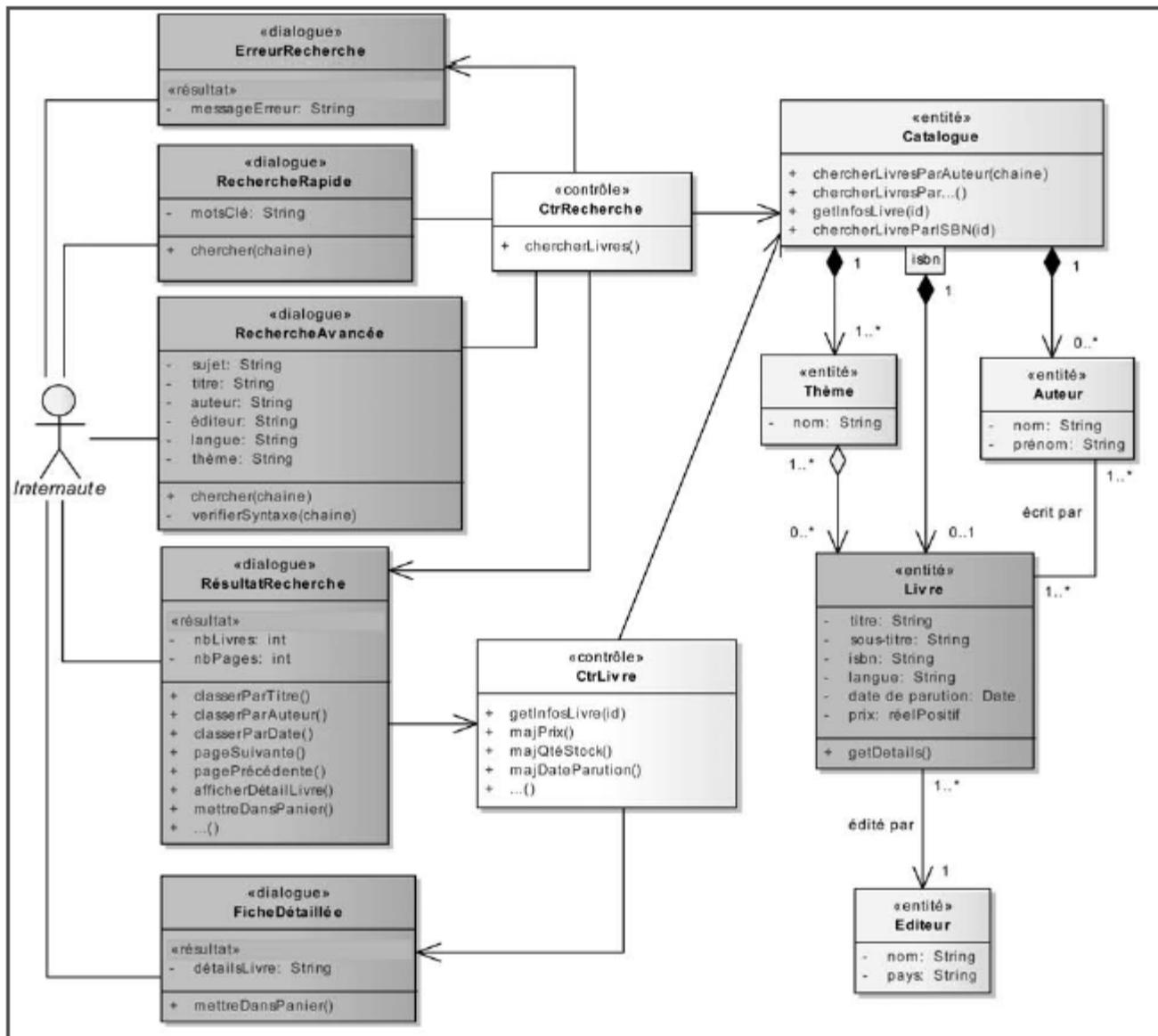
Quelques stéréotypes et mots-clés

- « Classe d'implémentation » : ce stéréotype est utilisé pour décrire des classes de niveau physique.
- « Type » : ce stéréotype permet de spécifier des opérations applicables à un domaine d'objets.
- « Utilitaire » : ce stéréotype qualifie toutes les fonctions utilitaires de base utilisées par les objets.
- « MétaClasse » : ce stéréotype permet de regrouper des classes dans une famille de classe.

Exemple d'un diagramme de classes



Exemple d'un diagramme de classes de conception



Conseils

- Ne pas confondre classe et acteur;
- Relation 1-1 possible mais à justifier;
- Une classe doit:
 - ✓ Correspondre à une seule responsabilité;
 - ✓ Représenter une abstraction pertinente;
 - ✓ Etre bien nommée;
 - ✓ Etre complètement décrite à un endroit du document;
- Cohérence entre les diagrammes.



DIAGRAMMES STRUCTURELS

DIAGRAMME D'OBJETS

Diagramme d'objets...?

- Aussi appelé diagramme d'instance ;
- Ensemble d'objets respectant les contraintes du diagramme de classes :
 - ✓ respect des cardinalités ;
 - ✓ chaque attribut d'une classe a une valeur affectée dans chaque instance de cette classe.
- Facilite la compréhension de structures complexes.

Objet (1/2)

- Instance particulière d'une classe :
- ✓ Nom de l'objet ;

Syntaxe d'un objet (comme dans le diagramme de séquence et le diagramme de communication) :

nomObjet : NomClasse

- ✓ Valeurs d'attribut (optionnelles);

Syntaxe pour la valeur d'attribut :

nomAttribut = valeurAttribut

Exemple:

<u>:Etudiant</u>
nom = roger
age = 20

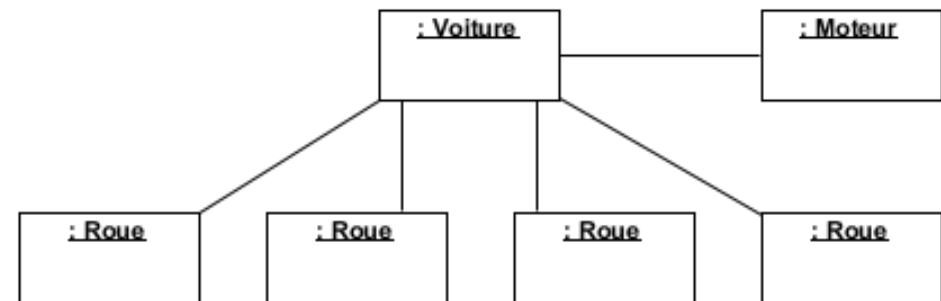
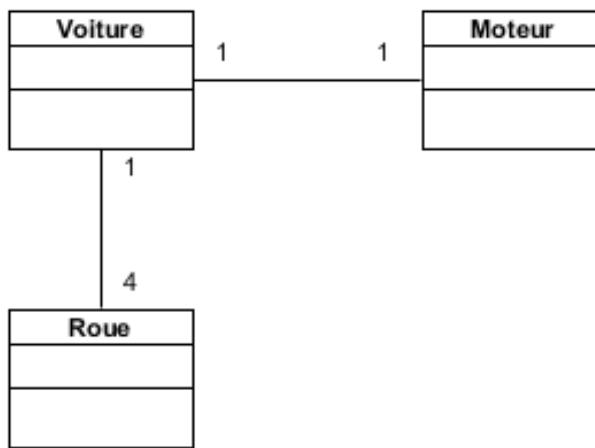
Objet (2/2)

Nommage de l'objet :

Nommant format	Notation
Un objet d'une classe non spécifiée.	<div style="display: flex; justify-content: space-around;"><div>objet : <input type="text"/></div><div>objet <input type="text"/></div></div>
Un objet nommé d'une classe spécifiée.	<div style="border: 1px solid black; padding: 5px;">objetX : Classe</div>
Un objet sans nom d'une classe spécifiée.	<div style="border: 1px solid black; padding: 5px;">: Classe</div>

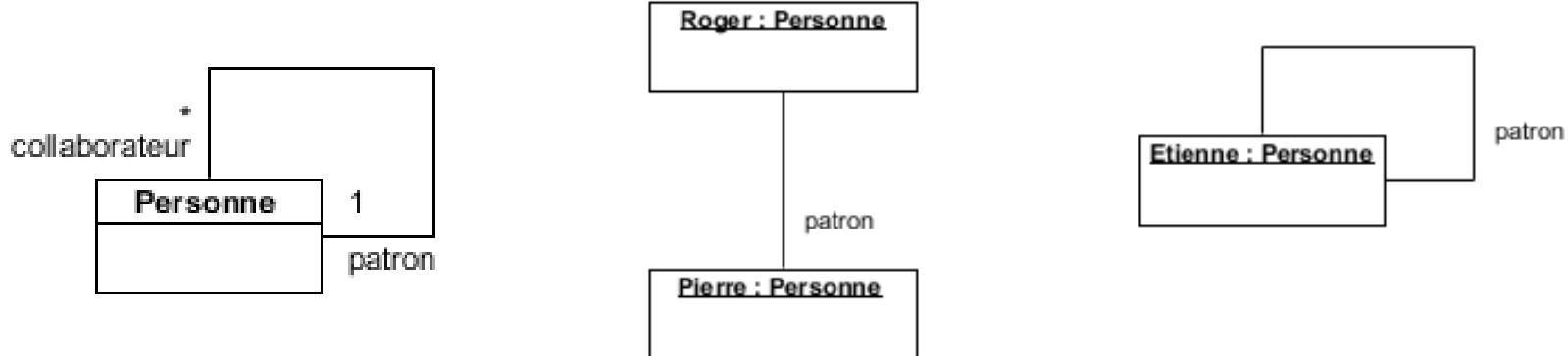
Liens (1/4)

- Les valeurs des attributs sont optionnelles ainsi que les liens entre objets.



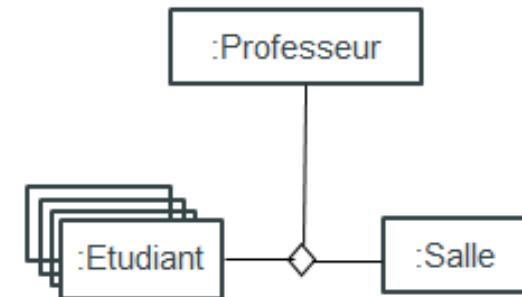
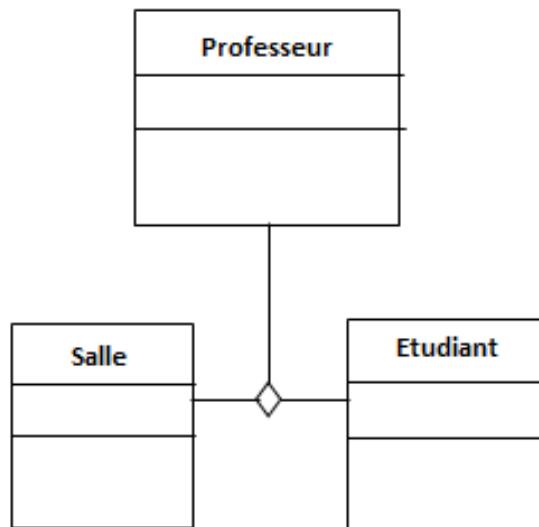
Liens (2/4)

- Les liens instances des associations réflexives peuvent relier un objet à lui-même.



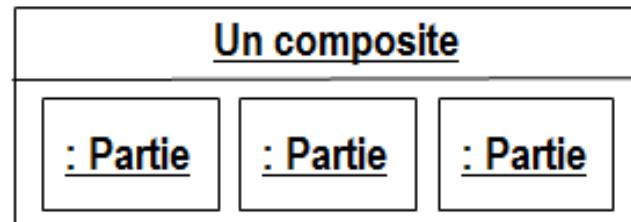
Liens (3/4)

- Les liens d'arité supérieure à 2 ou la multiplicité peuvent être représentés.



Liens (4/4)

- Les objets composés de sous-objets peuvent être représentés au moyen d'un objet composite.



- Les objets composites sont instances de classes composites.

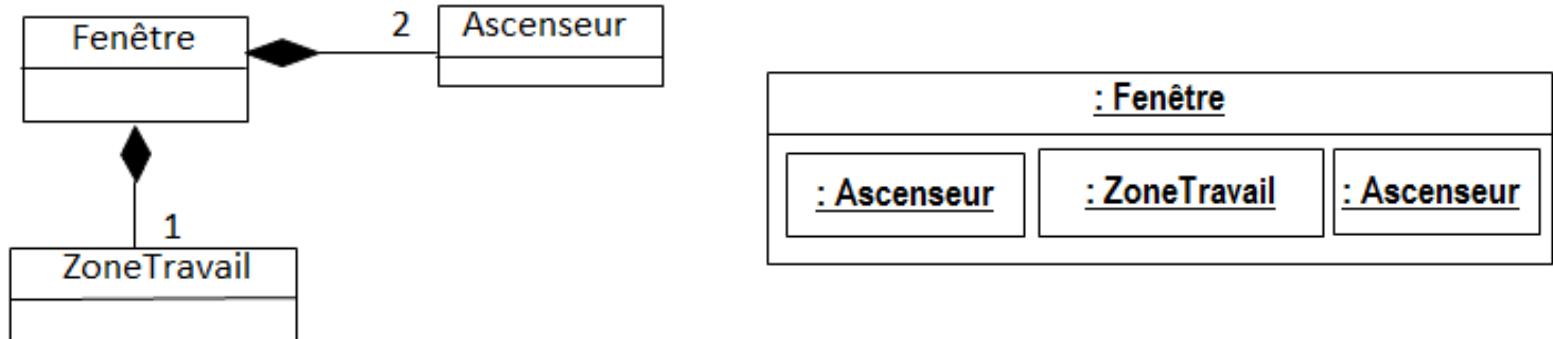
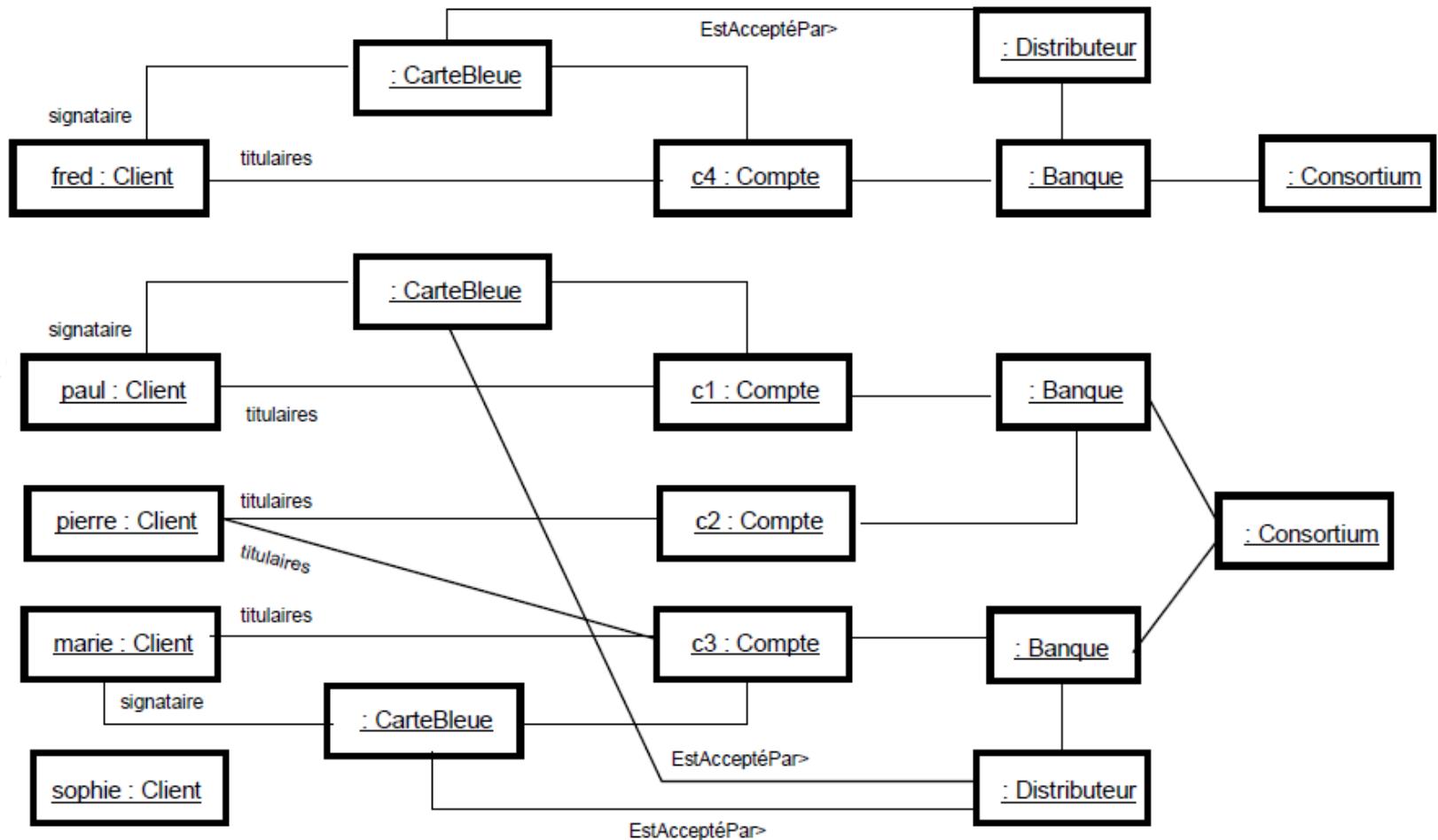


Diagramme de classes et Diagramme d'objets

- Le diagramme de classes représente un cas général.
- Le diagramme d'objets représente un cas particulier.

Exemple d'un diagramme d'objets





DIAGRAMMES STRUCTURELS

DIAGRAMME DE STRUCTURES COMPOSITES

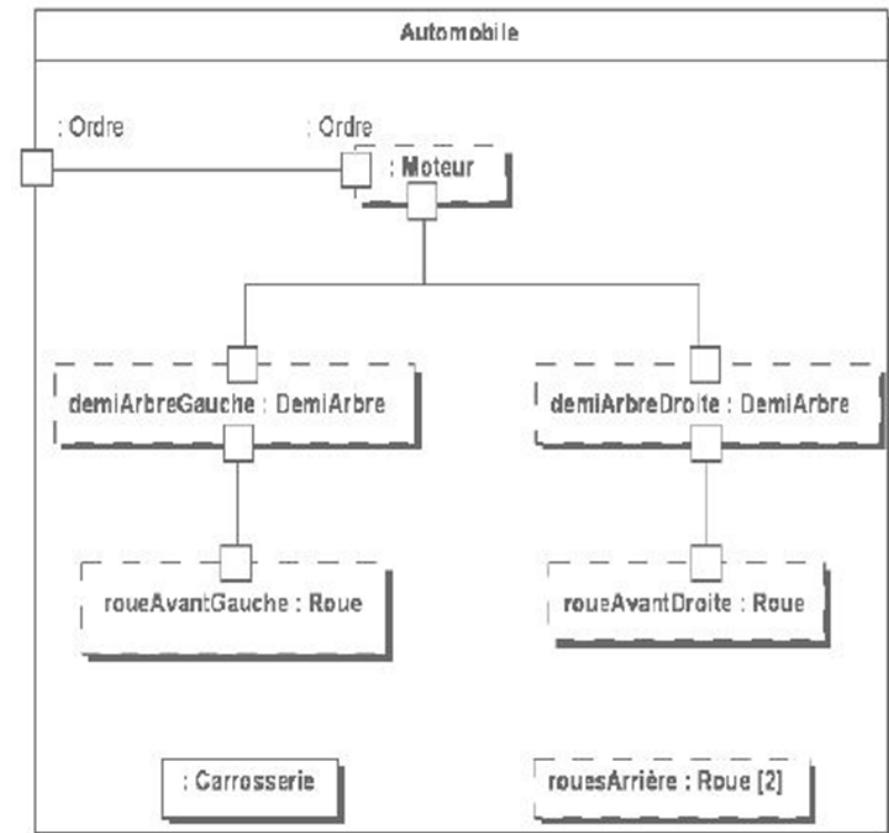
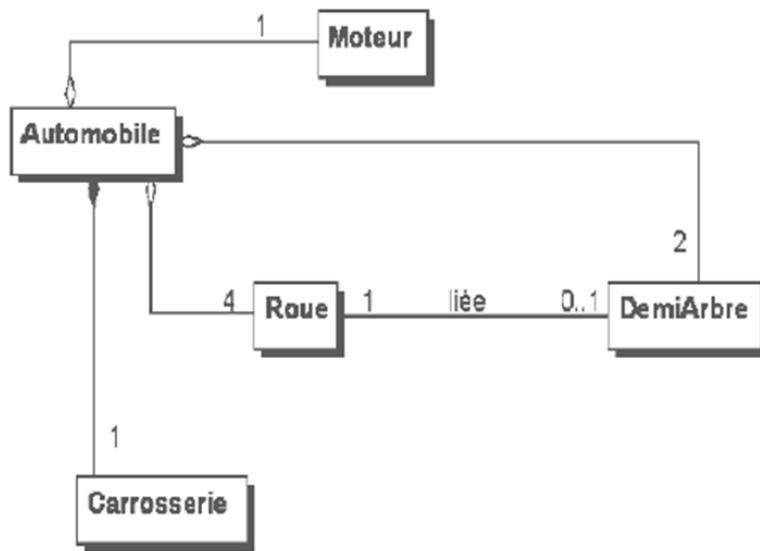
Diagramme de structures composites...?

- Aussi appelé diagramme d'architecture ;
 - Nouveau diagramme d'UML 2.x ;
 - Représentation de la collaboration d'instances de classes via des liens de communication ;
 - Même rôle qu'un diagramme de classes ;
-
- Permet d'afficher :
 - ✓ la structure interne d'un classificateur ;
 - ✓ les interactions avec l'environnement par le biais des ports ;
 - ✓ un comportement d'une collaboration.

Classificateur, port , et partie

- Classificateur :
 - ✓ Décrit les caractéristiques structurelles et comportementales.
- Port :
 - ✓ Relie les classificateurs avec son environnement ou ses parties internes ;
 - ✓ Spécifier les points d'interactions d'un objet.
- Partie :
 - ✓ Représente une ou plusieurs instances d'une classe grâce à des contraintes de multiplicité;
 - ✓ Zone bien délimitée à l'intérieur d'une classe ou d'un composant.

Exemple





DIAGRAMMES STRUCTURELS

DIAGRAMME DE COMPOSANTS

Diagramme de composants...?

- Permet de représenter les composants logiciels d'un système, et les liens existant entre ces composants ;
- Pour les systèmes complexes.

Composant

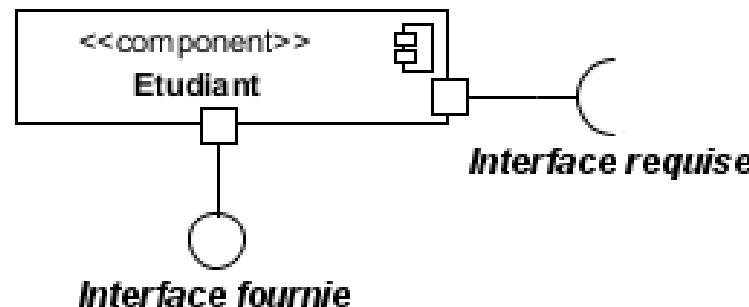
- Partie modulaire d'un système qui encapsule son contenu ;
- Définir le comportement en termes d'interfaces fournies et requises ;
- Il peut être :
 - ✓ les codes sources ;
 - ✓ les exécutables et bibliothèques ;
 - ✓ les tables, les fichiers, les documents.
- Caractérisé par :
 - ✓ un nom ;
 - ✓ un port de connexion ;
 - ✓ une spécification externe sous forme soit d'une ou plusieurs interfaces requises, soit d'une ou plusieurs interfaces fournies.

<<component>>
NonComposant

NonComposant

Port de connexion et interface de composant

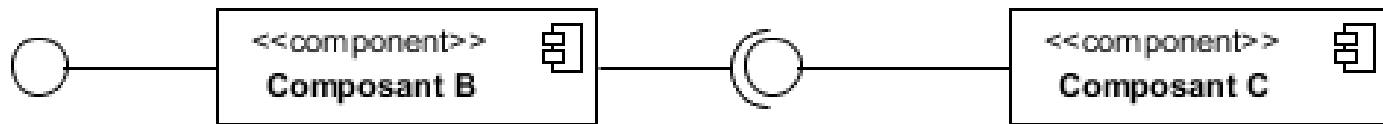
- Port de connexion :
 - ✓ Représente le point de connexion entre le composant et une interface.
- Interface de composant :
 - ✓ Interaction entre composants au travers des interfaces fournies et requises ;
 - ✓ Une interface requise est une interface nécessaire au bon fonctionnement du composant ;
 - ✓ Une interface fournie est une interface proposée par le composant aux autres composants.



Relation entre composants (1/3)

Connecteur d'assemblage :

- Regroupement de deux composants par un connecteur d'assemblage.



Relation entre composants (2/3)

Dépendance :

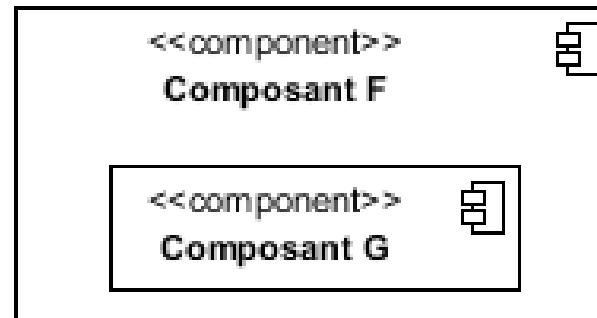
- Regroupé par la flèche de dépendance.



Relation entre les composants (3/3)

Contenance :

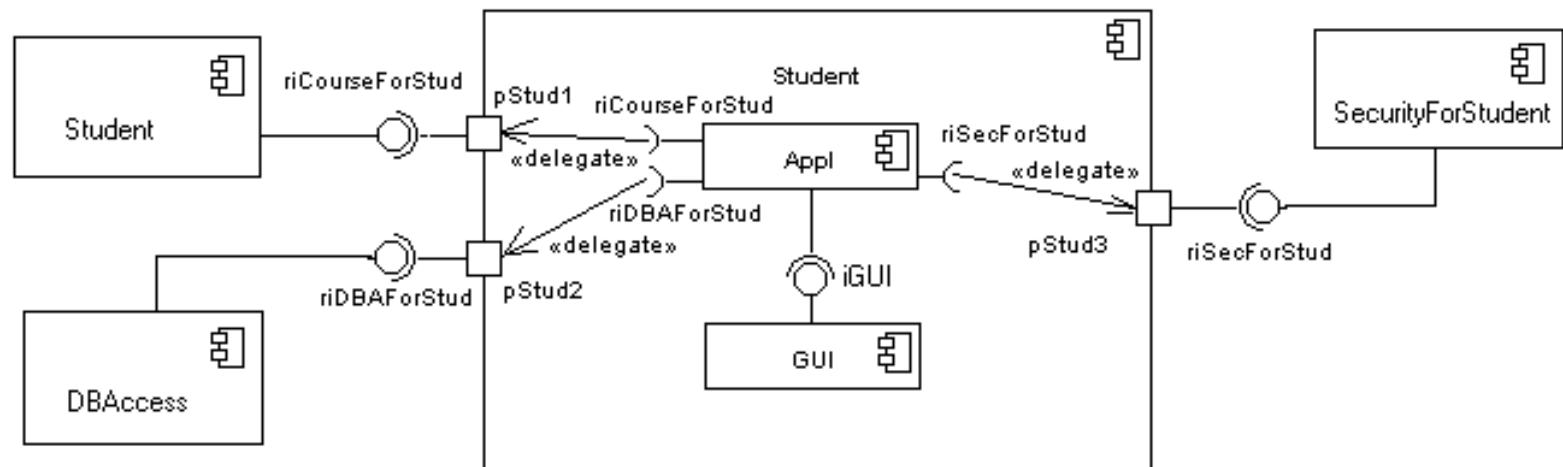
- Un composant peut être contenu dans un autre composant.



Connecteur de délégation

- L'interface fournie d'un composant peut être réalisée par l'une de ses parties internes ;
- Son interface requise peut être imposée par l'une de ses parties ;
- Les connecteurs de délégation montrent que ces parties internes réalisent ou utilisent les interfaces du composant, pouvant être stéréotypé « delegate ».

Exemple d'un diagramme de composants





DIAGRAMMES STRUCTURELS

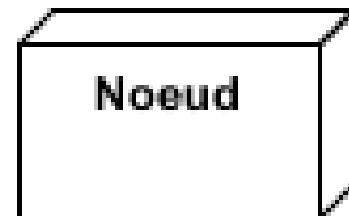
DIAGRAMME DE DEPLOIEMENT

Diagramme de déploiement...?

- Permet de représenter l'architecture physique supportant l'exploitation du système ;
- Constitué de « nœuds » connectés par des voies physiques ;
- Montre des instances de nœuds (un matériel précis), ou des classes de nœuds.

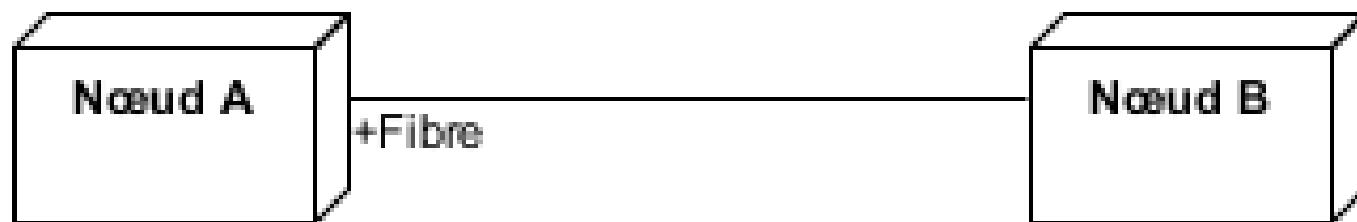
Nœud

- Représente un ensemble d'éléments matériels du système ;
- Interconnectés par intermédiaire d'un réseau d'éléments physiques ;
- Contient un ou plusieurs composants qui peuvent être liée entre eux ;
- Prend en charge l'exécution des composants ;
- Un nœud ou une instance de nœud se représente par un cube ou parallélépipède.

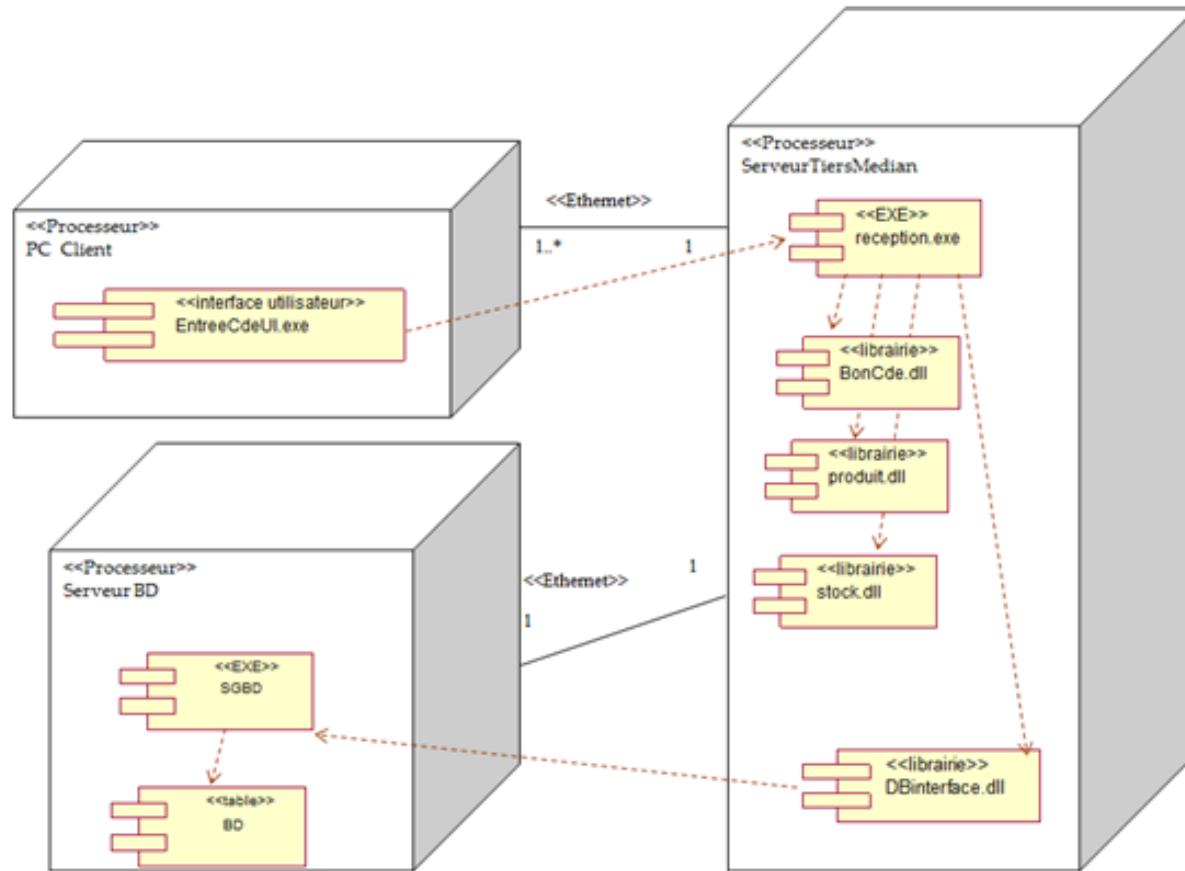


Relation entre nœuds (Association)

- Indique une voie physique (connexion ethernet, bus de communication, ...) entre deux nœuds.



Exemple d'un diagramme de déploiement





DIAGRAMMES STRUCTURELS

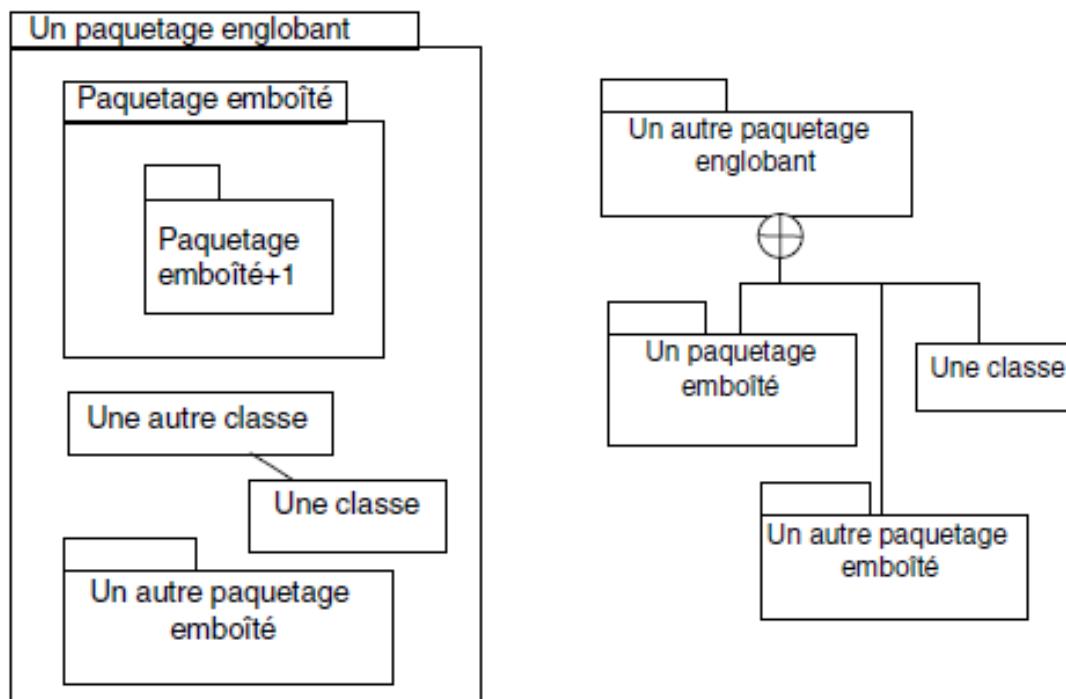
DIAGRAMME DE PAQUETAGES

Diagramme de paquetages...?

- Nouveau diagramme d'UML 2.x ;
- Représenter la structure hiérarchique et modulaire ;
- Permet de regrouper les éléments de modélisation.

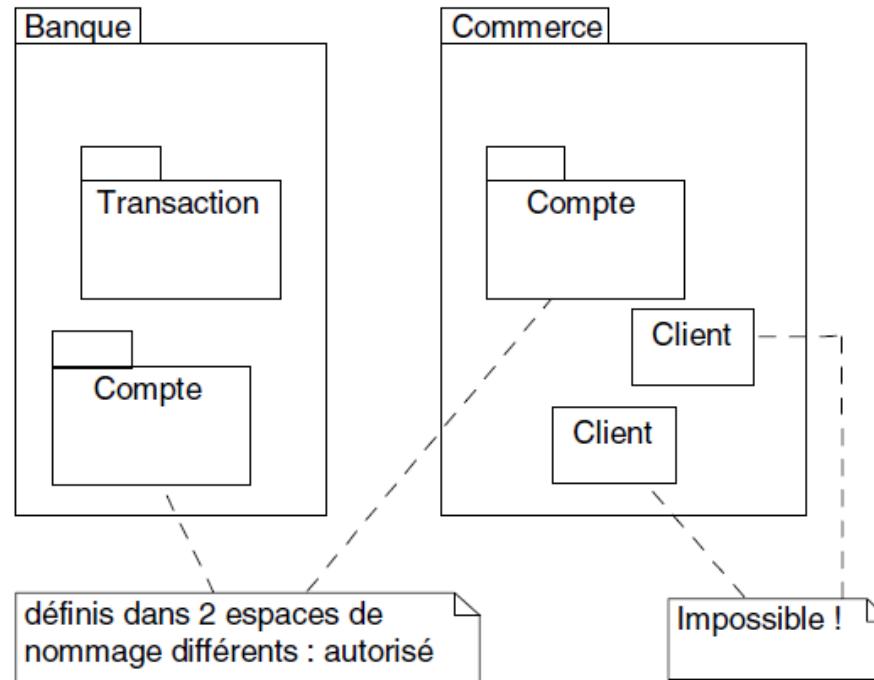
Paquetage

- Elément de modélisation qui contient d'autres éléments de modélisation : des classes, des interfaces, des composants, des nœuds, des cas d'utilisation, des diagrammes, d'autres paquetages... ;
- ✓ Possible de ne pas représenter tous les éléments contenus ;
- Peut importer les éléments d'un autre paquetage ;
- Peut-être fusionné avec un autre paquetage ;
- Elément peut avoir une visibilité déclarée soit de type public (+) soit privé (-).



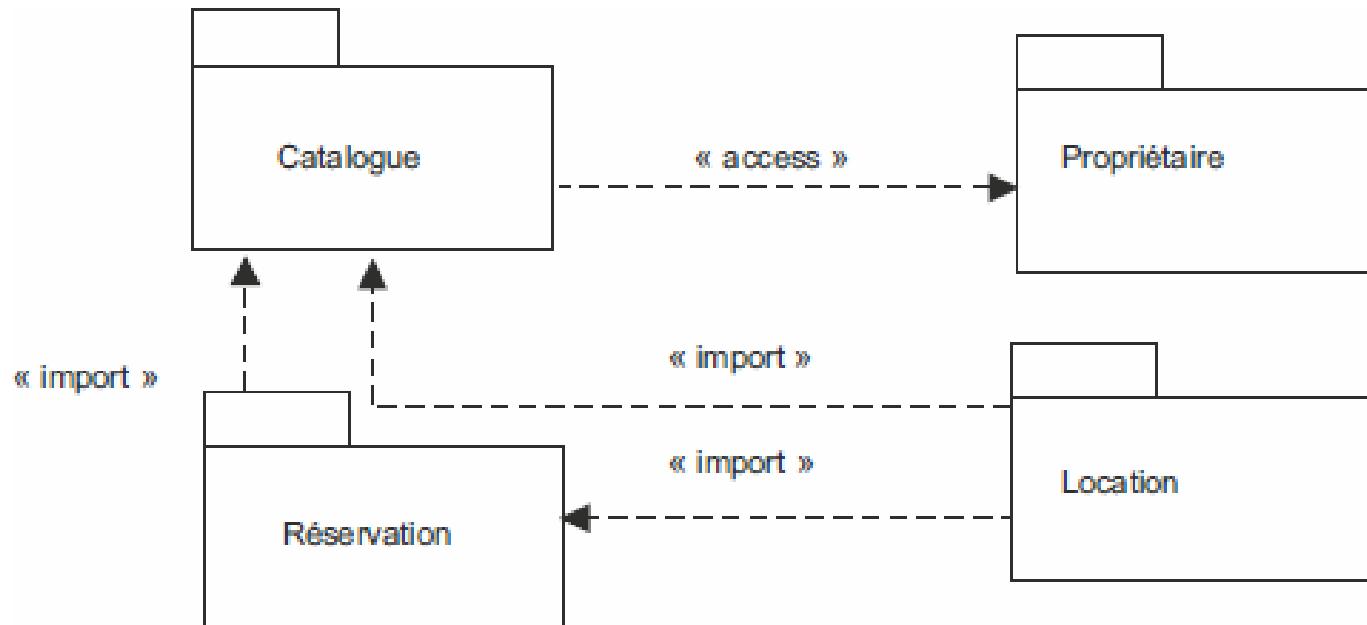
Espaces de nom

- Deux éléments ne peuvent pas avoir le même nom dans un paquetage ;
- Deux éléments dans deux paquetages différents sont différents, quel que soit leur nom ;
- Nom complet = nom préfixé par les noms des paquetages englobant : Banque::Compte ≠ Commerce::Compte



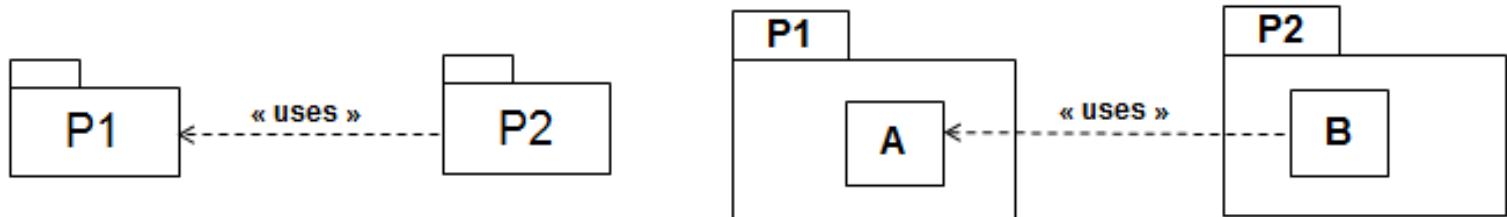
Dépendance entre paquetages (1/3)

- « import » :
 - ✓ correspond à un lien ayant une visibilité « public » ;
 - ✓ permet d'importer l'espace de nommage d'un autre paquetage ;
 - ✓ tous les membres du paquetage donné ont accès à tous les noms des membres du paquetage importé sans avoir à utiliser explicitement le nom du paquetage concerné.
- « access » :
 - ✓ correspond à un lien ayant une visibilité « privé » ;
 - ✓ permet d'avoir accès à l'espace de nommage d'un paquetage cible ;
 - ✓ L'espace de nommage n'est donc pas importé et ne peut être transmis à d'autres paquetages par transitivité.



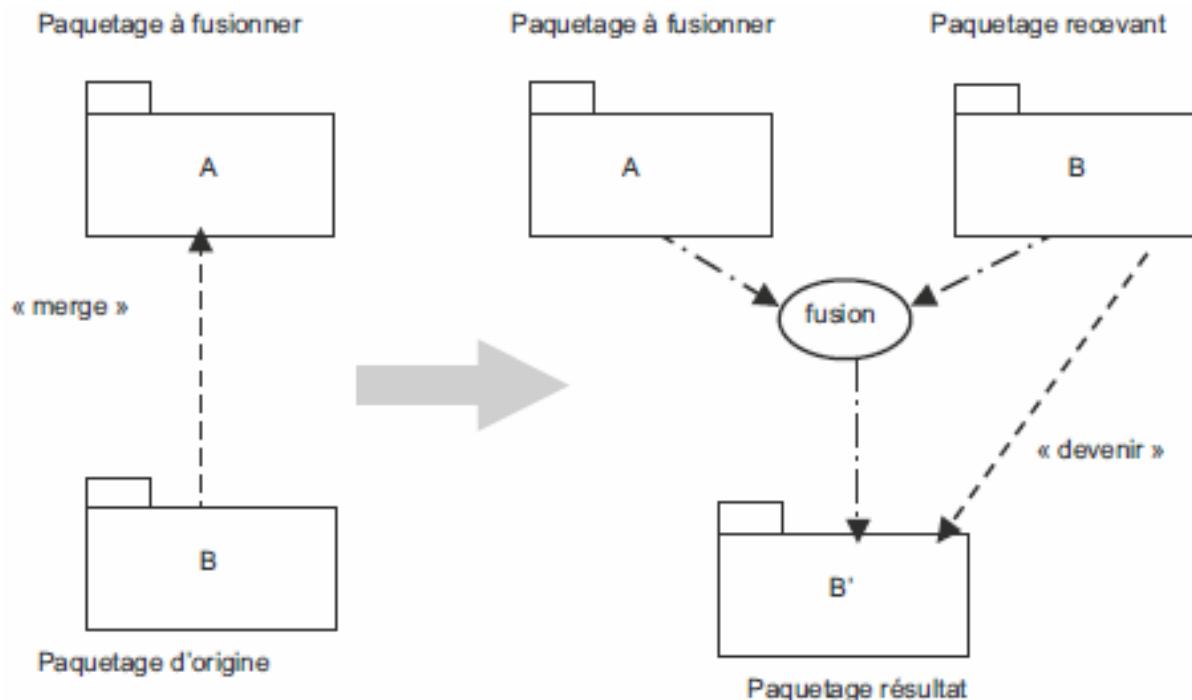
Dépendance entre paquetages (2/3)

- « uses » :
 - ✓ B nécessite A pour sa mise en œuvre ou son fonctionnement ;
 - ✓ La nature de l'utilisation peut être :
 - l'invocation d'une opération ;
 - la création d'un objet.



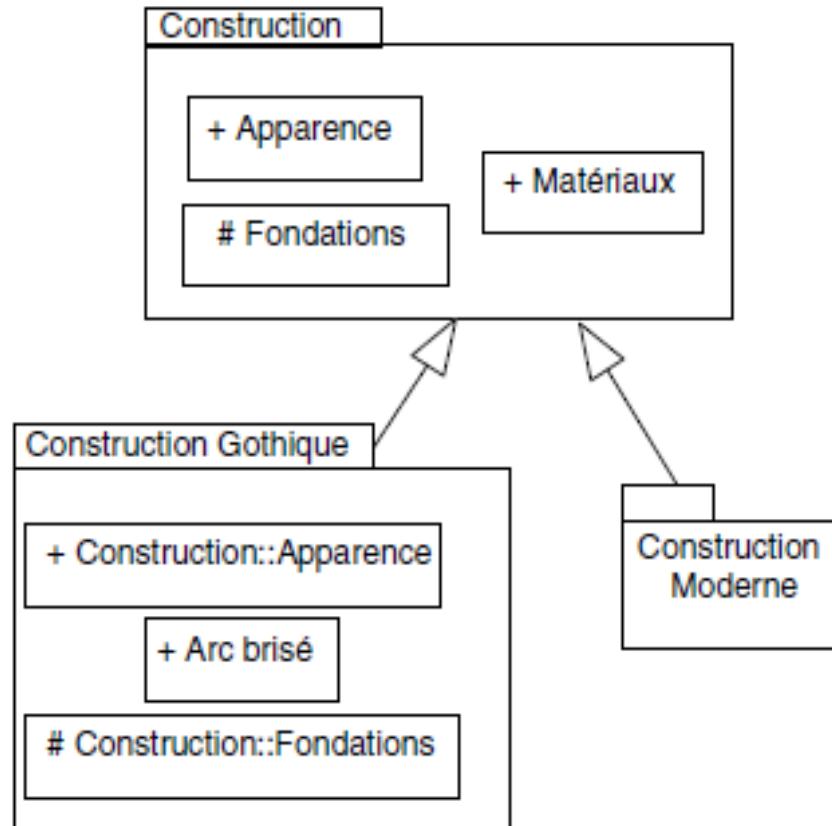
Dépendance entre paquetages (3/3)

- « merge » :
 - ✓ Permet de fusionner deux paquetages et d'obtenir ainsi un paquetage contenant la fusion des deux paquetages d'origine.

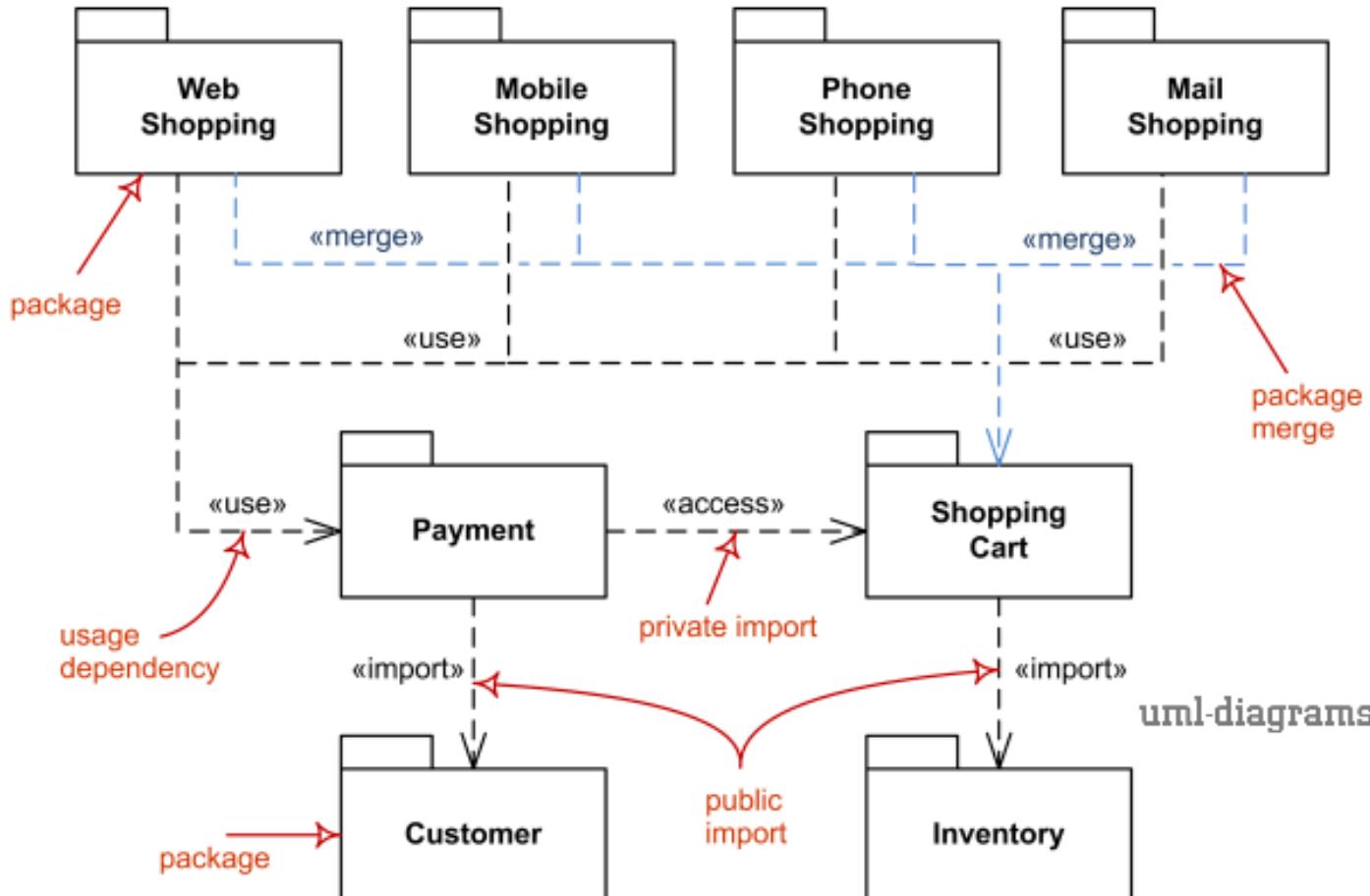


Généralisation des paquetages

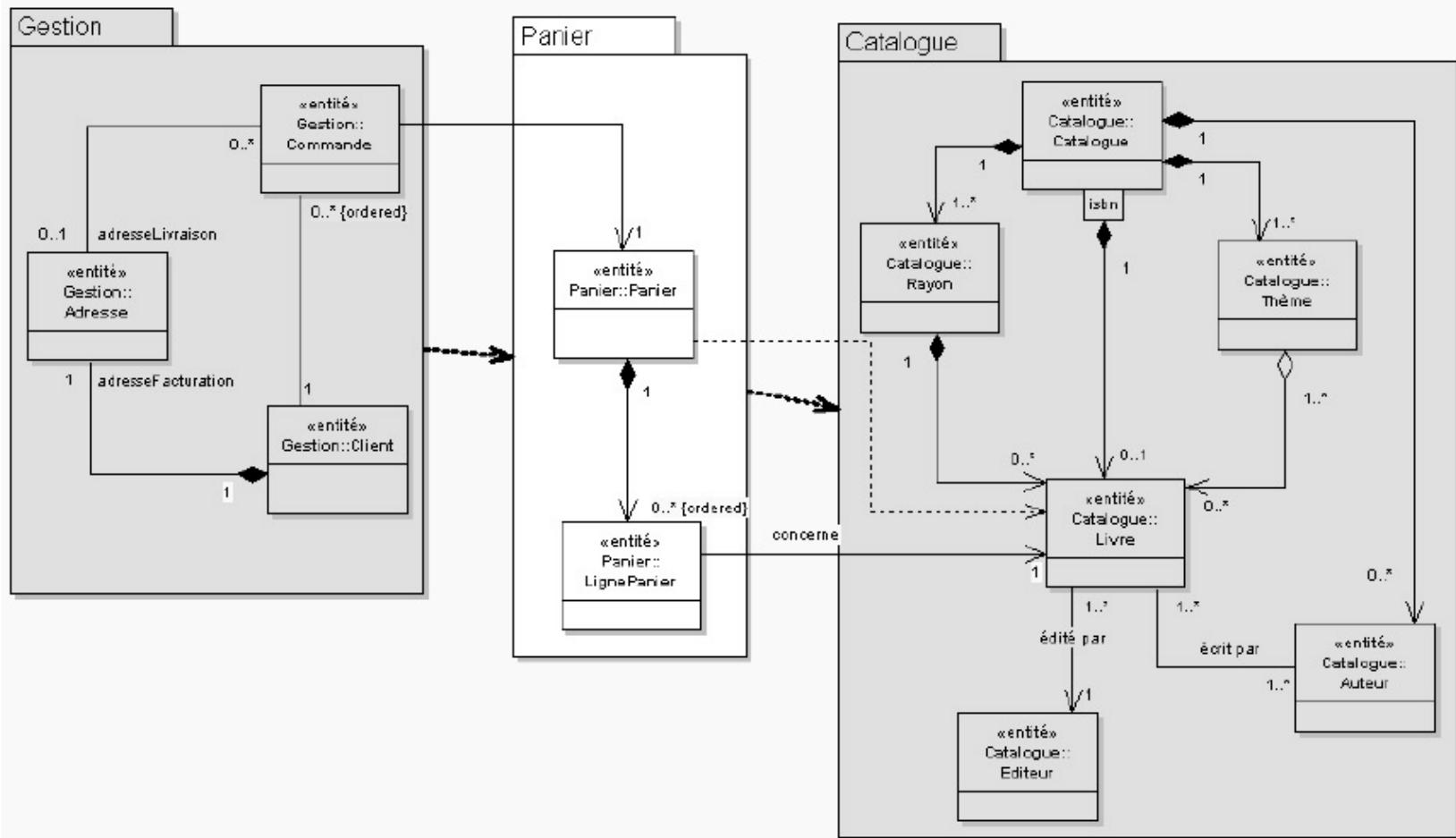
- Comparable à celle de la classe ;
- Possible d'ajouter ou de spécialiser les éléments de paquetages en modélisant les relations de généralisation entre paquetages ;
- Les éléments publics et protégés sont alors accessibles dans les paquetages de spécialisation ; les éléments privés non.



Exemple d'un diagramme de paquetages (1/2)



Exemple d'un diagramme de paquetages (2/2)



Outils supportant UML

Critères de sélection des outils UML

- Respect des normes UML ;
- Plateforme supportés ;
- Stabilité;
- Exhaustivité des diagrammes ;
- Correspondance relationnel ;
- Licence ;
- Code source ouverte ;
- Génération de code ;
- Rétro-ingénierie;
- Format de documentation ;
- Gestion de configuration ;
- Gestion de tests ;
- Intégration IDE.

Outils UML

- ArgoUML
- BOUML
- Enterprise Architect
- Modelio
- Objecteering
- Open ModelSphere
- Poseidon
- Rational Rose
- StarUML
- Visual Paradigm for UML
- WinDesign Module OBJECT
- ...

Bibliographie et webographie

Bibliographie

- Fien VAN DER HEYDE et Laurent DEBRAUWER (2008). UML 2 : Initiation, exemples et exercices corrigés (2ième édition). Saint-Herblain : Editions ENI.
- Hugues M., Jean-Pierre S. et Gilles M. Règles de cohérence UML 2.0 (Version 1.1).
- James R., Ivar J. et Grady B. (2004). The Unified Modeling Language - Reference Manual (2e ed.). Canada : Addison Wesley.
- Joseph Gabay et David Gabay (2008). UML 2 : Analyse et Conception - Mise en œuvre guidée avec études de cas. Paris : Dunod.
- OMG (2003). UML 2.0 Superstructure Specification.
- Pascal Roques (2008). Les Cahiers du Programmeur : UML 2 - Modéliser une application web (4ème édition). France : Editions Eyrolles.
- Pascal Roques et Franck Vallée (2007). UML 2 en action - De l'analyse des besoins à la conception (4ème éd.). France : Editions Eyrolles.
- Pierre-Alain Muller et Nathalie Gaertner (2004). Modélisation objet avec UML. France : Editions Eyrolles.
- Pierre-Alain Muller (1997). Modélisation objet avec UML. France : Editions Eyrolles.

Webographie

- <http://bdd.crzt.fr/mod/prs/co/modUE04.html?mode=html>
- <http://uml.free.fr/>
- <http://www.omg.org/>
- <http://www.uml-diagrams.org>
- <https://openclassrooms.com>
- <https://www.ibm.com>
- www.agilemodeling.com/essays/umlDiagrams.htm