Modélisation des Systèmes Informatiques (MSI)

Responsable : Pr. Samba DIAW

Audience: DUT 2/DST 2 INFORMATIQUE

Plan

- Généralités
- Formalisme de modélisation
  - MERISE
  - UML
  - UML vs MERISE
- Etude détaillée du Langage UML
- Initiation à l'utilisation d'un AGL

Généralités

Notion de Systéme

- Un système est un dispositif formé par la réunion d'éléments analogues. En d'autres termes un système est une combinaison de parties qui se coordonnent pour concourir à un résultat, de manière à former un ensemble.
- Tout système fonctionne en transformant des flux d'entrée en flux de sortie selon des processus plus ou moins complexes.

Système d'Information d'une organisation

- **Un système d'information** (noté SI) représente l'ensemble des éléments qui participent, à la saisie, au stockage, au traitement, au transport et à la diffusion de l'information au sein d'une organisation (entreprise, mairie, association).
- Fonctions
  - Saisie des informations
  - Stockage des informations
  - Traitement des informations
  - Transmission et diffusion des informations
  - Restitution des informations

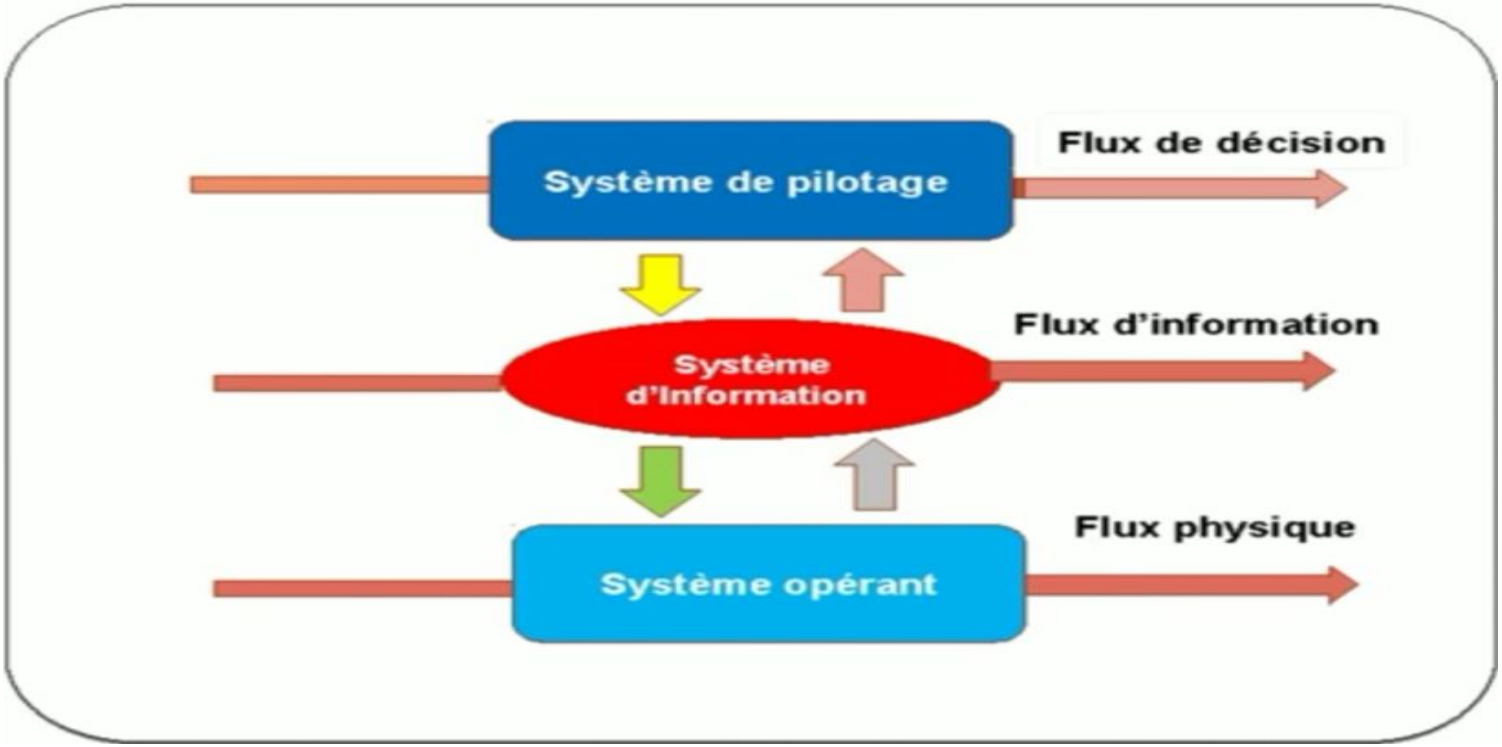
Notion de Système Informatique

- C’est la partie automatisable du Système d’information,
- On parle aussi de Système d’Information automatisé (SIA)

Organisation d’un SI

SI: Interface entre les Systèmes opérant et de pilotage

- Le SI peut recevoir du Système de pilotage des décisions destinées à son propre pilotage.
- Le SI peut émettre vers le système opérant des informations-interactions, c'est-à-dire qu'il peut réagir sur le système opérant .



Système opérant et système de pilotage

- Un système physique ou système opérant est l'ensemble des moyens humains, matériels, organisationnels qui exécutent les ordres du système de pilotage.
- Par ailleurs un système de pilotage procède à la régulation et au contrôle du système opérant. Il définit les missions et les objectifs, organise l'emploi des moyens et coordonne l'exécution des travaux.
  - Ce système se compose par exemple de la direction financière, de la direction commerciale, de la direction des ressources humaines, etc.

Système opérant et système de pilotage

- Le système de pilotage reçoit du système opérant des informations sur l'état du système qui lui permettent de mesurer l'écart avec les objectifs
- Il réagit par des décisions sur le processus du système opérant par régulation des flux (fixation des cadences de production, décision de lancer un nouveau produit ou de modifier le prix de vente d'un article).

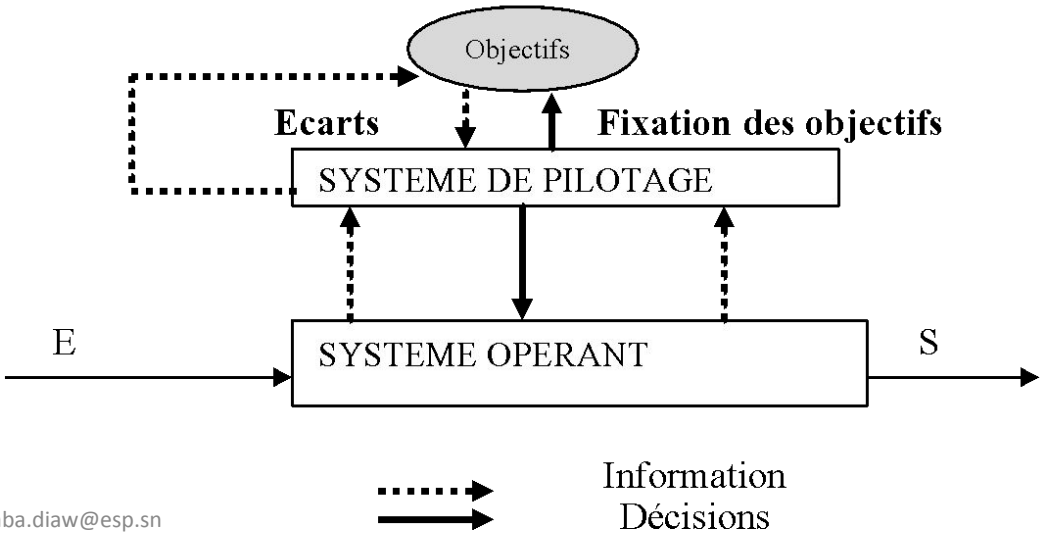


Illustration d'un SI d'une entreprise de vente de produits

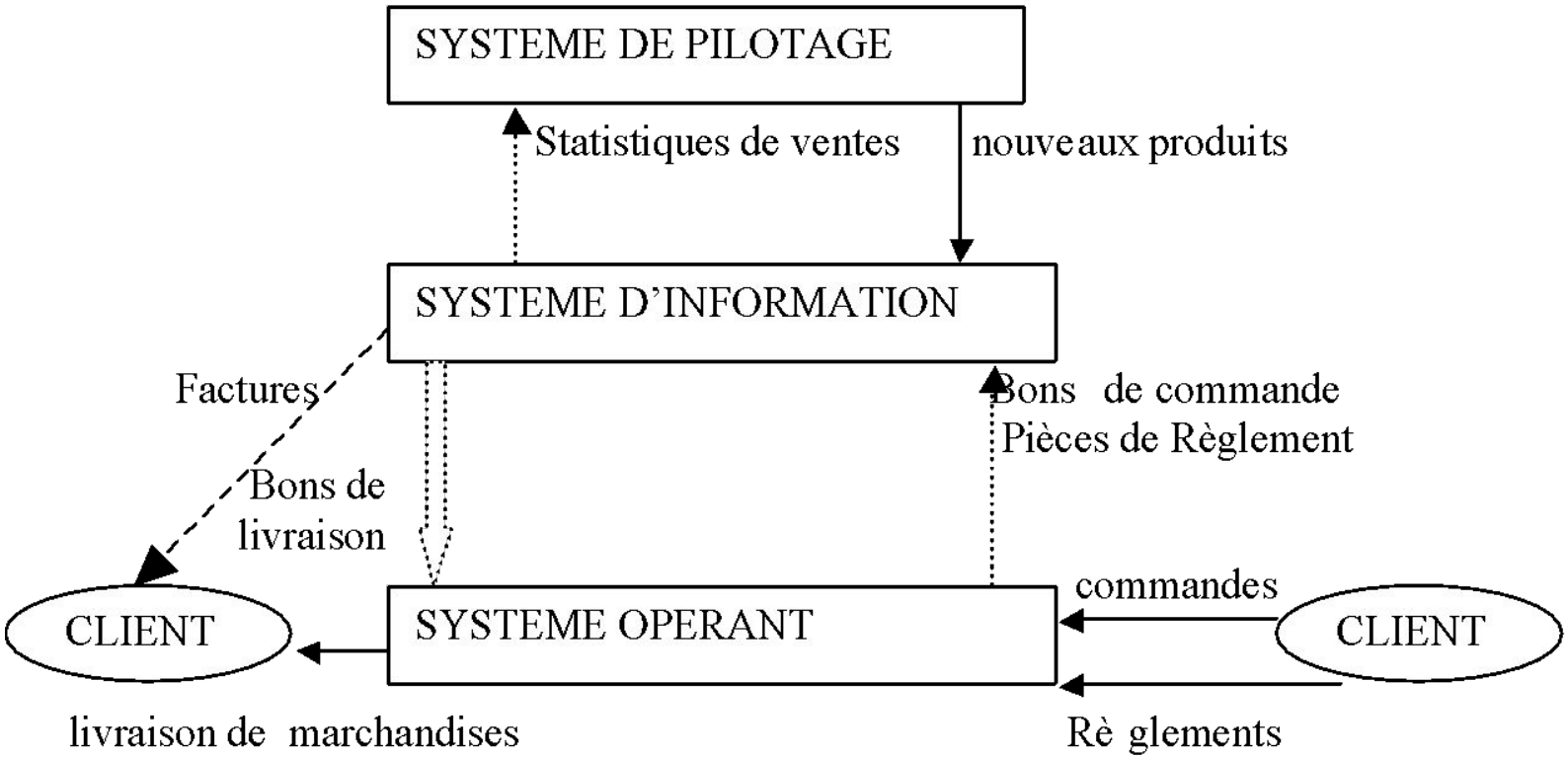


Illustration d'un SI

- **Flux physiques**: (bons de commande, bon de livraison, factures, pièces de règlement.)
- **Flux d'information**: Le numéro d'une commande, sa date, la référence d'un produit, sa désignation et son prix, un numéro de client, son nom , etc. Information plus synthétique; nombre de commandes par jour, top-10 des produits les plus vendus
- **Flux de décision**: Le prix d'un produit, le seuil d'approvisionnement du stock
- Le SI de cette entreprise est en liaison d'une part avec un environnement interne (système opérant et système de pilotage) et d'autre part avec un environnement externe (clients, fournisseurs, etc.)

Langages de modélisation

Définition

- Un modèle est une représentation simplifiée d'une réalité (système réel). Le modèle doit répondre aux questions que les utilisateurs se posent sur le système qu'il représente.
- Exemple de modèles: Une carte routière , un schéma, un plan etc.
- Un modèle sert à :
  - Communiquer : vérifier que l'analyse est bien comprise par les utilisateurs (phase analyse)
  - Préparer la réalisation du futur système grâce à un modèle de la solution (phase conception)
- Un langage de modélisation est un langage qui permet de décrire des modèles Exemple: UML; MERISE, etc.

Aperçu sur la méthode MERISE

Présentation générale de MERISE

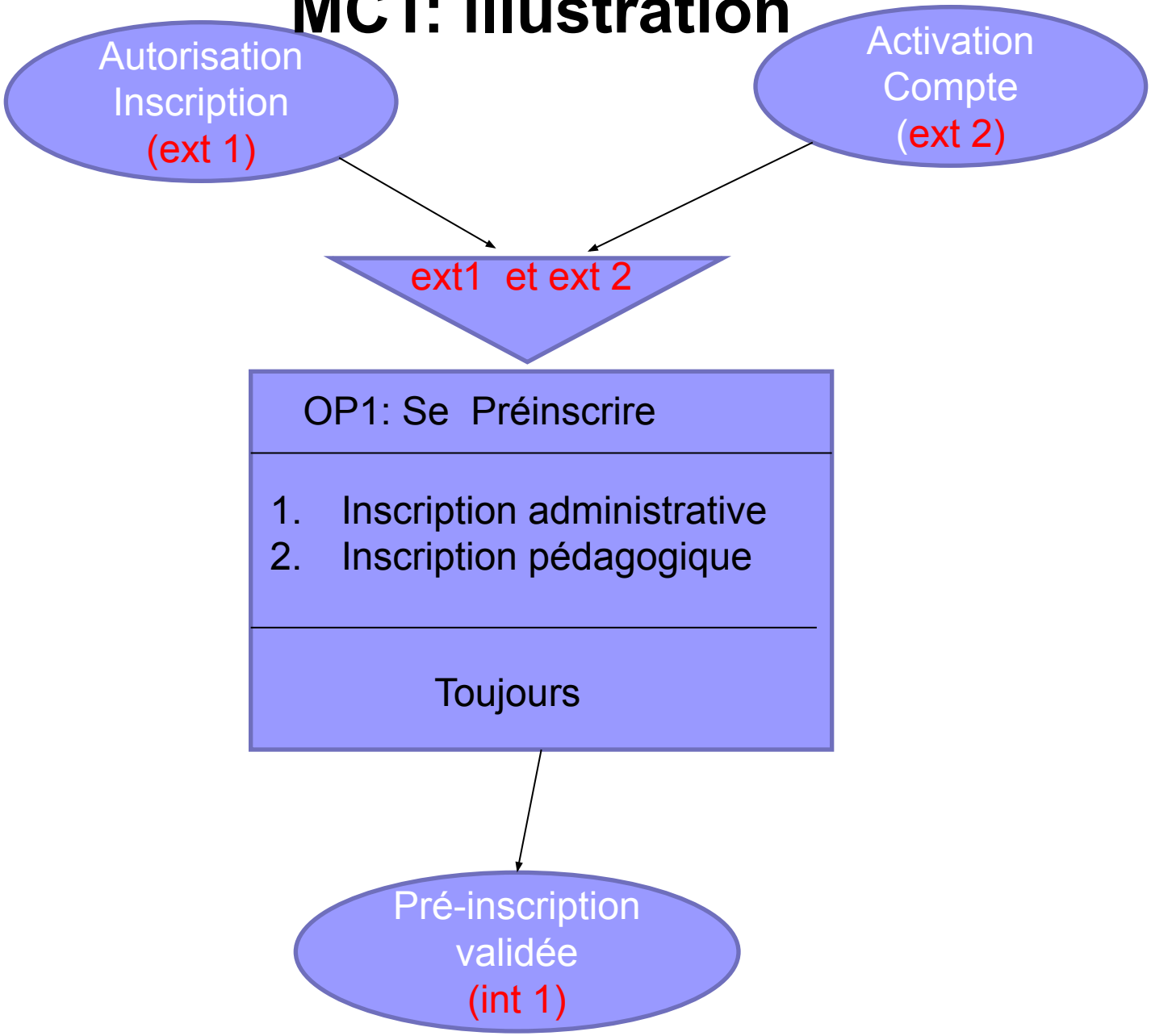
- **La méthode MERISE**
  - MERISE est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques.
  - La séparation des données et des traitements assure une longévité au modèle.
  - Elle date de 1978-1979, et fait suite à une consultation nationale lancée en 1977 par le ministère de l'Industrie dans le but de choisir des sociétés de conseil en informatique afin de définir une méthode de conception de systèmes d'information.
  - Version horticole: MERISE provient d'un arbre fruitier le merisier qui sert de porte greffes au cerisier: (La méthode MERISE permet de greffer l'informatique dans une organisation)

Niveaux de MERISE

NIVEAUX	DONNEES	TRAITEMENTS
CONCEPTUEL	**MCD** : sémantique des données (modèle entité/association)	**MCT** quoi ? (fonctions du SI)
ORGANISATIONNEL (ou LOGIQUE)	**MLD** : organisation des données	**MOT** qui fait quoi, ou, quand ?
PHYSIQUE	**MPD** implantation des données (SGF, SGBD)	**MPT** comment on fait ?

MCD : Modèle conceptuel des données
MLD : Modèle logique (organisationnel) des données
MPD : Modèle physique des données
MCT : Modèle conceptuel des traitements
MOT : Modèle organisationnel des traitements
MPT : Modèle physique des traitements

MCT: illustration



Langage UML

Historique

Des Méthodes de modélisation

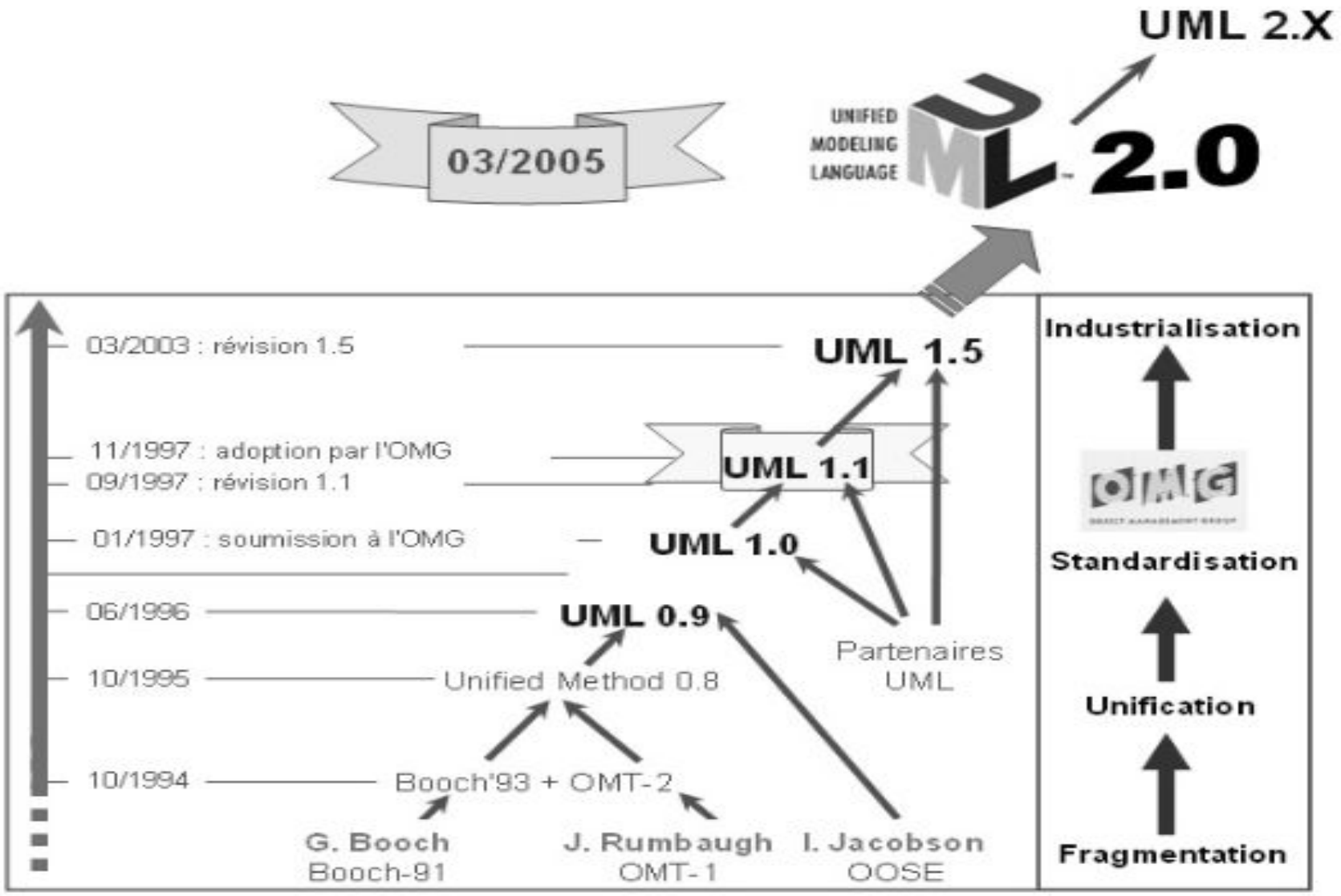
- L'apparition du paradigme objet à permis la naissance de plusieurs méthodes de modélisation
  - OMT, OOSE, Booch,
- Chacune de ces méthodes fournissait sa propre notation graphique et ses propres règles pour élaborer ses modèles
- Certaines méthodes sont outillées

Historique

Trop de Méthodes

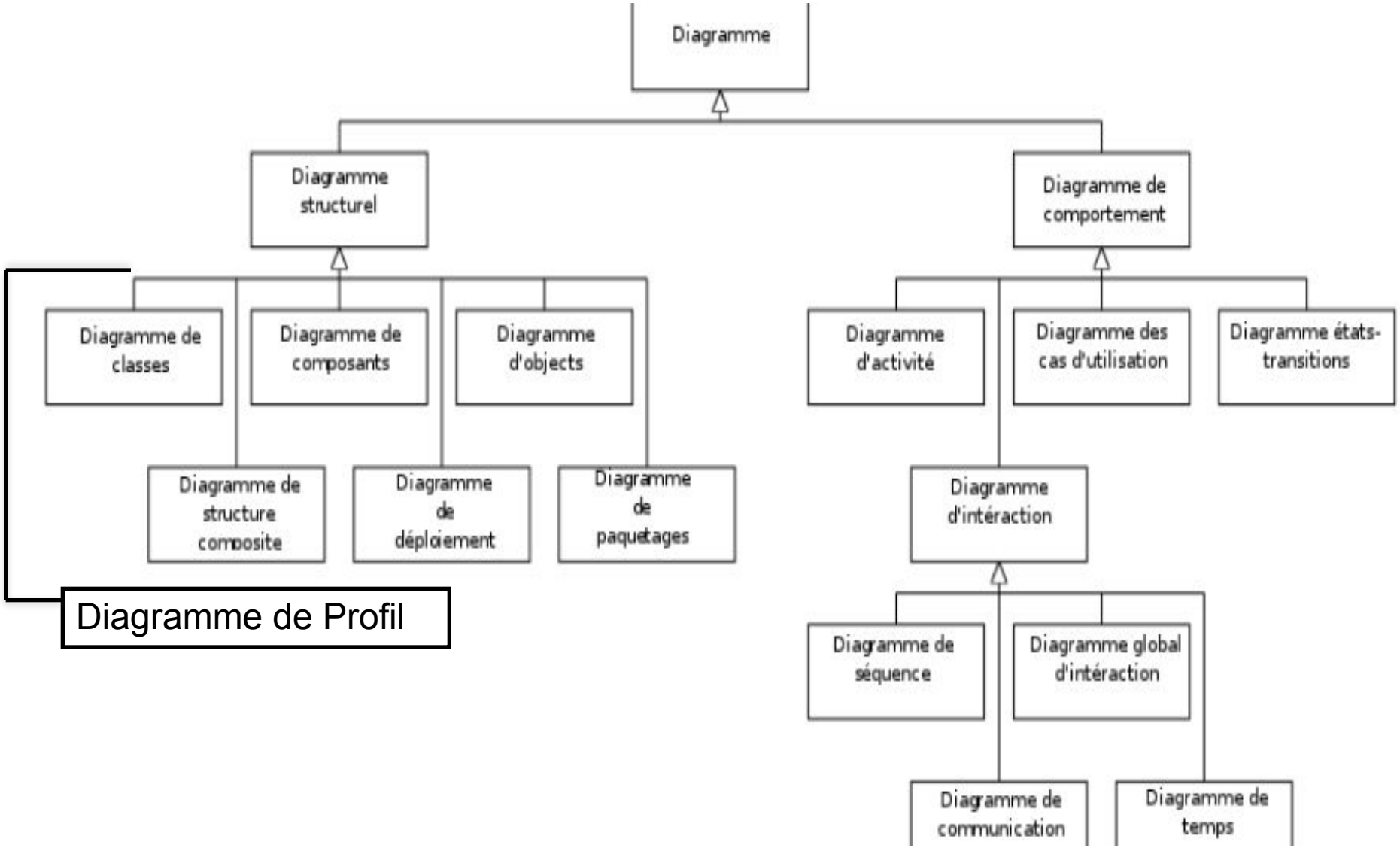
- Entre 89 et 94 : le nombre de méthodes orientées objet est passé de 10 à plus de 50
- Toutes les méthodes avaient pourtant d'énormes points communs (objets, méthode, paramètres, …)
- Au milieu des années 90, G. Booch, I. Jacobson et J. Rumbaugh ont chacun commencé à adopter les idées des autres. Les 3 auteurs ont souhaité créer un langage de modélisation unifié

Historique


UML 2.X
03/2005
UNIFIED MODELING LANGUAGE
UML 2.0
03/2003 : révision 1.5
UML 1.5
11/1997 : adoption par l'OMG
09/1997 : révision 1.1
UML 1.1
01/1997 : soumission à l'OMG
UML 1.0
06/1996
UML 0.9
10/1995
Unified Method 0.8
Partenaires UML
10/1994
Booch'93 + OMT-2
G. Booch
Booch-91
J. Rumbaugh
OMT-1
I. Jacobson
OOSE
Industrialisation
Standardisation
Unification
Fragmentation

- UML propose diverses notations pour décrire les différentes facettes d'un système matériel ou logiciel
- UML 2.3 s'articule autour de quatorze types de diagrammes, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel. Ces types de diagrammes sont répartis en trois grands groupes:
  - Sept diagrammes structurels ou statiques
  - Trois diagrammes comportementaux
  - Quatre diagrammes d'interactions ou dynamiques

Architecture fonctionnelle d'UML



Le Diagramme d’objets

Représentation des objets et leurs relations

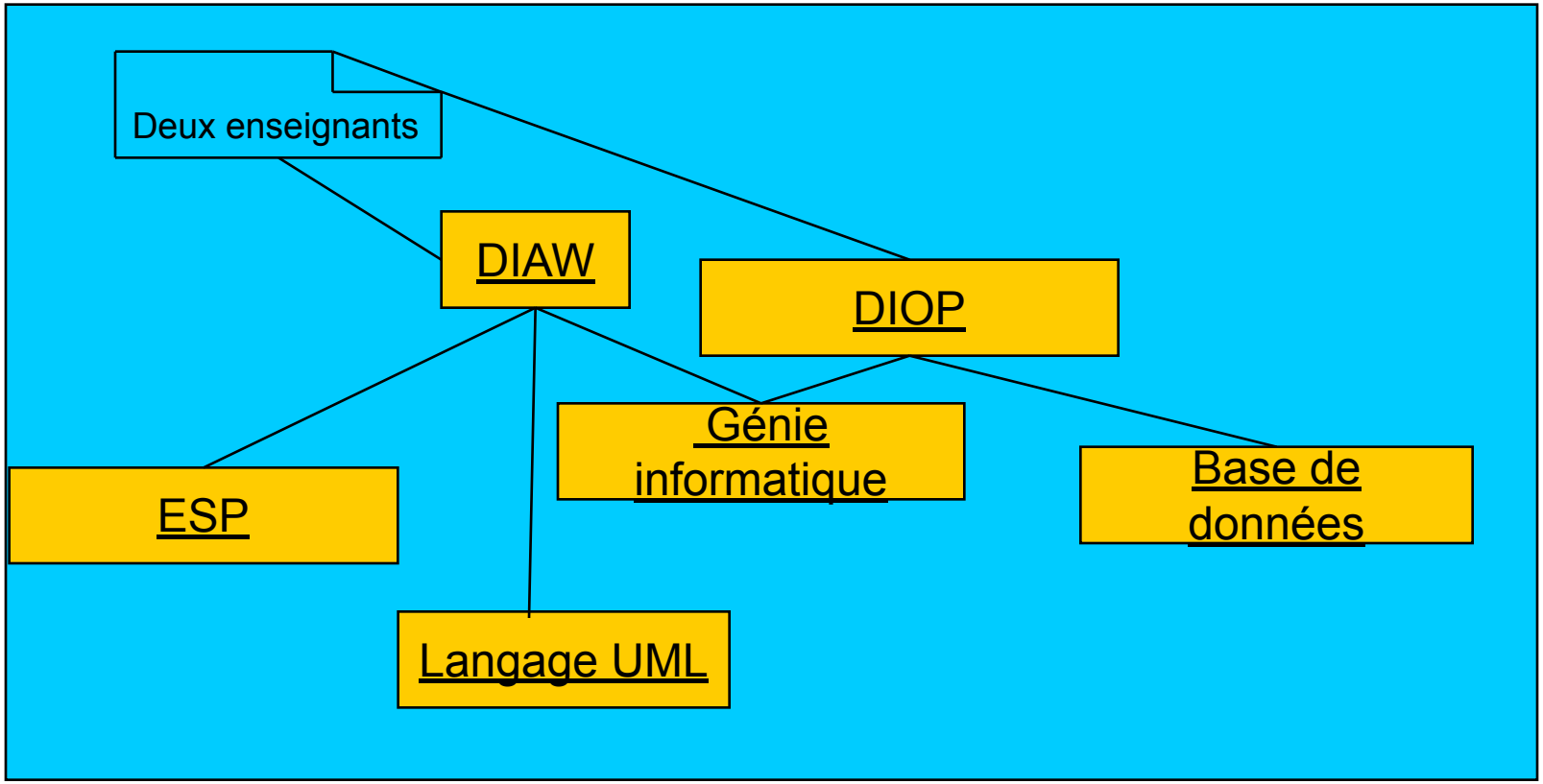
- **Définition**
  - Un diagramme d'objets représente des objets (instances de classes) et leurs liens (instances de relations) pour donner une vue de l'état du système à un instant donné afin :
    - d'illustrer le modèle de classes en montrant un exemple explicatif ;
    - de préciser certains aspects du système en mettant en évidence des détails imperceptibles dans le diagramme de classes ;

Objet

- Définition
  - Objet du monde réel :
    - Perception fondée sur le concept de masse
  - Objet informatique :
    - Est une unité atomique formée de l'union d'un état et d'un comportement
    - Définit une représentation abstraite d'une entité du monde réel ou virtuel, dans le but de la piloter ou de la simuler
      - Grain de sable, étoile
      - Compte en banque, police d'assurance, équation mathématiques, etc.
  - Les objets du monde réel et du monde informatique naissent, vivent et meurent

Objet

- Représentation d'objets en UML

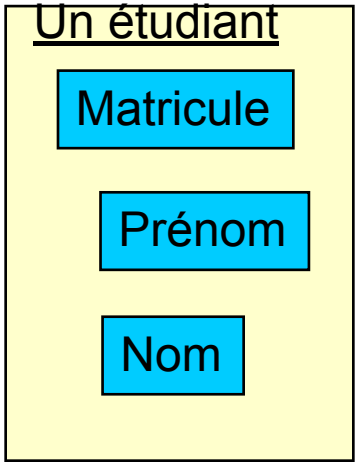


Objet

- Caractéristique fondamentales d'un objet (informatique)

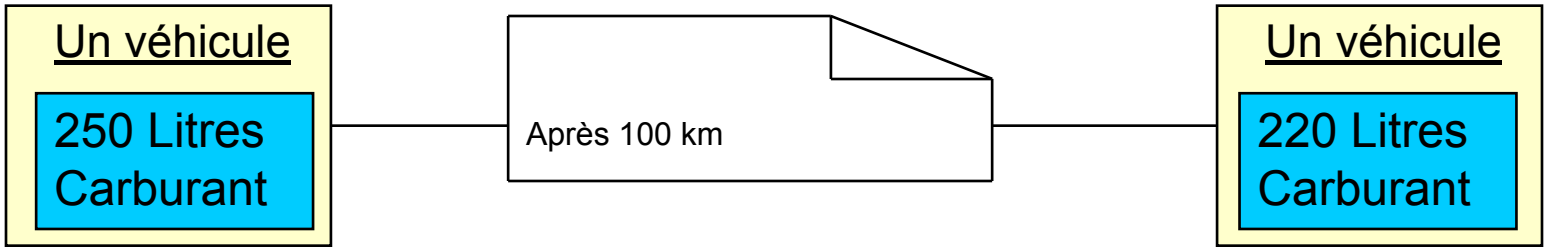
  Objet = Etat + Comportement + Identité

  - État
    - Regroupe les valeurs instantanées de tous les attributs d'un objet :
      - attribut est une information qui qualifie l'objet qui le contient
      - Chaque attribut peut prendre une valeur dans un domaine de définition donné



Exemple : Un objet Etudiant regroupe les valeurs des attributs matricule, Prénom et Nom

- # Objet
  - Caractéristique fondamentales d'un objet (informatique)
    - **État**
      - L'état d'un objet à un instant donné, correspond à une sélection de valeurs, parmi toutes les valeurs possibles des différent attributs
      - L'état évolue au cours du temps, il est la conséquence de ses comportement passés
        - Un véhicule roule, son niveau de carburant diminue



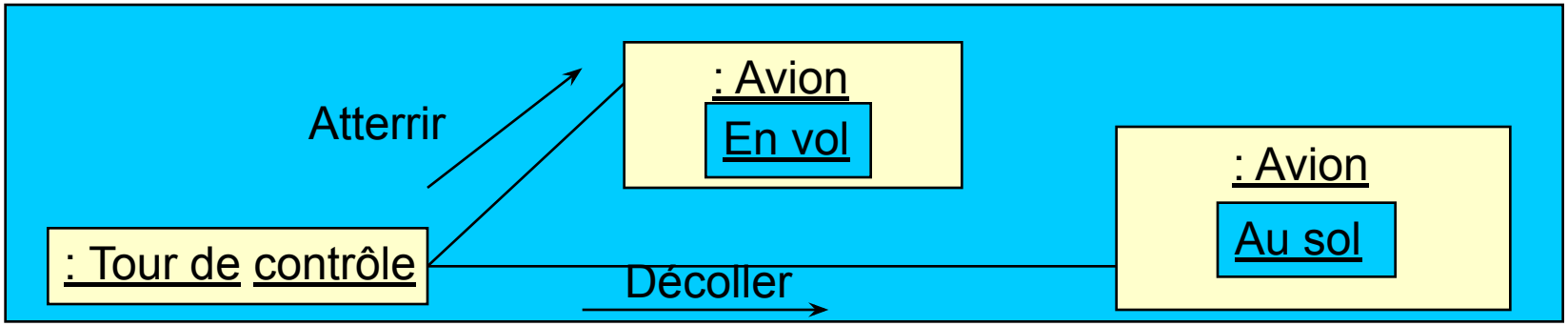
- Certaines composantes de l'état peuvent être constantes
  - La marque et le modèle d'un véhicule par exemple

Objet

- Caractéristique fondamentales d'un objet (informatique)
  - **Le comportement**
    - Regroupe toutes les compétences d'un objet et décrit les actions et les réactions de cet objet
    - Chaque atome (partie) de comportement est appelé opération
      - Les opérations d'un objet sont déclenchées suite à une stimulation externe, représentée sous la forme d'un message envoyé par un autre objet
      - L'état et le comportement sont liés

Objet

- Exemple
  - **Le comportement**
    - Le comportement à un instant donné dépend de l'état courant et l'état peut être modifié par le comportement
      - Il n'est pas possible de faire atterrir un avion que s'il est en train de voler: le comportement *Atterrir* n'est valide que si l'information *En vol* est valide
      - Après l'atterrissage, l'information *En vol* devient invalide et l'opération *Atterrir* n'a plus de sens



Objet

- Caractéristique fondamentales d'un objet
  - **L'identité**
    - Chaque objet possède une identité qui caractérise son existence propre
    - L'identité permet de distinguer tout objet de façon non ambiguë, indépendamment de son état
    - Permet de distinguer deux objets dont toutes les valeurs d'attributs sont identiques
      - deux pommes de la même couleur, du même poids et de la même taille sont deux objets distincts.
      - Deux véhicules de la même marque, de la même série et ayant exactement les mêmes options sont aussi deux objets distincts.

- Objet
- Caractéristique fondamentales d'un objet
  - **L'identité**
    - L'identité est un concept, elle ne se représente pas de manière spécifique en modélisation. : chaque objet possède une identité de manière implicite
    - En phase de réalisation, l'identité est souvent construite à partir d'un identifiant issu naturellement du domaine du problème.
    - Nos voitures possèdent toutes un numéro d'immatriculation, nos téléphones un numéro d'appel

Etude de Cas

Exemple: Gestion d'une bibliothèque

- Un internaute peut rechercher des ouvrages et peut visualiser les détails de la recherche ou faire un tri.
- Un visiteur est un internaute qui peut créer un compte
- Un abonné est un internaute qui peut faire des emprunts
- Pour chaque emprunt l'abonné devra s'authentifier.
- Au moment de l'emprunt, le système vérifie si l'abonné est autorisé à faire un emprunt ou non.
- L'abonné pourra à tout moment se désabonner

Le Diagramme de Cas d'utilisation

Représentation des fonctionnalités d'un système du point de vue utilisateur

Comment élaborer un Diagramme de cas d'utilisation ?

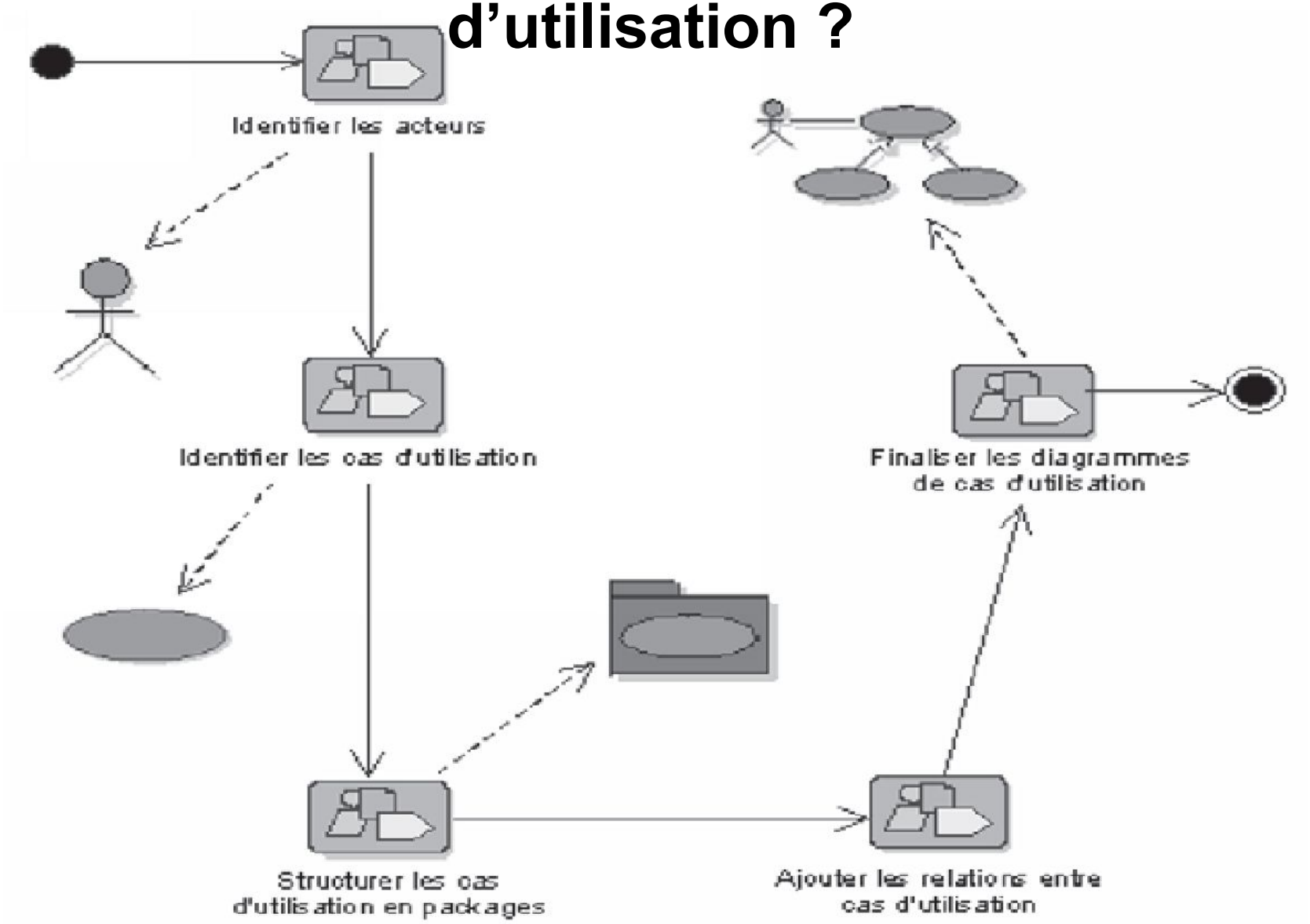
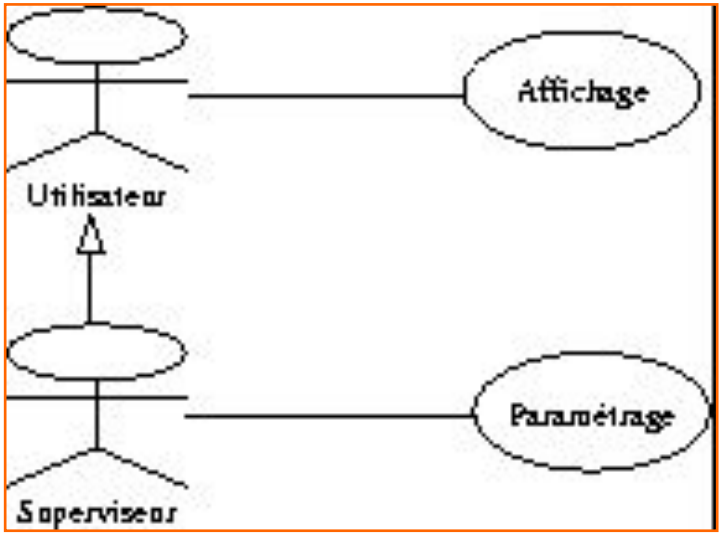


Diagramme de cas d'utilisation

- Le diagramme de cas d'utilisation est la première étape UML d'analyse d'un système.
- Moyen simple et facilement compréhensible pour exprimer les besoins des utilisateurs.
- Permet de recenser les grandes fonctionnalités d'un système.
- Ne pas négliger cette étape pour produire un logiciel conforme aux attentes des utilisateurs.
- L'élaboration de ce diagramme se fonde sur des entretiens avec les utilisateurs.

Acteurs

- Un acteur est l'abstraction d'un rôle joué par une ou plusieurs entités externes (personne, système, processus) qui interagissent avec un système
- Un acteur représente un utilisateur externe du système
- Un acteur est en relation avec un ou plusieurs cas d'utilisation
- Il est possible de définir des relations d'héritage entre Acteurs



Acteurs

- Un acteur est principal pour un cas d'utilisation lorsque ce cas rend service à cet acteur (association stéréotypée « primary »).
- Acteur secondaire
  - Les autres acteurs sont alors qualifiés de secondaires (association stéréotypée « secondary »).

Conseils

- Ne prenez pas comme acteur une entité qui n'interagit pas directement avec le système mais uniquement par le biais d'un véritable acteur
- Eliminez les acteurs physiques
  - Acteurs physiques : terminaux (écran, imprimantes, claviers, etc.)

Cas d'utilisation

- Un cas d'utilisation représente une fonctionnalité du système
- Une suite d'actions ininterruptibles et qui produit un résultat observable pour l'acteur
- Il est possible de définir des relations de dépendance entre cas d'utilisation
- Il est possible de définir des relations d'inclusion entre cas d'utilisation
- Il est possible de définir des relations d'héritage entre cas d'utilisation

Relation d'association

- Une relation d'association est chemin de communication entre un acteur et un cas d'utilisation et est représenté un trait continu.

Relations entre cas d'utilisation

- La relation d'inclusion .
- Un cas A inclut un cas B si le comportement décrit par A inclut le comportement de B : le cas A dépend de B.
- Représentation : flêche avec un trait pointillé stéréotypé « include »
- si A inclut B, flêche dirigée de A vers B.
- Lorsque A est sollicité, B l'est obligatoirement.
- La relation d'inclusion est utiliser pour factoriser deux cas qui ont une partie commune

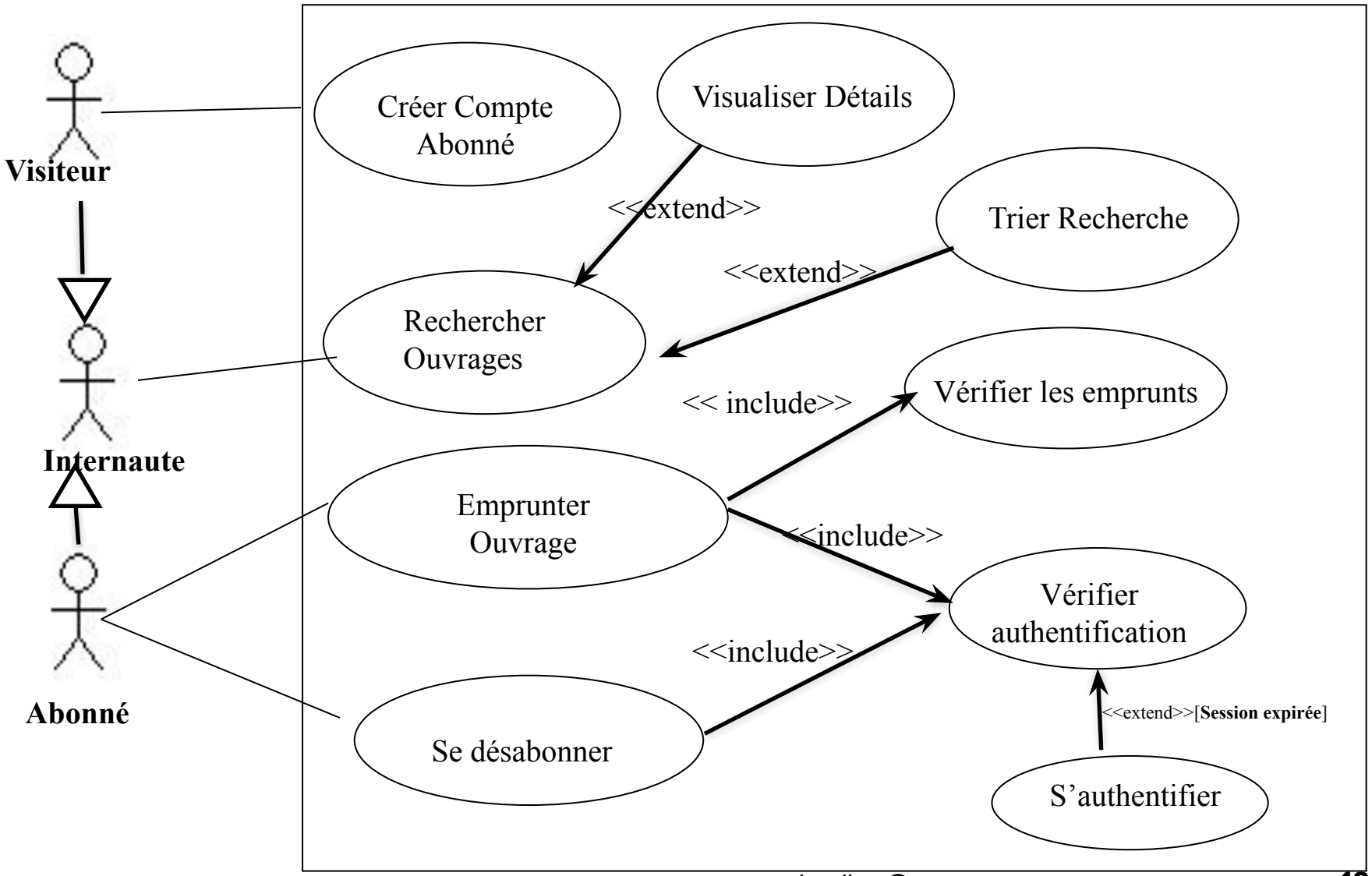
Relations entre cas d'utilisation

- **La relation d'extension**
  - On dit qu'un cas d'utilisation A étend un cas d'utilisation B lorsque A peut être appelé au cours de l'exécution de B.
  - Représentation : flêche avec un trait pointillé stéréotypée « extend »,
  - si A étend B, flêche dirigée de A vers B.
  - Exécuter B peut éventuellement entraîner l'exécution de A.
  - La relation la plus utile car elle a un sens du point de vue métier

Relation de généralisation

- Un cas A est une généralisation d'un cas B si B est un cas particulier de A.
- Représentation : flêche avec un trait pleins dont la pointe est un triangle fermé désignant le cas le plus général.
- Cette relation se traduit par le concept d'héritage dans les langages orientés objet.

Diagramme de cas d'utilisation du cas d'étude



Fiche textuelle d'un cas d'utilisation

Titre	Nom du cas
Description/Objectif	Descriptif/objectif du cas
ACTEURS	Les acteurs qui participent au cas
Préconditions	Conditions de déclenchement du cas
Scénario nominal	1. Action 1 2. Action 2 3. Action 3
Scénario alternatif	Les variantes du scénario nominal
Postconditions	Indicateur qui montre que le cas a été exécuté avec succès

Fiche textuelle du cas “s’authentifier”

Titre	S’authentifier
Description/objectifs	L'utilisateur a l’intention de s’authentifier sur le système avec son identifiant et son mot de passe
Acteur(s)	Abonné
Précondition	Créer Compte Abonné (cas d’utilisation)
Scénario nominal	1. L’abonné saisit son identifiant et son mot de passe 2. L’abonné valide les informations saisies 3. Le système vérifie les informations de connexion 4. L’utilisateur est redirigé vers son espace perso
Scénario alternatif	Après l’étape 3 si les informations de connexion sont inexactes alors : 4 b: Le système affiche une message d’erreur de connexion Le scénario reprend à l’étape 1
Post-condition	L’utilisateur est redirigé vers son espace perso

Le Diagramme de classes d'UML

Représentation de la structure statique en termes de classes et relations

Classe

Définition

- Une classe décrit une abstraction d'objets ayant
  - Des propriétés similaires
  - Un comportement commun
  - Des relations identiques avec les autres objets
  - Une sémantique commune
- Par exemple Etudiant (resp. Filière, resp. Cours) est la classe de tous les étudiants (resp. filières, resp. cours)

Classe

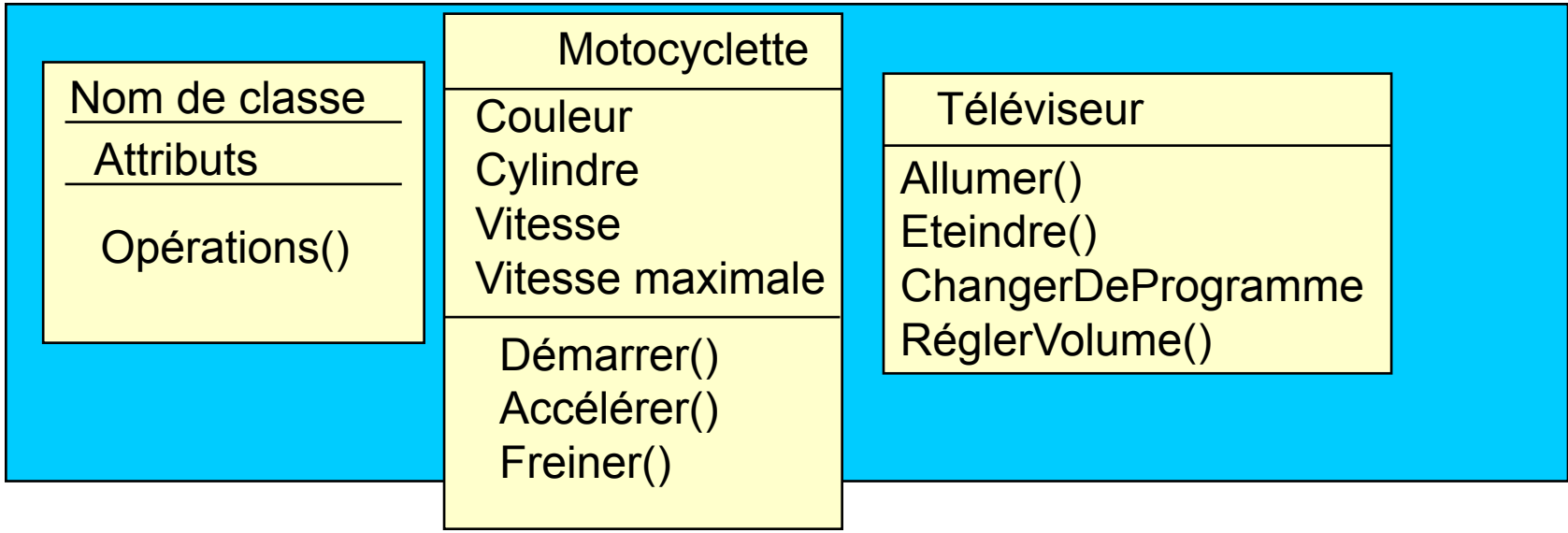
Caractéristiques d'une classe

- Un objet créé par (on dit également appartenant à) une classe sera appelé une *instance de cette classe* ce qui justifie le terme '*variables d'instances* '
- les valeurs des *variables d'instances* sont propres à chacune de ces instances et les caractérisent
- Les généralités sont contenues dans la classe et les particularité sont contenues dans les objets
- Les objets sont construits à partir de la classe, par un processus appelé instanciation : tout objet est une instance de classe
- Nous distinguons deux types de classes
  - Classe concrète : peut être instanciée
  - Classe abstraite : est une classe qui ne donne pas directement des objets et ne contient que des méthodes abstraites.

Classe

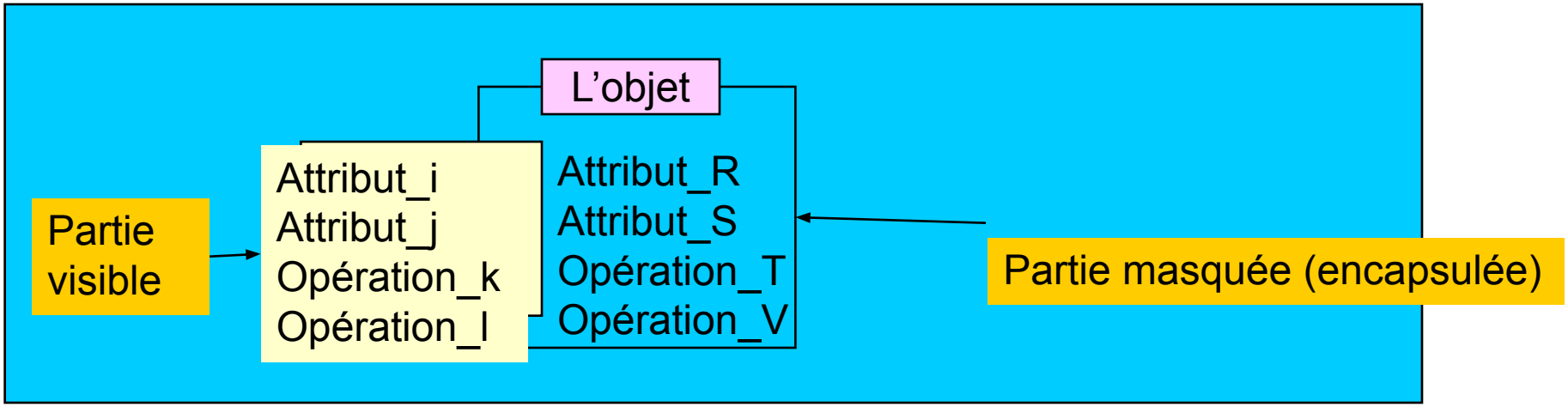
Représentation graphique d'une classe en UML

- Chaque classe est représentée sous la forme d'un rectangle divisé en trois compartiments
- Les compartiments peuvent être supprimés pour alléger les diagrammes



Encapsulation

- Consiste à masquer les détails d'implémentation d'un objet en définissant une interface
- L'interface est la vue externe d'un objet, elle définit les services accessibles (offerts) aux utilisateurs de l'objet.
- Est la séparation entre les propriétés externes, visibles des autres objets, et les aspects internes, propres aux choix d'implantation d'un objet.



Encapsulation

- Présente un double avantage
  - facilite l'évolution d'une application car elle stabilise l'utilisation des objets : on peut modifier l'implémentation des attributs d'un objet sans modifier son interface
  - garantit l'intégrité des données, car elle permet d'interdire ou de restreindre l'accès direct aux attributs des objets (utilisation d'accesseurs)

Règle de visibilité
+ Attribut public # Attribut protégé - Attribut privé
+ Opération publique() # Opération protégée() - Opértion privée()

Encapsulation

- Il est possible d'assouplir le degré d'encapsulation au profit de certaines classes utilisatrices bien particulière en définissant des niveaux de visibilité
- Les trois niveaux distincts d'encapsulation couramment retenus sont:
  - Niveau privé : c'est le niveau le plus fort; la partie privée de la classe est totalement opaque
  - Niveau protégé : c'est le niveau intermédiaire ; les attributs placés dans la partie protégée ne sont visibles que par les classes dérivées de la classe fournisseur. Pour toutes les autres classes, les attributs restent invisibles
  - Niveau publique : ceci revient à se passer de la notion d'encapsulation et de rendre visibles les attributs pour toutes les classes

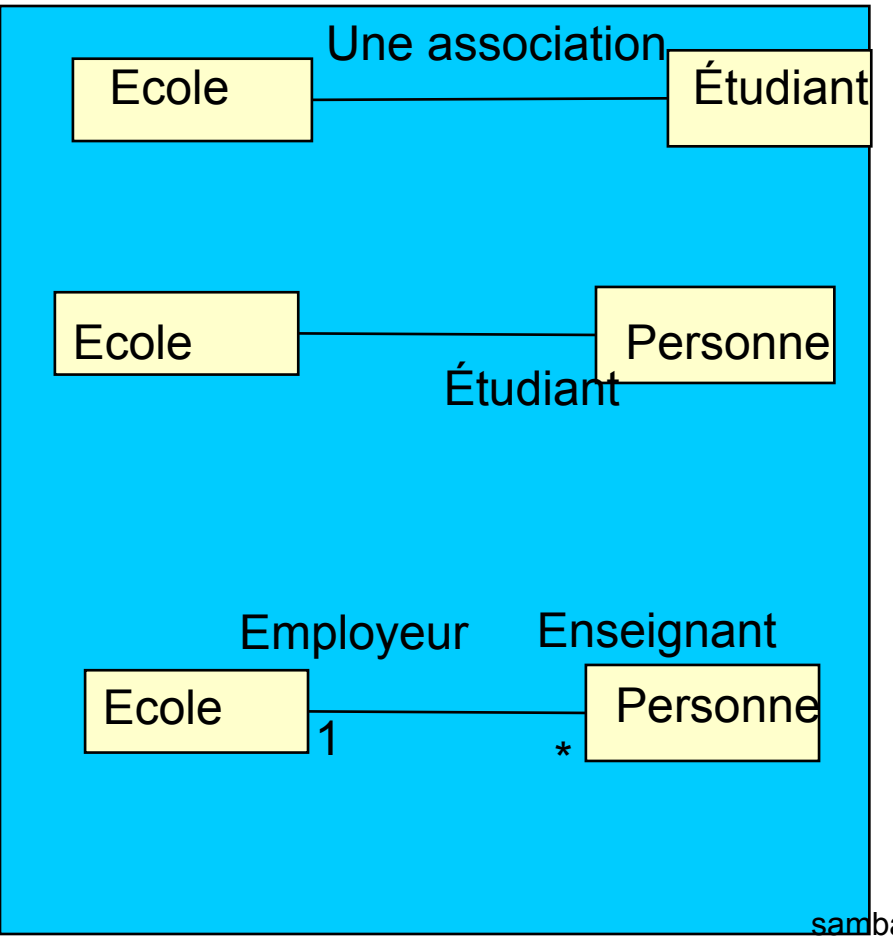
Salarié
+ nom # age - salaire
+ donnerSalaire() # changerSalaire() - calculerPrime()

Attributs

- Une classe peut contenir des attributs
- La syntaxe d'un attribut est :
  visibilité nom : type
- La visibilité est:
  - '+' pour public
  - '#' pour protected
  - '-' pour private
- UML définit son propre ensemble de types
  - Integer, real, string, …
- Un attribut peut être un attribut de classe, il est alors souligné.
- Un attribut peut être dérivé, il est alors préfixé par le caractère '/'

Les relations entre les classes

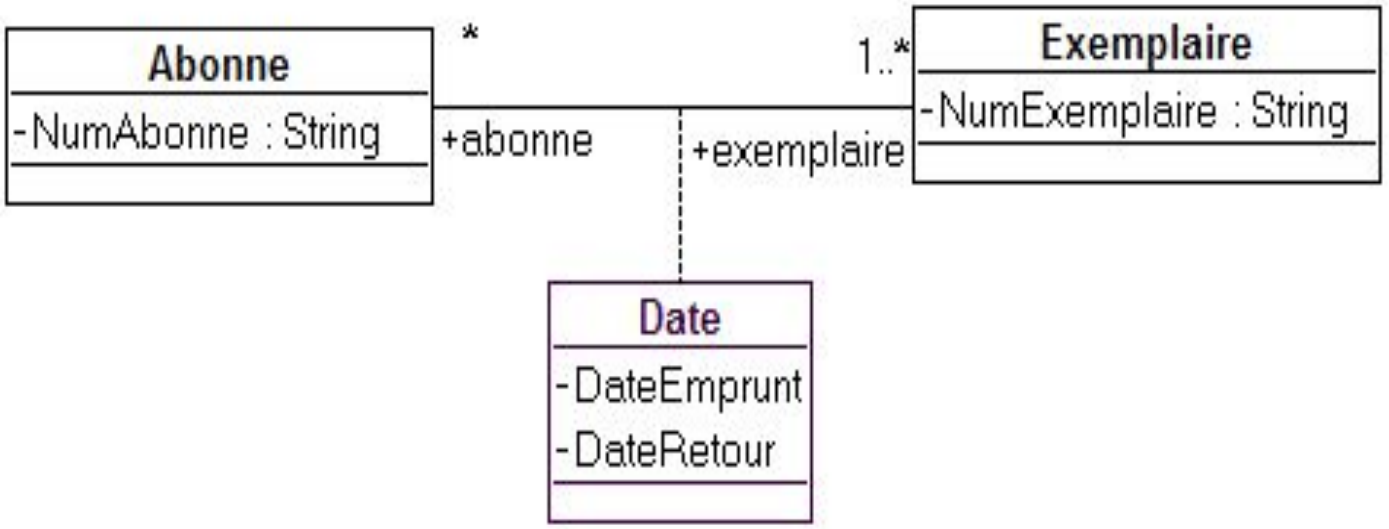
Associations



- Représentent les relations structurelles qui existent entre objets de différentes classes.
- Est une abstraction des liens qui existent entre les objets instances des classes associées
- La plupart des associations sont binaires
- Une association est caractérisée par:
  - un nom, qui peut être omis notamment quand les rôles des classes sont spécifiés,
  - un rôle pour chacune des classes qui participent à l'association. Le rôle d'une classe, écrit à l'extrémité du lien d'association, décrit la manière dont cette classe est ``vue'' par l'autre classe.
  - une multiplicité

Relations entre classes

- Classe d'association
  - Une classe-association est une association qui est aussi une classe.
  - Les classes-associations sont utilisées lorsque les associations doivent porter des informations



Les relations entre les classes

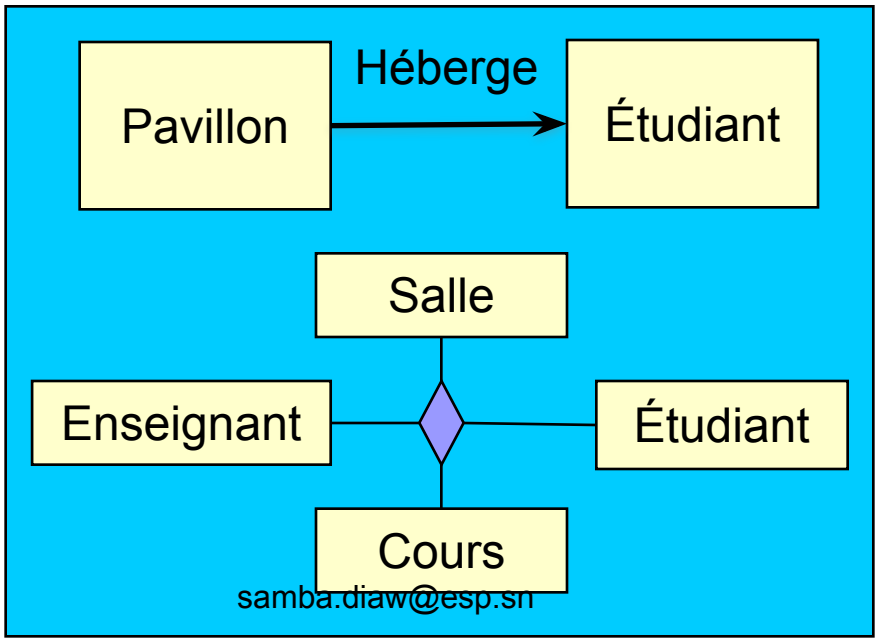
Multiplicité

- Chaque extrémité d'une association peut porter une indication de multiplicité (nombre d'occurrences) qui montre combien d'objets de la classe considérée peuvent être liés à un objet de l'autre classe.

1	Un et un
0..1	Zéro ou un
M..N	De M à N (entiers naturels)
*	De zéro à plusieurs
0..*	De zéro à plusieurs
1..*	D'un à plusieurs

Les relations entre les classes

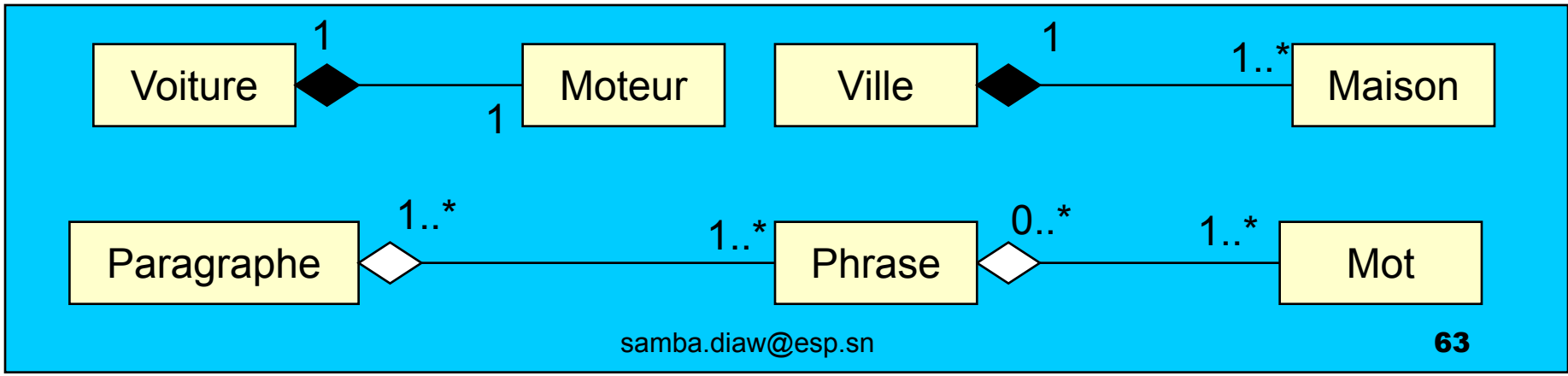
- Association n-aire et navigabilité
- Le nom de l'association peut être suivi d'une flèche de direction qui précise le sens de lecture (on dit que l'association est navigable)
- Il existe des associations ternaires (ou d'ordre supérieur) qui sont représentées par un diamant relié à chacune des classes participant à la relation.



Les relations entre les classes

La composition et l'agrégation (shared)

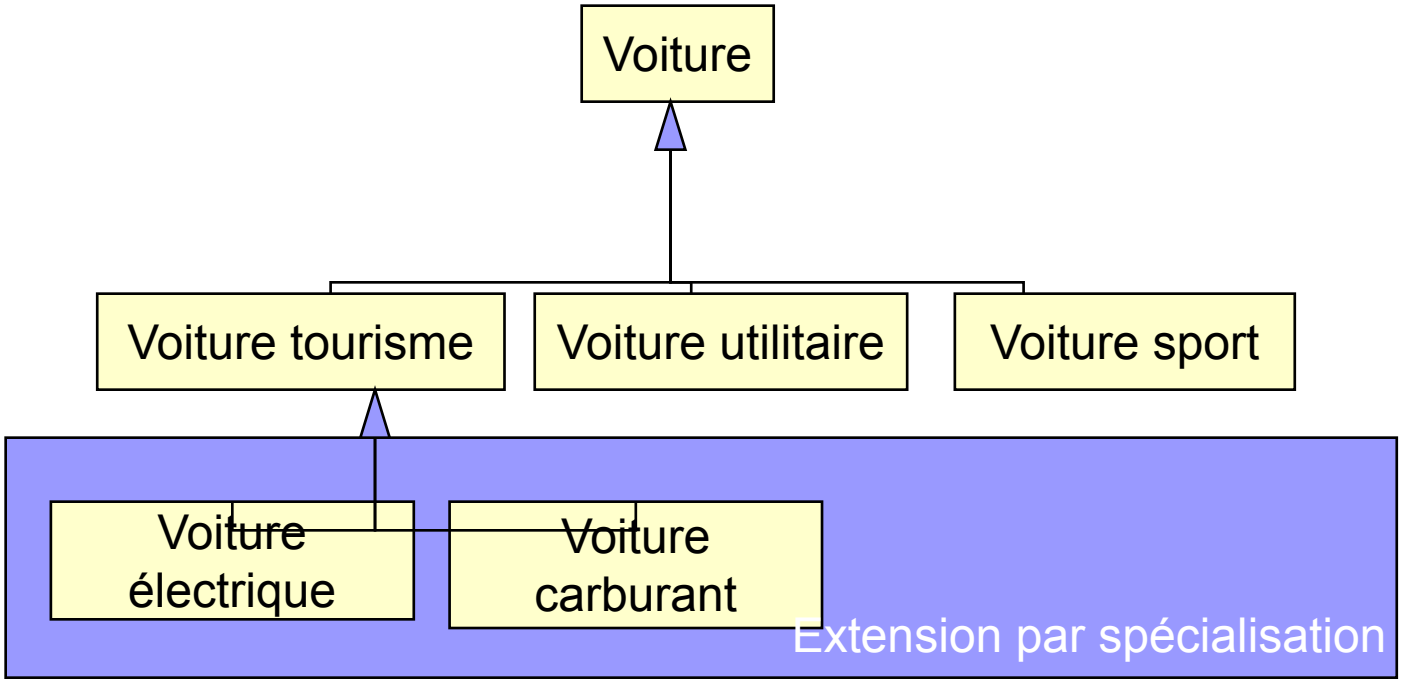
- UML propose deux sémantiques de contenance : une sémantique de contenance faible, dite d'agrégation, qui permet de préciser que les éléments contenus peuvent être partagés entre plusieurs conteneurs,
- et une sémantique de contenance forte, dite composition, qui permet de préciser que les éléments contenus ne peuvent être partagés entre plusieurs conteneurs.
- Du point de vue graphique, la relation de contenance se représente à l'aide d'un losange sur l'extrémité. Le losange est blanc pour l'agrégation et noir pour la composition.



Les hiérarchies de classes

Généralisation et spécialisation

- La spécialisation permet de capturer les particularités d'un ensemble d'objets non discriminés par les classes déjà identifiées
  - Les nouvelles caractéristiques sont représentées par une nouvelle classe, sous-classe d'une des classes existantes



Les hiérarchies de classes

Généralisation et spécialisation

- La généralisation et la spécialisation sont deux point de vue antagonistes du concept de classification; elle expriment dans quel sens une hiérarchie de classe est exploitée
- La généralisation ne porte aucun nom particulier ; elle signifie toujours : est un ou est une sorte de
- La généralisation ne concerne que les classes, elle n'est pas instanciable en liens
- La généralisation ne porte aucune indication de multiplicité

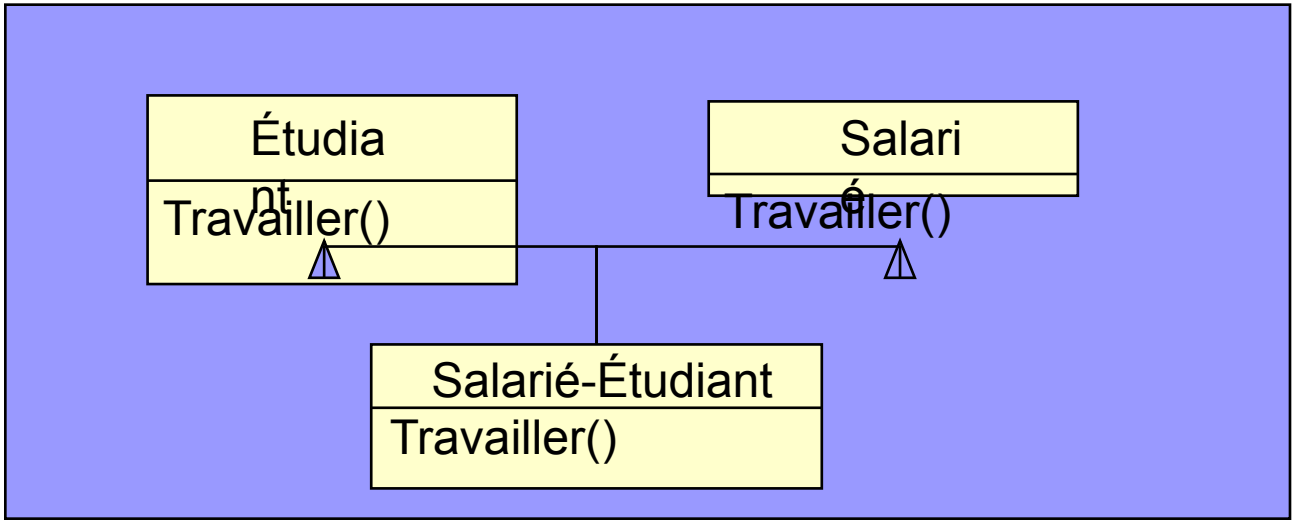
Les hiérarchies de classes

Généralisation et spécialisation

- La généralisation est une relation non réflexive : une classe ne peut pas dériver d'elle-même
- La généralisation est une relation non symétrique : si une classe B dérive d'une classe A, alors la classe A ne peut pas dériver de la classe B
- La généralisation est une relation transitive : si une classe C dérive d'une classe B qui dérive elle-même d'une classe A, alors C dérive également de A
- Les objets instances d'une classe donnée sont décrit par la propriétés caractéristiques de leur classe, mais également par les propriétés caractéristiques de toutes les classes parents de leur classe

Les hiérarchies de classes

- Généralisation multiple :
- Elle existe entre arbres de classes disjoints
  - Une classe ne peut posséder qu'une fois une propriété donnée



Exemple de diagramme de Classes

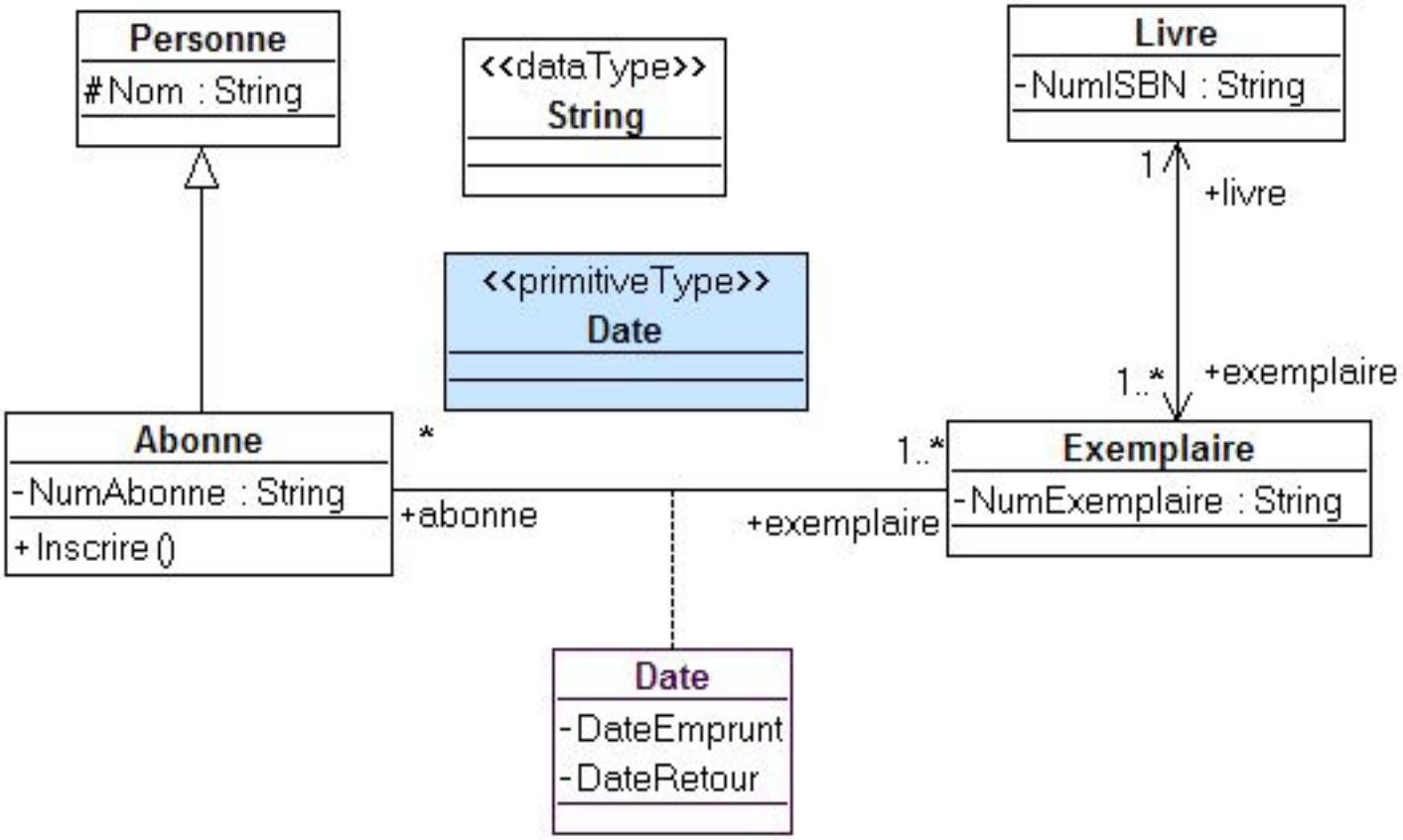


Diagramme de Séquence

- Un diagramme de séquence représente une interaction entre plusieurs éléments
- Les éléments interagissent par envoi de messages
- Les éléments interagissant sont des instances jouant des rôles.

Instances

- Un diagramme de séquence met en œuvre des instances
  - Instance de classe, Instance d'acteur
- Graphiquement une instance se distingue de son type car elle est soulignée
- Il est possible de définir des instances sans préciser leur classe
- La durée de vie des instances est définie sur l'axe vertical du diagramme
- Graphiquement l'activité d'une instance se voit grâce à un rectangle sur l'axe du temp

DIAGRAMME DE SEQUENCE

Activation d'un objet

- Il est possible de représenter de manière explicite les différentes périodes d'activité d'un objet
  - au moyen d'une bande rectangulaire superposée à la ligne de vie de l'objet
- On peut aussi représenter des messages récursifs
  - en dédoublant la bande d'activation de l'objet concerné
- Pour représenter de manière graphique une exécution conditionnelle d'un message,
  - on peut utiliser les opérateurs d'interaction combinés

DIAGRAMME DE SEQUENCE

- Remarques :
  - Ne confondez pas la période d'activation d'un objet avec sa création ou sa destruction.
    - Un objet peut être actif plusieurs fois au cours de son existence
  - Le retour des messages asynchrones devrait toujours être matérialisé, lorsqu'il existe
  - Préférez l'utilisation de contraintes à celle de pseudo-code
  - Un message réflexif ne représente pas l'envoi d'un message, il représente une activité interne à l'objet (qui peut être détaillée dans un diagramme d'activités) ou une abstraction d'une autre interaction (qu'on peut détailler dans un autre diagramme de séquence).

- Un message définit une communication particulière entre des lignes de vie.
- Type de message
  - Envoi d'un signal
    - Un signal (ex : interruption, événement) est, par définition, un message asynchrone.
    - Il n'attend pas de réponse.
    - Il ne bloque pas l' émetteur qui ne sait pas si le message arrivera `a destination.
  - Invocation d'une opération (flêche pleine avec retour explicite)
    - L'invocation d'une opération est le type de message le plus utilisé en programmation objet.
    - L'invocation peut être asynchrone ou synchrone.
    - Dans la pratique, la plupart des invocations sont synchrones : l' émetteur reste alors bloqué le temps que dure l'invocation de l'opération.

- message synchrone
  - Bloque l'expéditeur jusqu'à prise en compte du message par le destinataire.
- message asynchrone
  - N'interrompt pas l'exécution de l'expéditeur.
  - Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré (jamais traité).
- message minuté (timeout) :
  - Bloque l'expéditeur pendant un temps donné (qui peut être spécifié dans une contrainte), en attendant la prise en compte du message par le récepteur. L'expéditeur est libéré si la prise en compte n'a pas eu lieu pendant le délai spécifié
- Message perdu ou trouvé
  - Une flêche qui pointe sur une petite boule noire représente un message perdu.
  - Une flêche partant d'une petite boule noire représente un message trouvé

Fragments d'interaction combinés

- Permet de décomposer une interaction complexe.
- Défini par un opérateur et des opérandes.
- L'opérateur conditionne la signification du fragment combiné. Il existe 12 opérateurs définis dans la notation UML 2.0 :
- **choix et boucle** : alternative, option, break et loop ;
- **concurrence** : parallel et critical region ;
- **envoi de messages** : ignore, consider, assertion et negative ;
- **ordre d'envoi des messages** : weak sequencing , strict sequencing.
- Se représente de la même façon qu'une interaction.
- Ses opérandes sont séparés par une ligne pointillée.

Diagramme de séquences

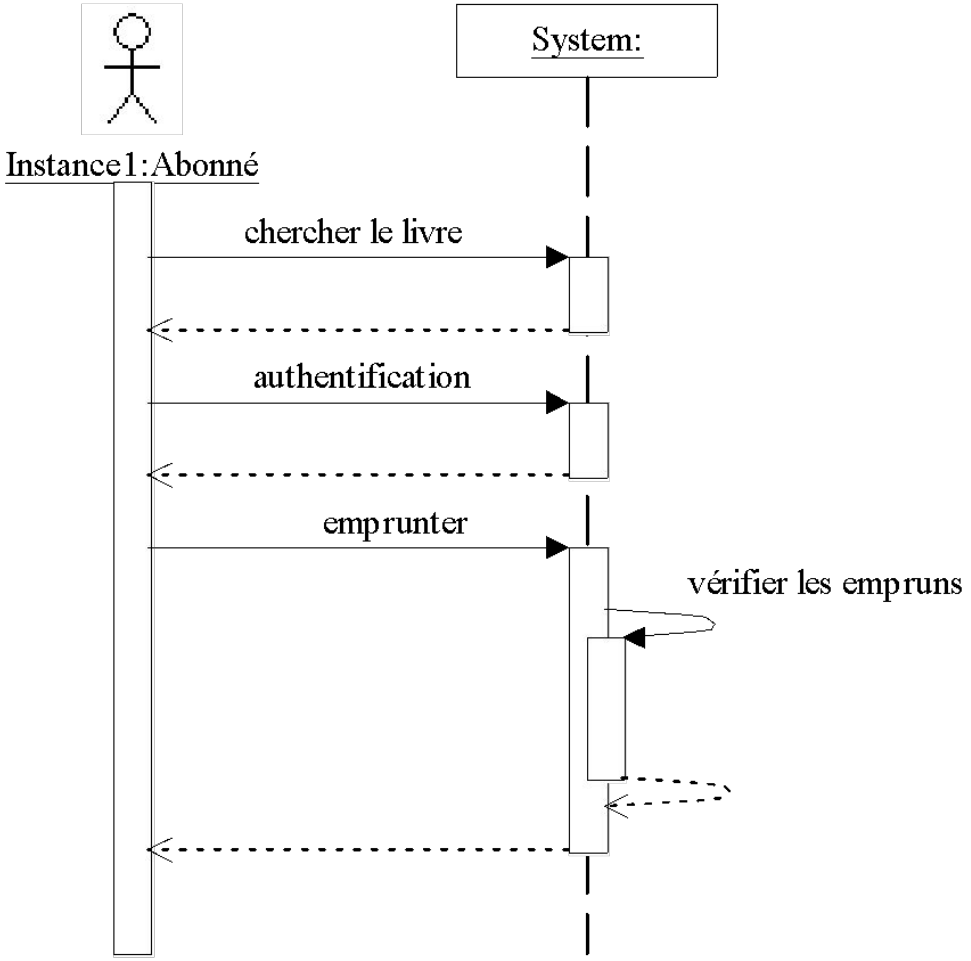


Diagramme de Séquence - Fin

- Les diagrammes de séquence sont de plus en plus utilisé
  - Ils permettent de décrire la dynamique d'un système
  - Ils permettent de faire le lien entre les diagrammes de cas d'utilisation et les diagrammes de classes
- La sémantique de ces diagrammes est encore un peu flou
  - Les techniques de génération de code n'exploitent pas encore très pleinement ces diagrammes

FIN DU COURS