

# SLAM345

Éléments pour UML

# Progression

1. Introduction
2. Préliminaires
3. Les règles d'UML
4. Les diagrammes UML
  1. Diagrammes de classes (DCL)
  2. Diagrammes d'objets (DOB)
  3. Diagrammes des cas d'utilisation (USE CASE)->(DCU)
  4. Diagrammes d'état transition (DET)
  5. Diagrammes d'activité (DAC)
  6. Diagrammes de séquence (DES)
  7. Diagrammes de collaboration (DCO)
  8. Diagrammes de composants (DCP)
  9. Diagrammes de déploiement (DPO)
5. Les outils de modélisation UML
6. L'étude préalable avec UML (cas d'inscription)
7. Conclusion

# Attention

- \* Ce support ne fait pas le lien avec un langage de programmation.
- \* Cela viendra plus tard
- \* Vous ne trouverez pas de transformation du DCL en langage de programmation Java, C++, ou autre.

# 1. Introduction

- \* UML : langage de modélisation
- \* Comme MERISE, UML peut établir des méta-modèles, pour définir la structure des modèles lui-même
- \* Permet de construire des modèles objets ou autres
- \* Utilise un formalisme graphique comme MERISE
- \* Les solutions sont donc visuelles
- \* Attention comme avec MERISE la sémantique joue un rôle important
- \* UML n'est pas sensible aux langages d'implémentation

## 2. Préliminaires

- \* Les origines d'UML : histoire
- \* La démarche de conception et d'analyse
- \* UP : Unified Processus dans l'histoire d'UML
  - \* Lien avec les autres méthodes (AM, RUP, ...)

# Origines d'UML

- \* Mis en place en 1995, 1996
- \* Issu de la pratique industrielle et de la modélisation des systèmes
- \* C'est une méthode anglo-saxonne
- \* Il provient de l'unification des méthodes
  - \* Ivar Jacobson (OOSE) -> USE CASE
  - \* Grady Booch (BOOCH)
  - \* James Rumbaugh (OMT)
- \* Initialement il fallait être compétent dans l'une, l'autre, ou toutes les méthodes pour balayer une approche similaire à UML

# Origines d'UML

- \* Normalisé en 1997
- \* Version 2 depuis 2007
- \* Méthodes fonctionnelles : années 1960
  - \* Structuration modulaire, paradigme sous-programme
  - \* Séparation des données et du code
- \* Méthodes objets : années 1980
  - \* Apparition des interfaces
  - \* Modélisation événementielles, concept d'objets
  - \* Apparition des langages objets
  - \* Lien entre les données et les méthodes
- \* UML apporte une nouvelle technique sans rejeter les autres méthodes

# Démarche de conception et d'analyse

- \* Analyse du problème : Unified Processus
  - \* Guidée par les besoins des utilisateurs du système
  - \* Centrée sur l'architecture logicielle
  - \* Itérative et incrémentale
- \* Utilisation d'un langage de modélisation UML
  - \* permet d'améliorer progressivement les méthodes de travail
  - \* préserve les modes de fonctionnement,
  - \* boîte à outil (tendance du moment) -> objectif des méthodes AM
- \* L'itération peut se faire à toutes les phases
  - \* Étude préalable
  - \* Construction
  - \* Tests et mise au point



# Processus unifié

- \* UML + UP = méthode globale
- \* PU complète la systématique UML
- \* C'est une méthode générique, itérative et incrémentale, contrairement à la méthode séquentielle MERISE ou SADT
- \* UP : Processus de développement proposé par J-B-R
- \* Processus :
  - \* Recensement des cas d'utilisation
  - \* Construction de l'architecture du système dès le début
  - \* Avec principes d'itération et incrémentations
  - \* Et évaluation des risques à toutes les étapes

# Processus unifié

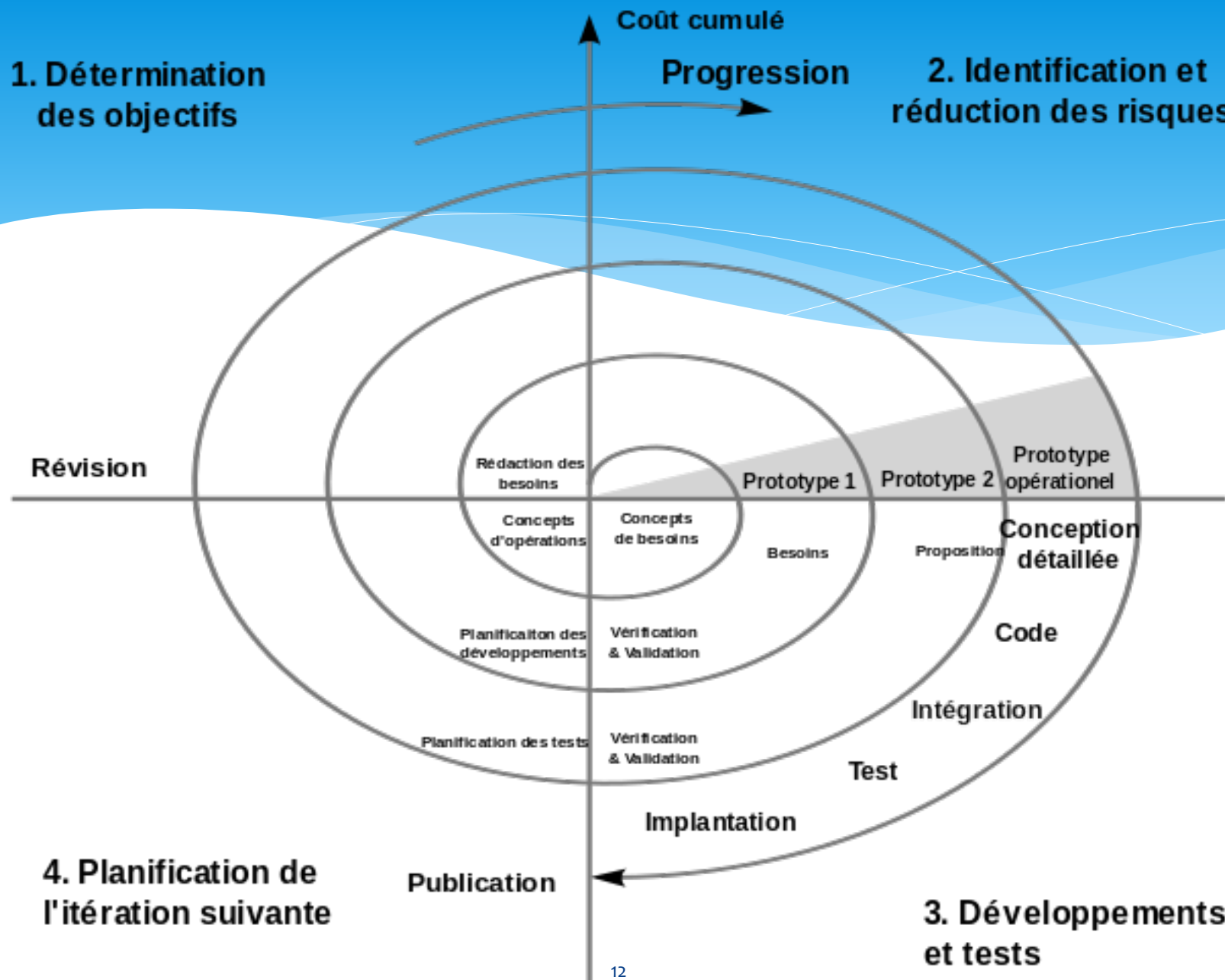
- \* Les utilisateurs décrivent les « use case »
  - \* Recensement des besoins
    - \* Description des composants ou objets
    - \* Description des modes opératoires
  - \* Recensement des contraintes
  - \* Interactions entre les besoins et les contraintes
  - \* Construction d'une architecture du système en adéquation
- \* Progression de la construction
  - \* En affinant l'étude, par ajouts, détails des « use case »
  - \* Description plus détaillée des composants
  - \* Ajustement de l'architecture
- \* Utilisation de maquettes et prototypes

# Processus unifié

- \* Evaluation permanente du système en terme de bon choix :
  - \* Le bon produit
  - \* Une bonne construction
  - \* Un bon prix
  - \* Les bonnes performances
  - \* Evolution
  - \* Amélioration
- \* Validation ou rejet des solutions
- \* Objectif :
  - \* minimiser les risques au fur et à mesure de la spirale de développement

**1. Détermination  
des objectifs**

**2. Identification et  
réduction des risques**



# Processus unifié

- \* Phases du processus UP :
  - \* Etude d'opportunité :
    - \* Mesures des risques
    - \* Définitions des limites
    - \* Construction d'une maquette des premiers use case
  - \* Réalisation :
    - \* Première version
    - \* Proposition d'architecture, développement, test
    - \* Amélioration incrémentale et itérative -> produit final
  - \* Mise en exploitation

# Processus unifié

- \* Les activités dans les phases UP :
  - \* Expression des besoins
  - \* Analyse
  - \* Conception
  - \* Implémentation
  - \* Tests
- \* RUP : Rational Unified Process, célèbre implémentation de PU pour le développement logiciel (Version UP de Rational Software)

# Processus unifié

- \* Synthèse :
  - \* PU est à base de composants
  - \* PU utilise UML
  - \* PU est piloté par les cas d'utilisation
  - \* PU est centré sur l'architecture
  - \* PU est itératif et incrémental
- \* Les méthodes agiles AM sont justes des améliorations (best practices) de PU
- \* Scrum est une AM, qui utilise dans son concept UML

# 3. Les règles d'UML

- \* Développement orienté objet
- \* UML langage de modélisation
  - \* Règle d'écriture de représentation normalisées
  - \* 9 diagrammes initiaux -> 13 en version 2.1.2 (2007)
  - \* Encore plus avec les versions récentes
- \* Méta-modèle des concepts et notation des diagrammes
  - \* Construire les outils de modélisation selon les règles UML
  - \* Règles :
    - \* Stéréotypes
    - \* Notes
    - \* Contraintes
    - \* Règles d'écriture des noms et expressions : nom, étiquette, valeur composant
    - \* Paquetage



# Règles d'UML

- \* Les stéréotypes
  - \* Adaptation du modèle UML aux éléments de l'application
  - \* Permet d'étendre UML afin de créer de nouveaux modèles
  - \* Nouveau type d'élément défini depuis un type du modèle
  - \* Application principale aux classes
  - \* Distinction d'utilisation entre guillemets
  - \* Classe client stéréotypée : « client A »
- \* Notes
  - \* Commentaires d'un élément UML



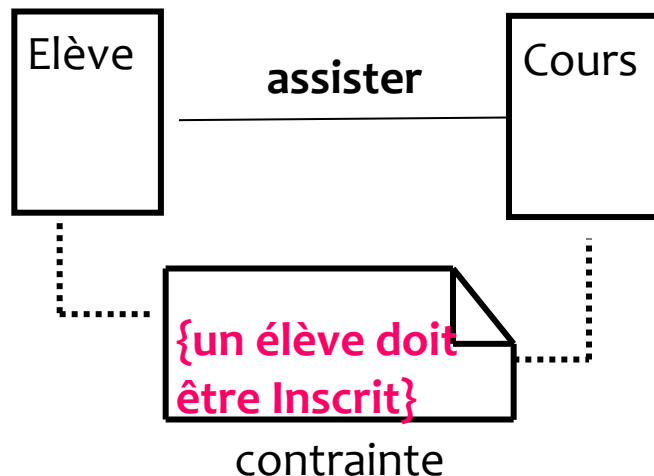
stéréotype



commentaire

# Règles d'UML

- \* Contraintes :
  - \* Note sémantique pour un élément
  - \* Ecriture en { }
  - \* Fait partie du langage OCL (Object Constraint Language)
- \* Ecriture des noms et des expressions
  - \* Nom : identifiant d'un élément, chaîne de caractères
  - \* Expression : valeur



NomEleve  
Cycle.UE

expressions

After (7 minutes)

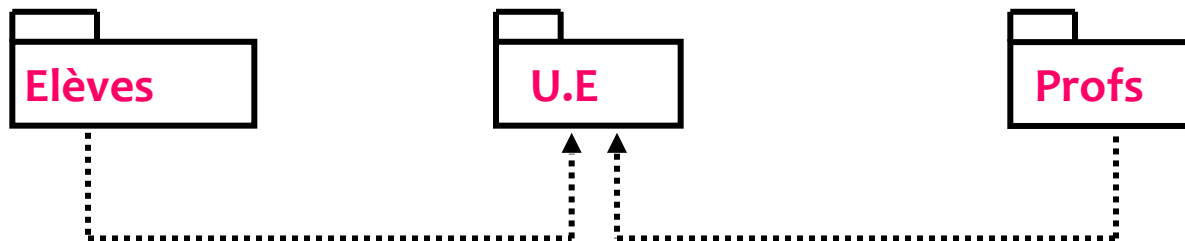
Date = 7 juillet 2005

# Règles d'UML

- \* OCL UML :
  - \* Contribution d'IBM
  - \* Limite les ambiguïtés dans les spécifications
- \* OCL décrit :
  - \* pré et post-conditions pour une opération
  - \* expressions de navigation
  - \* expressions booléennes
- \* Aucun âge de personne ne peut être négatif :
  - \* context Personne inv: self.age >=0

# Règles d'UML

- \* **Paquetage :**
  - \* Décomposition du système en paquetages (groupe d'éléments)
  - \* Ensemble logique d'éléments du modèle (pour faire des groupes cohérents)
  - \* Nommage du paquetage
  - \* Relations entre paquetage



# 4. Les principaux diagrammes UML

- \* Diagramme des cas d'utilisation (use-cases ou Use Case Diagram)
  - \* Besoins des utilisateurs
- \* Diagramme de classes (Class diagram)
  - \* Description statique des données et des traitements
- \* Diagrammes d'objets (Object diagram)
  - \* Instances des classes
- \* Diagramme états-transitions (State Machine Diagram)
  - \* États des objets selon les événements
- \* Diagramme d'activités (Activity Diagram)
  - \* Vue des enchaînements des activités d'un cas d'utilisation ou d'une opération

# Les principaux diagrammes d'UML

- \* Diagramme de séquence (Sequence Diagram)
  - \* Scénario d'un cas d'utilisation : chronologie des opérations
- \* Diagramme de collaboration
  - \* Scénario d'un cas d'utilisation: activités des objets et des messages échangés
- \* Diagramme des composants (Component diagram)
  - \* Représentation des composants logiciels d'un système
- \* Diagramme de déploiement (Deployment diagram)
  - \* Description de l'architecture technique du système
- \* Diagramme de paquetage (Package diagram)
  - \* un paquetage étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML

# Classification des modèles UML

- \* Diagrammes structurels ou statiques (Structure Diagram) :
  - \* (Class diagram)
  - \* (Object diagram)
  - \* (Component diagram)
  - \* (Deployment diagram)
  - \* (Package diagram)
  - \* (Composite Structure Diagram)
  - \* (Profile Diagram)

# Classification des modèles UML

- \* Diagrammes comportementaux (Behavior Diagram) :
  - \* (use-cases ou Use Case Diagram)
  - \* (State Machine Diagram)
  - \* (Activity Diagram)



# Classification des modèles UML

- \* Les diagrammes d'interaction ou dynamiques (Interaction Diagram)
  - \* (Sequence Diagram)
  - \* (Communication Diagram)
  - \* (Interaction Overview Diagram)
  - \* (Timing Diagram)

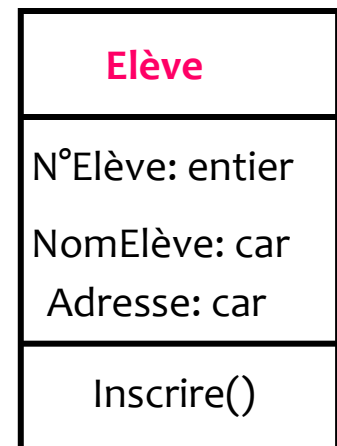
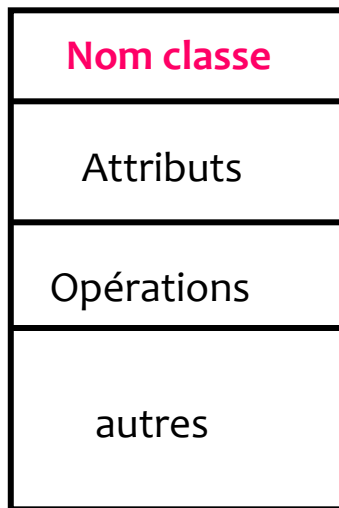
# Principaux diagrammes UML

- \* Diagramme de structure composite (UML 2.x : Composite Structure diagram)
  - \* Décrit sous forme de boîte blanche les relations entre composant d'une classe
- \* Diagrammes de profils (UML 2.2 : Profil Diagram) :
  - \* Permet de spécialiser, de personnaliser pour un domaine particulier un méta-modèle
- \* Diagramme de communication (Communication Diagram)
  - \* depuis UML 2.x, représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets
- \* Diagramme globale d'interaction (Interaction Overview Diagram) :
  - \* depuis UML 2.x, permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité)
- \* Diagramme de temps (Timing Diagram) :
  - \* depuis UML 2.3, permet de décrire les variations d'une donnée au cours du temps

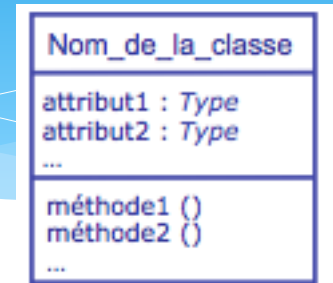
# Les principaux diagrammes d'UML

## \* 4.1 Diagramme de classe

- \* les objets sont identifiés dans le système et portent un nom
- \* Les classes sont créées en regroupant les objets ayant les mêmes propriétés, mêmes comportements
- \* Un objet est une instance d'une classe
- \* Représentation UML d'une classe



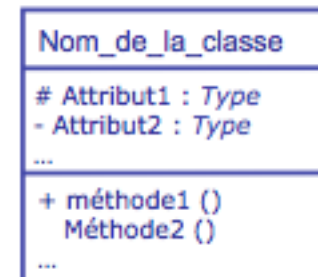
# Diagramme de classe



- \* Notion de classe :
  - \* Structure d'un objet
  - \* Un objet est issu d'une classe, c'est le produit qui sort du moule
- \* Notation
  - \* Un objet est une instance (occurrence d'une classe)
- \* Une classe est composée :
  - \* Attributs : données dont les valeurs représentent l'état de l'objet
  - \* Méthodes : opérations applicables aux objets

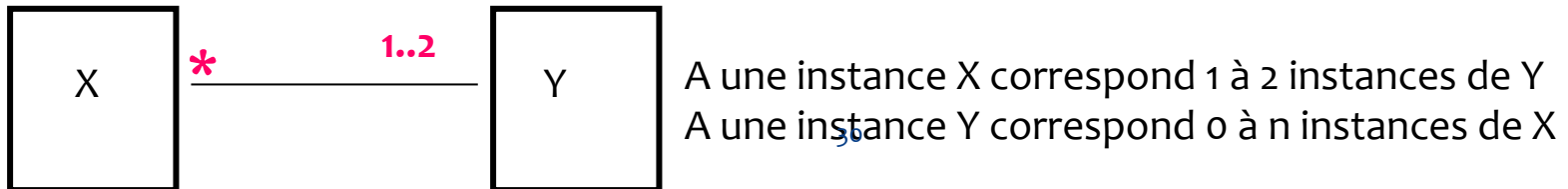
# Diagramme de classe

- \* Visibilité des objets :
  - \* définissent les droits d'accès aux données (pour la classe elle-même, d'une classe héritière, ou bien d'une classe quelconque)
- \* Publique (+)
  - \* les classes peuvent accéder aux données et
  - \* méthodes d'une classe définie avec le niveau
  - \* de visibilité public
- \* Protégée (#) : l'accès aux données est
  - \* réservé aux fonctions des classes héritières
- \* Privée (-) : l'accès aux données est limité
  - \* aux méthodes de la classe elle-même



# Diagramme de classe

- \* Association entre classes
  - \* Liens entre les instances
  - \* Rôle de l'association et son sens
  - \* Cardinalité des instances associées



# Diagramme de classe

- \* Association : Connexion sémantique entre deux classes
- \* Navigabilité :
- \* Par défaut une association est navigable dans les deux sens

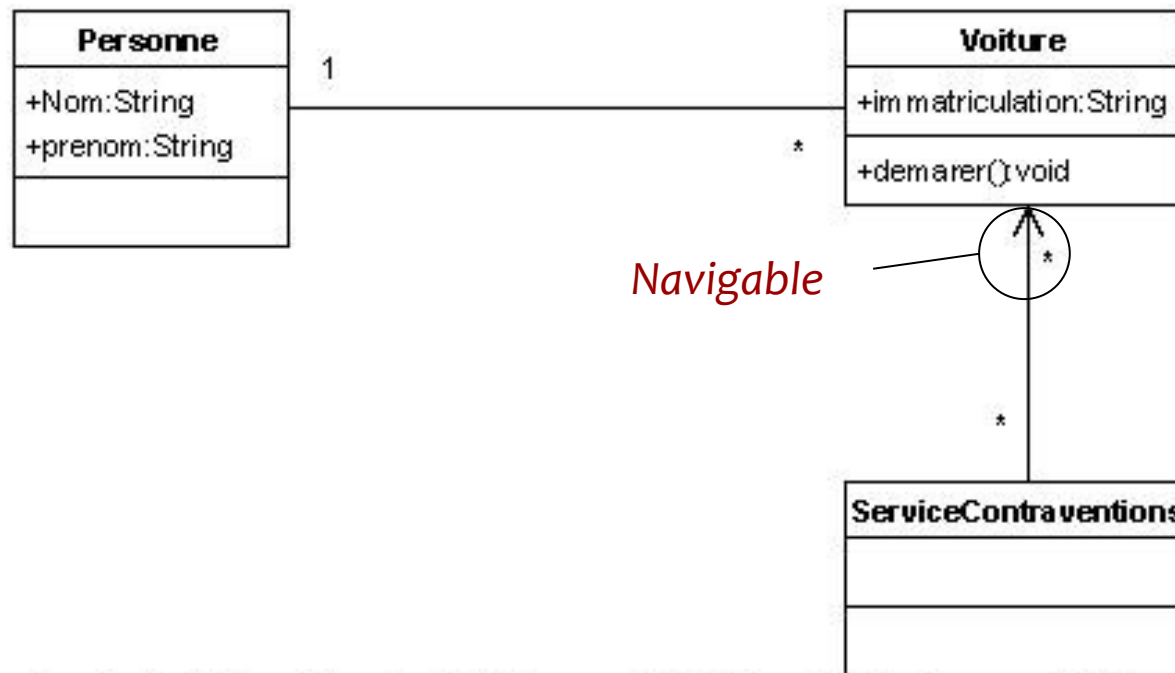


Created with Poseidon for UML Community Edition. Not for Commercial Use.

- \* Chaque instance de voiture a un lien vers le propriétaire
- \* Chaque instance de Personne a un ensemble de lien vers les voitures

# Diagramme de classe

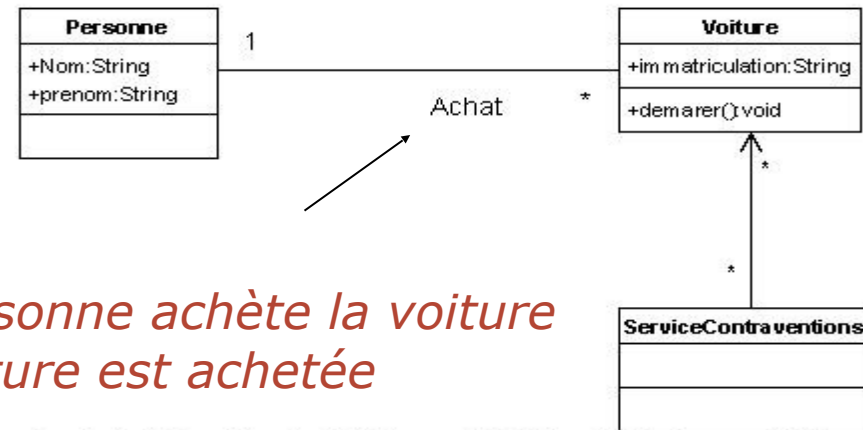
- \* Restriction de la navigabilité
- \* Le service de contravention est associé à une ou plusieurs voiture(s)
- \* La voiture ne connaît pas service de contravention





# Diagramme de classe

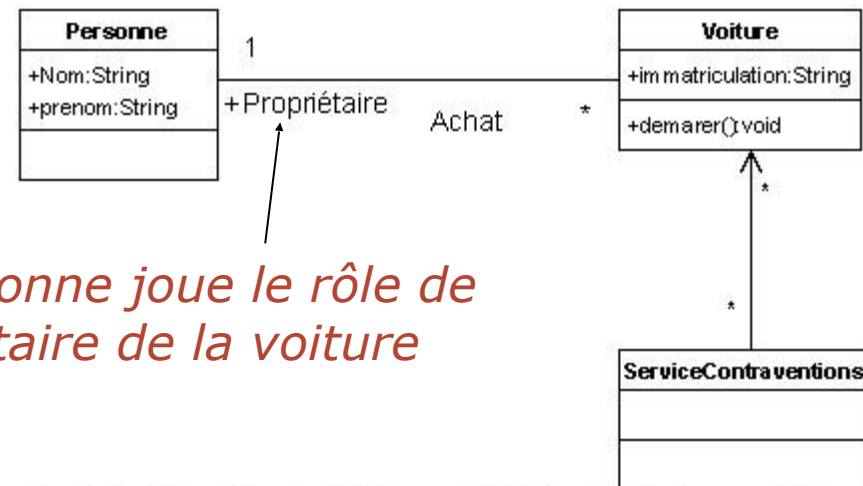
- \* Documentation d'une association :
- \* Nom de l'association
- \* lien sémantique entre les classes



*La personne achète la voiture*  
*La voiture est achetée*

Created with Poseidon for UML Community Edition. Not for Commercial Use.

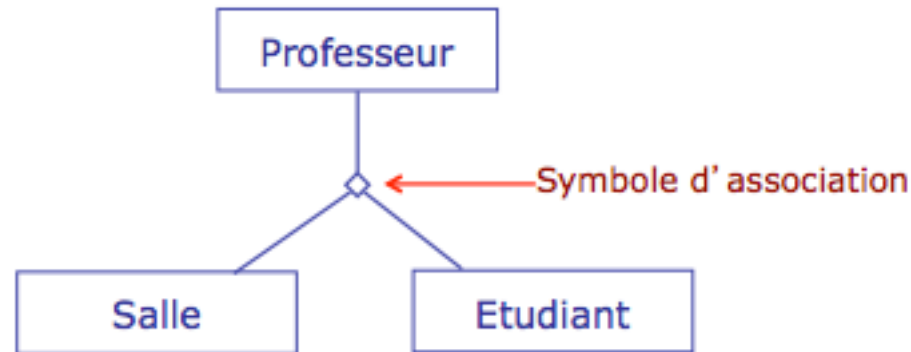
- \* Rôle d'une association
- \* Spécification du rôle de la classe



*La personne joue le rôle de propriétaire de la voiture*

# Diagramme de classe

- \* Relation n-aire
- \* Type particulier d'association qui relie plus de deux classes



# Diagramme de classe

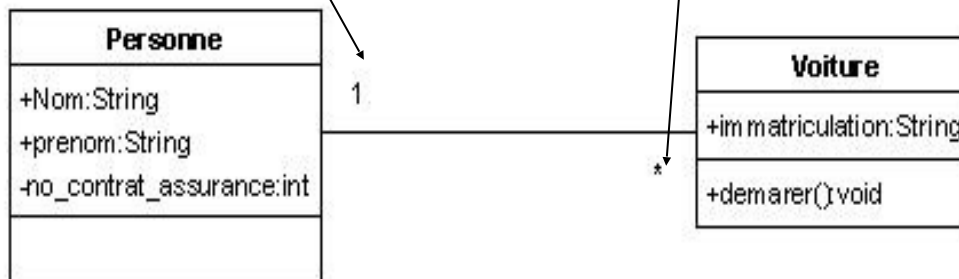
- \* Multiplicités :
- \* 1 : la classe est en relation avec un et un seul objet de l'autre classe
- \* 1..\* : la classe est en relation avec au moins un objet de l'autre classe
- \* 0..\* : la classe est en relation avec 0 ou n objets de l'autre classe
- \* 0..1 : la classe est en relation avec au plus un objet de l'autre classe

# Diagramme de classe

- \* Multiplicités, règles de gestion, sens de lecture (inverse de MERISE)

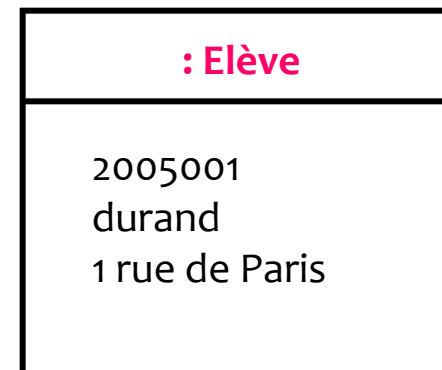
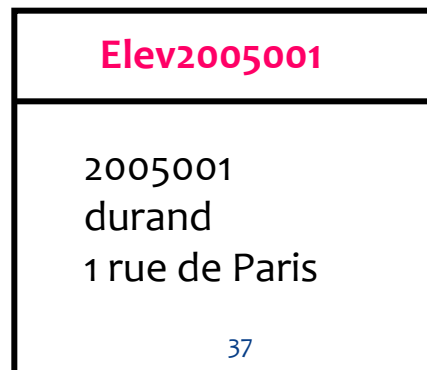
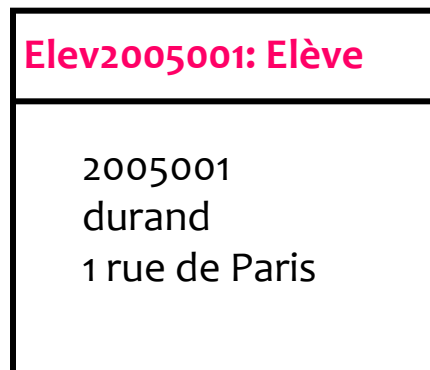
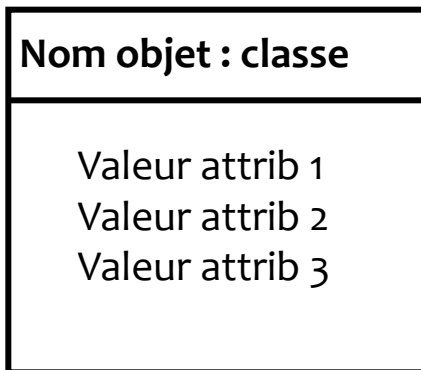
*Une voiture est achetée par une et une seule personne*

*Une personne peut acheter 0 ou n voitures*



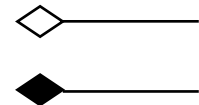
# Diagramme d'objet !

## \* 4.2. Représentation UML d'un objet :



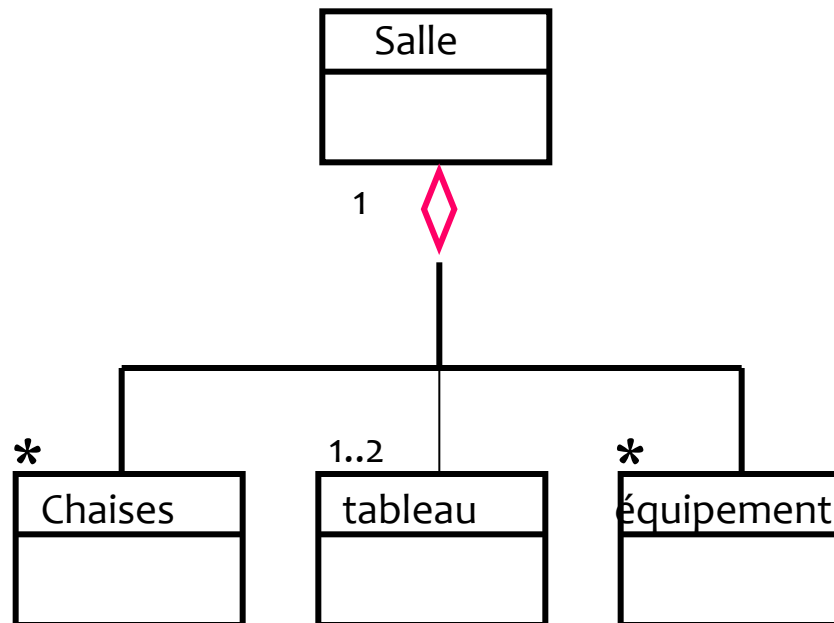
# Diagramme de classe

- \* Cas particulier d'association exprimant une relation de contenance
- \* Exemples :
  - \* Une voiture a 4 roues
  - \* Un dessin contient un ensemble de figures géométriques
  - \* Une présentation PowerPoint est composé de transparents
  - \* Une équipe de recherche est composée d'un ensemble de personnes
- \* Deux types de relations de contenance en UML
  - \* Agrégation (l'un sans l'autre)
  - \* Composition (Agrégation forte, pas l'un sans l'autre)
  - \* Composition = identifiant relatif avec MERISE



# Diagramme de classe

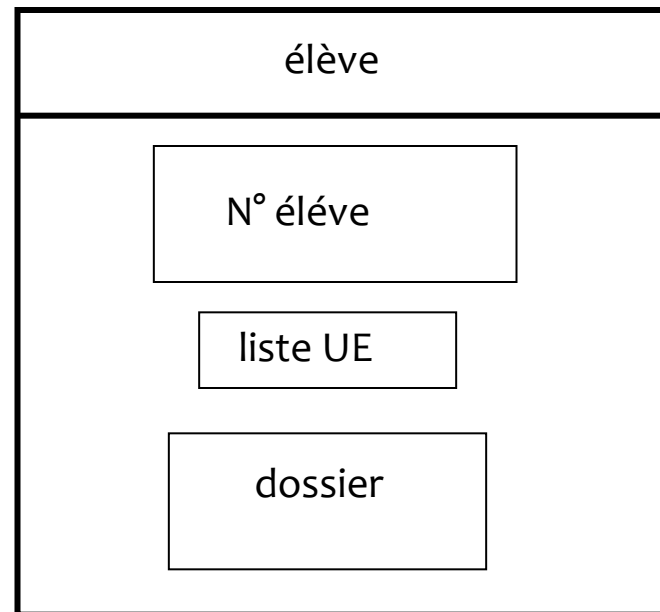
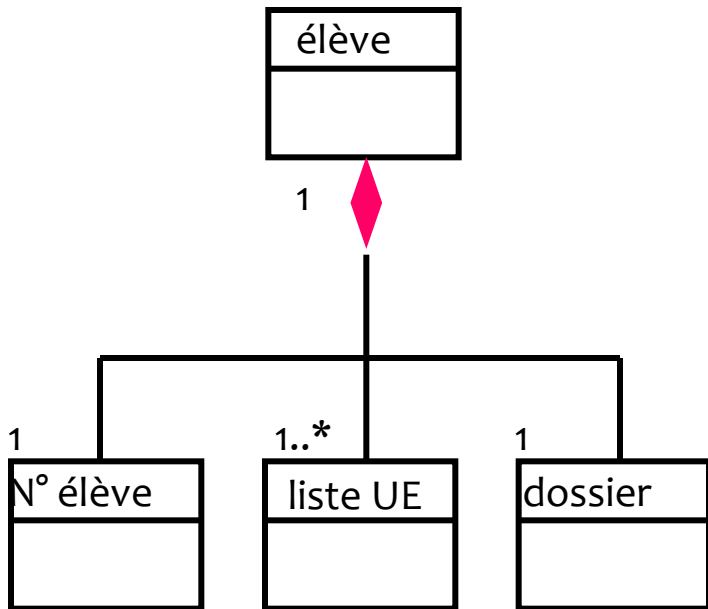
- \* Agrégation
  - \* Association entre une classe de type « ensemble » avec plusieurs classes de type « éléments »



# Digramme de classe

- \* Composition

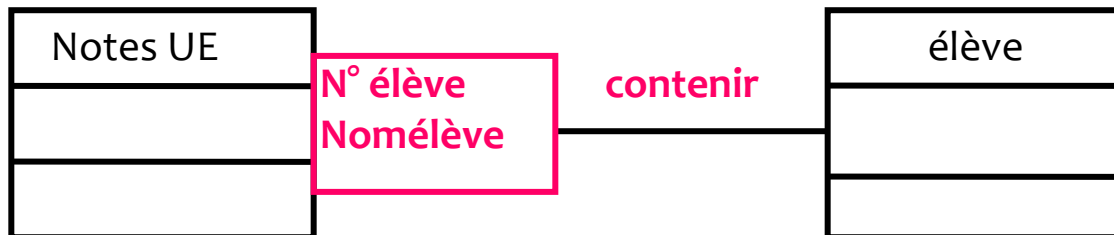
- \* Agrégation avec une contrainte de durée de vie
- \* La suppression de la classe « composé » implique la suppression des classes « composant »





# Diagramme de classe

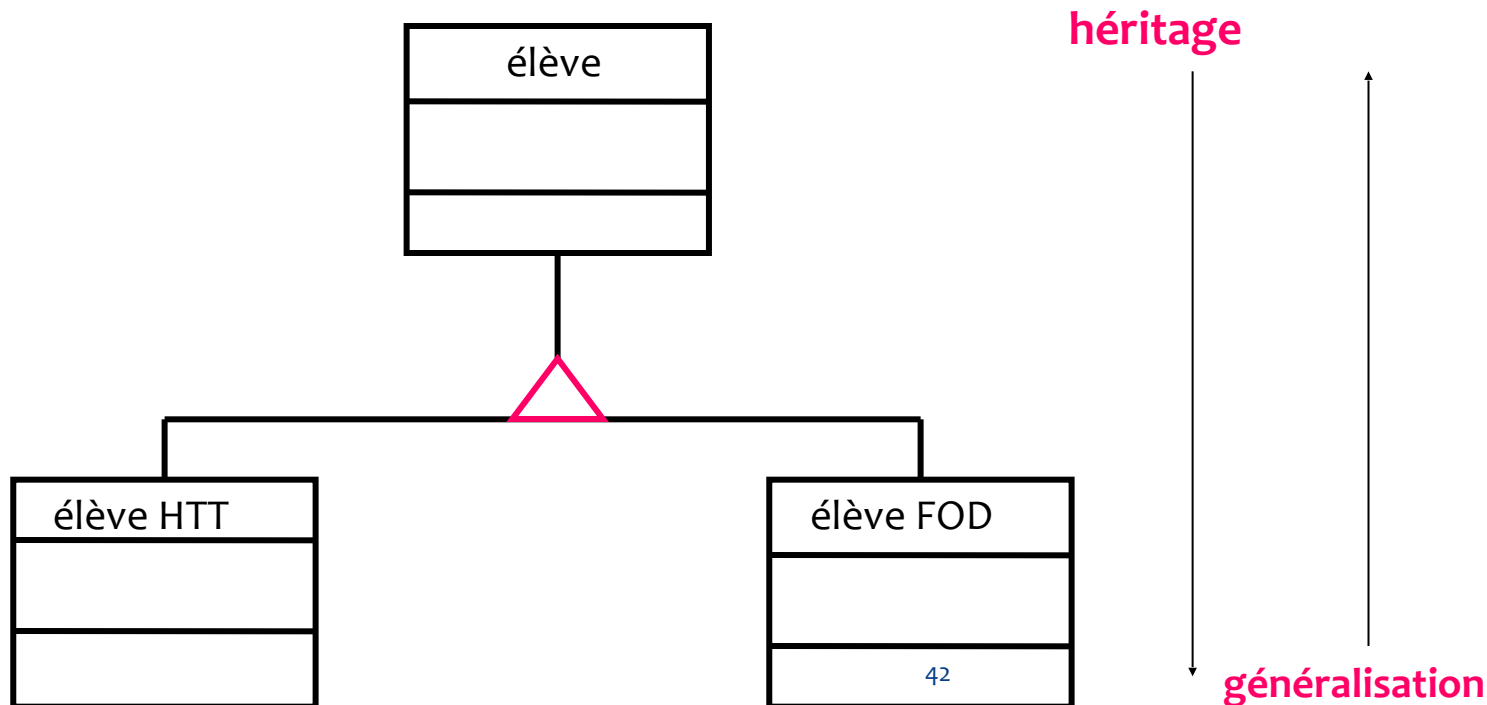
- \* Qualification d'une association
  - \* Sémantique d'une association entre deux classes
  - \* Restriction d'une association



« La liste des notes d'une UE contient le n° des élèves et leur nom »

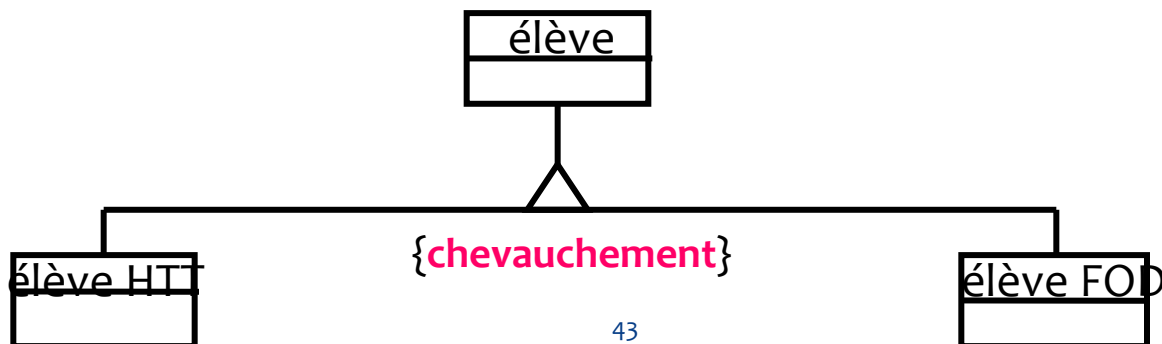
# Diagramme de classe

- \* Généralisation et héritage simple
  - \* Généralisation: création d'une superclasse à partir de classes
  - \* Héritage: création de sous classes à partir d'une classe



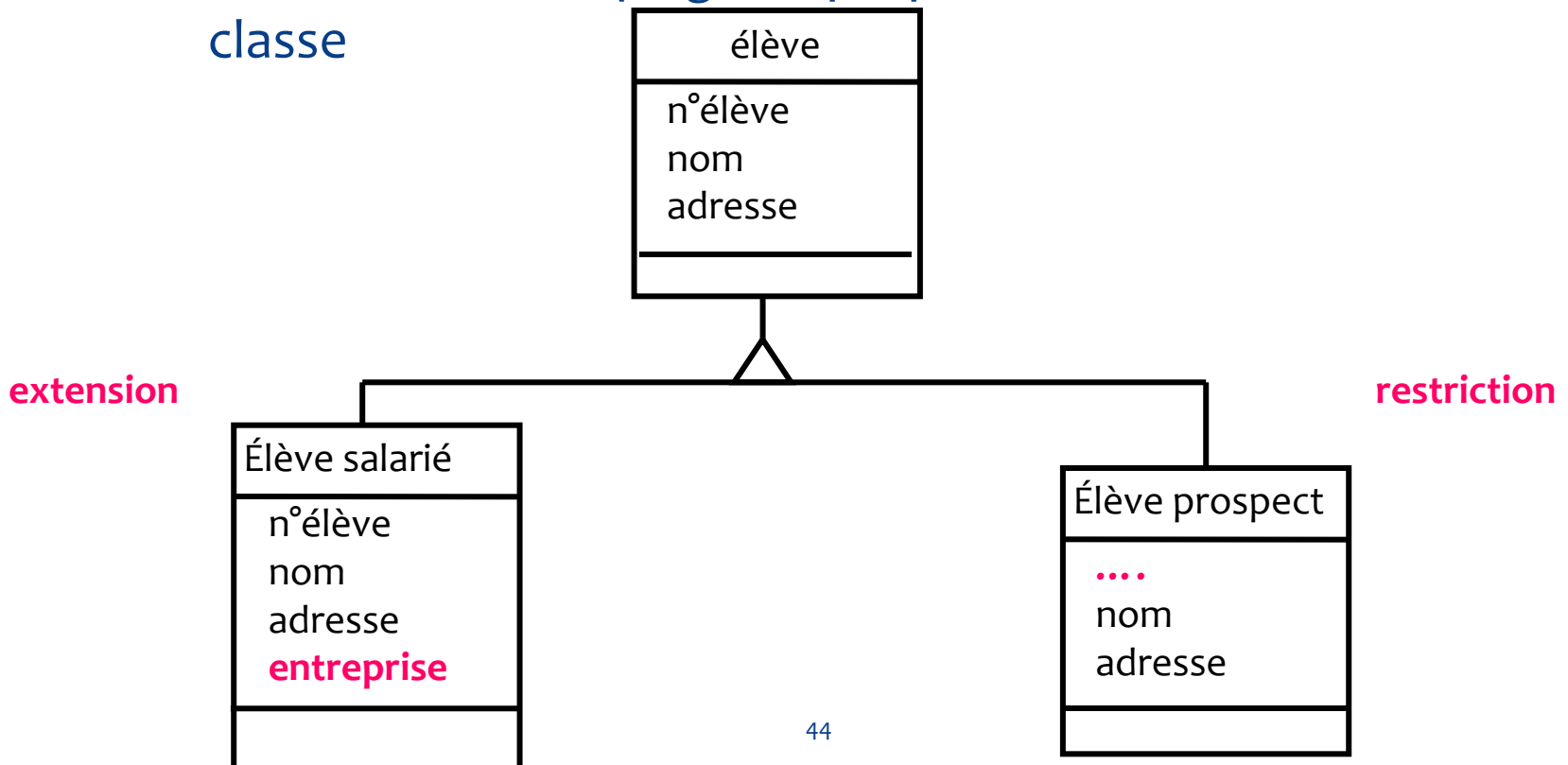
# Diagramme de classe

- \* Héritage avec recouvrement
  - \* Chevauchement : deux sous classes avec des instances identiques
  - \* Disjoint : les instances d'une sous classe ne peuvent être dans une autre sous classe
  - \* Complète : la généralisation ne peut être étendue
  - \* Incomplète : la généralisation peut être étendue



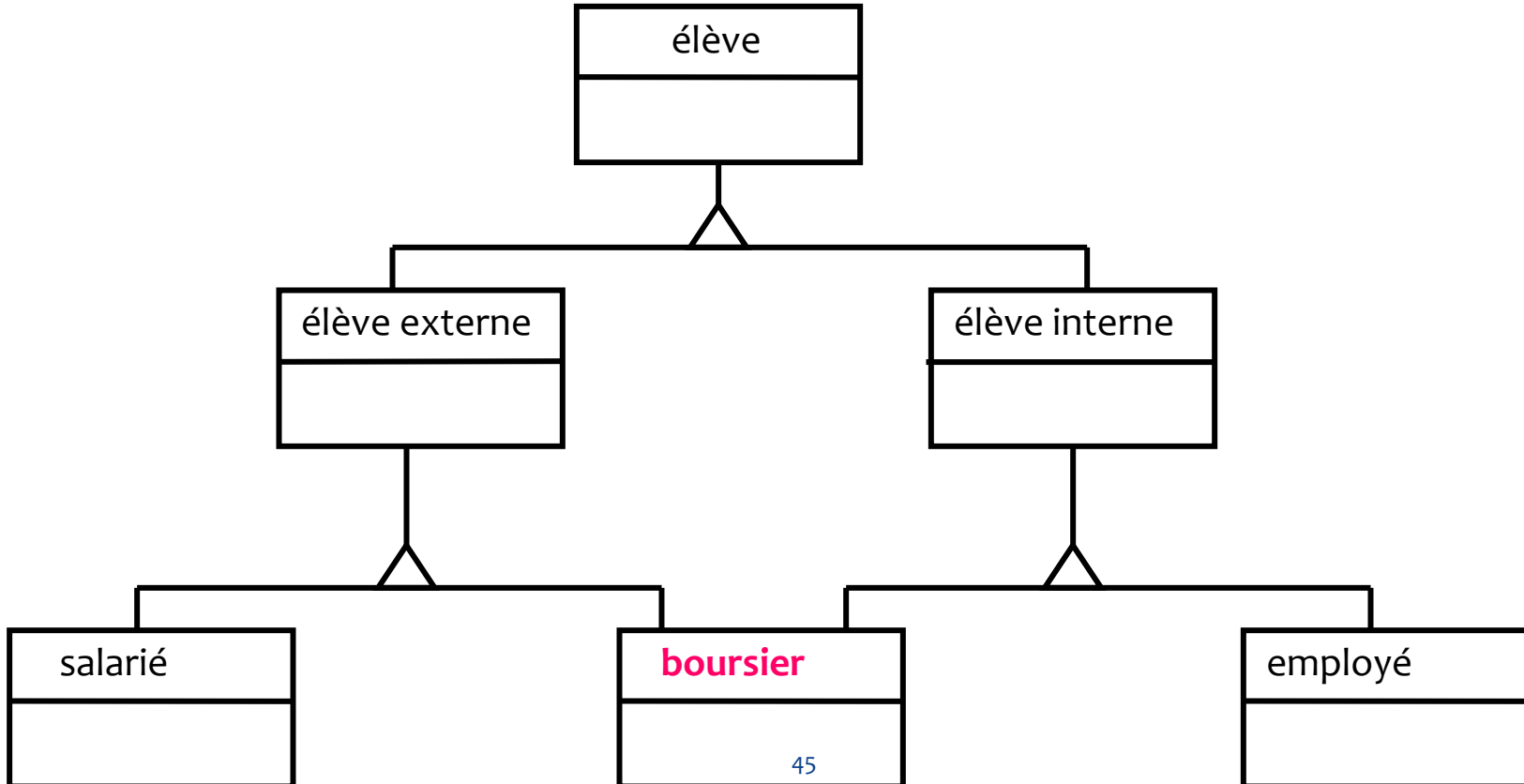
# Diagramme de classe

- \* Extension et restriction de classe
  - \* Extension : ajout de propriétés dans une sous classe
  - \* Restriction : masquage de propriétés dans une sous classe



# Diagramme de classe

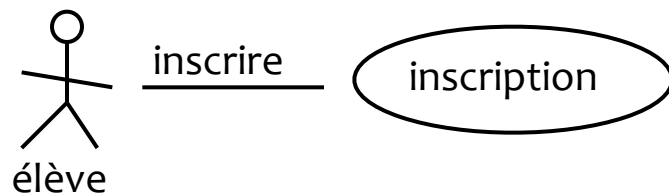
- \* Héritage multiple
- \* Une classe hérite de deux classes parentes



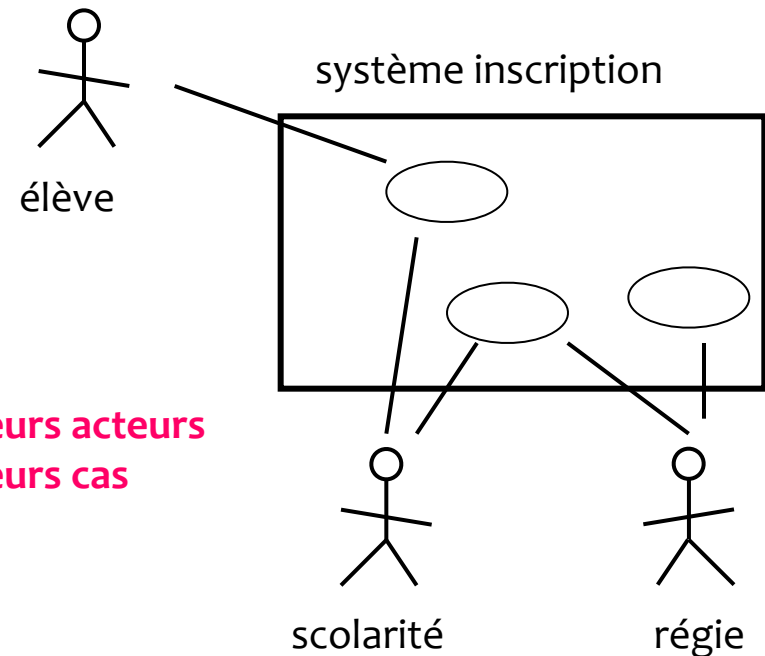
# Diagramme « use case »

## \* 4.3 Diagramme des cas d'utilisation

- \* Description de l'interaction entre l'utilisateur et le système
- \* Recensement des **besoins des utilisateurs**
- \* Descriptions textuelles + diagrammes de tous les cas d'utilisation



**Cas d'un élève qui s'inscrit**



**Plusieurs acteurs  
Plusieurs cas**

# Diagramme de « use case »

Structurer les besoins des utilisateurs et les objectifs correspondants du système.

- \* Préoccuper des cas "réels" des utilisateurs ; ils ne présentent pas de solutions d'implémentation et ne forment pas un inventaire fonctionnel du système.
- \* Notation :



Cas d'utilisation

Objectif du système, motivé par un besoin d'un (ou plusieurs) acteur(s)



Acteur

Personne ou composant d'origine d'une interaction avec le système



Note

Documente un élément du modèle



Relation d'utilisation

Le cas source contient aussi le comportement décrit dans la cas destination

# Diagramme « use case »

- \* Description textuelle

- \* Transcription textuelle de la description des cas d'utilisation

- \* Compléments aux diagrammes

- \* Avantages:

- \* La rédaction permet de corriger le diagramme

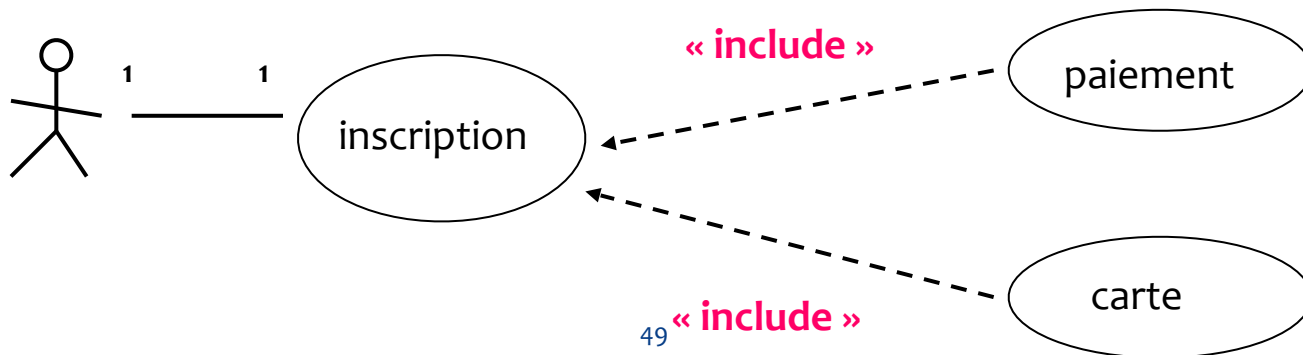
- \* Le diagramme oblige à rédiger chaque cas

cas	<b>Inscription d'un élève</b>
résumé	Procédure d'inscription d'un élève jusqu'à la délivrance de sa carte
acteur primaire	L'élève
acteur secondaire	La scolarité, la régie
pré-conditions	Vérification préalable des conditions d'inscription
résultats	Dossier d'inscription, paiement, délivrance carte élève
description	<ol style="list-style-type: none"><li>1. l'élève présente sa demande</li><li>2. Saisie des UE</li><li>3. Calcul du coût</li><li>4. ....</li></ol>
exceptions	Pas de délivrance <sup>48</sup> de carte si 1 UE à agrément



# Diagramme de « use case »

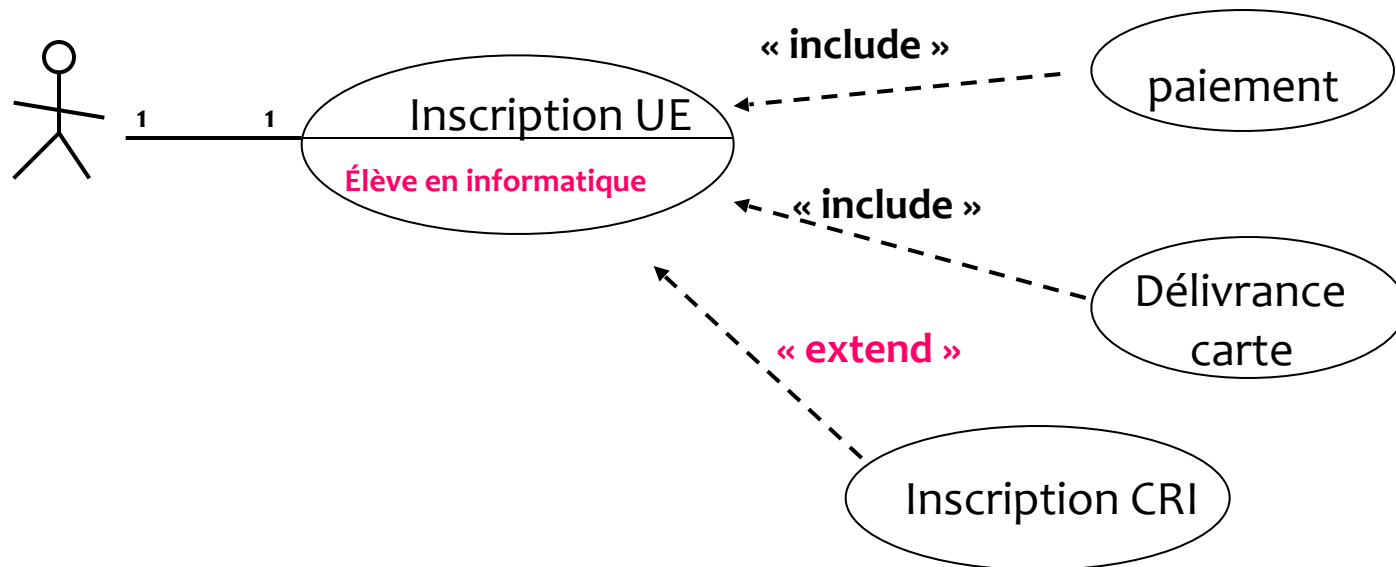
- \* Relations entre cas d'utilisation
  - \* Réutilisation de cas en utilisant les relations
    - \* D'inclusion *include*
    - \* D'extension *extend*
    - \* de généralisation
- \* Inclusion
  - \* Une instance contient le comportement d'une autre instance



# Diagramme de « use case »

- \* Extension

- \* Le comportement d'une instance peut être étendue par le comportement d'une autre instance
- \* Point d'extension mentionné dans le cas d'utilisation



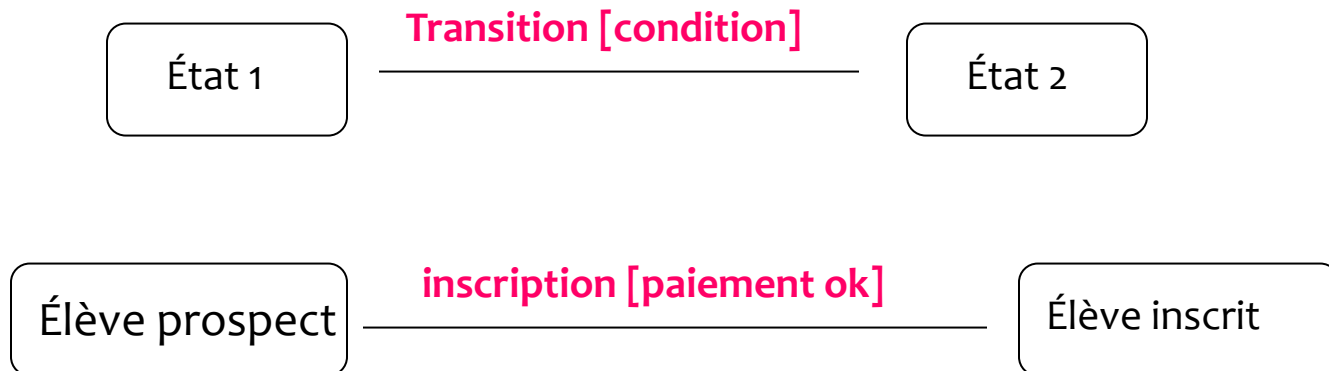
# Diagramme « use case »

- \* Les cas d'utilisation doivent au final être permis par le nouveau logiciel
- \* Ce sont eux qui sont implantés et ce sont les cas d'utilisation prévus qui vont présider à l'élaboration des lignes de tests
- \* Ce sont eux qui unifient et qui donne naissance à PU

# Diagramme état transition

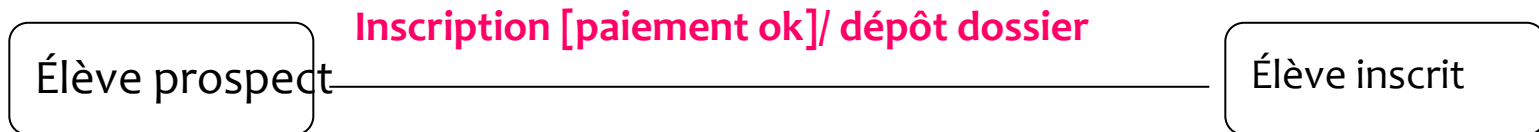
## \* 4.4. Diagramme états-transitions

- \* Montre les états simples, les transitions et les états composites imbriqués
- \* L'état d'un objet à un instant  $t$  peut changé à l'instant  $t+1$
- \* Le passage d'un état à un autre est une transition
- \* La condition de passage est appelée « garde »



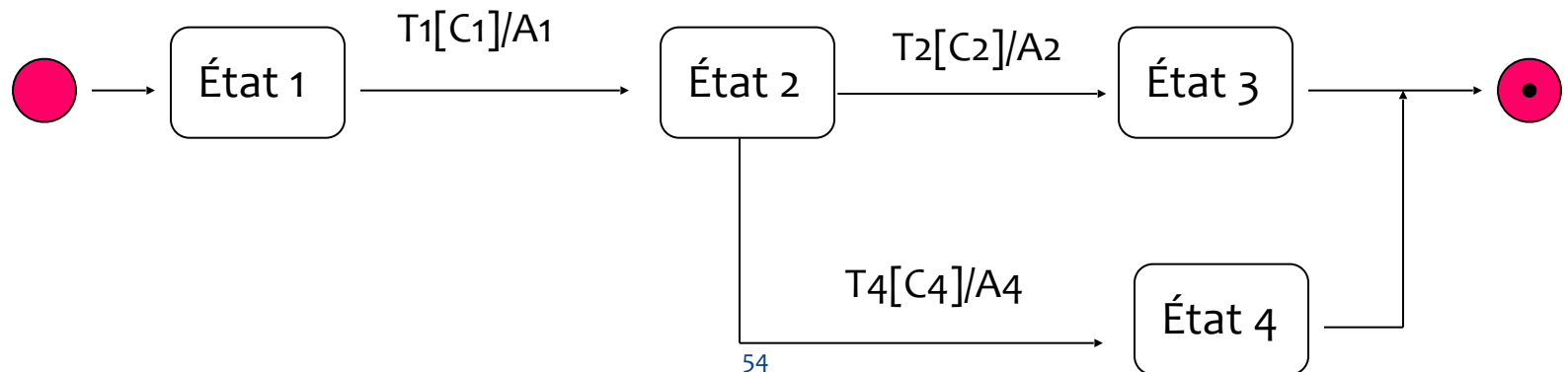
# Diagramme état transition

- \* On peut préciser une action ou activité à la transition



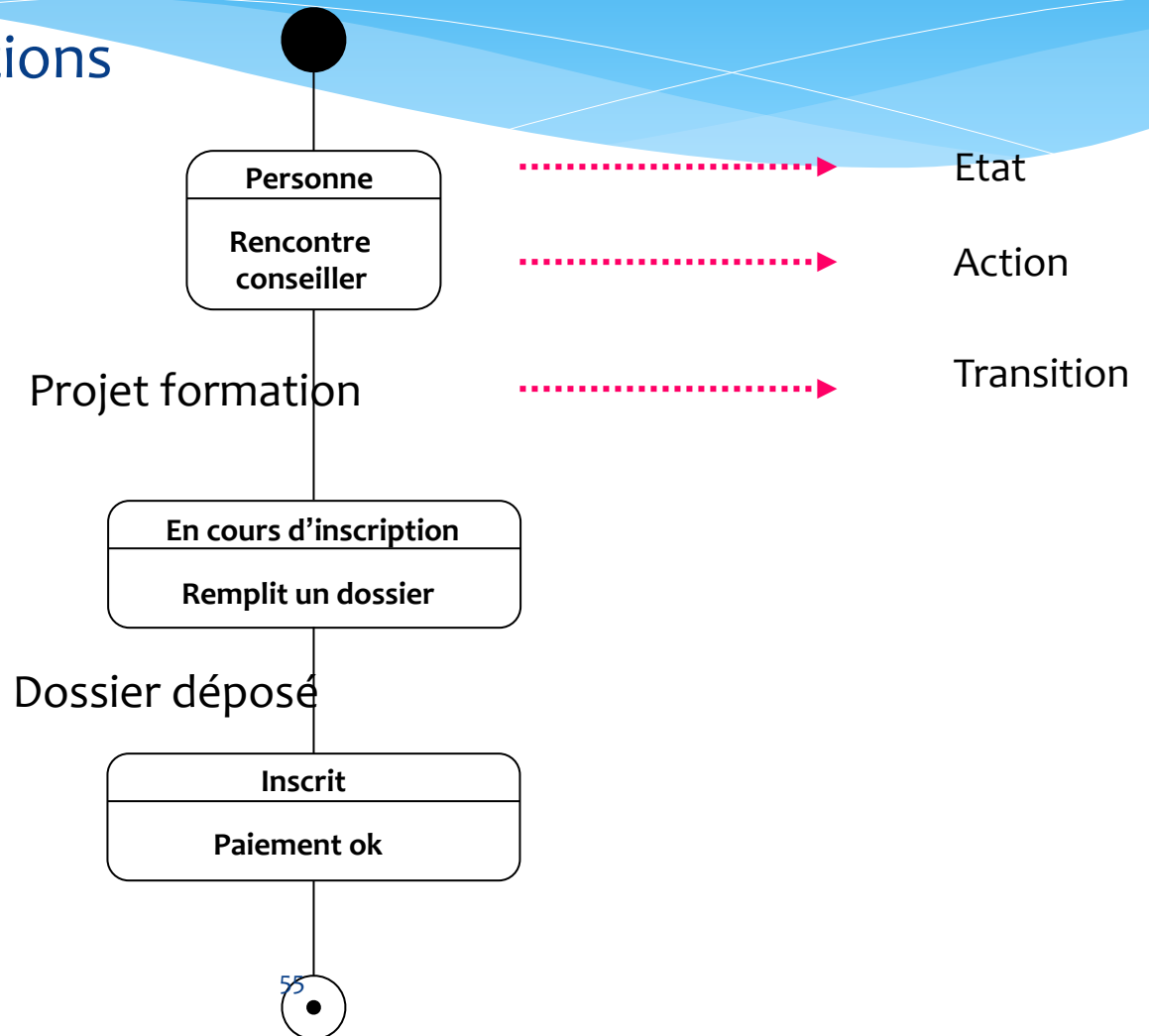
# Diagramme état transition

- \* Suite d'états et de transitions avec
  - \* Début: ●
  - \* Fin: ⊙
  - \*  $T_i$  : transition –  $C_i$  : condition –  $A_i$  : action



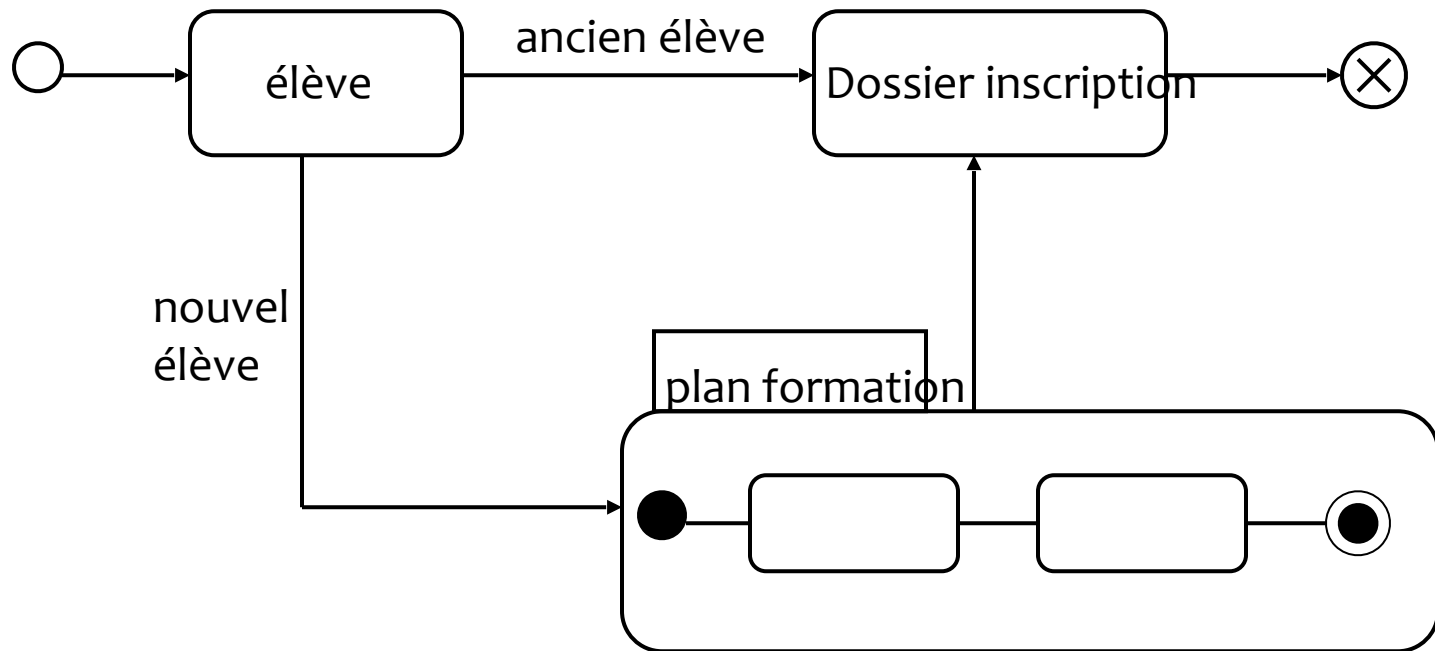
# Diagramme état transition

## \* Exemples états transitions



# Diagramme d'état transition

- \* Exemple de diagramme machine à états



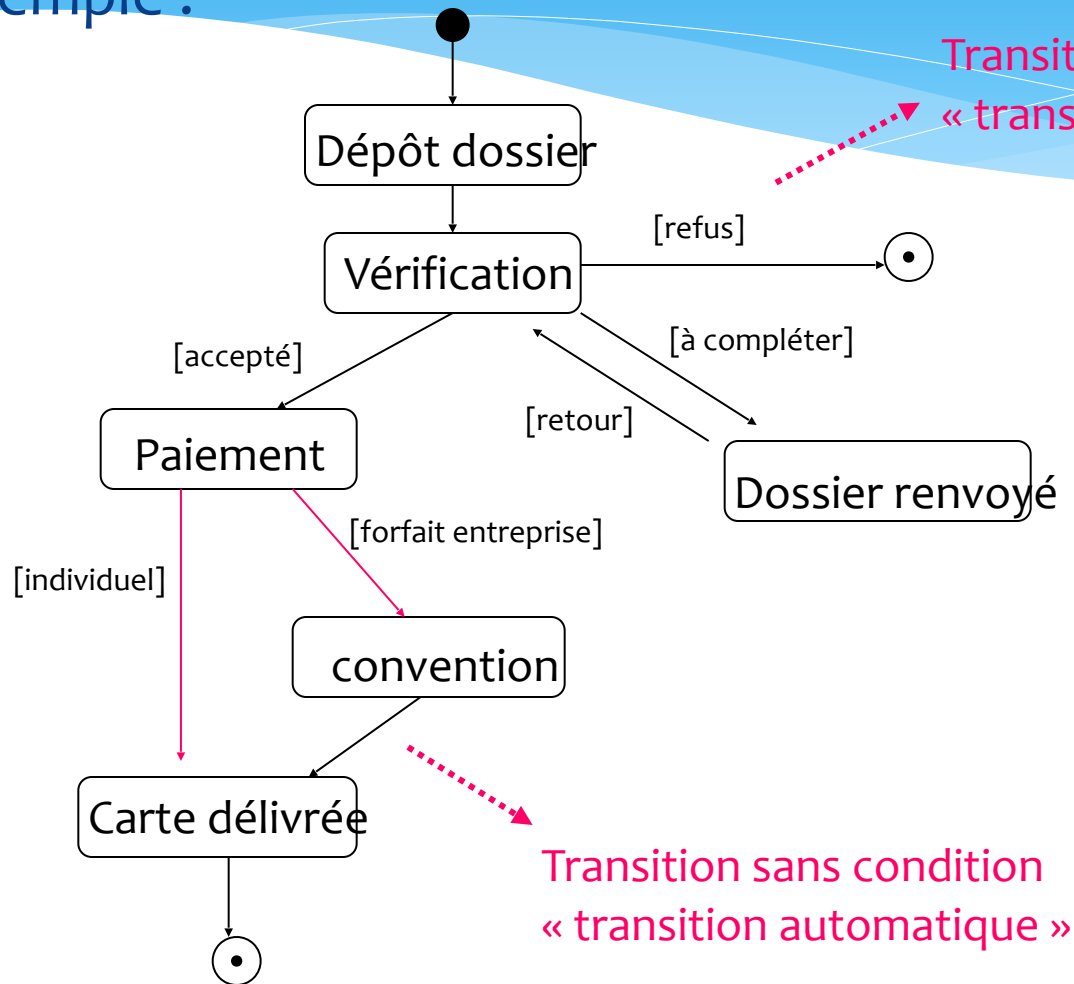


# Diagramme d'activité

- \* 4.5. Diagramme d'activités :
  - \* Description des **activités d'un cas d'utilisation** ou d'une opération
  - \* Diagramme de type : état-action
  - \* Exécution d'activités différentes selon le résultat de l'activité précédente
  - \* Exécution synchronisée : plusieurs activités en // avant de passer à l'activité suivante

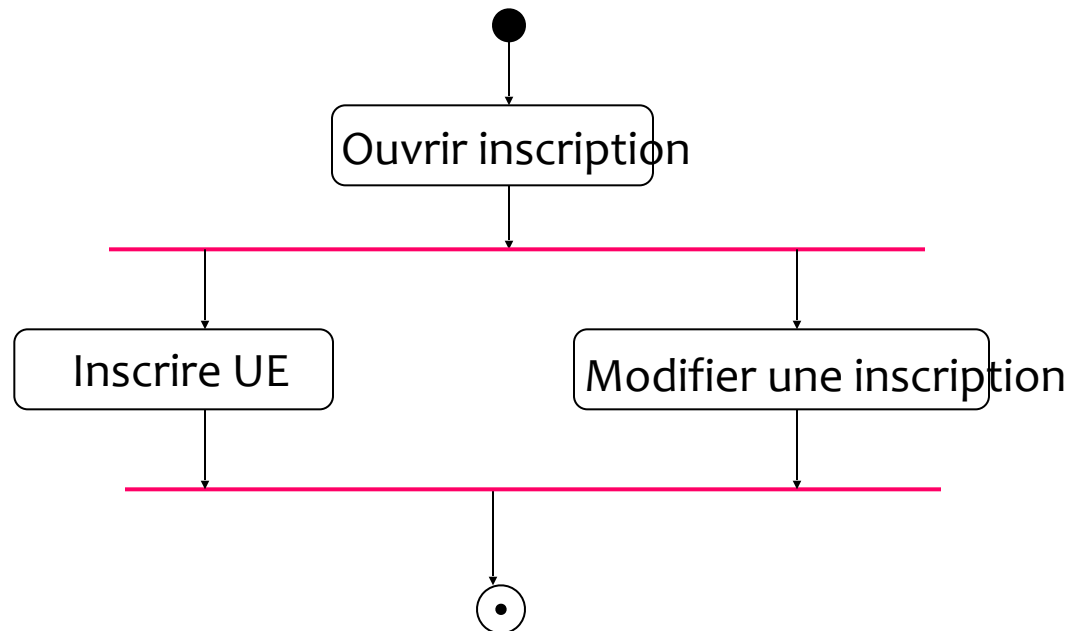
# Diagramme d'activité

\* Exemple :





# Diagramme d'activité

## \* Exemple synchronisation



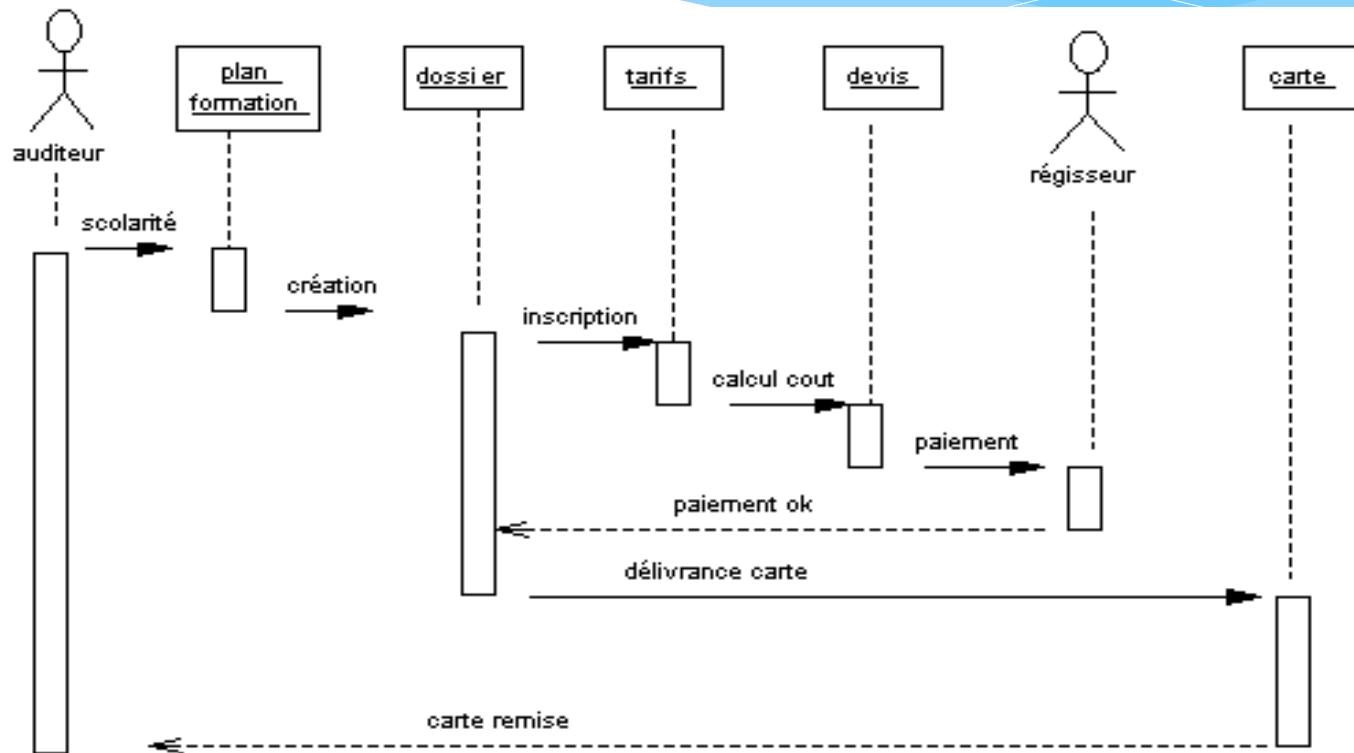
Inscrire UE et modifier élève sont deux actions //

# Diagramme de séquence

- \* 4.6. Diagramme de séquence :
  - \* Interactions entre objets et **chronologie des échanges** entre ces objets
  - \* Base : les cas d'utilisation
  - \* Échange de messages pour déclencher une opération
  - \* Mention des objets créés ou détruits lors des exécutions
  - \* Spécification des contraintes de temps : durée
  - \* Messages synchrones : 
  - \* Messages asynchrones : 

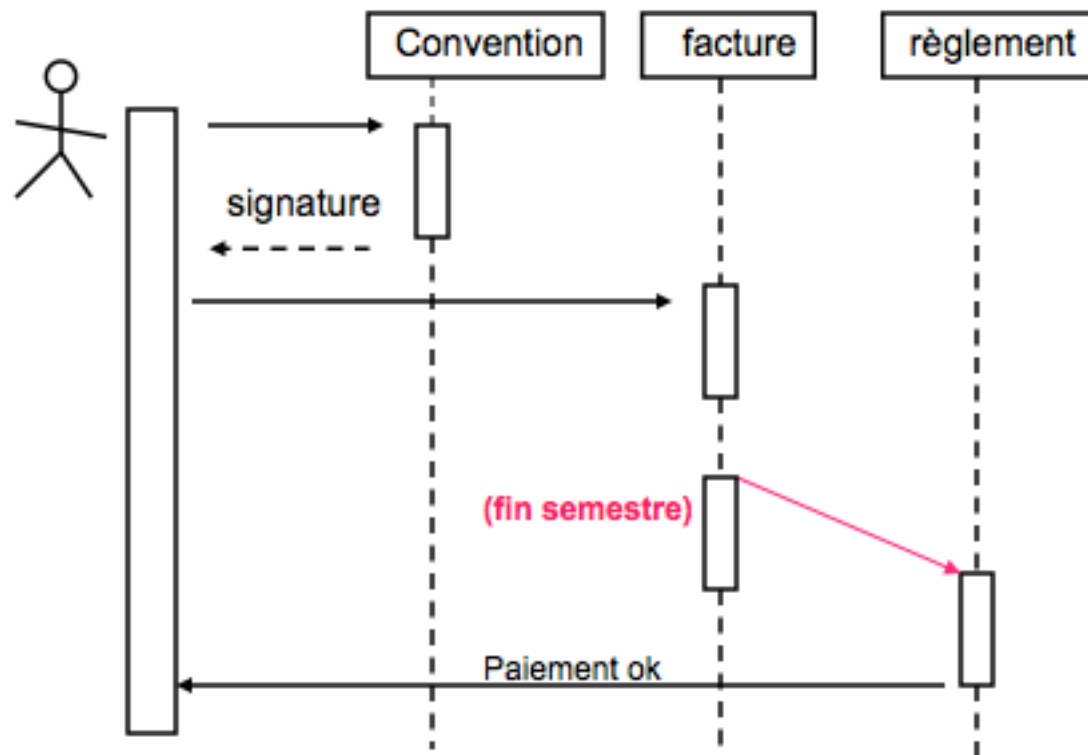
# Diagramme de séquence

## \* Exemple diagramme de séquence



# Diagramme de séquence

- \* Exemple diagramme de séquence : durée



# Diagramme de collaboration

- \* 4.7. Diagramme de collaboration
  - \* Relations entre les **objets** et **messages** échangés
  - \* Diverses informations mentionnées sur le diagramme
    - \* synchronisation : les préalables à l'envoi du message;
    - \* n° du message : ordre chronologique du message;
    - \* condition du déclenchement de l'envoi;
    - \* type d'envoi: séquentiel ou parallèle;
    - \* résultat : valeur retournée...

## Exemples:

[activité professionnelle=Ok] 1: établir plan()

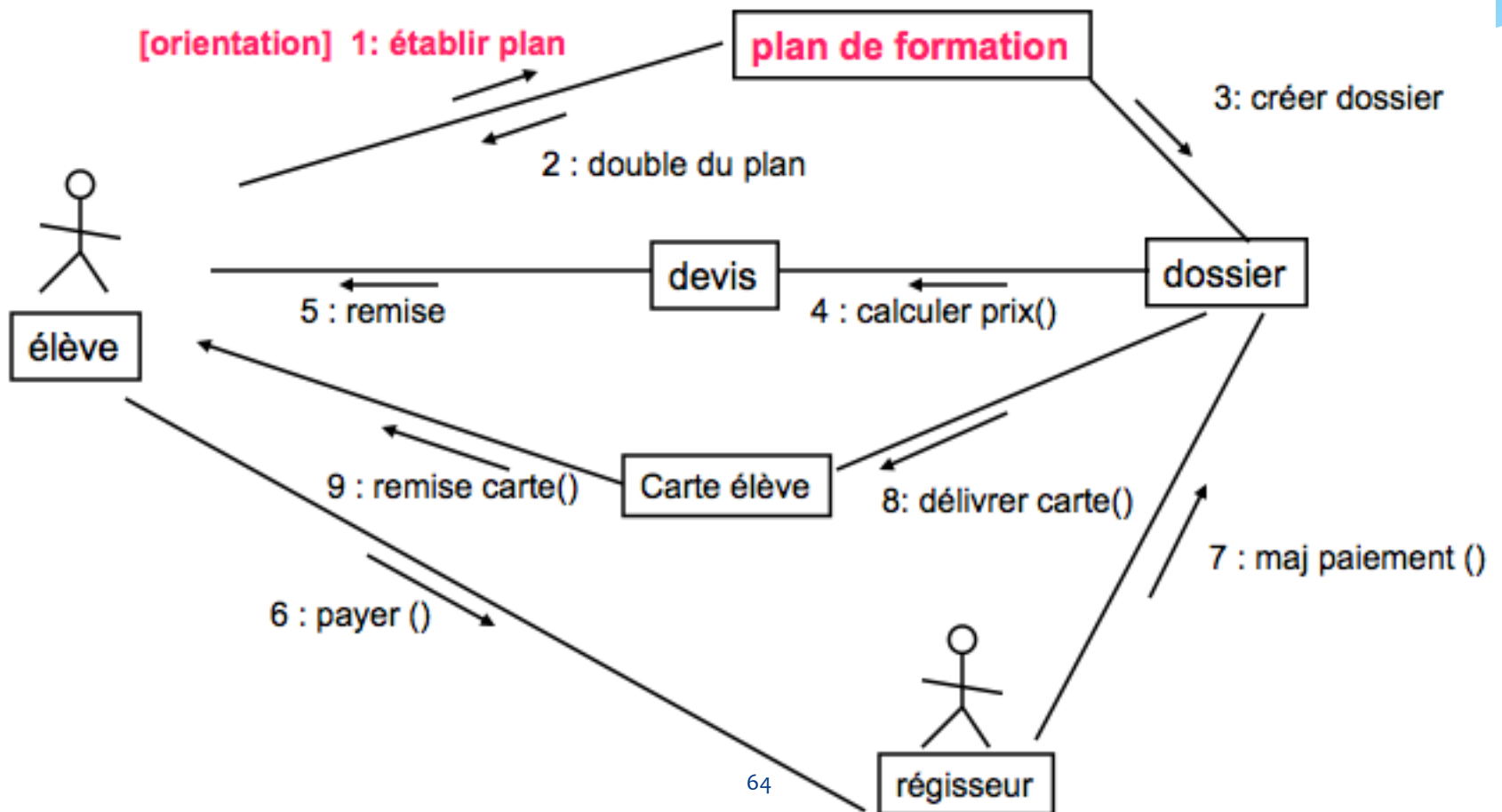
Un élève ne peut s'inscrire que s'il a une activité professionnelle

2 / || [j=1...n] 3 : inscrireUE()

Le message 3 ne sera envoyé qu'après le message 2

Envoi en // de n messages

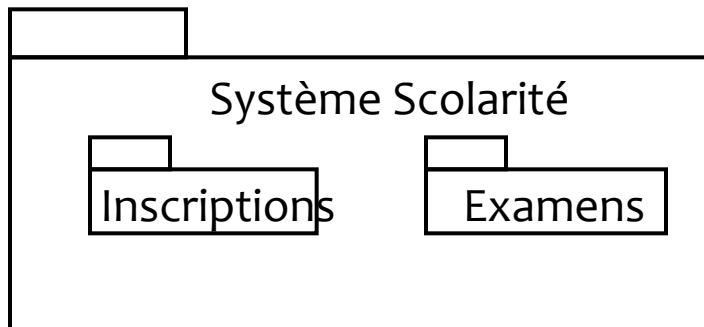
# Diagramme de collaboration





# Diagramme de composants

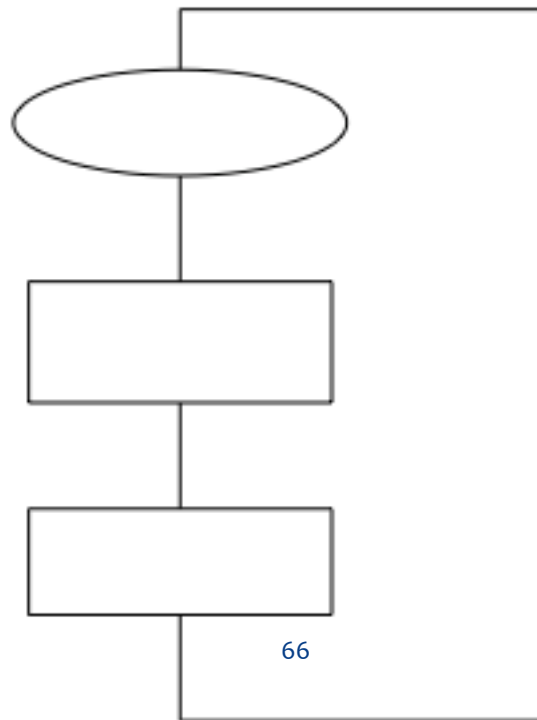
- \* 4.8. Diagramme de composants :
  - \* Description des composants du système
  - \* Décomposition en sous-système, programme, processus, tâche, module

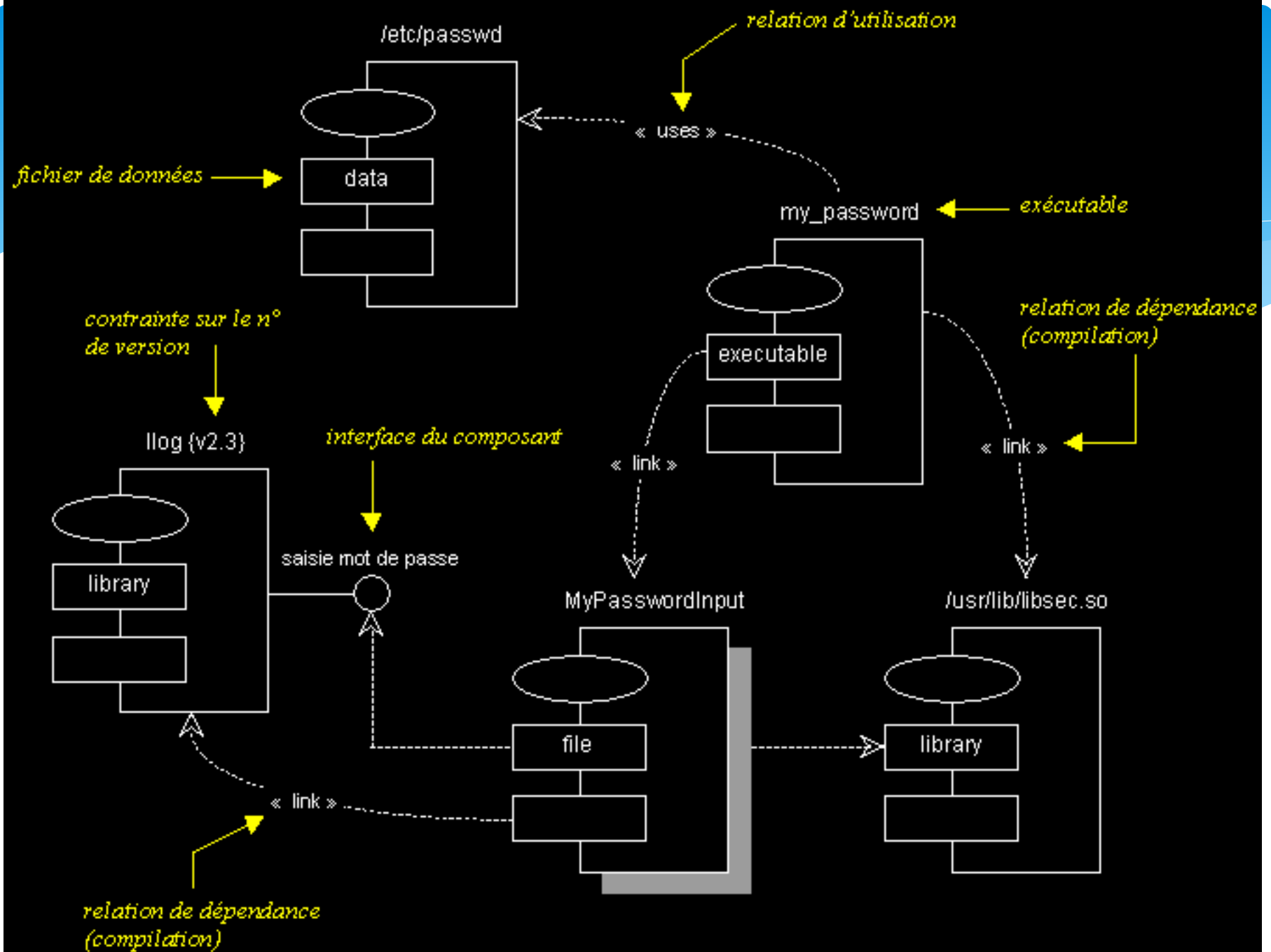


**Système et sous-systèmes**

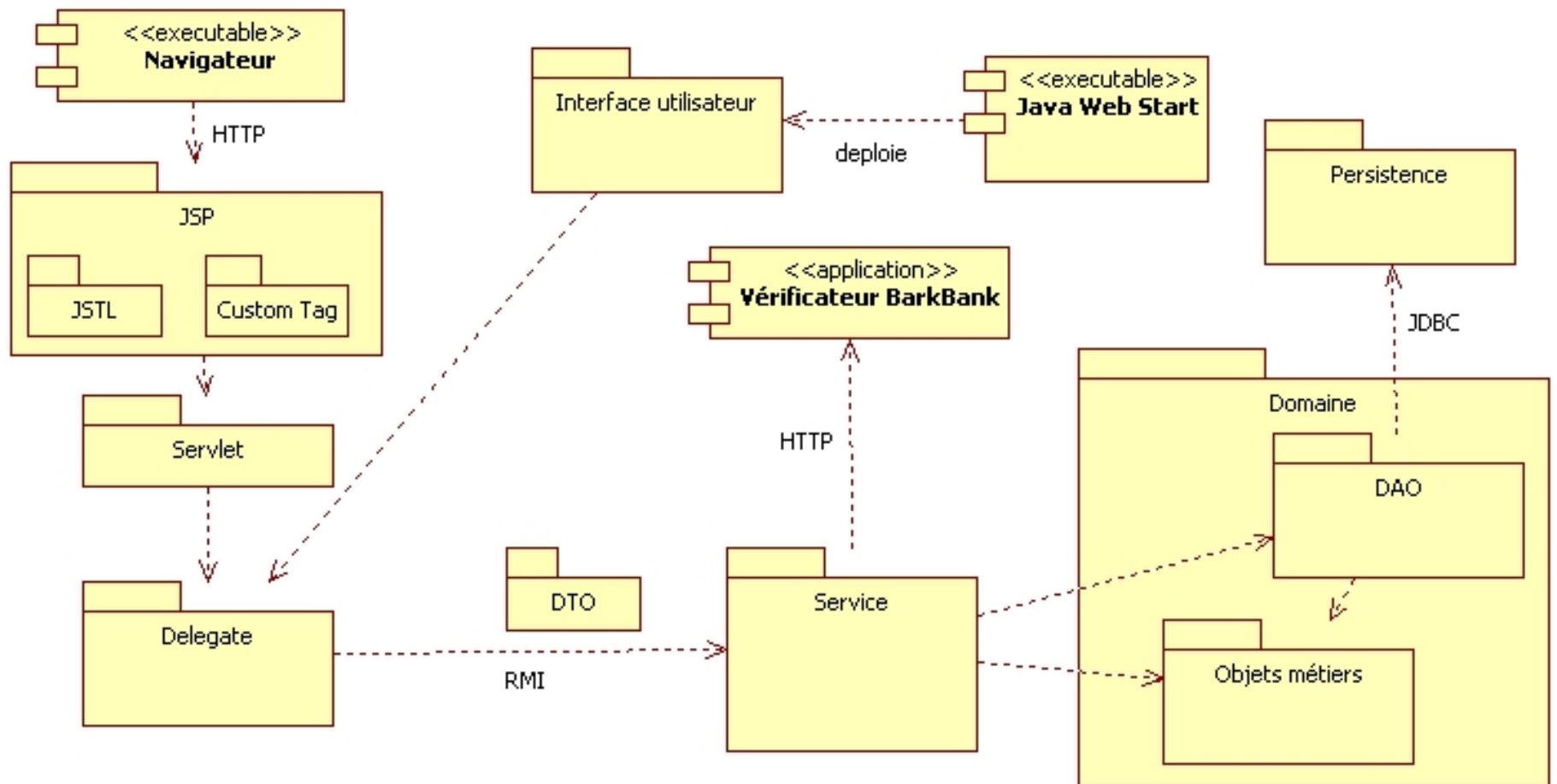
# Diagramme de composants

- \* Diagramme d'un module
  - \* Décomposition d'un sous système en modules





# Diagramme de composant

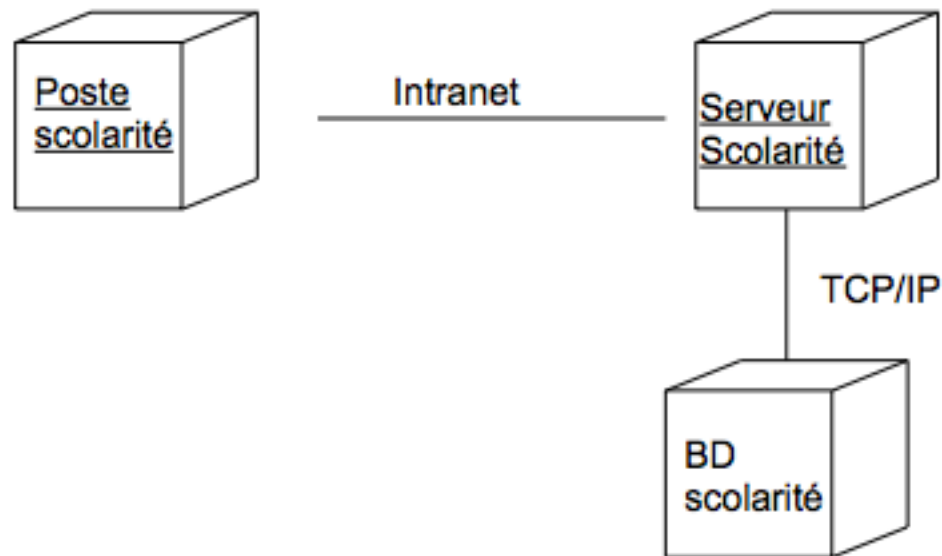


# Diagramme de déploiement

- \* 4.9. Diagramme de déploiement :
  - \* Description de l'architecture physique des composants matériels du système
  - \* Un nœud = un composant matériel = un cube
  - \* Lien entre les cubes = communication entre les nœuds
  - \* Classes de composants = noms des classes
  - \* Objets de composants = noms soulignés des instances
  - \* Deux types de noeuds:
    - \* Processeur
    - \* Périphérique au processeur

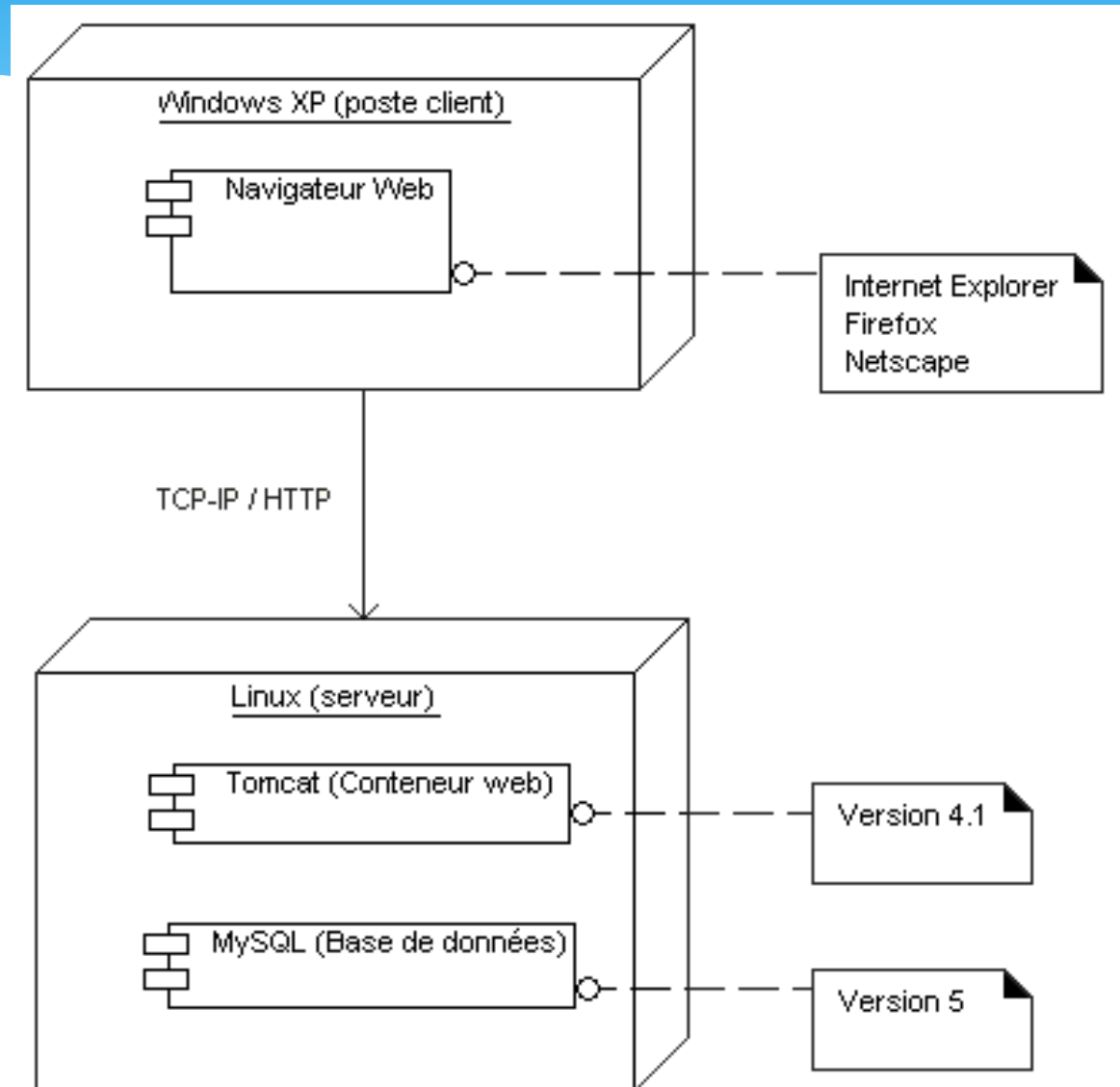
# Diagramme de déploiement

- \* Diagramme de déploiement d'un système d'inscription



# Diagramme de déploiement

\* Exemple :



# 5. Les outils de modélisation UML

- \* Plusieurs logiciels, citons
    - \* Rational Rose, IBM
    - \* Together, Borland
    - \* Rhapsody Modeler, I-Logix
    - \* Win Design
    - \* Visio, Microsoft
    - \* Poseidon UML , Gentleware
    - \* PowerAMC/PowerDesigner , Sybase
    - \* Plugin Omondo , Eclipse
    - \* ArgoUML
    - \* Visual Paradigm
- Et tous les autres....



# 6. Etude préalable avec UML

- \* L'avant projet « Inscription »
  - \* objectif
  - \* coût
  - \* rentabilité
  - \* abandon ou poursuite
- \* Commençons l'étude préalable
  - \* Recensement des **cas d'utilisation**
  - \* Recensement des termes, des concepts, des objets: **diagrammes de classes**
  - \* Les processus et leurs acteurs : **diagrammes d'utilisation et d'interaction (séquence et collaboration)**

# Séquence de création des diagrammes

Diagramme	étape du cycle en V
1. Diagramme de cas d'utilisation	Spécification, cahier des charges
2. Diagramme de séquence	
3. Diagramme d'activité (processus métiers)	
4. Diagramme d'activité (cinématique et/ou processus applicatifs)	
5. Diagramme de classe	Conception Architecturale
6. Diagramme d'objet	
7. Diagramme de communication	
8. Diagramme de déploiement	
9. Diagramme de composant	

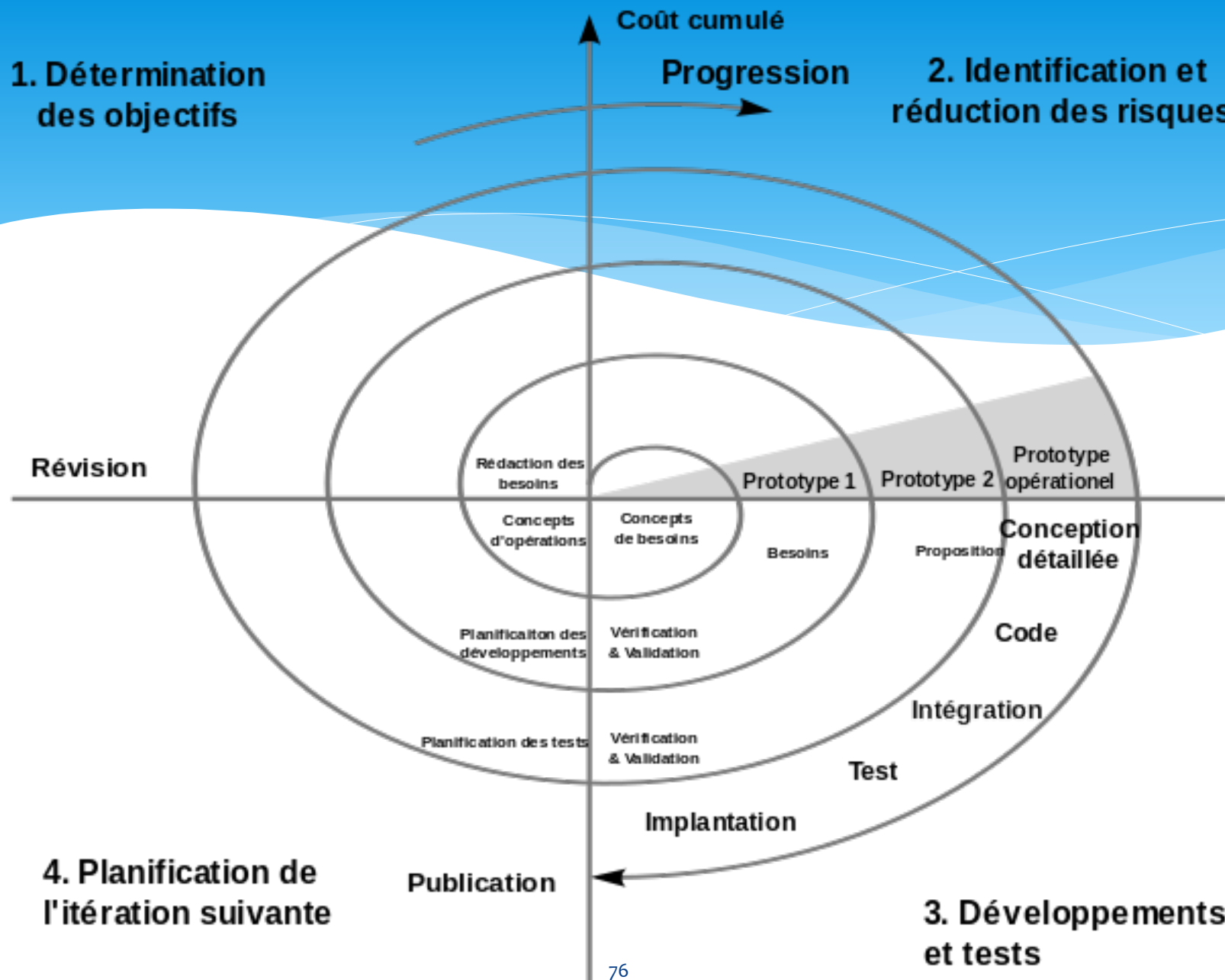
# Construction du modèle

- \* Le modèle général : ensemble des diagrammes
  - \* Cohérence: détection des incohérences, incomplétudes
  - \* Ordre précis d'exécution
- \* Évolution du modèle par suite d'itérations et incrémentations des cas d'utilisations
- \* Avant de passer à la phase d'analyse informatique, les besoins doivent être tous recensés

« Développement en spirale »

**1. Détermination  
des objectifs**

**2. Identification et  
réduction des risques**



# Cas d'utilisation

- \* Cas d'utilisation:
  - \* Les scenari d'utilisation du système « inscription » par les utilisateurs
  - \* Description de l'utilisation
    - \* Textuelle : phrases ordonnées des opérations
    - \* Graphique: diagramme d'enchaînement des opérations
    - \* Tableau: une ligne/ rubrique caractérisant le cas
- \* Éléments décrits:
  - \* Les acteurs
  - \* Le scénario
  - \* Les pré-conditions et garanties
  - \* Les exceptions, les extensions, les utilisations d'autres cas

# Cas d'utilisation

## Description textuelle : cas d'utilisation « inscription »

1. L'auditeur se présente à la scolarité avec son plan de formation
2. L'agent ouvre un dossier d'inscription
3. L'auditeur donne son identité et le mode de financement
4. les sessions de formations sont saisies
5. Les tarifs sont appliqués en fonction du mode de financement et des sessions à inscrire
6. Un devis est établi
7. Le régisseur encaisse le montant selon le devis
8. La carte d'auditeur est éditée

# Cas d'utilisation

Alternative : Tiers payeur

6a. Un tiers payeur est identifié, une convention est établie.

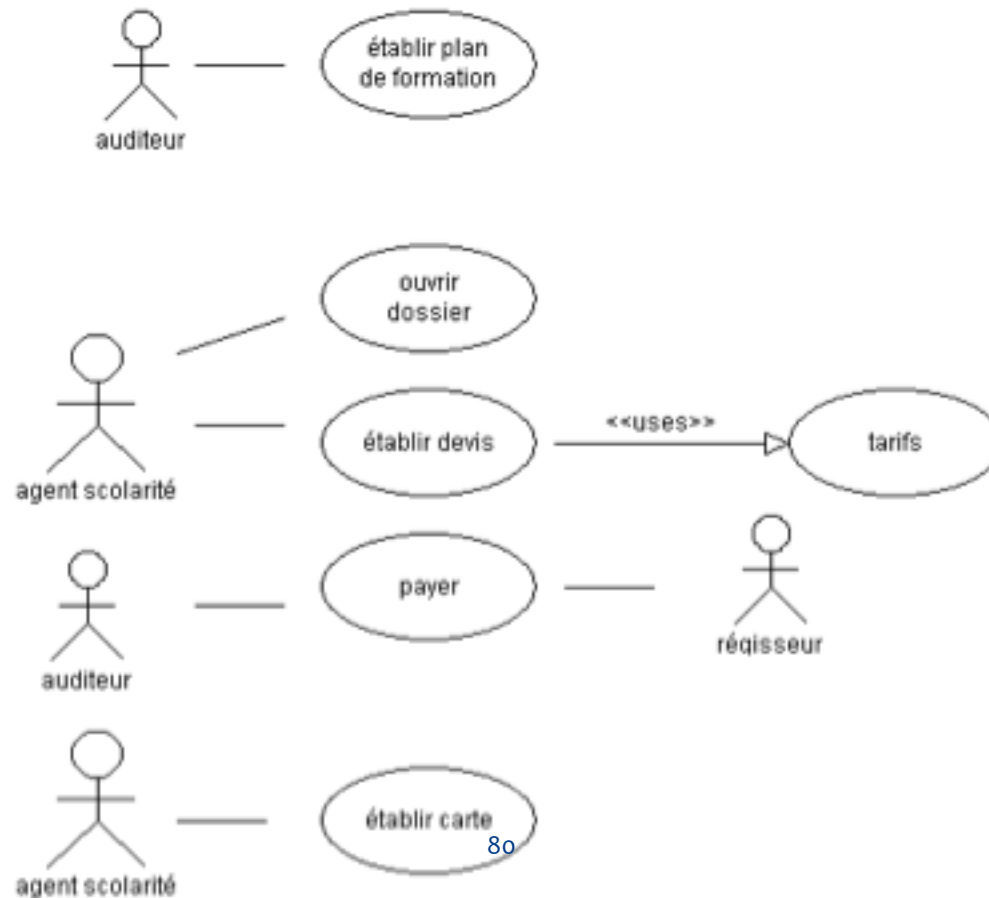
7a. Pas d'encaissement. Retour de la convention signée.

8a. La carte d'auditeur est éditée

9a. L'agent comptable assure le recouvrement de la somme forfaitaire.

# Cas d'utilisation

- \* Description graphique : cas d'utilisation « inscription »





# Cas d'utilisation

## \* Tableau : cas d'utilisation « inscription »

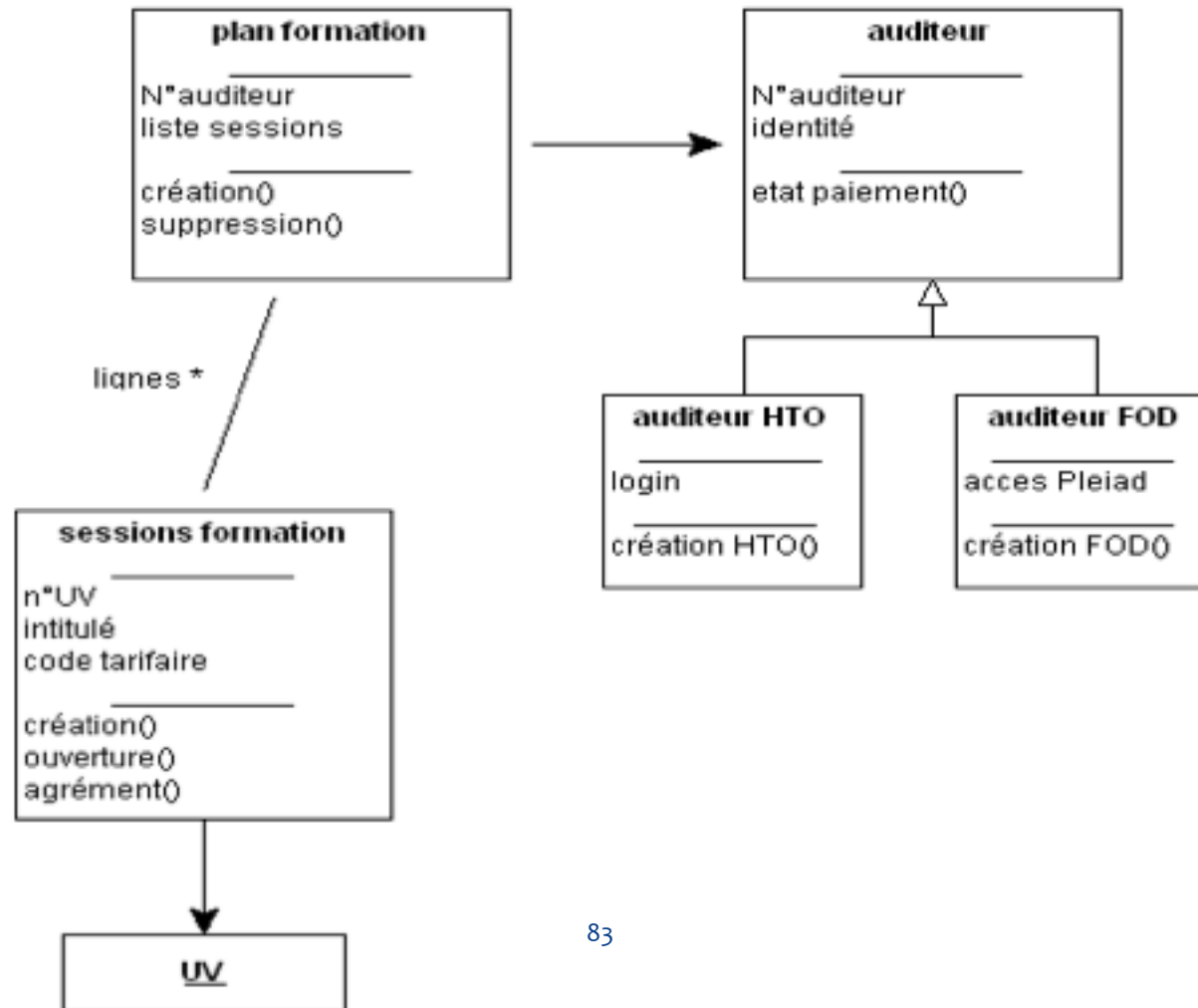
Cas	Inscription d'un auditeur
Définition	Inscription d'un auditeur et délivrance de la carte
Acteur principal	auditeur
Acteurs secondaires	Agent scolarité, régisseur
Pré conditions	Plan de formation de l'auditeur Base de l'offre de formation et tarifs
Garanties	Paieement et édition de la carte de l'auditeur
Scénario	Plan de formation pour création dossier auditeur, devis selon les tarifs, règlement auprès du régisseur et édition carte de l'auditeur
Exception	Le financement est assuré par un tiers payeur. Une convention est établie. Après signature, la carte est délivrée. Le recouvrement est suivi par l'agent comptable.

# Diagramme de classe

- Description générale des types d'objets et leurs relations
- Avec
  - leurs attributs principaux
  - les opérations principales

# Diagramme de classe

## \* Diagramme de classe: cas inscription

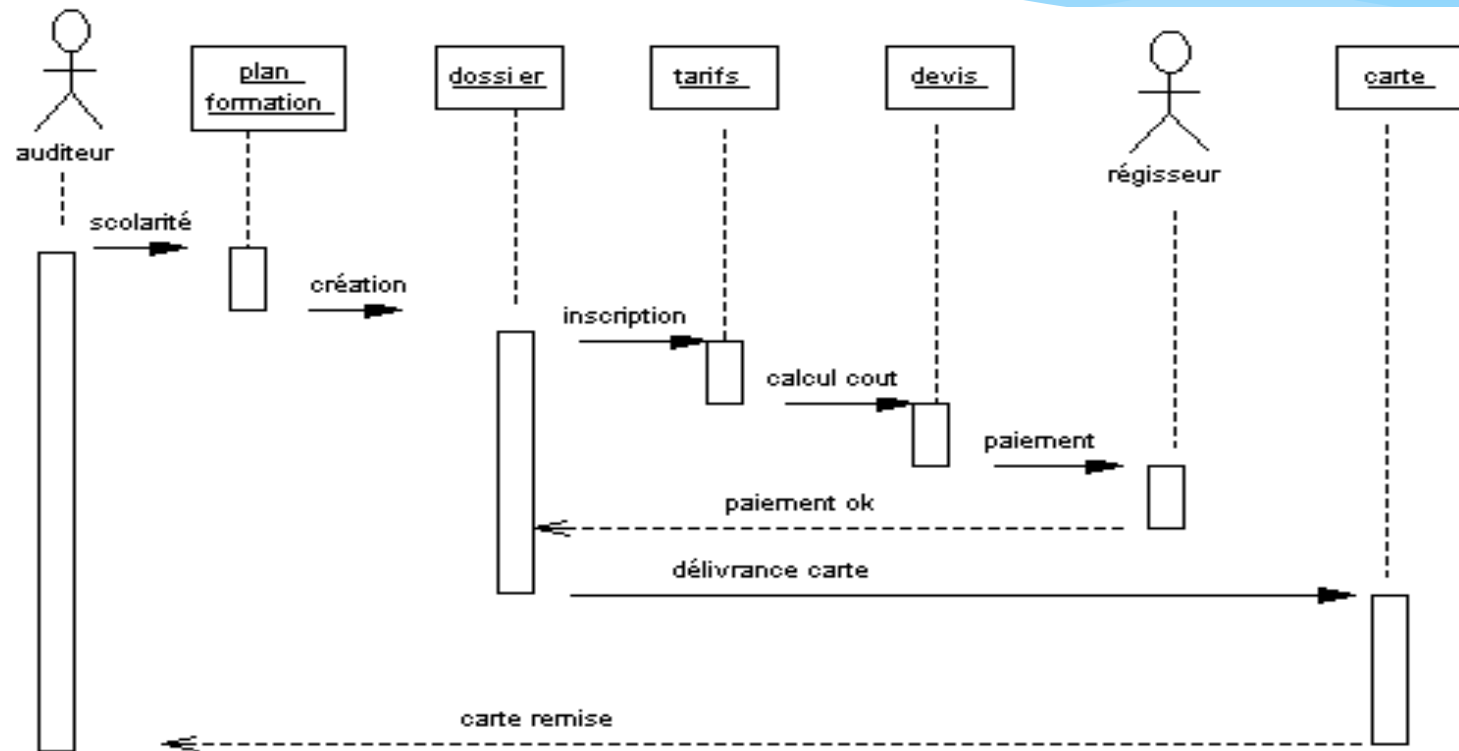


# Diagramme d'interaction

- \* Description d'un comportement donné d'objets
- \* Un diagramme d'interaction = un cas d'utilisation
- \* Transmission des messages entre objets
- \* Deux diagrammes d'interaction
  - \* De séquence
  - \* De collaboration

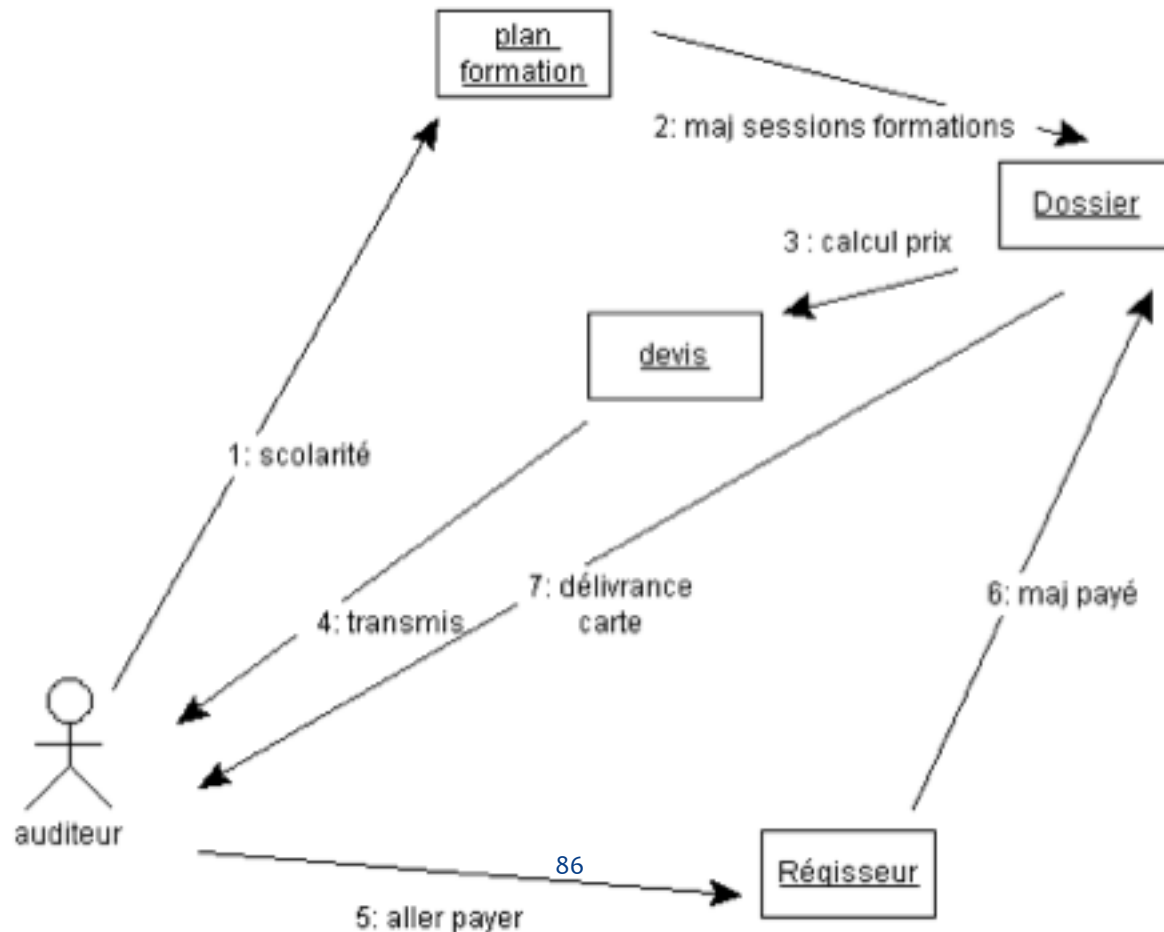
# Diagramme de séquence

## \* Diagramme de séquence: cas inscription



# Diagramme de collaboration

- \* Diagramme de collaboration: cas inscription

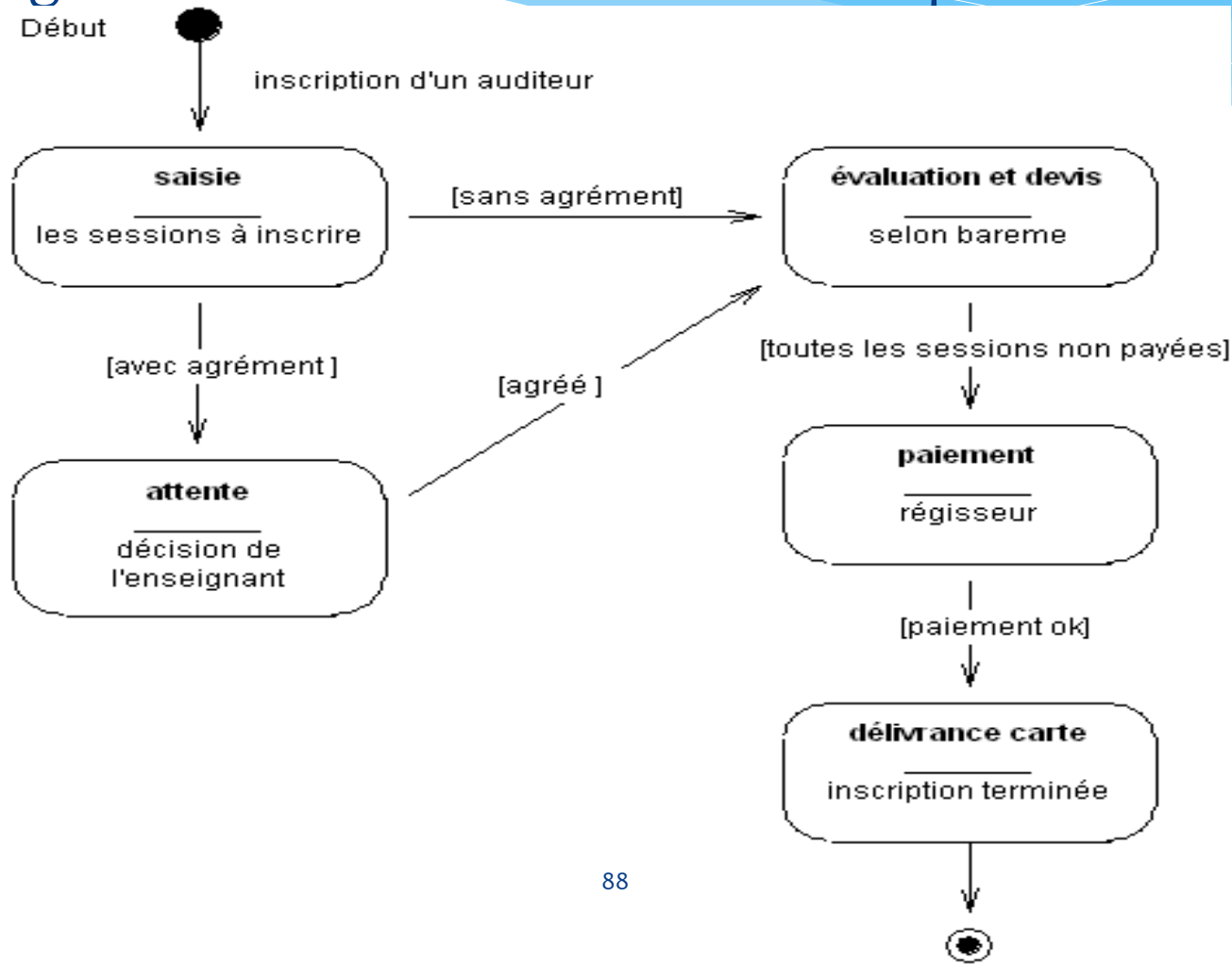


# Diagramme état transition

- Comportement d'un système
- Etats possibles d'un objet
- Changements des états / événements: transitions

# Diagramme état transition

## \* Diagramme états-transitions: cas inscription



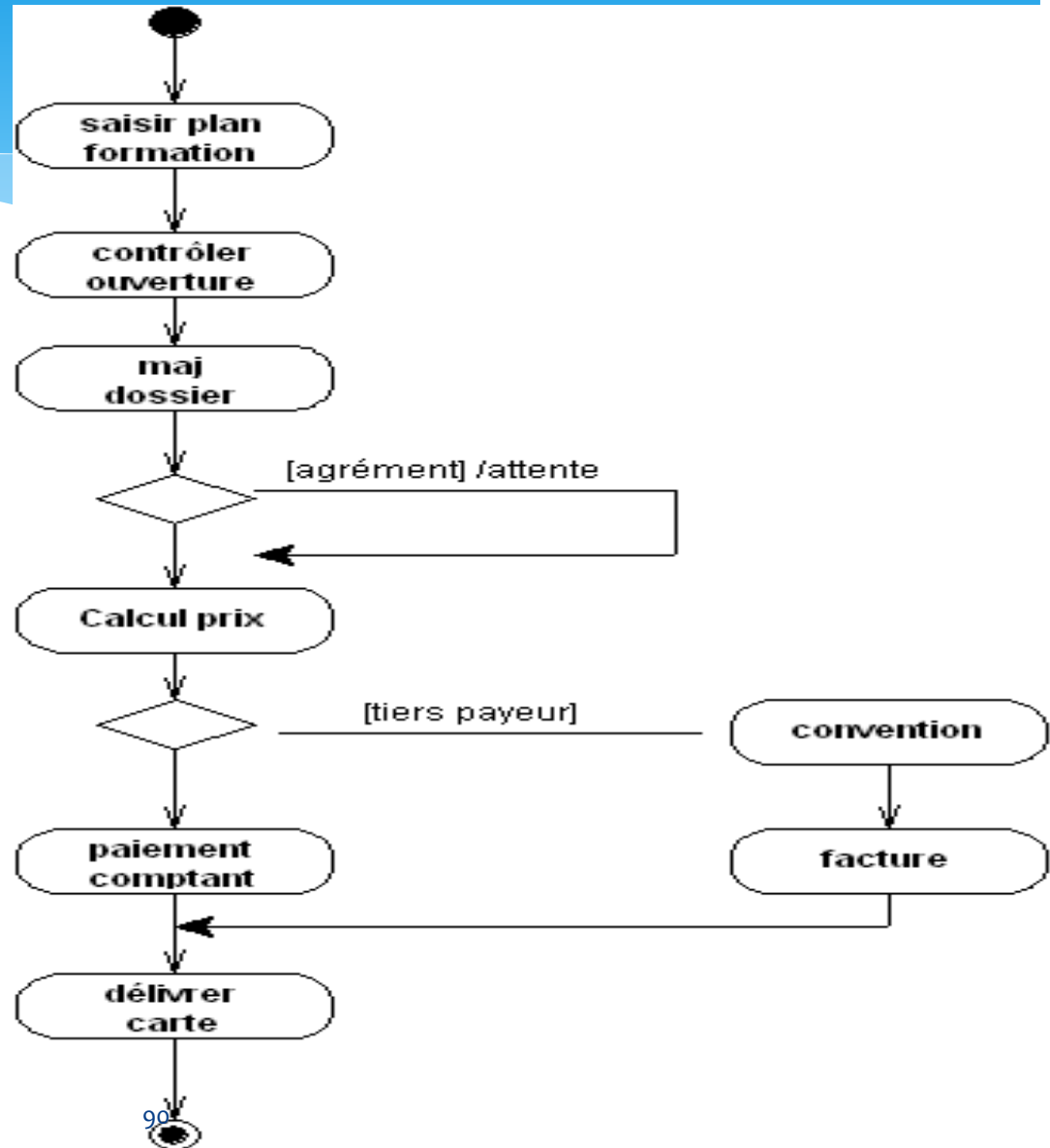


# Diagramme d'activité

- Organisation des activités :
  - Activités en séquence
  - Activités parallèles
  - Activités conditionnelles
- Une activité peut être composite
- Quand ?
  - Analyse cas d'utilisation,
  - modélisation métier,
  - Algorithme complexe

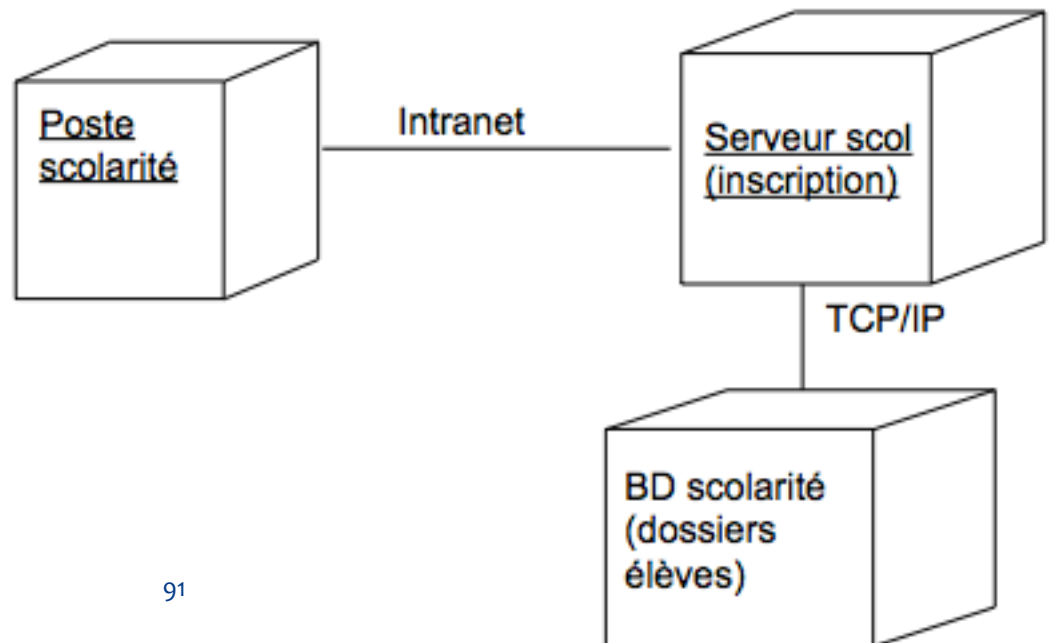
# Diagramme d'activité

- \* Diagramme d'activités :
- \* cas inscription



# Diagramme de déploiement

- \* On peut représenter l'architecture logicielle et matérielle du système avec un diagramme de déploiement



# Les autres étapes de construction du système

## Informatisation du système « inscription »

- \* on introduit les composants informatiques
  - \* on construit les modules
  - \* on précise les méthodes associées aux classes
  - \* on intègre les contraintes techniques
  - \* on propose un modèle de données
- 
- \* Toutes les informations ont été recensées lors de l'étude préalable. Il s'agit de proposer un modèle informatique pour le système « Inscription »
  - \* A chaque étape suivante on affine le système en utilisant les concepts UML
  - \* On utilise les diagrammes si nécessaires

# Conclusion

- UML une aide à toutes les étapes de conception du projet
- Avantages
  - Descriptions graphiques
  - Vues différentes à des étapes différentes
  - Recoupement des descriptions
    - Incohérences
    - Incomplétudes mises en évidence
  - Adaptation facile aux méthodes
  - Projet : un bon outil de démarrage du projet
- S'apprend par la pratique

# Exercice

## **Diagramme de classe** d'une application de gestion d'hôtel : I

Un hôtel est constitué d'un nombre de chambres. Un responsable de l'hôtel gère la location des chambres. Chaque chambre se loue à un prix donné.

L'accès aux salles de bain est compris dans le prix de la location d'une chambre. Certaines chambres comportent une salle de bain, mais pas toutes. Les hôtes de chambres sans salle de bain peuvent utiliser une salle de bain sur le palier. Ces dernières peuvent être utilisées par plusieurs hôtes.

Des personnes peuvent louer une ou plusieurs chambres de l'hôtel, afin d'y résider. En d'autre termes : l'hôtel héberge un certain nombre de personnes, ses hôtes (il s'agit des personnes qui louent au moins une chambre de l'hôtel...).

# Exercice

## \* Solution :

