

# **Éléments d'UML**

## **pour le projet**

### **(Unified Modeling Language)**

# PLAN

1. Introduction
2. Préliminaires
3. Les règles UML
4. Les diagrammes UML
5. Outils de modélisation UML
6. L'étude préalable avec UML
7. Conclusion

# 1- Introduction

- UML: langage de modélisation
- Méta-modèle UML
  - définit la structure des modèles UML
  - permet la description du modèle concerné par l'application.
  - une notation UML avec des éléments de la notation extensibles à condition d'en définir la sémantique
- Construction de modèles objets ou autres
- Utilisation de la notation graphique
  - une solution visuelle
  - limite les ambiguïtés
  - indépendance par rapport aux langages

# Exemple

- > métamodèle      -> Classe, Attribut, Opération
- > modèle            -> Fournisseur  
Identification, nomFournisseur, adresseFournisseur  
commander()
- > objet              -> « Dupont »  
« 40222 », « ChipsAndChips », « 13 rue Parmentier »  
« commander(12005A,13) »

## 2- Préliminaires

- Les origines d'UML
- La démarche de conception et d'analyse
- UP: Processus unifié

# Origines d'UML

- Issu en 1996 de la pratique industrielle et de la modélisation des systèmes logiciels.
- Unification des méthodes objets de J-B-R
  - Ivar Jacobson (OOSE)
  - Grady Booch (BOOCH'93) ,
  - James Rumbaugh (OMT) et
- Normalisation OMG en 1997. En 2007: UML 2.1.2
- Méthodes
  - **Fonctionnelles** : années 60  
Inspirée de l'architecture des ordinateurs  
études des fonctions en séparant les données du code.
  - **Objets** : années 80  
Modélisation objet avec composition et décomposition  
des objets ayant des propriétés et des comportements
  - **Méthodes** qui couvrent le cycle de vie d'un logiciel.
- **UML : de nouvelles techniques sans rejeter les méthodes existantes**

# Démarche de conception et d'analyse

- Analyse du pb: processus unifié UP
  - 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 outils
- L'itération peut se faire à toutes les phases
  - étude préalable
  - construction
  - tests et mise au point

# Processus unifié

- Langage de modélisation UML + Processus unifié UP
    - 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
      - Principe d'itérations et incrémentations
      - Évaluation des risques à toutes les étapes
- « on part des cas d'utilisations connus, on construit un premier modèle d'architecture; on complète et affine par itérations et incrémentations et on évalue par étape les risques pour faire les meilleurs choix »



- Les utilisateurs décrivent les cas d'utilisation
  - 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 complétant et affinant l'étude
  - Ajouts, compléments, détails des cas d'utilisation
  - Description plus détaillée des composants
  - Ajustement de l'architecture
- Utilisation de maquettes et prototypes

- Evaluation permanente du système en terme de bon choix : le bon produit, une bonne construction, le bon prix, les bonnes performances....
  - Évolution
  - Amélioration
  - Validation ou rejet des solutions
  - Objectif: minimiser les risques au fur et à mesure de la spirale de développement

- Phases du processus UP:
  - Étude d'opportunité
    - Mesures des risques
    - Définitions des limites
    - Construction d'une maquette des premiers cas d'utilisation
    - Décision
  - Réalisation
    - Première version
    - Proposition d'architecture, développements, tests
    - Rentabilité: décision
    - Puis processus incrémental et itératif jusqu'au produit final
  - Mise en exploitation

- Les activités dans les phases:

- Expression des besoins
- Analyse
- Conception
- Implémentation
- Tests

« les activités sont celles des méthodes connues mais ces activités se déroulent selon les phases UP »

RUP: Rational Unified Process.

Version UP de la société Rational Software

## 3- les règles UML

- Développement orienté objet
- UML langage de modélisation
  - Règles d'écriture et de représentation graphiques normalisées
  - Neuf diagrammes (UML 2.1.2: 13 diagrammes )
- Méta-modèle des concepts et notations des diagrammes
  - Construire les outils de modélisation selon les règles UML et adaptés à l'étude
  - Règles
    - Stéréotypes;
    - Notes;
    - Contraintes;
    - règles d'écriture des noms et expressions: nom, étiquette valeur d'un composant;
    - Paquetage.

- Stéréotypes
  - Adaptation du modèle aux éléments de l'application
  - Nouveau type d'élément défini depuis un type du modèle
  - Application principale aux classes
  - Distinction d'utilisation entre guillemetsEx: classe Client stéréotypée « clientA »
- Notes
  - Commentaires d'un élément UML

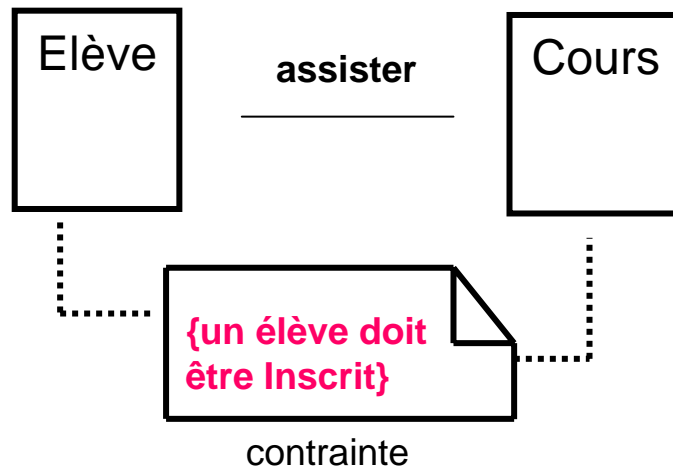


stéréotype



commentaire

- Contrainte
  - Note sémantique pour un élément
  - Écriture entre { }
  - Aussi langage OCL Objet Constraint Language d'UML
- Écriture des noms et des expressions
  - Nom: identifiant d'un élément, chaîne de caractères
  - Expression: valeur



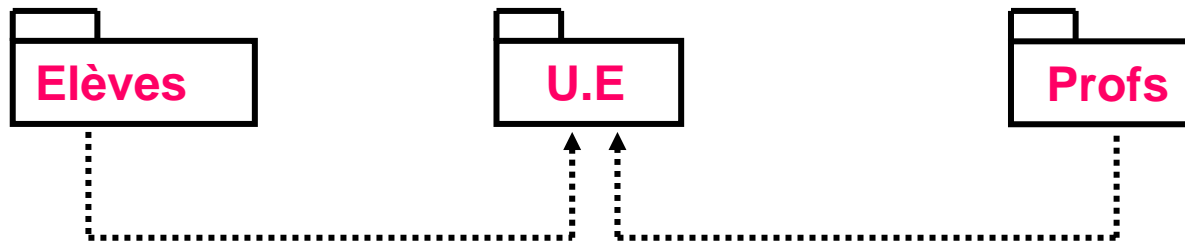
**noms**

NomEleve  
Cycle.UE

**expressions**

After (7 minutes)  
Date = 7 juillet 2005

- Paquetage
  - Décomposition du système en paquetages
  - Ensemble logique d'éléments du modèle
  - Nommage du paquetage
  - Relations entre paquetages





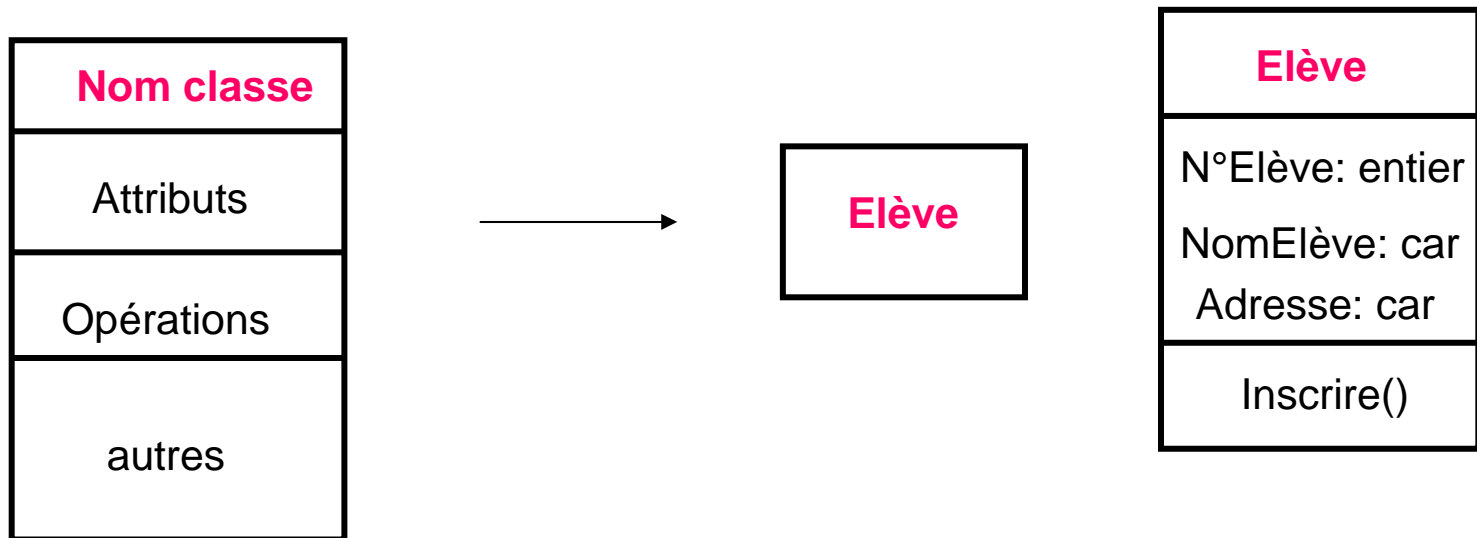
## 4- Les principaux diagrammes UML

- Diagramme des cas d'utilisation
  - Besoins des utilisateurs
- Diagramme de classes
  - Description statique des données et des traitements
- Diagrammes d'objets
  - Instances des classes
- Diagramme états-transitions
  - États des objets selon les événements
- Diagramme d'activités
  - Vue des enchaînements des activités d'un cas d'utilisation ou d'une opération

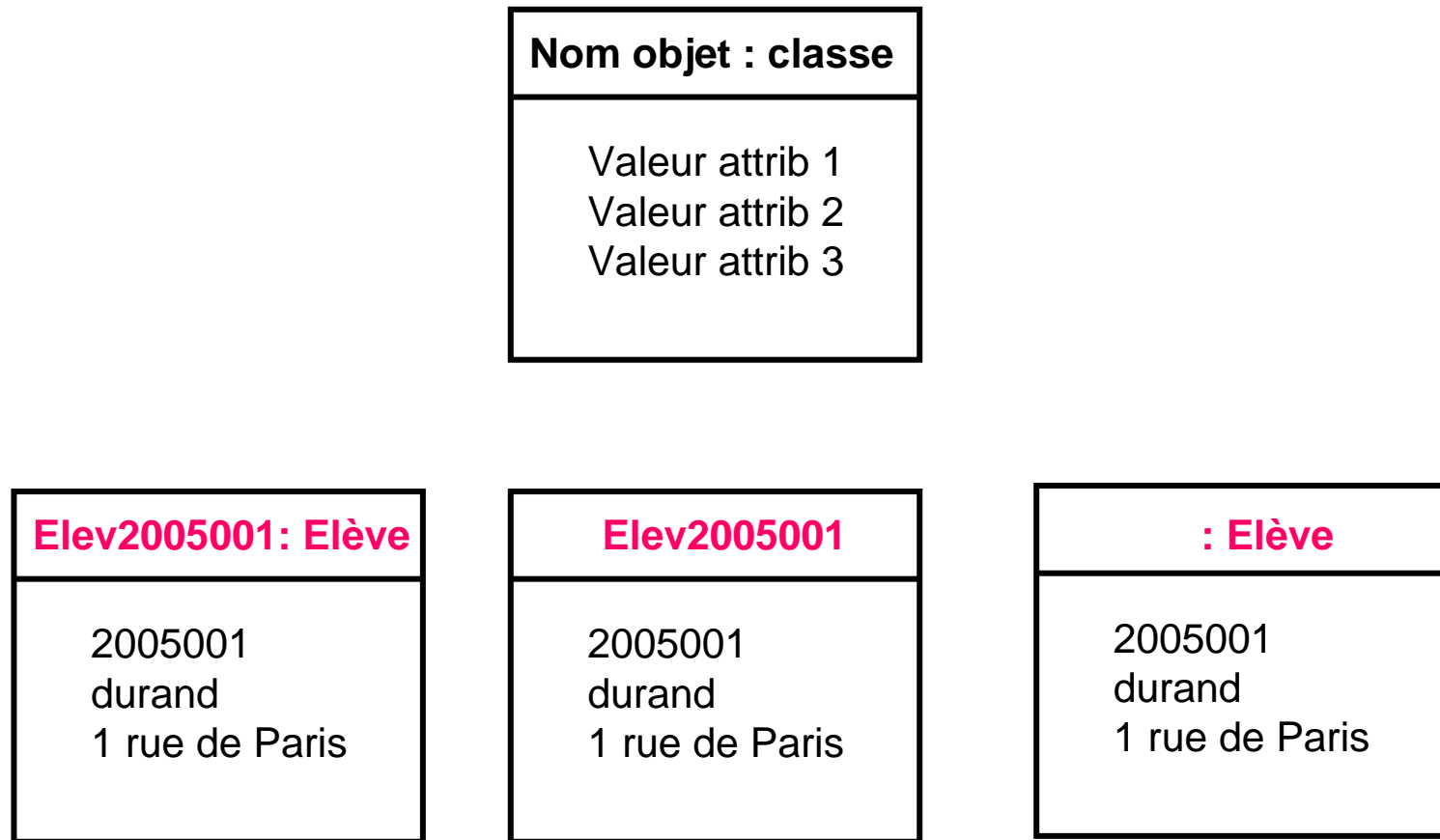
- Diagramme de séquence
  - Scénario d'un cas d'utilisation : chronologie des opérations
- Diagramme de collaboration
  - Scénario d'un cas d'utilisation: activités des objets et des messages échangés
- Diagramme des composants
  - Représentation des composants logiciels d'un système
- Diagramme de déploiement
  - Description de l'architecture technique du système

## 4.1. Diagramme de classe et d'objets

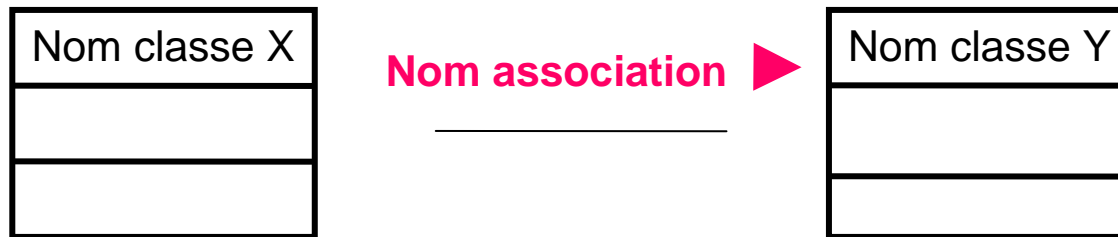
- les objets sont identifiés dans le système et portent un nom
- Les classes sont créées en regroupant les objets ayant les mêmes propriétés, mêmes comportements
- Un objet est une instance d'une classe
- Représentation UML d'une classe



- Représentation UML d'un objet



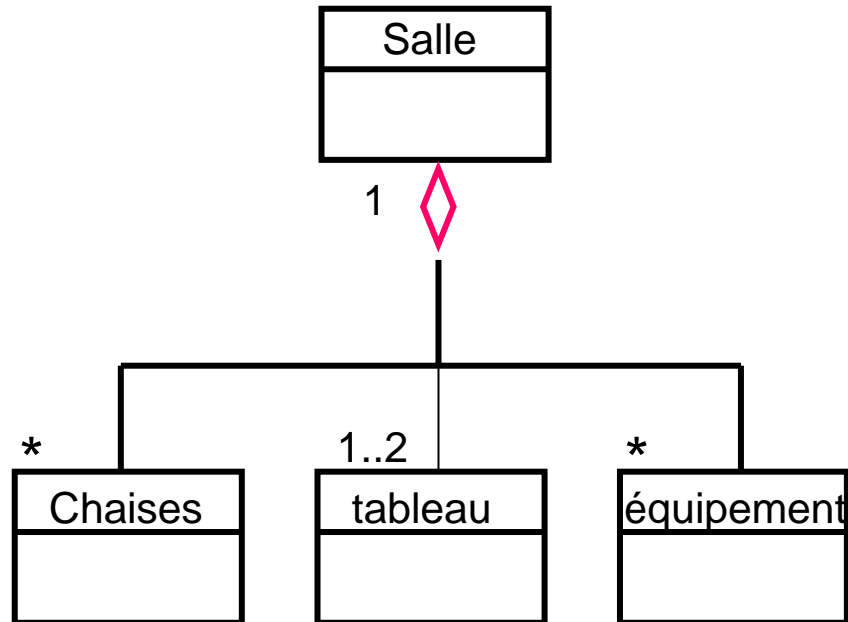
- Association entre classes
  - Liens entre les instances
  - Rôle de l'association et son sens
  - Cardinalité des instances associées



A une instance X correspond 1 à 2 instances de Y  
A une instance Y correspond 0 à n instances de X

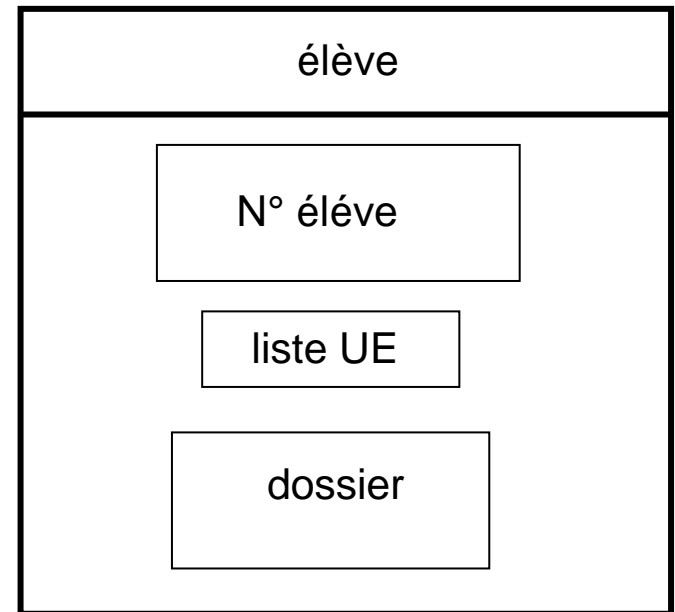
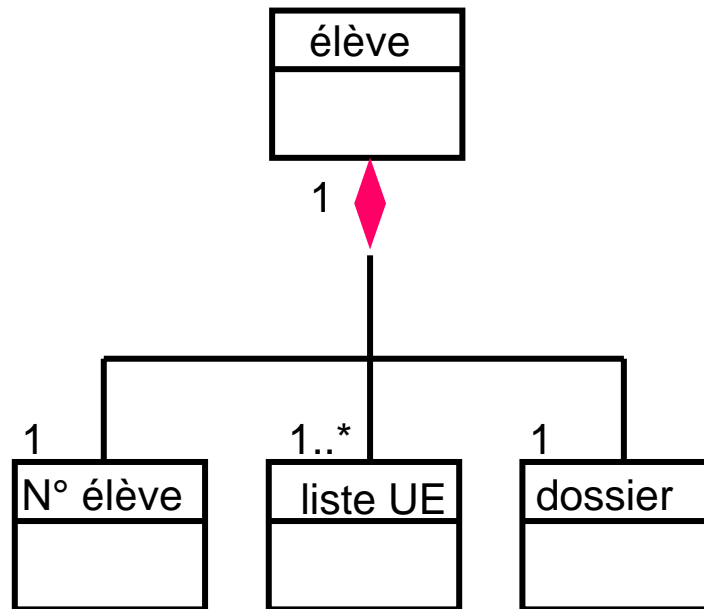
- Agrégation

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

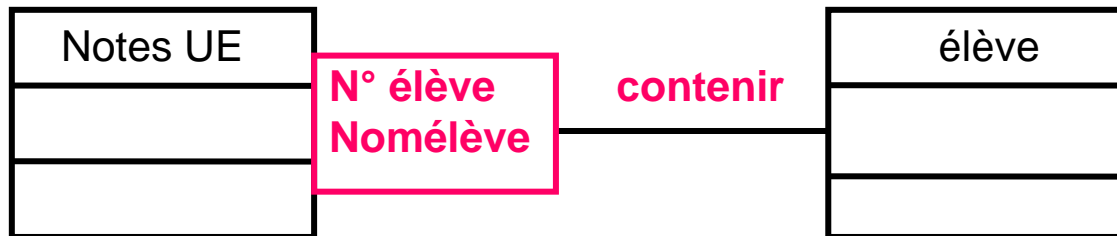


- Composition

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



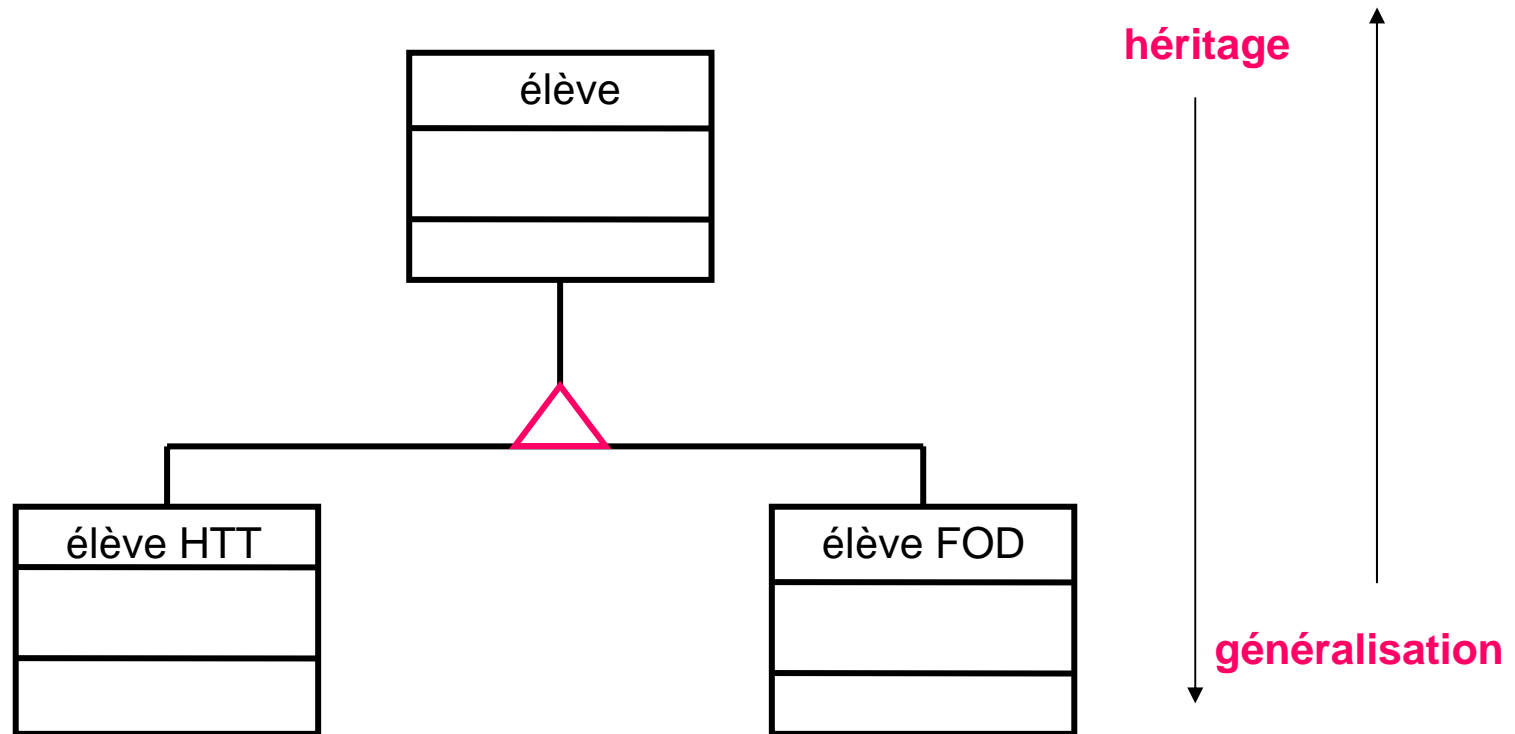
- Qualification d'une association
  - Sémantique d'une association entre deux classes
  - Restriction d'une association



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

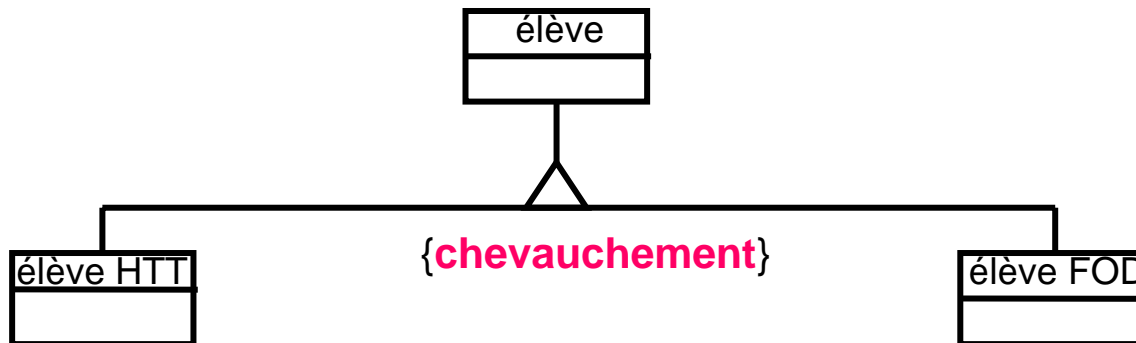


- Généralisation et héritage simple
  - Généralisation: création d'une superclasse à partir de classes
  - Héritage: création de sous classes à partir d'une classe

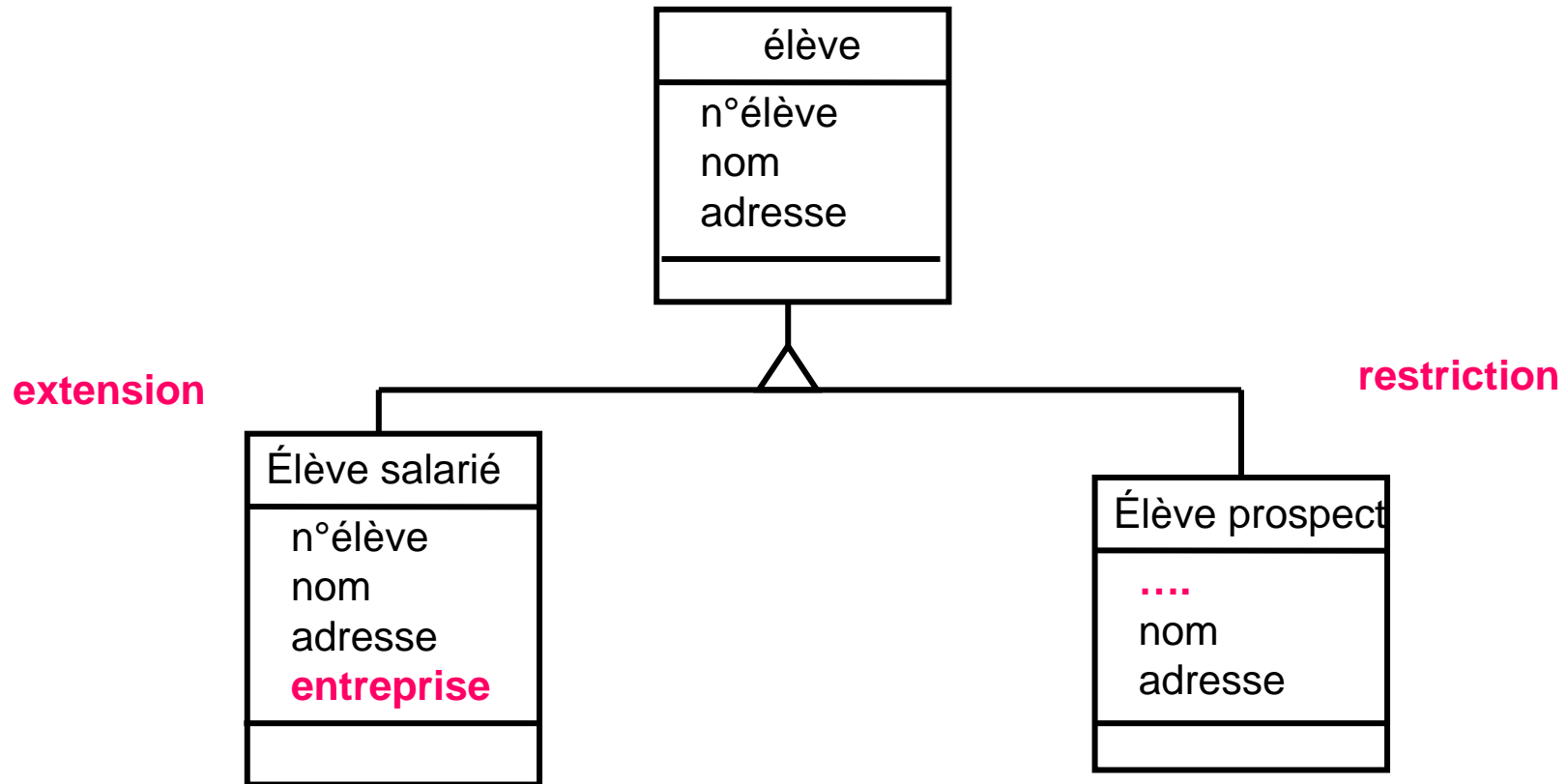


- Héritage avec recouvrement

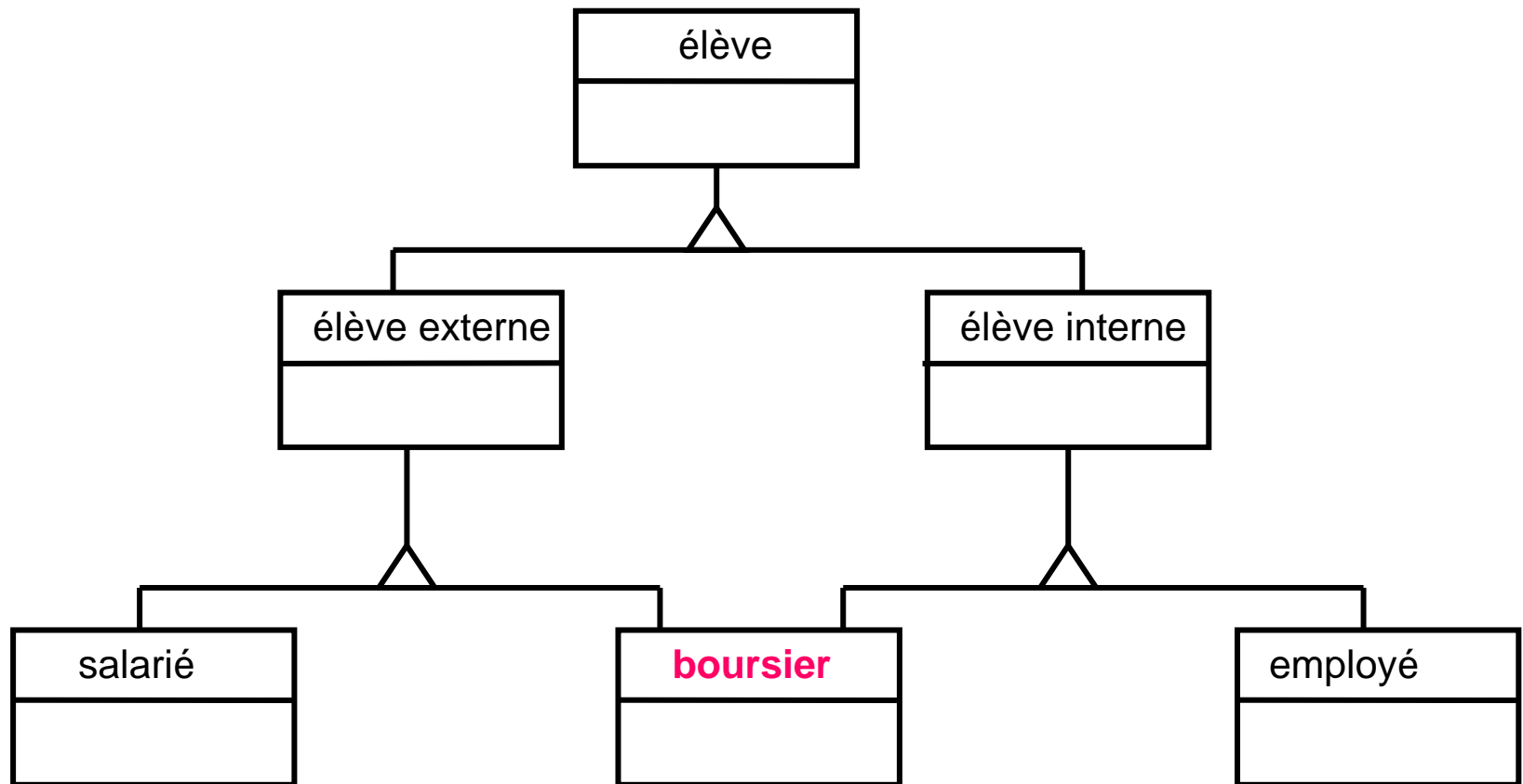
- Chevauchement : deux sous classes avec des instances identiques
- Disjoint : les instances d'une sous classe ne peuvent être dans une autre sous classe
- Complète : la généralisation ne peut être étendue
- Incomplète : la généralisation peut être étendue



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

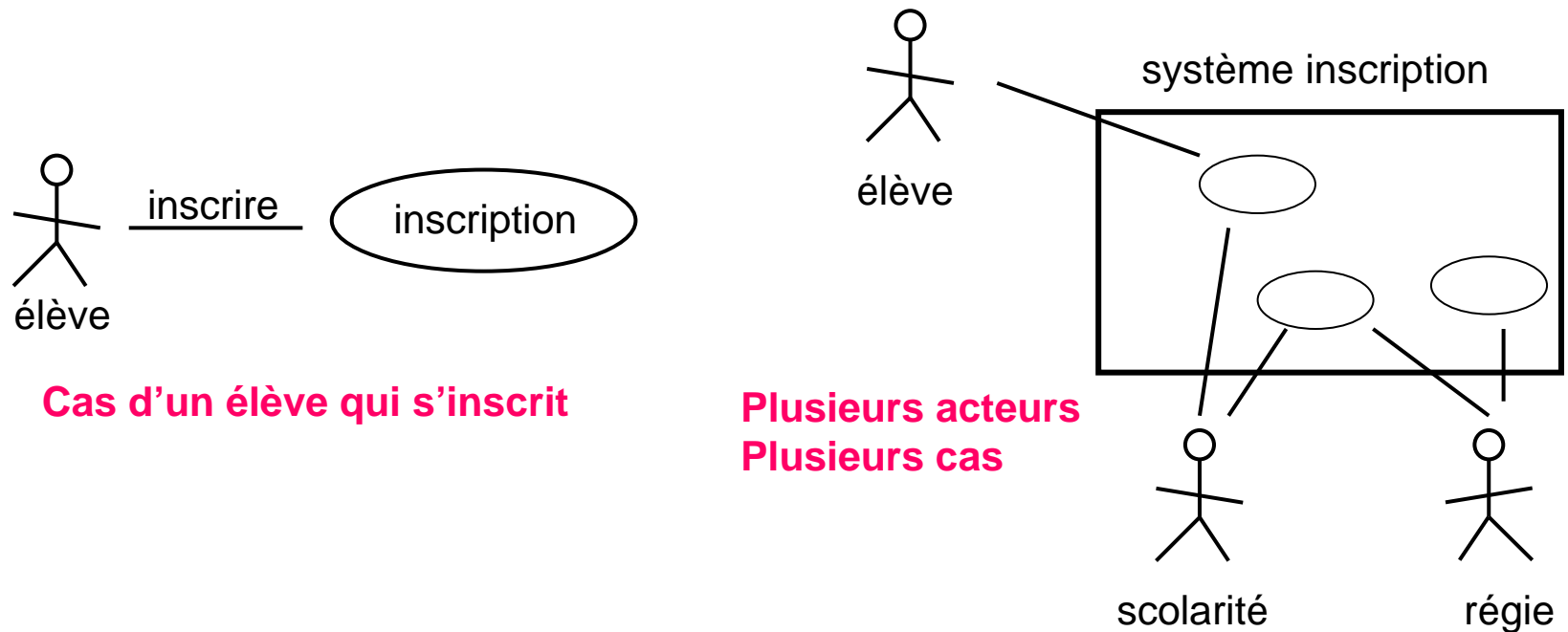


- Héritage multiple
  - Une classe hérite de deux classes parentes



## 4.2. 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

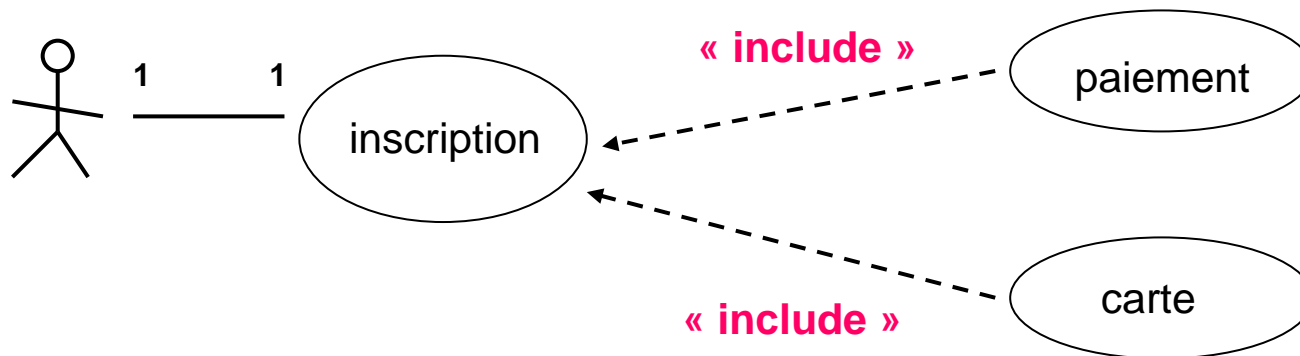


- Description textuelle

- 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

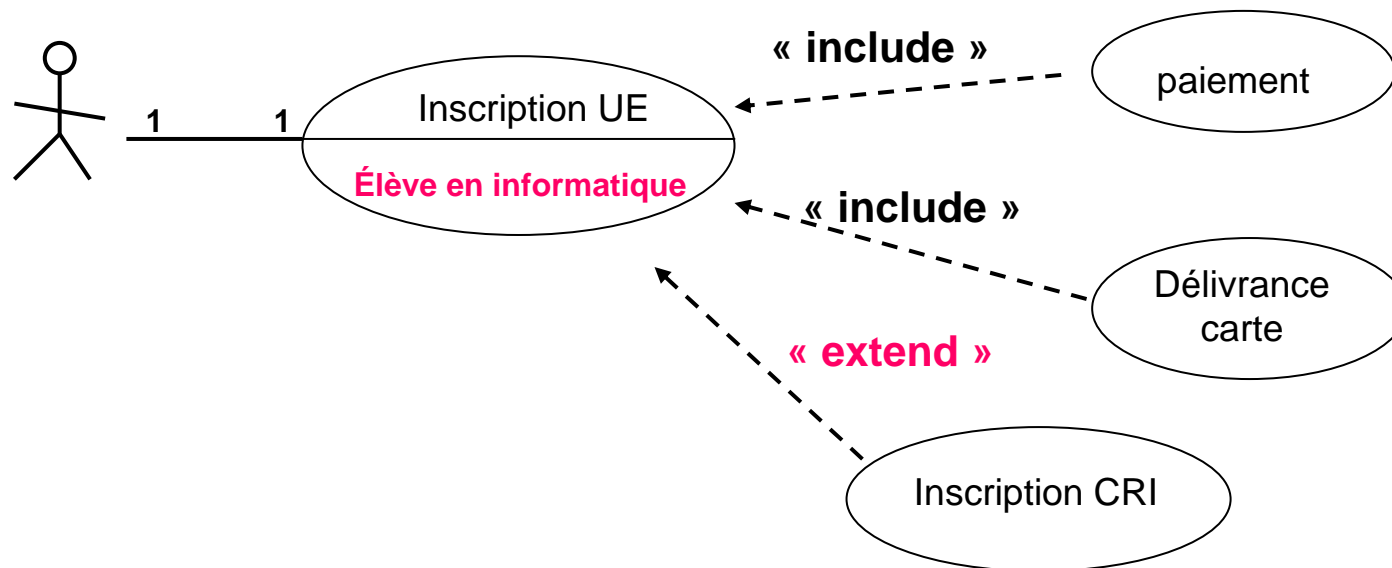
cas	Inscription d'un élève
résumé	Procédure d'inscription d'un élève jusqu'à la délivrance de sa carte
acteur primaire	L'élève
acteur secondaire	La scolarité, la régie
pré-conditions	Vérification préalable des conditions d'inscription
résultats	Dossier d'inscription, paiement, délivrance carte élève
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 de carte si 1 UE à agrément

- 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



- 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

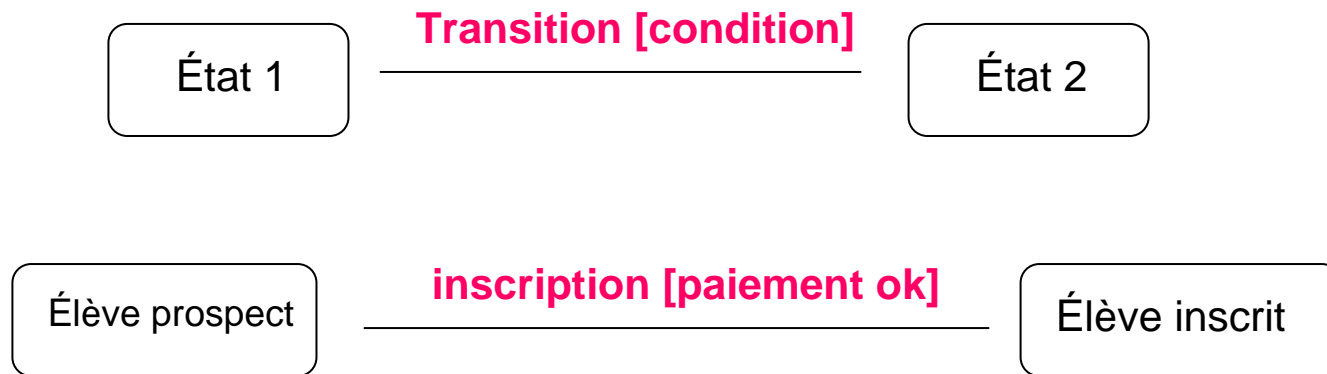




### 4.3. Diagramme états-transitions

- 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 »

Etat-Transition

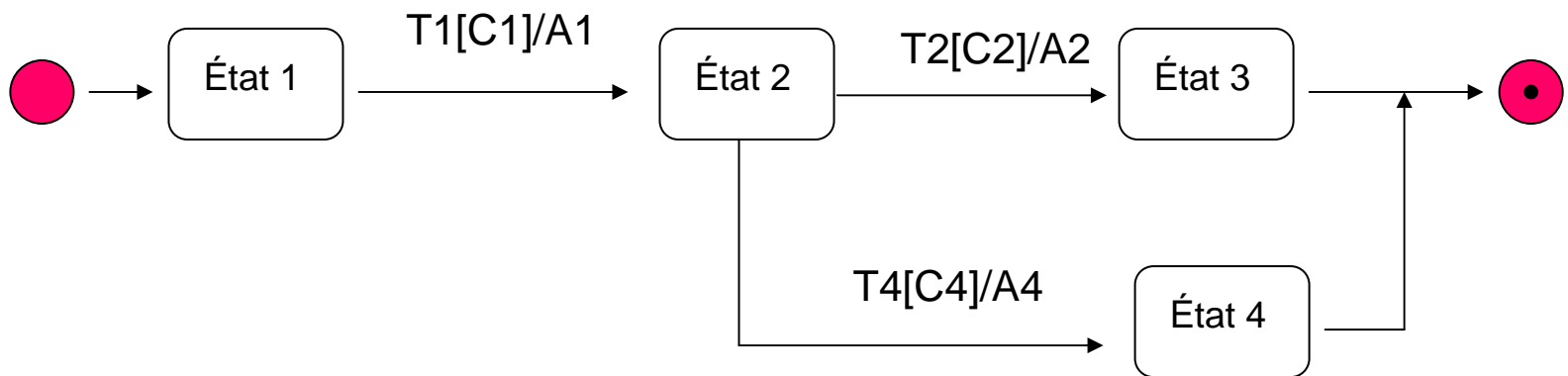


- On peut préciser une action ou activité à la transition

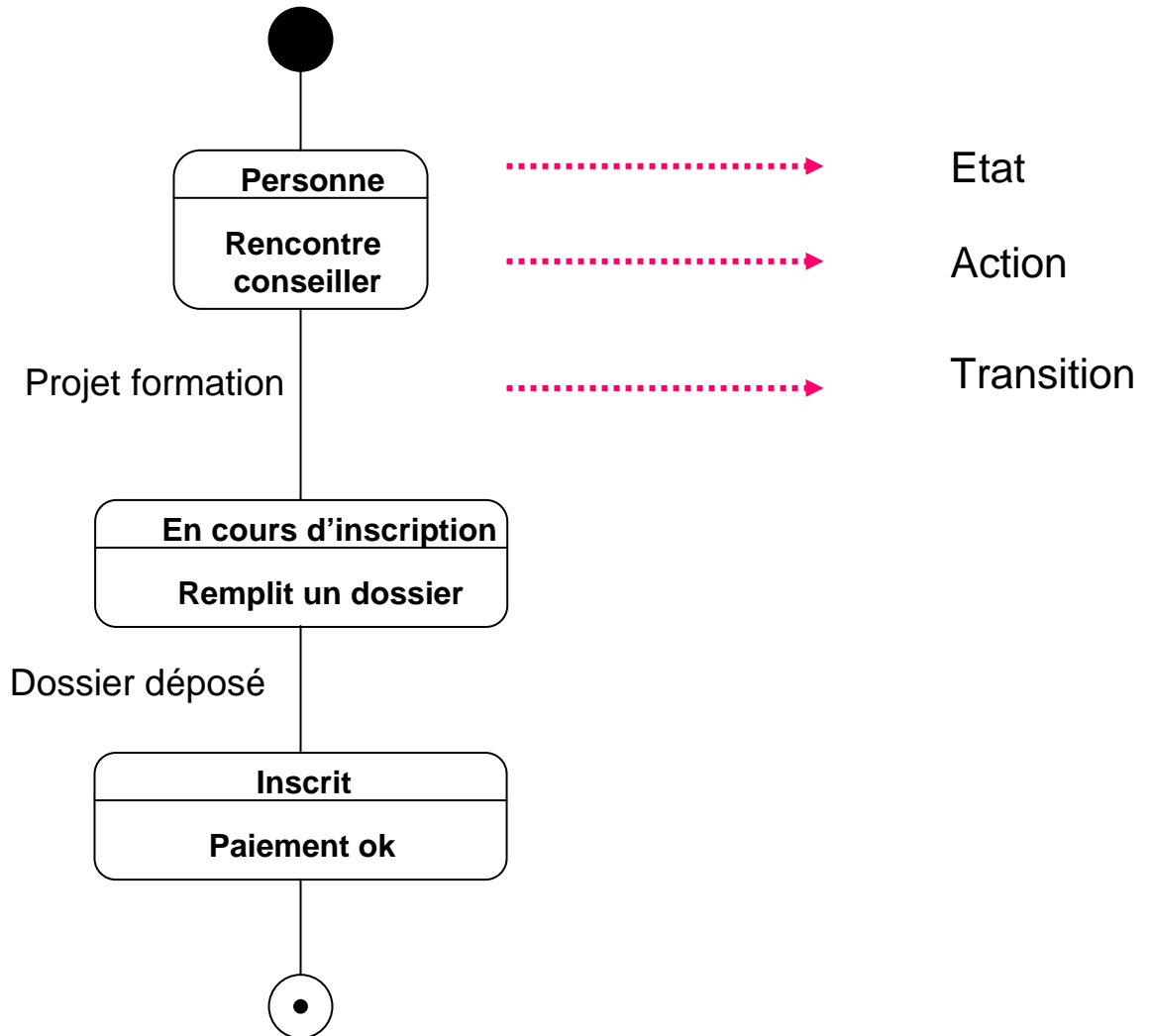


– Suite d'états et de transitions avec

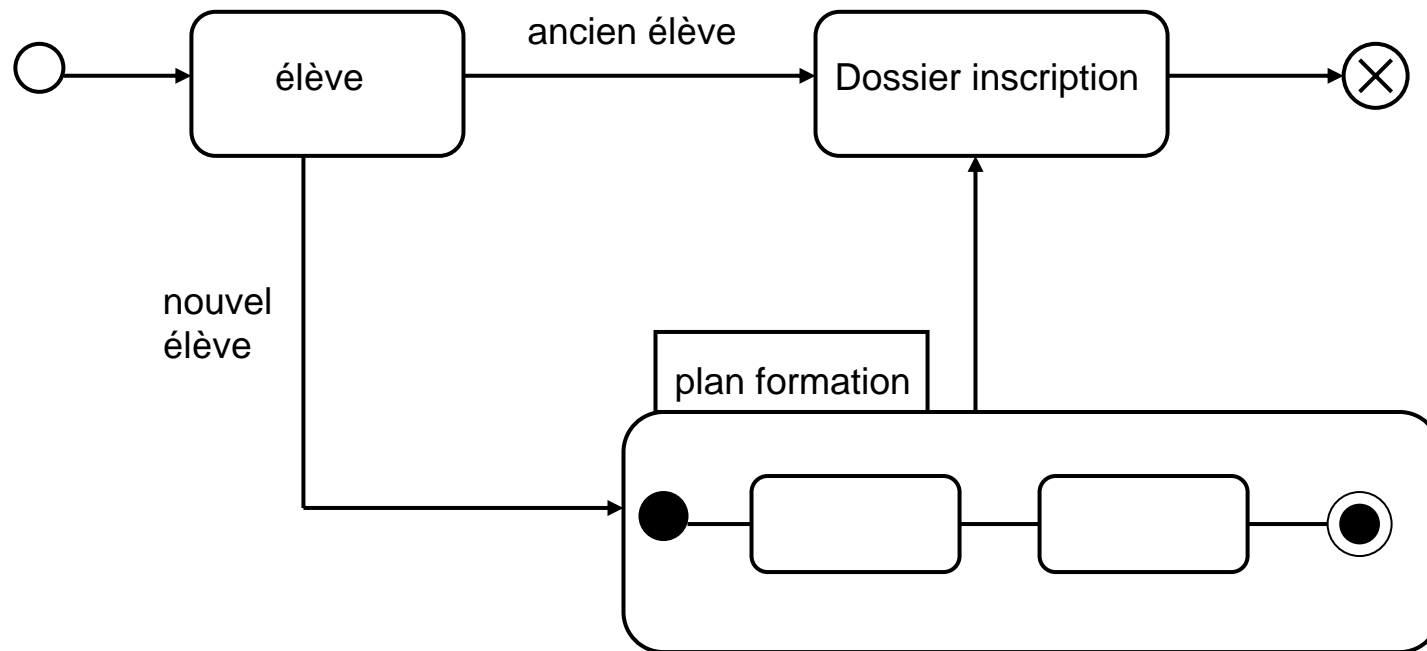
- Début: ●
- Fin: ⊙
- Ti : transition – Ci : condition – Ai : action



## – Exemple d'états-transitions



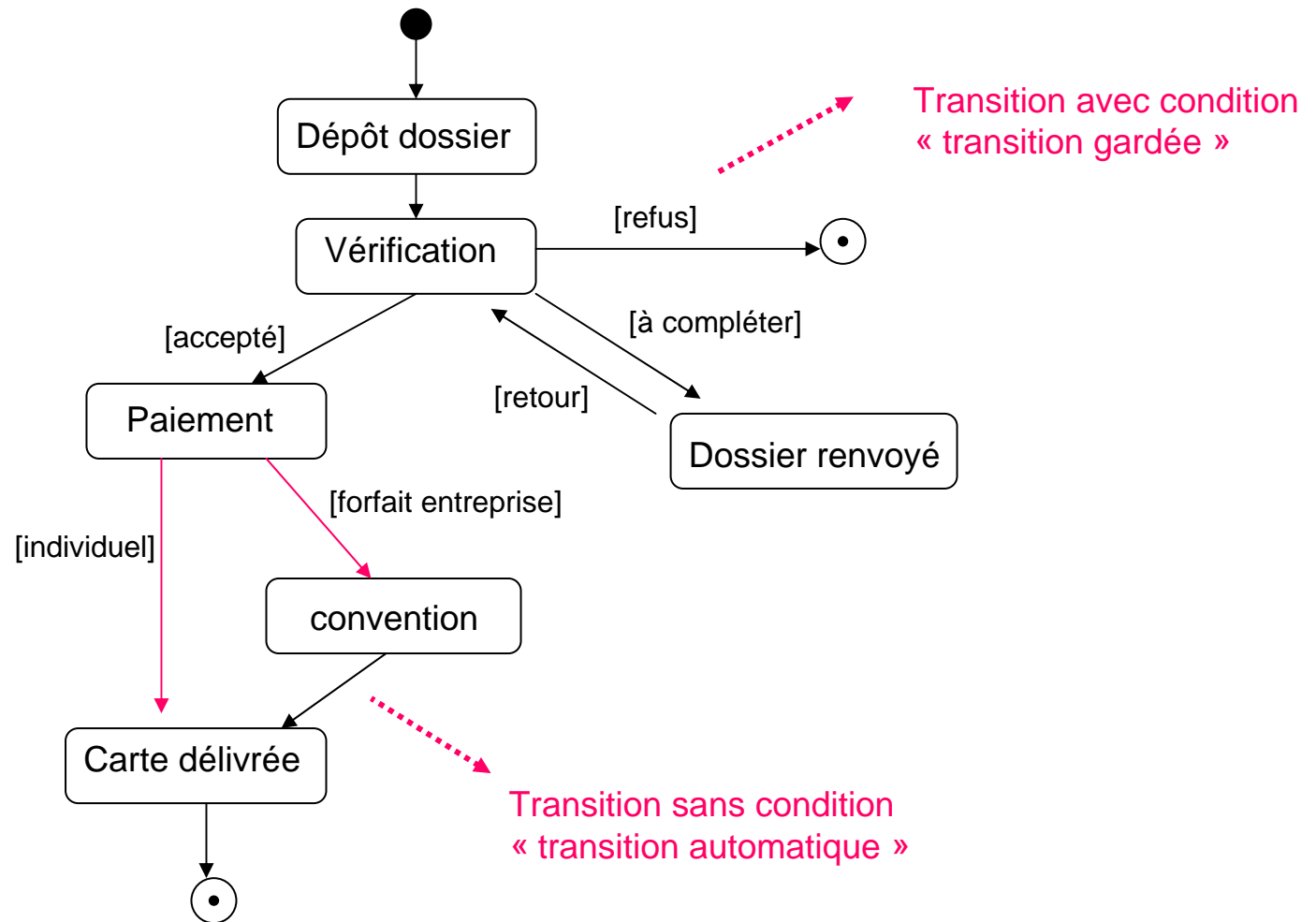
- Exemple de diagramme machine à états



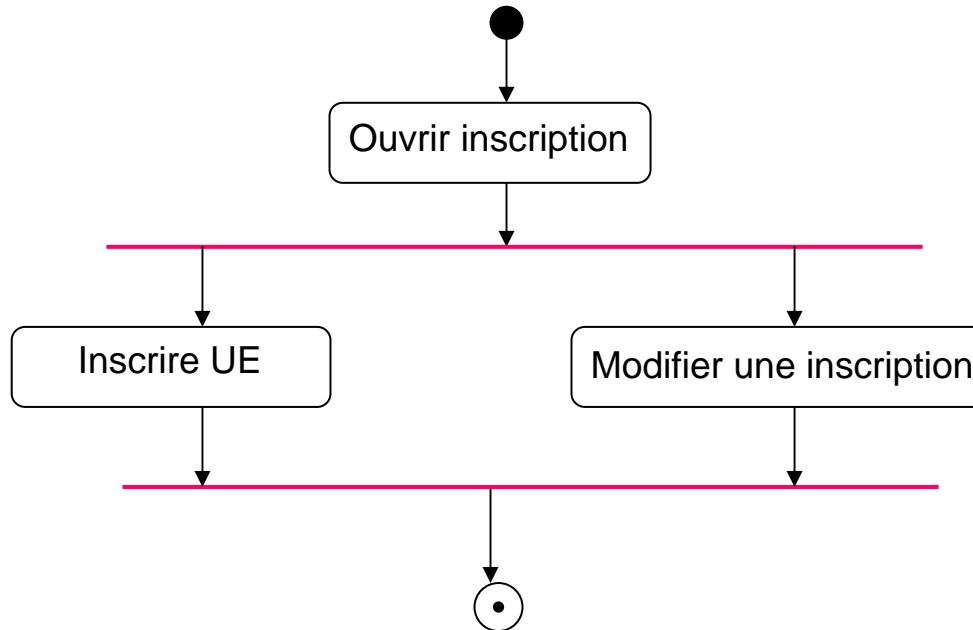
## 4.4. Diagramme d'activités

- 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

- Exemple diagramme d'activités





- Exemple synchronisation



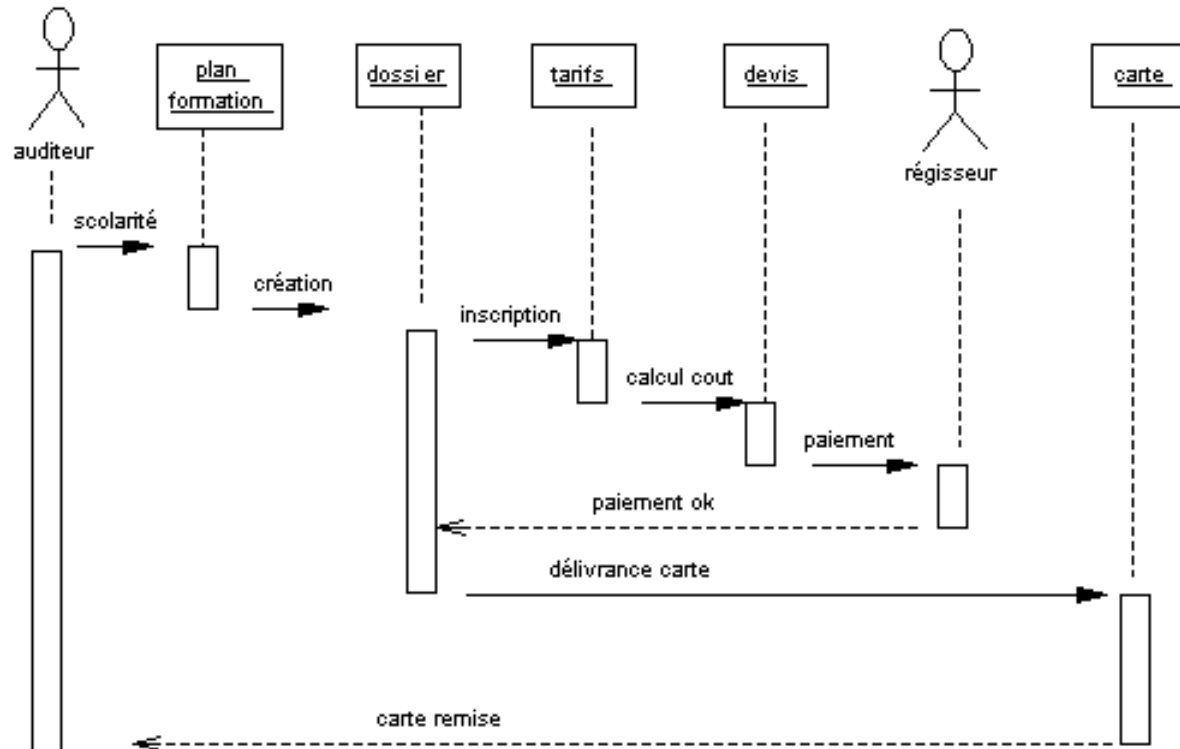
Inscrire UE et modifier élève sont deux actions //



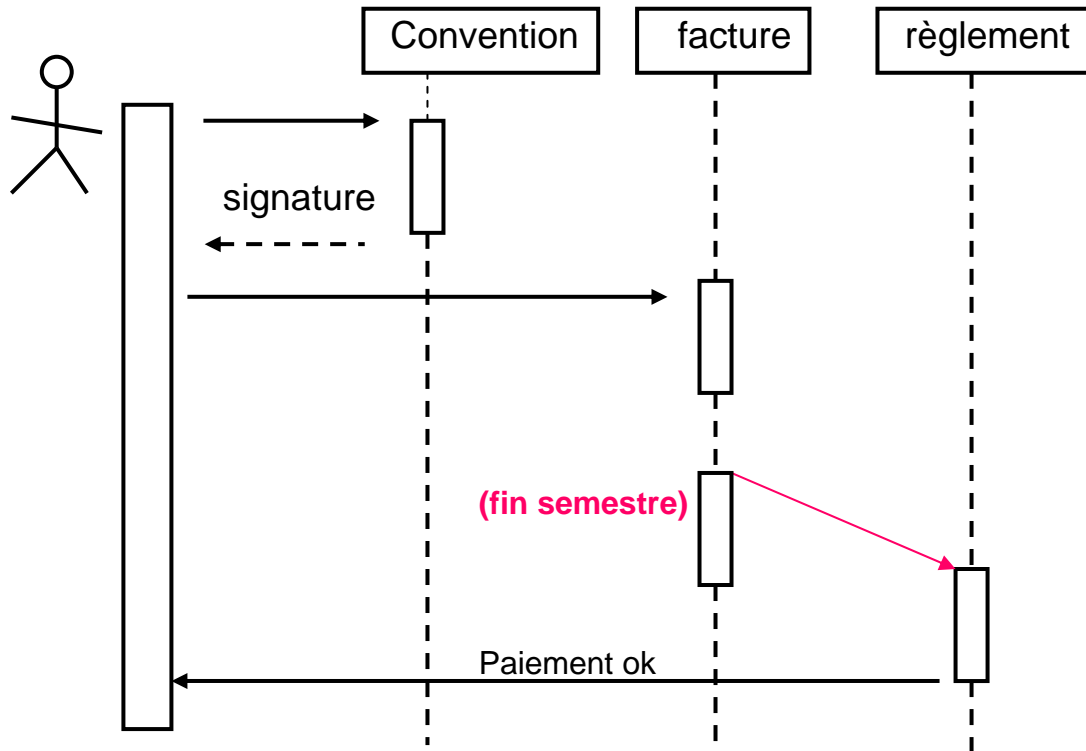
## 4.5. Diagramme de séquence

- 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 : 

- Exemple diagramme de séquence



- Exemple diagramme de séquence : durée



## 4.6. 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:

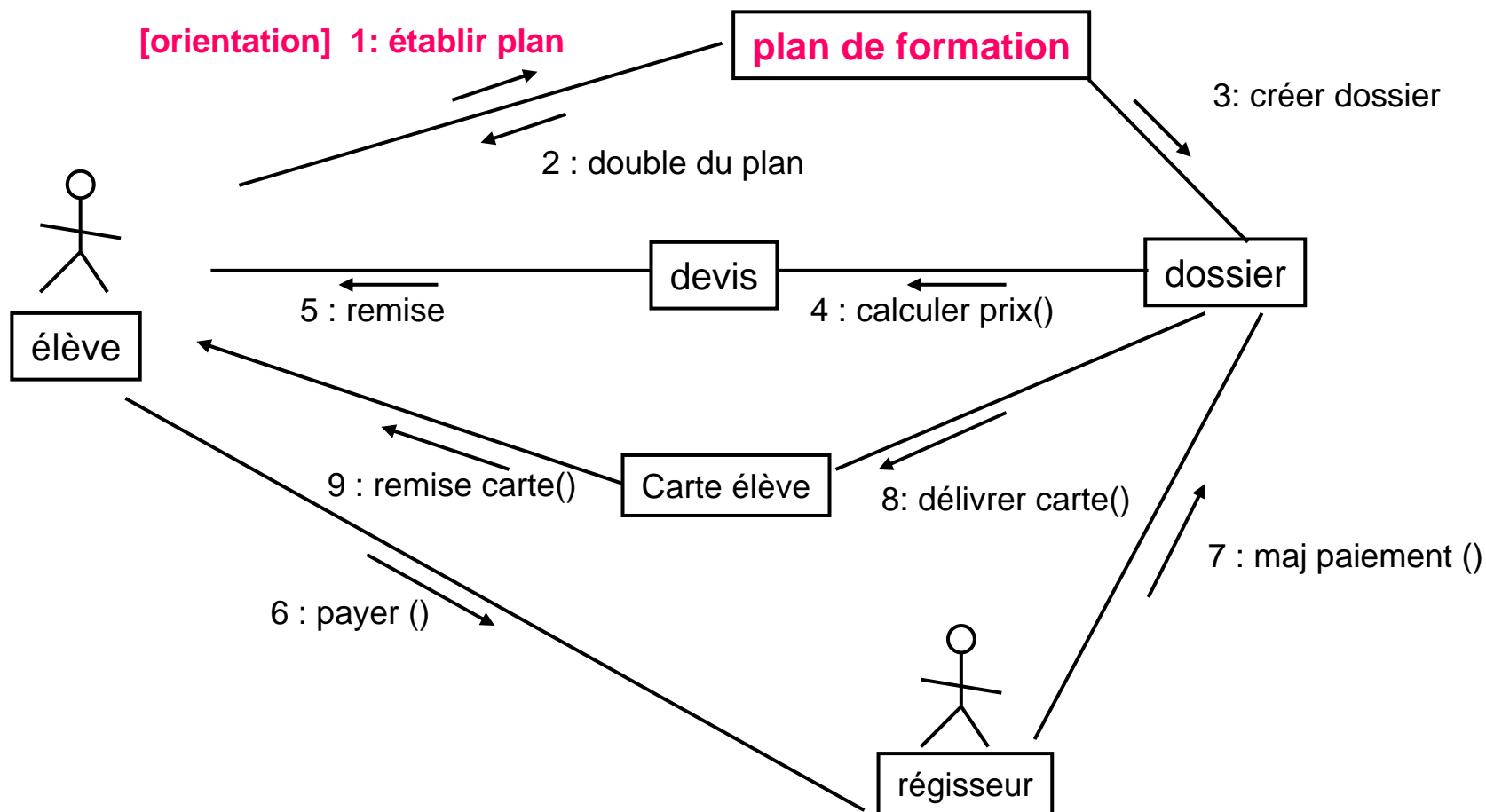
[activité professionnelle=Ok] 1: établir plan()

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

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

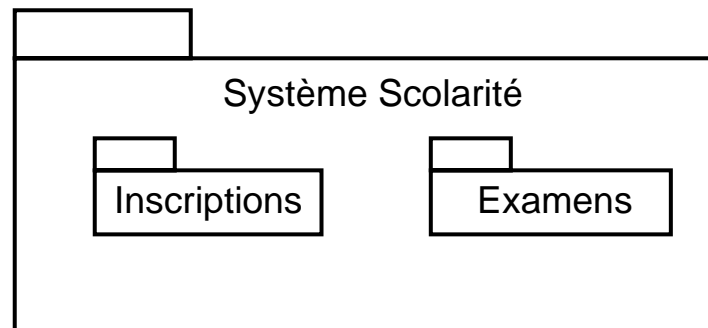
Le message 3 ne sera envoyé qu'après le message 2

Envoi en // de n messages



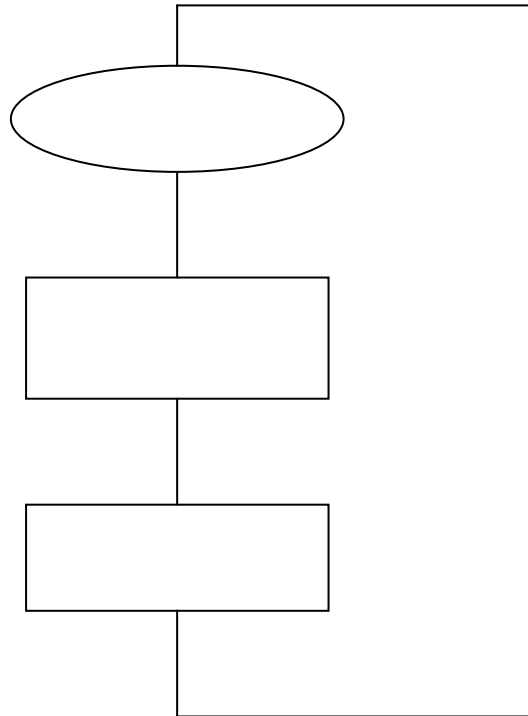
## 4.7. Diagramme de composants

- Description des composants du système
- Décomposition en sous-système, programme, processus, tâche, module



**Système et sous-systèmes**

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

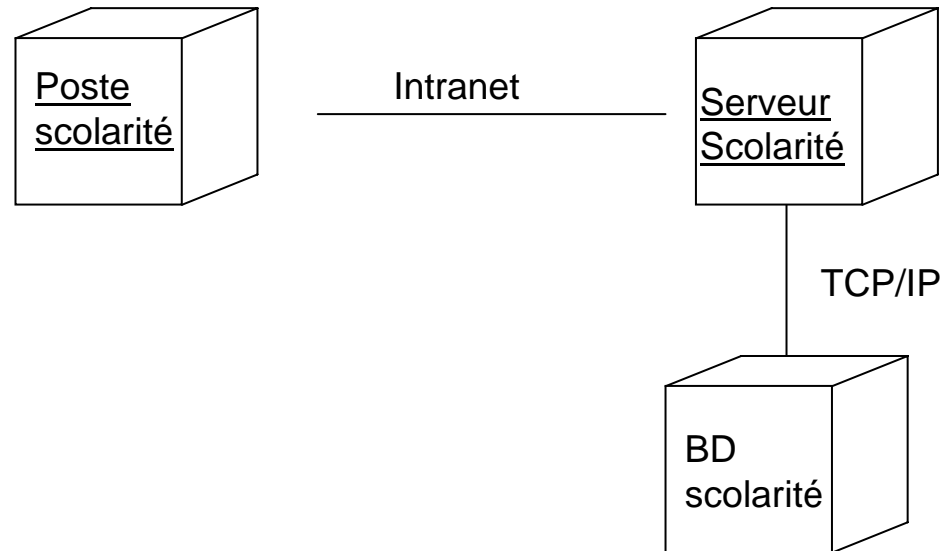


## 4.8. Diagramme de déploiement

- 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 d'un système d'inscription



## 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)**

# 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 »

# 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
- Eléments décrits:
  - Les acteurs
  - Le scénario
  - Les pré-conditions et garanties
  - Les exceptions, les extensions, les utilisations d'autres cas

## 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

Alternative: Tiers payeur

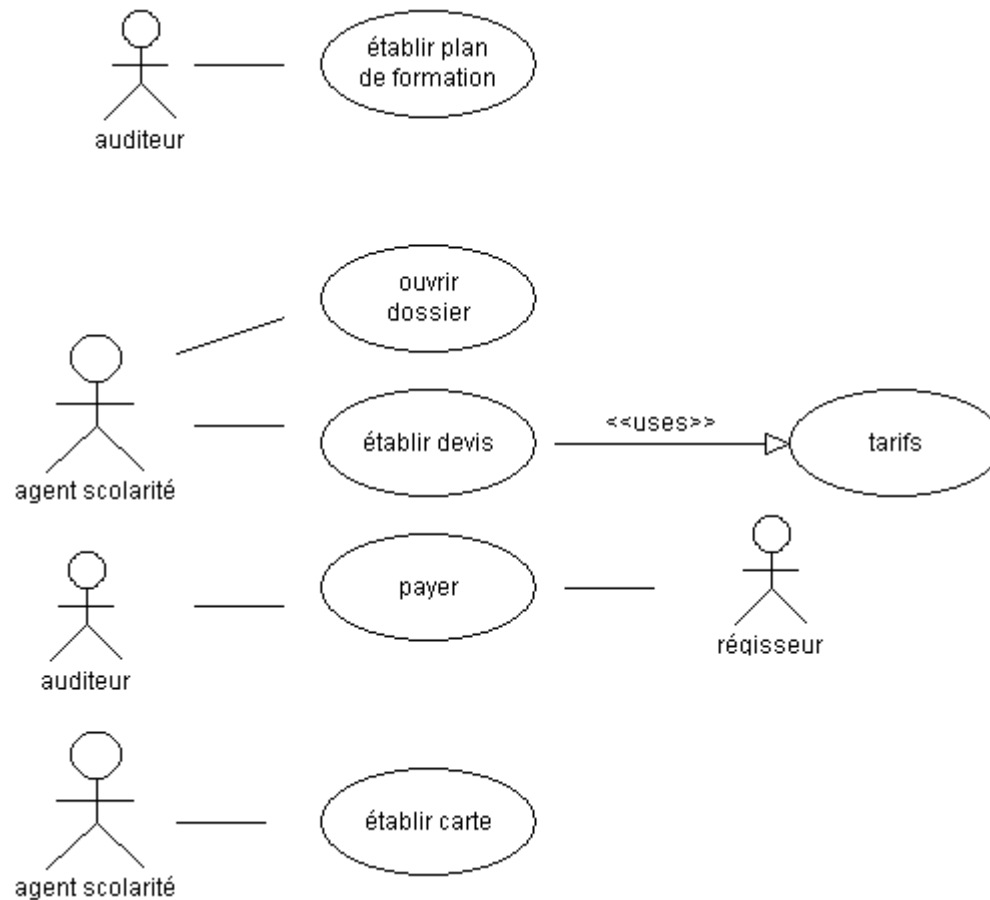
6a. Un tiers payeur est identifié, une convention est établie.

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.

# Description graphique: cas d'utilisation « inscription »





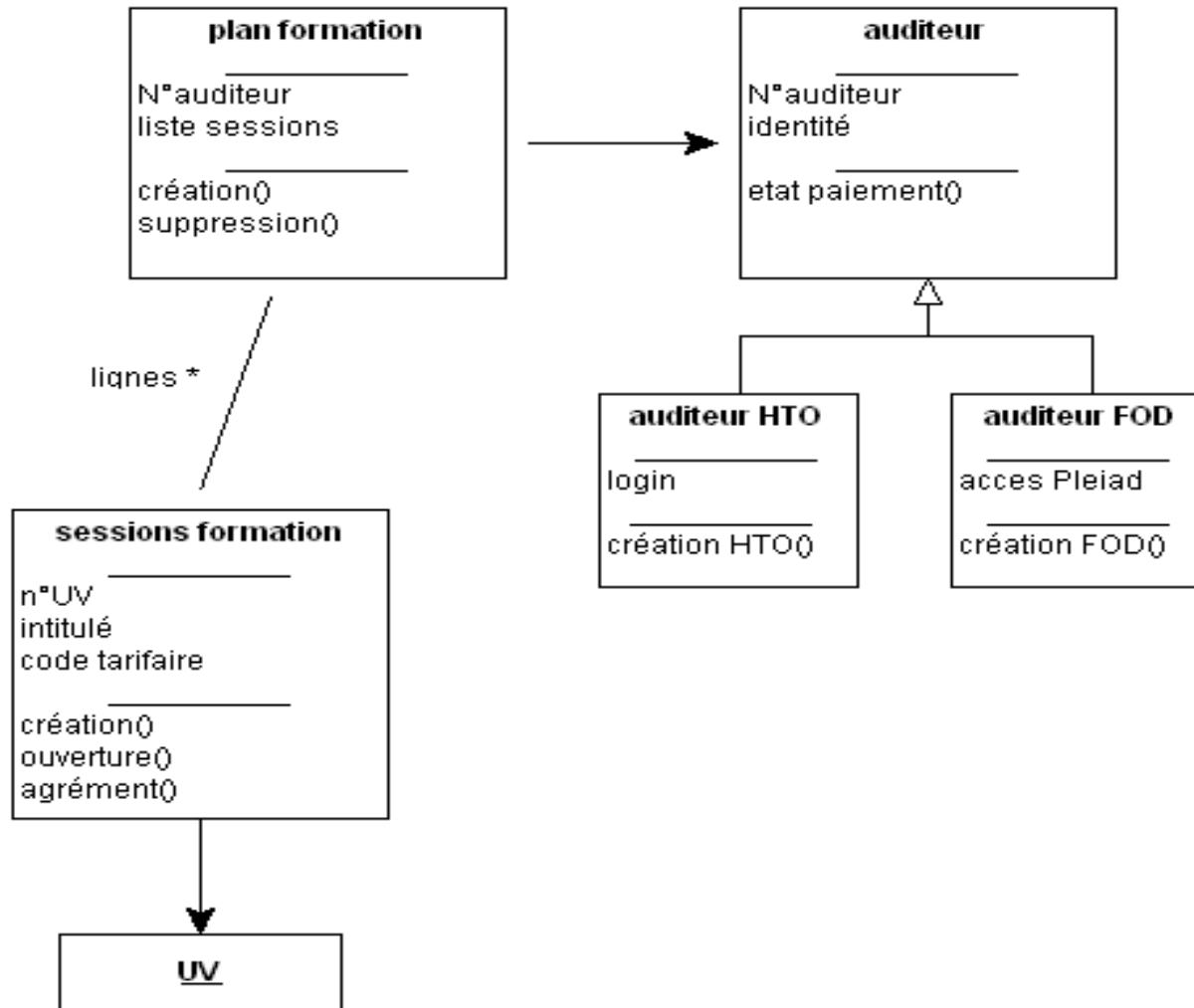
## Tableau: cas d'utilisation « inscription »

Cas	<b>Inscription d'un auditeur</b>
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	Paielement 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

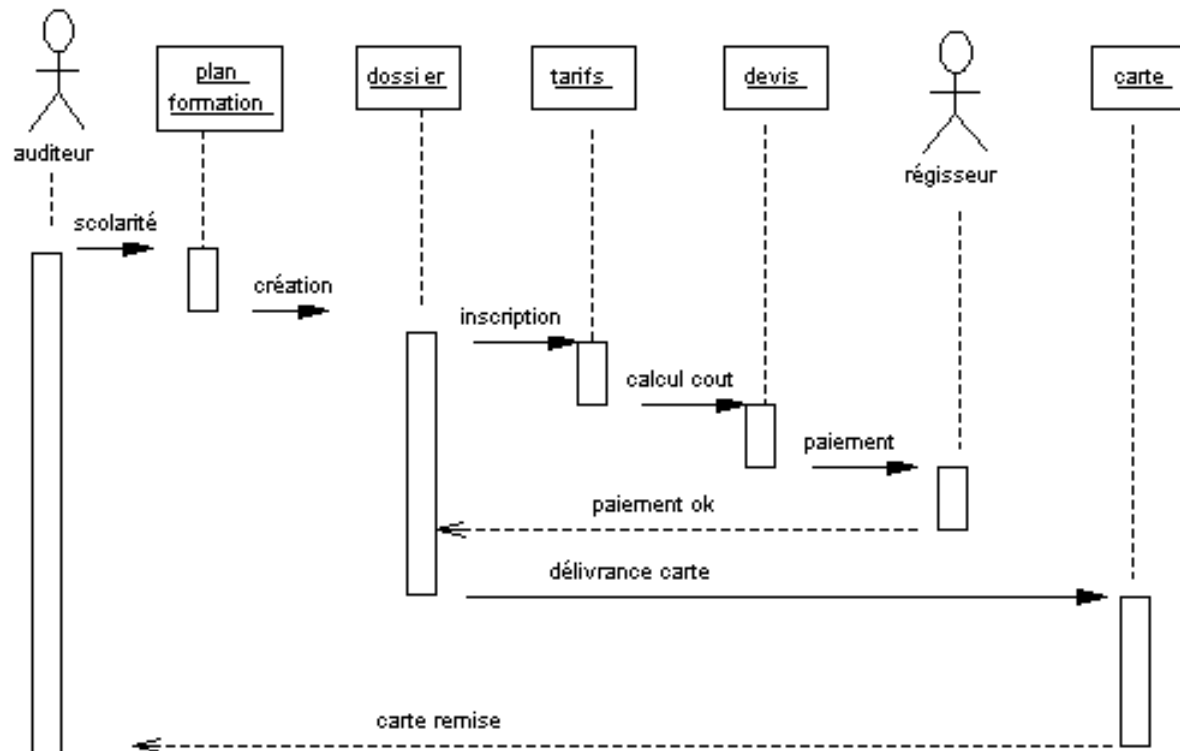
## Diagramme de classe: cas inscription



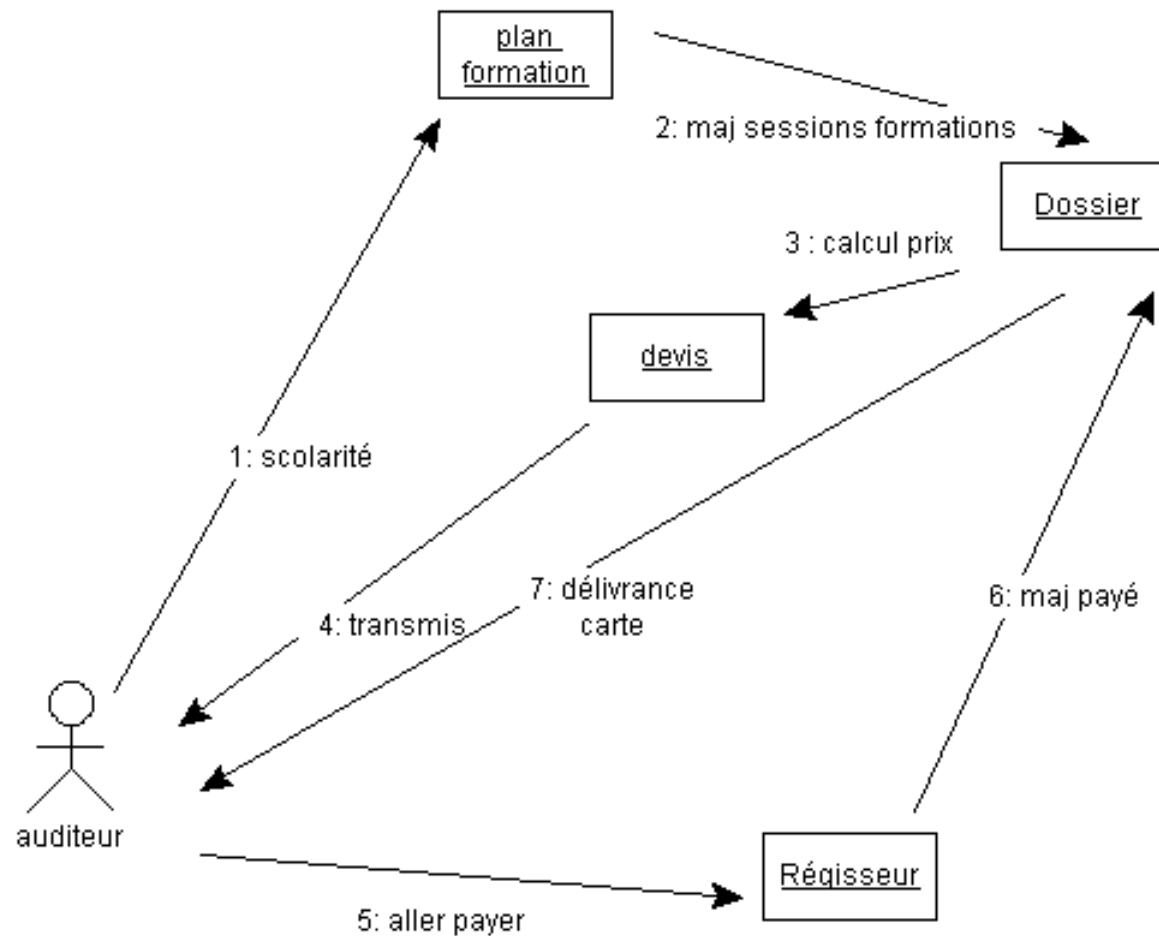
# Diagramme d'interaction

- 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: cas inscription



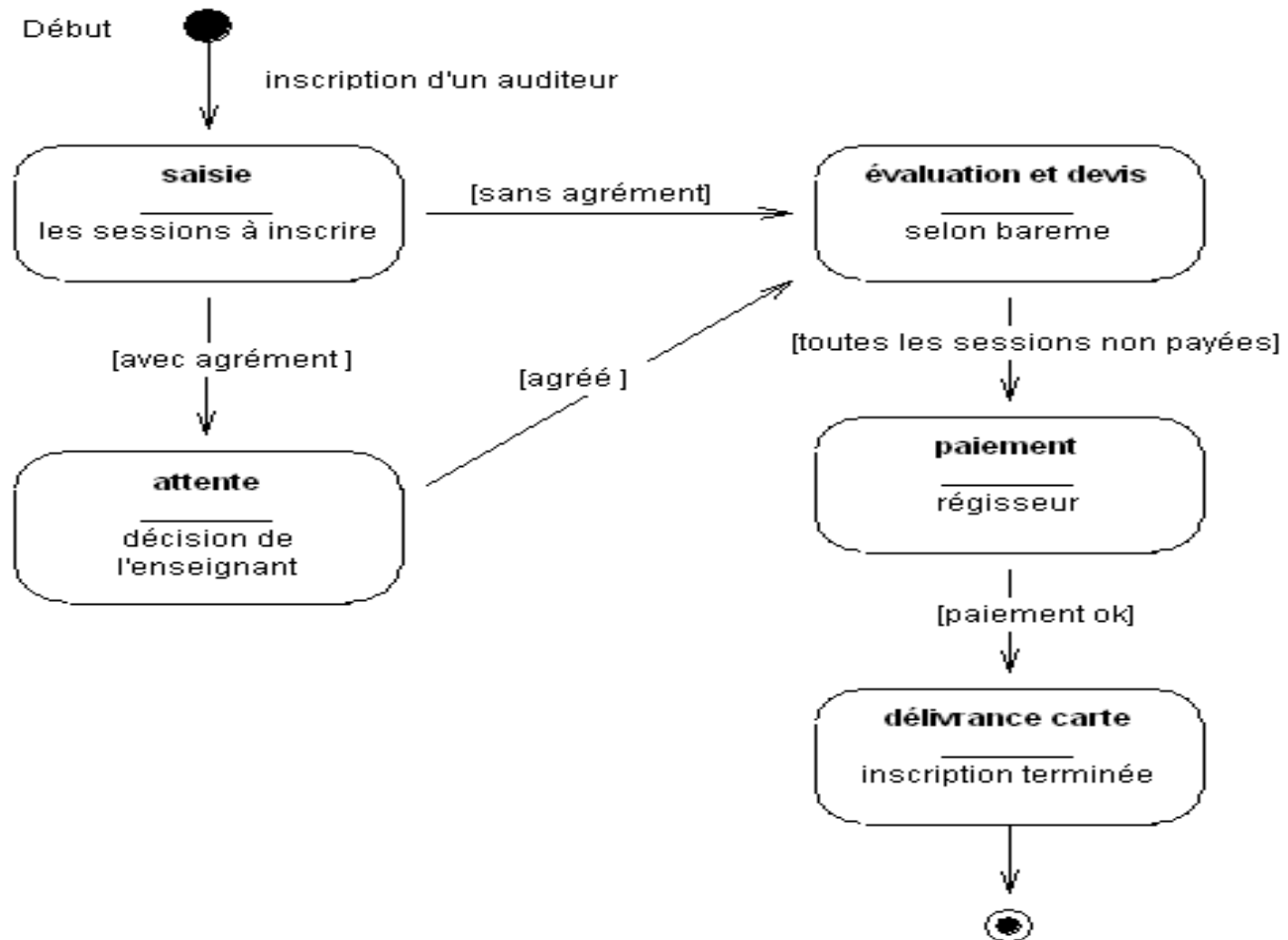
## Diagramme de collaboration: cas inscription



# Diagramme Etats-Transitions

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

## Diagramme états-transitions: cas inscription

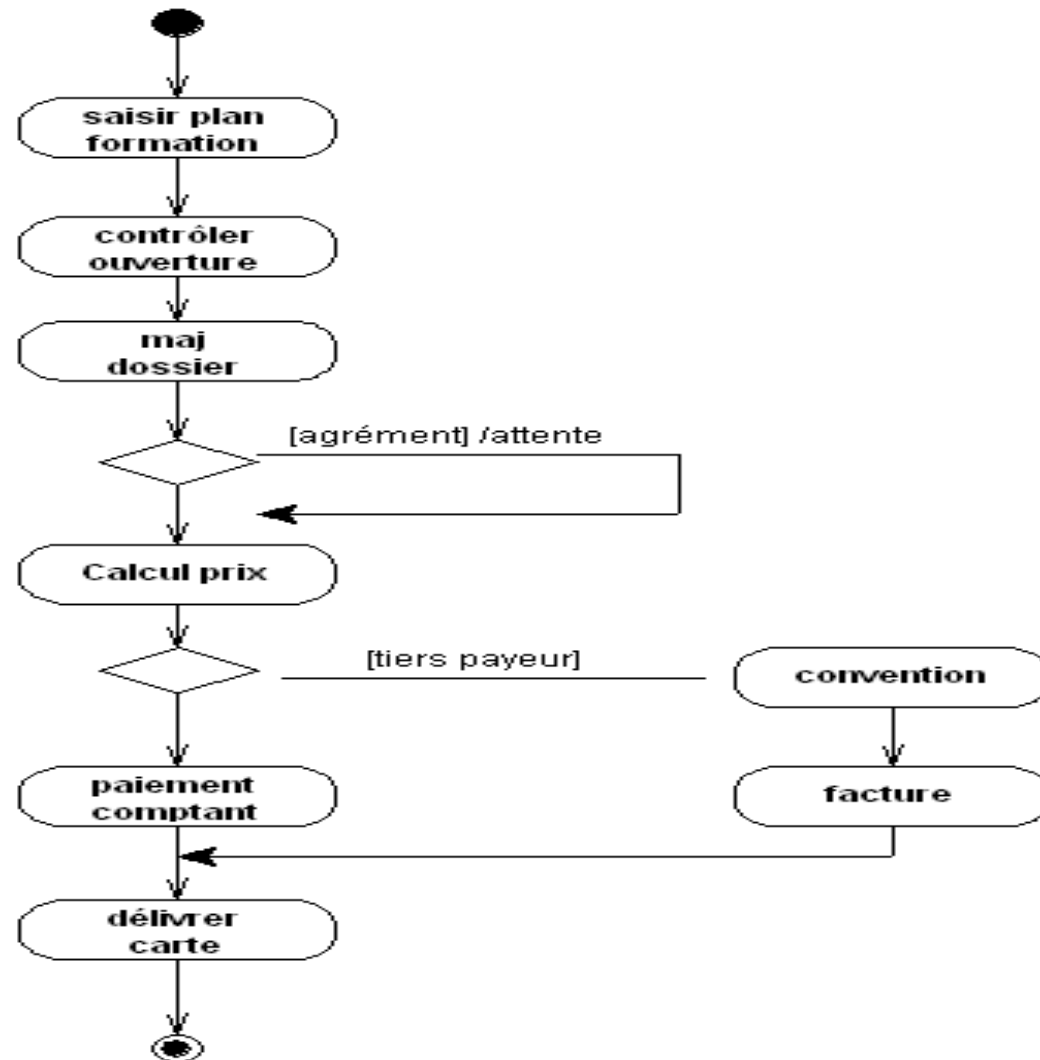




# Diagramme d'activités

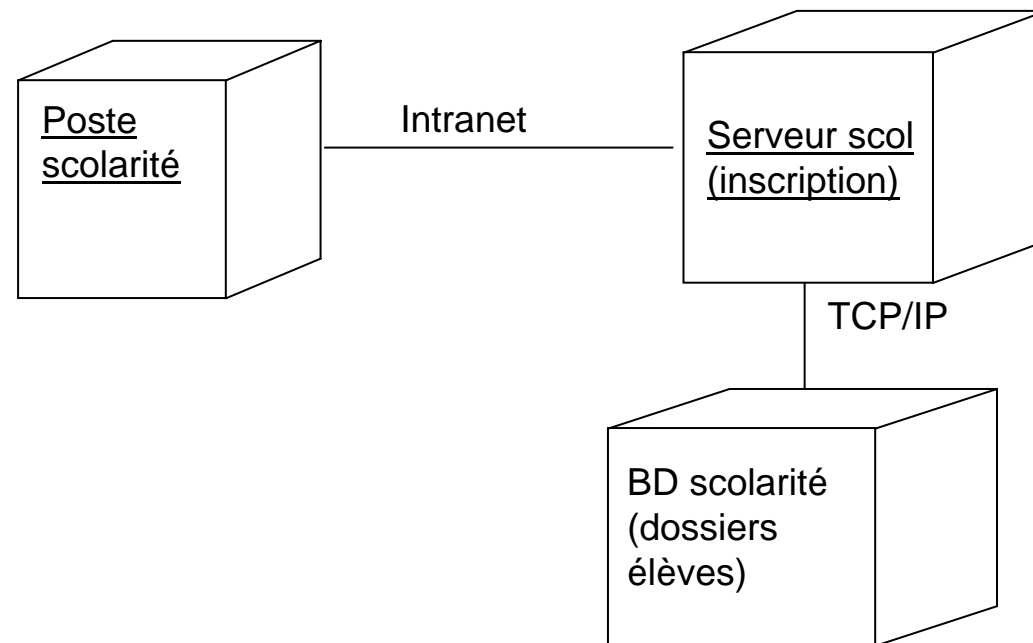
- Organisation des activités:
  - Activités en séquence
  - Activités parallèles
  - Activités conditionnelles
- Une activité peut être composite
- Quand?
  - Analyse cas d'utilisation,
  - modélisation métier,
  - Algorithme complexe

## Diagramme d'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