UML
0 - Introduction : UML et la méthode
Version courte

Bertrand LIAUDET

SOMMAIRE

Version Mai 2025

INTRODUCTION : UML ET LA METHODE

Il est facile de décrire la méthode encore que
son application exige à coup sûr savoir et pratique.
La méthode est dénuée de sens tant qu'elle est déconnectée du rapport au savoir.

1. Introduction à la méthode (slide 1)

- Voir le cours d'introduction à la méthode pour UML :

  http://bliaudet.free.fr/IMG/pdf/ESGI-UML2-1-2-Introduction_a_la_methode_version_PPT.pdf

2. Introduction UML

Notions

UML

- Unified Modeling Language : Langage de Modélisation Unifié

Modèle

- Représentation simplifiée d'un système ou d'un phénomène pour en faciliter l'analyse.

Modélisation

- Processus de création d'un ou plusieurs modèles pour étudier ou simuler un système réel.

UML : des langages (et non pas un seul langage !) (slide 2)

- UML propose des langages graphiques de modélisation
  - ⇒ Unifié :
    - ⇒ Un ensemble de langages communs pour les informaticiens
    - ⇒ Il y a des syntaxes partagées d'un langage UML à un autre.

- UML propose des langages graphiques pour :
  - ⇒ L'analyse des architectures d'un système (point de vue statique).
  - ⇒ L'analyse des processus d'un système (point de vue dynamique).
    - ⇒ Ces langages permettent de réaliser toutes les étapes de la conception.

Des diagrammes

- Chaque analyse réalisée en UML s'appelle un « **diagramme** ».
  - ⇒ Un diagramme UML, c'est un modèle graphique.

- Exemples de diagramme d'architecture :
  - ⇒ Le diagramme des cas d'utilisation
  - ⇒ Le diagramme des classes

- Exemples de diagramme de processus :
  - ⇒ Le diagramme de séquence

Méthode (slide 3)

- L'UML propose des langages, pas une méthode :

  ⇒ On peut en faire ce qu'on veut !

- Mais concrètement, les usages d'UML suivent une méthode :

  ⇒ On utilisera UML en suivant notre méthode d'analyse des Systèmes d'Information.

Historique (slide 4)

Fin 80, Années 90 :

« **Crise du logiciel** ». Le paradigme procédural atteint ses limites avec l'accroissement exponentiel du développement logiciel.

Premières utilisations des langages orientés objets dans l'industrie.

Fort développement du C++ et du Java.

Plusieurs dizaines de méthodes objets. Signalons particulièrement : **OMT** (Object Modeling Technique, Rumbaugh), **OOD** (Object Oriented Design, Booch), **OOSE** (Object Oriented Software Engineering, Jacobson)

1995	UML 0.8 de Booch, Jacobson et Rumbaugh. Première ébauche publique.
1996	UML 0.9 de Booch, Jacobson et Rumbaugh
1997	**UML 1.0**
1999	RUP, Le processus unifié de développement logiciel, Booch, Jacobson, Rumbaugh, Eyrolles, 1999. Le RUP, c'est une méthode.
1999	UML 1.3
2003	UML 1.5
2004	**UML 2**. : UML 2.0 - Guide de référence, Booch, Jacobson et Rumbaugh, CampusPress, 2004

➔ On voit que **l'UML est intimement lié à la P.O.O.**

➔ Les diagrammes de classes et d'objets sont toujours très utiles quand on fait de la P.O.O : c'est technique.

➔ Aujourd'hui, on utilise l'UML pour la conception de base.

Les 4 diagrammes les plus importants pour conception (slide 5)

- Le **diagramme de cas d'utilisation** :
  ⇒ Pour la description statique des **usages du système**.

- Le **diagramme de classes** :
  ⇒ Pour la description statique des **classes métier du système**.
  ⇒ Les classes métier correspondent aux tables de la BD.

- Les **diagrammes de séquence système et objet**:
  ⇒ D'un point de vue fonctionnel : **pour chaque scénario nominal de chaque cas d'utilisation.**
  ⇒ D'un point de vue technique : **pour le MVC de chaque cas d'utilisation.**

- Les **diagrammes d'activités** :
  ⇒ D'un point de vue fonctionnel : **pour décrire tous les scénarios d'un cas d'utilisation.**
  ⇒ Autre point de vue fonctionnel, plus général : on utilise un **diagramme d'activités à travées multi-utilisateurs** pour décrire des processus fonctionnels complexes.

Syntaxe (slide 6)

- L'UML propose un ensemble de **langages formalisés** :

  ⇒ les symboles graphiques ont une signification précise.

  ⇒ **Il y a donc une syntaxe à connaître !**

Logiciels pour faire de l'UML (slide 7)

- Il existe de nombreux logiciels permettant de produire une conception UML.
  - ⇒ Divers « **modeler UML** » :
    - ⇒ StarUML, BoUML, WinDesign, UMLet, etc.
    - ⇒ Rational Software Modeler : descendant de « **Rational Rose** », le premier né. Aujourd'hui propriété d'IBM.
    - ⇒ **PowerDesigner** aujourd'hui propriété de **SAP** (anciennement PowerAMC)
  - ⇒ Des logiciels de dessin :
    - ⇒ **Draw.io** = http://diagrams.net/ : logiciel de dessin JavaScript en accès libre.
  - ⇒ Des logiciels d'IA:
    - ⇒ **chatGPT** (arrivé le 30 novembre 2022) : ce n'est pas parfait, mais ça peut aider dans l'analyse.

- Les logiciels payants permettre de :
  - ⇒ **Travailler sur l'ensemble des diagrammes** de façon cohérente
  - ⇒ **Générer du code** automatiquement.
  - ⇒ Faire du **reverse engineering**

- Pour ce cours, on utilise :
  - ⇒ PowerDesigner
    - ⇒ N'existe que sur PC
    - ⇒ Pour les MAC, il faut une machine virtuelle !

But du cours : un usage actuel d'UML ! (slide 8)

1 Diagramme des Classes métier

2 Cas d'utilisation :

⇒ Diagramme de cas d'utilisation

⇒ Notion de scénario nominal

⇒ Diagramme de séquence-système

⇒ Diagramme d'activités

3 Maquettage – Prototypage (papier ou avec outil de prototypage)
https://www.blogdumoderateur.com/tools/design/prototypage/

4 Structuration MVC en UML

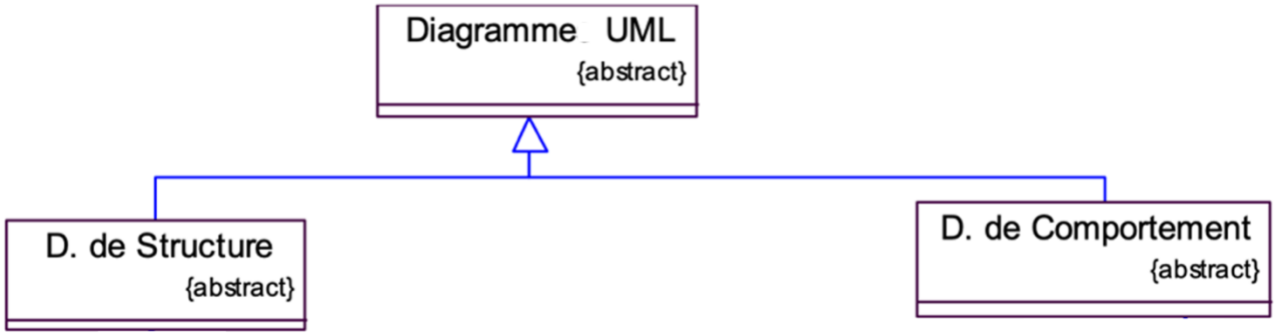
5 Simulation CRUD du SI et de la modélisation UML à partir du MVC

3. Classification détaillées des diagrammes UML (slide 9)

Il y a plusieurs classifications possibles. Comme toujours !

Remarque syntaxique : flèche d'héritage et diagramme de classes

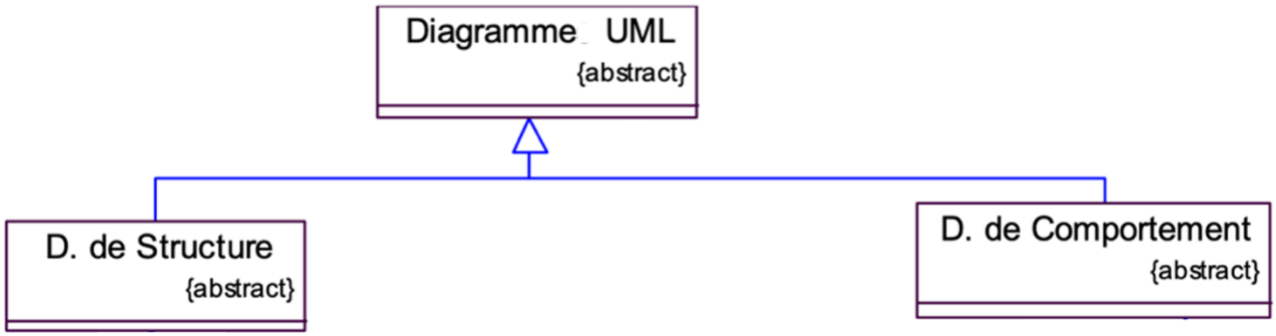
- La première classification utilise le formalisme UML : c'est un diagramme de classes :



- Il y a **3 classes** représentées par des rectangles : ici on ne représente que le nom de la classe.
  - ⇒ Diagramme UML
  - ⇒ Diagramme de Structure
  - ⇒ Diagramme de Comportement

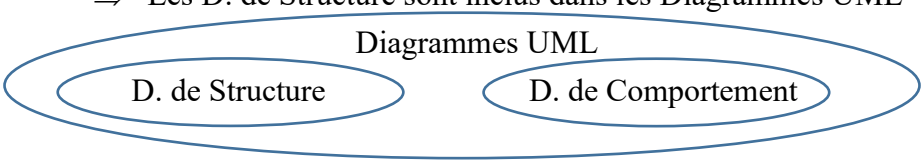
- Les classes sont « **abstraites** » :
  - ⇒ ça veut dire qu'**aucun objet n'instancie directement la classe** :
    - ⇒ il n'existe pas de diagramme UML en général
    - ⇒ ni de Diagramme de structure en général
    - ⇒ ni de Diagrammes de comportement en général :
  - ⇒ ce sont des abstractions pour la classification.

Remarque syntaxique, suite : la flèche d'héritage (slide 10)



- La flèche est une **flèche d'héritage** : 
  - ⇒ Elle veut dire « **est un** » :
    - ⇒ un D. de Structure « est un » D. UML
    - ⇒ Un D de Comportement « est un » D. UML

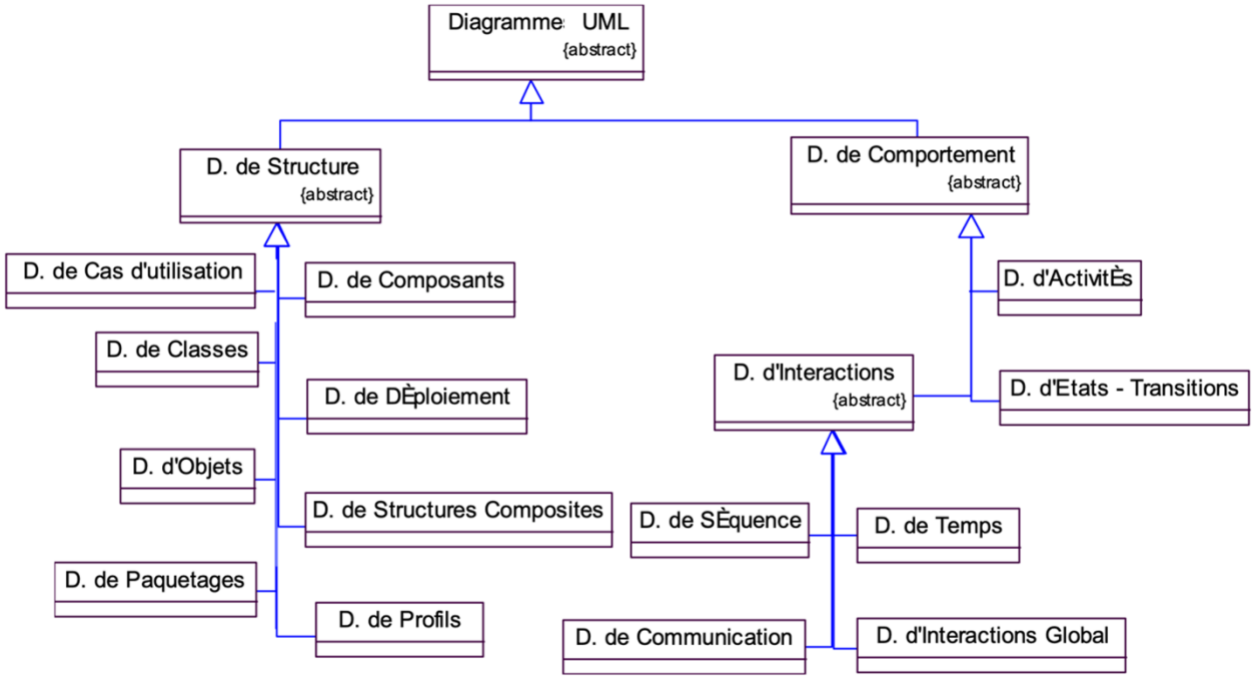
- L'héritage peut être compris comme une inclusion ensembliste :
  - ⇒ Les D. de Structure sont inclus dans les Diagrammes UML



- En UML, les données sont en général représentées par des rectangles
- En UML, les processus sont en général représentés par des ovales (ici il n'y en a pas).

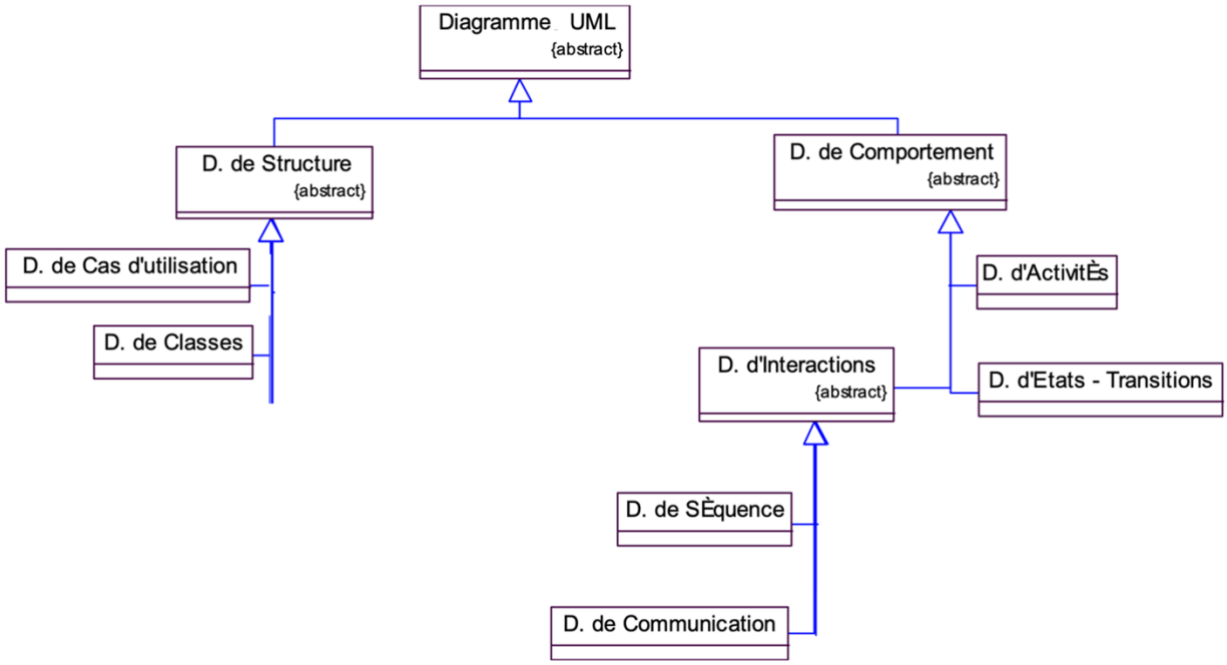
Les 14 diagrammes UML2 (slide 11)

- UML propose **14 diagrammes différents** qui sont adaptés à différentes étapes de la conception fonctionnelle et technique.
- Les diagrammes évoluent un peu avec le temps l'usage : http://www.uml-diagrams.org/uml-24-diagrams.html



Les 4 diagrammes utiles d'UML2 pour ce cours (slide 12)

- **On utilisera que 4 diagrammes UML** pour faire notre conception :
  - ⇒ Classes,
  - ⇒ Use Case,
  - ⇒ Séquence (le diagramme de communication est une variante du diagramme de séquence)
  - ⇒ Activités.
    - ⇒ (Le diagramme d'états-transitions est utile dans certains contextes. Pas directement pour notre projet).



Distinction entre diag. de structure et diag. de comportement (slide 13)

8 diagrammes de structures

- http://www.uml-diagrams.org/uml-24-diagrams.html#structure-diagram

- Les diagrammes de structures correspondent à la description des structures logiques et physiques du logiciel :
  - ⇒ les classes (structures logiques), les packages, les objets
  - ⇒ les fichiers (structures physiques)
  - ⇒ les matériels (structures physiques).

- Ils sont tous « statiques ».

- Un diagramme de structure est toujours une architecture (une organisation structurelle).

- **Le diagramme des cas d'utilisation**
  - ⇒ Il ne fait que lister les usages possibles, sans notion de séquence : il est donc statique et il décrit une architecture => on le classe comme diagramme de structures
  - ⇒ Toutefois, les usages étant des comportements, **il est aussi souvent classé (à tort) en diagramme de comportement.**

6 diagrammes de comportement (slide 14)

- http://www.uml-diagrams.org/uml-24-diagrams.html#behavior-diagram

- Les diagrammes de comportements correspondent à la description des comportements du logiciel.

- Ils sont **dynamiques** quand leur description correspond à une séquence c'est-à-dire un déroulement dans le temps :
  - ⇒ diagramme d'activités,
  - ⇒ diagramme d'états-transitions.
  - ⇒ diagramme d'interactions (donc le diagramme de séquence)

- Un diagramme dynamique est toujours un algorithme.

Distinction entre diagrames fonctionnels, techniques et mixtes (slide 15)

- **Les diagrammes fonctionnels** servent pour l'analyse fonctionnelle :
  - ⇒ Le diagramme des cas d'utilisation,
  - ⇒ Les diagrammes de séquences et d'activités,
  - ⇒ Les diagrammes de classes métier.

- **Les diagrammes techniques** servent pour l'analyse technique.
  - ⇒ Les diagrammes de classes techniques,
  - ⇒ les diagrammes de séquence objet,
  - ⇒ Les diagrammes d'objets.

- **Les diagrammes mixtes** servent pour l'analyse fonctionnelle et pour l'analyse technique.
  - ⇒ Les diagrammes de séquence,
  - ⇒ Le diagramme d'activités,
  - ⇒ Le diagramme de classes,
  - ⇒ Les diagrammes d'états-transitions

4. Détails et exemples (slide 16)

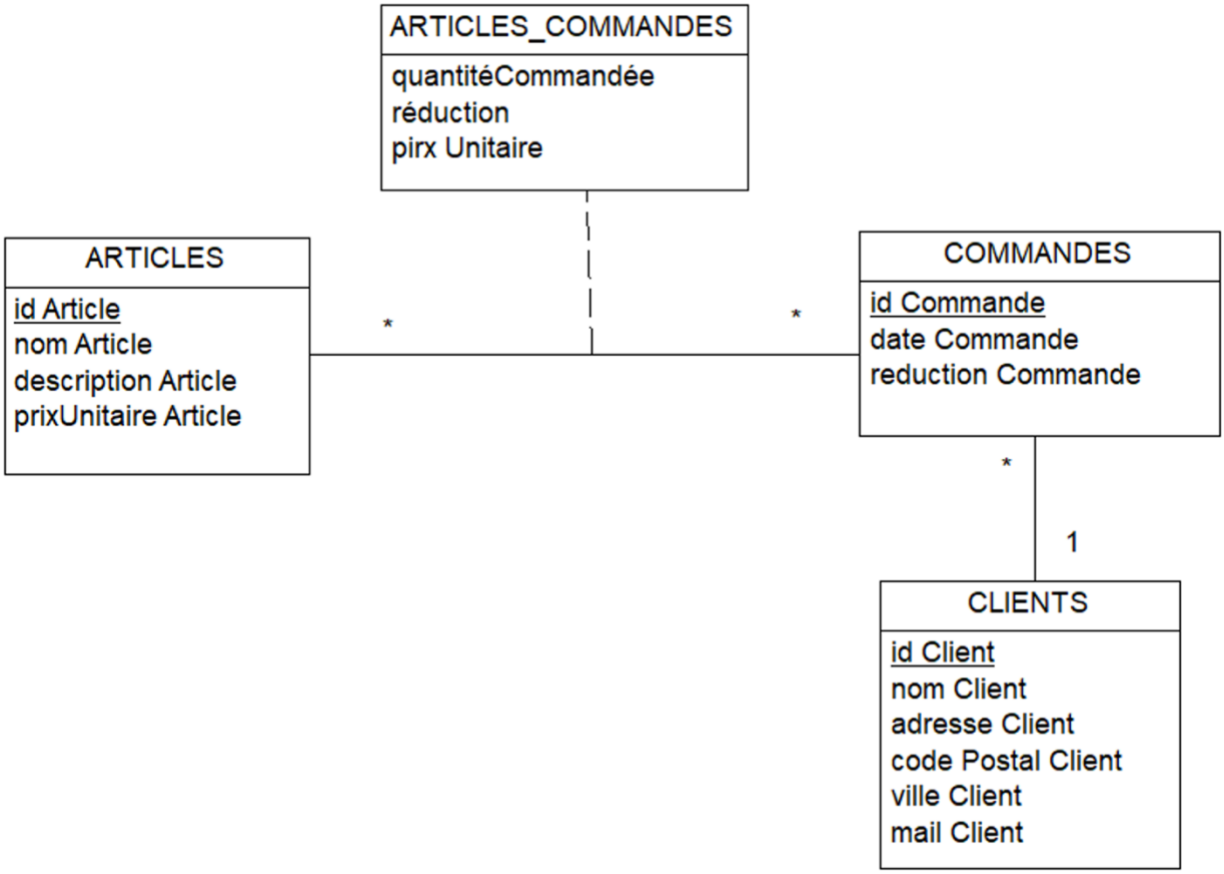
- On montre des exemples pour donner une idée de ce à quoi cela ressemble esthétiquement.
  - ⇒ On commence par les principaux diagrammes (slides 17 à 29)
  - ⇒ On termine par les autres

Diagramme de classes

- Il représente la **structure statique** du système en termes de classes et de relations entre les classes.
  - ⇒ **Le diagramme des classes métier** correspond à la base de données. **C'est fonctionnel.**
  - ⇒ **Le diagramme des classes techniques** détaille toutes les classes, les méthodes, les interfaces et les paquetages. **C'est technique.**

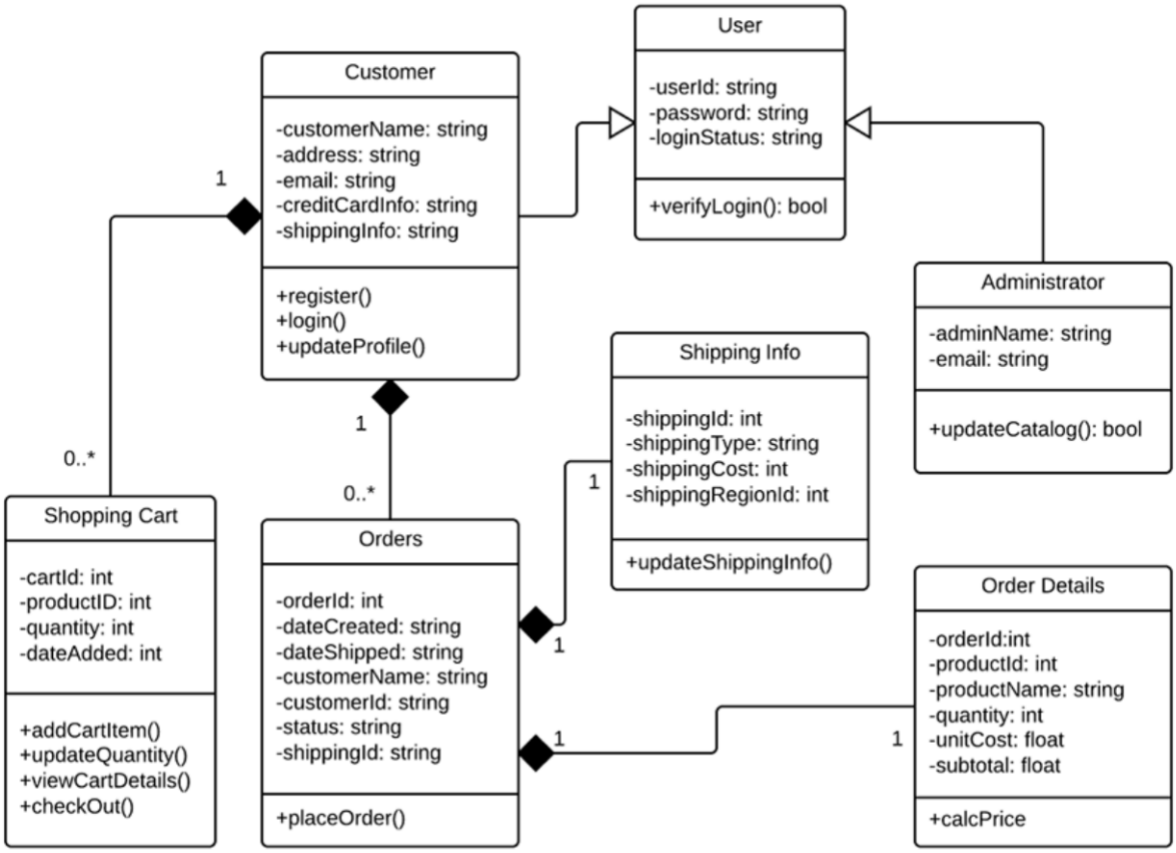
1 : Diagramme de Classes métier (slide 17)

- Exemple d'un pattern de diagramme de Classes métier pour gérer des commandes :



- Ici on peut voir une « classe association »

2 : Diagramme de Classes techniques, avec méthodes (slide 18)



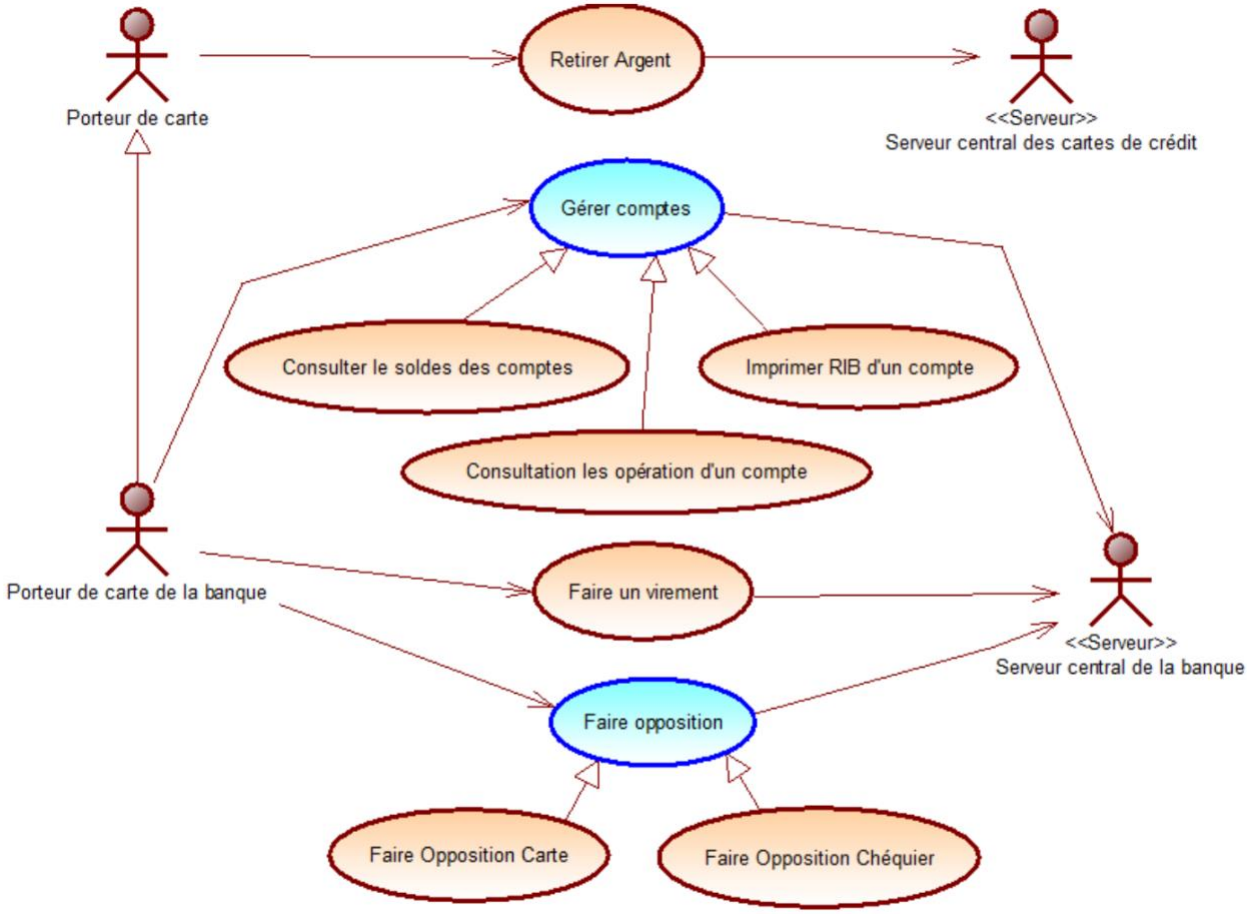
Ici on peut voir des méthodes, des compositions et des héritages.

Diagramme de cas d'utilisation (Use Cases) (slide 19)

- Un « use case » représente un **usage du système** et les différents acteurs de ces usages.

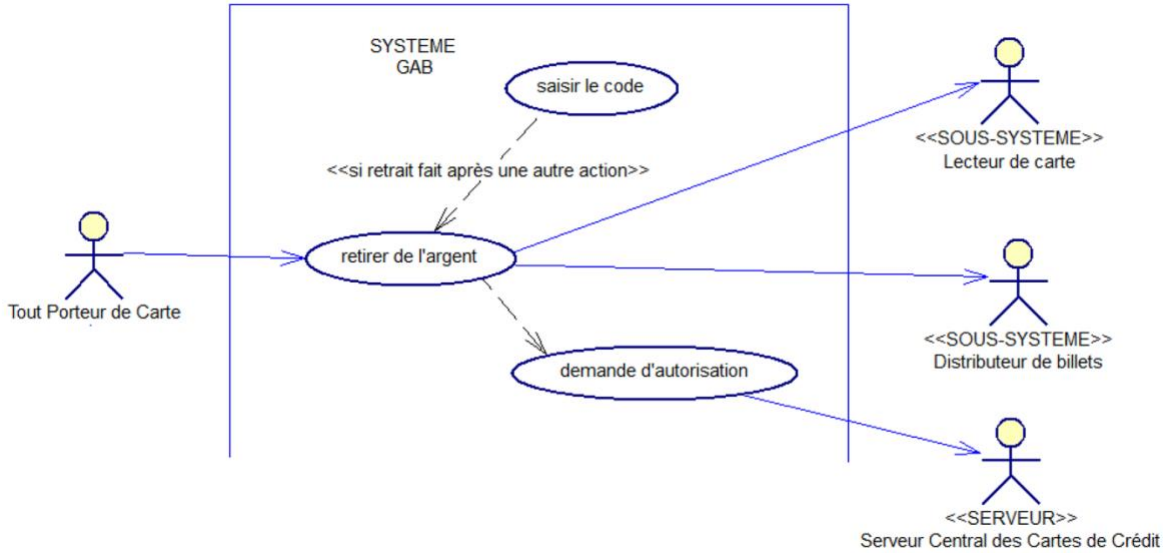
  ⇒ Un usage, c'est une **fonctionnalité complète du système** : ce que l'utilisateur est venu faire.
- Le diagramme de cas d'utilisation représente tous les use cases et tous les acteurs.

1 : Exemple d'un pattern de cas d'utilisation pour un GAB (guichet automatique de banque) :



Ici, on peut voir des héritages et des acteurs « serveurs ».

2 : Détail d'un Use Case avec include, extend et Sous-Systèmes (slide 20)



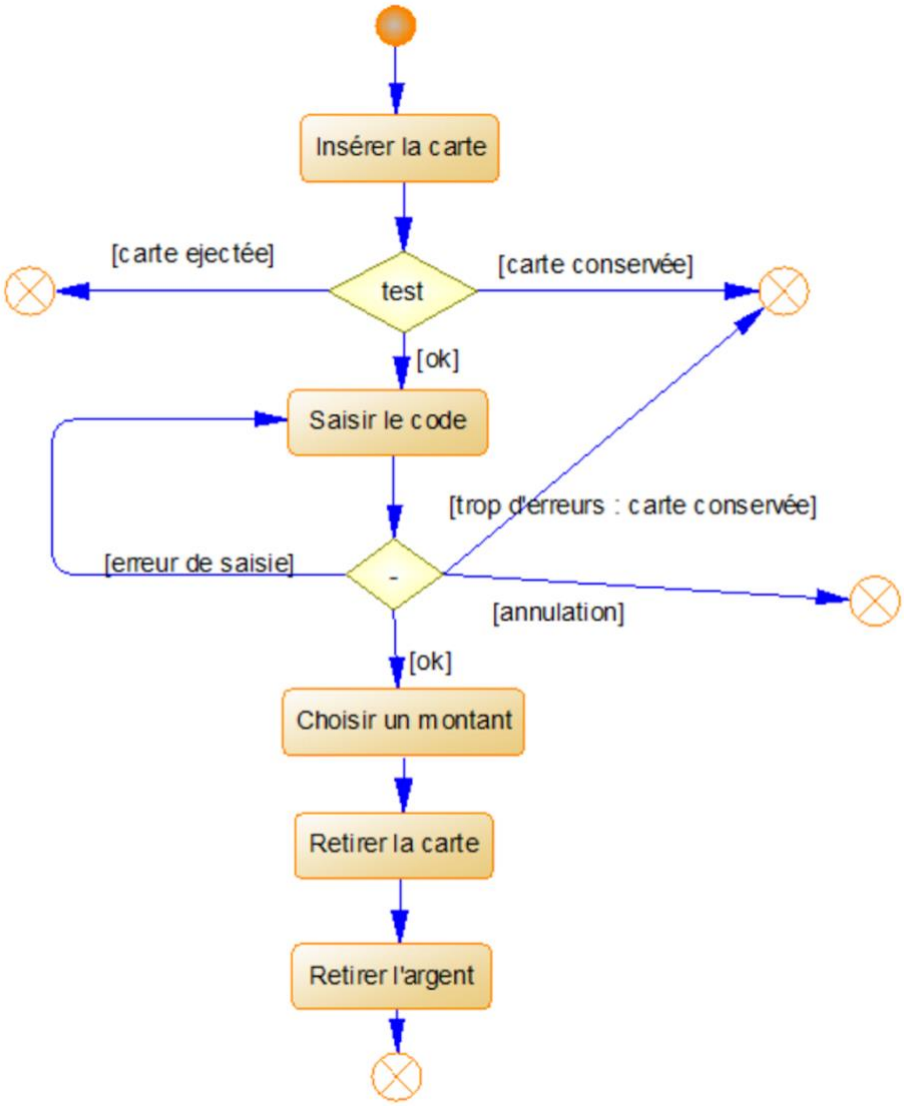
Ici on peut voir un include et un extend et des acteurs « sous-système ».

Diagramme d'activités (slide 21)

- Il permet de représenter **toutes les activités algorithmiques de programmation impérative** classique (de l'affectation au programme) avec des tests et des boucles.
  - ⇒ Il est adapté pour montrer **tous les scénarios d'un Use Case** : c'est du fonctionnel.
  - ⇒ Il est aussi adapté pour montrer des **processus complexes multi-utilisateurs** dans un **diagramme à travées**.
  - ⇒ Il correspond aux **anciens organigrammes** (algorithme graphique) : c'est du technique.

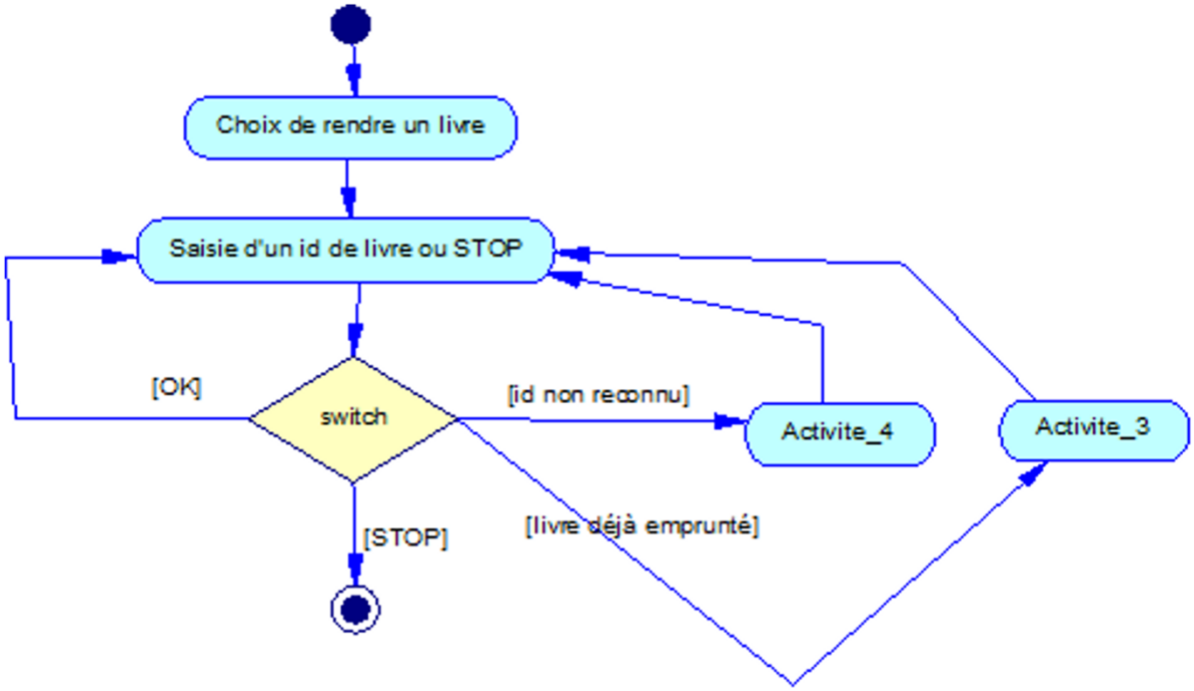
1er exemple : retrait d'argent sur un GAB (slide 22)

⇒ Diagramme d'activités fonctionnel



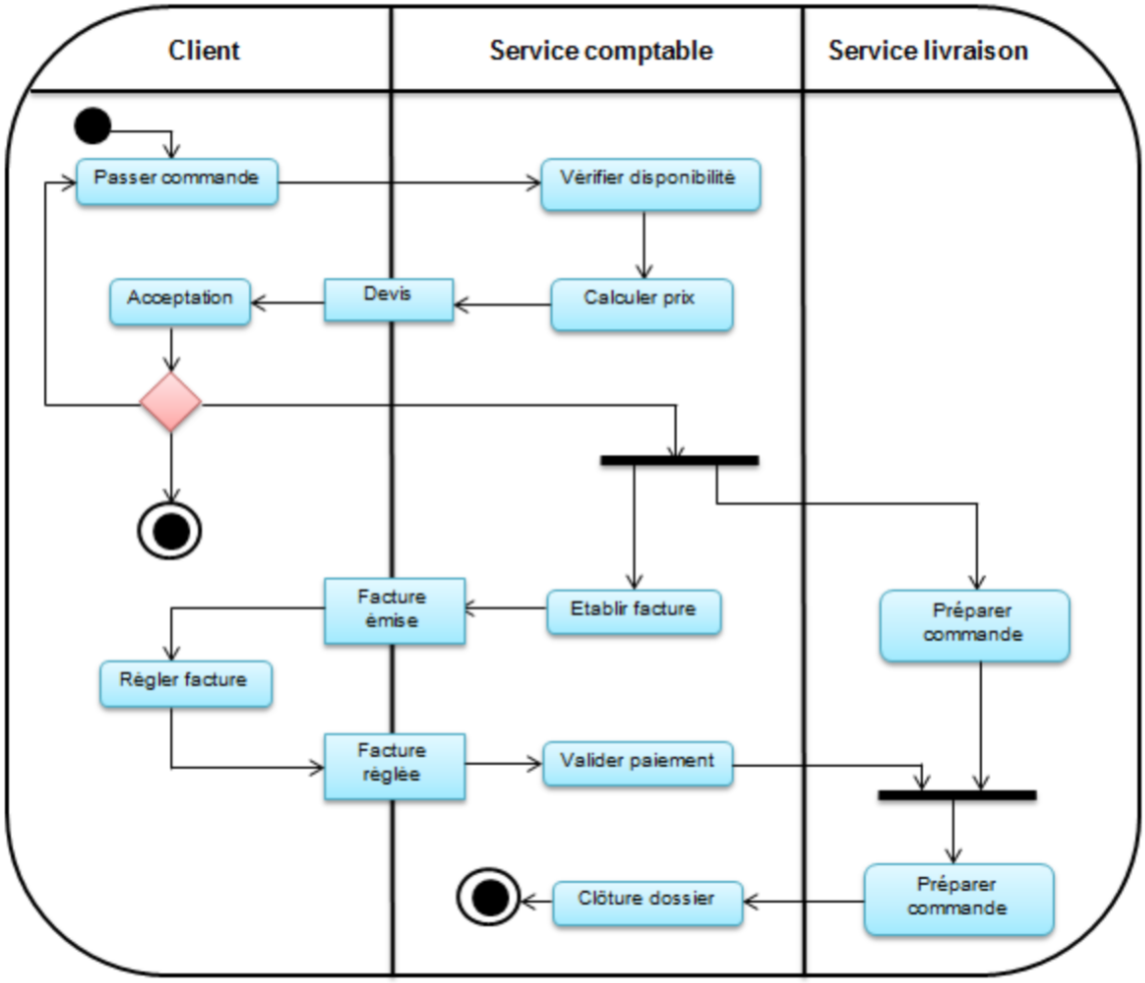
2ème exemple : retour d'un livrre en bibliothèque (slide 23)

⇒ Diagramme d'activités fonctionnel



3ème exemple : diagramme d'activité à travées (slide 24)

⇒ Diagramme d'activités fonctionnel



4è exemple : Anciens organigrammes (slide 25)

⇒ Diagramme d'activités technique

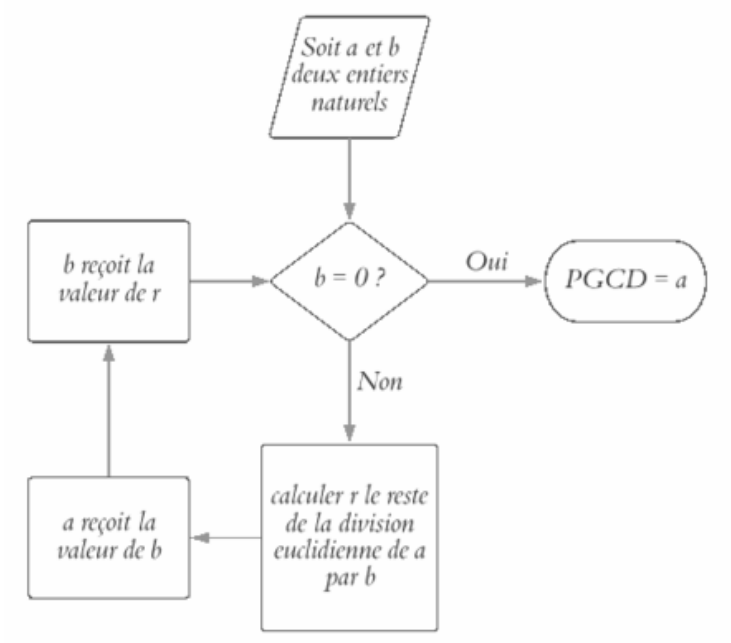


Diagramme de séquence système (slide 26)

- C'est une espèce de diagramme de communication.
- Tout comme les diagrammes de communication, il montre **les interactions** entre des composants du système dans une séquence temporelle.

  ⇒ **Le diagramme de séquence système** montre des interactions entre les « acteurs » et un « système » ou entre les « systèmes ».

  ⇒ Il permet de représenter le déroulement des scénarios (instances des cas d'utilisation). **C'est fonctionnel.**

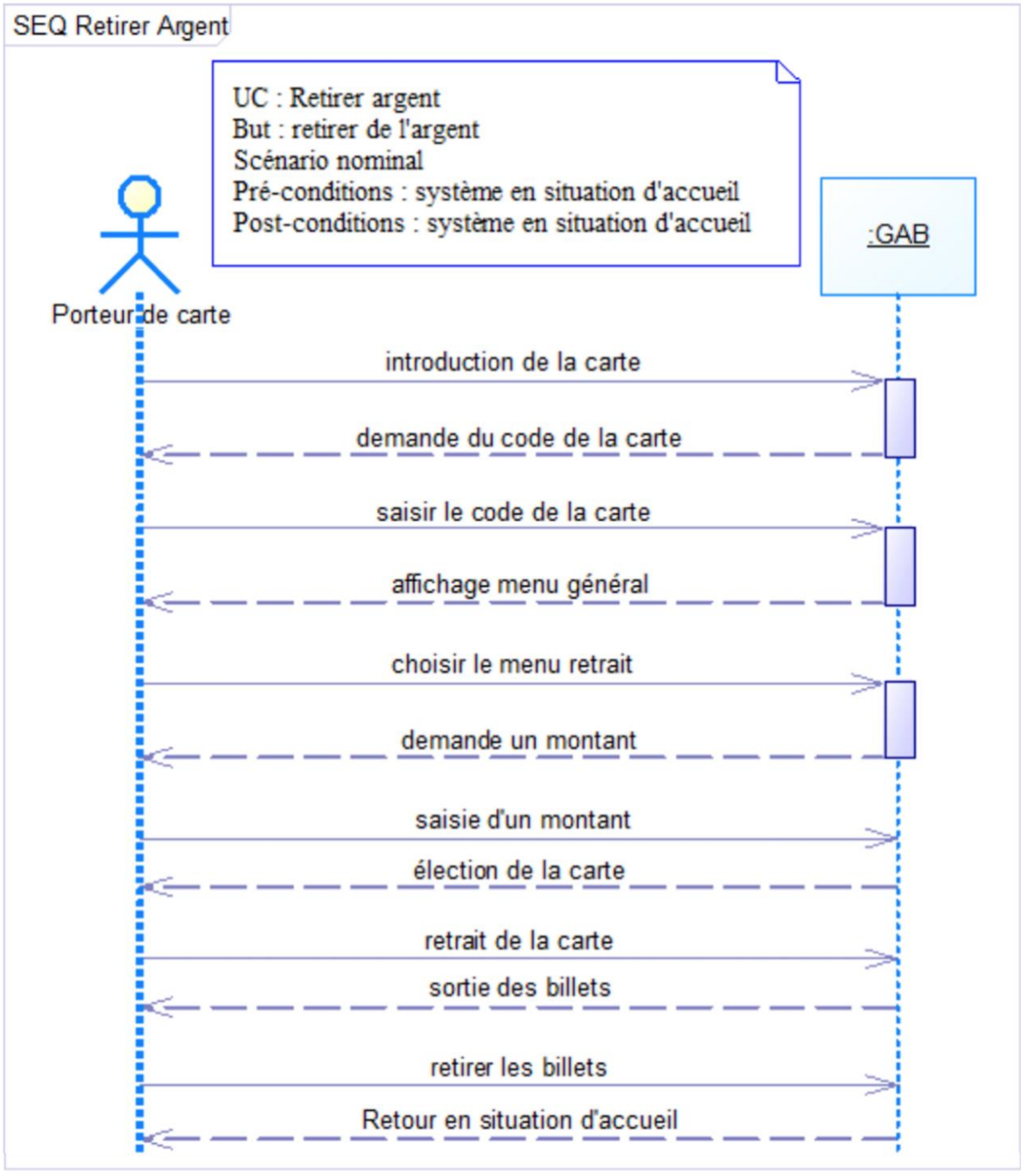
  ⇒ **Le diagramme de séquence objet** montre des interactions entre des « objets » au sens de la POO :

  ⇒ il montre les appels de méthode (=fonction) entre objets. **C'est technique.**

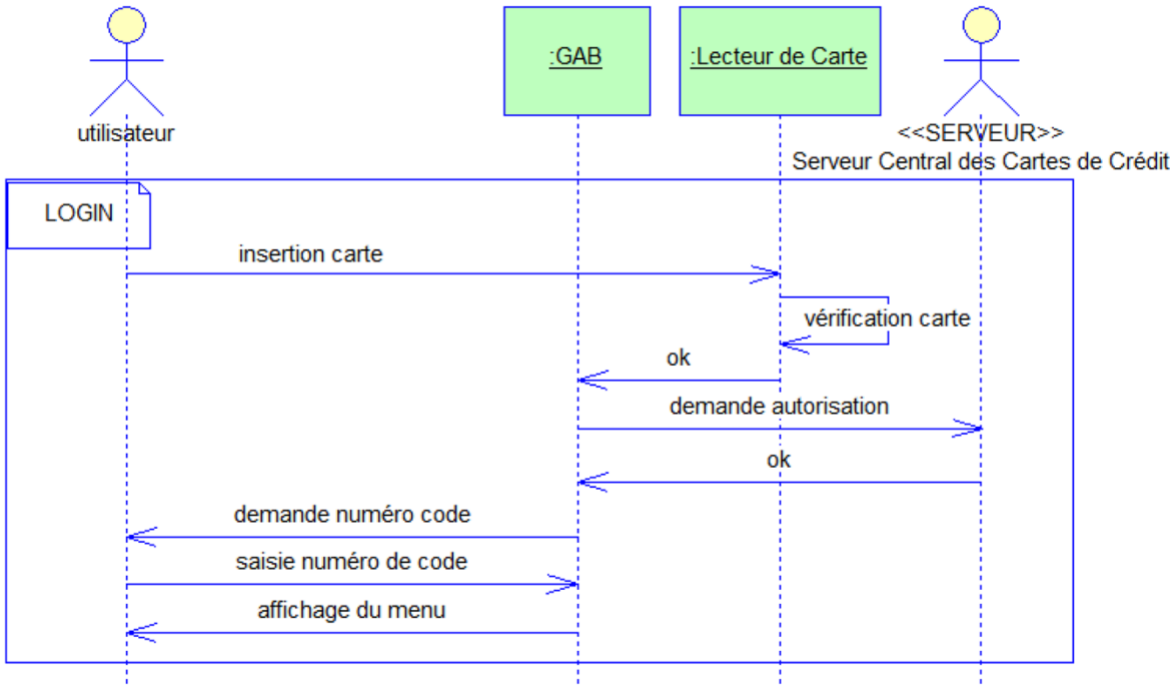
1 : Diag. de Séquence système pour le Use Case Retrait d'argent (slide 27)

⇒ Diagramme de séquence système (ou diagramme de séquence fonctionnelle)

⇒ GAB : Guichet Automatique de Banque



2 : Diag. de séquence système pour l'authentification sur le GAB (slide 28)

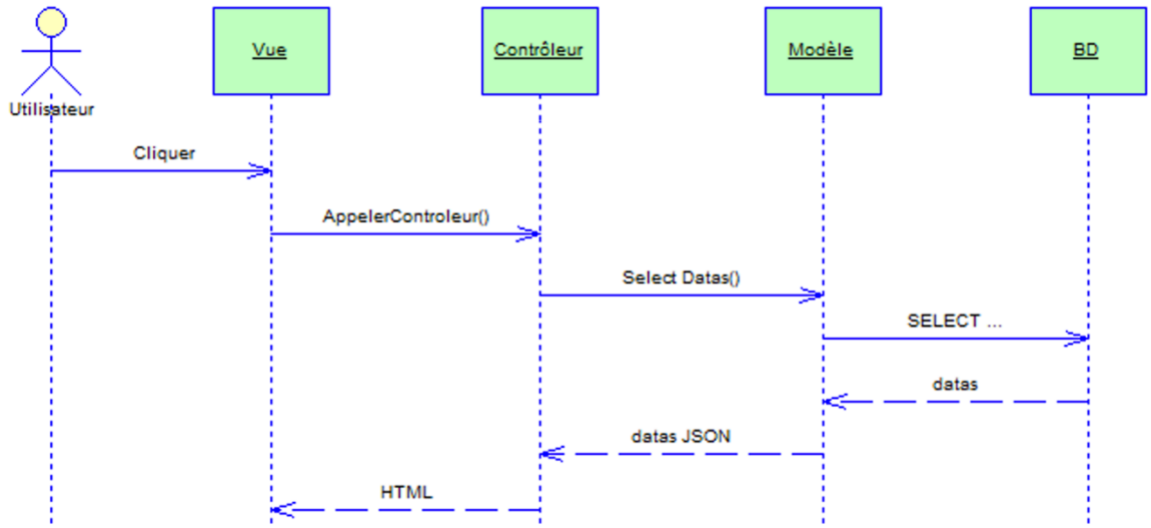


- Ici, **un bloc de séquence** correspond à une procédure : le Login
- On à **2 objets** : le GAB et le Lecteur de Carte.

Diagramme de séquence MVC (slide 29)

- C'est un diagramme de séquence objet ou **diagramme de séquence technique**.

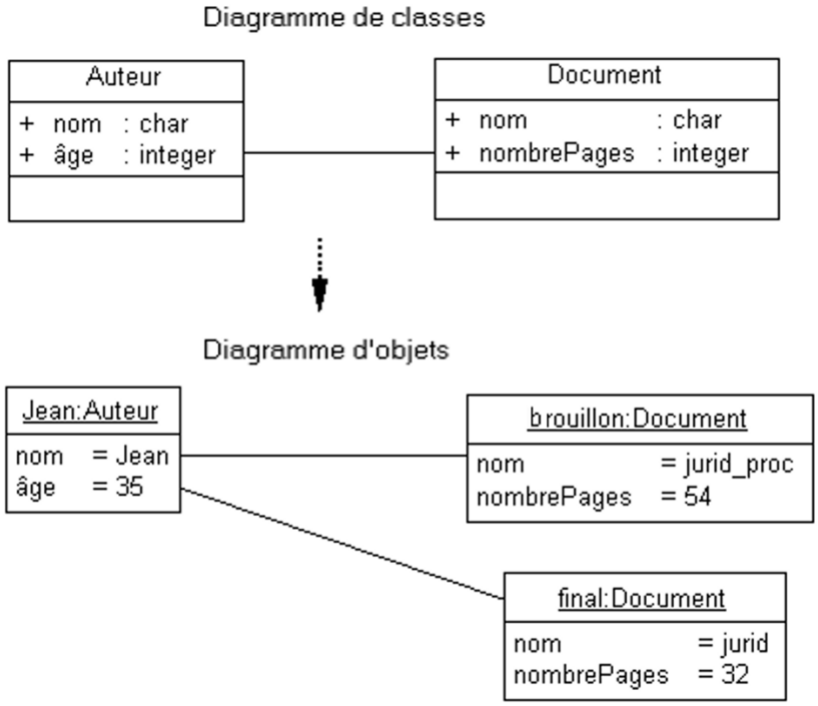
- **Principe général d'un diagramme de séquence MVC :**
  1. **Cliquer** : L'utilisateur clique sur une vue (son écran).
  2. **AppelerControleur()** : La vue appelle un contrôleur.
  3. **Select Datas()** : Le contrôleur appelle une méthode « select datas() » d'un modèle.
  4. **SELECT** : Le modèle fait une requête SELECT … à la BD
  5. **Retour de datas** : La BD retourne des datas
  6. **Retour de datas JSON** : Le modèle retourne les datas au format JSON
  7. **Retour de HTML** : le contrôleur fabrique une nouvelle vue et retourne du HTML
  8. Retour en 1 : l'utilisateur voit une nouvelle vue et peut re-cliquer.



3 autres diagrammes importants d'UML (slide 30) (on peut sauter au slide 34)

5 : Diagramme d'objets (cf. classes)

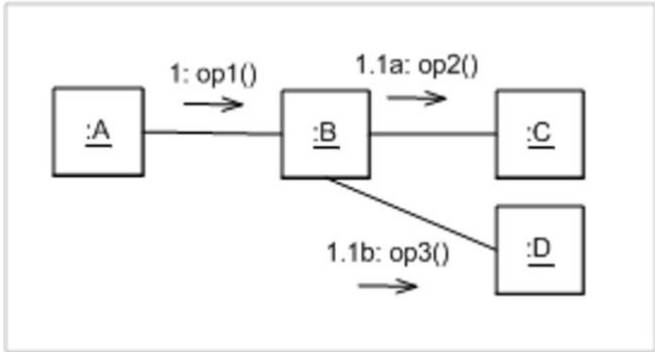
- Un diagramme d'objets représente **un état du système** des objets à un instant donné.
- On y représente les objets, qui sont des instances des classes, avec les valeurs effectives pertinentes des attributs et les liens pertinents entre les objets.
- **C'est un diagramme technique** qui présente une instance du diagramme de classes à un instant donné de la vie du système.



- **brouillon** est un objet de la classe Document. Il est souligné.
- **final** est un objet de la classe Document
- **Jean** est un objet de la classe Auteur

6 : Diagramme de communication (cf. séquence) (slide 31)

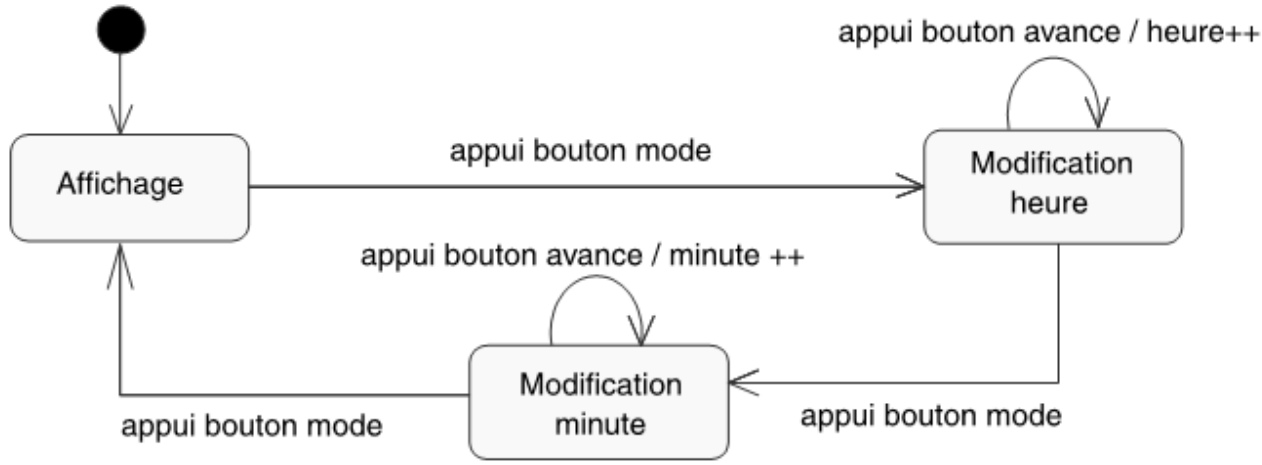
- Comme les diagrammes de séquence, il montre les interactions (les appels de méthodes) entre les objets (les instances des classes) : **c'est un diagramme technique.**
- La séquence temporelle des interactions n'est pas visualisée graphiquement : il faut numéroter.
- Par contre il montre les relations structurelles entre les objets.



:A est un objet générique, sans nom, de la classe A
Idem pour :B, :C et :D.
:A appelle la méthode op1 de :B
:B appelle la méthode op2 de :C
:B appelle la méthode op3 de :D

7 : Diagramme d'états-transition (cf. activité) (slide 32)

- Le diagramme d'états-transitions est une variante du diagramme d'activités.
  - ⇒ Il met en avant les états (point de vue statique) et les transitions d'un état à un autre.
  - ⇒ Inversement, le diagramme d'activités met en avant les activités (point de vue dynamique) et les bifurcations.
- Il montre les évolutions d'un état particulier du système.
  - ⇒ Il est centré sur la présentation des différents états et des transitions entre ces états.



- L'état « Affichage » permet de passer à l'état « Modification heure » en appuyant sur le bouton « mode ».
- Etc.

Diagrammes d'UML plus ou moins secondaires (slide 33)

8 : Diagramme de paquetages

- Les packages permettent de regrouper les classes ou n'importe quel autre élément de modélisation.

9 : Diagramme de composants

- Il représente les composants physiques d'une application (fichiers, bibliothèques, etc.).

10 : Diagramme de déploiement

- Il représente le déploiement des composants sur les dispositifs matériels.

11 : Diagramme de structures composites

- Il représente la structure interne d'une classe : c'est très technique.

12 : Diagramme de profils

- Un profil se représente comme un package, avec un stéréotype <<profil>> en plus.

13 : Diagramme d'interactions global

- C'est un cas particulier de diagramme d'activités.

14 : Diagramme de temps

- Les diagrammes de temps présentent les comportements et interactions de 2 éléments UML en fonction du temps.

5. UML et le cycle en V (slide 34)

Les différents diagrammes selon l'étape de la conception

	Point de vue	Diagramme UML	
ANALYSE FONCTIONNELLE	**Statique – non objet** **Analyse générale**	**Cas d'utilisation**	
	Dynamique – non objet **Analyse détaillée**	**Séquence**	
		Activités	
		Etats-transitions	
ANALYSE DES DONNEES	Statique – non objet	MEA équivalent Classes	**D O N N E E S**
	Statique – objet	Classes-métier	
ANALYSE TECHNIQUE **Le MVC**	Statique – objet	Classes	
		Objets	
	Dynamique - objet	Séquence	
		Communication	
	Dynamique Plutôt non objet	Etats-transitions	
		Activités	

Synthèse des diagrammes utilisés dans ce cours (slide 35)

Type d'analyse	Types de diagramme
Foctionnelle	**Cas d'utilisation** **Séquence système** **Activités**
Données	**Classes métier**
Technique : MVC	**Séquence objet**

Bilan (slide 36)

- Le **diagramme de classe UML** reste un **outil indispensable** pour tous ceux qui ont compris que programmer objet, c'est d'abord concevoir objet.
  - ➔ De plus, il permet de faire de la modélisation de BD et des classes métier.

- Le **diagramme de cas d'utilisation (use case, UC)** est un outil d'analyse fonctionnelle et d'architecture fonctionnelle très efficace.
  - ➔ Aujourd'hui, les UC sont remplacés, dans certaines situations, par les User Story agile (US) ou par les scénarios UX (User eXpérience).

- Les **diagrammes de séquences et d'activités** un permettent de détailler l'analyse.

Perspectives (slide 37)

- Principaux concurrents d'UML :
  - L'intuition
  - Le langage naturel
  - La facilité (relative) offerte par les **frameworks**
  - Les **méthodes agiles**
  - L'**approche UX**

- Critique d'UML 2024 :
  - **https://jaystechbites.com/posts/2024/uml-alternatives-in-2024/**

6. Bibliographie (slide 38)

Référence du cours

- **UML 2, de l'apprentissage à la pratique – Cours et exercices, Laurent Audibert, 2009 (UML 2)**
  Ouvrage proche du cours proposé. Beaucoup d'exercices élémentaires. Application en java.
  Accès internet :
  http://laurent-audibert.developpez.com/Cours-UML/

Ce cours

- **http://bliaudet.free.fr/rubrique.php3?id_rubrique=29**