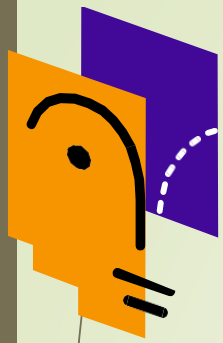
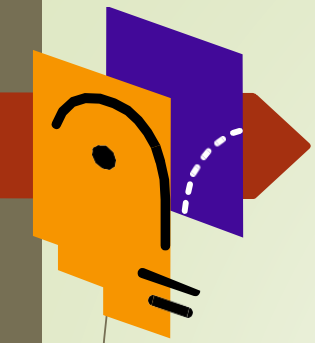


Modélisation UML
Diagrammes de Cas d'utilisation

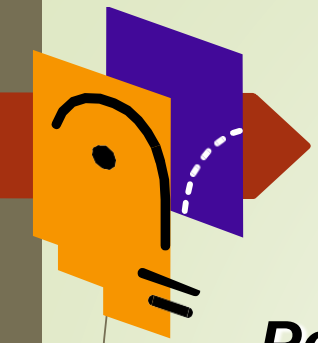
Mohamed Nemiche
nemiche@uv.es



Modélisation Objet : UML

- **Pour programmer une application (développer un logiciel), il ne convient pas de se lancer tête baissée dans l'écriture du code : il faut d'abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation.**

- **C'est cette démarche antérieure à l'écriture que l'on appelle *modélisation* ; son produit est un *modèle***



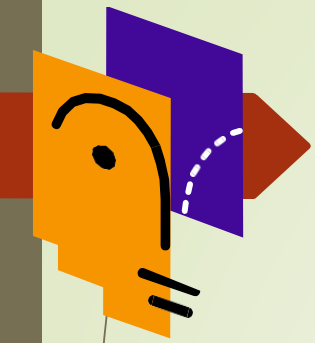
Modélisation Objet : UML

Pourquoi modéliser

Un modèle est une simplification de la réalité qui permet de mieux comprendre le système à développer.

Il permet :

- De visualiser le système comme il est ou comme il devrait l'être.
- De valider le modèle vis à vis des clients
- De spécifier les structures de données et le comportement du système.
- De fournir un guide pour la construction du système.
- De documenter le système et les décisions prises.



Modélisation Objet : UML

- **L'objectif de UML est d'assister le design et le développement du logiciel**
  - C'est un *langage de modélisation*, pas une *méthode*

Historique

Début des années 1990

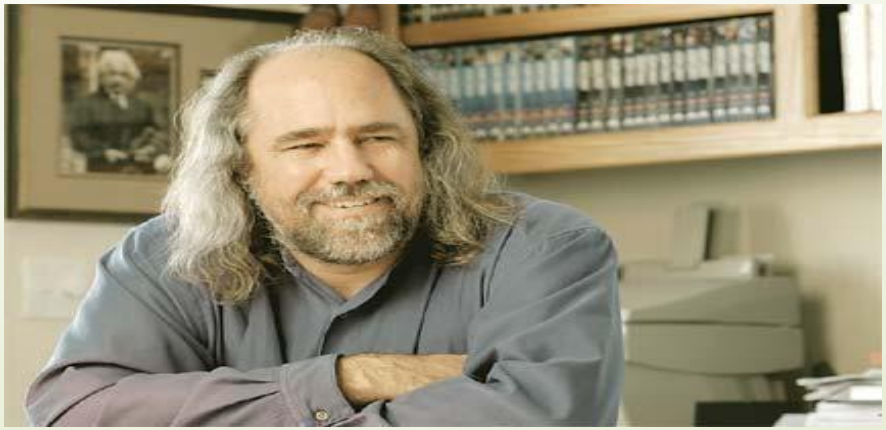
- les premiers processus de développement **OO** apparaissent

- **Entre 1990 et 1994 : Plus de 50 méthodes objet sont apparues:**
  - méthode **OOD** de Grady Booch (1991)
  - méthode OMT de James Rumbaugh (1991)
  - méthode OOSE de Ivar Jacobson (1991)
  - méthode OOA/OOD de Coad and Yourdon (1992)
  - méthode de Schlaer and Mellor (1992)
  - Etc.

Grady Booch et OOD

Description

- **OOD** signifie « Object Oriented Design ».
- Cette méthode a été créée en **1993** par Grady Booch, alors qu'il travaillait chez General Electric pour faciliter la phase de conception orientée objet des gros projets.
- Cette méthode propose des vues logiques et physiques du système.



Ivar Jacobson et OOSE

Description

- **OOSE** signifie « Object Oriented Software Engineering ».
- Cette méthode, créée en **1995** par Ivar Jacobson dans le cadre de ses activités chez Ericsson, introduit la notion de *use-cases* (cas d'utilisation).

John Rumbaugh et OMT

Description

- **OMT** est l'acronyme de « Object Modeling Technique ».
- John Rumbaugh a créé cette méthode en **1996** et a commercialisé un logiciel appelé **Rational Rose** (de la société Rational Rose Software) qui est une référence dans le domaine de la modélisation.
- Cette méthode propose des vues statiques, dynamiques et fonctionnelles d'un système.

Historique

Fin 1994

- G. Booch rejoint J. Rumbaugh chez Rational Software
- **OMT + OOD ➔ Unified Method *(oct 1995)***

Fin 1995

- I. Jacobson les rejoint chez Rational Software
- **Unified Method + OOSE ➔ UML 0.9 *(juin 1996)***

Début 1997

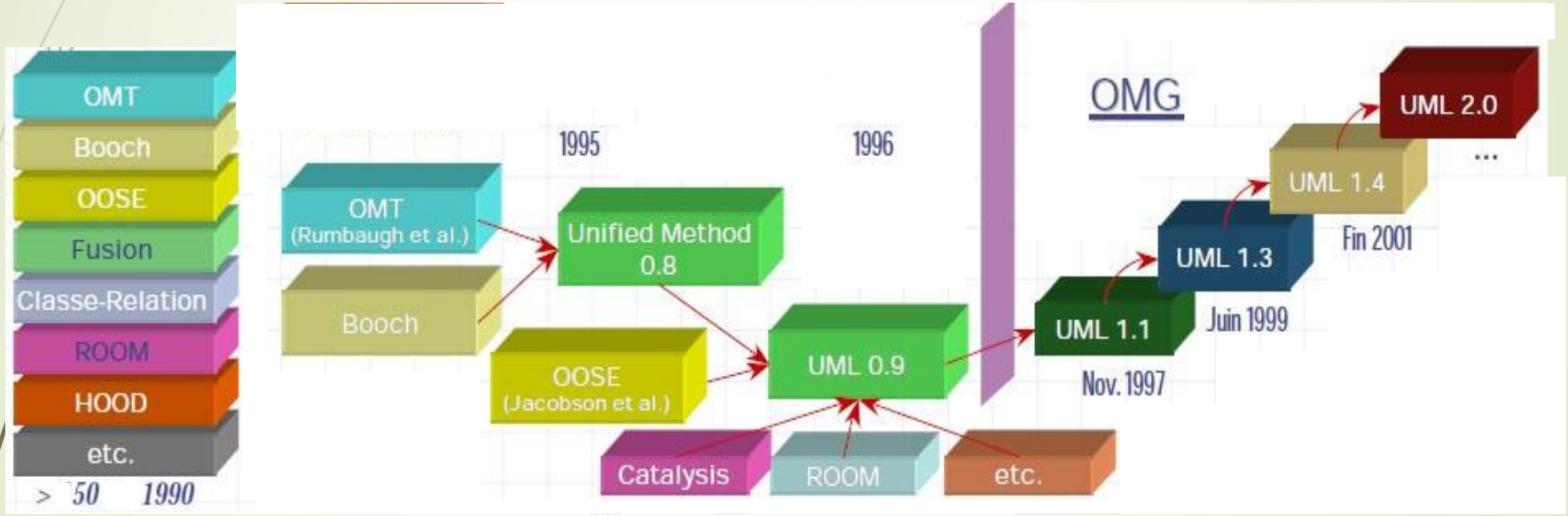
- Partenaires divers : Microsoft, Oracle, IBM, HP et autres leaders collaborent
- **➔ UML 1.0 *(jan 1997)***

Fin 1997

- l'OMG (Object Management Group) retient UML 1.1 comme norme de modélisation

- Version 2.0 en septembre 2004, Version 2.4.1 en aouˆt 2011.

Historique


OMT
Booch
OOSE
Fusion
Classe-Relation
ROOM
HOOD
etc.
> 50
1990
1995
1996
OMT
(Rumbaugh et al.)
Booch
Unified Method
0.8
OOSE
(Jacobson et al.)
UML 0.9
Catalysis
ROOM
etc.
OMG
UML 1.1
Nov. 1997
UML 1.3
Juin 1999
UML 1.4
Fin 2001
UML 2.0
...

L'arrivée d'UML

La normalisation

- **UML** devient une norme de l'OMG en **1997**.
- L'OMG (Object Management Group) est un organisme créé en 1989 afin de promouvoir des standards (comme CORBA par exemple) qui garantissent l'interopérabilité entre des applications orientées objet développées sur des réseaux hétérogènes.
- Cet organisme a été créé et est soutenu par des industriels comme HP, Sun, Unisys, American Airlines, Philips …

L'arrivée d'UML

Qu'est-ce qu'UML ?

UML : Unified Modeling Language

- Langage de Modélisation Unifié.
- Appliqué à l'analyse et à la conception des logiciels.
- Langage essentiellement graphique.
- Facile à lire et à comprendre.

En clair

- UML: norme qui définit les diagrammes et les conventions à utiliser lors de la construction de modèles décrivant la structure et le comportement d'un logiciel.
- Les modèles sont des diagrammes constitués d'éléments graphiques et de texte.
- UML n'est pas une méthode, mais un langage.

Modélisation Objet : UML

Les différents diagrammes

- **UML** propose 13 types de diagrammes (UML 2).
- Ces diagrammes sont présentés dans la norme sous forme d'un diagramme de classes afin de mettre en évidence les deux types de diagrammes :
  - les diagrammes de structure pour modéliser l'aspect **statique** d'un système : les diagrammes de structure montrent les différents objets trouvés dans un système
  - Les diagrammes de comportement pour modéliser l'aspect plutôt **dynamique** d'un système . Ces diagrammes modelisent la façon dont les objets du système interagissent les uns avec les autres

L'utilisation de diagrammes

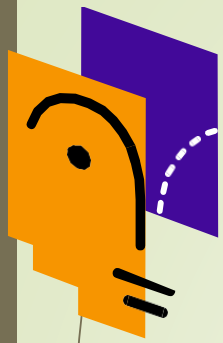
UML permet de définir et de visualiser un modèle, à l'aide de diagrammes :

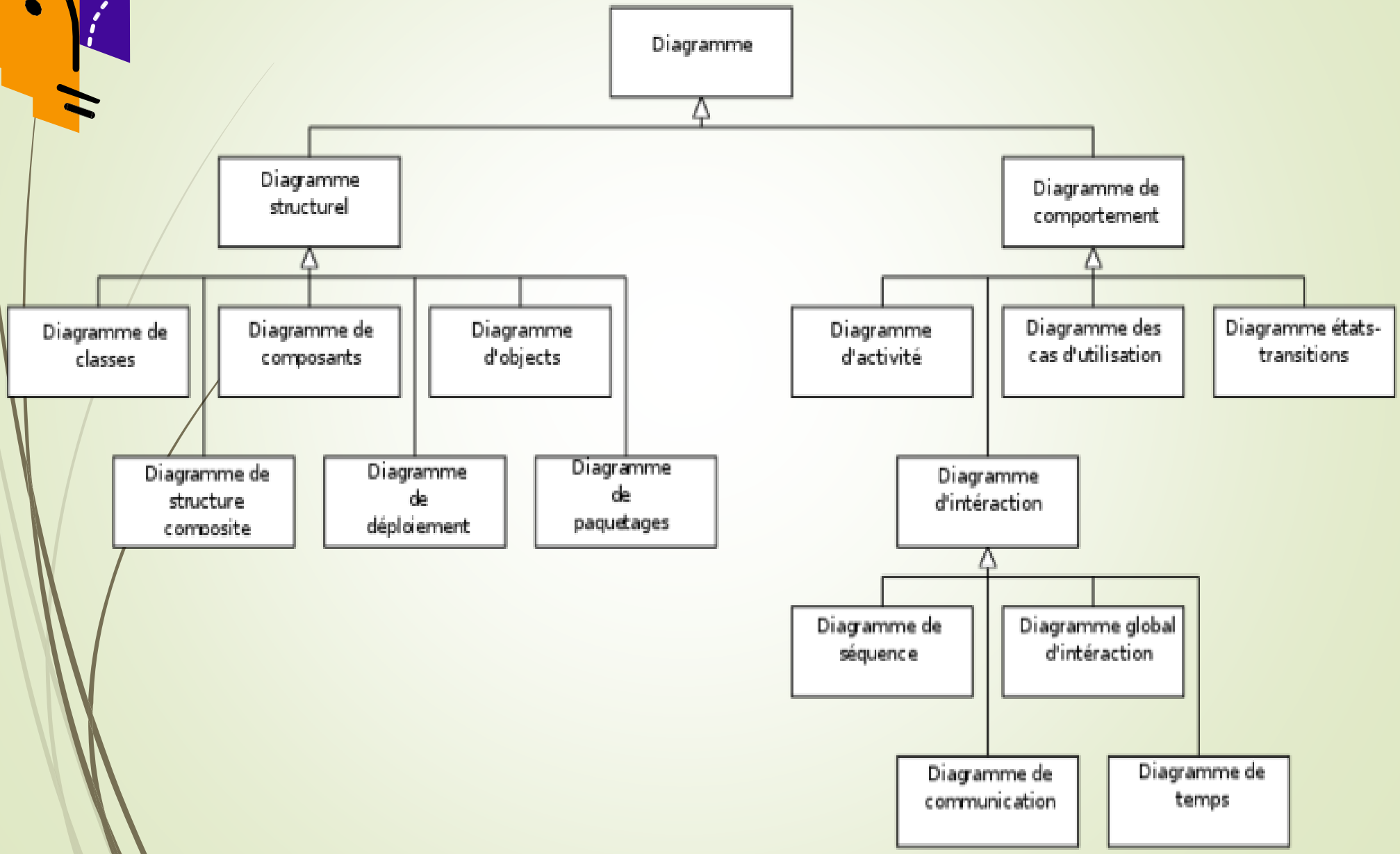
- **Définition d'un diagramme**
- **Caractéristiques des diagrammes UML**
- **Les différents types de diagrammes UML**

Définition d'un diagramme

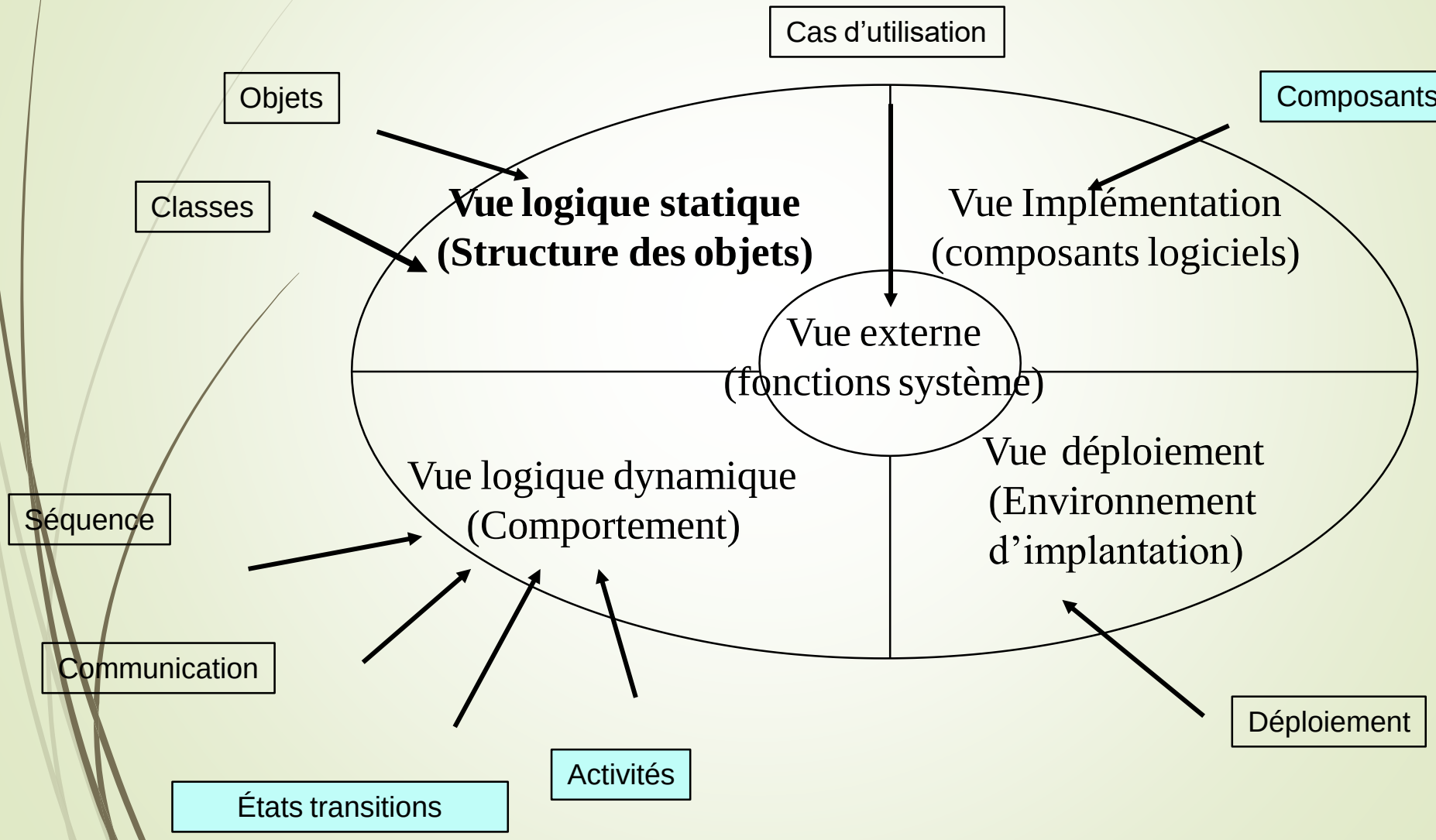
- Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle;
- Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis);
- Un type de diagramme UML offre toujours la même vue d'un système (il véhicule une sémantique précise);
- Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.

Modélisation Objet : UML

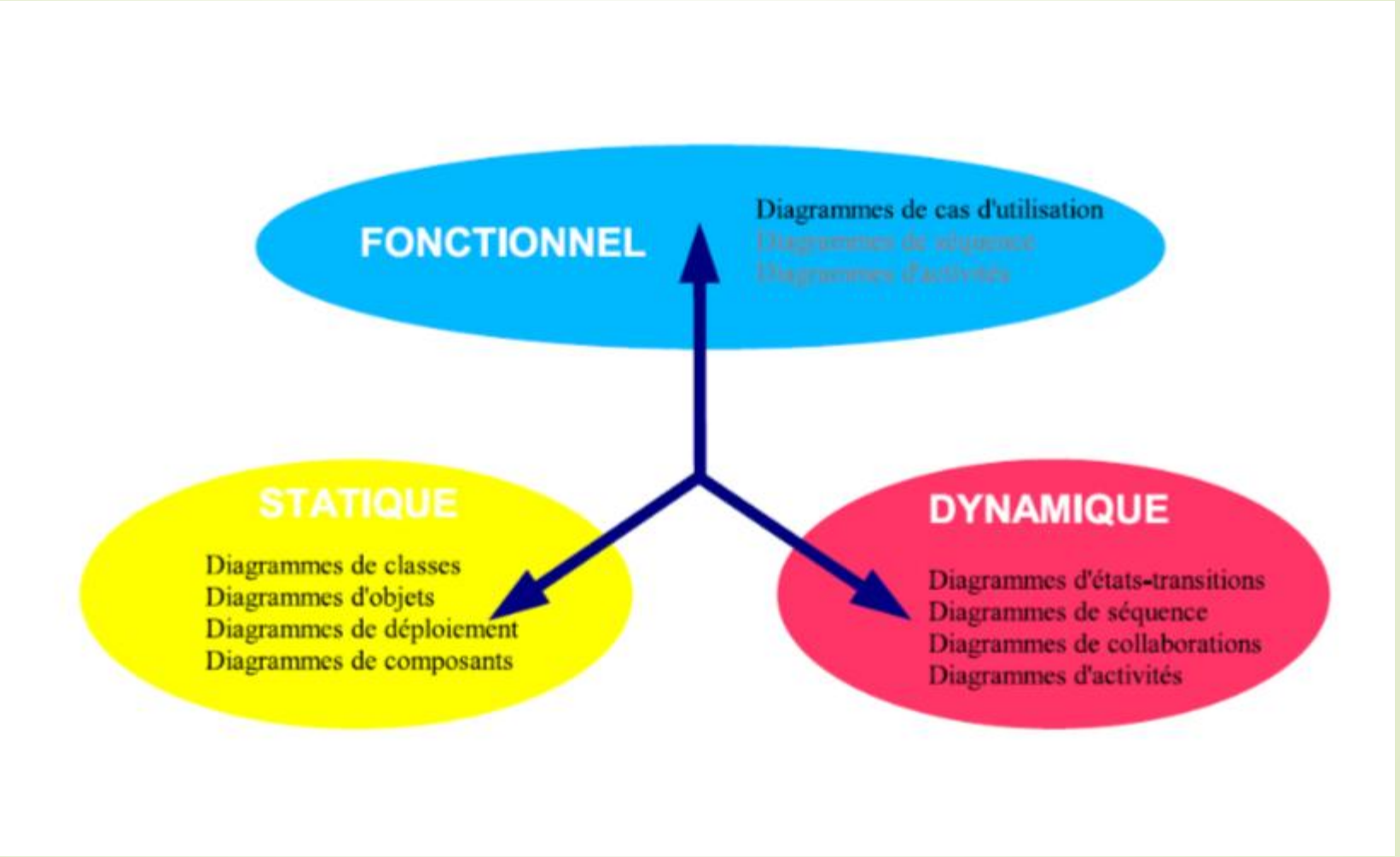




Modélisation Objet : Diagrammes et vue d'architecture



Diagrammes selon les axes de modélisation



Logiciels de modélisation UML

- **Il existe de nombreux outils logiciels de modélisation UML.**
- **Aucun d'entre eux ne respecte strictement aucune des versions de UML, particulièrement UML2**
- **Logiciels open-source: ArgoUML, Papyrus UML, StarUML, BOUML…**
- **Logiciels payants: Rational Rose ,EDGE Diagrammer, Visual Paradigm, PowerAMC, …**

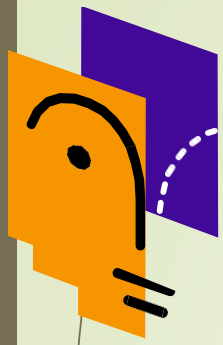


Diagramme de cas d'utilisation (Use Case Diagram)

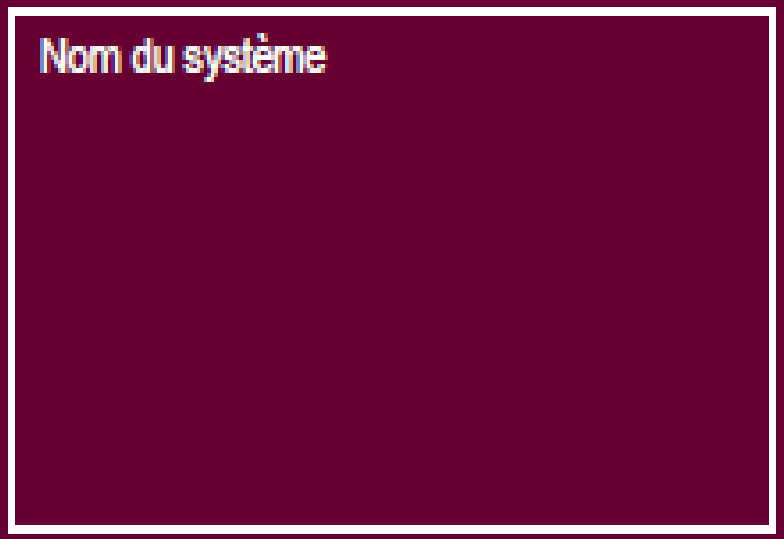
Diagramme de cas d'utilisation

- Un **diagramme de cas d'utilisation** :
  - décrit
    - les **acteurs** 
    - les **cas d'utilisation** 
    - le **système** 
  - contient
    - des **descriptions** textuelles

Le système

- Le système est un ensemble de cas d'utilisation
- Le système ne comprend pas les acteurs.

Nom du système



Acteur

- Élément extérieur au système qui interagit avec le système
- Rôle typique joué par un humain ou un système connexe par rapport au système

  Ex. : un client, un guichetier, un responsable maintenance, …
- Une même personne peut jouer plusieurs rôles

  Ex. : Maurice est directeur mais peut faire le guichetier
- Plusieurs personnes peuvent jouer un même rôle

  Ex. : Paul et Pierre sont deux clients
- Un acteur n'est pas forcément un être humain (dispositif matériel, …)

  Ex. : un distributeur de billets peut être vu comme un acteur
- Un acteur exécute un ou plusieurs cas d'utilisation

Acteur

- Est représenté par:
  - un petit bonhomme (*stick man*) avec son nom dessous ou
  - par un rectangle contenant le mot-clé << actor>> avec son nom dessous ou
  - Par un mélange de ces 2 représentations

Nom acteur

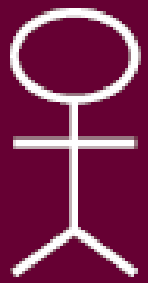
acteur humain

acteur non humain

- Pour les identifier :
  - Quelles sont les entités externes au système qui interagissent directement avec le système ?

Description des acteurs

- Pour chaque acteur :
  - choisir un identificateur représentatif de son rôle
  - donner une brève description textuelle



Guichetier

Un guichetier est un employé de la banque.
Interface entre le système informatique et les clients.
Peut effectuer une série d'opérations : création d 'un compte, dépôt et retrait d 'argent, etc.

Utilité des acteurs

- La définition d'acteurs permet
  - d'identifier les cas d'utilisation

    Ex. : que peut faire un guichetier ? un client ? le directeur ?
  - de voir le système de différents points de vues
  - de déterminer des droits d'accès par type d'acteur
  - de fixer des ordres de priorité entre acteurs
  - ...

Cas d'utilisation

- Un cas d'utilisation (*use case*) décrit:
  - Une fonctionnalité du système vue par un acteur (et utile à ce dernier)
  - une suite d'interactions entre un utilisateur et le système qui produit un résultat observable et intéressant pour un acteur particulier
  - Un comportement attendu du système du point de vue d'un ou de plusieurs acteurs
  - Un service rendu par le système

Cas d'utilisation (quelques caractéristiques)

- **Un cas d'utilisation N'EST PAS un diagramme, NI un symbole dans un diagramme….**

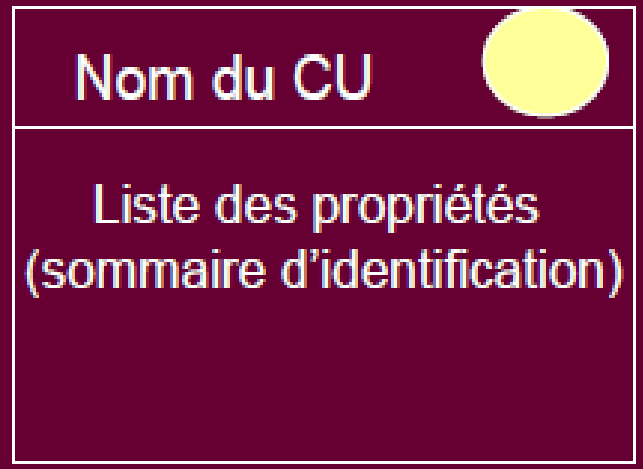
… c'est une manière de décrire un scénario d'interaction entre utilisateur et système ..

… Les diagrammes viennent après (ou avant) et représentent une vision générale des cas d'utilisation, ses relations avec les acteurs et avec d'autre cas d'utilisation.

Cas d'utilisation

- Est représenté par un ovale. Le nom du cas est inclus dans l'ellipse ou figure en-dessous
- Relié par des associations « participe à » à ses acteurs
- L'ensemble des cas d'utilisation décrit exhaustivement les fonctionnalités du système (1 CU : 1 fonction métier du système)
- Pour les identifier : par acteur, quelles sont les différentes manières d'utiliser le système ?

Nom du CU



Description des cas d'utilisation

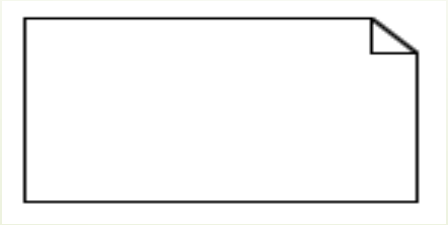
- Pour chaque cas d 'utilisation
  - choisir un identificateur représentatif : commencer par un verbe à l'infinitif, puis un complément du point de vue de l'acteur (pas du point de vue du système)
  - réaliser une description détaillée plus ou moins formelle : préciser ce que fait le système, ce que fait l'acteur
- Les cas d'utilisation ne doivent pas se chevaucher
- Si un cas d'utilisation concerne beaucoup d'acteurs, il faut probablement le découper en plusieurs CU

DESCRIPTION TEXTUELLE D'UN CU (en résumé)

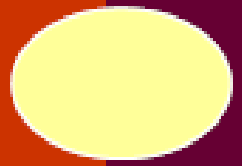
- Non normalisé par UML
- Exemple:
  1) **Identification :**
     - Titre
     - Résumé
     - Dates de création/modification
     - Version
     - Responsable
  2) **Description des scénarios** (nominal, alternatifs, d'erreur) + préconditions + postconditions

Description textuelle des cas d'utilisation

- **Une description textuelle d'un cas d'utilisation comprend:**
  - Les acteurs
  - Les pré-conditions: L'ensemble des conditions qui doivent être satisfaites avant de déclencher le cas d'utilisation
  - Les post-conditions: L'état du système après le déroulement du cas d'utilisation



Exemple de description détaillée d'un CU: sommaire d'identification



Retirer de l'argent

Titre : Retirer de l'argent
Résumé : CU qui permet à un client de la banque de retirer de l'argent
Acteurs : client
Précondition :
Le distributeur contient des billets, il est en attente d'une opération, il n'est ni en panne, ni en maintenance
Début : lorsqu 'un client introduit sa carte bancaire dans le distributeur.
Fin : lorsque la carte bancaire et les billets sont sortis.
Postcondition :
Le GAB contient moins de billets qu'au début du CU.
Transaction de retrait enregistrée par le GAB.

Scénario

- **Un Scénario et une succession d'actions et réactions entre les utilisateurs (acteurs) et le système.**
  - Par exemple
    - Le Porteur de carte introduit sa carte dans le lecteur de cartes du GAB.
    - Le GAB vérifie que la carte introduite est bien une carte bancaire.
    - Le GAB demande au Porteur de carte de saisir son code d'identification.
    - Le Porteur de carte saisit son code d'identification.
    - ……

Exemple de description détaillée d'un CU: description des scénarios

Retirer de l'argent

Scénario nominal = déroulement normal :

(1) le *client* introduit sa carte bancaire
(2) le *système* lit la carte et vérifie si la carte est valide
(3) le *système* demande au client de saisir son code
(4) le *client* saisit son code confidentiel
(5) le *système* vérifie que le code correspond à la carte
(6) le *client* choisit une opération de retrait
(7) le *système* demande le montant à retirer

...

Scénarios d'erreurs (se terminent brutalement) :

•*Carte invalide* : au cours de l 'étape (2) si la carte est jugée invalide, le système affiche un message d'erreur, rejette la carte et le cas d 'utilisation se termine.

•*Code erroné (pour la 3ème fois)* : au cours de l 'étape (5) ...

Scénarios alternatifs

•Montant demandé supérieur au solde du compte

•Code erroné (pour la 1ère ou 2ème fois)

Les scénarios

- **Un scénario peut être présenté dans un tableau de la forme suivante:**

Actions des acteurs	Actions du système
1. L'acteur déclanche…	2. Le système répond…
3. l'acteur choisi…	4. Le système répond…
….	…..

Scénario

- CU = ensemble de scénarios d'utilisation d'un système reliés par un but commun du point de vue d'un acteur
- Le CU a un début et une fin identifiés
- Scénario = succession particulière d'enchaînements, s'exécutant du début à la fin du CU.
- Un enchaînement = unité de séquence d'actions
- Un CU contient en général:
  - un scénario nominal
  - différents scénarios alternatifs (qui se terminent de façon normale)
  - des scénarios d'erreur (qui se terminent en échec)

L'objectif de l'acteur principal est atteint

Acteurs principaux et secondaires

- Acteur principal d'un CU
  - Celui pour qui le CU produit un résultat observable
  - A gauche des CU
  - Rôle indiqué éventuellement sur l'association côté acteur : <<principal>> (valeur par défaut)
- Acteur secondaire d'un CU
  - Celui pour qui le CU ne produit pas un résultat observable par l'utilisateur
  - Souvent sollicités pour des informations complémentaires
  - Peuvent uniquement consulter ou informer le système (pas d'objectif à part entière de la part de l'acteur secondaire)
  - A droite des CU
  - Rôle indiqué éventuellement sur l'association côté acteur : <<secondaire>>
  - Ex. : système d'authentification appelé par le distributeur de billets

Relations entre acteurs et cas d'utilisation

- Relation d'association (« participe à ») :
  Lien entre un acteur et un cas d'utilisation qu'il peut exécuter
- Un cas d'utilisation doit être relié au moins à un acteur

Relations entre cas d'utilisation

Relations entre cas d'utilisations : permettent la structuration des cas d'utilisation

- **Il existe 3 types de relations entre cas d'utilisation :**
  - la relation d'inclusion (*include*)
  - la relation d'extension (extends)
  - la relation de généralisation

Relation entre cas d'utilisation

- ***La relation d'inclusion :***

  Le cas d'utilisation source inclue c-a-d contient obligatoirement le comportement du cas d'utilisation destination

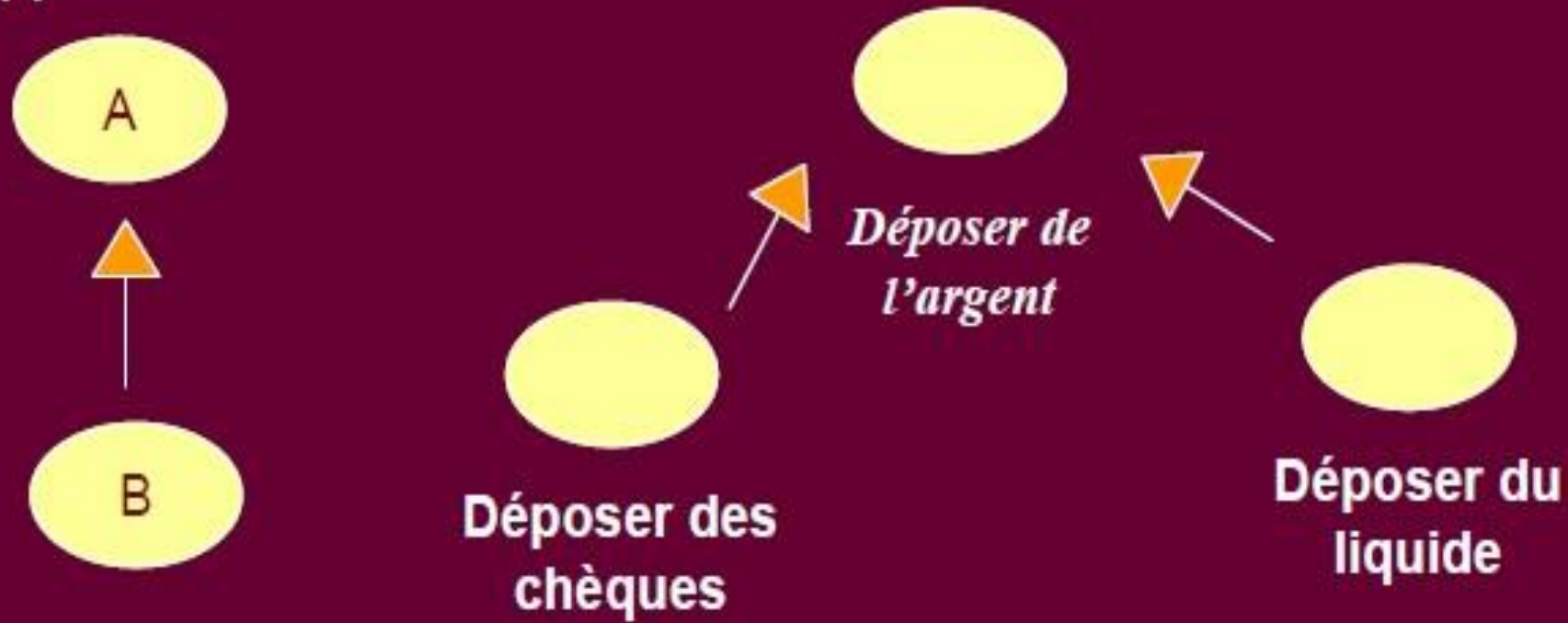
Relation entre cas d'utilisation

- ***La relation d' extension :***

  Le cas d'utilisation source étend cad ajoute son comportement (optionnellement) au comportement du cas d'utilisation destination

Relations entre cas d'utilisation (3)

- Relation de généralisation
- Un cas A est une généralisation d'un cas B si B est un cas particulier de A



- Déposer de l'argent est cas d'utilisation généralisé. Il devient abstrait (italique) car il ne s'instancie pas directement, mais uniquement par le biais de l'un des 2 cas spécialisés

Relations entre acteurs

- Uniquement relation de généralisation/spécialisation
- A est une généralisation de B si tous les CU accessibles à A sont accessibles à B, mais pas l'inverse

Structuration en package

- Si les cas d'utilisation sont nombreux
  - les regrouper **par acteur**
    - Opérations client
    - Opérations administrateur
    - etc.
  - les regrouper **par domaine fonctionnel**
    - Gérer les contrats
    - Gérer les clients
    - Etc.

ETAPES DE LA MODELISATION FONCTIONNELLE

- Identification des acteurs
- Identification des CU (par acteur)
- Réalisation des diagrammes de CU
- Description textuelle des CU (scénarios) *
- Organisation des CU (relations entre CU + packages)

PUIS description graphique * * des CU :

- Modélisation dynamique:
  - diagramme de séquence système
  - diagramme d'activité

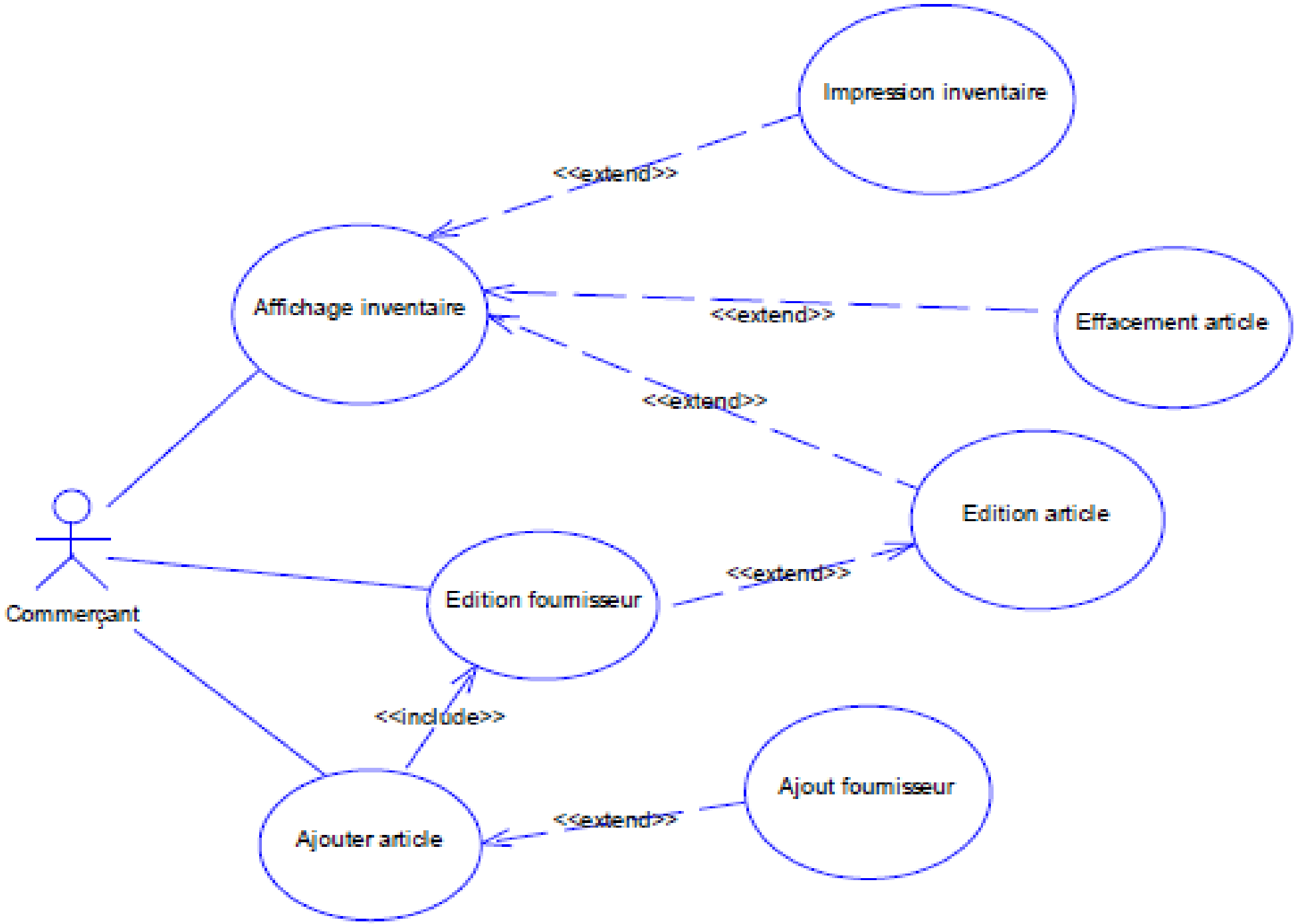
* Pour communiquer facilement avec les utilisateurs et s'entendre sur la terminologie métier employée

* * Pour mieux montrer la succession des enchaînements

Exemple

- **Dans un magasin, un commerçant dispose d'un système de gestion de son stock d'articles, dont les fonctionnalités sont les suivantes :**
  - Edition de la fiche d'un fournisseur
  - Possibilité d'ajouter un nouvel article (dans ce cas, la fiche fournisseur est automatiquement éditée. Si le fournisseur n'existe pas, on peut alors le créer)
  - Edition de l'inventaire. Depuis cet écran, on a le choix d'imprimer l'inventaire, d'effacer un article ou d'éditer la fiche d'un article).

Modéliser cette situation par un diagramme de cas d'utilisation


Impression inventaire
<<extend>>
Affichage inventaire
<<extend>>
Effacement article
<<extend>>
Edition article
Commerçant
Edition foumisseur
<<extend>>
<<include>>
Ajout foumisseur
<<extend>>
Ajouter article