UVS
UNIVERSITE VIRTUELLE DU SENEGAL

Analyse et Conception Orientée Objet

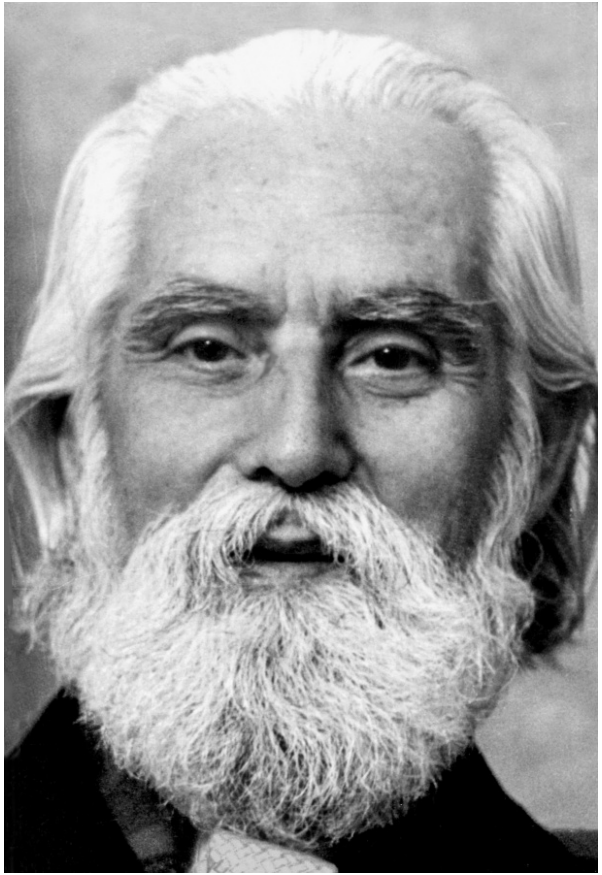
UML

Un peu d'histoire…

«Depuis des temps très anciens, les hommes ont cherché un langage à la fois universel et synthétique, et leurs recherches les ont amenés à découvrir des images, des symboles qui expriment, en les réduisant à l'essentiel, les réalités les plus riches et les plus complexes.

Les images, les symboles parlent, ils ont un langage.»

Omraam Mikhaël Aïvanhov
Philosophe bulgare
31.01.1900 – 25.12.1986



Plan

- UML
- Diagramme de cas d'utilisation
- Diagramme de classe
- Diagramme de séquence

Plan

- UML
- Diagramme de cas d'utilisation
- Diagramme de classe
- Diagramme de séquence

Qu'est ce qu'un modèle?

Un modèle est :

- Une représentation simplifiée d'une réalité.
- Une représentation abstraite de la réalité c'est-à-dire une abstraction.
- Une vue subjective mais pertinente de la réalité.

Par exemple: Un physicien modélisera la Lune comme un corps céleste ayant une certaine masse et se trouvant à une certaine distance de la Terre, alors qu'un poète la modélisera comme une dame avec laquelle il peut avoir une conversation.

Léopold Sédar Senghor….Lune

Joal !
Je me rappelle.
Je me rappelle les signares à l'ombre verte des vérandas
Les signares aux yeux surréels comme un clair de **lune** sur la grève.

Poème Joal
Léopold Sédar Senghor
[1906 - 2001]
poète, écrivain, homme d'État sénégalais

Crédit: http://www.oeuvresouvertes.net/autres_espaces/senghor.html

A quoi sert un modèle?

Les modèles ont différents usages:

- Ils facilitent la compréhension d'un objet complexe (fusée, bâtiment, économie, atmosphère, cellule, logiciel, système d'information, etc.).
- Ils optimisent l'organisation des systèmes (divisions, départements, services, etc.).
- Ils permettent de se focaliser sur des aspects spécifiques d'un système sans s'embarrasser des données non pertinentes.
- Ils permettent de décrire avec précision et complétude les besoins sans forcément connaître les détails du système.
- Ils facilitent la conception d'un système (la réalisation de maquette ou de plan approximative, à échelle réduite, etc.).
- Ils permettent de tester une multitude de solutions à moindre coût et dans des délaisréduits et de sélectionner celle qui résout les problèmes posés.
- Ils servent de documents d'échange ou de contrats entre clients et fournisseurs.

Le modèle

«Le modèle s'exprime sous une forme simple et pratique pour le travail»

James RUMBAUGH
Co-créateur d'UML

La modélisation

- L'expression d'un modèle, la modélisation, se fait dans un langage compatible avec le système modélisé et les objectifs attendus. Ainsi, le physicien qui modélise la lune utilisera les mathématiques comme langage de modélisation.

- Ainsi la modélisation est tout le processus de mise en œuvre d'un modèle.

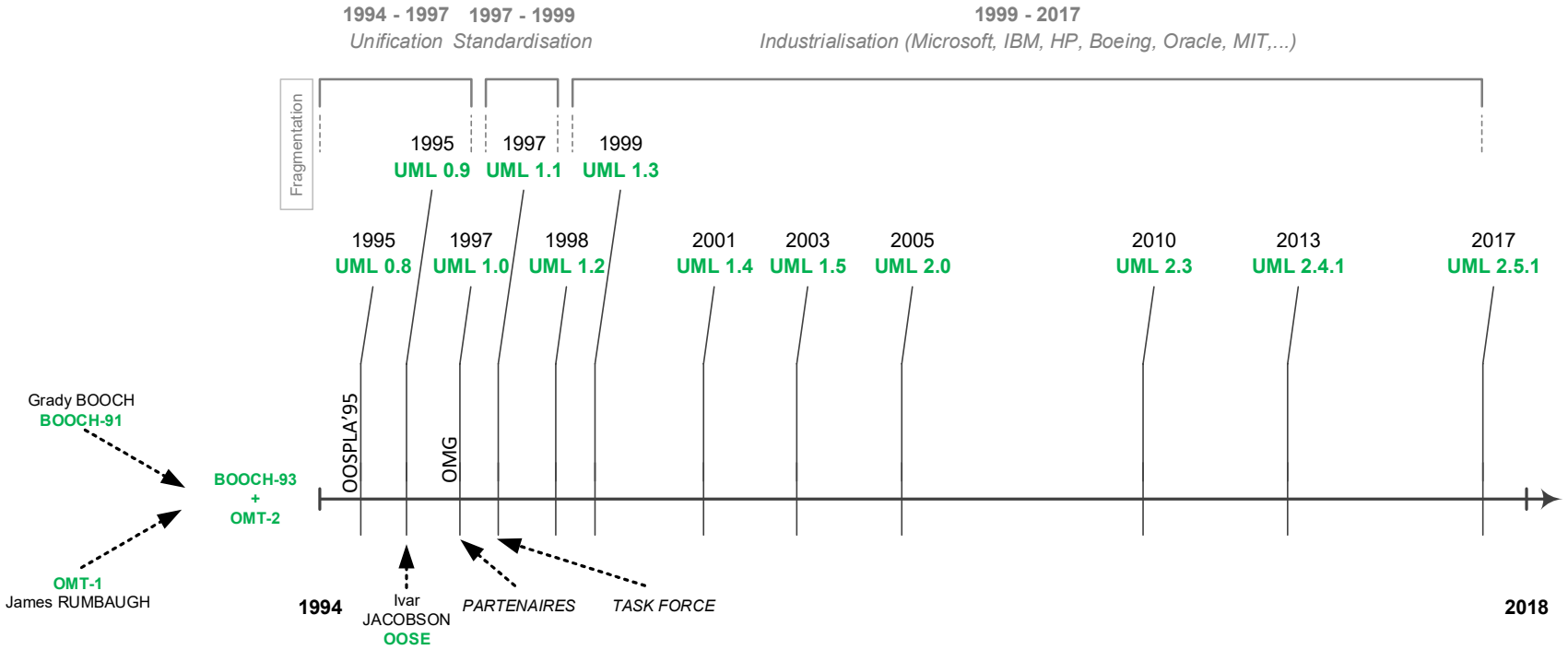
La modélisation Objet

- La modélisation objet produit des modèles permettant de regrouper un ensemble de configurations possibles du système et pouvant être implémentés dans un langage de programmation objet.
- La modélisation objet présente de nombreux avantages à travers un ensemble de propriétés (classe, encapsulation, héritage et abstraction, paquetage, modularité, extensibilité, adaptabilité, réutilisation) qui lui confère toute sa puissance et son intérêt.

C'est quoi UML?

- UML = **Unified Modeling Language** (i.e. Langage de Modélisation Unifié).
- UML est un langage de modélisation graphique.
- UML est apparu dans le cadre de l'Analyse et de la Conception Orientée Objet (ACOO).
- UML est né en 1995 de la fusion de trois méthodes qui ont le plus influencé la modélisation objet: OMT, Booch et OOSE.
- UML a été normalisé par l'OMG (Object Management Group), en fin 1997. OMG regroupe les plus grandes Entreprises du monde (Microsoft, IBM, HP, Boeing, Oracle, MIT,...).
- **UML 2.5.1** a été publié en **décembre 2017** et propose **14 types de diagrammes** qui permettent la modélisation d'un projet durant tout son cycle de vie.

Historique d'UML



OOSPLA - Object-Oriented Programming, Systems, Languages & Applications
OMG – Object Management Group

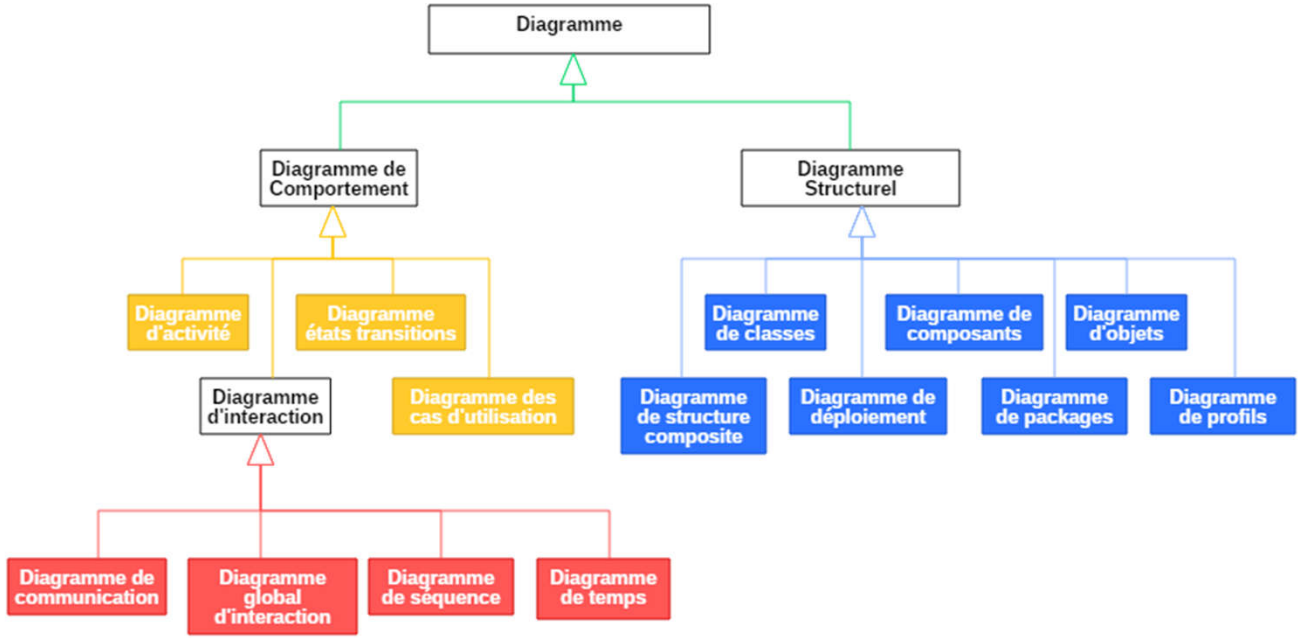
Définition d'UML

- **UML est un langage de modélisation pour :**
  - comprendre et décrire des besoins
  - spécifier des systèmes, simples ou complexes
  - concevoir et construire des solutions
  - documenter un système
  - communiquer entre les membres de l'équipe de projet

- **UML est un langage à usage général, quel que soit :**
  - le type de système - logiciel, matériel, organisation, etc.
  - le domaine métier - gestion, ingénierie, télécoms, etc.
  - le processus de développement - cascade, itératif, etc.

14 Diagrammes d'UML

Les diagrammes sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie. Il en existe quatorze depuis UML 2.3.



Diagrammes de structure ou statiques

Les diagrammes de structure (structure diagrams) ou diagrammes statiques (static diagrams) rassemblent :

1. **Diagramme de classes (class diagram) :** représentation des classes intervenant dans le système.
2. **Diagramme d'objets (object diagram) :** représentation des instances de classes (objets) utilisées dans le système.
3. **Diagramme de composants (component diagram) :** représentation des composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données…)
4. **Diagramme de déploiement (deployment diagram) :** représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage…) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.
5. **Diagramme des paquets (package diagram) :** représentation des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions.
6. **Diagramme de structure composite (composite structure diagram) :** représentation sous forme de boîte blanche les relations entre composants d'une classe (depuis UML 2.x).
7. **Diagramme de profils (profile diagram) :** spécialisation et personnalisation pour un domaine particulier d'un meta-modèle de référence d'UML (depuis UML 2.2).

Diagrammes de comportement ou fonctionnels

Les diagrammes de comportement (behavior diagrams) rassemblent :

8. **Diagramme des cas d'utilisation (use-case diagram) :** représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.

9. **Diagramme états-transitions (state machine diagram) :** représentation sous forme de machine à états finis le comportement du système ou de ses composants.

10. **Diagramme d'activité (activity diagram) :** représentation sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants.

Diagrammes d'interaction ou dynamiques

Les diagrammes d'interaction (interaction diagrams) ou diagrammes dynamiques (dynamic diagrams) rassemblent :

11. **Diagramme de séquence (sequence diagram) :** représentation de façon séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs.

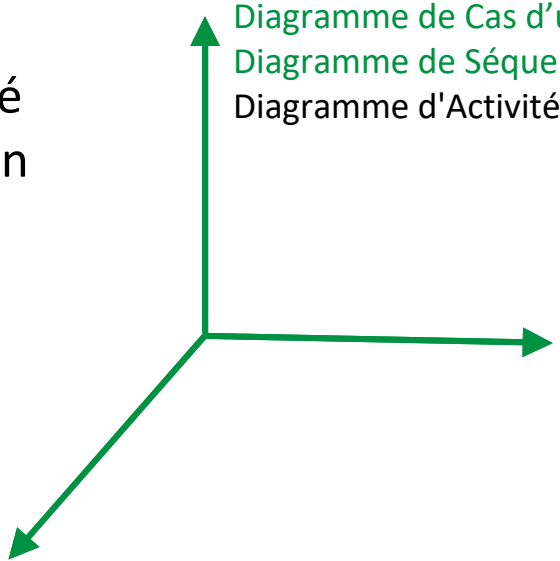
12. **Diagramme de communication (communication diagram) :** représentation de façon simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets (depuis UML 2.x).

13. **Diagramme global d'interaction (interaction overview diagram) :** représentation des enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité) (depuis UML 2.x).

14. **Diagramme de temps (timing diagram) :** représentation des variations d'une donnée au cours du temps (depuis UML 2.3).

3 axes de modélisation

Une modélisation classique s'effectue selon trois points de vue (fonctionnel, statique et dynamique), et dans la majorité des cas seuls les diagrammes en vert sont utilisés.



Fonctionnel

Diagramme de Cas d'utilisation
Diagramme de Séquence
Diagramme d'Activité

Dynamique

Diagramme d'États
Diagramme d'Activité
Diagramme de Séquence
Diagramme de Communication

Statique

Diagramme de Classes
Diagramme de Packages
Diagramme d'Objets
Diagramme de Structure Composite
Diagramme de Déploiement

Ce qu'est et n'est pas UML

- **Savoir lire et écrire des diagrammes UML ne signifie pas forcément que les compétences en Analyse et Conception OO sont acquises**
  - De la même manière, savoir lire et dessiner des plans ne fait pas forcément de nous des architectes ...
- **Ce qui est important c'est :**
  - Comment analyser et concevoir OO et "penser objets"
  - Savoir utiliser les bonnes règles de conception OO et évaluer une conception
  - Savoir attribuer les bonnes responsabilités aux bons objets
- **C'est ce que nous voulons vous enseigner dans ce cours !**

Plan

- UML
- Diagramme de cas d'utilisation
- Diagramme de classe
- Diagramme de séquence

Rappel…

«UML n'est qu'un langage et il ne sert ici qu'à formaliser les besoins, c'est-à-dire à les représenter sous une forme graphique suffisamment simple pour être compréhensible par toutes les personnes impliquées dans le projet. N'oublions pas que bien souvent, le client et les utilisateurs ne sont pas des informaticiens. Il leur faut donc un moyen simple d'exprimer leurs besoins. C'est précisément le rôle des diagrammes de cas d'utilisation. Ils permettent de recenser les grandes fonctionnalités d'un système.»

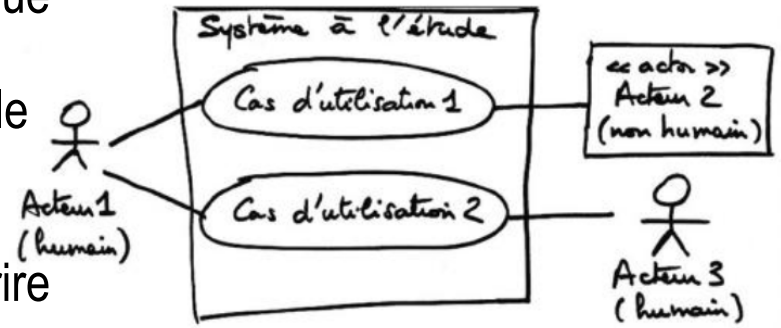
Diagramme de cas d'utilisation est....

une représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.

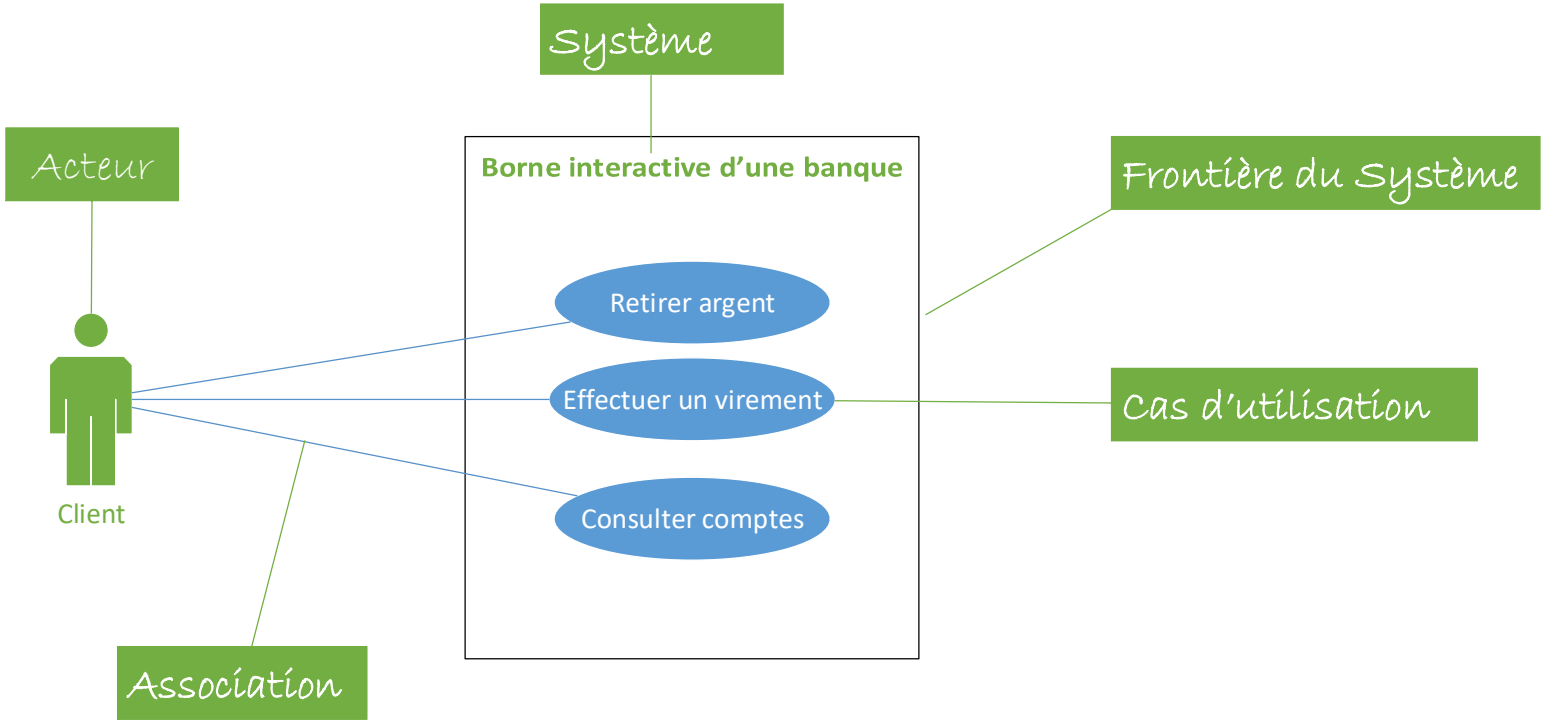
- Les acteurs modélisent tout ce qui interagit avec le système.

- Le diagramme de cas d'utilisation permet de décrire ce que le futur système devra faire, sans spécifier comment il le fera.

- Le cas d'utilisation («use case») c'est l'usage que les acteurs font du système.



Eléments du diagramme de cas d'utilization (1/)



Acteur

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié:

- Il peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.
- Il est celui qui bénéficie de l'utilisation du système.
- En réponse à l'action d'un acteur , le système fournit un service qui correspond à son besoin.
- Une même personne peut jouer le role de différents acteurs.
- Un acteur peut être un autre système.

Représentations graphiques possibles d'un acteur

La représentation graphique standard de l'acteur en UML est l'icône appelée stick man, avec le nom de l'acteur sous le dessin.

On peut également figurer un acteur sous la forme rectangulaire d'une classe, avec le mot-clé <<actor>>.

Une troisième représentation (intermédiaire entre les deux premières) avec l'utilisation d'un icône représentatif de l'acteur

Stéréotype

Un stéréotype représente une variation d'un élément de modèle existant.

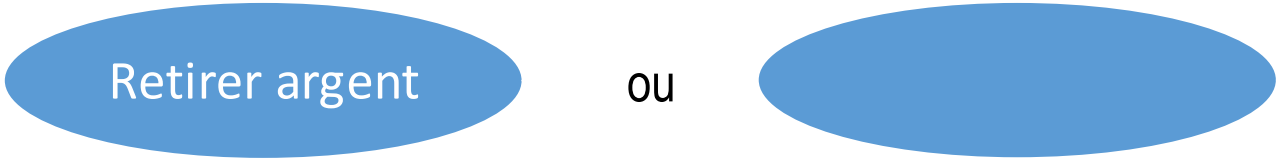
- Par exemple, le rectangle représente un stéréotype acteur. Les stéréotypes permettent d'adapter UML à des situations particulières.

Cas d'utilisation (Use case)

Un cas d'utilisation (« use case ») représente un ensemble d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

- L'ensemble des cas d'utilisations décrit les objectifs (le but) du système.
- Les cas d'utilisations permettent de structurer les besoins des utilisateurs.
- Les cas d'utilisations centrent l'expression des exigences du système sur ses utilisateurs.

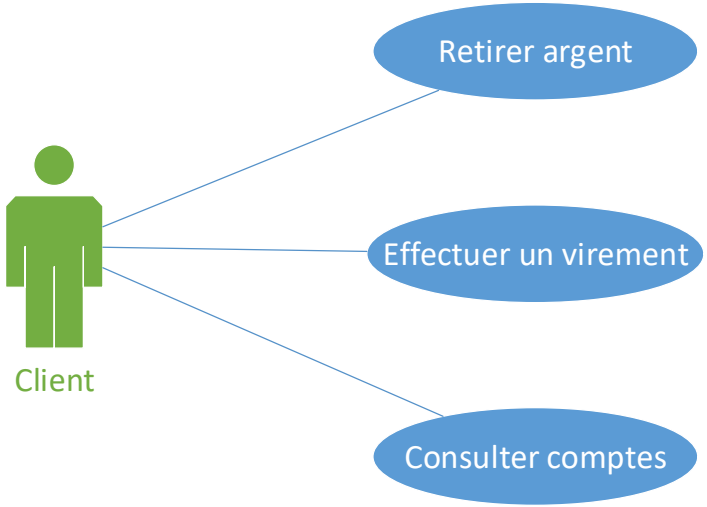
Un cas d'utilisation se représente par une ellipse. Le nom du cas est inclus dans l'ellipse la plupart du temps ou bien il figure dessous.



Relations entre cas d'utilisation et acteurs

Relation d'association

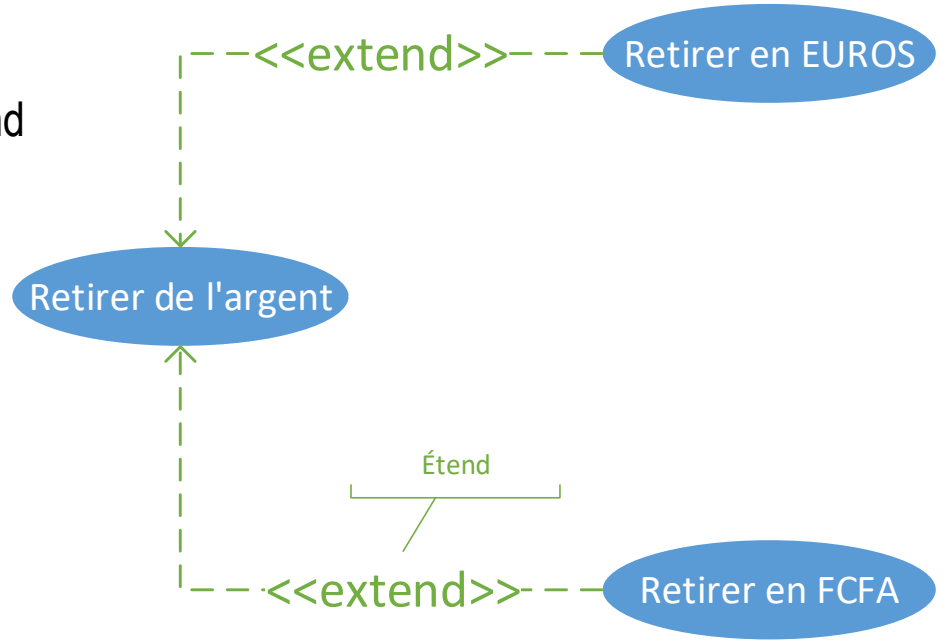
- Cette relation est un lien symbolisé par un trait (ou une flèche) partant de l'acteur au cas d'utilisation signifiant que l'acteur est le déclencheur (ou l'initiateur) du cas d'utilisation.
- Chaque association signifie simplement « participe à ». Un cas d'utilisation doit être relié à au moins un acteur.



Relations entre cas d'utilisation

Relation d'extension

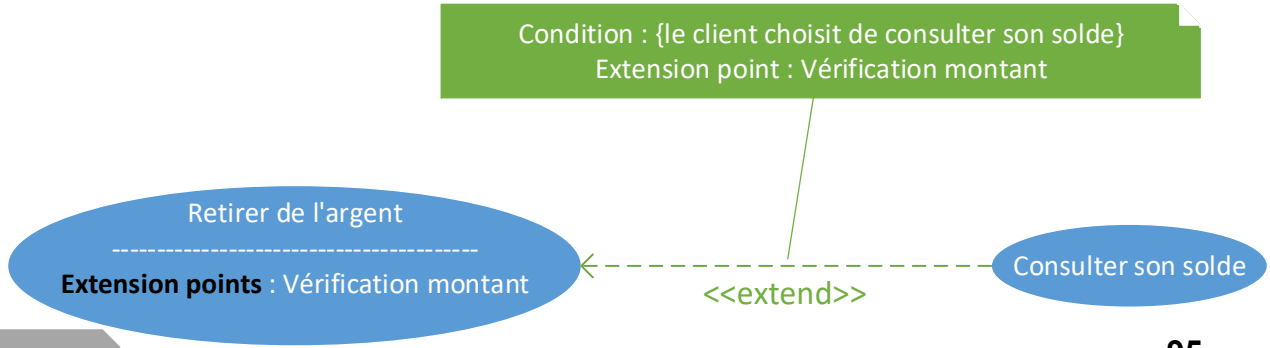
- Si le comportement de A peut être étendu par le comportement de B, on dit alors que B étend A. Une extension est souvent soumise à condition.
- Cette relation se représente par une flèche pointillée et un stéréotype «extend» pour étend, est souvent rajouté au dessus du trait.
- Graphiquement, le client a la possibilité au moment de retirer de l'argent de choisir entre des fcfa ou des euros.



Relations entre cas d'utilisation

Relation d'extension - Exemple

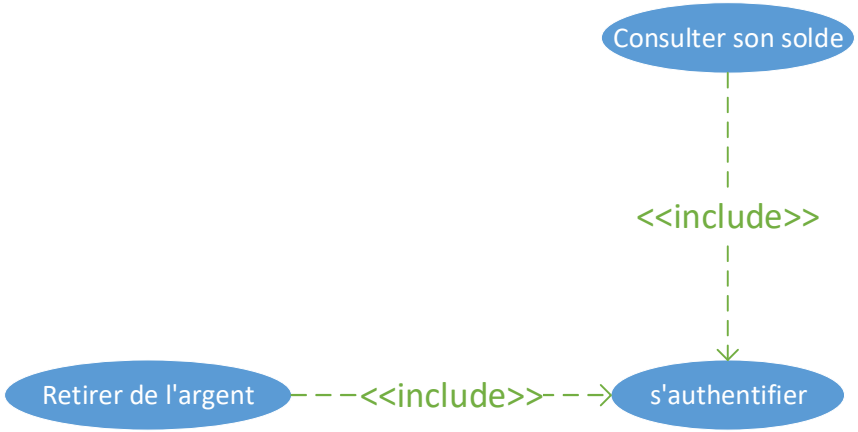
- Si le client souhaite pouvoir consulter son solde juste avant de choisir le montant de son retrait? Il pourrait ainsi ajuster le montant demandé avec ce qu'il lui reste à ce moment sur son compte.
- Si l'on retient ce nouveau besoin fonctionnel, pour le modéliser en UML il suffit d'ajouter une relation d'extension optionnelle comme cela est indiqué sur la figure suivante.
- Les deux cas d'utilisation peuvent bien sûr s'exécuter indépendamment, mais *Consulter le solde* peut également venir s'intercaler à l'intérieur de *Retirer de l'argent*, au point d'extension *Vérification montant.*



Relations entre cas d'utilisation

Relation d'inclusion

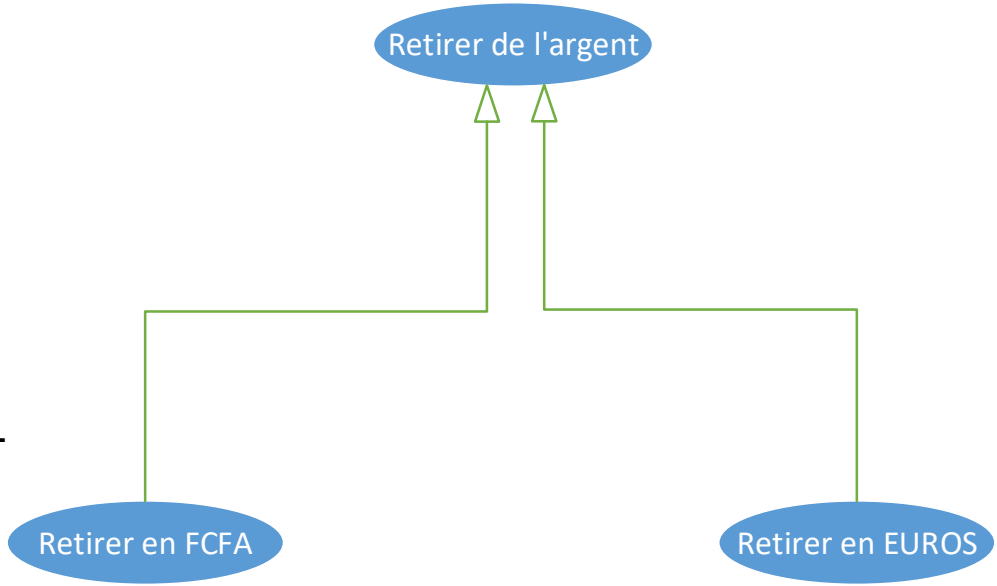
- Un cas B est inclus dans un cas A si le comportement décrit par le cas B est inclus dans le comportement du cas A : on dit alors que le cas A dépend de B.
- Cette relation se représente par une flèche pointillée et un stéréotype «include» pour inclut, est souvent rajouté au dessus du trait.
- Par exemple, l'accès aux informations d'un compte bancaire inclut nécessairement une phase d'authentification avec un mot de passe



Relations entre cas d'utilisation

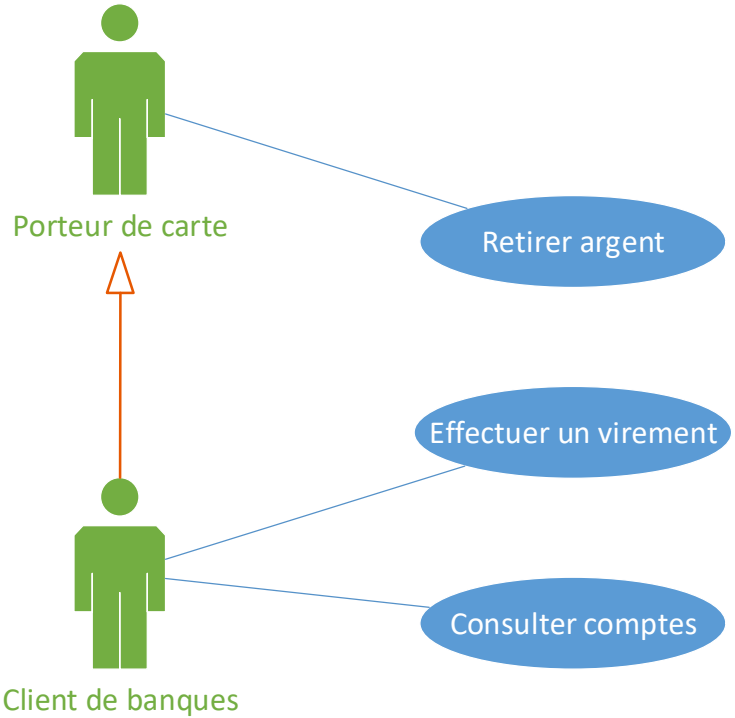
Relation de généralisation

- La relation de généralisation. Un cas A est une généralisation d'un cas B si B est un cas particulier de A.
- Cette relation se représente par une flèche.
- Par exemple, *Retirer des francs CFA* est un cas particulier de *retirer de l'argent*. Cette relation de généralisation/spécialisation est présente dans la plupart des diagrammes UML et se traduit par le concept d'héritage dans les langages orientés objet.



Relations entre acteurs

- La seule relation possible entre deux acteurs est la généralisation : un acteur A est une généralisation d'un acteur B si l'acteur A peut être substitué par l'acteur B (tous les cas d'utilisation accessibles à A le sont aussi à B, mais l'inverse n'est pas vrai).
- Par exemple, un *client de banque* peut faire tout ce qu'un *porteur de carte* peut faire. Par contre l'inverse n'est pas vrai.
- Le cas d'utilisation *Retirer de l'argent* a deux acteurs principaux possibles (mais exclusifs du point de vue de la simultanéité). Une autre façon de l'exprimer consiste à considérer l'acteur *Client de la banque* comme une spécialisation (au sens de la relation d'héritage) de l'acteur plus général *Porteur de carte*. Un *client de la banque* est en effet un *Porteur de carte* particulier qui a toutes les prérogatives de ce dernier, ainsi que d'autres qui lui sont propres en tant que client.



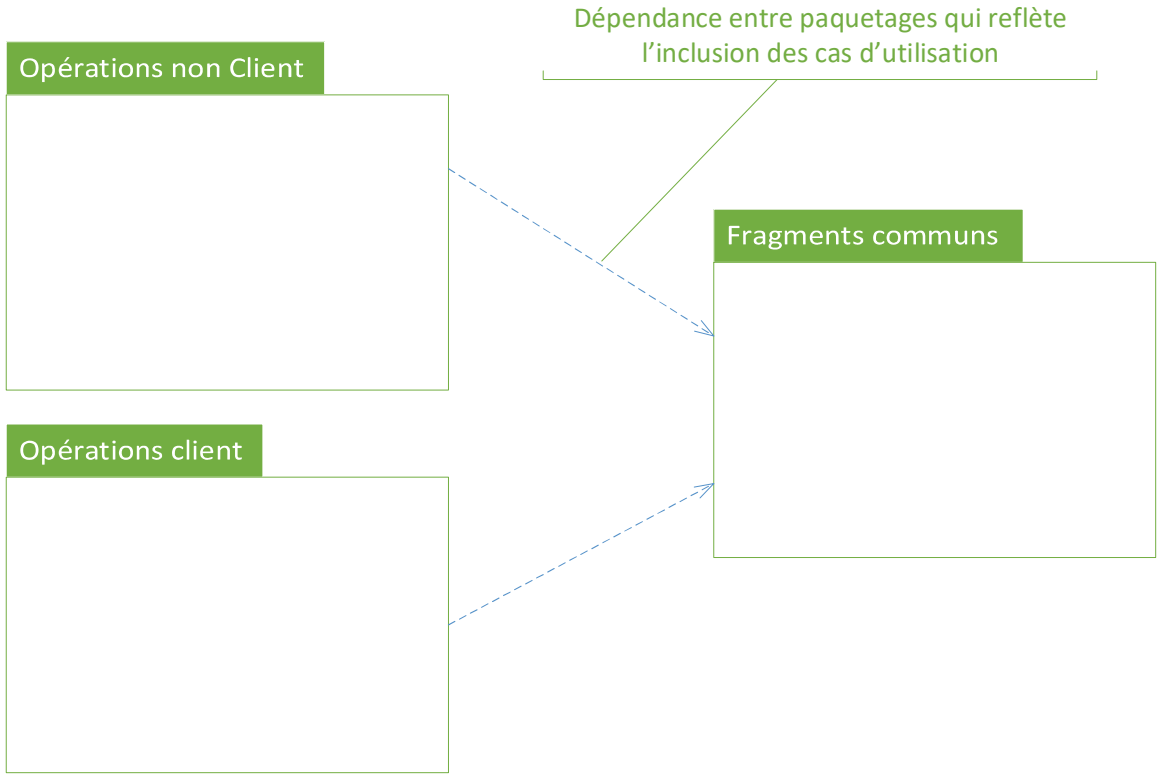
Regroupement des cas d'utilisation en paquetages (1/2)

- UML permet de regrouper des cas d'utilisation dans une entité appelée « paquetage ». Le regroupement peut se faire par acteur ou par domaine fonctionnel. Un diagramme de cas d'utilisation peut contenir plusieurs paquetages et des paquetages peuvent être inclus dans d'autres paquetages.

- En d'autres termes, un paquetage permet d'organiser des éléments de modélisation en groupe, selon des critères purement logique. Un paquetage peut contenir des cas d'utilisations, des classes, des interfaces, etc.

Regroupement des cas d'utilisation en paquetages (2/2)

- Trois paquetages ont été créés :
  - Opération non Client,
  - Opérations clients,
  - Fragments communs.



Description textuelle des cas d'utilisation

- Bien que de nombreux diagrammes d'UML permettent de décrire un cas, il est recommandé de rédiger une description textuelle car c'est une forme souple qui convient dans bien des situations.
- Une description textuelle couramment utilisée se compose de trois parties:
  - Identification
  - Description des scénarios
  - Rubriques optionnelles

Description textuelle - Identification

Cette première partie permet d'identifier le cas. Elle doit contenir:

- le nom du cas (Titre);
- un résumé de son objectif (Résumé) ;
- les acteurs impliqués (Acteurs) ;
- les dates de création et de mise à jour de la description courante ;
  - Date de création
  - Date de mise à jour
- le nom des responsables (Responsable);
- un numéro de version (Version).

Description textuelle – Description des scenarios (1/2)

La deuxième partie contient la description du fonctionnement du cas sous la forme d'une séquence de messages échangés entre les acteurs et le système.

Cette deuxième partie se développe en trois points :

- Les pré-conditions. Elles indiquent dans quel état est le système avant que se déroule le scénario normale (le distributeur est alimenté en billets par exemple).
- L'enchaînement des messages.
- Les post-conditions. Elles indiquent dans quel état se trouve le système après le déroulement du scénario nominal (une transaction a été enregistrée par la banque par exemple).

Description textuelle – Description des scenarios (2/2)

- L'enchaînement des messages, contient toujours un **scénario nominal** qui correspond au fonctionnement «normal» du cas d'utilisation (par exemple, un retrait d'argent qui se termine par l'obtention des billets demandés par l'utilisateur).

- Les acteurs n'étant pas sous le contrôle du système, ils peuvent avoir des comportements imprévisibles. Le scénario nominal ne suffit donc pas pour décrire tous les comportements possibles. Au scénario nominal s'ajoutent fréquemment des **enchaînements alternatifs** (par exemple, le client choisit d'effectuer un retrait en euros ou en fcfa) et des **enchaînements d'erreur** (par exemple, la carte introduite n'est pas valide).

- Ces deux types d'enchaînements se décrivent de la même façon que le scénario nominal mais il ne faut pas les confondre. Un enchaînement alternatif diverge du scénario nominal (c'est un embranchement dans un scénario nominal) mais y revient toujours, alors qu'une séquence d'exception intervient quand une erreur se produit (l'enchaînement nominal s'interrompt, sans retour au scénario nominal).

Description textuelle – Rubriques optionnelles

La dernière partie de la description d'un cas d'utilisation est une rubrique optionnelle.

- Elle contient généralement des spécifications non fonctionnelles (performance, sécurité, disponibilité, confidentialité, etc.)(ce sont le plus souvent des spécifications techniques, par exemple pour préciser que l'accès aux informations bancaires doit être sécurisé).

- Cette rubrique contient aussi d'éventuelles contraintes liées aux interfaces homme-machine (par exemple, pour donner la possibilité d'accéder à tous les comptes d'un utilisateur à partir de l'écran principal)

Soyez agiles et itératifs !

- Les descriptions textuelles des cas d'utilisation et les diagrammes UML ne sont jamais parfaits. Il leur manque certains éléments et certaines assertions sont fausses. La solution ne consiste pas à adopter l'attitude du processus en cascade et à déployer toujours plus d'efforts pour obtenir d'emblée des spécifications exactes et exhaustives.

- Il s'agit simplement de faire de notre mieux dans les délais impartis et en appliquant les meilleures pratiques. Il existe un moyen terme entre la démarche en cascade et la programmation sauvage : le développement itératif et incrémental. Dans cette approche, les cas d'utilisation et les autres modèles sont progressivement affinés, vérifiés et clarifiés grâce à la programmation et aux tests.

Plan

- UML
- Diagramme de cas d'utilisation
- Diagramme de classe
- Diagramme de séquence

Diagramme de classes

«Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet. Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il contient principalement des classes. Une classe contient des attributs et des opérations. Le diagramme de classes n'indique pas comment utiliser les opérations : c'est une description purement statique d'un système. L'aspect dynamique de la modélisation est apporté par les diagrammes d'interactions»

Rappel…

Voiture
marque modèle couleur vitesse
démarrer accélérer freiner éteindre

Avion
marque modèle Altitude vitesse
décoller atterrir virer

Nom de la classe

Description des attributs

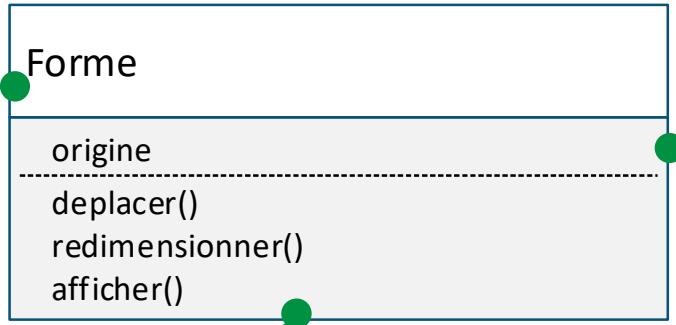
Description des méthodes

Rappel: Une classe est une description d'un ensemble d'objets ayant une sémantique, des attributs, des méthodes et des relations en commun. Un objet est une instance d'une classe.

Classe

1. Nom

Le nom de la classe doit être significatif et complet. Il commence par une majuscule. S'il est composé de plusieurs mots, la première lettre de chaque mot doit être une majuscule. Forme



2. Attributs

Liste des attributs de la classe avec les modificateurs d'accès éventuels. Certains attributs n'apparaissent pas directement, ils seront déduits à partir des relations entre classes.

3. Méthodes

Liste des méthodes de la classe. Quand la méthode accepte des paramètres alors ces paramètres et leurs types sont ajoutés entre les parenthèses.

Modificateurs d'accès

Les modificateurs d'accès ou de visibilité associés aux classes ou à leurs propriétés sont notés comme suit :

- Le mot-clé public ou le caractère +. Propriété ou classe visible partout.
- Aucun caractère, ni mot-clé. Propriété ou classe visible uniquement dans le paquetage où la classe est définie.
- Le mot-clé protected ou le caractère #. Propriété ou classe visible dans la classe et par tous ses descendants.
- Le mot-clé private ou le caractère -. Propriété ou classe visible uniquement dans la classe.

Relations entre classes – Association (1/3)

- Une association est une relation statique bidirectionnelle entre classes
  - C'est une représentation abstraite d'une relation qui existe dans le domaine modélisé

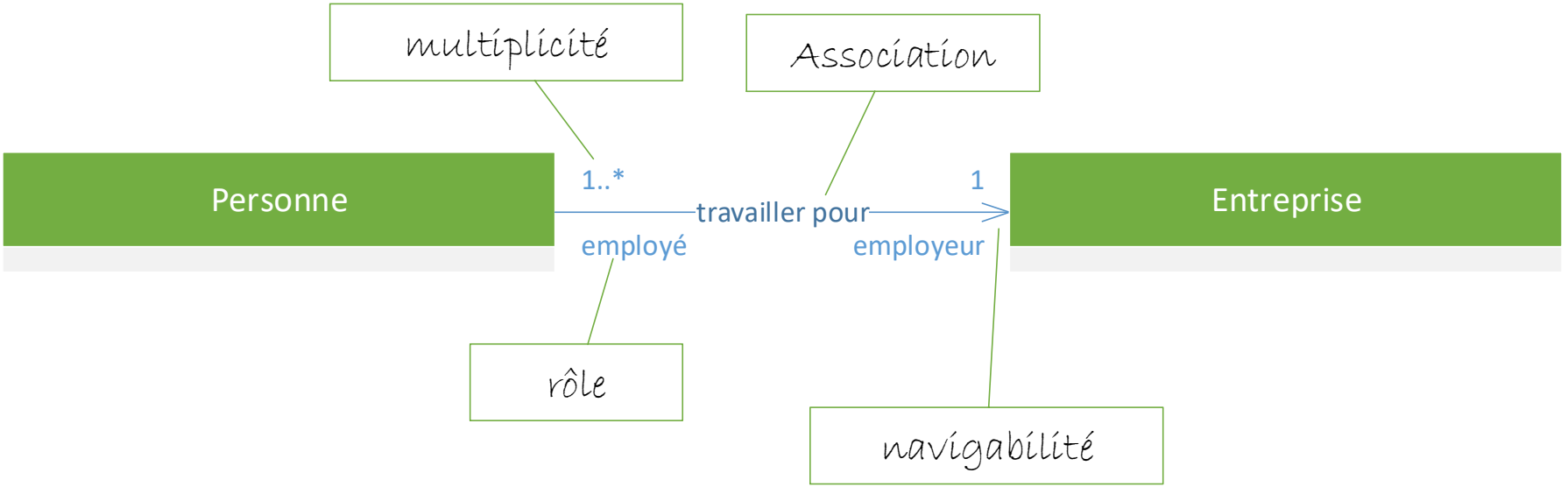
- Une association représente une relation sémantique entre les objets d'une classe.

- Pendant l'exécution d'un système les classes et les associations ne servent qu'à créer des instances:
  - Les objets et les liens

Relations entre classes – Association (2/3)

- Est représentée graphiquement par un trait plein entre les classes associées.
- Est complétée par un nom qui correspond souvent à un verbe à l'infinitif ou un groupe verbal. On spécifie le sens de la navigation en ajoutant une flèche.
- Chaque extrémité de l'association indique le rôle de la classe dans la relation et précise le nombre d'objets de la classe qui interviennent dans l'association.

Relations entre classes – Association (3/3)

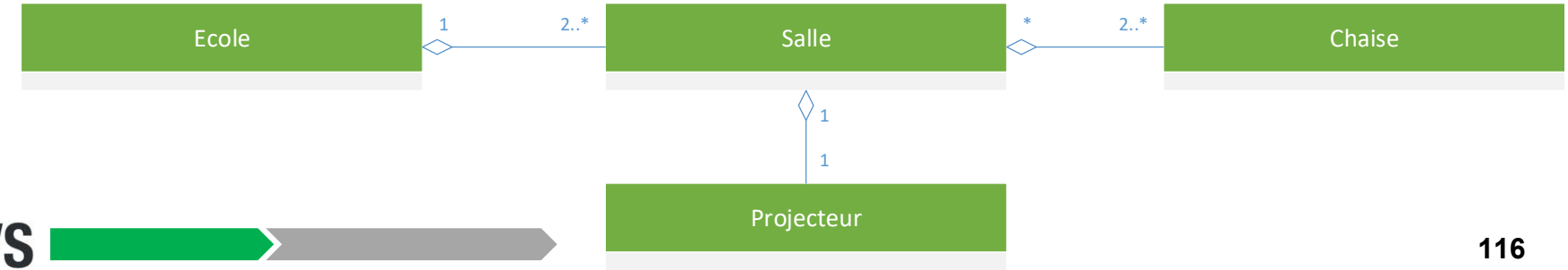


Relations entre classes – Multiplicité

- 0..1 = optionnel = (Zéro ou un)
- 1 = (exactement 1)
- 0..* = * = quelconque = (Zéro ou plusieurs)
- 1..* = (au moins 1) = (Un ou plusieurs)
- Min..Max = (entre Min et Max)
- n..* = (au minimum n) = (n ou plusieurs)
- n1..n2, n5, n7..n10 = (Intervalles convexes)

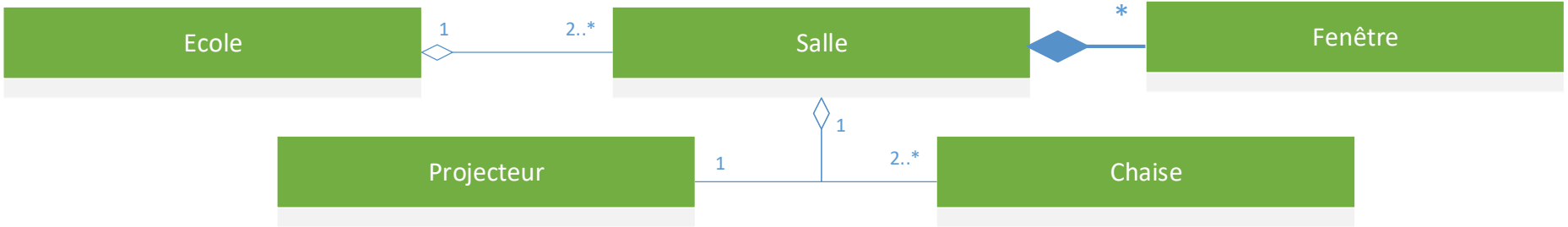
Relations entre classes – Agrégation

- Cas particulier d'association non symétrique exprimant une relation de contenance, mais sans contrainte supplémentaire.
- Elle représente la relation d'inclusion structurelle ou comportementale d'un élément dans un ensemble. Contrairement à l'association, l'agrégation est une relation transitive.
- Une agrégation se distingue d'une association par l'ajout d'un losange vide du côté de l'agrégat.



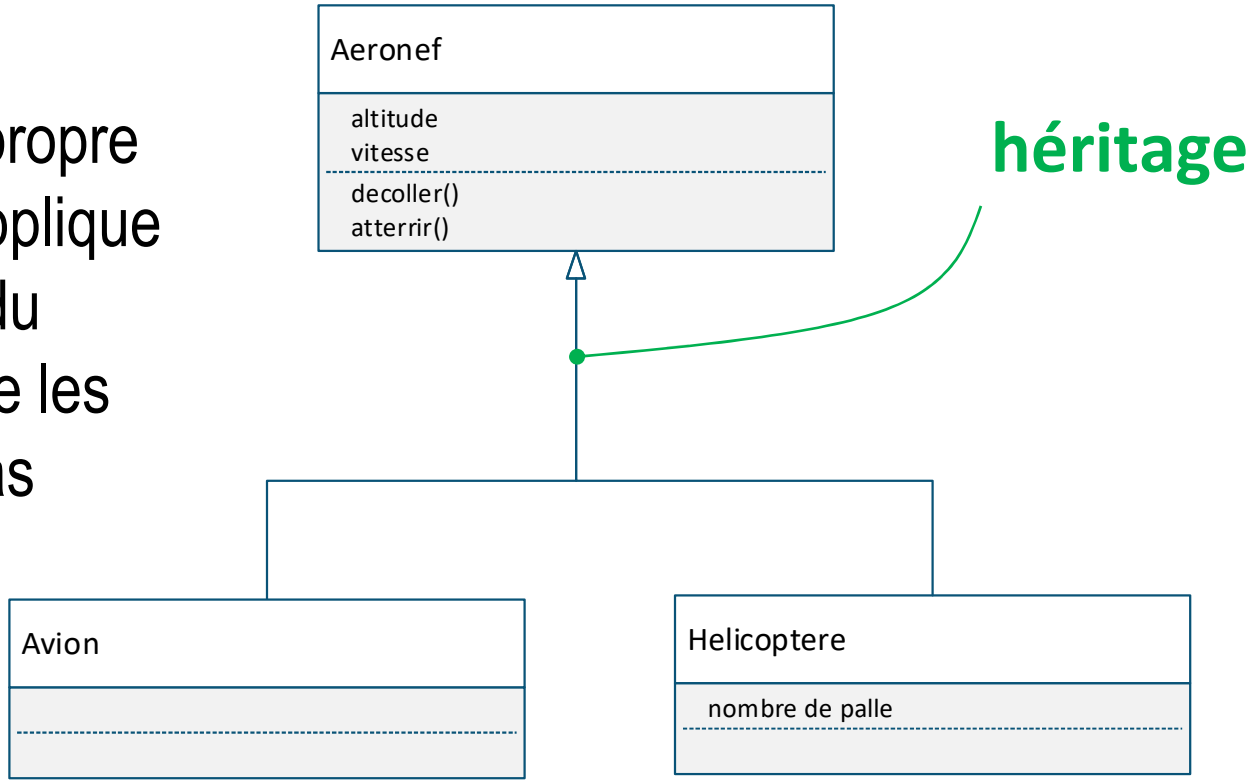
Relation entre classes – Composition

- Forme forte d'agrégation, dans laquelle les parties ne peuvent appartenir à plusieurs agrégats à la fois. Et où le cycle de vie des parties est subordonné à celui de l'agrégat. A savoir, la destruction ou la copie de l'objet composite implique respectivement la destruction ou la copie de ses composants.



Relations entre classes - Héritage

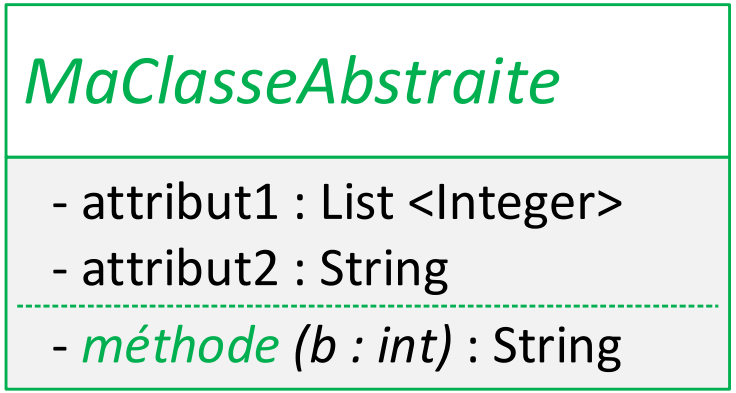
En UML, la relation d'héritage n'est pas propre aux classes. Elle s'applique à d'autres éléments du langage UML, comme les paquetages ou les cas d'utilisation.



Classe abstraite

Une classe abstraite (nom de la classe en italique) ne peut pas s'instancier directement, mais uniquement par le biais de ses sous-classes

- Elle sert à factoriser des propriétés communes à plusieurs sous-classes.
- On peut redéfinir une opération.
- Une méthode est dite abstraite lorsqu'on connaît son entête mais pas la manière dont elle peut être réalisée. Une classe est dite abstraite lorsqu'elle définit au moins une méthode abstraite ou lorsqu'une classe parent contient une méthode abstraite non encore réalisée.
- La méthode «***methode»*** est abstraite donc la classe est abstraite.

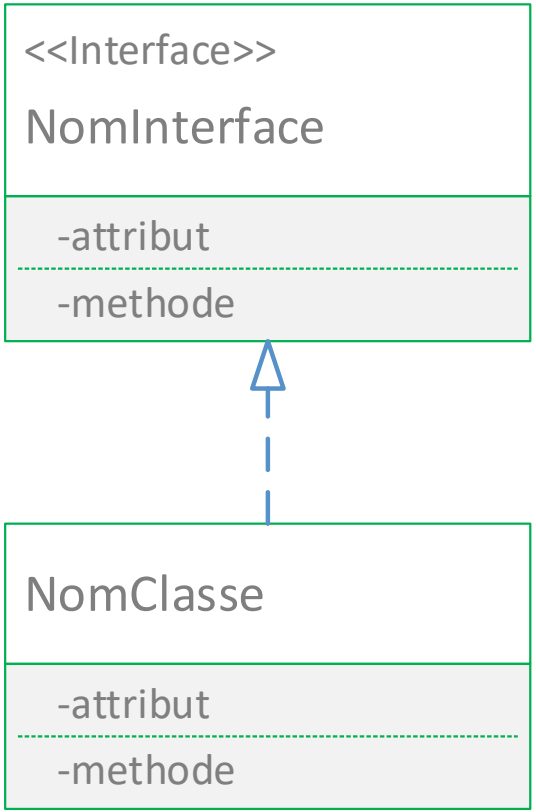


Interface (1/2)

- Une classe est purement abstraite si toutes ses méthodes sont abstraites.
- Une interface est une « classe » purement abstraite déclarée dont toutes les méthodes sont publiques.
- Une interface est une liste de noms de méthodes publiques. Dans l'interface on définit la signature des méthodes qui doivent être implémentées dans les classes qui les implémentent.
- Une interface est un modèle pour une classe

Interface (2/2)

- Une classe peut implémenter une ou plusieurs interfaces.
- Lorsqu'une classe implémente une interface, elle doit implémenter toutes les méthodes définies dans cette interface.
- Une interface peut hériter d'autres interfaces
- La réalisation d'une interface par une classe est représentée graphiquement par un trait discontinu qui se termine par une pointe triangulaire.



En résumé

- Malgré son importance, le diagramme de classes a plusieurs limites à la bonne compréhension d'un système. Notamment, ils ne montrent pas le cycle de vie des objets : nous ne savons pas, à la lecture de tels diagrammes, dans quel ordre les objets sont créés, utilisés puis détruits, pas plus qu'ils n'indiquent l'ordre dans lequel s'enchaînent les opérations. Néanmoins, le diagramme des classes est le plus utilisé pour la modélisation d'un système car il en montre la structure interne.

- Avec le diagramme des cas d'utilisation, vu à la section précédente, nous disposons, à présent, de deux façons de voir un système. En effet, le diagramme des classes donne une approche objet tandis que celui des cas d'utilisation montre un côté fonctionnel.

Plan

- UML
- Diagramme de cas d'utilisation
- Diagramme de classe
- Diagramme de séquence

Diagramme de séquence

Les diagrammes de cas d'utilisation montrent des interactions entre des acteurs et les grandes fonctions d'un système. Cependant, les interactions ne se limitent pas aux acteurs : par exemple, des objets au cœur d'un système, instances de classes, interagissent lorsqu'ils s'échangent des messages.

Le diagramme de classes montre la structure interne d'un système.

Pour ajouter un aspect dynamique à la modélisation, il faut « descendre » au niveau des instances et montrer comment elles interagissent.

Des instances de ces classes peuvent figurer sur un diagramme d'interaction particulier, appelé « diagramme de séquence».

Ligne de vie

- Une interaction décrit le comportement d'un système en se focalisant sur l'échange d'informations entre les éléments du système. Les participants à une interaction sont appelés « lignes de vie ».
- Une ligne de vie représente un participant unique à une interaction.
- Le terme « ligne de vie » est utilisé car, souvent, les interactions montrent des objets (instances de classes). Au cours d'une interaction, des objets peuvent être créés, utilisés, et parfois détruits. Ce cycle de vie des objets est matérialisé par une ligne verticale (la ligne de vie) dans le diagramme de séquence.
- Le diagramme de séquence représentent des interactions entre des lignes de vie. Un diagramme de séquence montre des interactions sous un angle temporel, et plus particulièrement le séquencement temporel de messages échangés entre des lignes de vie.

Message

- Elément de communication unidirectionnel entre lignes de vie qui déclenche une activité dans l'objet destinataire. La réception d'un message provoque un événement dans l'objet récepteur. La flèche pointillée représente un retour au sens UML. Cela signifie que le message en question est le résultat direct du message précédent.

- Un message synchrone se représente par une flèche à l'extrémité pleine qui pointe sur le destinataire du message. Ce message peut être suivi d'une réponse qui se représente par une flèche en pointillé. Un message asynchrone se représente par une flèche à l'extrémité ouverte. La création d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie. La destruction d'un objet est matérialisée par une croix qui marque la fin de la ligne de vie de l'objet.

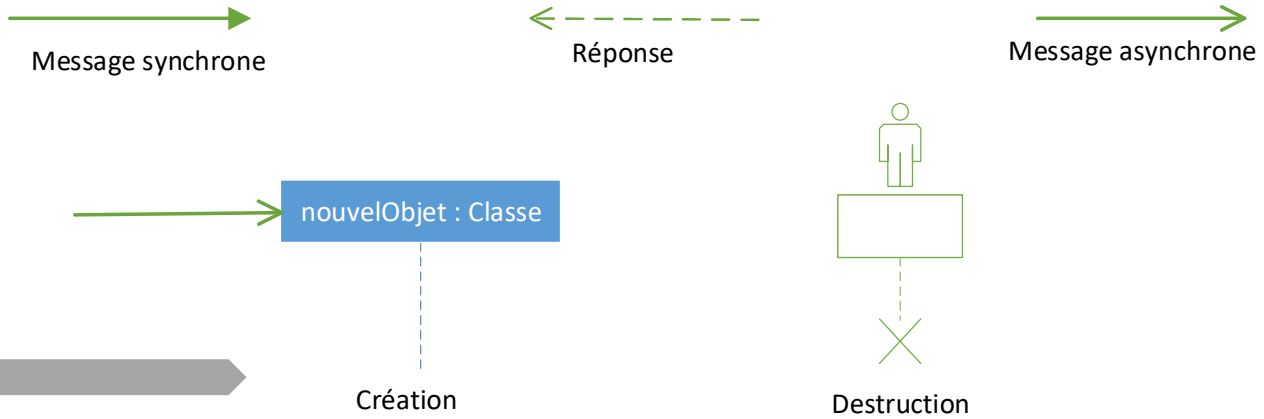
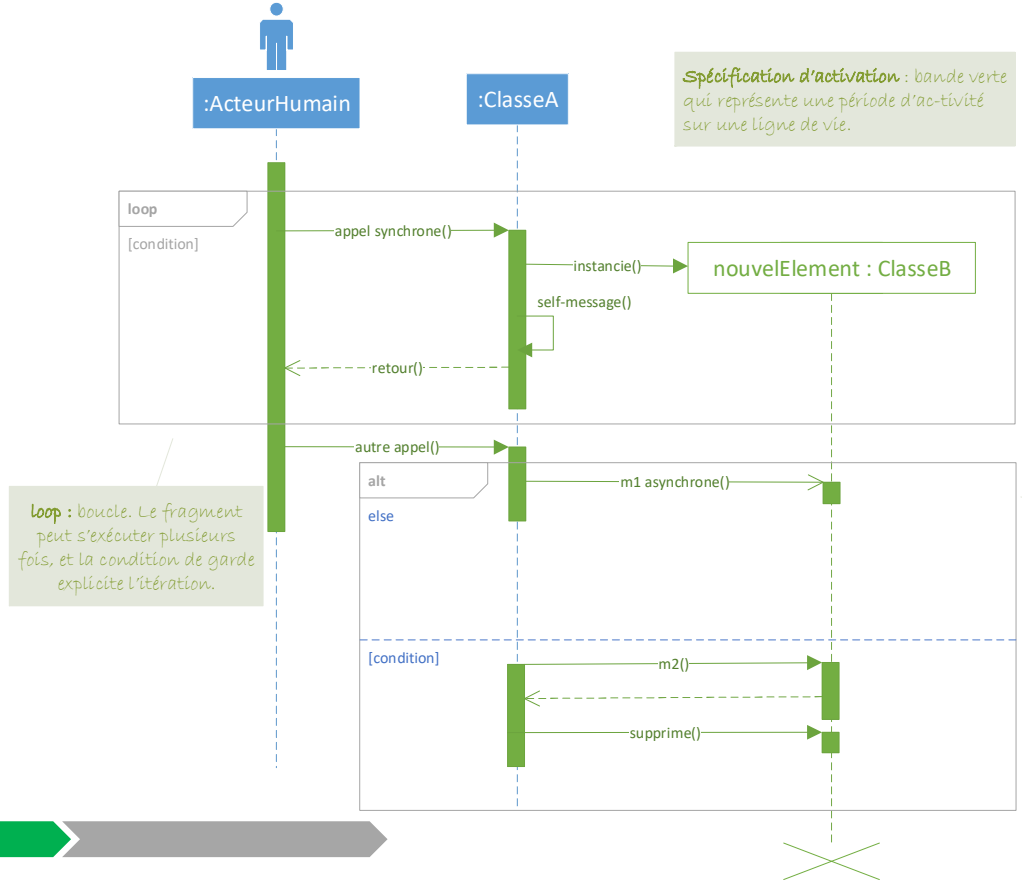


Diagramme de séquence (1/2)

Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie. Un message définit une communication particulière entre des lignes de vie. Plusieurs types de messages existent. Les plus communs sont :

- l'envoi d'un signal;
- l'invocation d'une opération;
- la création ou la destruction d'une instance.

Diagramme de séquence (2/2)



Spécification d'activation : bande verte qui représente une période d'ac-tivité sur une ligne de vie.

Occurrence d'interaction : une interaction peut faire référenceexplicitement à une autre interac-tion grâce à un cadre avec le mot-clé **ref** et indiquant le nom de l'autreinteraction

loop : boucle. Le fragment peut s'exécuter plusieurs fois, et la condition de garde explicite l'itération.

alt : fragments alternatifs. Seul le fragment possédant la condition vraie s'exécutera.

opt : optionnel. Le fragment ne s'exécute que si la condition fournie est vraie.

par : fragments s'exécutant en parallèle.

En résumé…

- Les diagrammes de séquence montrent l'aspect dynamique d'un système. Ce ne sont pas les seuls : les diagrammes de communication, les diagrammes d'activité et les diagrammes d'états-transitions décrivent eux aussi un système sous cet angle. Cependant, les diagrammes de séquence sont mieux appropriés à la description des interactions entre les éléments d'un système.

- les diagrammes de séquence montrent le séquencement temporel de messages, mais nous savons à présent comment construire les trois modèles essentiels d'un système:
  - le modèle fonctionnel et externe à l'aide du diagramme des cas d'utilisation ;
  - le modèle structurel et interne grâce au diagramme des classes ;
  - le modèle des interactions au sein du système avec les diagrammes de séquence.

UVS
UNIVERSITE VIRTUELLE DU SENEGAL

Analyse et Conception Orientée Objet

Analyse Orientée Objet

A est un bon modèle de B si A permet de répondre de façon satisfaisante à des questions prédéfinies sur B.

Douglas. T. Ross