

UML



Facile



Normal



Difficile



Professionnel



Expert

https://wiki.waze.com/wiki/Your_Rank_and_Points

- Le langage UML
- Quelques diagrammes

1 – Le langage UML



- Définition
- Points forts & Points faibles
- Utilisation UML
- Caractéristiques UML
- Les diagrammes UML 2.5
- Les vues UML
- La vue logique
- La vue de réalisation
- La vue des processus
- La vue de déploiement
- La vue des cas d'utilisation



UML : Unified Modeling Language



- **UML permet d'exprimer et d'élaborer des modèles objet, indépendamment de tout langage de programmation.** Il a été pensé pour servir de support à une analyse basée sur les concepts objet.
- UML est un **langage formel**, défini par un **métamodèle**.
- Le métamodèle d'UML décrit de manière très précise tous les éléments de modélisation et la sémantique de ces éléments (leur définition et le sens de leur utilisation).
UML normalise les concepts objet.

UML est avant tout **un support de communication performant**, qui facilite la représentation et la compréhension de solutions objet

1.2 - Historique : Méthodes Objets



En 1994, plus de 50 méthodes OO

- Fusion, Shlaer-Mellor, ROOM, Classe-Relation, Wirfs-Brock, Coad-Yourdon, MOSES, Syntropy, BOOM, OOSD, OSA, BON, Catalysis, COMMA, HOOD, Ooram, DOORS...

Les méta modèles se ressemblent de plus en plus

Les notations graphiques sont toutes différentes

L'industrie a besoin de standards

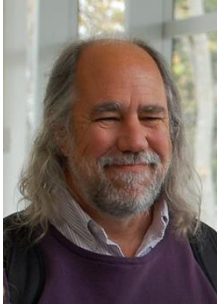


La pratique des méthodes a permis de faire le tri entre les différents concepts

Jim Rumbaugh, Grady Booch (1993) et plus tard ***Ivar Jacobson*** (1994) décident d'unifier leurs travaux:

- Methode OMT(Object Modeling Technique)
- Methode Booch
- Methode OOSE (Object Oriented Software Engineering)

1.2 - Historique : Les créateurs de la notation UML



Grady Booch

Méthode de Grady Booch

La méthode proposée par G. Booch est une méthode de conception, définie à l'origine pour une programmation Ada, puis généralisée à d'autres langages. Sans préciser un ordre strict dans l'enchaînement des opérations



James Rumbaugh

Méthode OMT

La méthode OMT (Object Modeling Technique) permet de couvrir l'ensemble des processus d'analyse et de conception en utilisant le même formalisme. L'analyse repose sur les trois points de vue: statique, dynamique, fonctionnel, donnant lieu à trois sous-modèles.

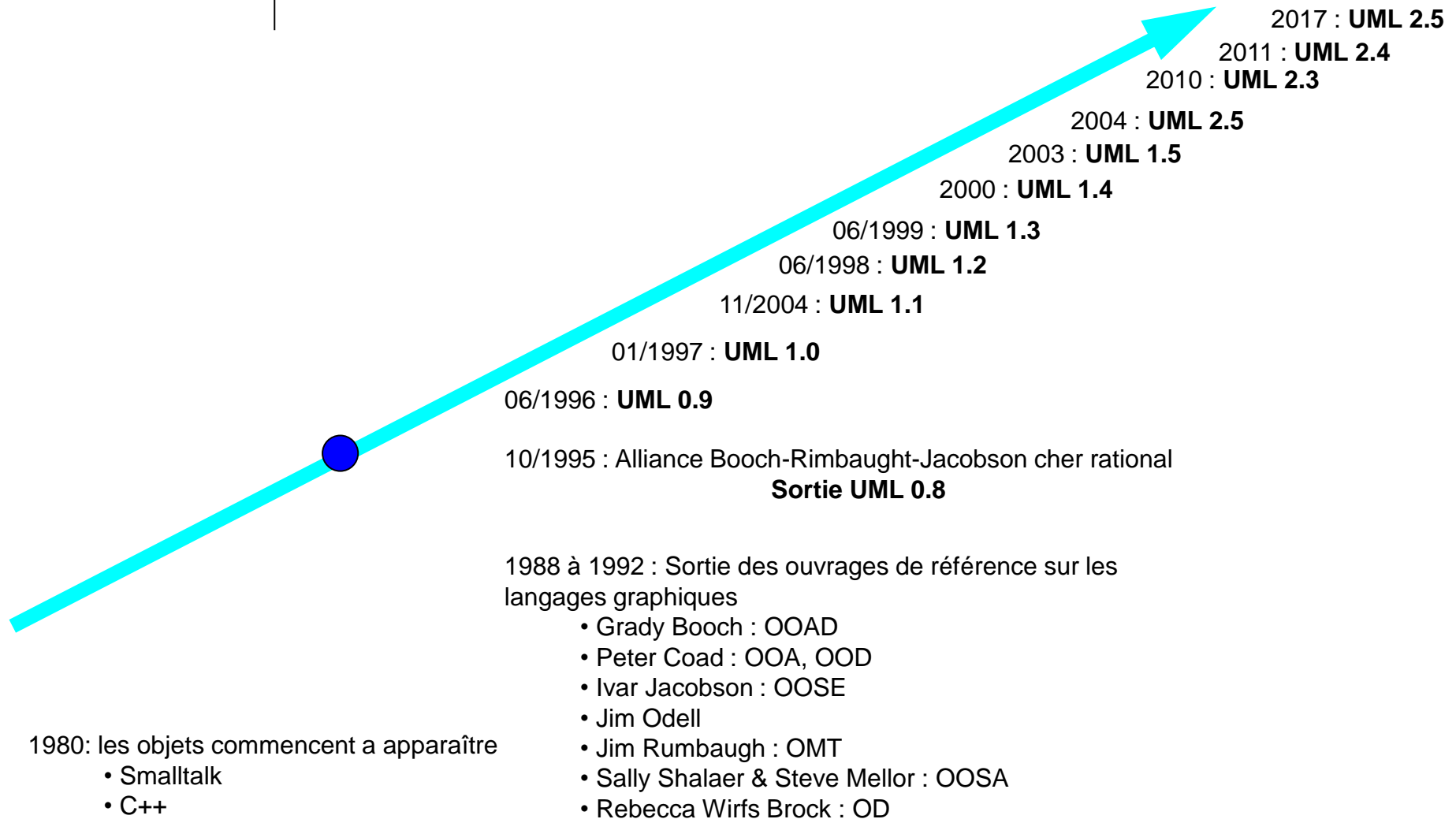


Ivar Jacobson

Méthode OOSE

Object Oriented Software Engineering (OOSE) est un langage de modélisation objet créé par Ivar Jacobson. OOSE est une méthode pour l'analyse initiale des usages de logiciels, basée sur les « cas d'utilisation » et le cycle de vie des logiciels.

1.2 - Historique

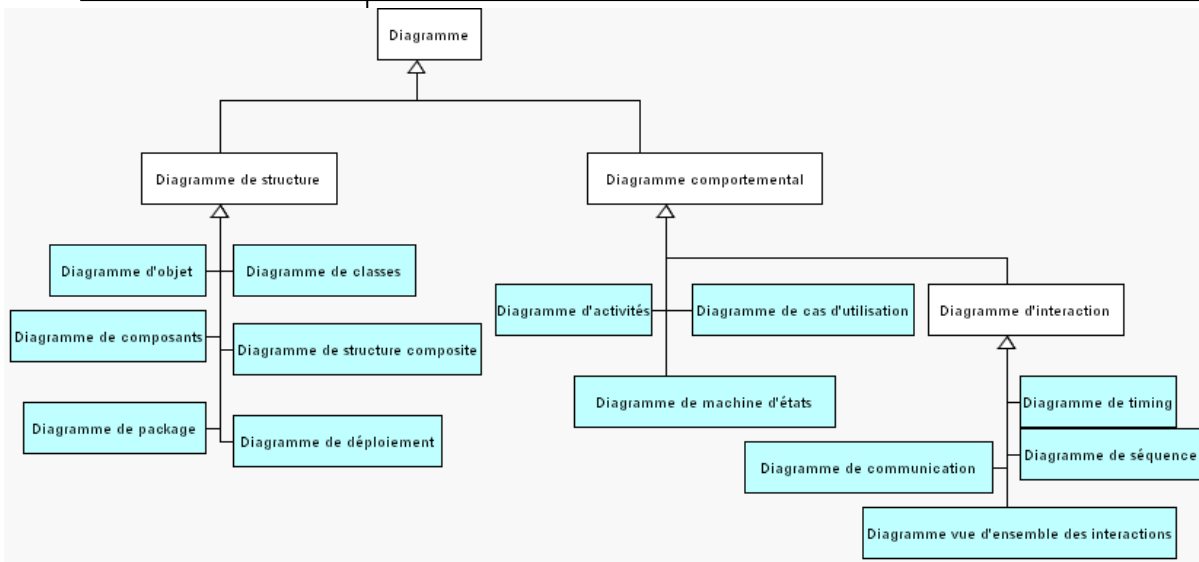




- Les points forts d'UML
 - UML est un langage formel et normalisé
 - UML est un support de communication performant
- Les points faibles d'UML
 - La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.
 - Le processus n'est pas couvert par UML

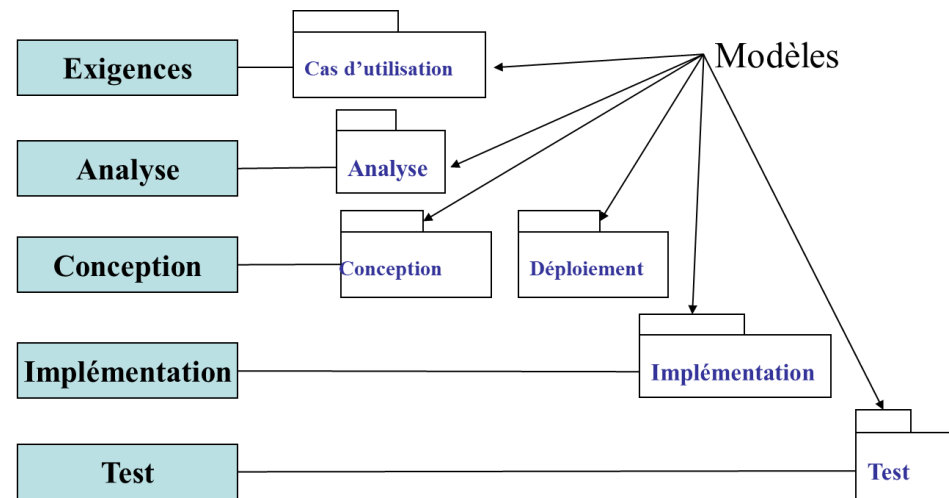


1.4 - Utilisation UML



Modélisation UML

- 14 diagrammes (diagramme de profile ajouter dans UML2.2)
- 1 Langage de contraintes objet (Object Constraint Langage : OCL)



<https://manurenaux.wp.imt.fr/2013/09/27/interet-de-luml-dans-un-projet-informatique/>

1.5 - Caractéristiques UML

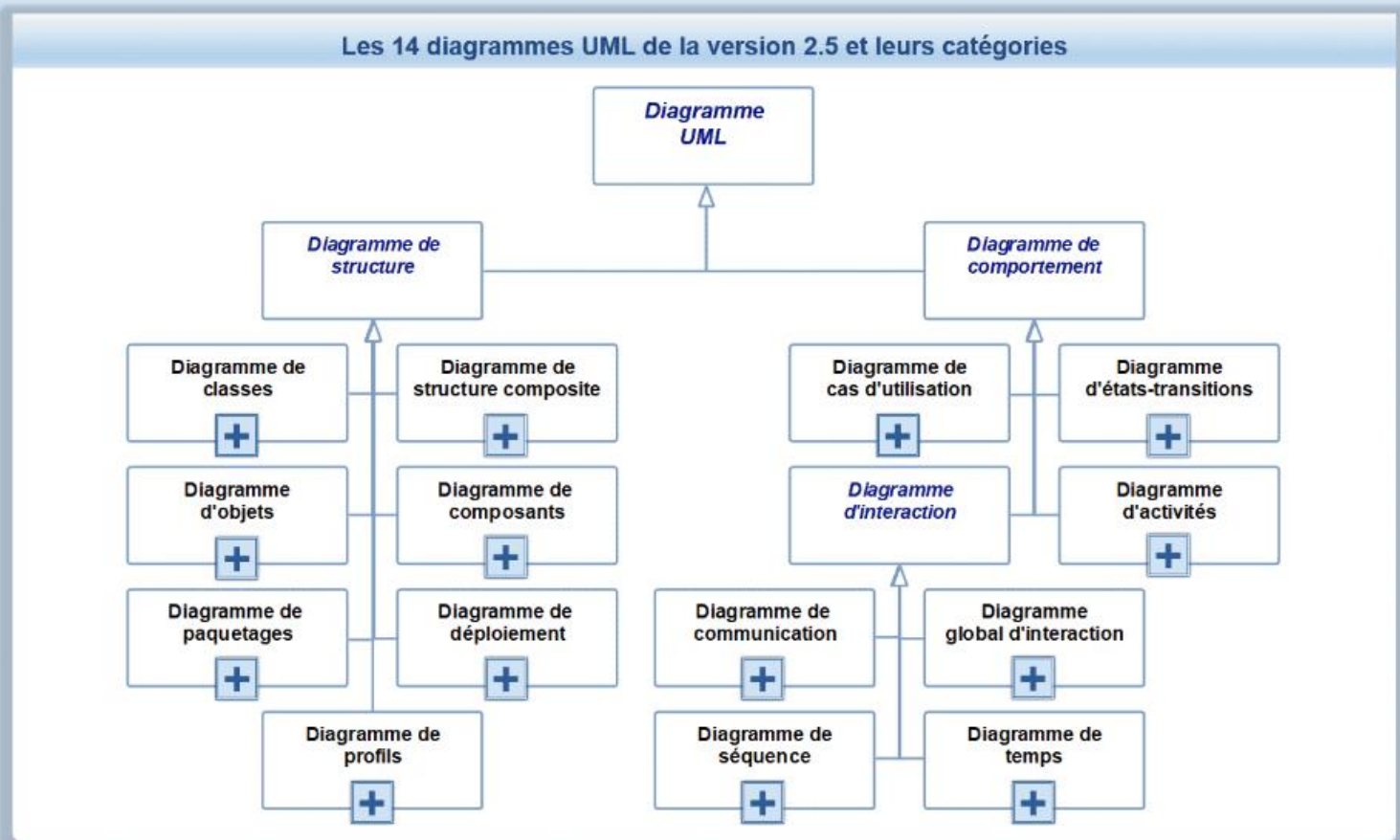


Notation	Graphique, Textuelle, Semi-formelle
auto-décrite	méta-modélisation : modélisation récursive des éléments de modélisation eux-mêmes
le paquetage	<p>Organisation du modèle, Espace de noms</p> <p>vue logique \Rightarrow catégorie</p> <p>vue de réalisation \Rightarrow sous-système</p> <p>liens de dépendance entre paquetages</p> <p>\Rightarrow importation = utilisation</p> <p>\Rightarrow inclusion</p>
les diagrammes	<p>14 diagrammes + 1 langage</p> <p>7 structurels et 7 comportementaux</p>
la démarche UML	<p>UML ne propose pas de démarche, et RECOMMANDE :</p> <p>un processus piloté par les cas d'utilisation, centré sur l'architecture, selon une démarche itérative, et incrémentale, privilégiant la réduction du risque comme objectif de gestion de projet.</p>

1.6 - Les diagrammes UML 2.5



UML. Survol des 14 diagrammes de la version 2.5

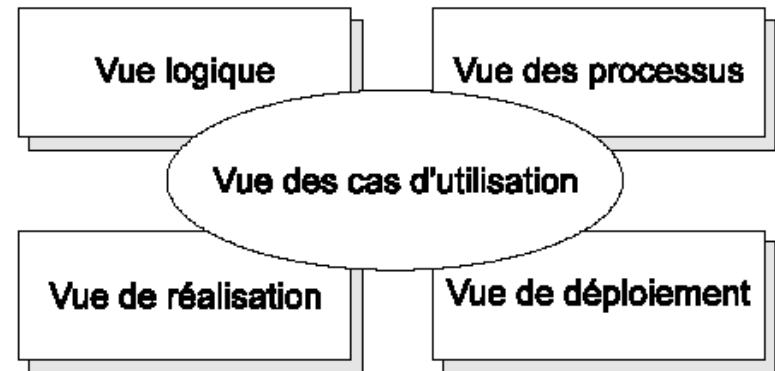


<https://cours.khalilmamouny.com/uml-survol-des-14-diagrammes-version-2-5-1/>



4 vues + 1 :

- Vue logique,
- Vue de réalisation,
- Vue des processus,
- Vue de déploiement,
- *Vue des cas d'utilisation.*





Objectif : analyser le problème (fonctionnel), et le résoudre formellement

La **vue logique** :

- Modélise les éléments et mécanismes principaux du système en se concentrant sur les abstractions et l'encapsulation,
- Identifie les éléments du domaine (métier, savoir-faire), ainsi que les relations et interactions entre ces éléments,
- Organise les éléments du domaine en catégories.

Les éléments :

- Les objets
- Les classes
- Les collaborations
- Les interactions
- Les paquetages <<Catégorie>>



Objectif : décrire la solution logicielle à mettre en œuvre

La vue de réalisation

- Montre l'allocation des éléments de modélisation dans des modules (fichiers sources, bibliothèques dynamiques, BDD, exécutables, etc...),
- Identifie les modules qui réalisent (physiquement) les classes de la vue logique,
- Organise les composants (distribution du code en gestion de configuration, les dépendances entre les composants...) et les contraintes de développement (bibliothèques externes...),
- Montre l'organisation des modules en sous-systèmes et leur interfaces.

Les éléments :

- Les modules
- Les sous-programmes
- Les tâches
- Les paquetages <<sous-système>>



Objectif : décrire le fonctionnement en dynamique de la solution

La **vue des processus** montre :

- La décomposition du système en terme de processus
- Les interactions entre les processus (leur communication).
- La synchronisation et la communication des activités parallèles.

Les éléments :

- Les tâches
- Les threads
- Les processus
- Les interactions



Objectif : décrire la projection du logiciel sur le matériel

La **vue de déploiement** décrit les ressources matérielles et la répartition du logiciel dans ces ressources :

- La disposition et nature physique des matériels, ainsi que leurs performances
- L'implantation des modules principaux sur les noeuds du réseau
- Les exigences en terme de performances

Les éléments:

- Les noeuds
- Les modules
- Les programmes principaux



Objectif : décrire le besoin (fonctionnel).

La vue des cas d'utilisation :

- Unifie les quatre autres vues de l'architecture,
- Définit les besoins des clients du système et centre la définition de l'architecture du système sur la satisfaction et la réalisation de ces besoins,
- Conduit à la définition d'un modèle d'architecture à l'aide de scénarios et de cas d'utilisation,
- Motive les choix et permet d'identifier les interfaces critiques et se concentrer sur les problèmes importants.

Les éléments:

- Les acteurs
- Les cas d'utilisation
- Les classes
- Les collaborations

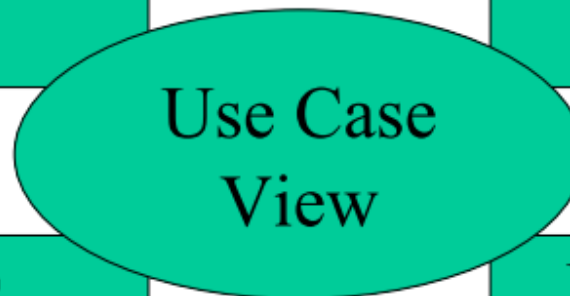
1.13 – Résumer des vues



- Classes
 - Interfaces
 - Collaboration
- => Les services du systèmes



- Composant
 - Fichiers Source
- => Configuration du système



=> Comportement du système

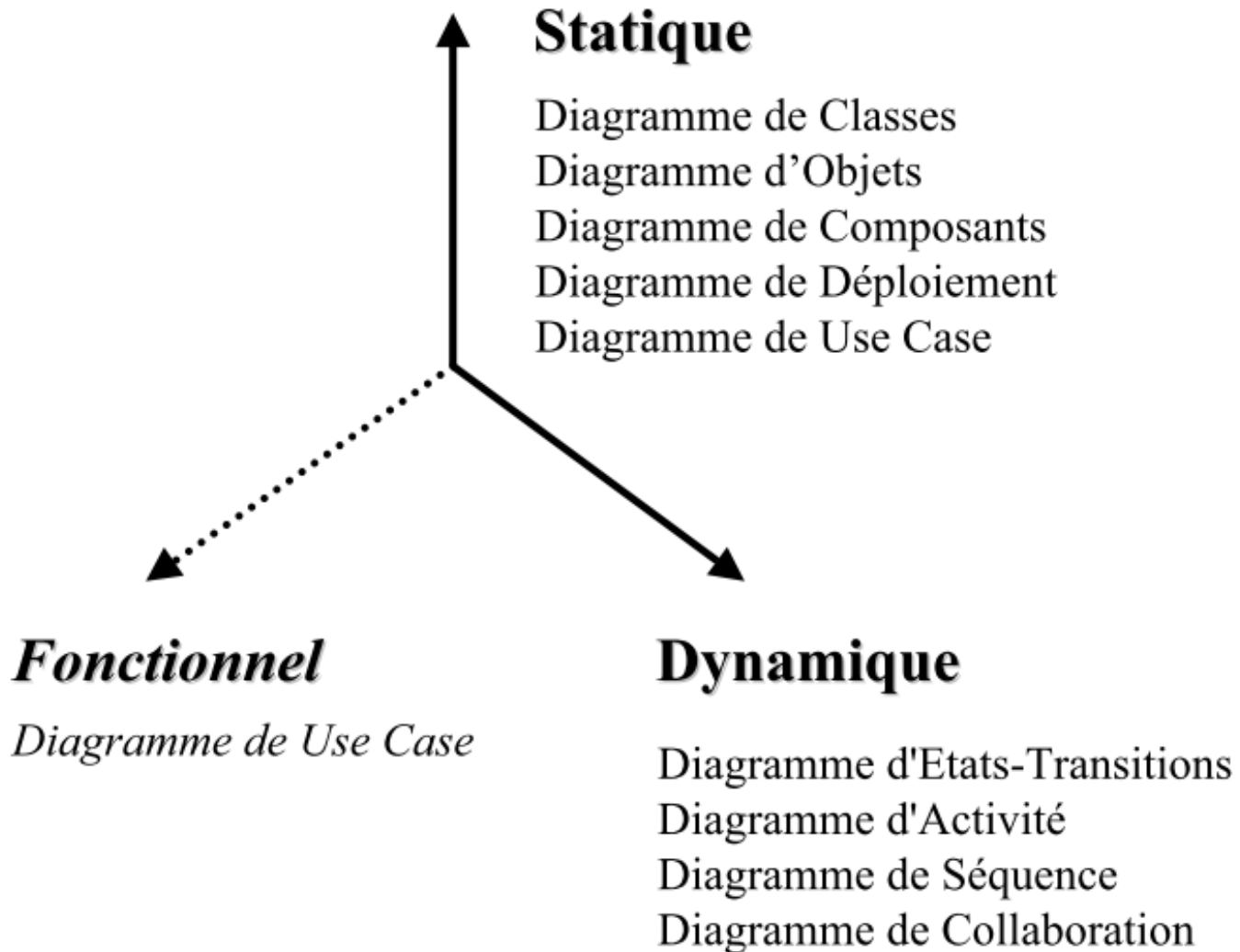


- Thread
 - Process
 - Concurrence
 - Synchronisation
- => Performance du système



- Architecture
 - Hardware
 - Distribution
- => Topologie du système

1.14 - Trois Axes de Modélisation





- 1 - UML pour les décideurs
- 2 - UML pour les concepteurs
- 3 - UML pour les développeurs



- Capture initiale des besoins
 - Cahier des charges préliminaires
- Capture des besoins fonctionnels
 - Cahier des charges
- Analyse
 - Expression des besoins
 - Architecture logique



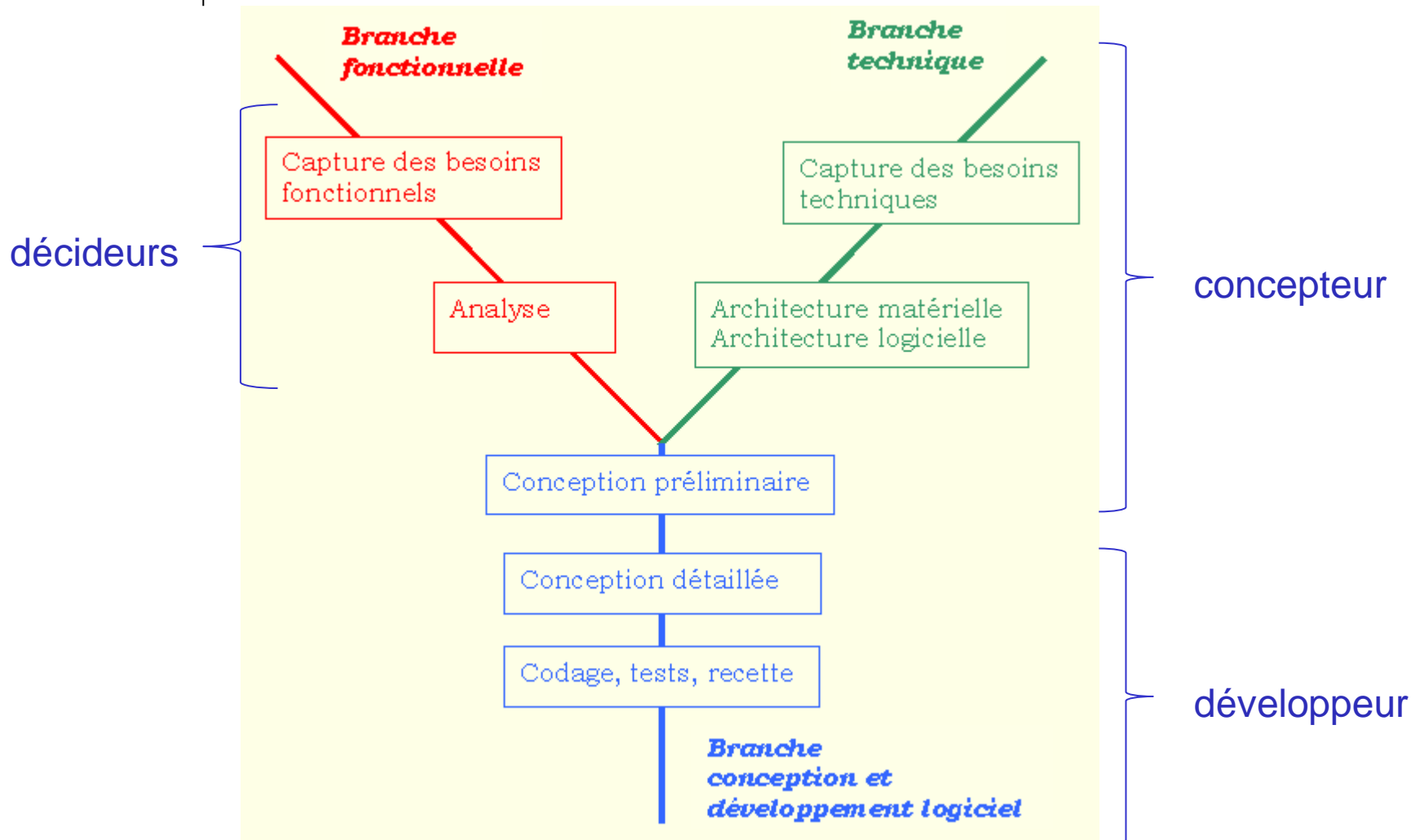
- Capture des besoins techniques
- Conception générique
 - Prototype
 - Générateur de code
 - Design Patterns
- Conception préliminaire
 - Interface
 - Design Patterns
- Architecture Technique

1.15.3 - UML pour les développeurs

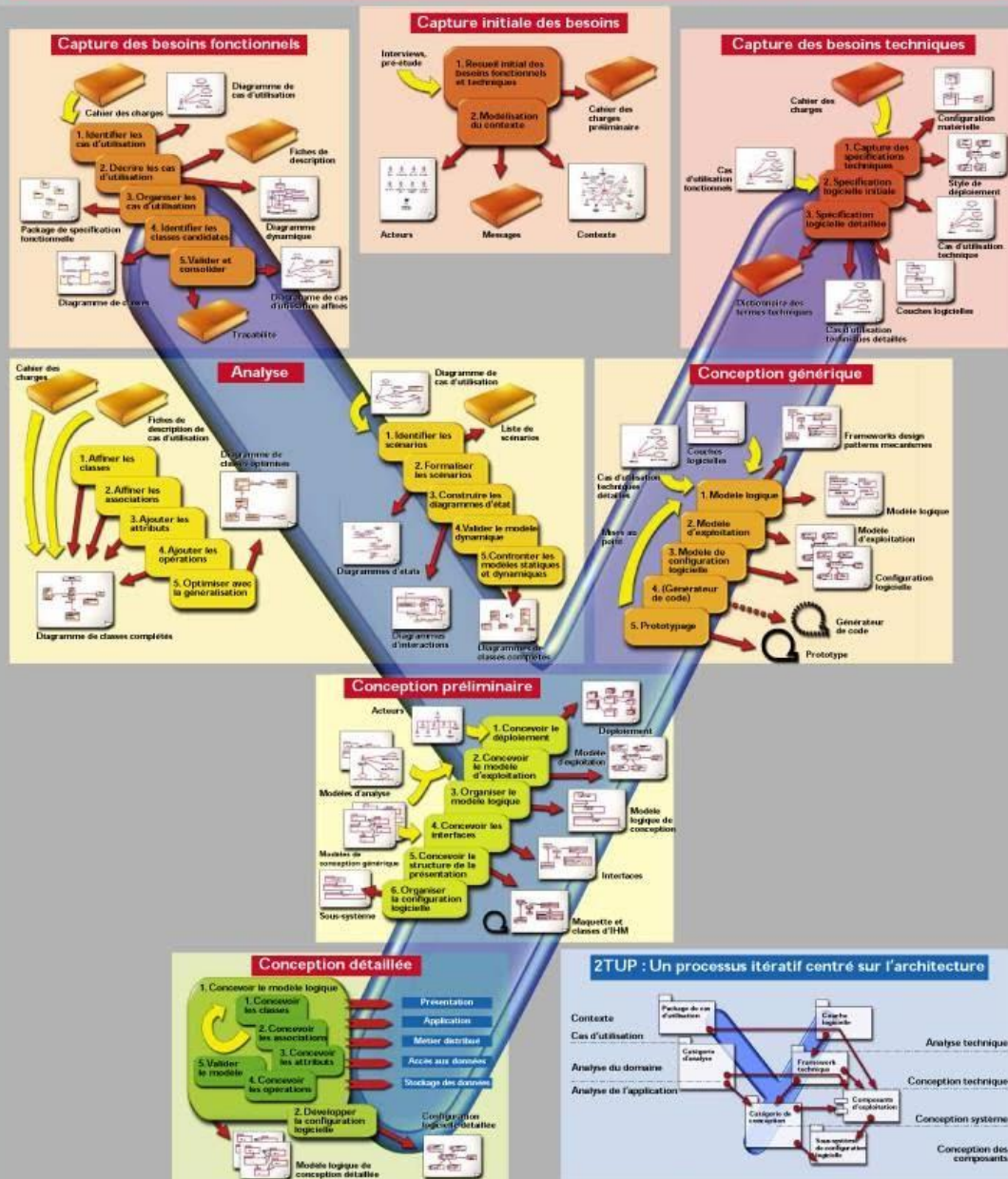


- Conception détaillée
- Générateur de code
- Retro ingénierie

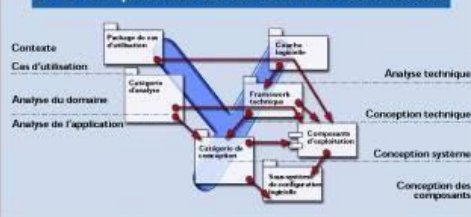
1.16 - Two Track Unified Process



Two Track Unified Process



2TUP : Un processus itératif centré sur l'architecture







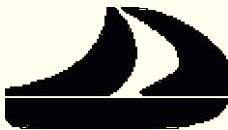









SysML est un langage de modélisation spécifique au domaine de l'ingénierie système.

Il permet la spécification, l'analyse, la conception, la vérification et la validation de nombreux systèmes et systèmes-de-systèmes.

SysML se définit comme une extension d'un sous-ensemble d'UML (Unified Modeling Language) via l'utilisation du mécanisme de profil défini par UML.

1.7 – Outils UML



 <p>Windows, Linux, Solaris, Mac OS X</p>	 <p>Enterprise Architect Version 7.0</p> <p>Entreprise Architect Sparx Systems</p>	 <p>Argo UML</p>	 <p>IBM : Rational Rose Le Leader Mondial</p>
 <p>I-Logix Rhapsody Modeler</p>	 <p>Objecteering Software Objecteering/UML</p>	 <p>gentleware just model</p> <p>Poseidon for UML</p>	 <p>Python UML Tool</p>
 <p>ModelMaker Un extraordinaire outil UML pour Delphi Windows</p>	 <p>Papyrus</p>	 <p>Visual Paradigm Online Diagrams</p> <p>Visual Paradigm for UML Windows, OSX, Linux</p>	 <p>BOUML BOUML</p>

<https://uml.developpez.com/telecharger/>

http://www.objectsbydesign.com/tools/umltools_byPrice.html

https://fr.wikipedia.org/wiki/Comparaison_des_logiciels_d%27UML


1.7 – Pourquoi UML ?



www.uml-sysml.org



Recherche

 Rechercher

☐ Seulement dans le dossier courant

Accueil

Modéliser

UML

SysML

Diagrammes

Documentation







Infos pratiques

[Se connecter](#) [Plan de site](#) [Accessibilité](#) [Contact](#) [Images du Site](#) [Extranet](#)

Vous êtes ici : [Accueil](#) → [Modéliser](#) → [Pourquoi UML ?](#)



Navigation

-  [Modéliser](#)
-  [Pourquoi UML ?](#)
-  [Processus de modélisation](#)
-  [L'approche Top Down](#)
-  [Rédaction du rapport](#)
-  [Evolution d'un projet](#)

Pourquoi UML ?

De la même façon qu'il vaut mieux dessiner une maison avant de la construire, il vaut mieux modéliser un système avant de le réaliser.

UML pour :

- Obtenir une modélisation de très haut niveau indépendante des langages et des environnements.
- Faire collaborer des participants de tous horizons autour d'un même document de synthèse.
- Faire des simulations avant de construire un système.
- Exprimer dans un seul modèle tous les aspects statiques, dynamiques, juridiques, spécifications, etc...
- Documenter un projet.
- Générer automatiquement la partie logiciel d'un système.

Références

- Le dernier livre de Pascal Roques sur SysML : [cliquez ici](#)
- Les précédents livres de Pascal Roques sur UML 2.0 : [cliquez ici](#)
- L'excellent site de Laurent Audibert : [cliquez ici](#)
- Le tutoriel SysML de Incose : [cliquez ici](#)

SysML France

SysML France :

Emploi recherché

uml java

Localité

France

Tri par

Pertinence

Poste

Développeur Java J2EE

Développeur Java

Développeur Java Angular

Ingénieur Développement Java

Full Stack

Technical Engineer

Localité

France

Ile-de-France

Occitanie

Provence-Alpes-Côte d'Azur

Auvergne-Rhône-Alpes

Pays de la Loire

Nouvelle-Aquitaine

Centre-Val de Loire

Hauts-de-France

Postuler facilement

Offres d'emploi : uml java

101 offres d'emploi

Tous

Nouveaux

Créer une alerte

Stage Logiciel : Customisation d'un outil de modélisation UML H/F

MBDA Systems

Bourges, Cher

Stage Logiciel : Customisation d'un outil de modélisation UML H/F 17/08/2022 Votre environnement de travail Située à 1h45 de Paris en train et 2h des premières pistes de ski,...

Il y a 15 jours plus...

Stage Logiciel : Modélisation UML pour la création d'un domaine speci...

MBDA Systems

Le Plessis-Robinson, Hauts-de-Seine

Stage Logiciel : Modélisation UML pour la création d'un domaine specific language H/F 16/08/2022 Votre environnement de travail Venez partager et développer vos compétences a...

Il y a 2 mois plus...

Développeur Senior Java Fullstack H/F

Potentiel-IT

Aix-en-Provence

La société : Depuis plus de 4 ans chez Potentiel-IT on développe une démarche inversée du recrutement. L'idée c'est de proposer un service d'agent de carrière: élaborer le cahi...

Il y a 1 jour plus...

JAVA - INGENIEUR DEVELOPPEUR SENIOR H/F

Koders

I von

Quoi UML



Où France

Rechercher

Date de publication

Posté par

Télétravail

Estimation du salaire

Type de poste

Informatique

Lieu

Entreprise

Publiez votre CV - Laissez les employeurs vous trouver

Offres d'emploi pour UML : France

Trier par : pertinence - date

294 offres d'emploi

Consultant MBSE – F/H

Accenture 4,0 ★

Paris (75)

Temps plein

- Intégrez un univers où le design de solutions rencontre la technologie de pointe et transformez en profondeur les plus grandes entreprises et institutions.

Offre publiée il y a plus de 30 jours

Maxi 800 + Architecte d'entreprise + BPMN/UML confirmé / Freelance

BlueSoftGroup

Télétravail à Paris (75)

De 600 € à 800 € par jour Temps plein +1

- Participer à la constitution du dossier de choix d'une solution ; - Rédiger les documents d'Architecture (architecture applicative, data et technique) ; -...

Offre publiée il y a 12 jour

nouveau

2 083,33 € et +/-mois (224)

2 500,00 € et +/-mois (205)

2 916,67 € et +/-mois (167)

3 333,33 € et +/-mois (104)

3 750,00 € et +/-mois (68)

MBSE – F/H

22,632 avis

avant de continuer sur le site web de l'entreprise.

postuler

Détails du poste

Type de contrat

Temps plein

Description du poste

Dans un souci d'accessibilité et de clarté, l'écriture inclusive n'est pas utilisée dans cette annonce. Les termes employés au masculin se réfèrent aussi bien au genre féminin que masculin.

Et si vous viviez l'expérience du changement à nos côtés ?

Intégrez un univers où le design de solutions rencontre la technologie de pointe et transformez en profondeur les plus grandes entreprises et institutions. Construisez une carrière agile dans un environnement de travail flexible qui vise à favoriser l'équilibre entre vie professionnelle et vie personnelle : télétravail à la carte, semaine de 4 à 5 jours, projet de mobilité...

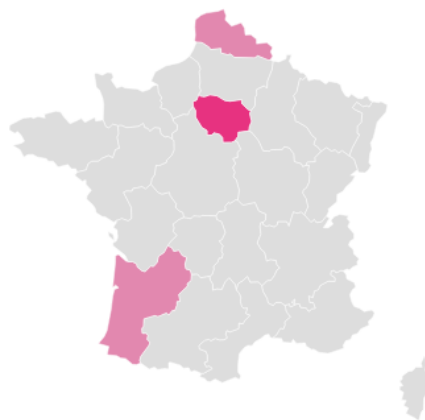
Rejoignez Accenture certifiée Great Place To Work, et venez imaginer, transformer et impacter positivement le monde. Be a change maker.

En tant que Consultant MBSE, vos missions seront :

- Accompagner nos clients dans la transformation digitale de leurs pratiques en Systems Engineering, le « move to MBSE » ou encore l'approfondissement de leurs pratiques dans les domaines suivants : requirements engineering, system

Filtrer parmi 1122 offres d'emploi

Régions



[Ile-de-France](#)

[Aquitaine](#)

[Nord-Pas-de-Calais](#)

Départements

[Gironde](#)

[Nord](#)



Vous voulez-vous dire :

[Cml Innovative Technologies](#)

[UMP](#)

[Union Mutualiste Retraite](#)



ESTIMER MON SALAIRE

[JE DÉPOSE MON CV](#)

En cliquant sur "JE DÉPOSE MON CV", vous acceptez les [CGU](#) ainsi que notre [politique de confidentialité](#) décrivant la finalité des traitements de vos données personnelles.

architecte informatique des données UML MERISE & Pacbase

[Paris 11ème](#) [Carrière Info](#) [Architecte informatique](#)

17 juin - Particularités de la mission ou connaissances particulières souhaitées: Bonne connaissance des référentiels de données bancaires Pratique de la modélisation ...

[Sponsorisé](#)

[Lire la suite](#) [Joblift](#)

Modelisation UML /MEGA obligatoire

[Paris 11ème](#) [Indépendant](#) [Umanis](#)

23 juillet - Modelisation **UML** /MEGA obligatoire Consultant Système d'Information Modélisation Base de données Dans le cadre de l'activité support Bases de ...


[Lire la suite](#) [freelance-info.fr](#)


IT Analyste UML méthodologie F/H





[EKXEL IT](#) [IT](#)



18 juillet - Compétences : * Issu d'une formation supérieure (Bac +3 minimum), Vous venez de terminer vos études ou disposez d'une d'expérience dans une fonction comparable ...

[Lire la suite](#) [jobijobleads](#)



 France

Emplois

Classer par

Date de publication

Fonctionnalités LinkedIn

Entreprise


Niveau d'expérience


Tous les filtres


UML - France


688 résultats


Alerte emploi désactivée



Ingénieur Développement Oracle (F/H)
 Sponsorisé
 Aareon France
 Meudon-la-Forêt, FR
 Recrutement actif
 Il y a 2 semaines • 10 candidats



ATS Project Architect (H/F) Sponsorisé
 Alstom
 Saint-Ouen, FR
 54 anciens élèves travaillent ici
 Il y a 5 jours • 10 candidats



Leader Technique H/F Sponsorisé
 Cnam (Caisse nationale de l'Assurance Maladie)
 Montreuil, FR
 5 anciens élèves travaillent ici
 Il y a 1 semaine • 4 candidats


INGENIEUR DEVELOPPEUR DE LOGICIEL (FM19067) (H/F)
 GROUPE STUDIEL
 Bordes, FR
 Il y a 2 semaines • 1 candidat


Ingénieur Développement C++ (H/F)
 SEGULA Technologies
 Bordes, FR
 8 anciens élèves travaillent ici
 Il y a 2 semaines • 0 candidat


Ingénieur / Ingénieure d'études-développement (H/F)
 Boulogne-Billancourt, FR
 Il y a 4 semaines • 0 candidat


Ingénieur étude et développement C++ (PACA) (H/F)
 Agilitech
 Valbonne, FR


Ingénieur Développement Oracle (F/H)
 Aareon France · Meudon-la-Forêt, FR
 Publié il y a 2 semaines · 93 vues

Poste <ul style="list-style-type: none"> 10 candidats Premier emploi 	Entreprise <ul style="list-style-type: none"> 201-500 employés Technologies et services... 	Relations <p>Vous n'avez aucune relation dans cette entreprise.</p> Ajouter >
---	---	---

Vos Missions

Au sein de l'équipe Recherche & Développement, vous réalisez les développements des solutions logicielles, de la phase d'étude à la mise en production (analyse et formalisation des besoins utilisateurs, spécifications fonctionnelles et techniques, développement, tests, livraison).

Votre esprit d'équipe vous permet de collaborer avec les autres services internes : tests et validation logiciels, consulting, hot-line.

Vos qualités professionnelles (rigueur, autonomie, organisation, force de proposition) vous permettent d'assurer cette mission dans le respect des procédures qualité (ISO 9001, normes groupe) de l'entreprise et dans une démarche d'amélioration continue.

L'ouverture d'esprit et la curiosité vous permettront d'appréhender des domaines fonctionnels riches et variés (gestion locative, demande de logement, gestion de copropriétés...) en lien avec l'offre digitale groupe.

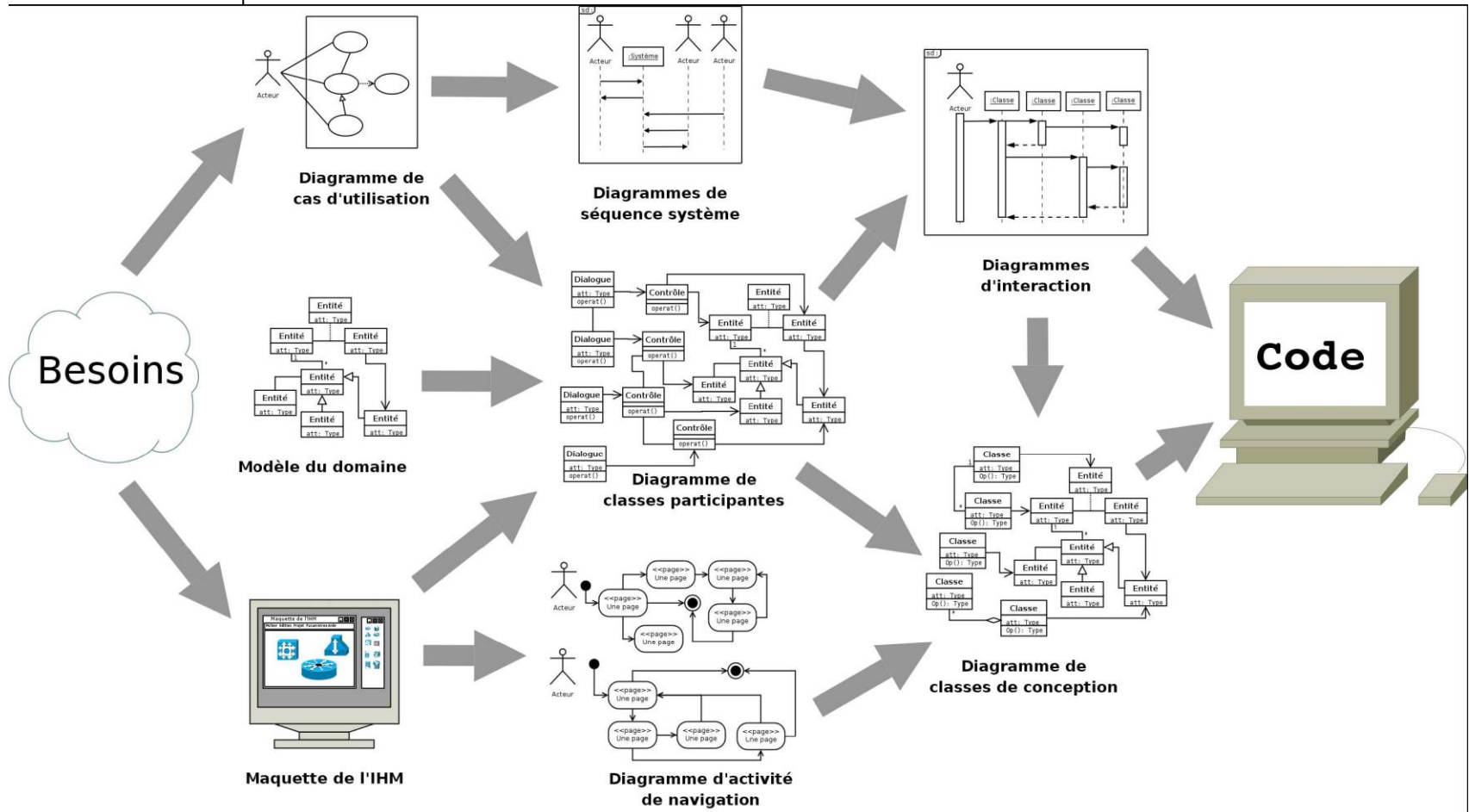
Votre implication facilitera votre évolution dans l'organisation vers de l'expertise ou du management.

Vous Êtes

De formation supérieure en informatique, de type ingénieur ou Master 2, vous justifiez d'une première expérience significative dans le développement logiciel.

Environnement Technique

- Langages de programmation :
 - Applications : SQL, PL/SQL, XML, Java, C, Forms
 - Reporting : SQL Report



<https://www.cours-gratuit.com/cours-uml/cours-pour-apprendre-uml>

2 - Les différents diagrammes



- 2.1 - Le diagramme de cas d'utilisation (diagramme de comportement)
- 2.2 - Le diagramme de classe (diagramme de structure)
- 2.3 - Le diagramme d'objet (diagramme de structure)
- 2.4 - Le diagramme de package (diagramme de structure)
- 2.5 - Le diagramme de séquence (diagramme d'interaction)

Autres diagrammes:

- *Le diagramme d'états-transitions (diagramme de comportement)*
- *Le diagramme d'activité (diagramme de comportement)*
- *Le diagramme de composants (diagramme de structure)*
- *Le diagramme de déploiement (diagramme de structure)*
- *Le diagramme de Communication/Collaboration (diagramme d'interaction)*
- *Le diagramme de vue global des interactions (diagramme d'interaction)*
- *Le diagramme de structure composite (diagramme de structure)*
- *Le diagramme de temps (diagramme d'interaction)*
- *Le diagramme de profils*
- *Langage d'expression de contraintes sur les objets : OCL*

2.1 - Le diagramme de cas d'utilisation



- Fonctionnalité
- Notation
- Mise en œuvre
- Exemple



Les cas d'utilisation (*use cases*) permettent de :

- Spécifier ce qu'il sera possible de demander de l'extérieur à l'entité ainsi représentée
- Spécifier une fonctionnalité offerte par cette même entité
- Capturer les exigences fonctionnelles selon le point de vue des utilisateurs
- Structurer les besoins des utilisateurs et les objectifs correspondants d'un système
- Centrer l'expression des exigences du système sur ses utilisateurs
- Identifier les utilisateurs du système (acteurs) et leur interaction avec le système
- Classer les acteurs et structurer les objectifs du système
- Servir de base à la traçabilité des exigences d'un système dans un processus de développement intégrant UML (tests, cahier de recette, ...)

Cas d'utilisation = Description + Diagramme



- Description textuelle des cas d'utilisation
- Diagrammes cas d'utilisation
- Acteurs
- Relations
- Système
- Note et commentaire
- Package

2.1.1 - Description textuelle des cas d'utilisations



Titre : 1 – Nom du scénario

Début cas d'utilisation : descriptif

Fin cas d'utilisation : descriptif

Condition :

pré-condition : descriptif

post-condition : descriptif

Acteur :

- Acteurs principaux
- Acteurs secondaires
- Matériel externe
- Autre systèmes

Scénario principal :

1.1 - descriptif de l'action (boucles, situations optionnelles)

1.2 - descriptif de l'action

1.n - etc...

Extension :

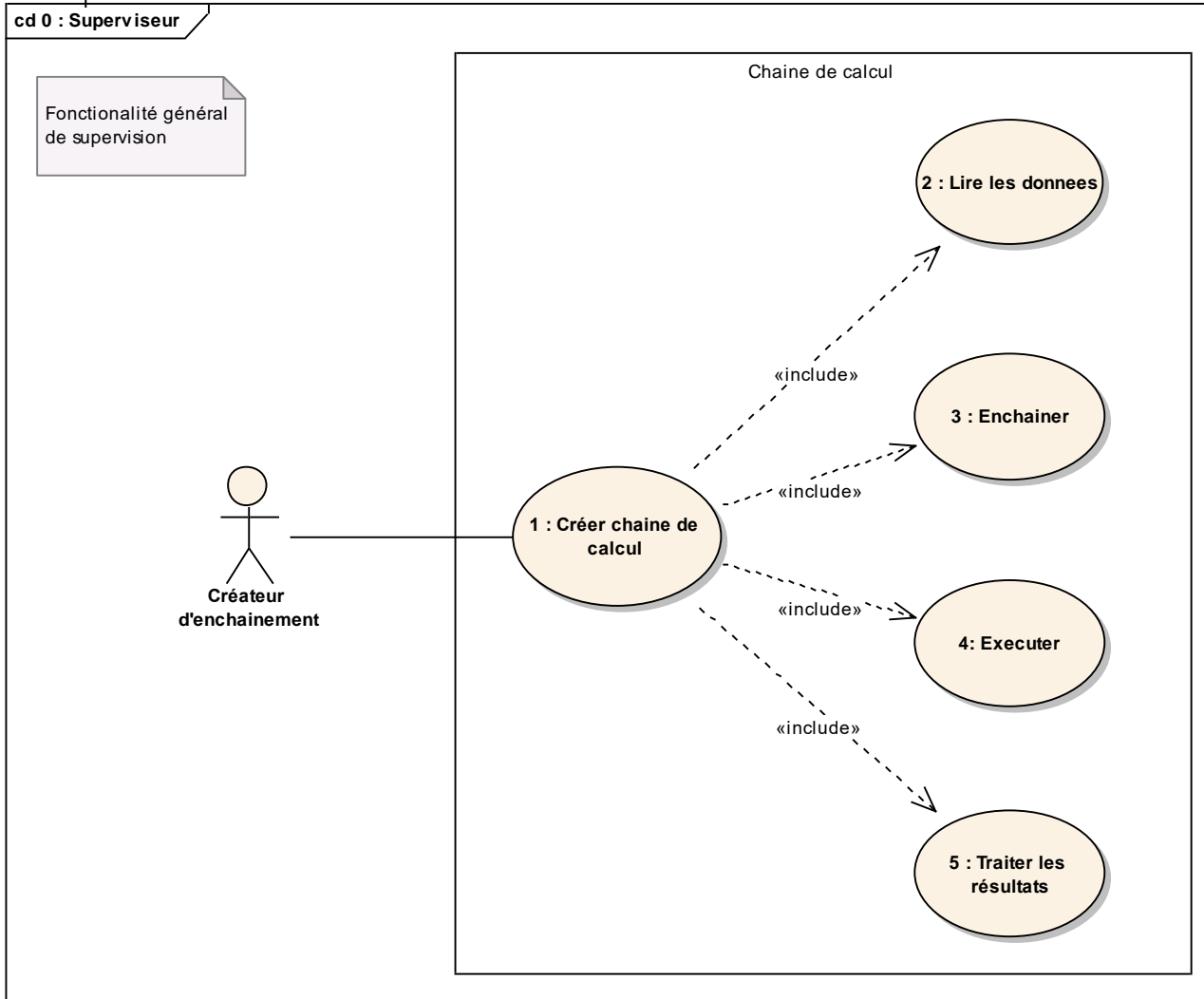
1.1 Créer la fiche modèle

1.1.1 - descriptif de l'action (boucles, situations optionnelles)

1.1.2 - descriptif de l'action

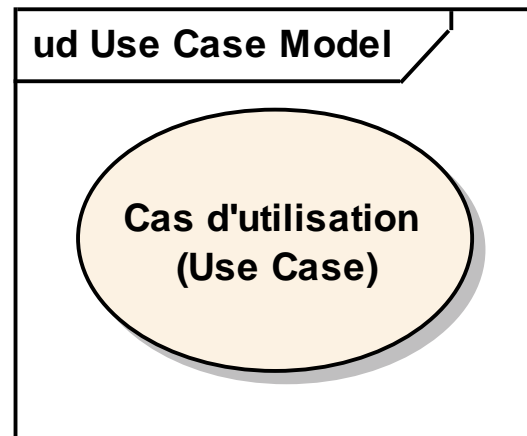
1.1.k - etc...

2.1.2 - Diagrammes cas d'utilisation



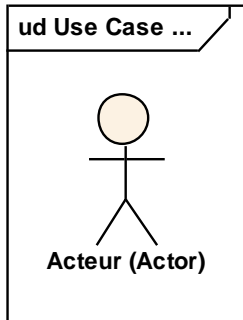


Les cas d'utilisation représentent un ensemble de scénarios d'interaction du système avec un acteur extérieur, reliés par un but commun





Les acteurs représentent le rôle d'un homme, d'une machine, d'un système extérieur ou du temps qui interagissent avec le système



4 sortes d'acteur :

- Acteurs principaux
- Acteurs secondaires
- Matériel externe
- Autres systèmes



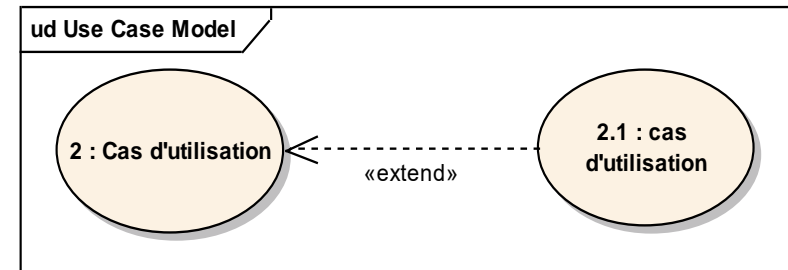
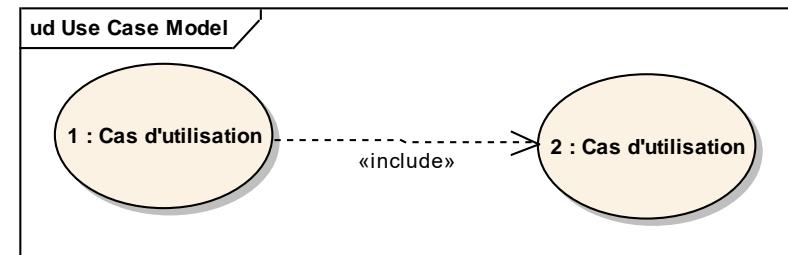
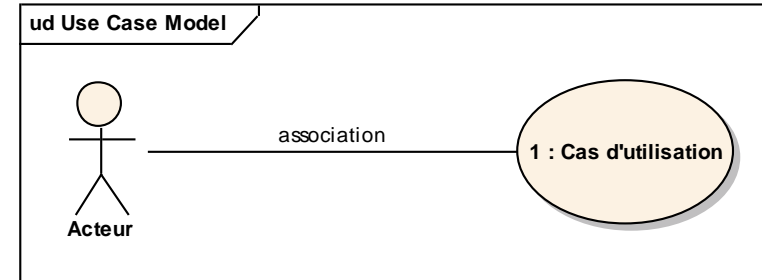
Un acteur représente le rôle d'une entité externe (*utilisateur humain ou non*) interagissant avec le système.

On représente généralement à gauche l'acteur principal, et à droite les acteurs secondaires.

Remarque : Un utilisateur peut amené à jouer plusieurs rôles vis-à-vis du système et à ce titre être modélisé par plusieurs acteurs.



- **Association entre acteur et cas**
 - un acteur interagit avec un cas d'utilisation en le déclenchant
- **Relation d'inclusion**
 - Le cas d'utilisation source (1) contient aussi le comportement décrit dans le cas d'utilisation destination (2) (*a besoin de*)
- **Relation d'extension**
 - Le cas d'utilisation source (2.1) précise le comportement du cas d'utilisation destination (2) (*peut avoir besoin de*)

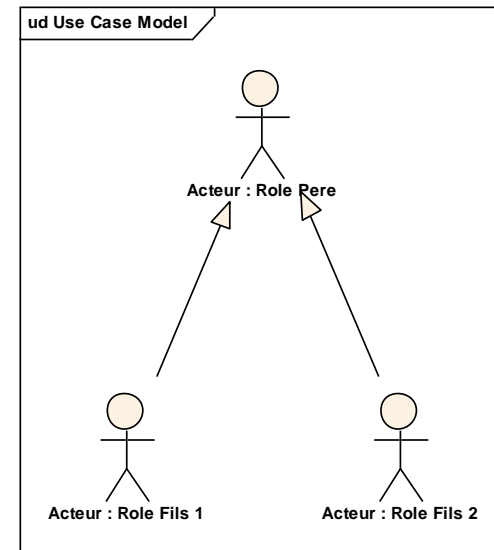
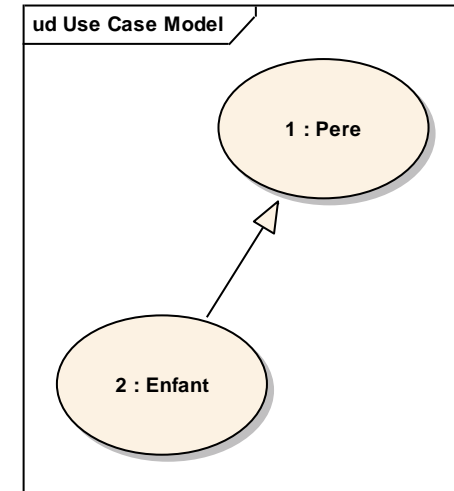




Quand un cas n'est pas directement relié à un acteur, mais qui est utiliser par un autre cas d'utilisation => il est qualifié de cas d'utilisation interne.

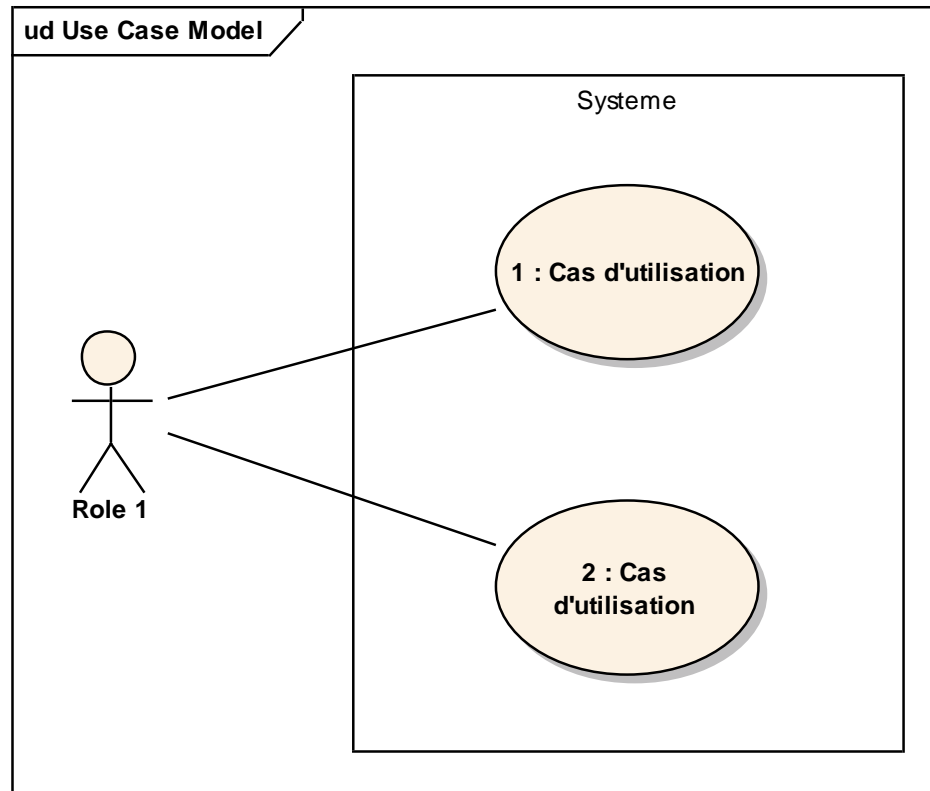


- **Relation de Généralisation entre cas d'utilisation**
 - Le cas d'utilisation enfant (5) est une spécialisation du cas d'utilisation père (4)
- **Relation de Généralisation entre acteurs**
 - Meilleure identification des rôles



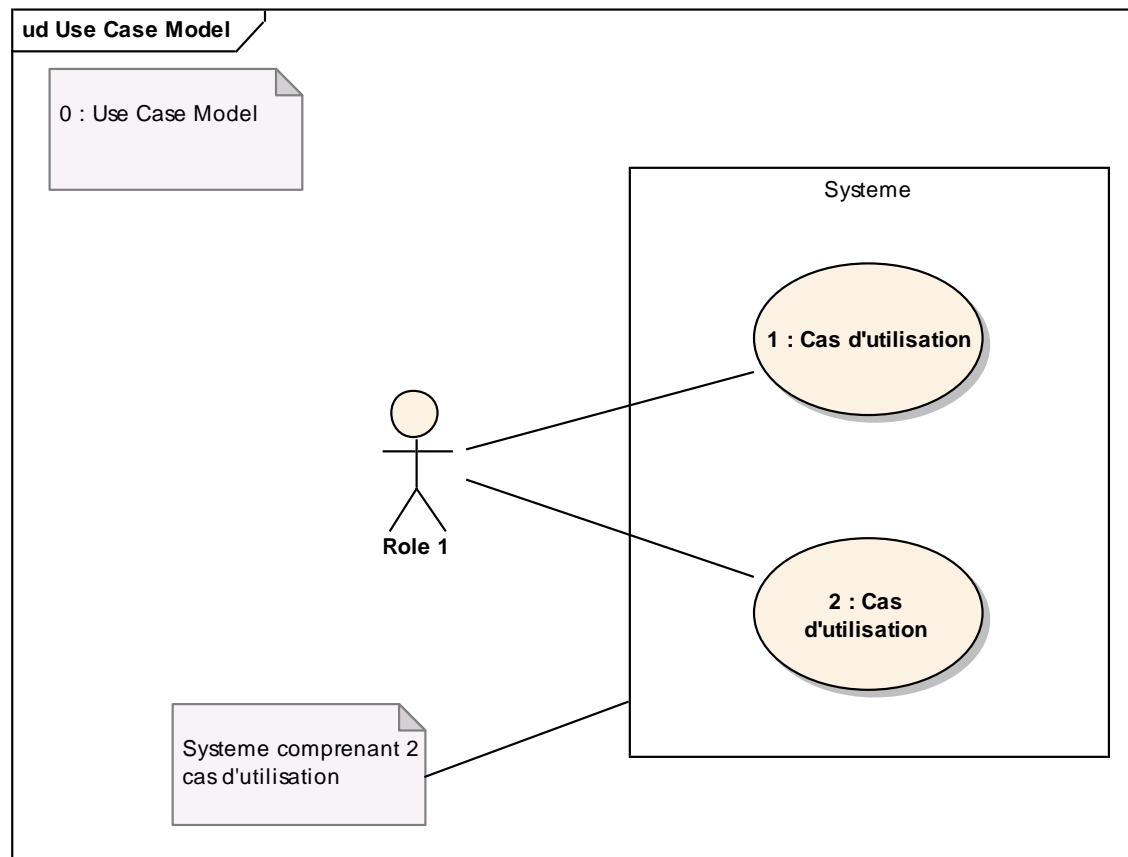


Les cas d'utilisation peuvent être contenus dans un rectangle représentant les limites du système





Une note est un symbole graphique qui contient des informations



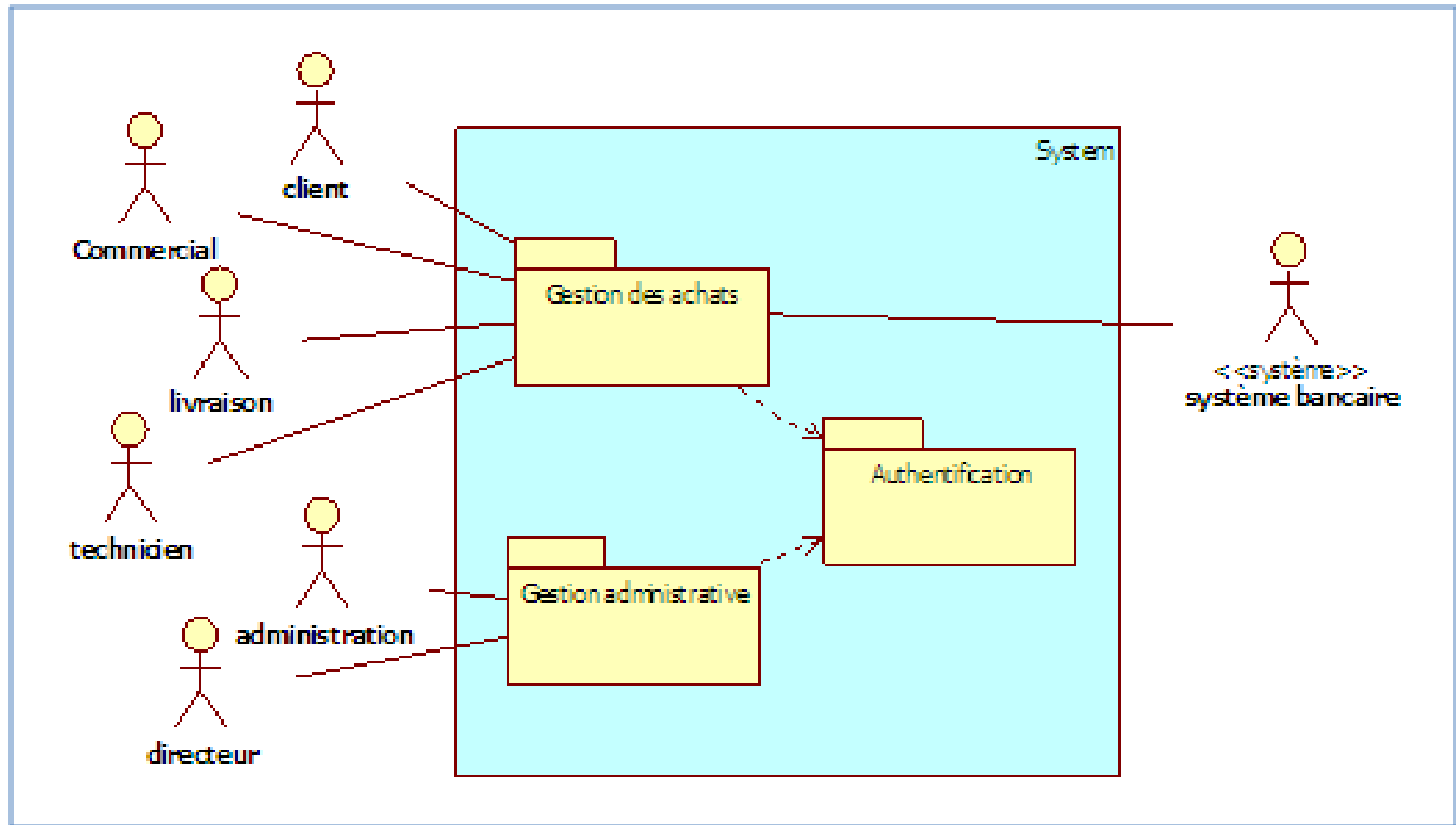


Le diagramme de packages pour les use case, permet de décomposer le système en catégories ou parties plus facilement observables, appelés « packages ».

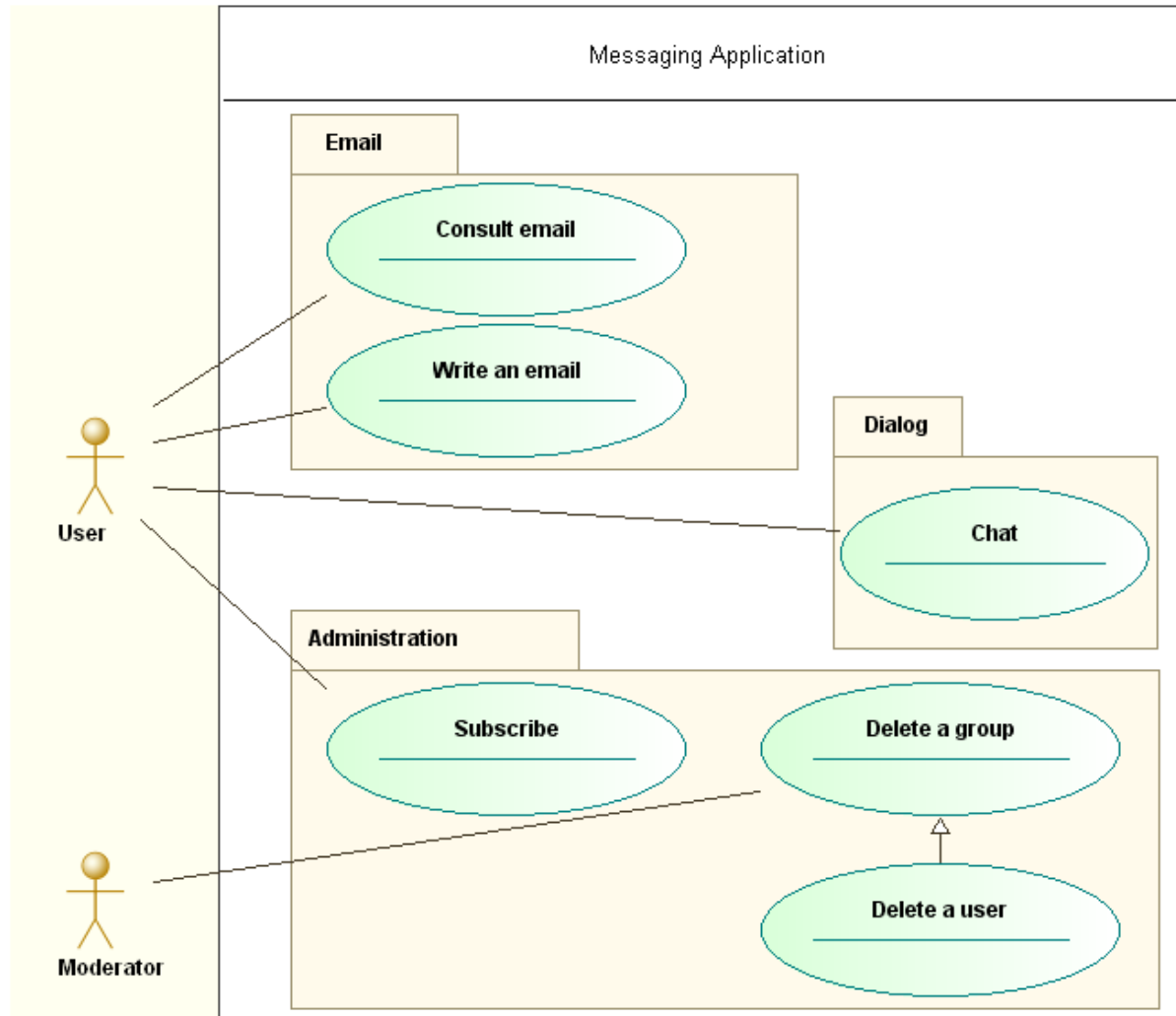
Un package est constituer de plusieurs fonctionnalités qui forment une famille.

Cela permet également d'indiquer les acteurs qui interviennent dans chacun des packages.

2.1.8 - Package



2.1.8 - Package





- **Écrire les premiers cas d'utilisation globaux**
- **Détailler les parties principales**
- **Développer les parties**

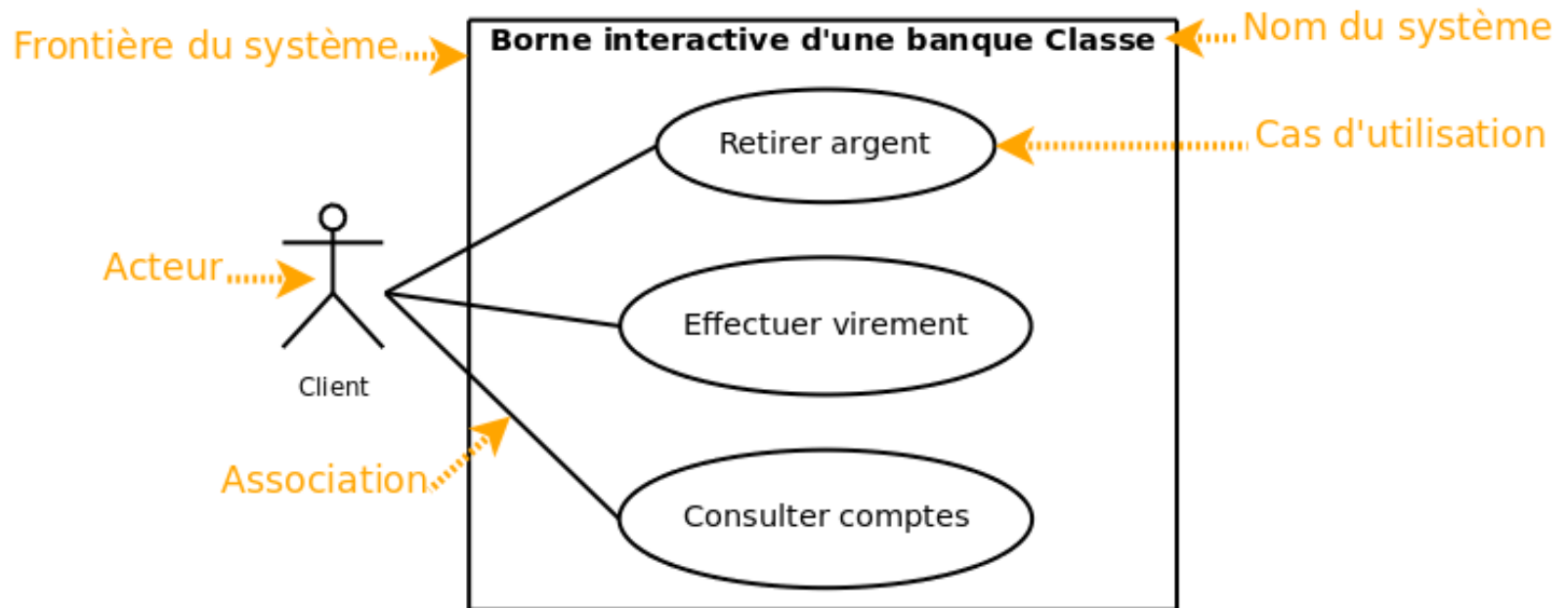
Attention :

- **Limiter les cas d'utilisation : 10-20 maximum pour un projet moyen**
- **Faire des cas d'utilisation simples**

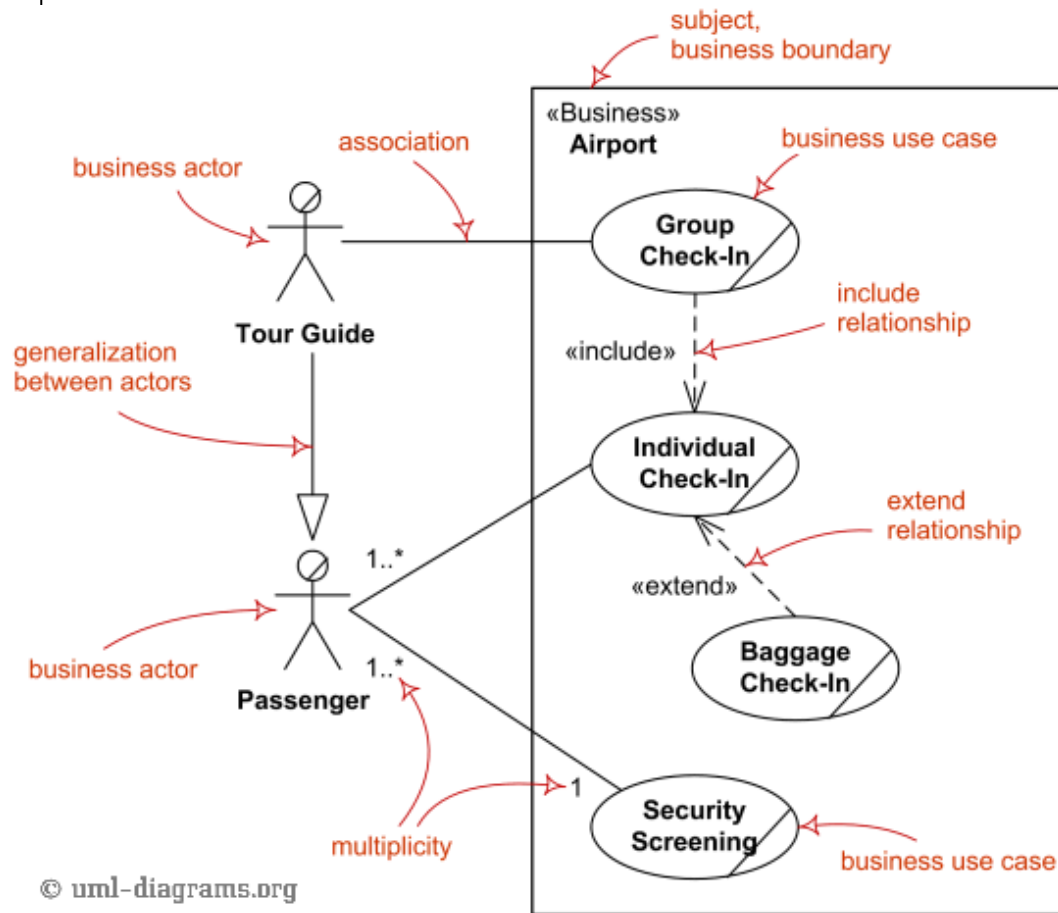
2.1.4.1 – Exemple : Borne Interactive d'une banque



<https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-cas-utilisation>

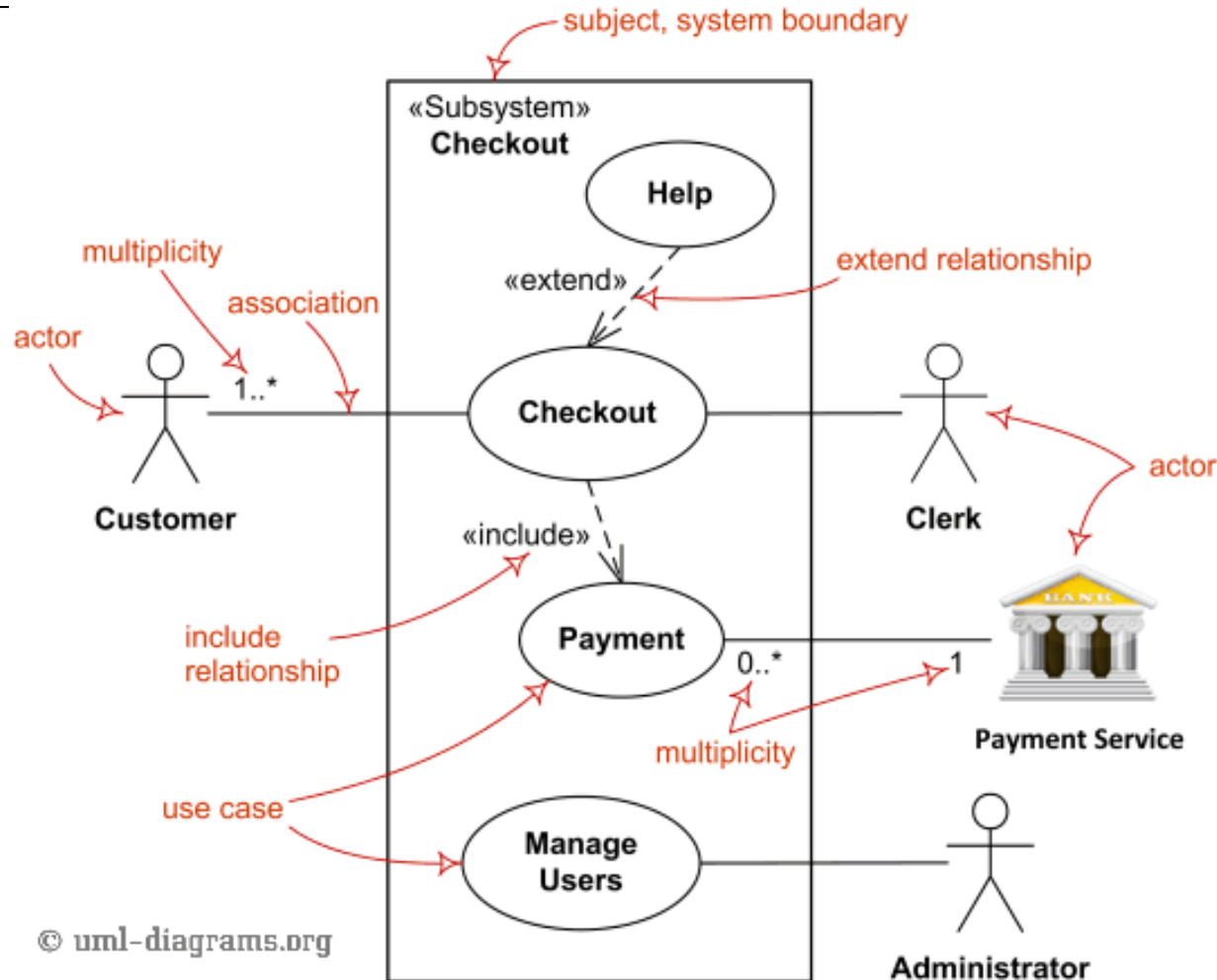


2.1.4.2 – Exemple : Business Use Case Diagrams



<https://www.uml-diagrams.org/use-case-diagrams.html>

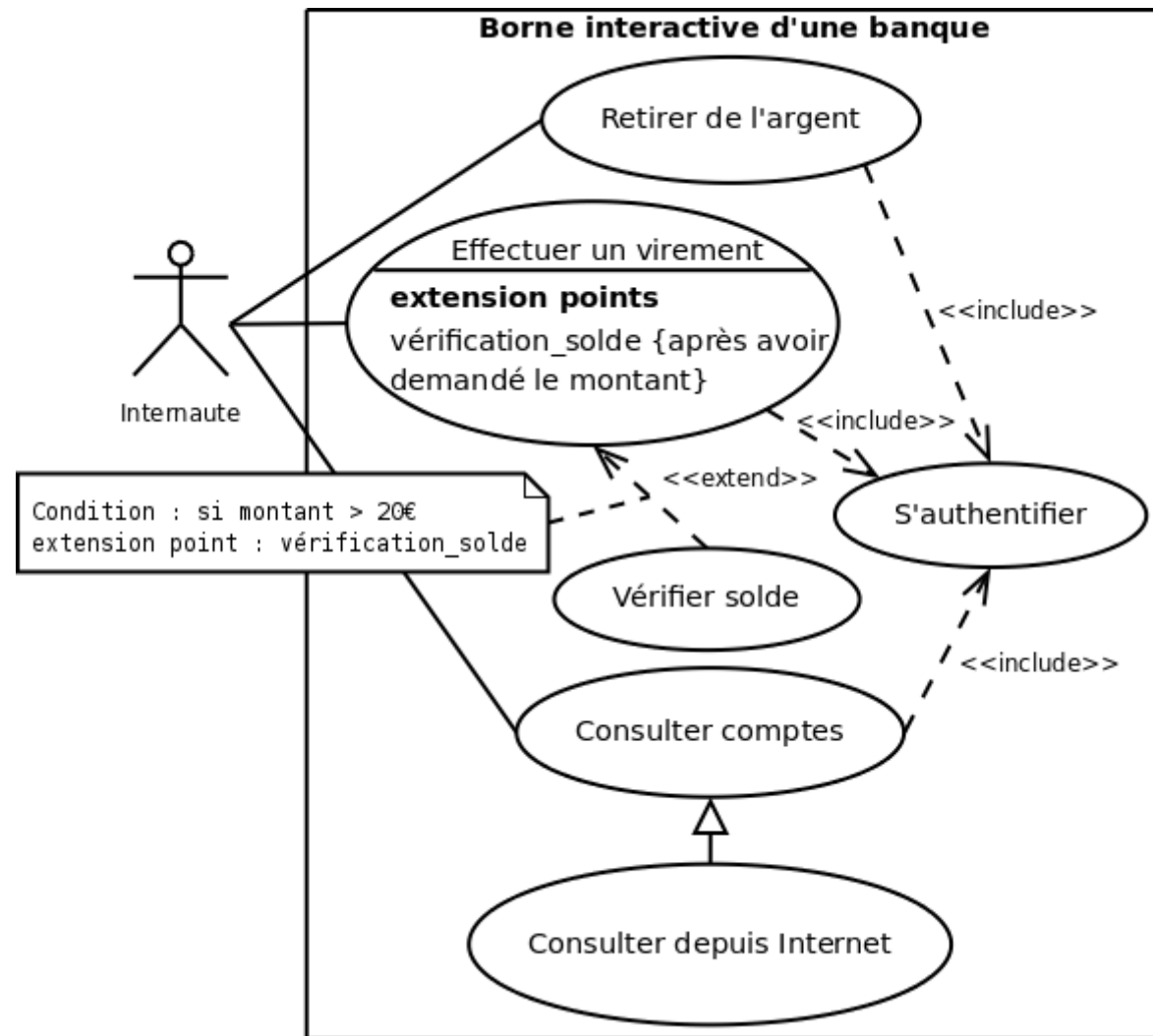
2.1.4.3 – Exemple : System Use Case Diagrams



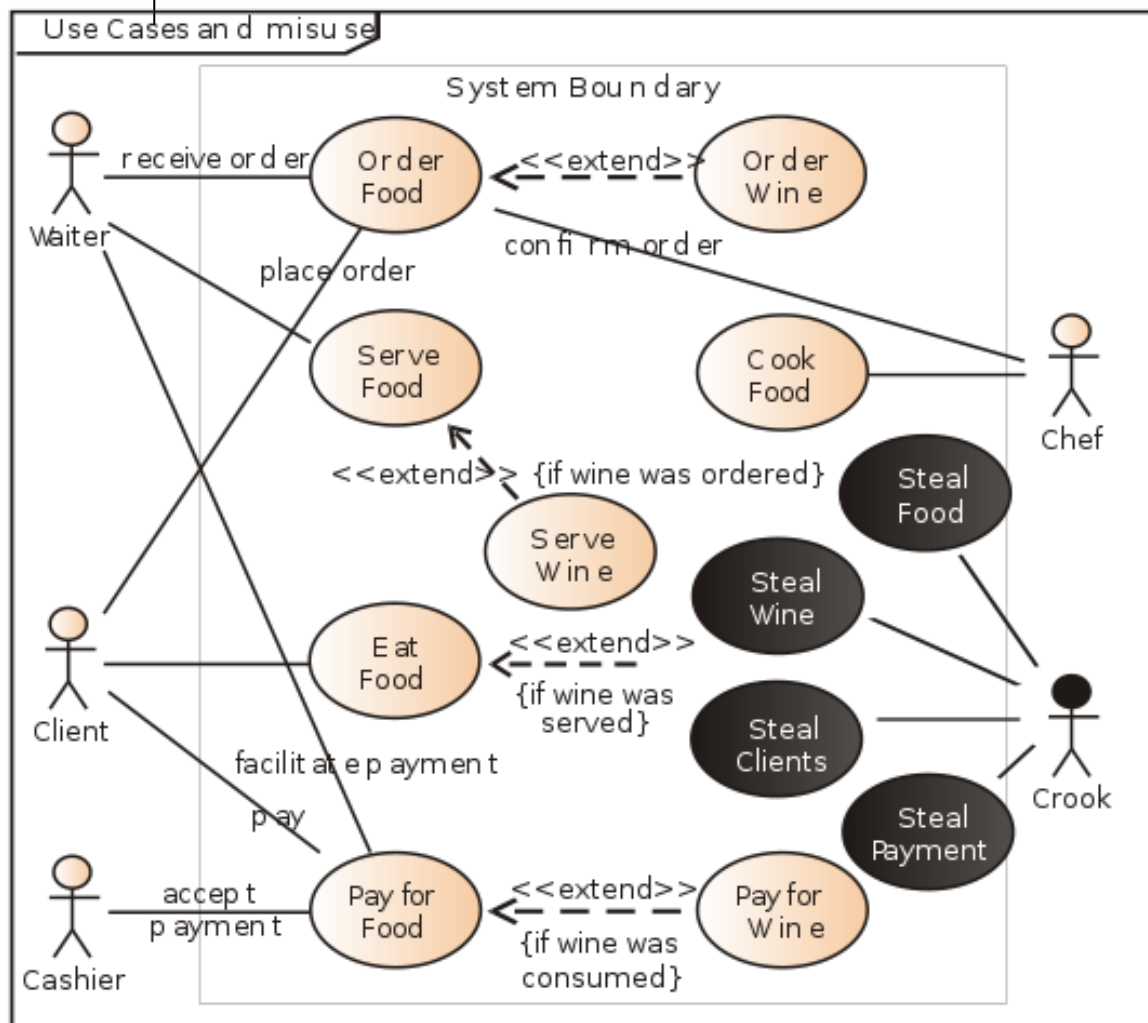
<https://www.uml-diagrams.org/use-case-diagrams.html>

<https://www.uml-diagrams.org/use-case-diagrams-examples.html>

2.1.4.4 – Exemple : : Borne Interactive d'une banque



2.1.4.5 - Exemple

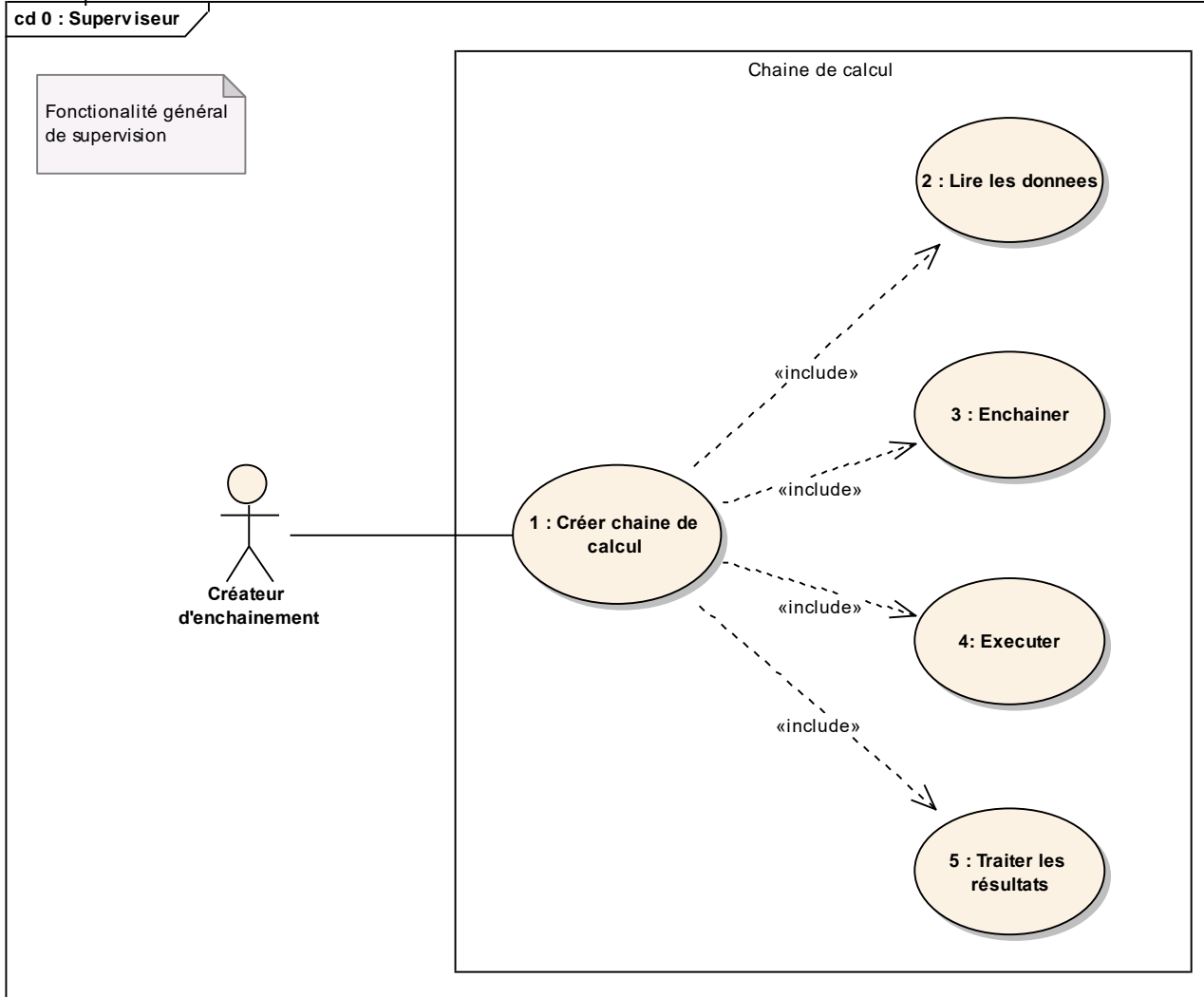


https://en.wikipedia.org/wiki/Misuse_case#From_use_to_misuse_case



- Exemple : Chaîne de calcul
- Exemple : Lecture des données
- Exemple : Enchaînement de code

2.1.4.6a – Exemple : Chaîne de calcul



2.1.4.6b – Exemple : Lecture des données



ud 2 : Lire les donnees

2 : Lire les données

:Créateur
d'enchainement

Lecture données

2.1 : Rentrer les
données

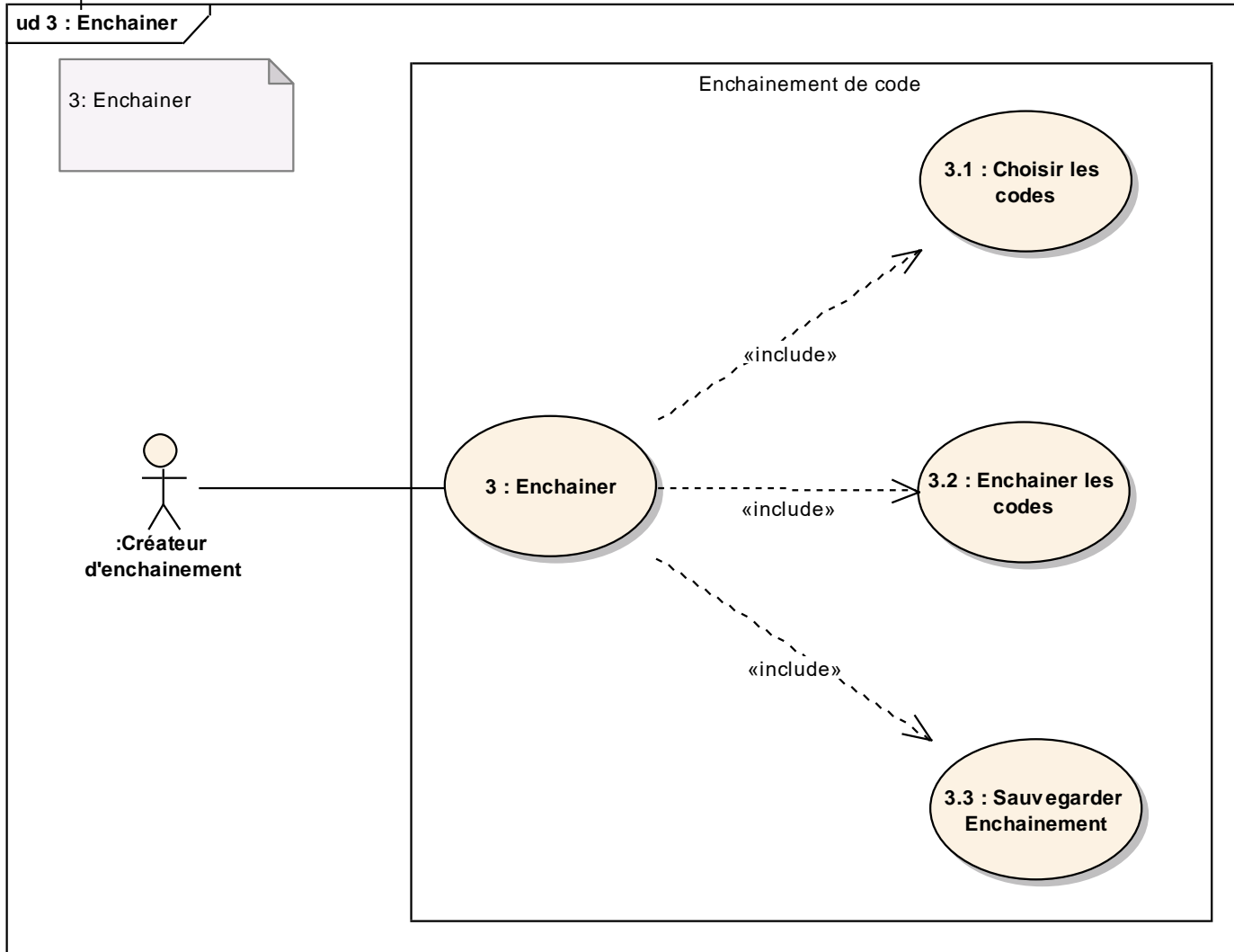
«include»

2 : Lire les
donnees

«include»

2.2 : Mettre en
forme les données

2.1.4.6c – Exemple : Enchaînement de code





Yantra Technologies



2.2 - Le diagramme de classe



- Fonctionnalité
- Notation
- Mise en œuvre
- Exemple



Un diagramme de classe est une collection d'éléments de modélisation statiques (classes, paquetage, ...) qui permet de:

- Montrer la structure statique d'un problème en termes de classes et de relations entre ces classes
- Faire abstraction des aspects dynamiques et temporels
- Représenter un contexte précis : un diagramme de classes peut être instancié en diagramme d'objets
- Définir la structure statique d'un problème
- D'organiser l'espace du problème (analyse : catalogue les objets) ou de la solution (conception : organise le système)

2.2.2 - Notation

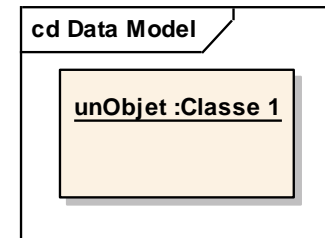
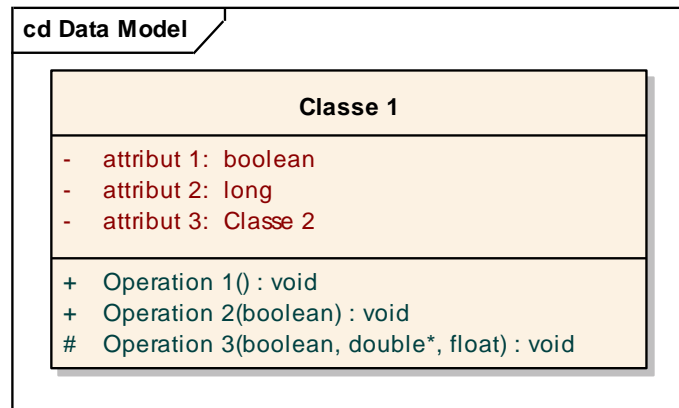


- **Classes**
- **Attributs**
- **Opérations**
- **Attributs et Opérations statiques**
- **Association**
- **Multiplicité**
- **Navigabilité**
- **Agrégations**
- **Compositions**
- **Associations qualifiées**
- **Classes associations**
- **Propriétés dérivées**
- **Dépendance**
- **Généralisation/classification statique**
- **Héritage multiple (A éviter en conception)**
- **Classification dynamique**
- **Interface et classes abstraites**
- **Classes actives**
- **Classes paramétrables**
- **Énumérations**
- **Visibilité**



Une **classe** est un **type abstrait** caractérisé par des **propriétés** (attributs, opération et états) communes à un ensemble d'objets et permettant de créer des objets ayant ces propriétés.

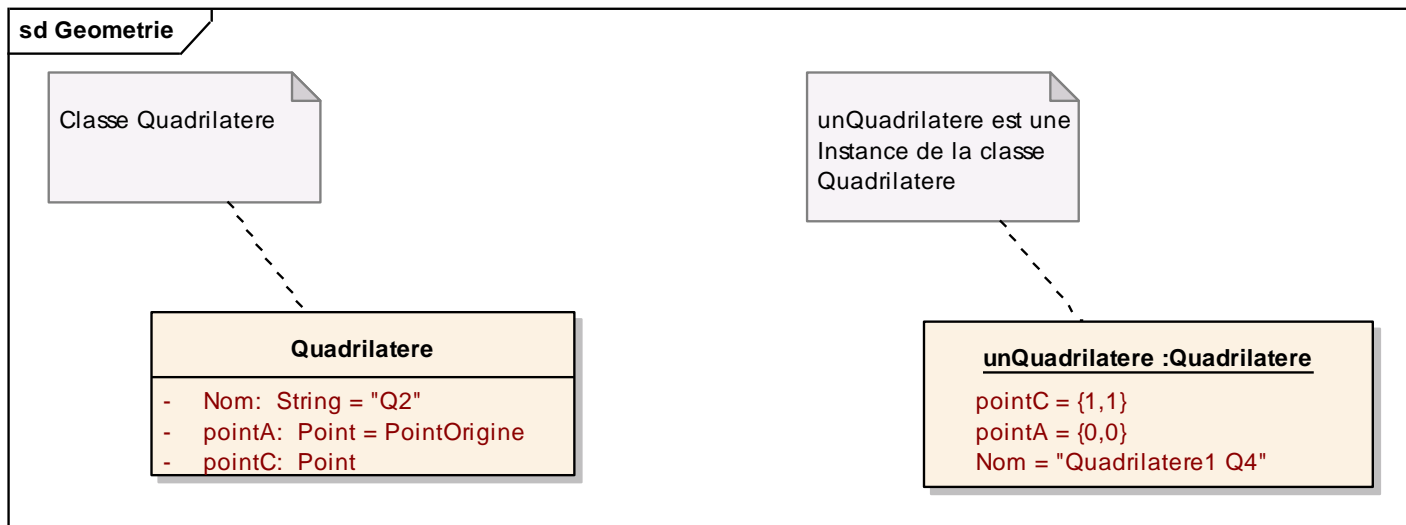
Classe = attributs + opération + instantiation





Les **attributs** correspondent aux propriétés de la classe, définis par

- Un **nom** unique
- Un **type**
- Une **valeur** initiale (*optionnel*)



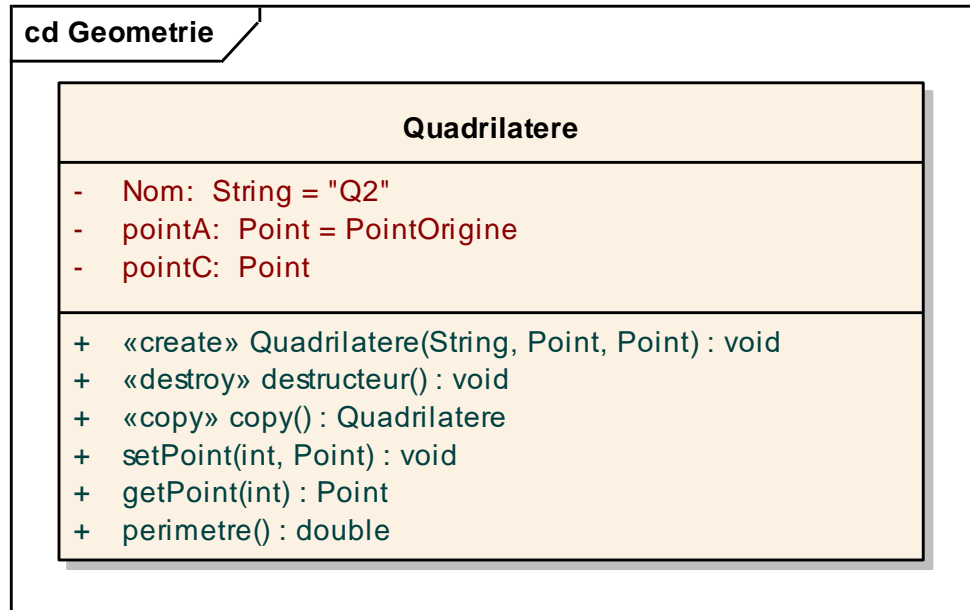


Les opérations spécifient le comportement d'un objet.
La réalisation du comportement est exprimée dans les méthodes

- Une opération est un service offert par les instances de la classe
- Une méthode est l'implémentation d'une opération

5 principaux types d'opération :

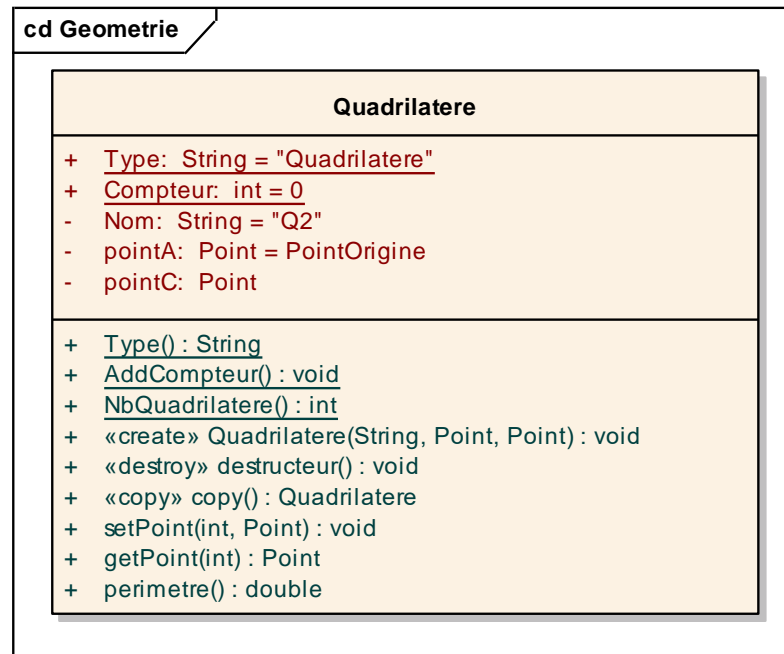
- Constructeurs
- Destructeurs
- Sélecteurs
- Modificateurs
- Itérateurs



2.2.2.4 - Attributs et Opérations statiques



Les attributs ou les opérations statiques s'appliquent à une classe et non à ses instances

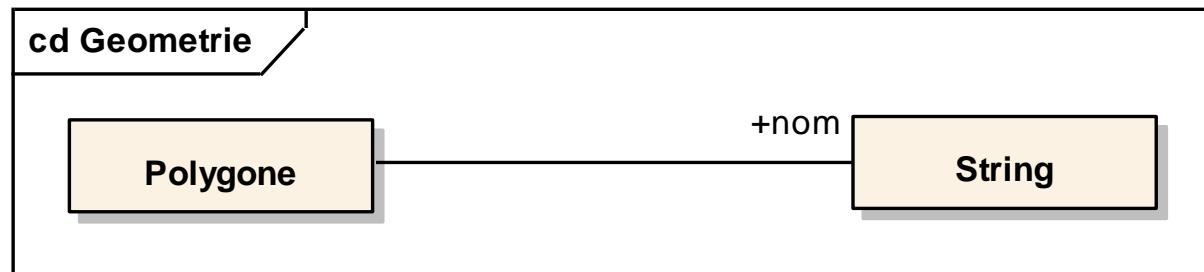


2.2.2.4 - Association



Une association représente des relations structurelles. Elle exprime une connexion sémantique bidirectionnelle entre deux classes .

L'association est instanciable dans un diagramme d'objet ou de collaboration sous forme de liens entre objets issus de classes associées.

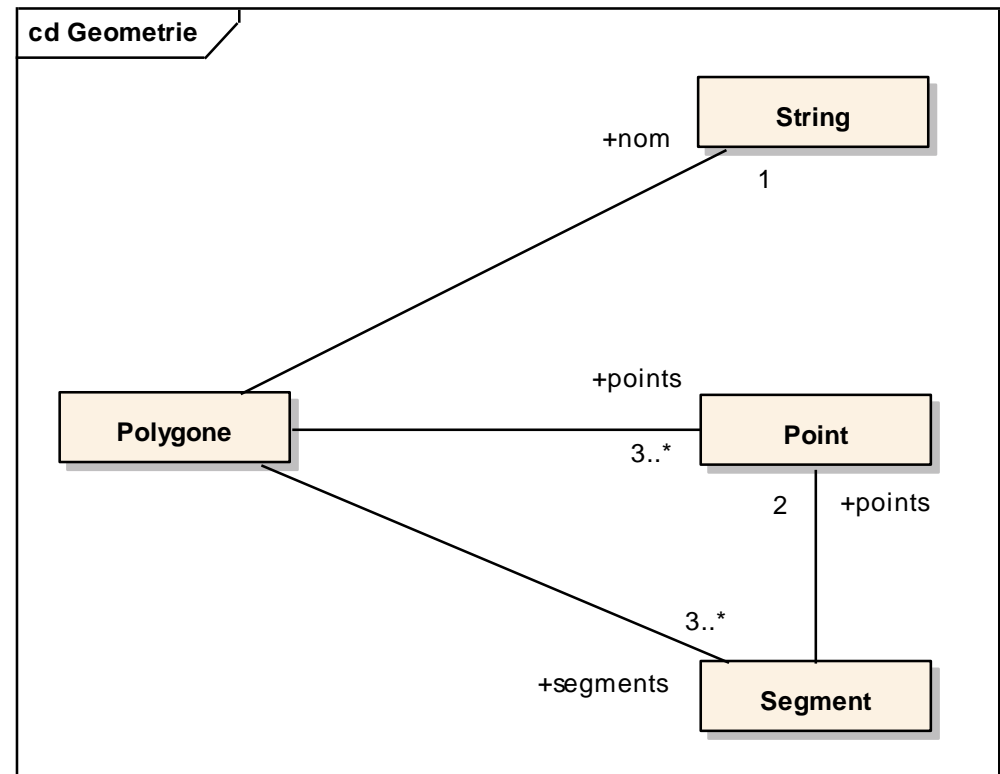


2.2.2.5 - Multiplicité



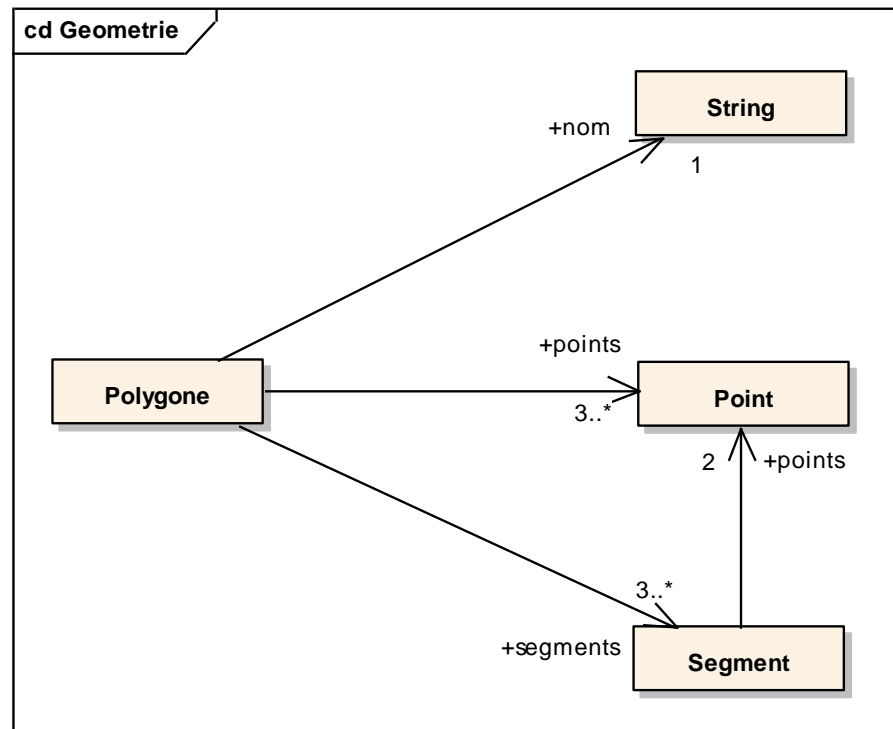
La **multiplicité** spécifie le **nombre d'instances** d'une classe **en relation** avec une instance de la classe associée

- 0..1
- 0..*
- 1..*
- 1
- *
- Autre : 4..6, 5, ...





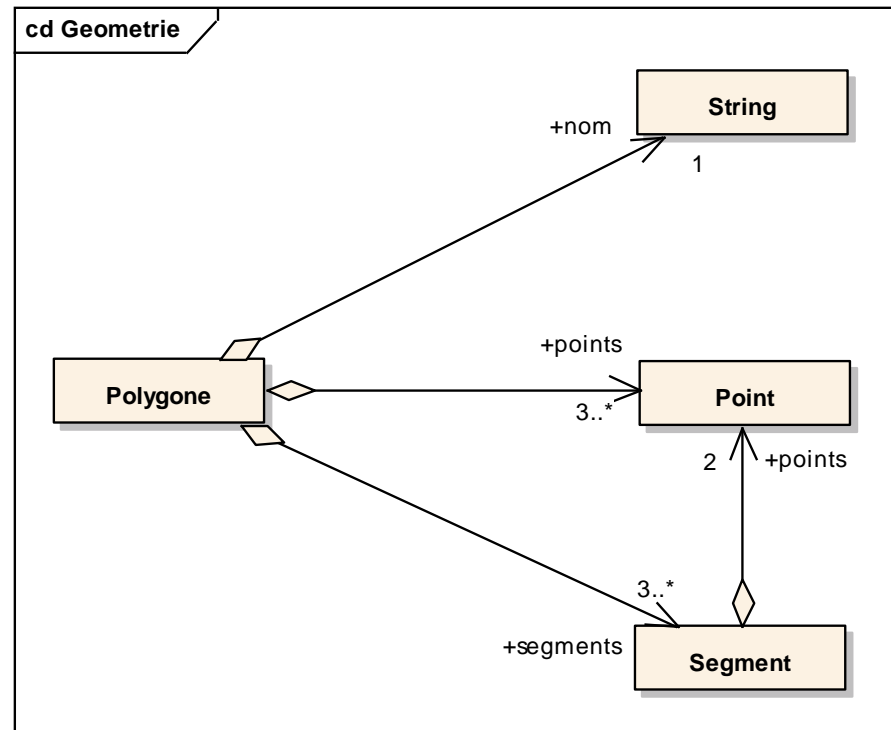
Une association représente des relations structurelles.
Par défaut une association entre deux classes est
bidirectionnelle (faux dans certain cas)





L'agrégation est un cas particulier d'association asymétrique.

Elle permet de modéliser une contrainte d'intégrité.

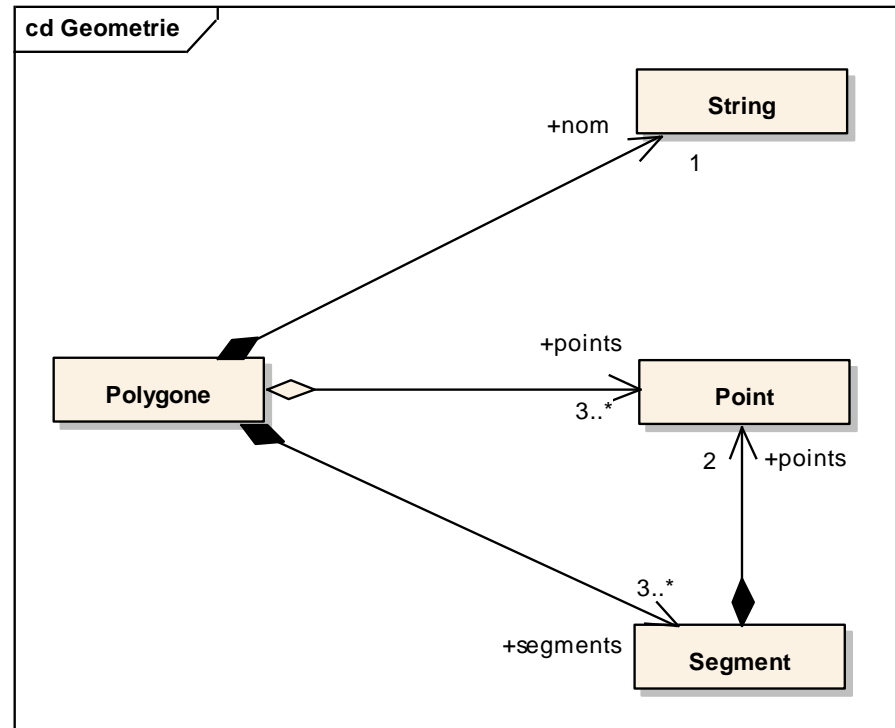


2.2.2.8 - Compositions

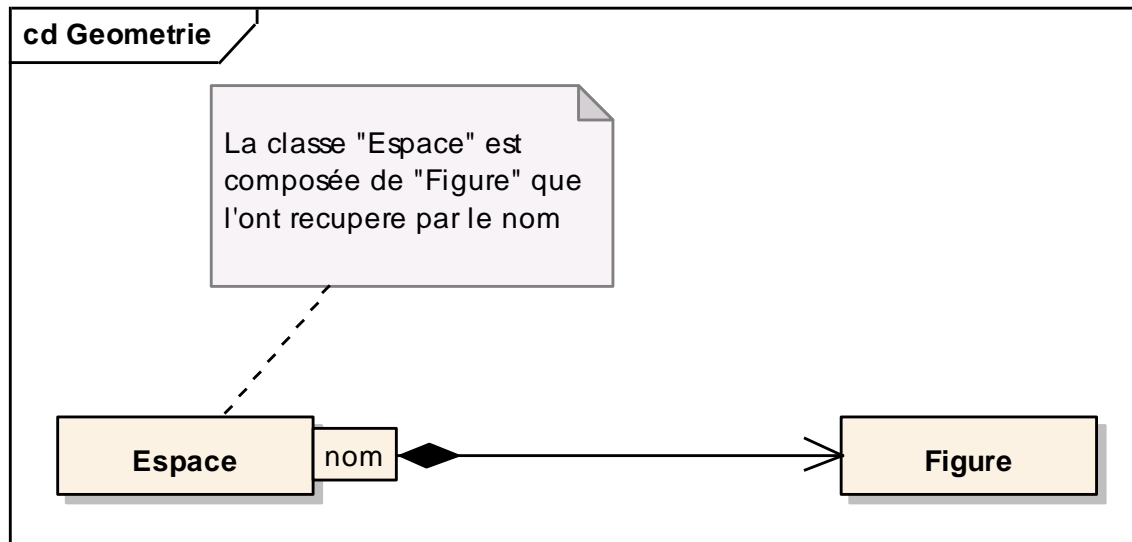


La composition est un cas particulier d'agrégation avec un couplage plus fort.

Elle implique une coïncidence des durées de vie du composant et du composite.

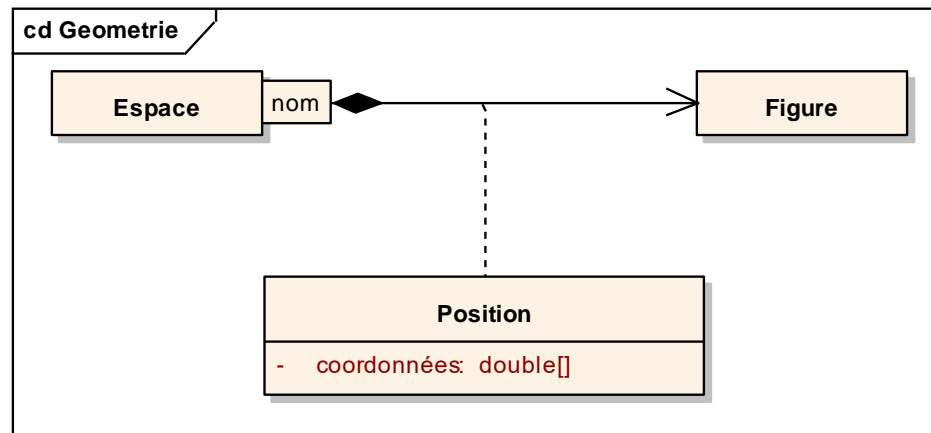


2.2.2.9 - Associations qualifiées





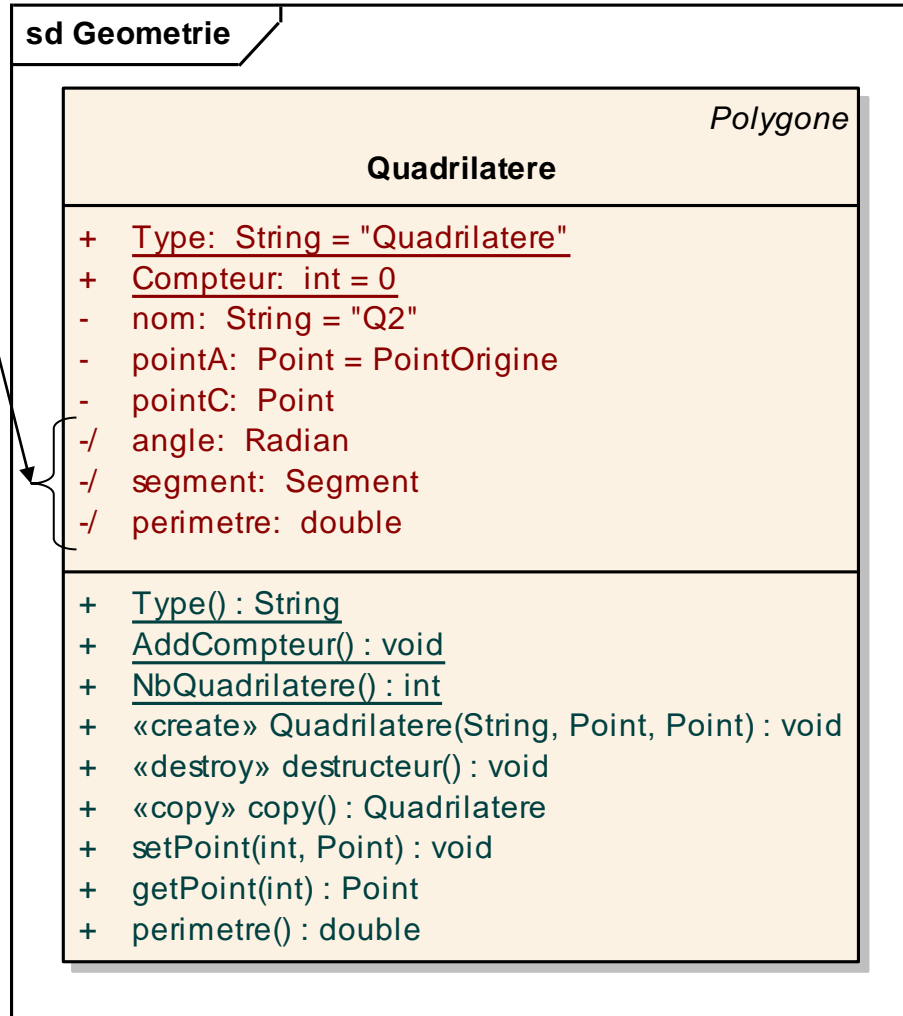
Les classes associations permettent d'ajouter des attributs, des opérations et d'autres fonctionnalités aux associations.



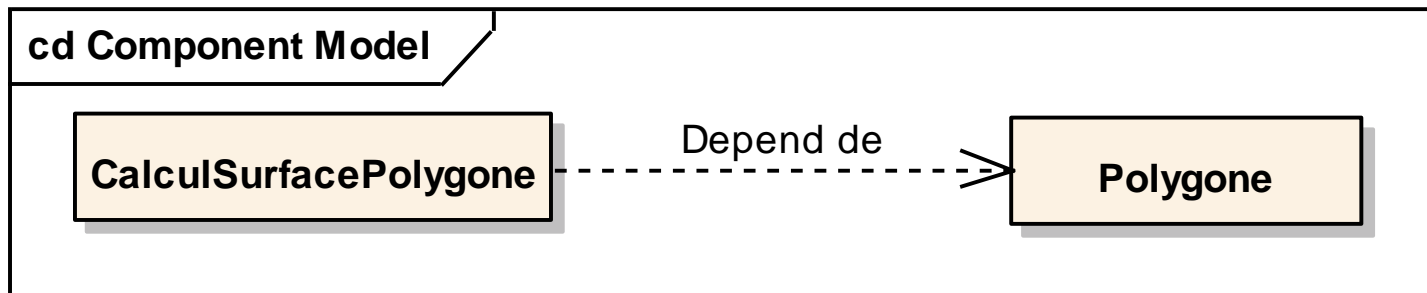
2.2.2.11 - Propriétés dérivées



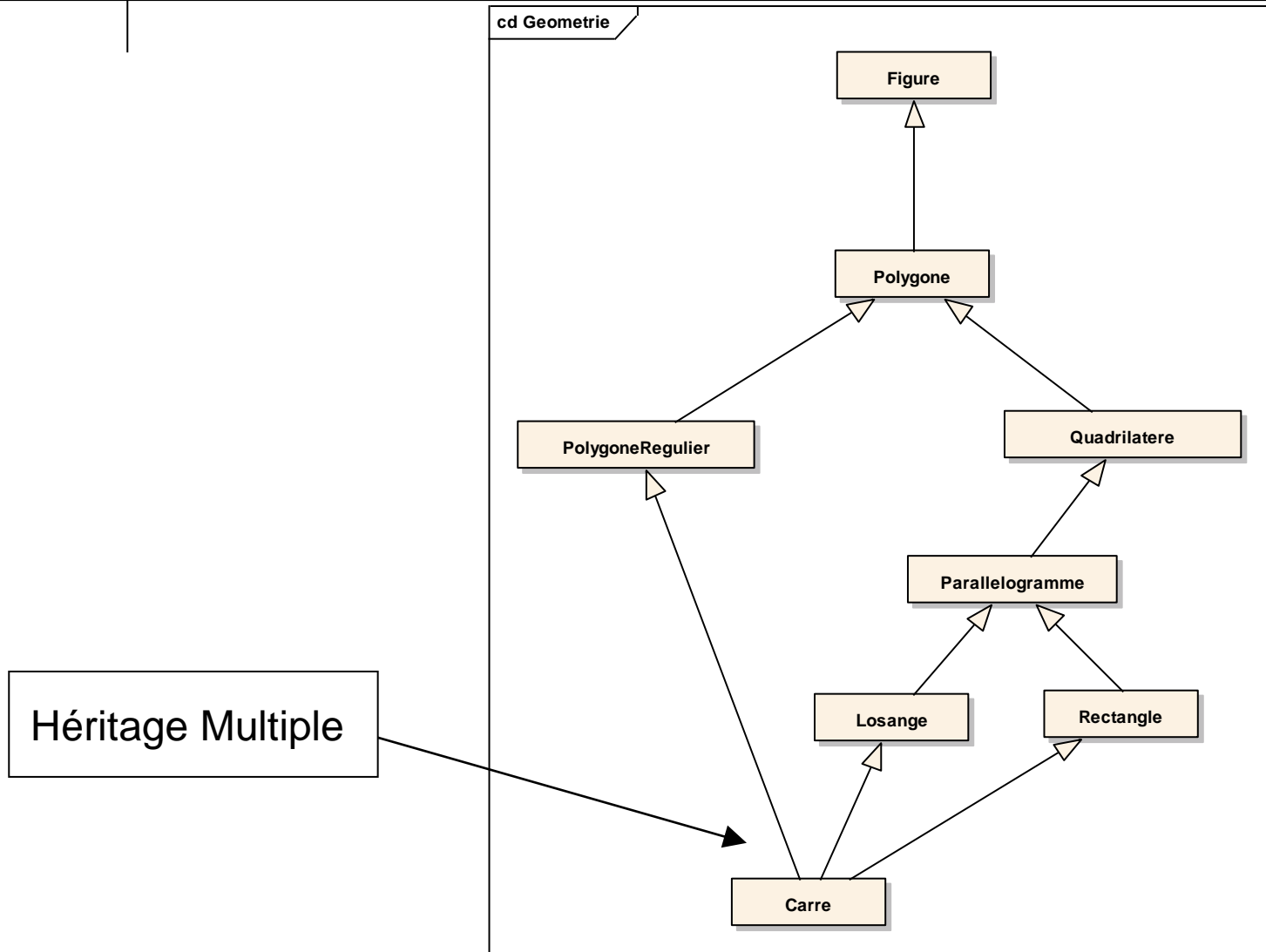
Propriétés dérivées



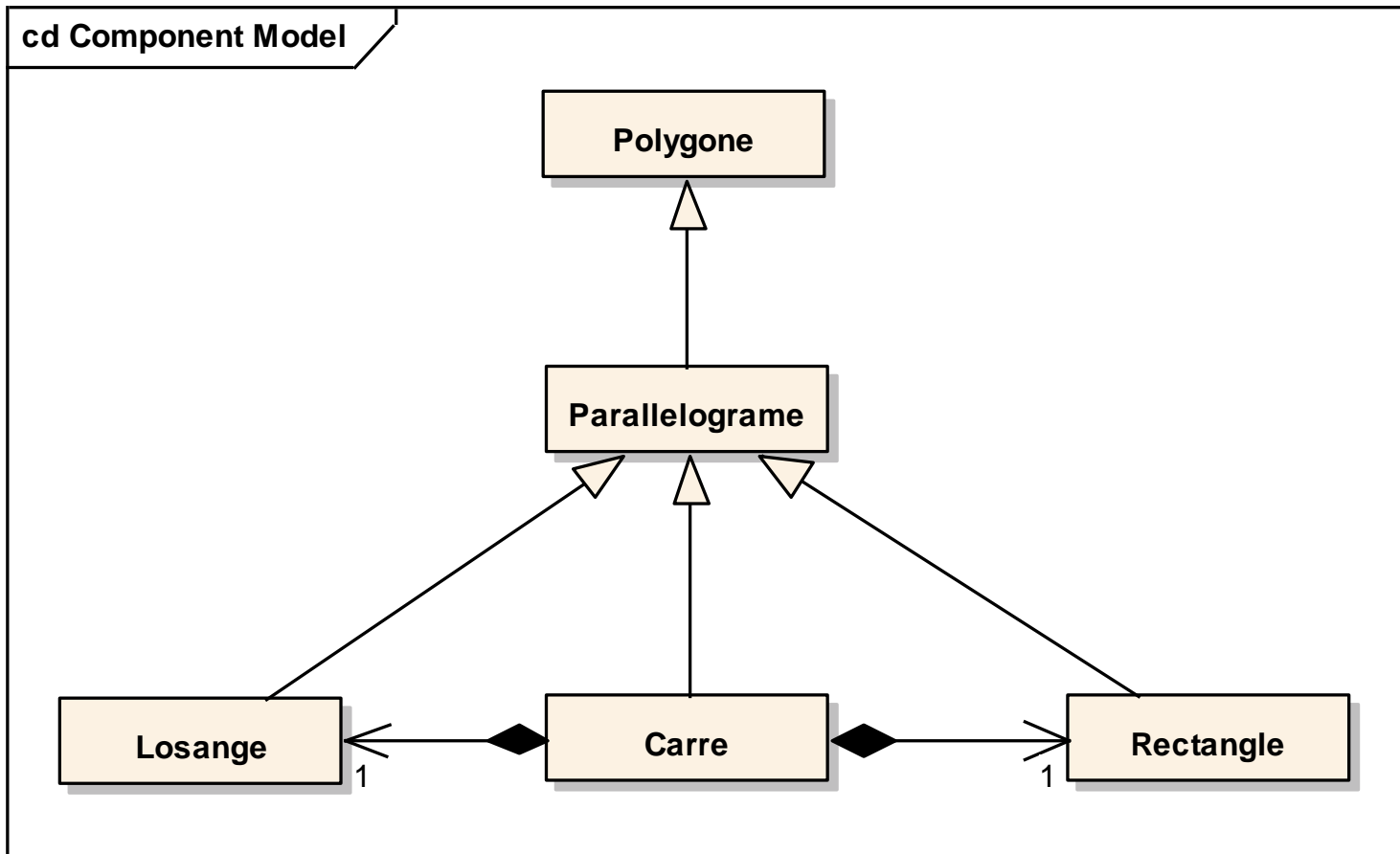
2.2.2.12 - Dépendance

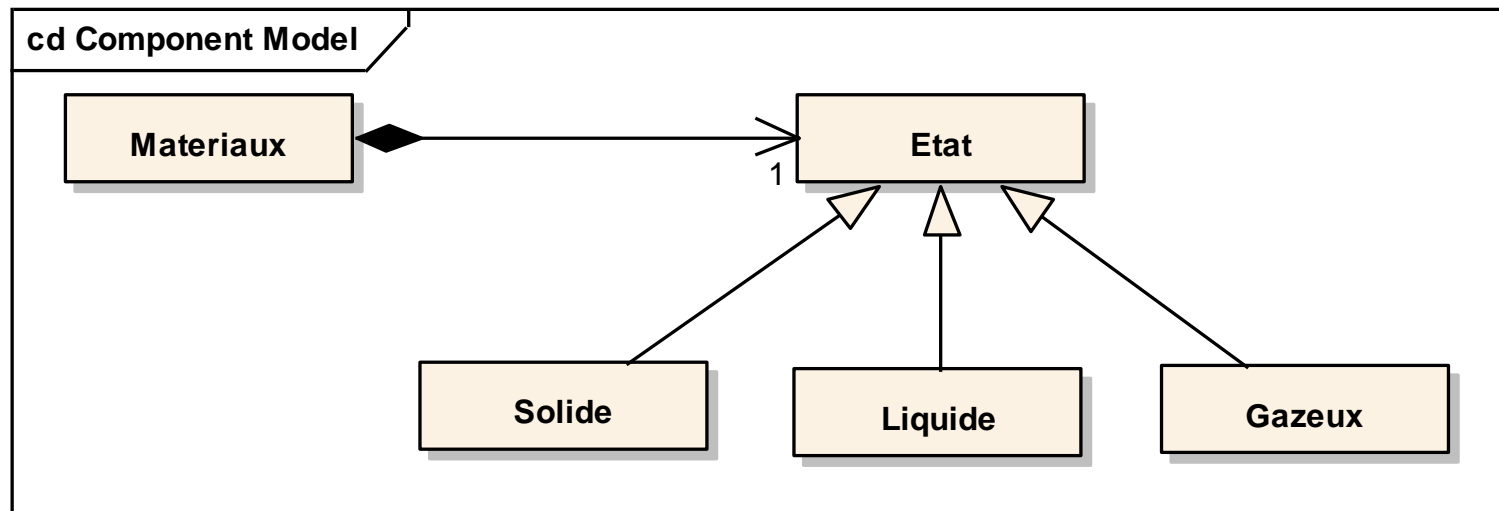


2.2.2.13 - Généralisations/classification statique



2.2.2.14 - Héritage Multiple (A éviter en conception)







Classe Abstraite :

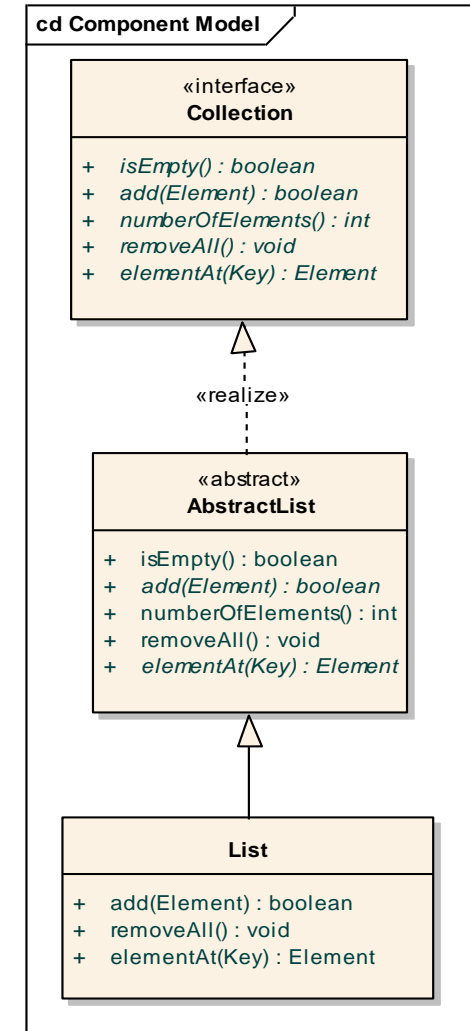
Une classe abstraite est une classe qui ne peut être instanciée directement.

Une opération abstraite n'a pas d'implémentation

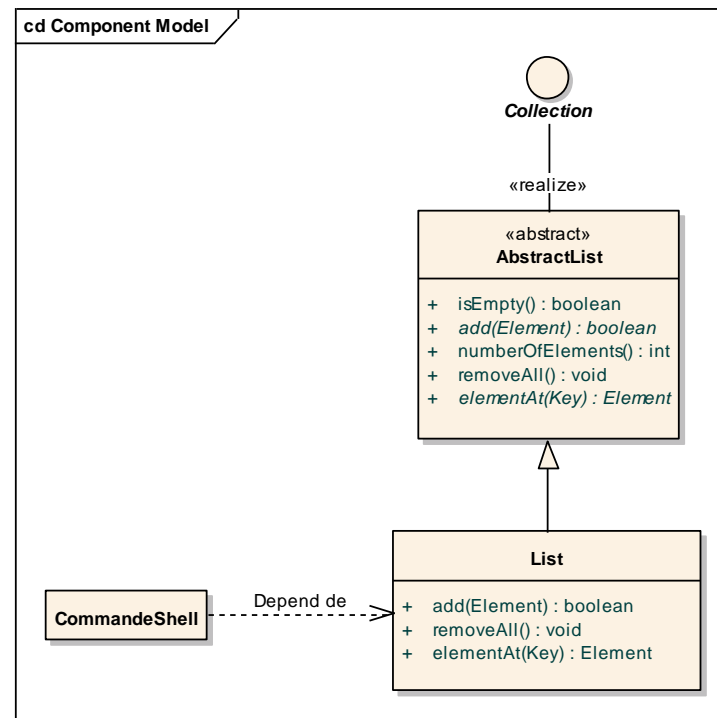
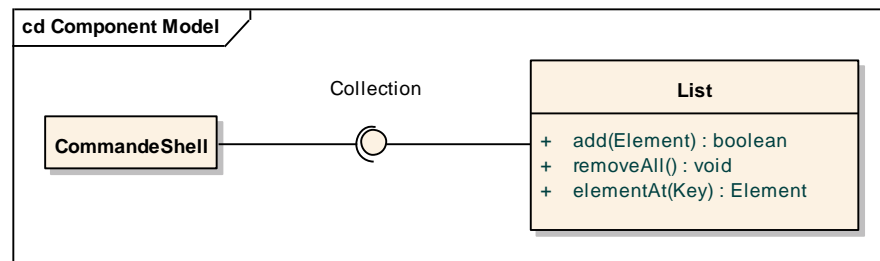
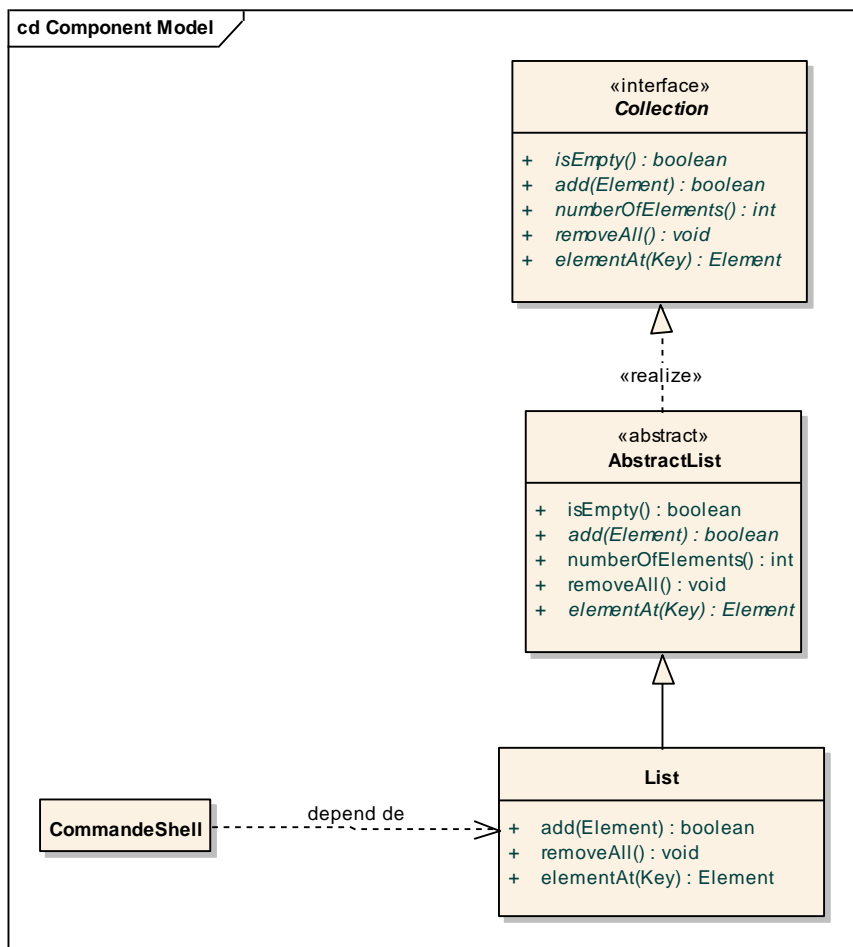
Classe Interface

Une interface est une classe qui n'a pas d'implémentation.

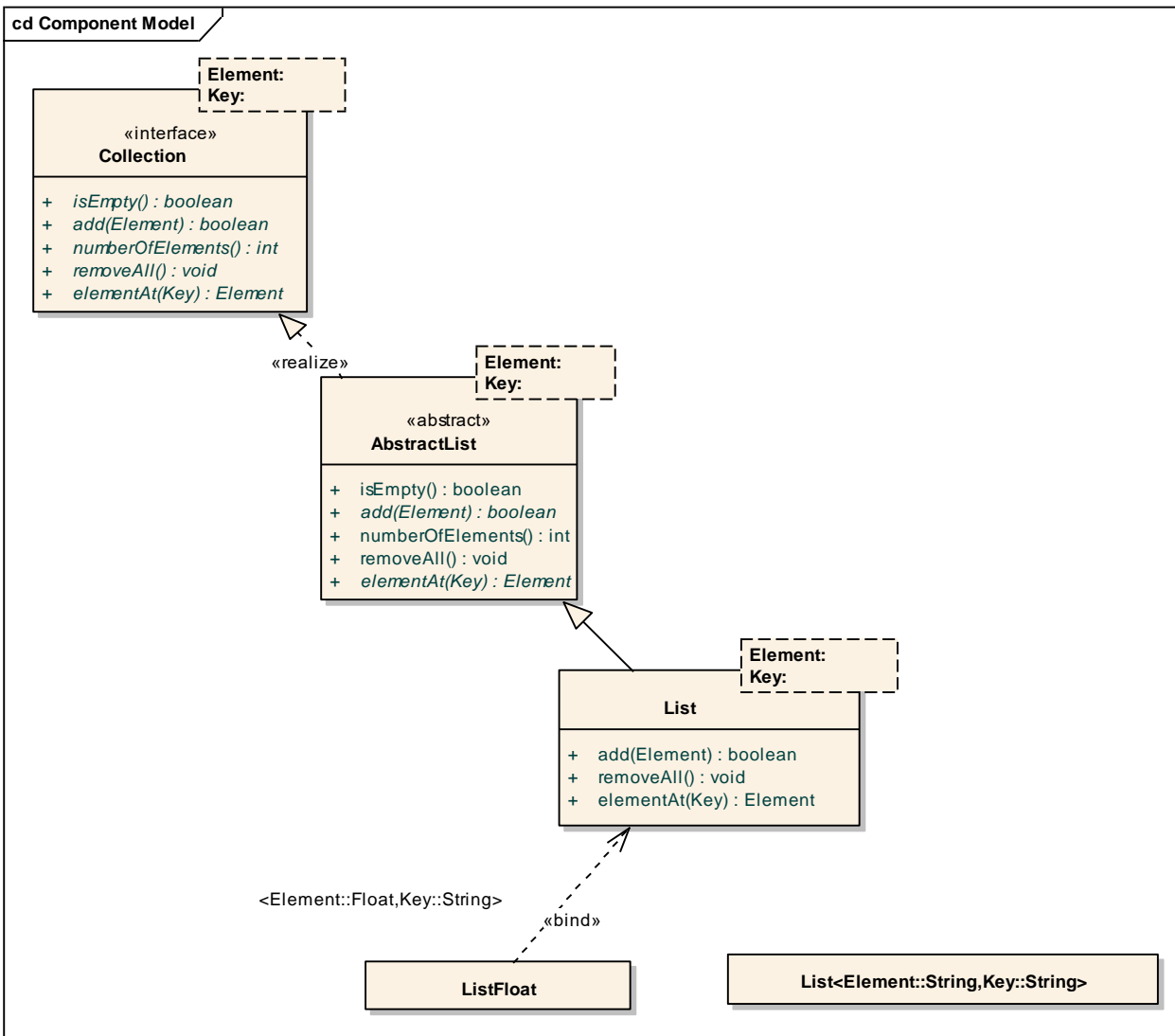
Toutes ses caractéristiques sont abstraites



2.2.2.16a - Interface et classes abstraites

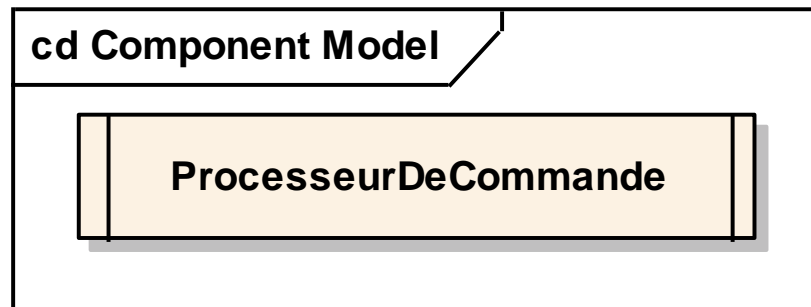


2.2.2.17 - Classes paramétrables



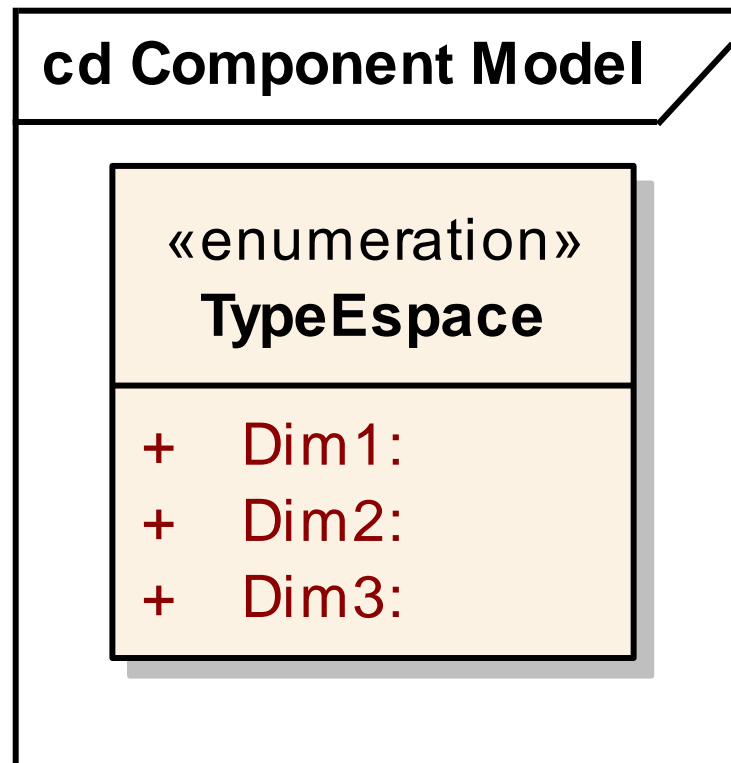


Une classe active est une classe qui a des instance exécutant et contrôlant son propre thread





Les énumérations servent à indiquer un ensemble fixé de valeurs qui n'ont aucune propriété

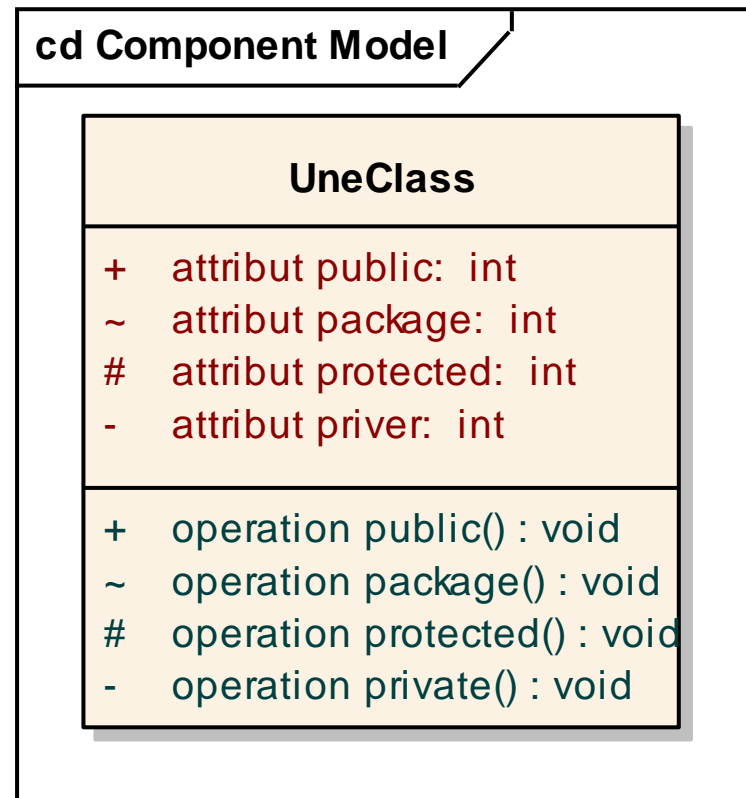




Toute classe a des éléments publics et privés.

UML fournit une représentation de ces différences à travers 4 abréviations :

- privé
- + public
- # protégé
- ~package



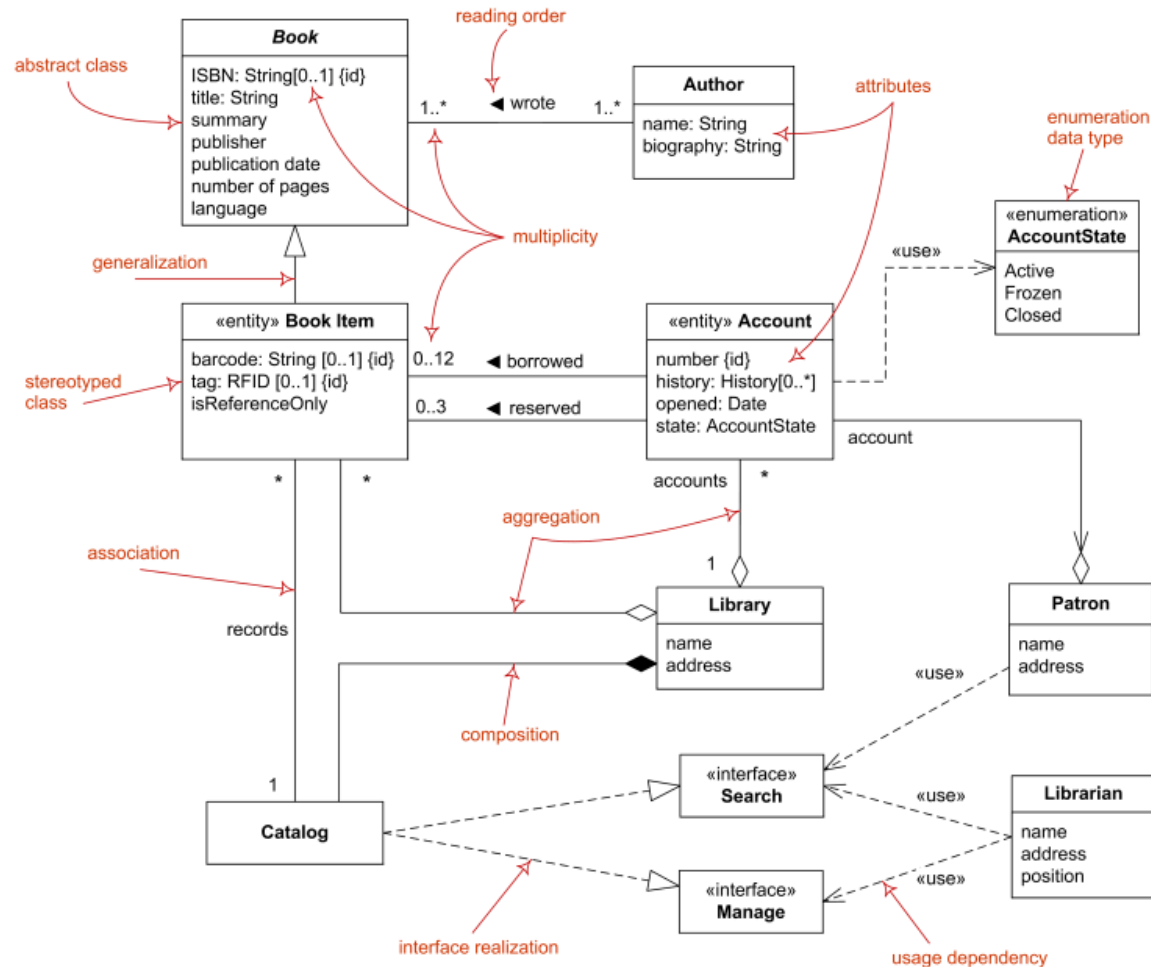


Les diagrammes de classe sont utilisés en permanence

Conseils de mise en œuvre :

- Ne pas essayer d'utiliser tous les concepts,
- commencer par des notions simples : classe, association attribut, généralisation, contraintes
- Ne pas créer systématiquement des diagrammes
- le diagramme de classe conceptuelle peut servir à identifier le langage métier

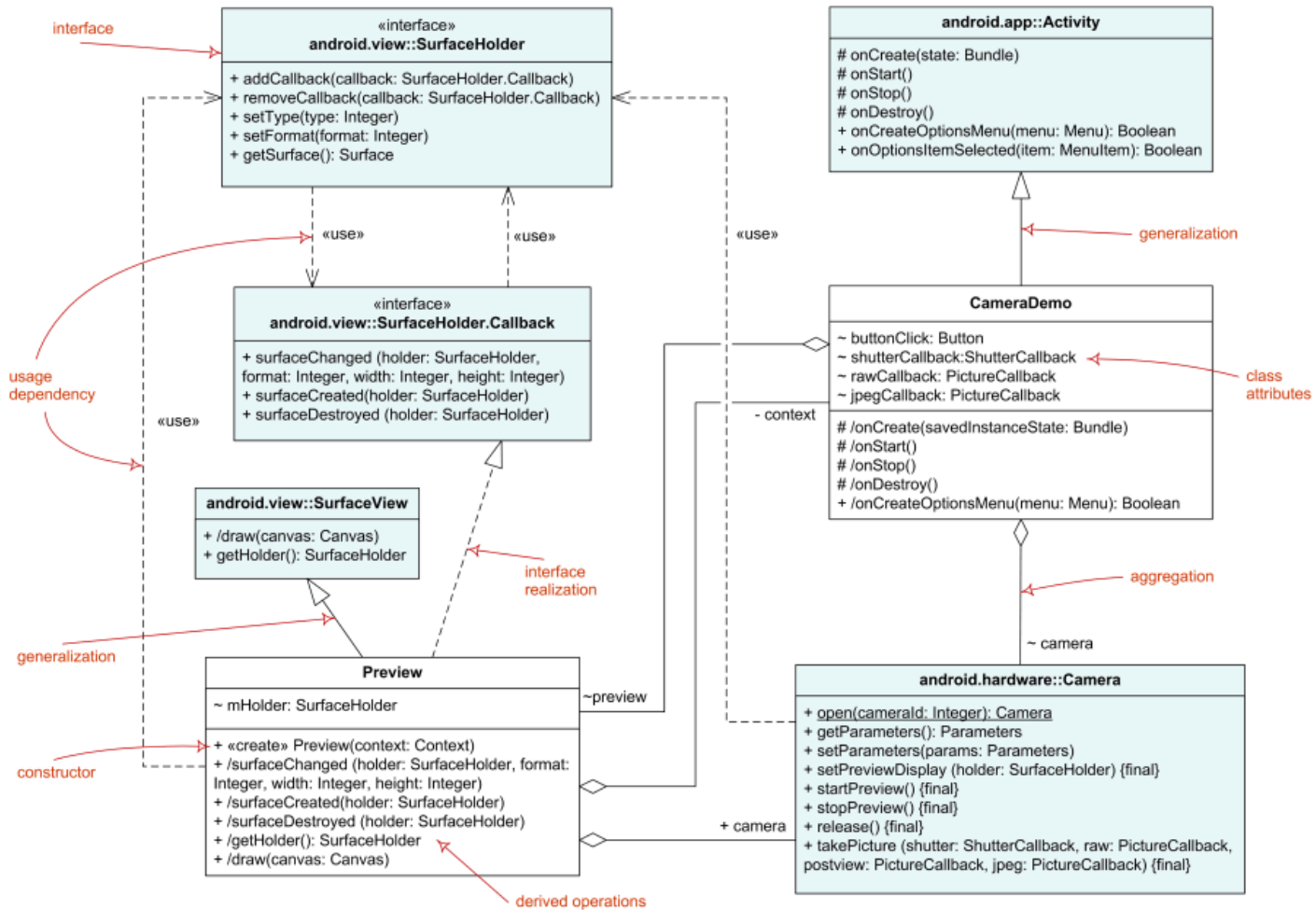
2.1.4.1 – Exemple : Domain Model Diagram



<https://www.uml-diagrams.org/class-diagrams-overview.html>

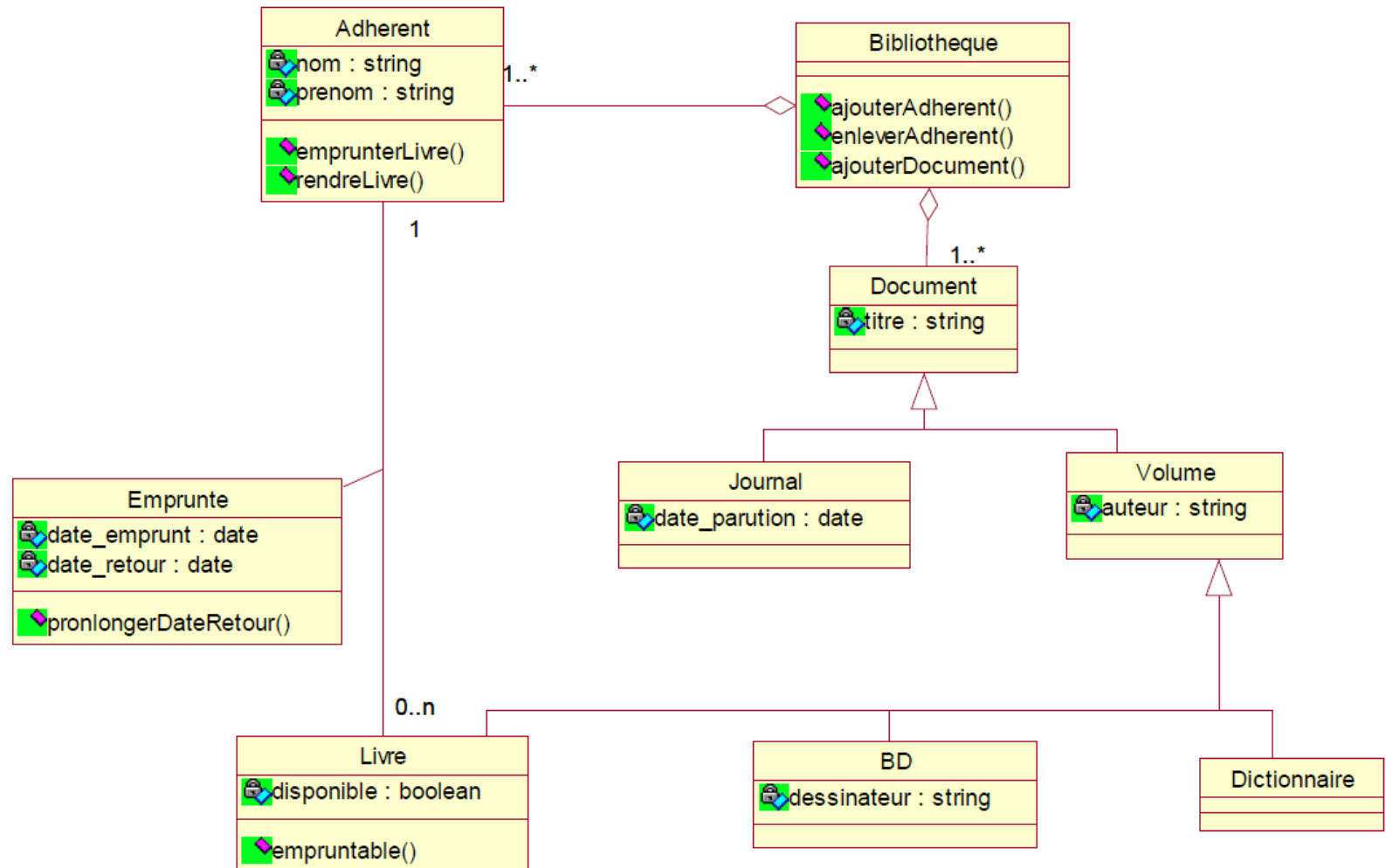


2.1.4.2 – Exemple : Diagram of Implementation Classes

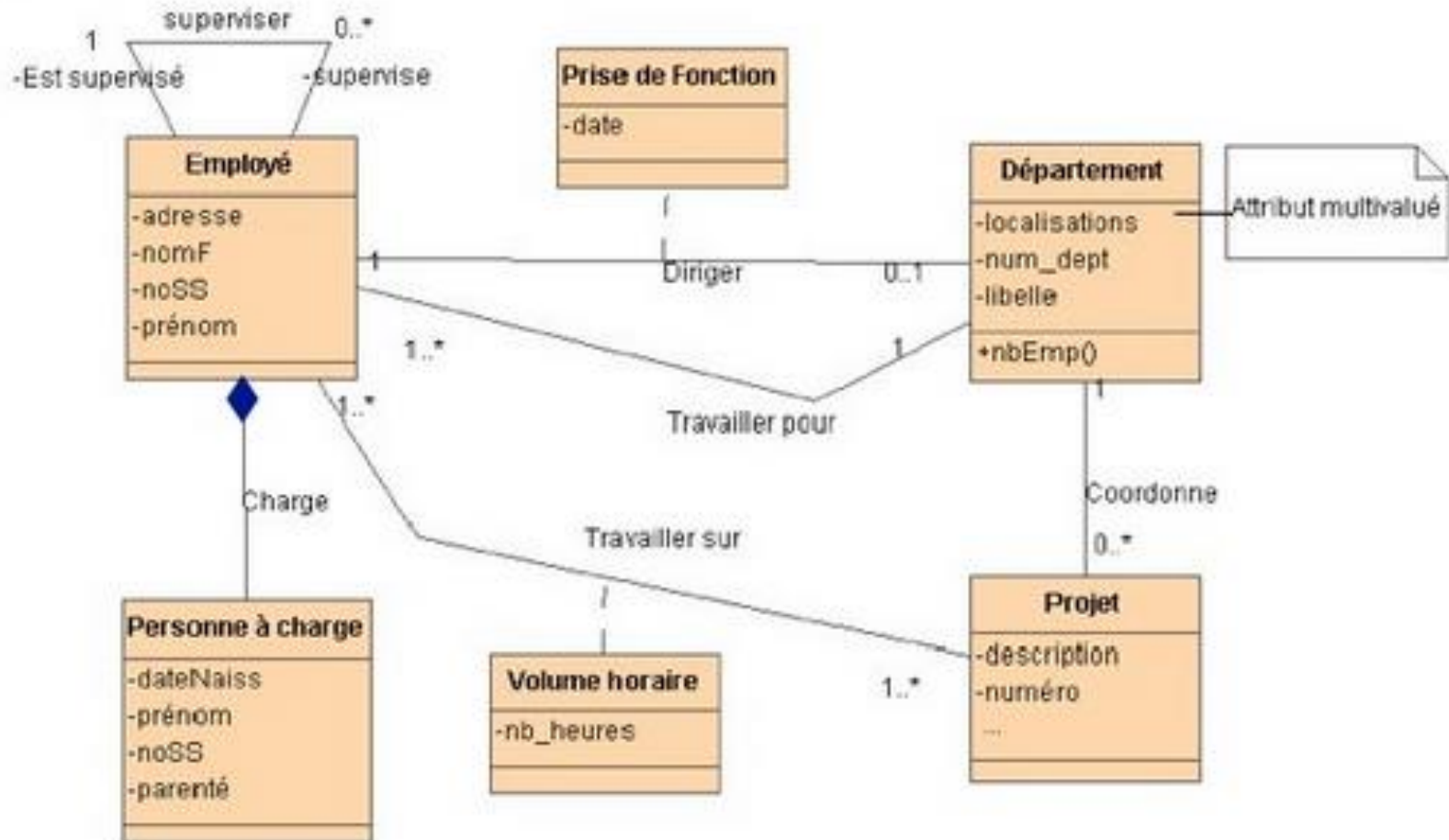


<https://www.uml-diagrams.org/class-diagrams-examples.html>

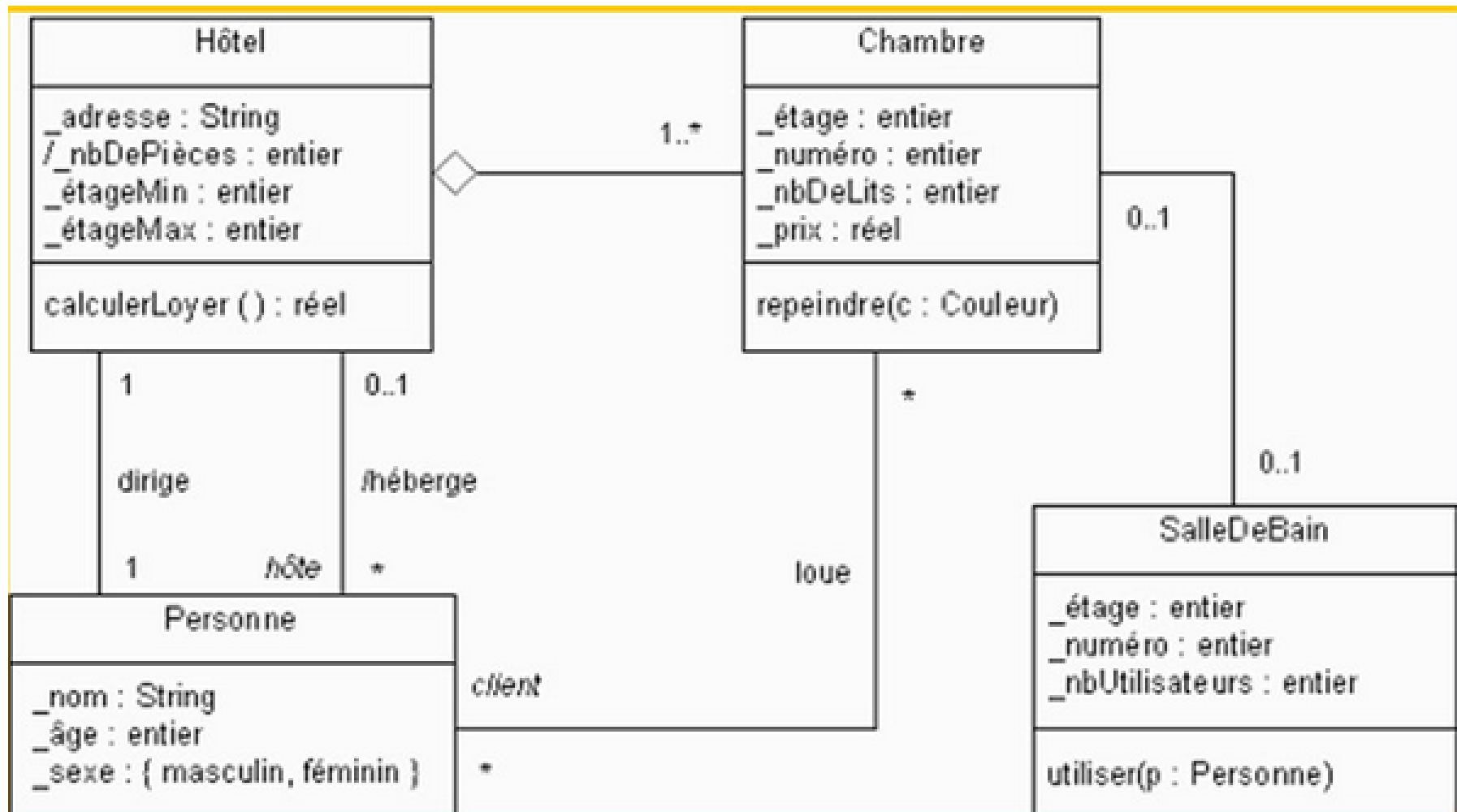
2.2.4.3 – Exemple



2.2.4.4 – Exemple



2.2.4.5 – Exemple

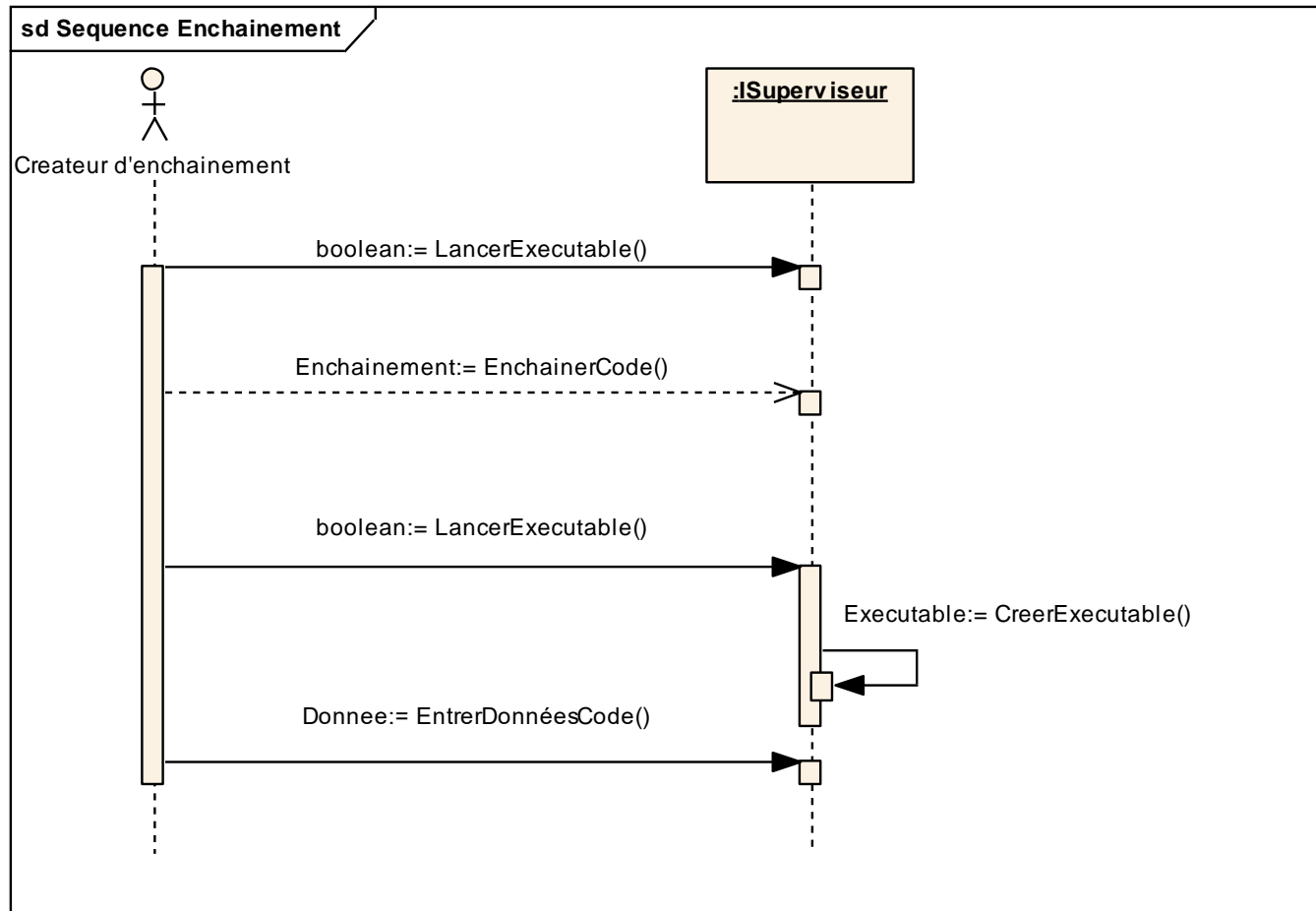


2.2.4.6a – Exemple

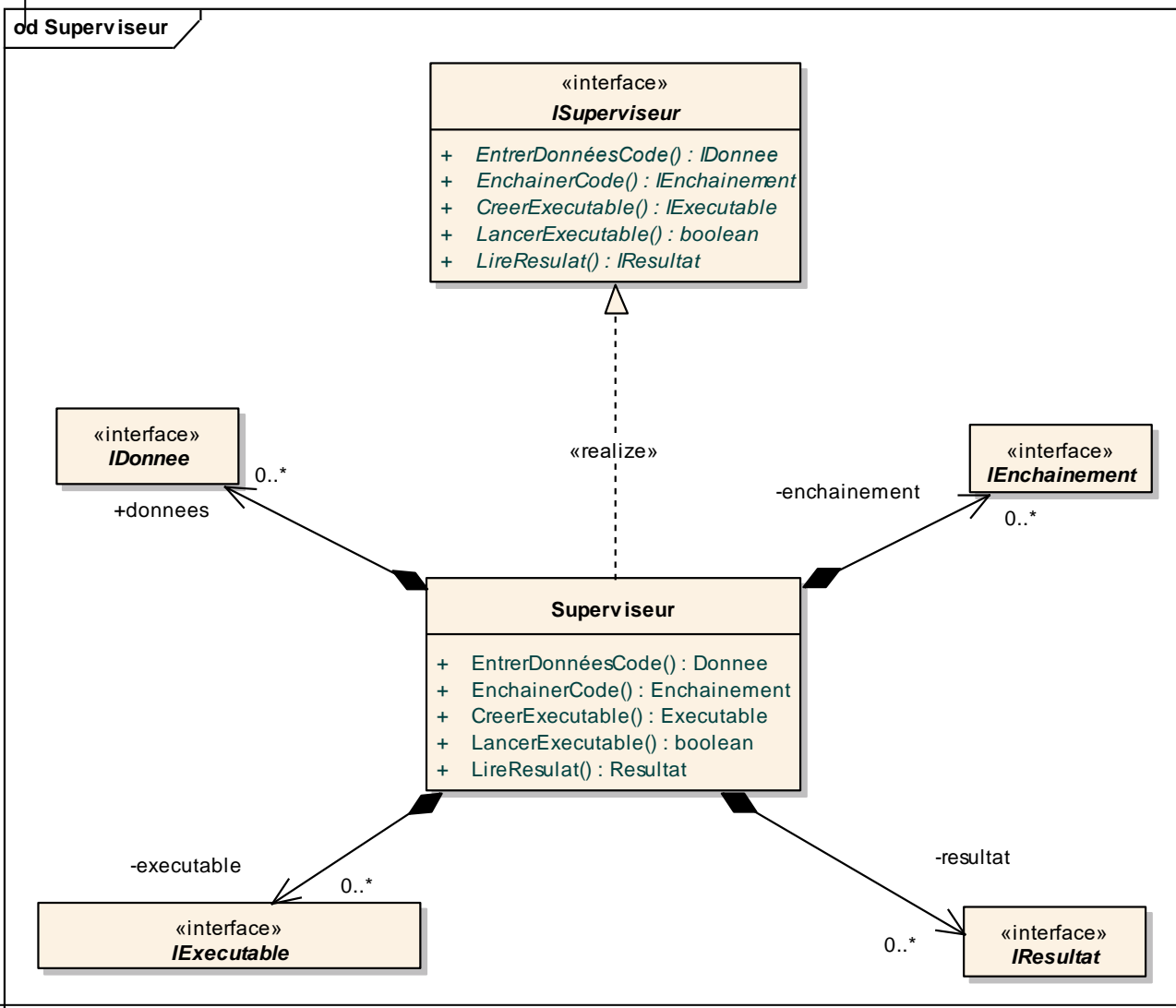
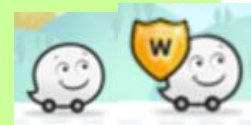


- Exemple : Séquence d'enchaînement
- Exemple : Classe Superviseur
- Exemple : Séquence de lancement d'exécutable
- Exemple : Classe Executable

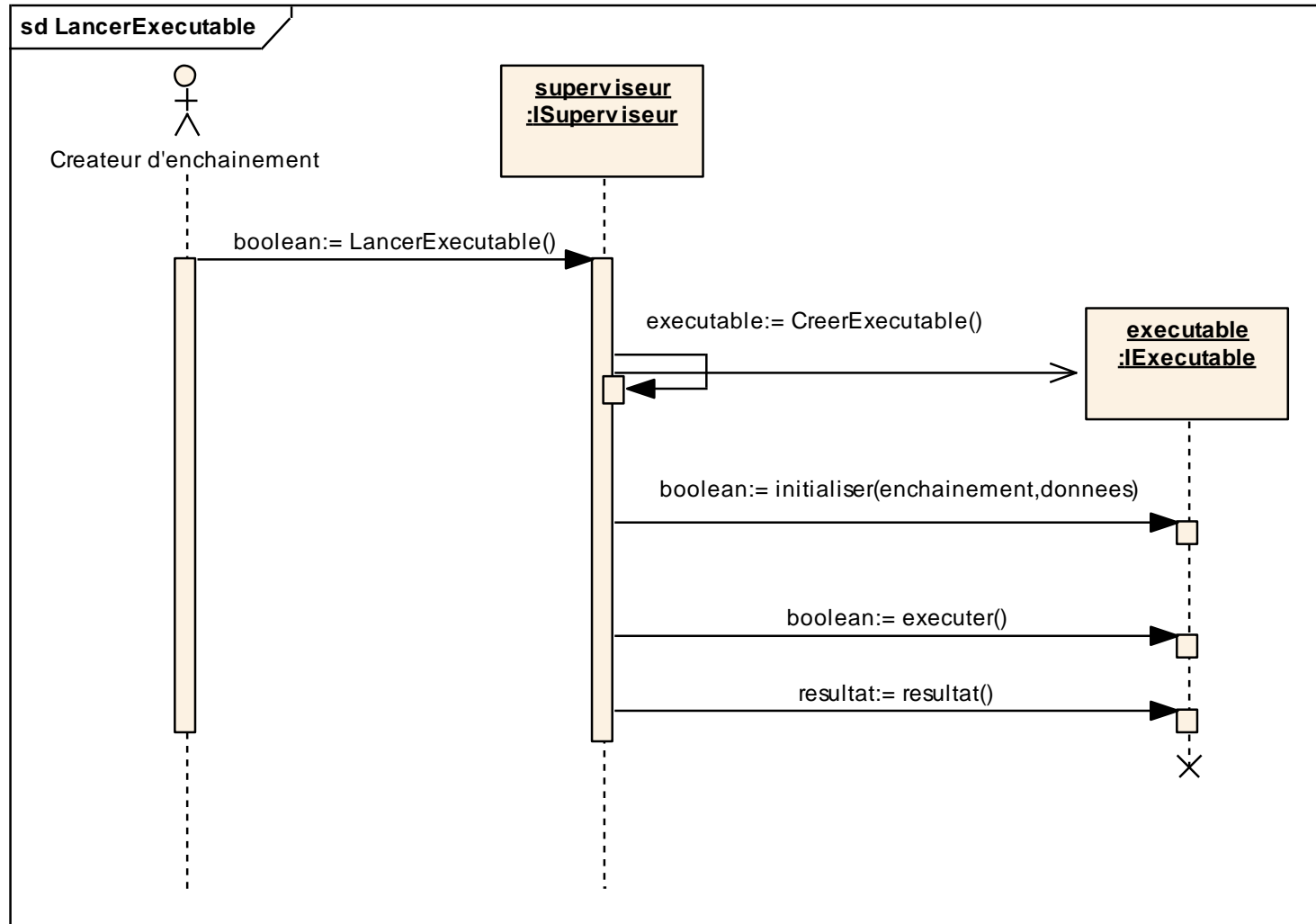
2.2.4.6b – Exemple : Séquence d'enchaînement



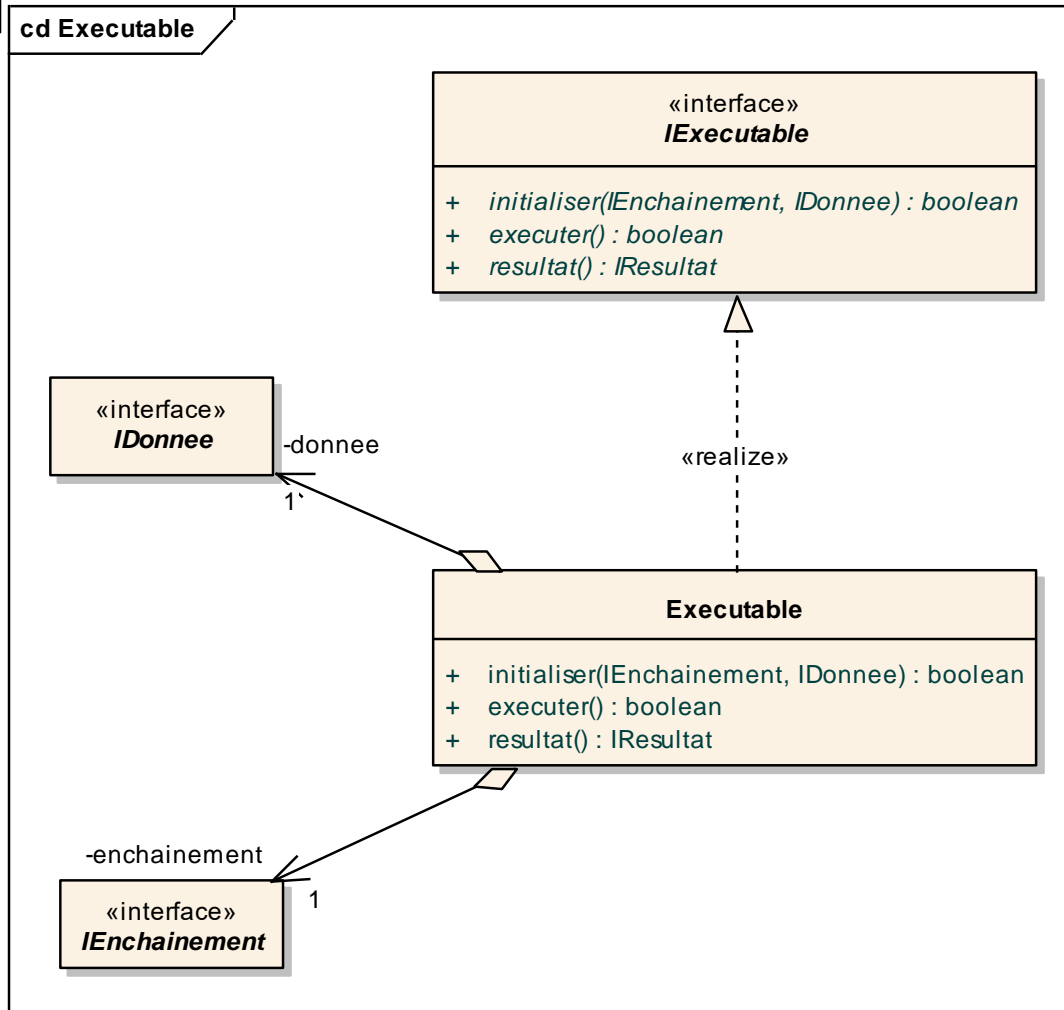
2.2.4.6c – Exemple : Classe Superviseur



2.2.4.6d – Exemple : Séquence de lancement d'exécutable



2.2.4.6e – Exemple : Classe Executable



2.3 - Le diagramme d'objet



- Fonctionnalité
- Exemple



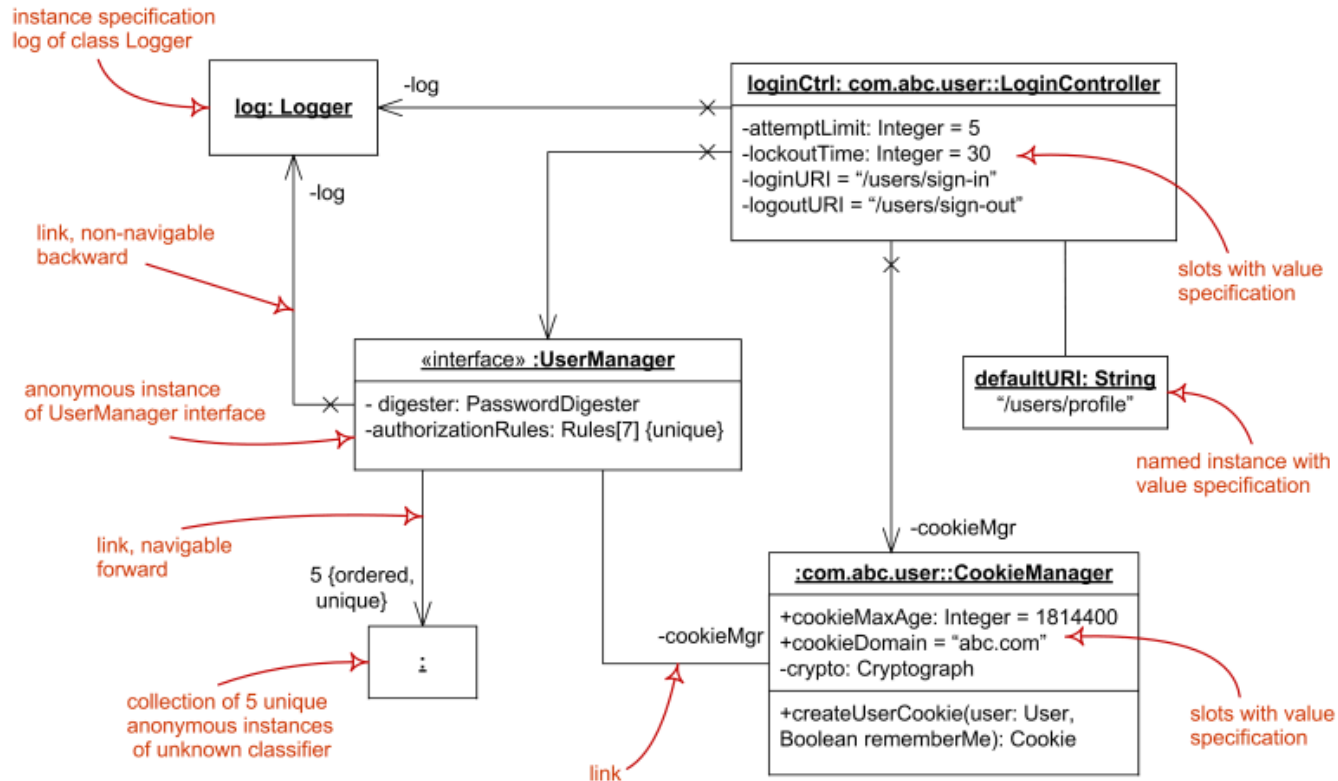
Le diagramme d'objet (d'instance) est un instantané des objets du système à un moment donné.

Il permet de :

- Montrer des **objets** (instances de classes dans un état particulier) et des **liens** (relations sémantiques) entre ces objets.
- Montrer un **contexte** (avant ou après une interaction entre objets par exemple).
- Explorer des solutions dans les phases exploratoires du projet

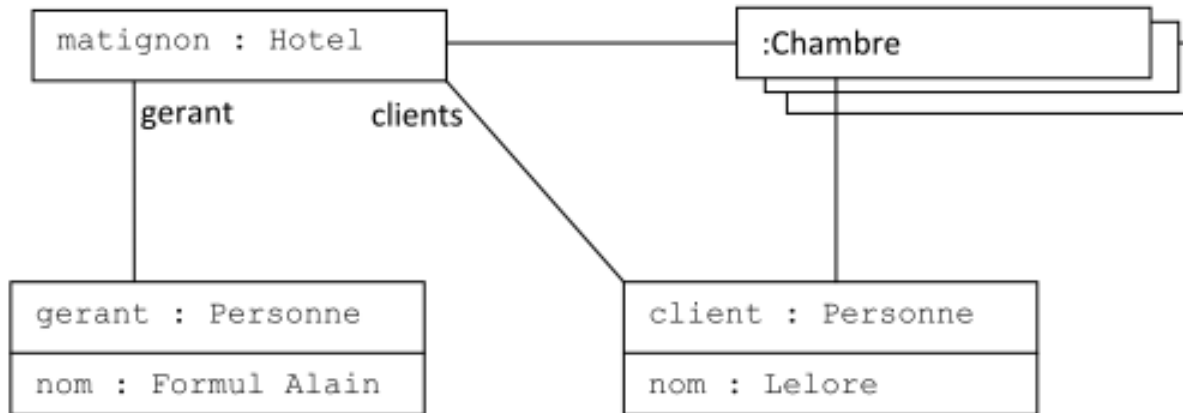
Le diagramme d'objet permet d'illustrer un diagramme de classe par des exemples.

2.3.2.1 – Exemple : Object Diagram

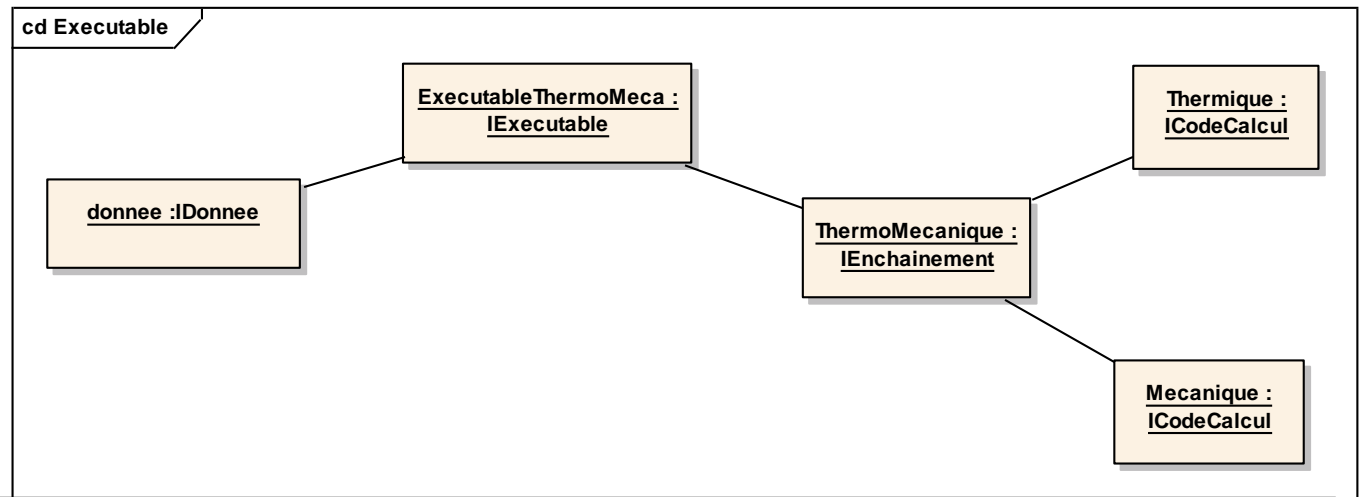
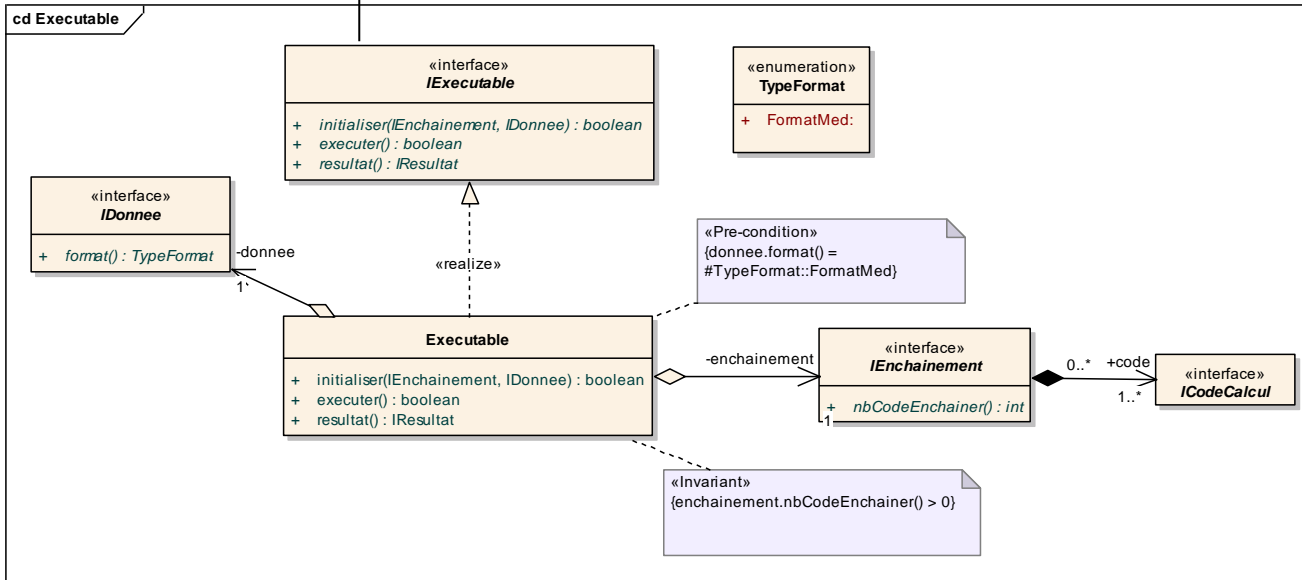


<https://www.uml-diagrams.org/class-diagrams-overview.html>

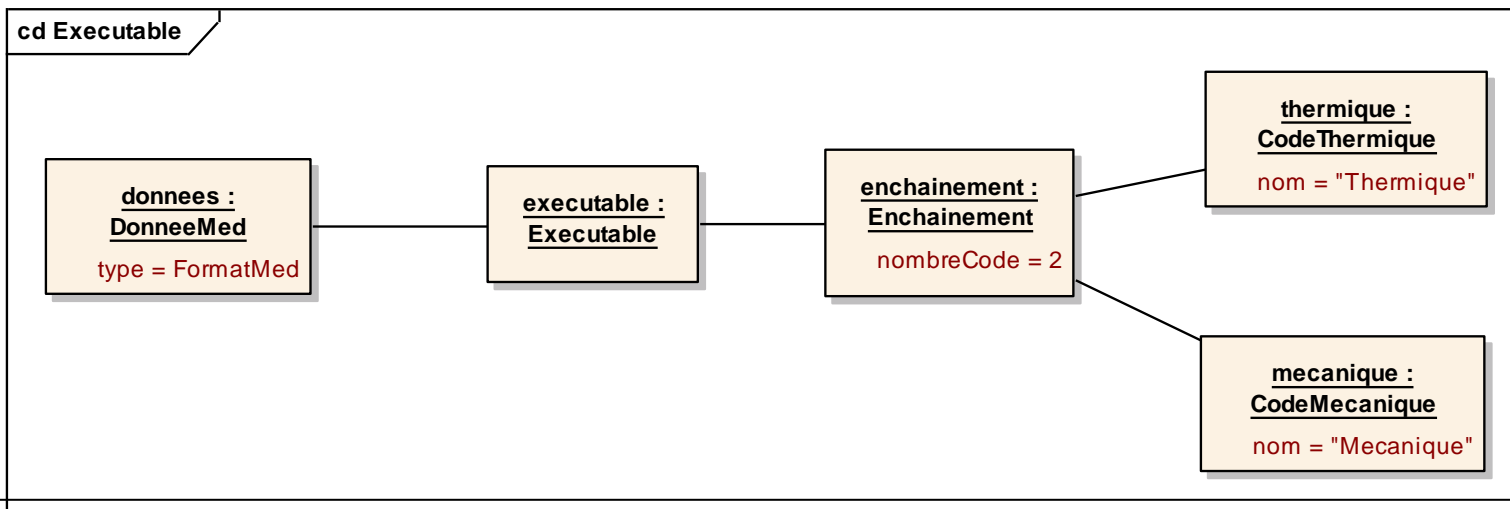
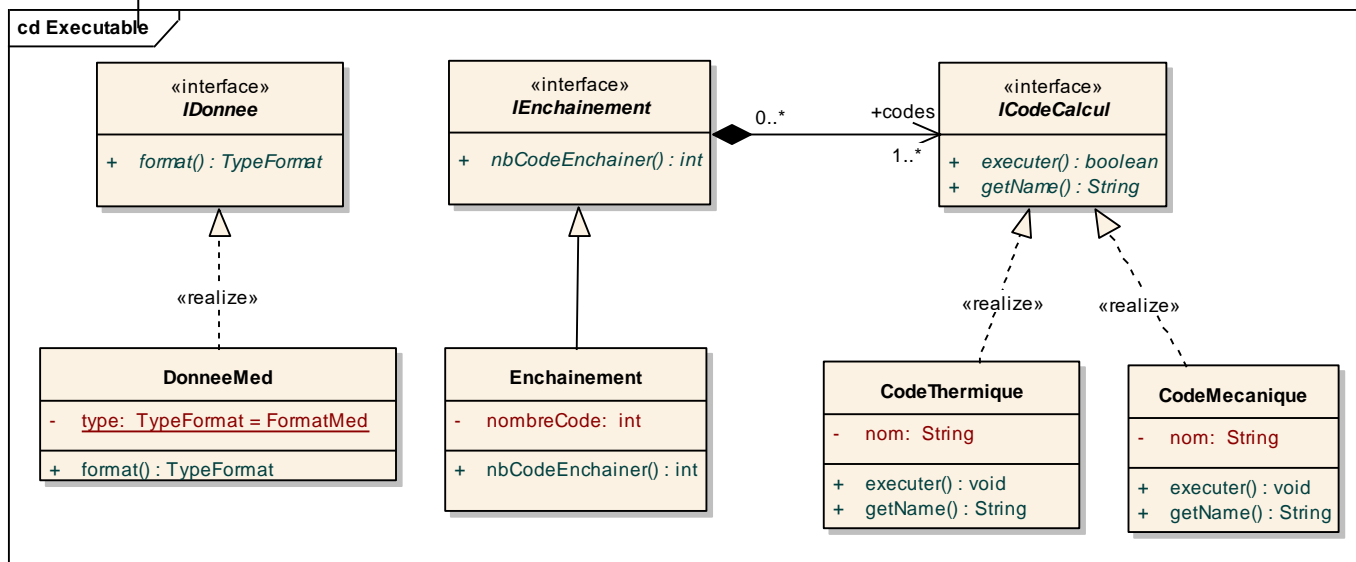
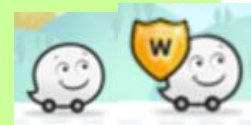
2.3.2.2 – Exemple



2.3.2.3 – Exemple : Exécutable (interface)



2.3.2.4 – Exemple : Exécutable



2.4 - Le diagramme de package



- Fonctionnalité
- Notation
- Mise en œuvre
- Exemple



Les **paquetages** sont des **éléments d'organisation** des modèles et servent de "briques" de base dans la construction d'une architecture.

Ils permettent de :

- Regrouper des éléments de n'importe quelle construction UML, selon des critères purement logiques
- Encapsuler des éléments de modélisation
- Structurer un système en catégories et sous-systèmes
- Représenter des espaces de nommage
- Voir ce qui est réutilisable

Un package en analyse = une catégorie

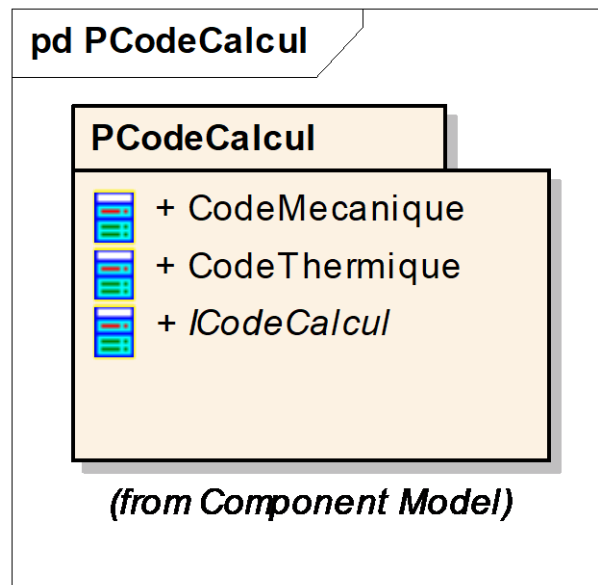
Un package en conception d'architecture = un sous-système



- Espaces de nommage
- Dépendance entre packages
- Généralisation



Tout élément contenu dans un package se distingue par son appartenance au package englobant



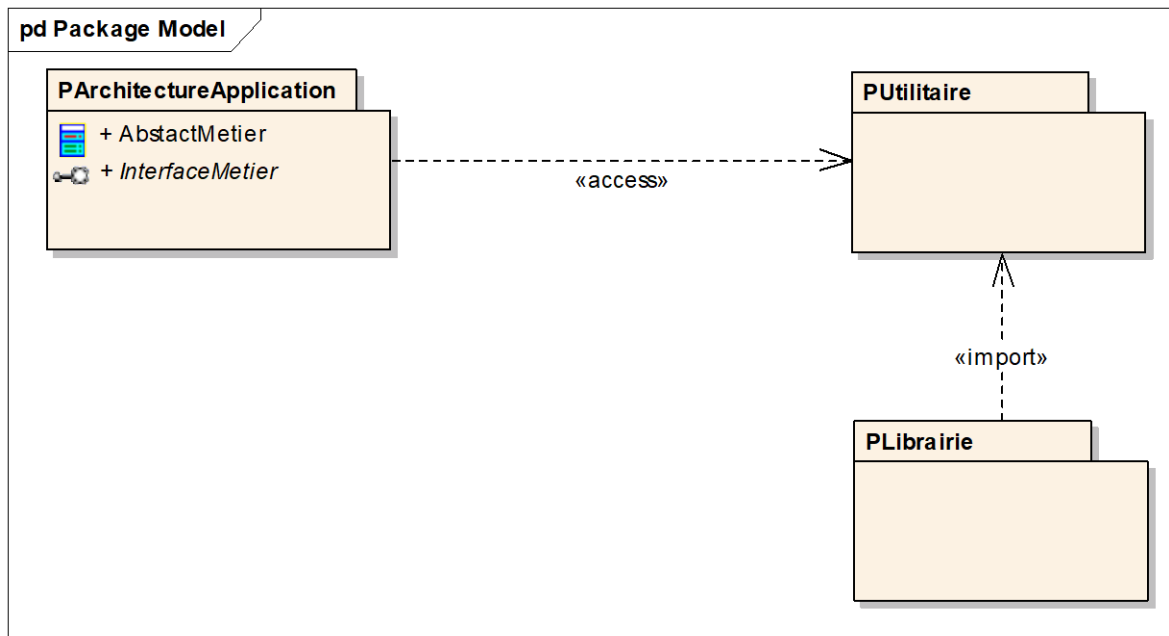
2.4.2.2 - Dépendance entre package



Les dépendances entre package représentent des relations entre packages et non entre éléments individuels

Séréotypes standards associés :

- « importe » : ajoute les éléments du package au package source
- « accede » : permet d'accéder à des éléments du package

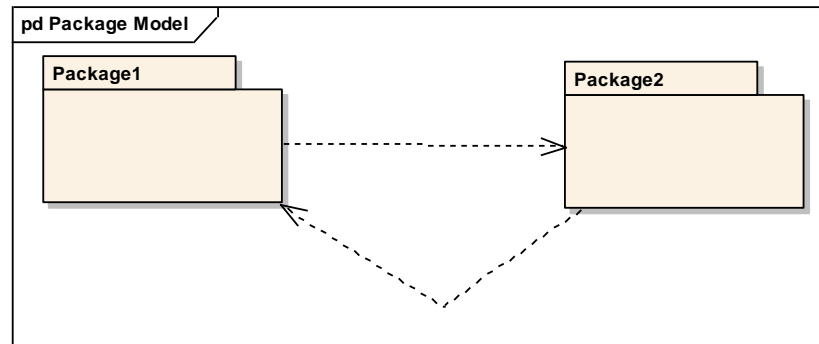


2.4.2.2 - Dépendance entre package (suite)

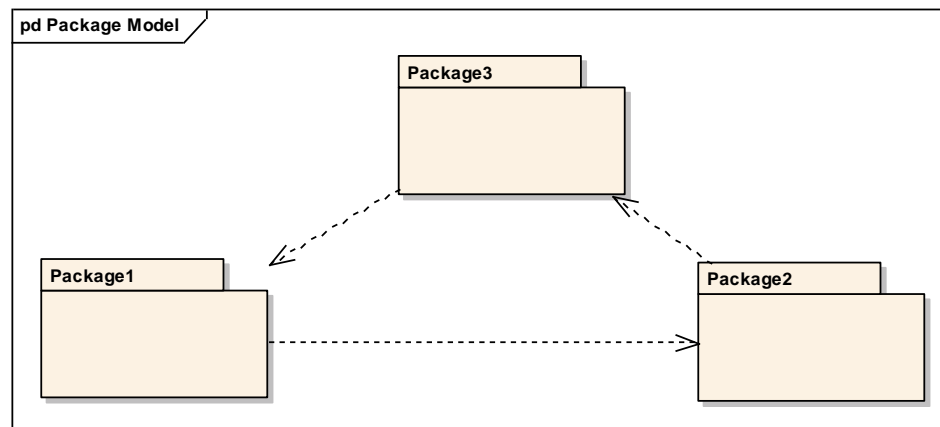


A éviter :

- Les dépendances circulaires

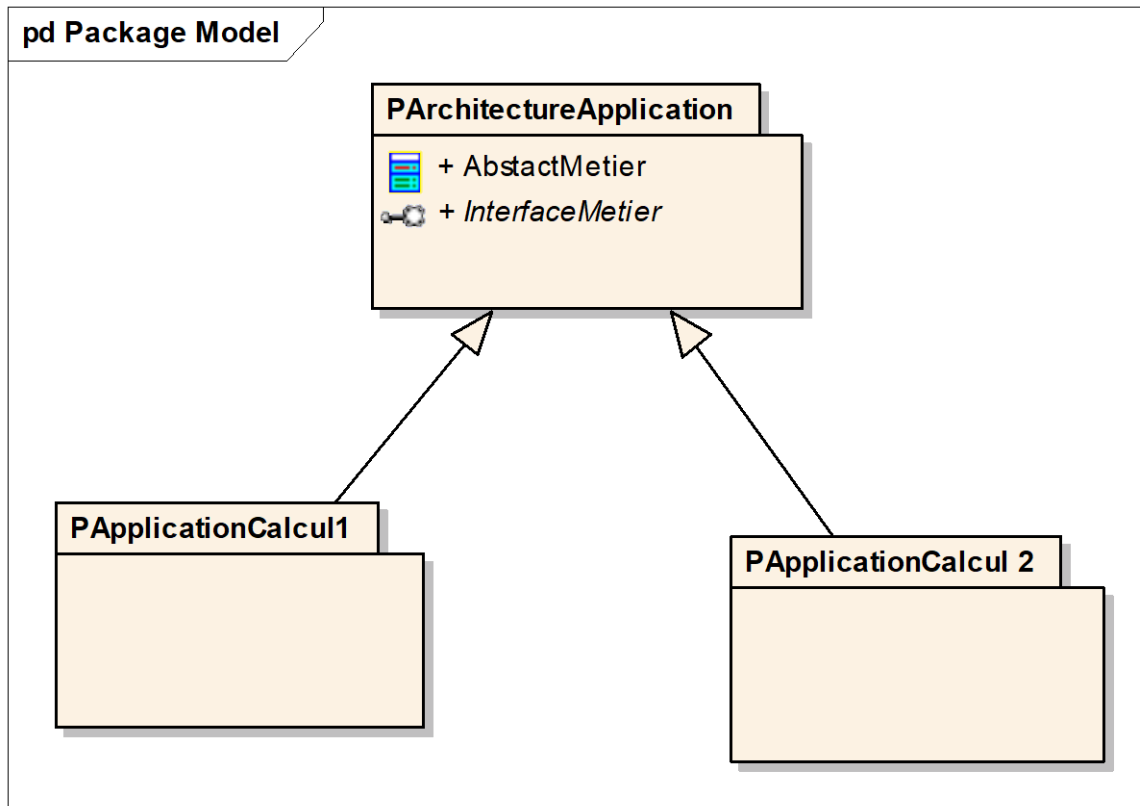


- Les dépendances circulaires transitives





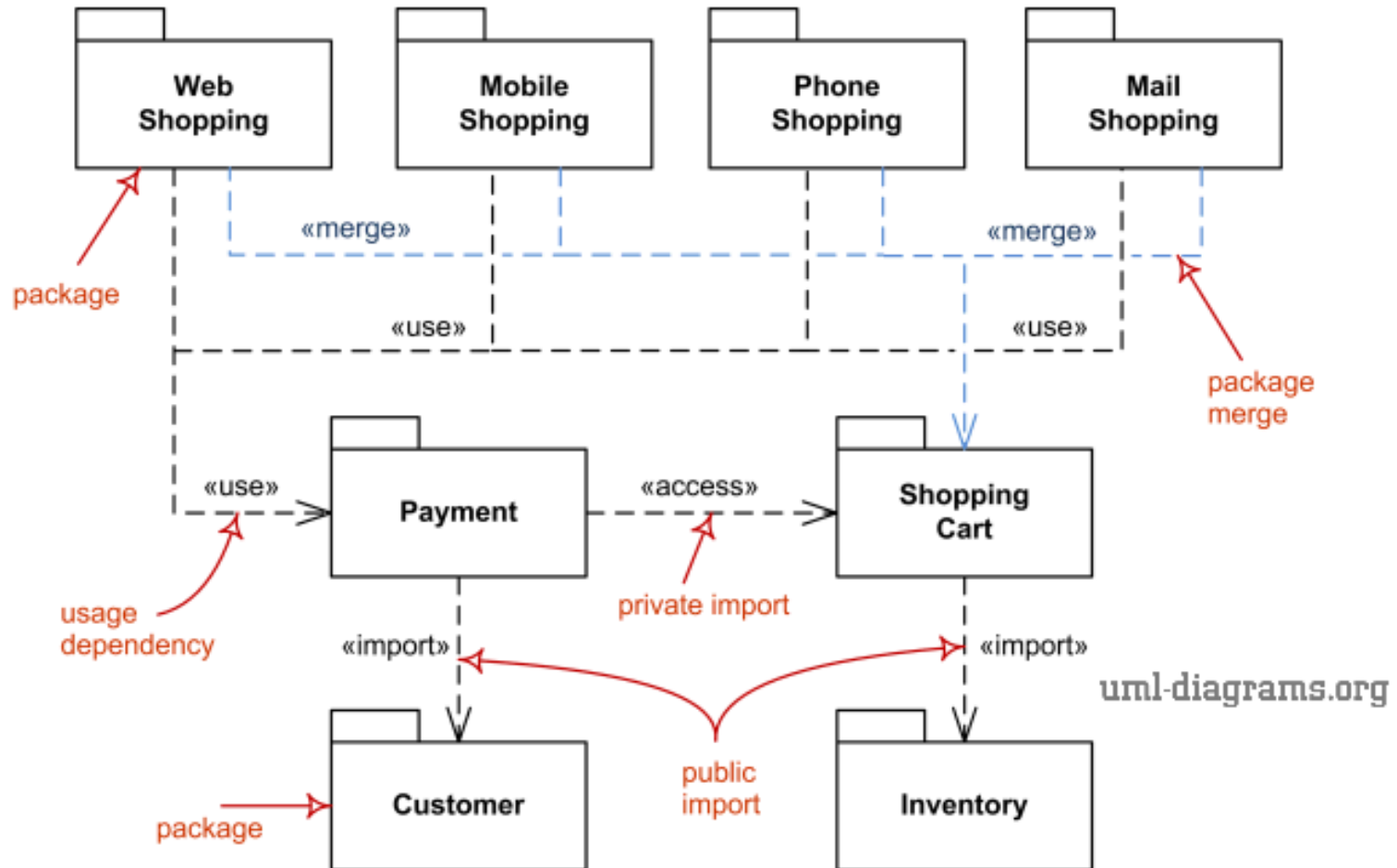
La généralisation des packages est équivalente à la généralisation des classes





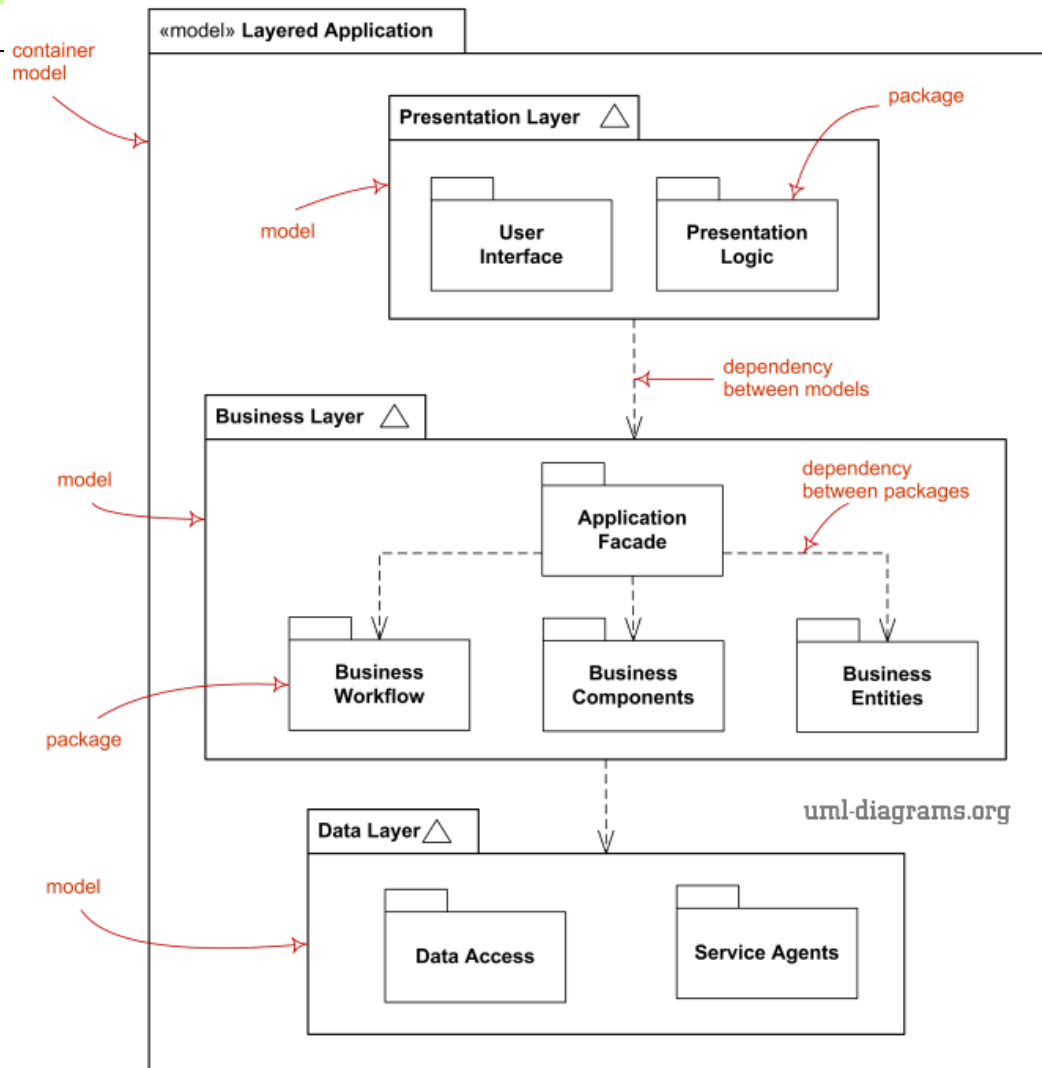
A utiliser pour regrouper les entités de modélisation afin d'avoir une image des dépendances, de les contrôler et les réduire

2.4.4.1 – Exemple : Package Diagram



<https://www.uml-diagrams.org/package-diagrams-overview.html>

2.4.4.2 – Exemple : Model Diagram



<https://www.uml-diagrams.org/package-diagrams-overview.html>

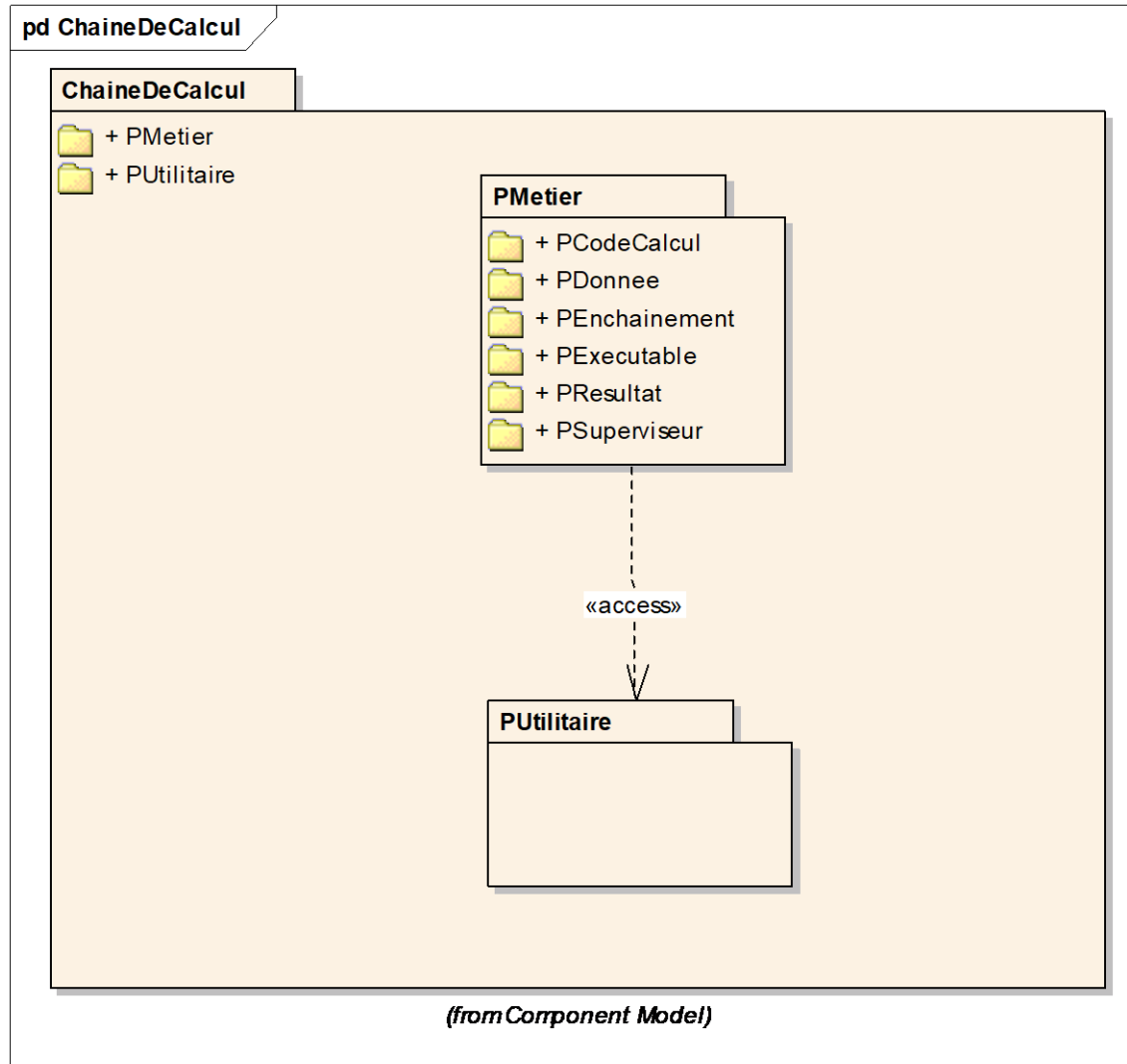
<https://www.uml-diagrams.org/package-diagrams-examples.html>

2.4.4.3 - Exemple

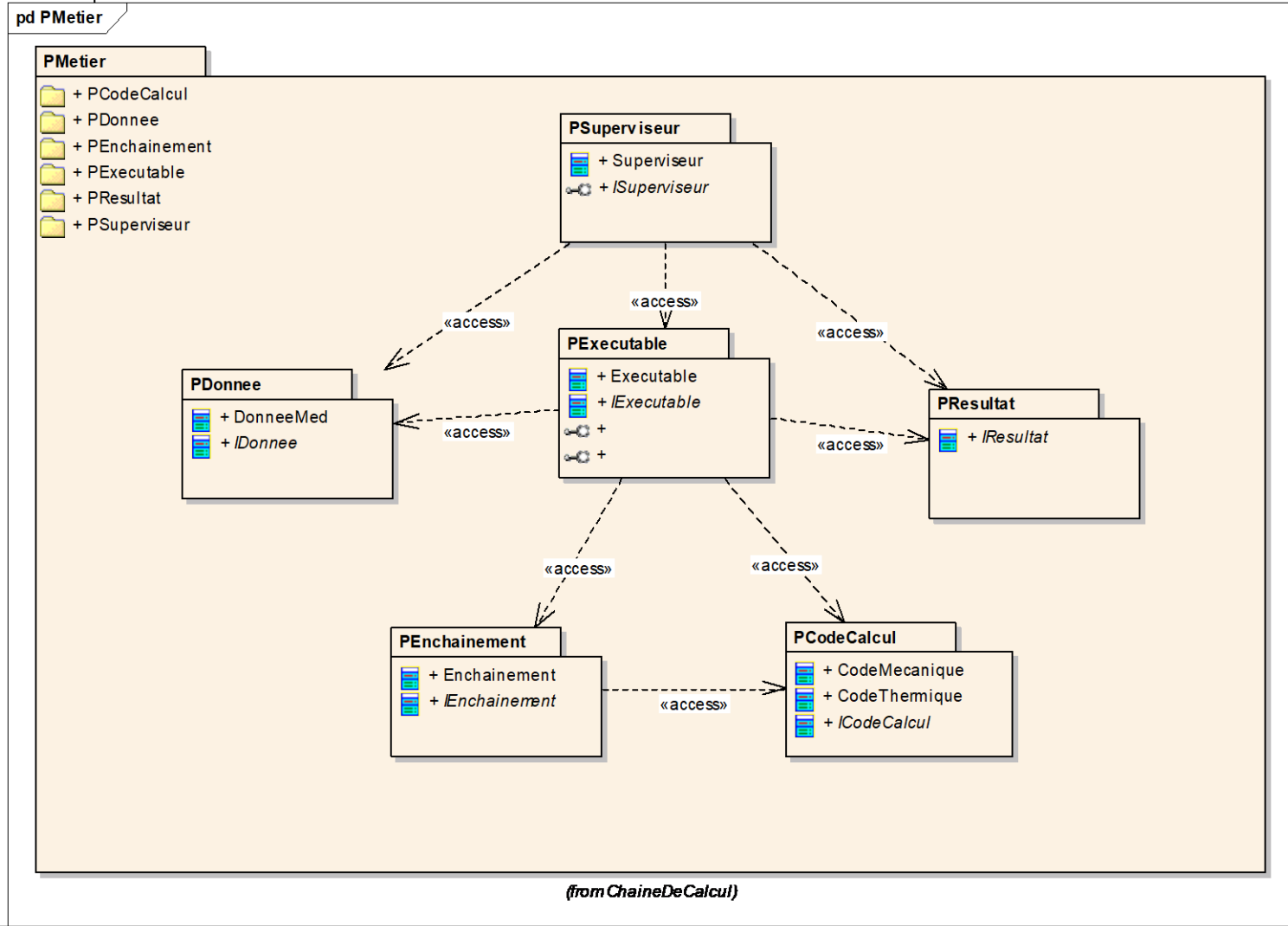


- Exemple : Package chaîne de calcul
- Exemple : Package métier

2.4.4.3a – Exemple : Package Chaîne de calcul



2.4.4.3b – Exemple : Package métier





Yantra Technologies



2.5 - Le diagramme de séquence



- Fonctionnalité
- Notation
- Mise en œuvre
- Exemple



Le diagramme de séquence est une vue dynamique qui permet de :

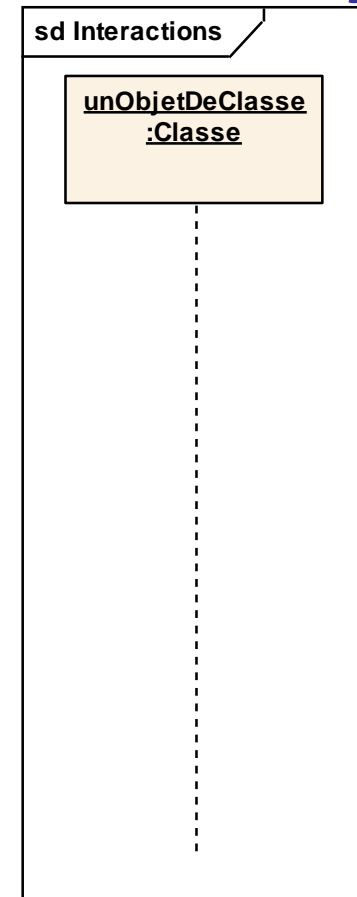
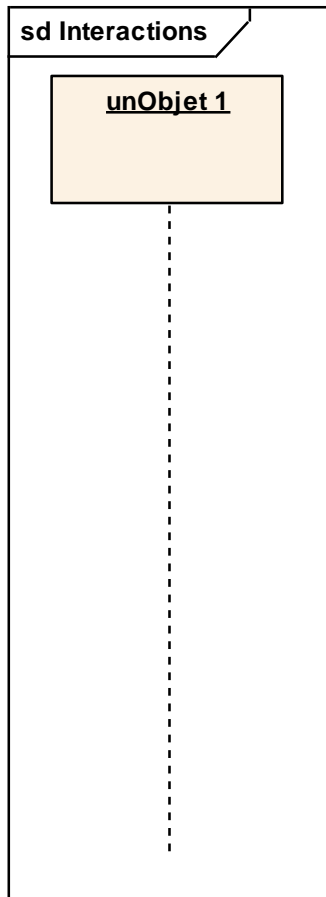
- Décrire la succession chronologique d'opérations réalisées par un acteur
- Décrire le contexte ou l'état des objets
- Décrire les différents scénarios d'un cas d'utilisation
- Exposer le comportement complexe d'une interaction (classe ou processus)
- Mettre en avant les contraintes temporelles
- Indiquer les objets que l'acteur va manipuler, et les opérations qui font passer d'un objet à un autre



- Interaction
- Activations et envois de message
- Contrainte temporelle
- Ligne de vie des objets
- Cadre d'interaction
- Opérateurs courants des cadres d'interaction
- Appel synchrone et asynchrone

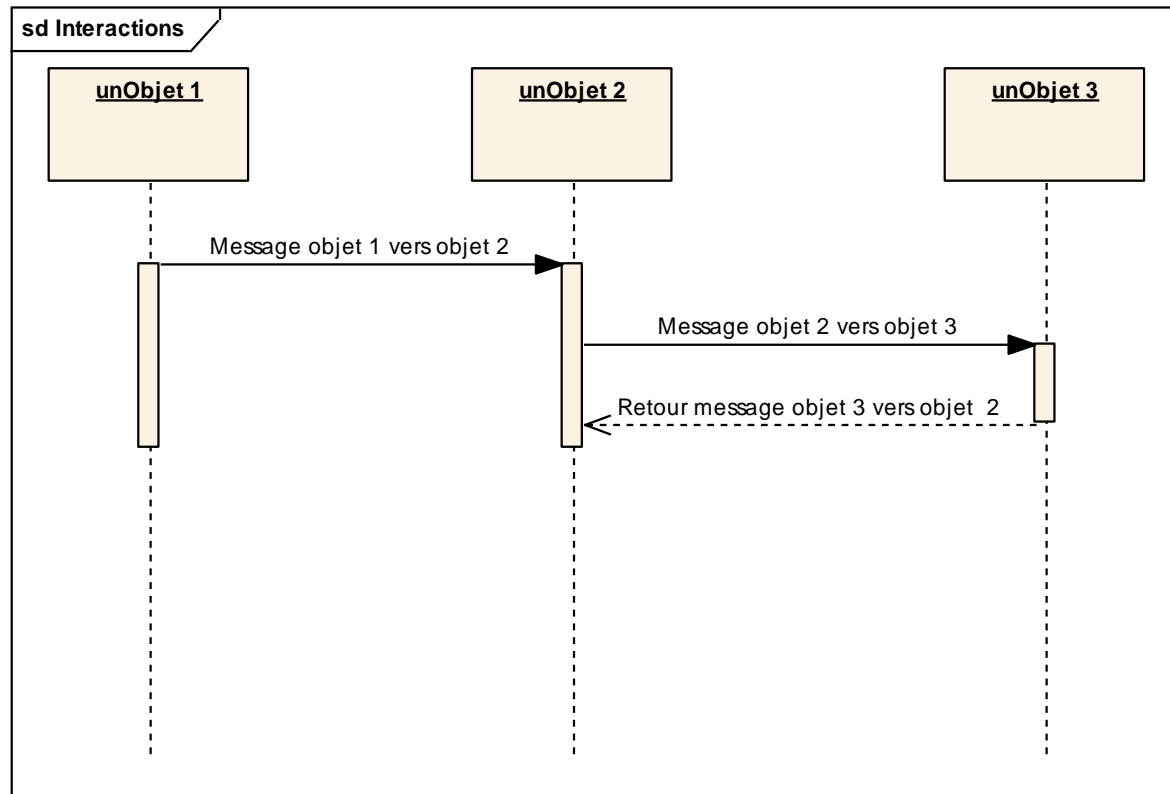


Représentation graphique d'un Objet

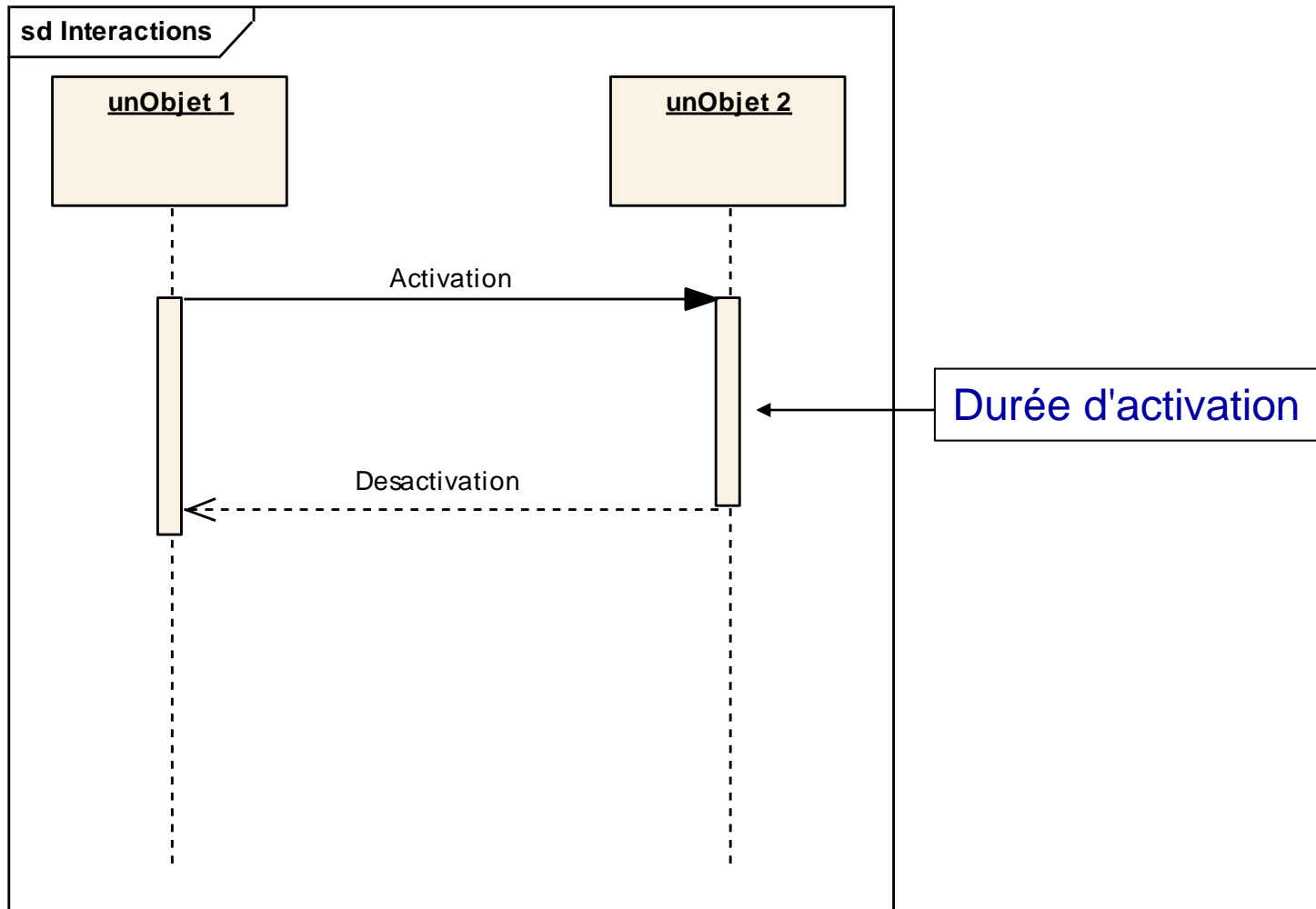




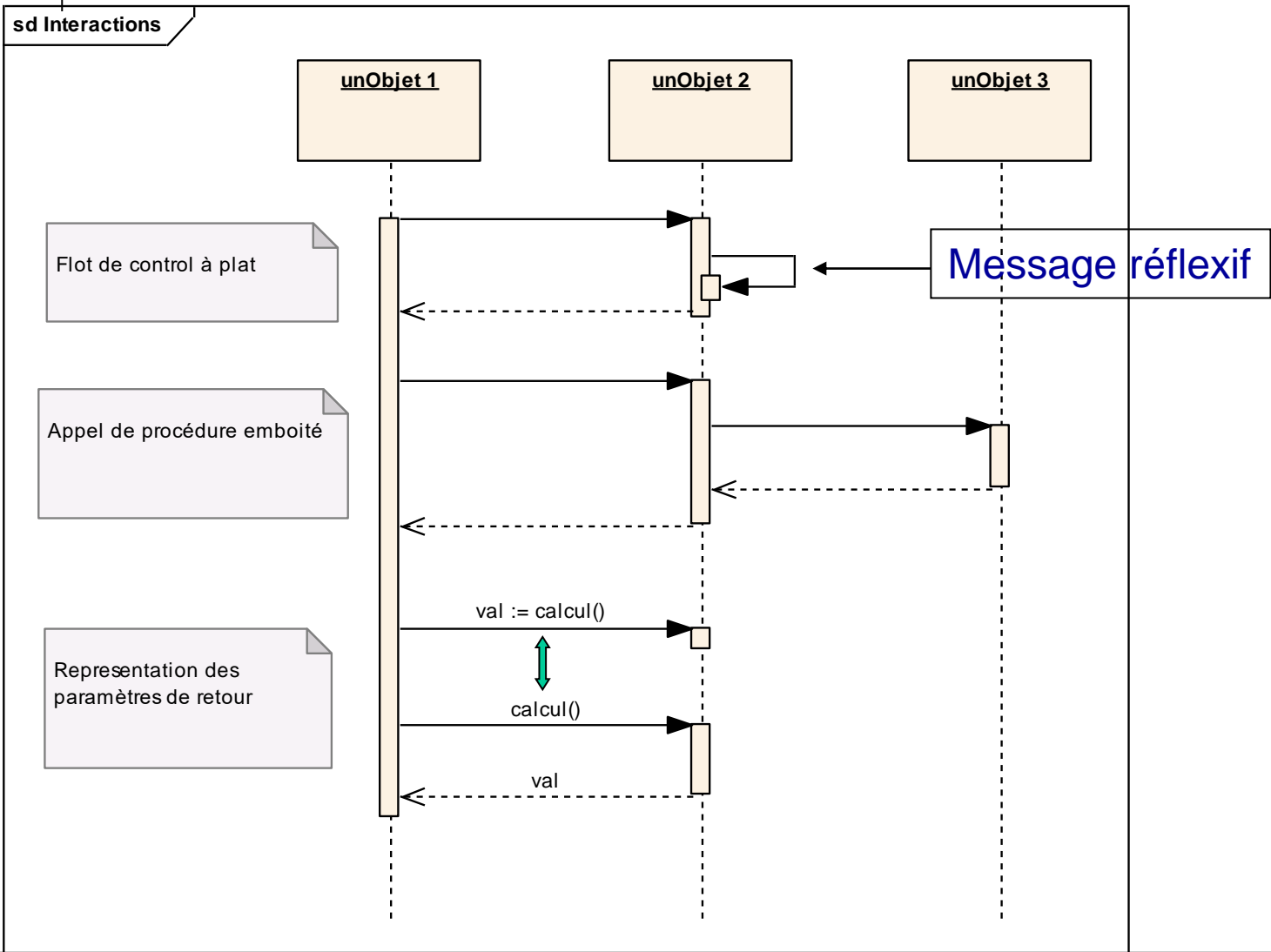
Exemple de diagramme de séquence



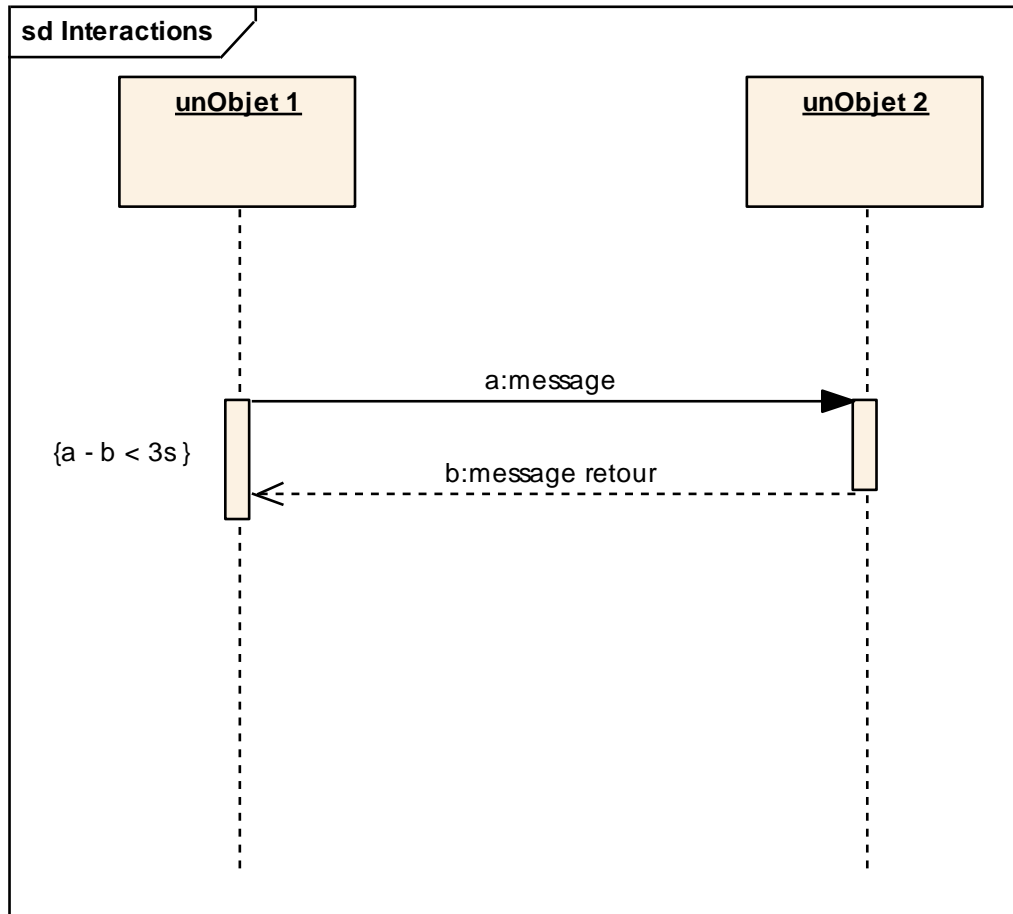
2.5.2.2 - Activations et envois de message



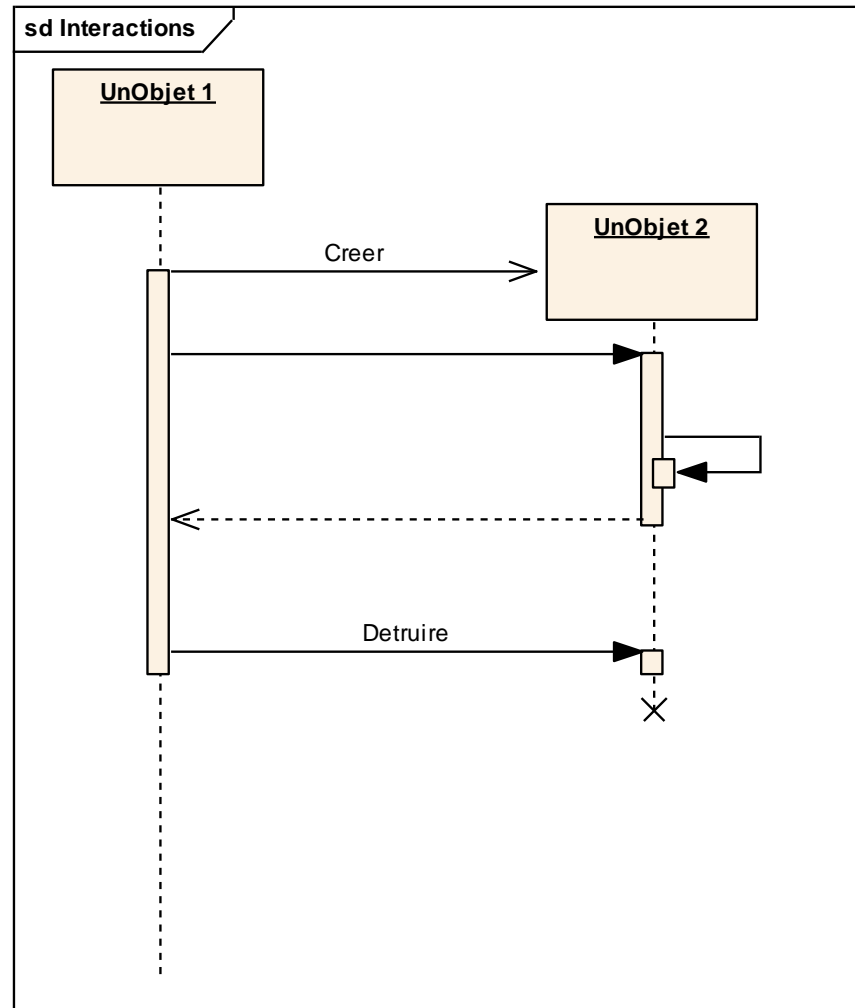
2.5.2.2a - Activations et envois de message



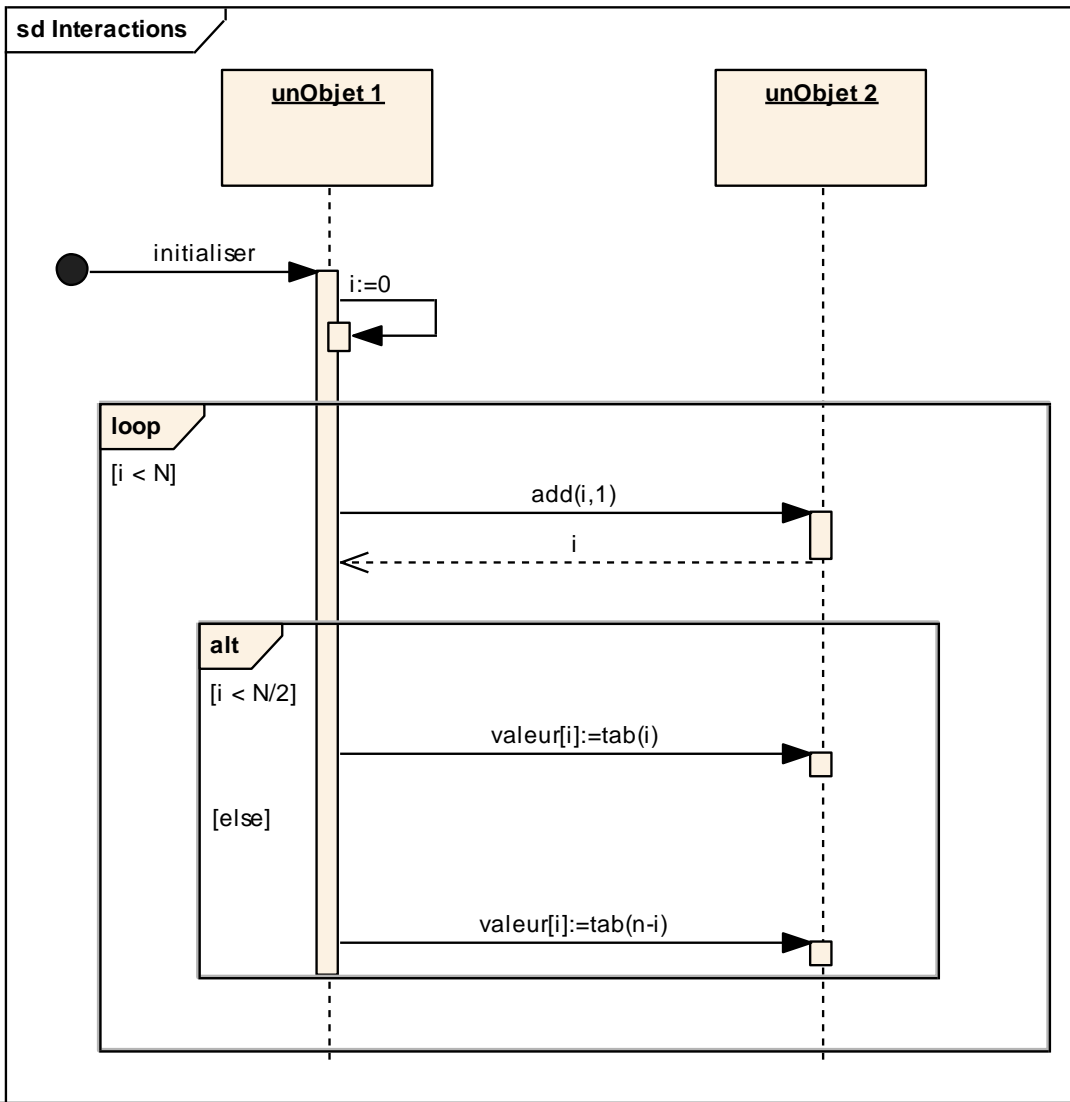
2.5.2.3 - Contrainte temporelle



2.5.2.4 - Ligne de vie des objets



2.5.2.5 - Cadre d'interaction

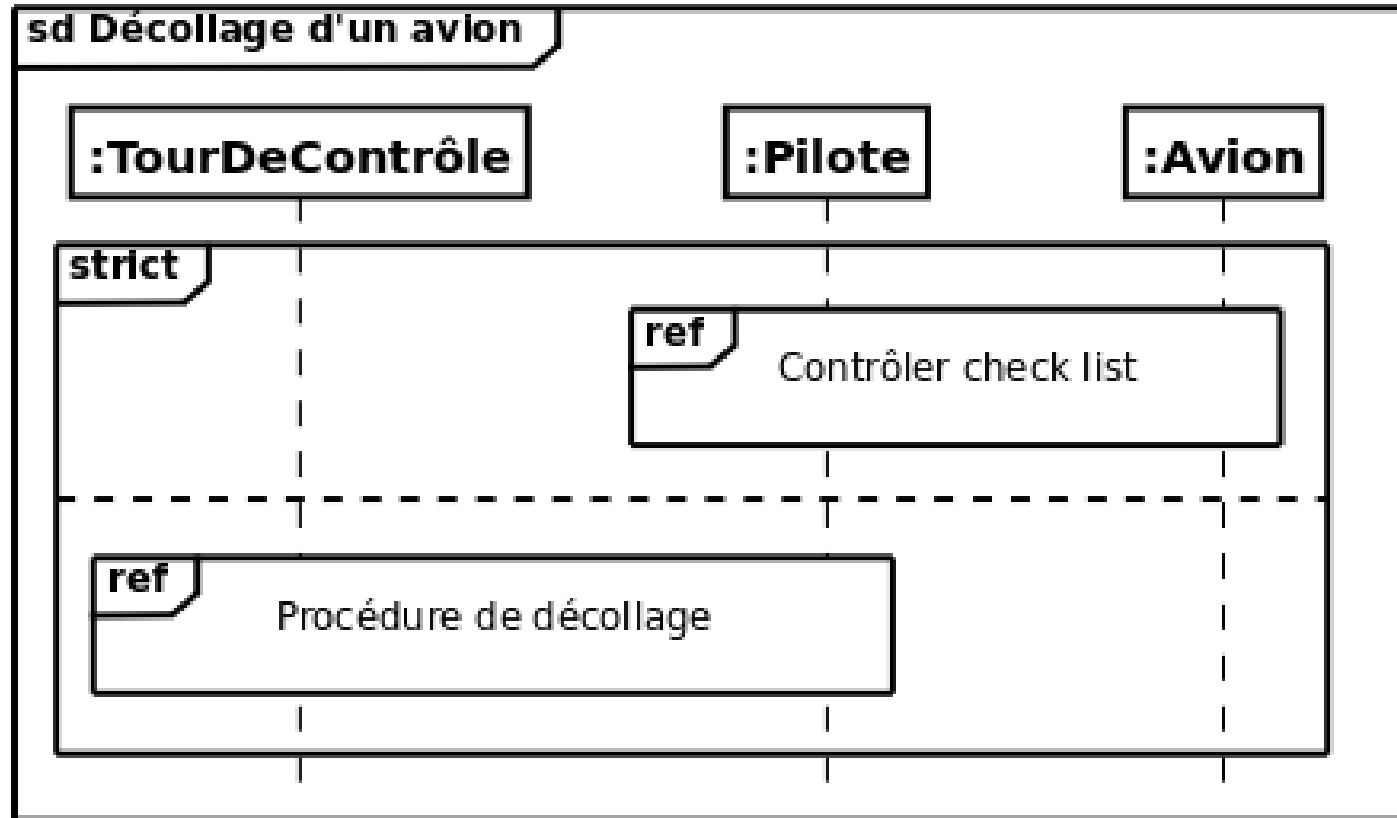


2.5.2.6 - Opérateurs courants des cadres d'interaction

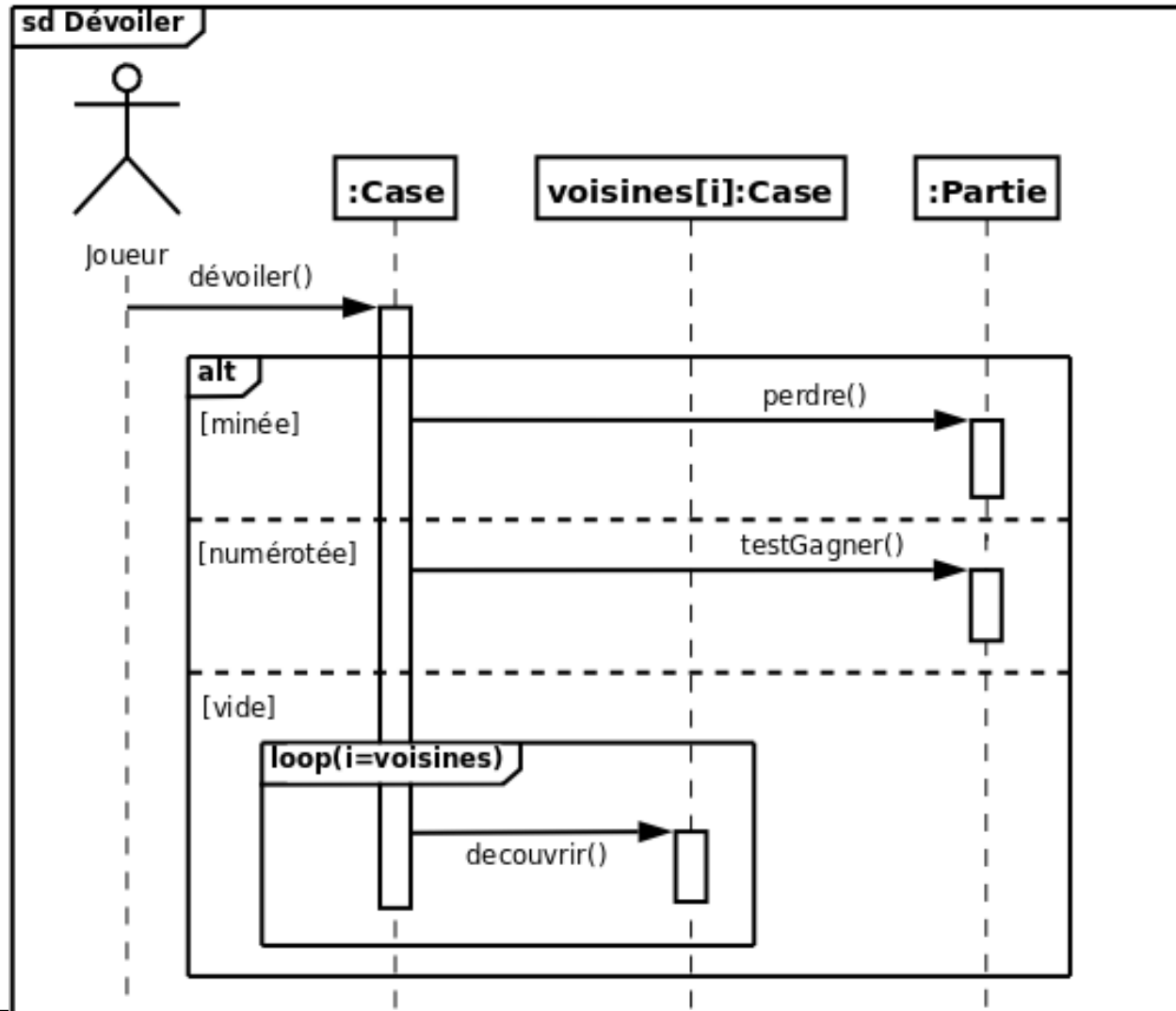


alt	Alternative
opt	Optionnel
par	Parallèle
loop	boucle
region	Région critique
neg	Négatif
ref	Référence à un autre diagramme
sd	Diagramme de séquence
break	Exception
critical	Section critique

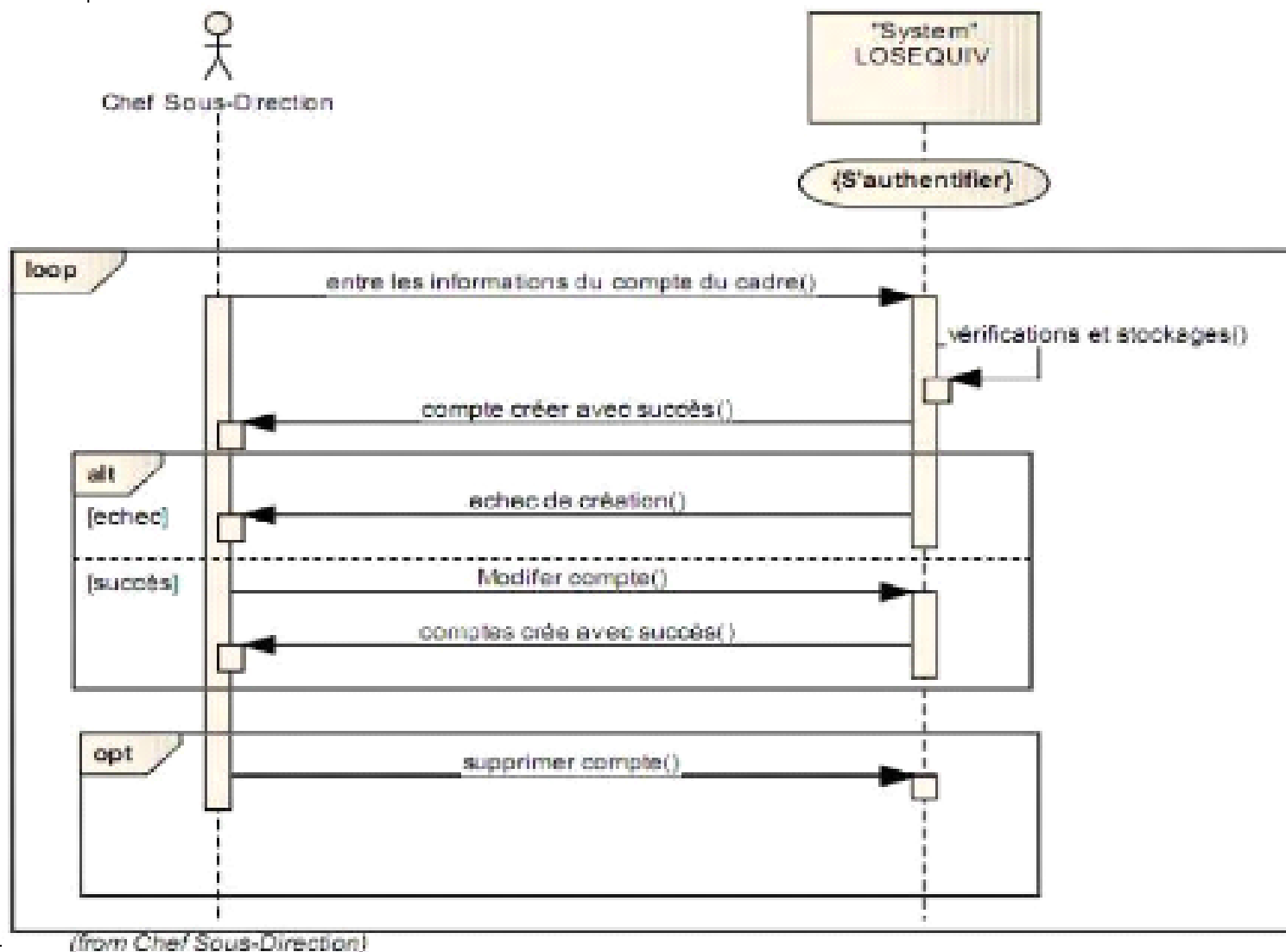
- les opérateurs de choix et de boucle : **alternative**, **option**, **break** et **loop** ;
- les opérateurs contrôlant l'envoi en parallèle de messages : **parallel** et **critical region** ;
- les opérateurs contrôlant l'envoi de messages : **ignore**, **consider**, **assertion** et **negative** ;
- les opérateurs fixant l'ordre d'envoi des messages : **weak sequencing** , **strict sequencing**.



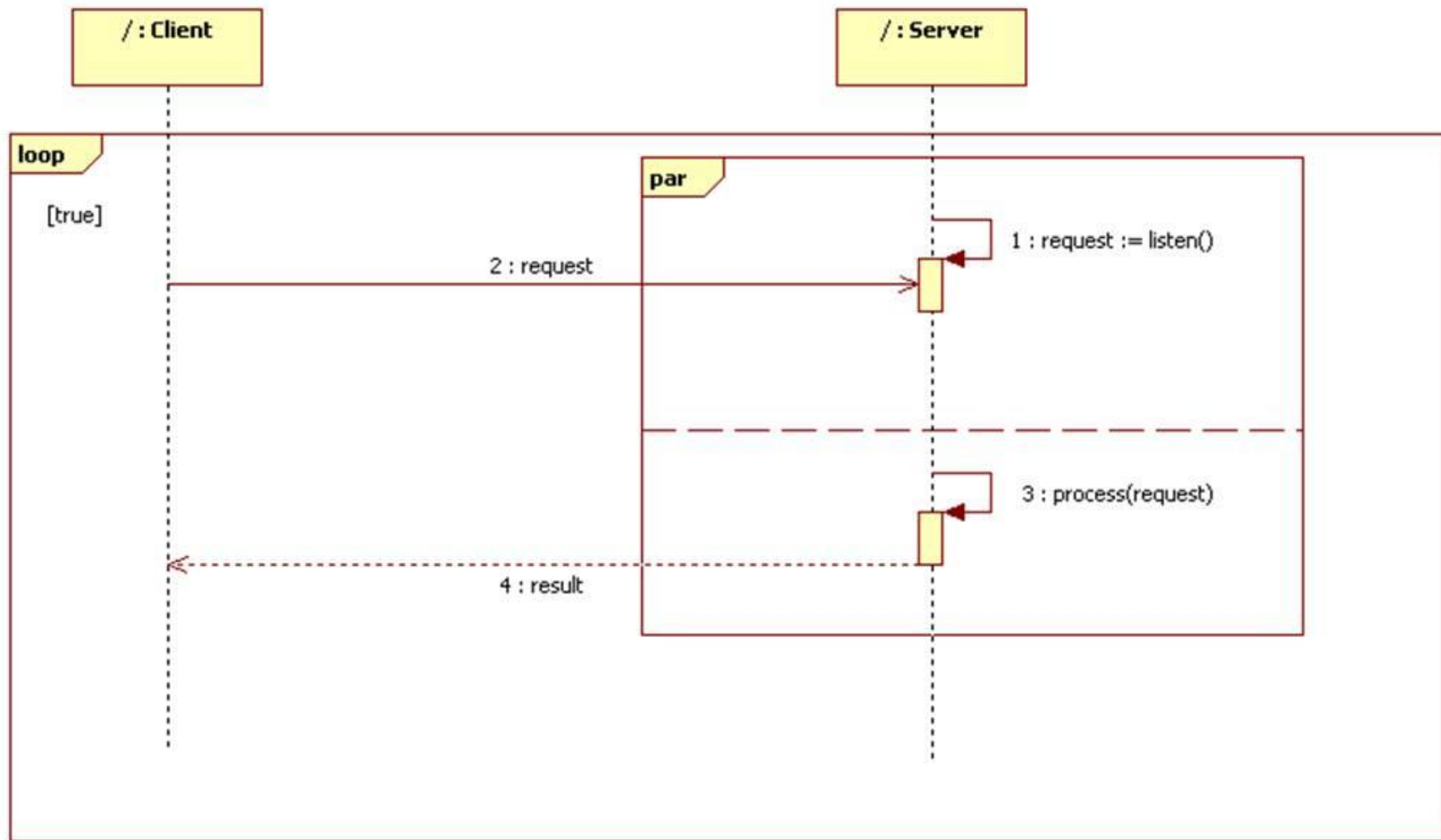
2.5.2.6 - Opérateurs courants des cadres d'interaction



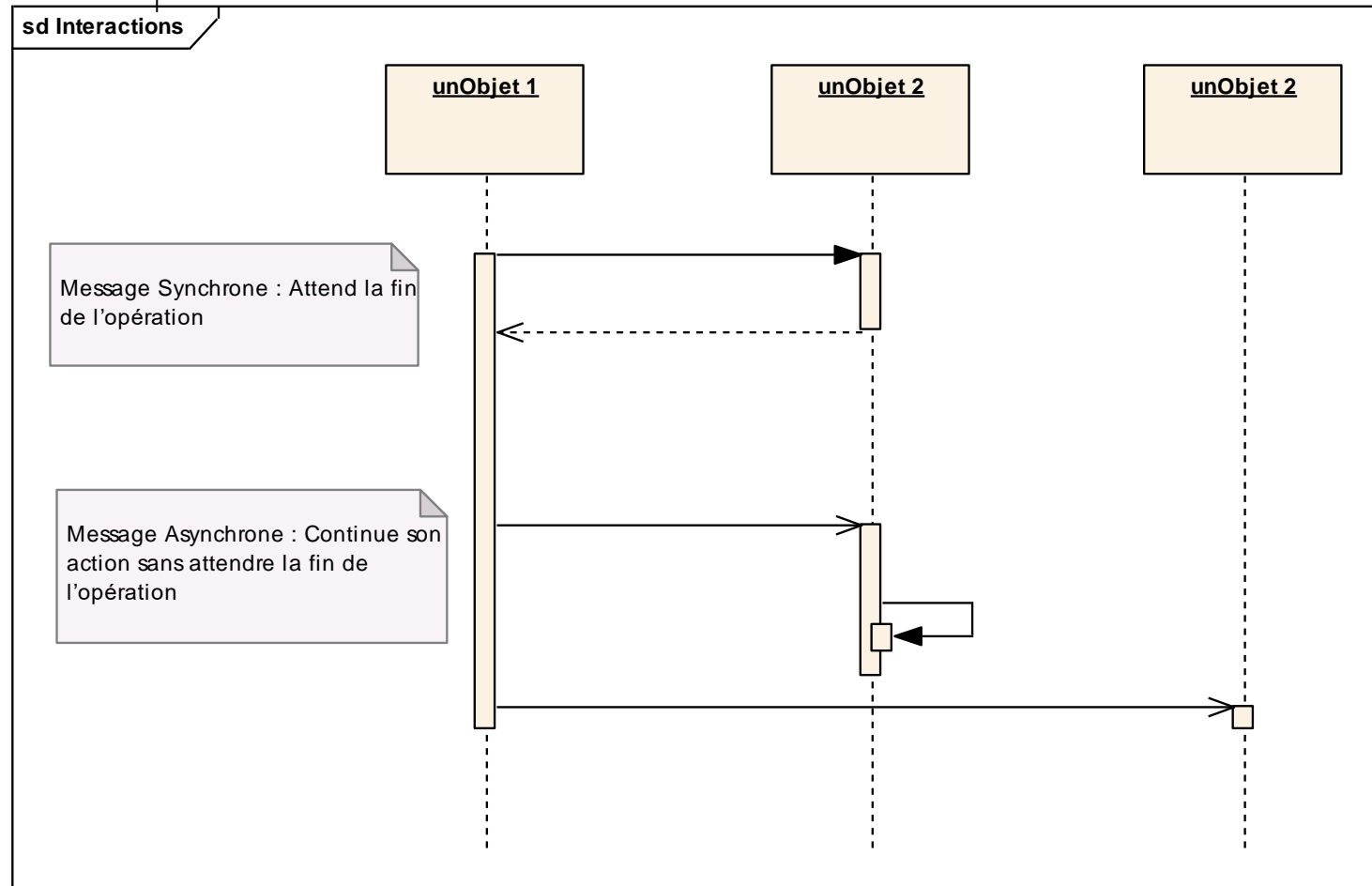
2.5.2.6 - Opérateurs courants des cadres d'interaction



2.5.2.6 - Opérateurs courants des cadres d'interaction



2.5.2.7 - Appel synchrone et asynchrone



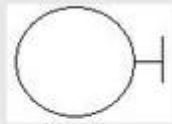
2.5.2.8 – Interface/Contrôleur/Entité



• Inclus dans UML :



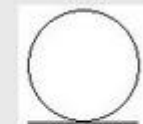
Acteur



Interface
boundary



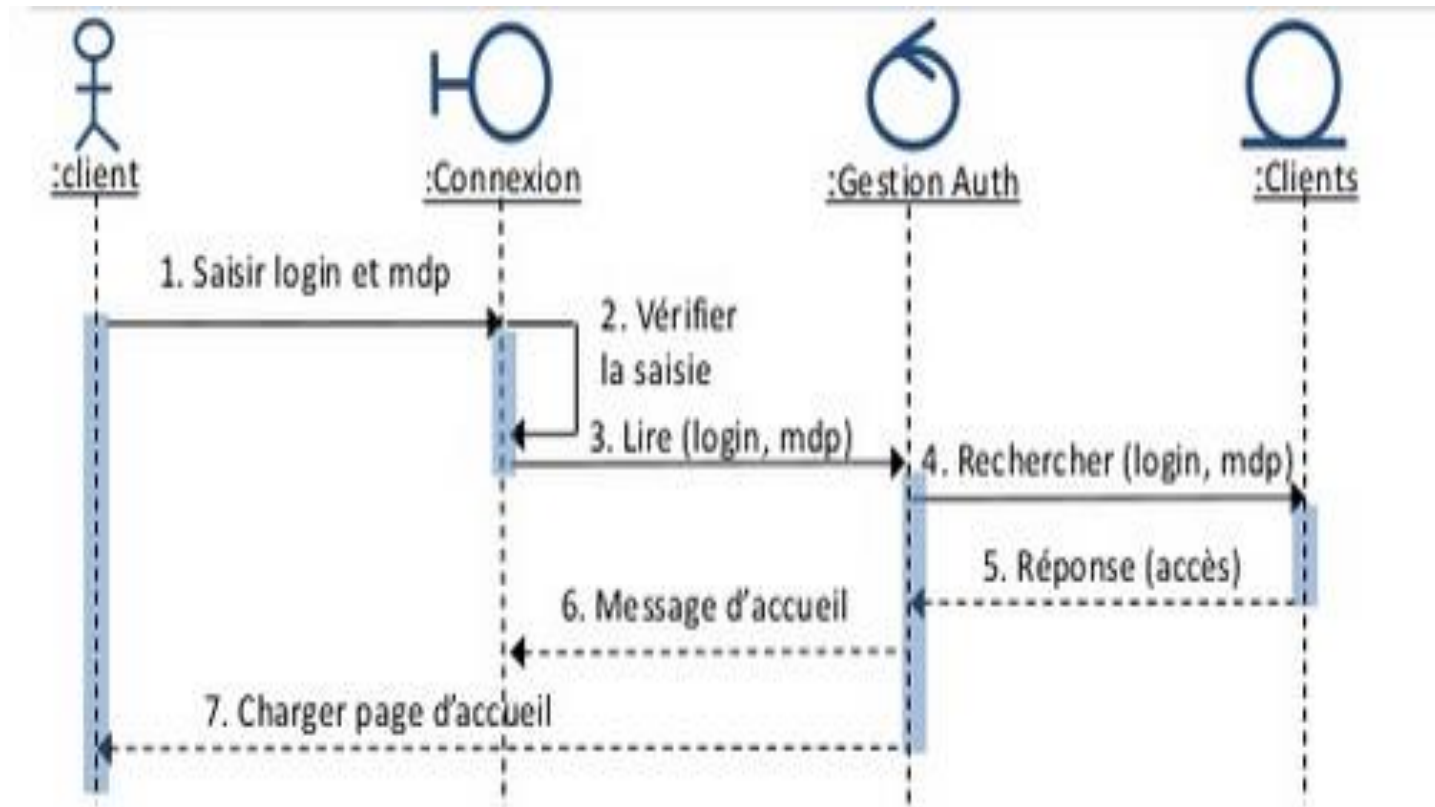
Contrôleur
Controller



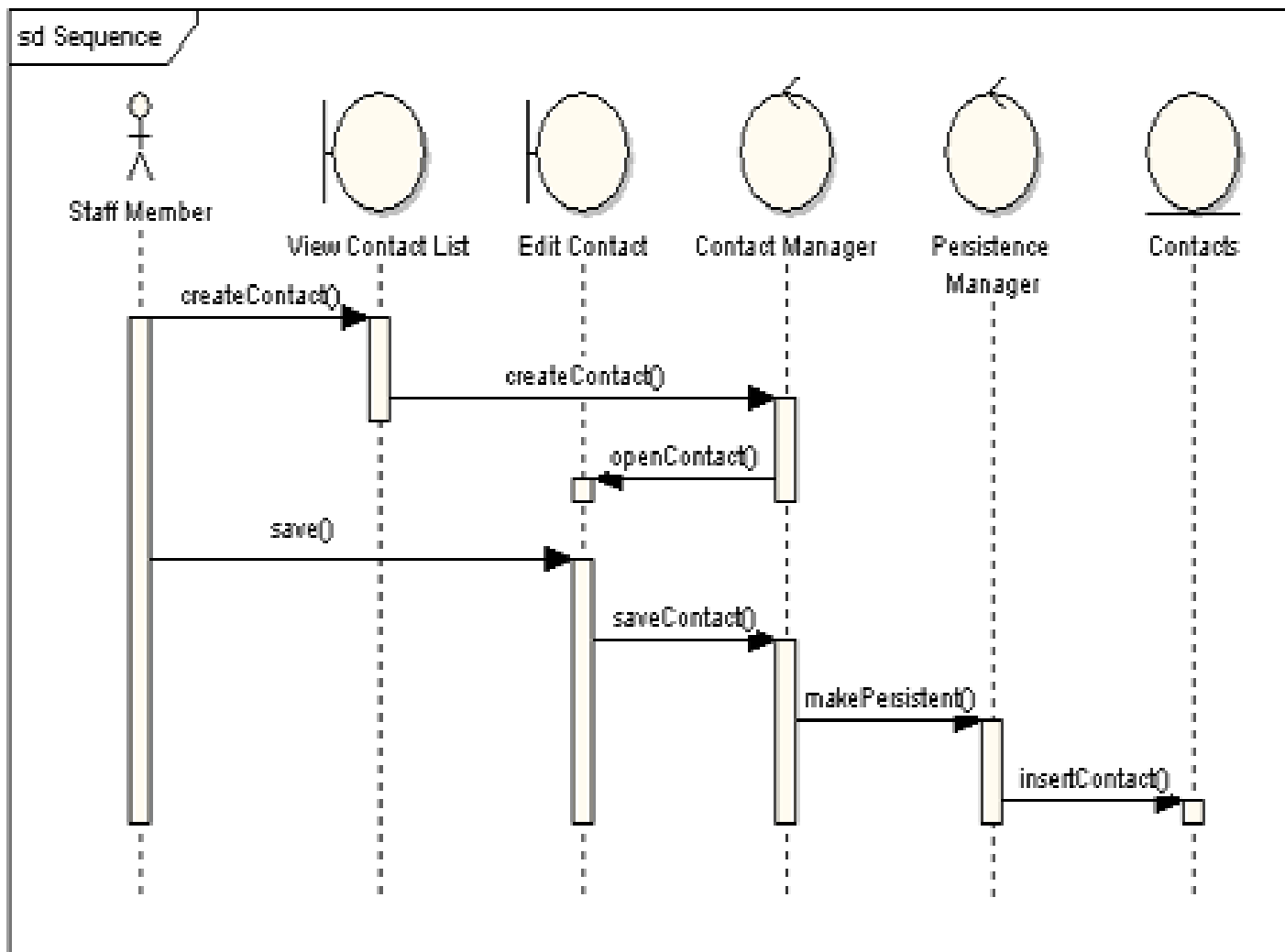
Entité
persistante
entity

- **Entities** (*modèle*)
Objets représentant des données système, souvent issues du modèle de domaine.
- **Boundaries** (*vue / service collaborateur*)
Objets qui s'interfaçent avec les acteurs du système (par exemple un **utilisateur** ou un **service externe**). Les fenêtres, les écrans et les menus sont des exemples de limites qui s'interfaçent avec les utilisateurs.
- **Contrôle**(*contrôleur*)
Objets qui interviennent entre les limites et les entités. Ceux-ci servent de lien entre les éléments de frontière et les éléments d'entité, mettant en œuvre la logique requise pour gérer les différents éléments et leurs interactions. Il est important de comprendre que vous pouvez décider d'implémenter des contrôleurs dans votre conception comme autre chose que des objets - de nombreux contrôleurs sont assez simples pour être implémentés en tant que méthode d'une entité ou d'une classe de limites par exemple

2.5.2.8 – Interface/Contrôleur/Entité



2.5.2.8 – Interface/Controleur/Entite

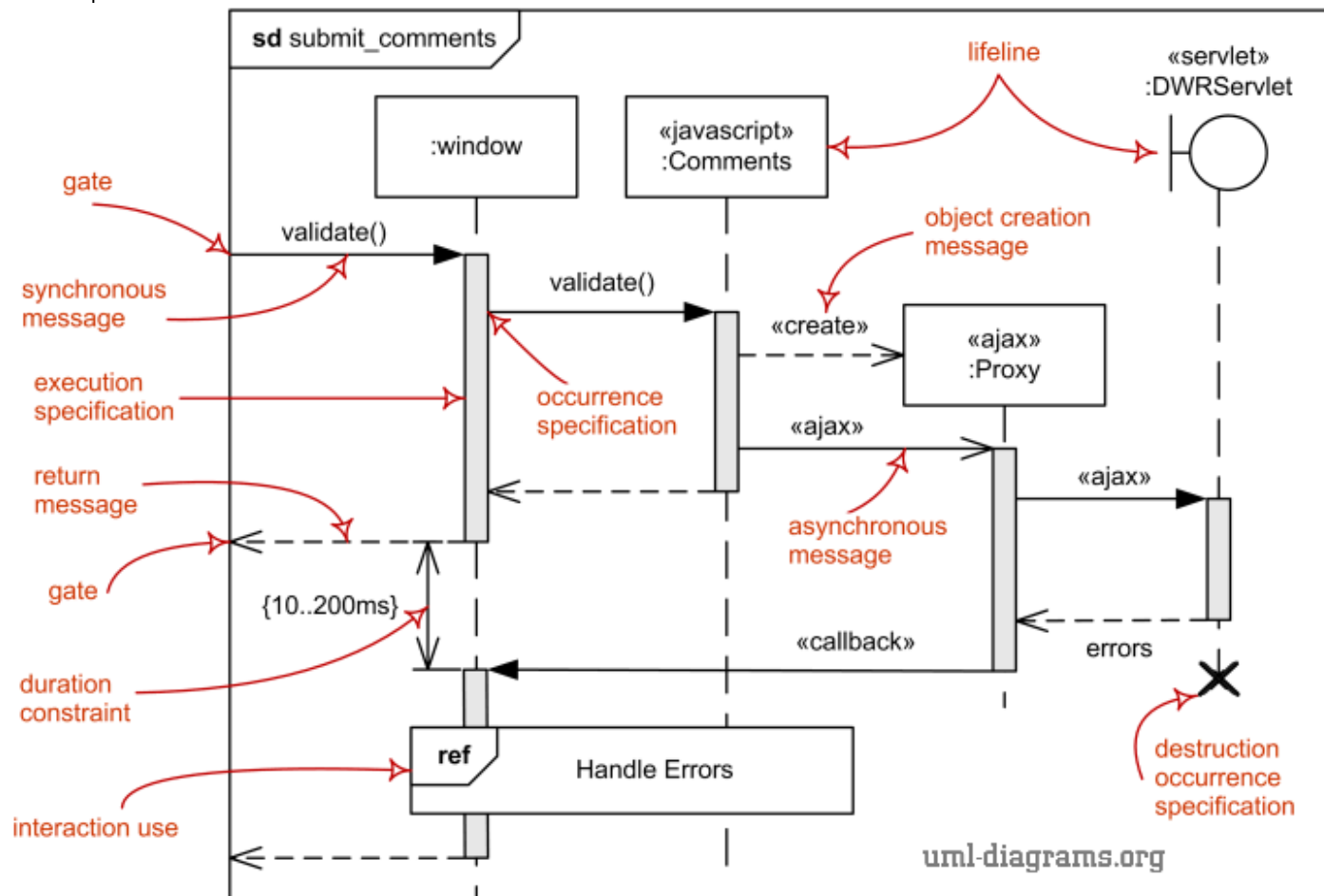




Les diagrammes de séquences peuvent servir à illustrer

- Les cas d'utilisation
 - instance du cas d'utilisation,
 - présente un scénario d'utilisation particulier du système,
 - met en œuvre les acteurs et le système.
- En analyse/Conception
 - les interactions temporelles des objets,
 - la description ou le résultat de cas test (unitaire, ..., recette)

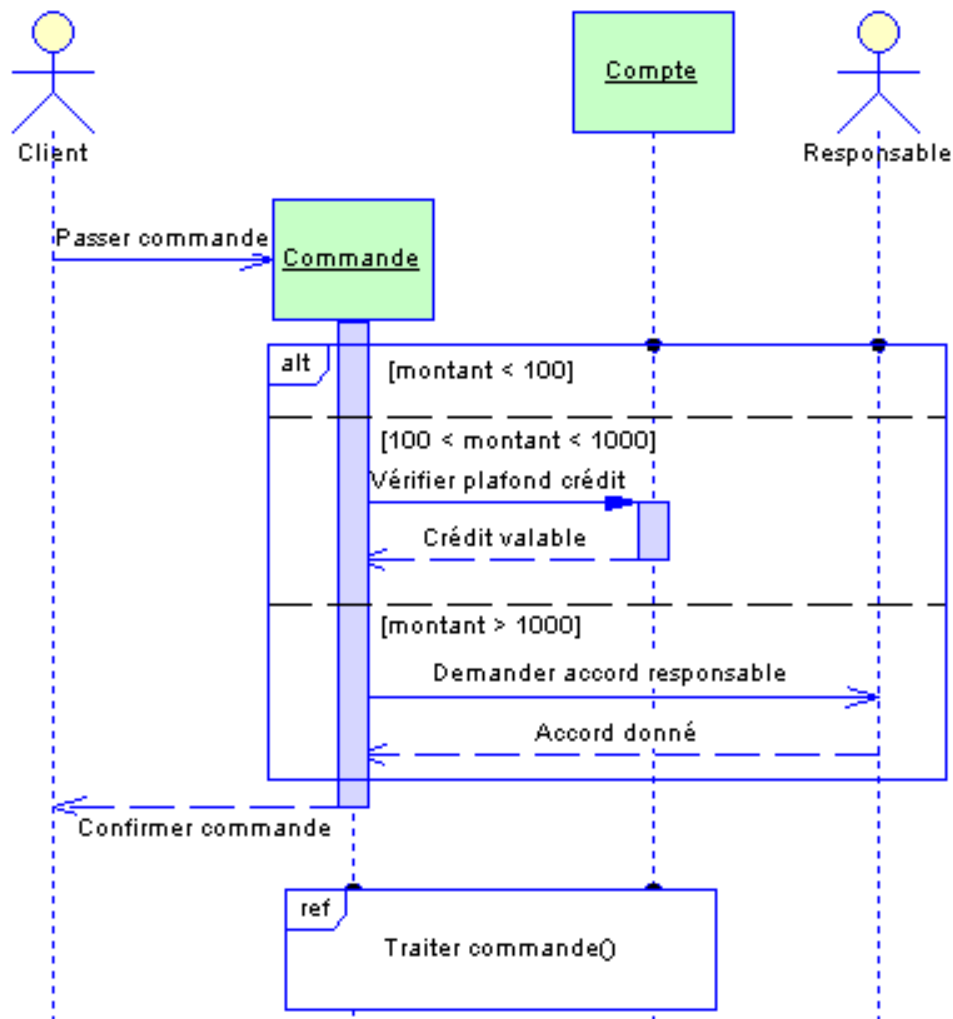
2.5.4.1 – Exemple :



<https://www.uml-diagrams.org/sequence-diagrams.html>

<https://www.uml-diagrams.org/sequence-diagrams-examples.html>

2.5.4.2 - Exemple



http://infocenter-archive.sybase.com/help/index.jsp?topic=/com.sybase.stf.poweramc.docs_12.0.0/html/dcgup/dcgup181.htm

2.5.4.3 - Exemple

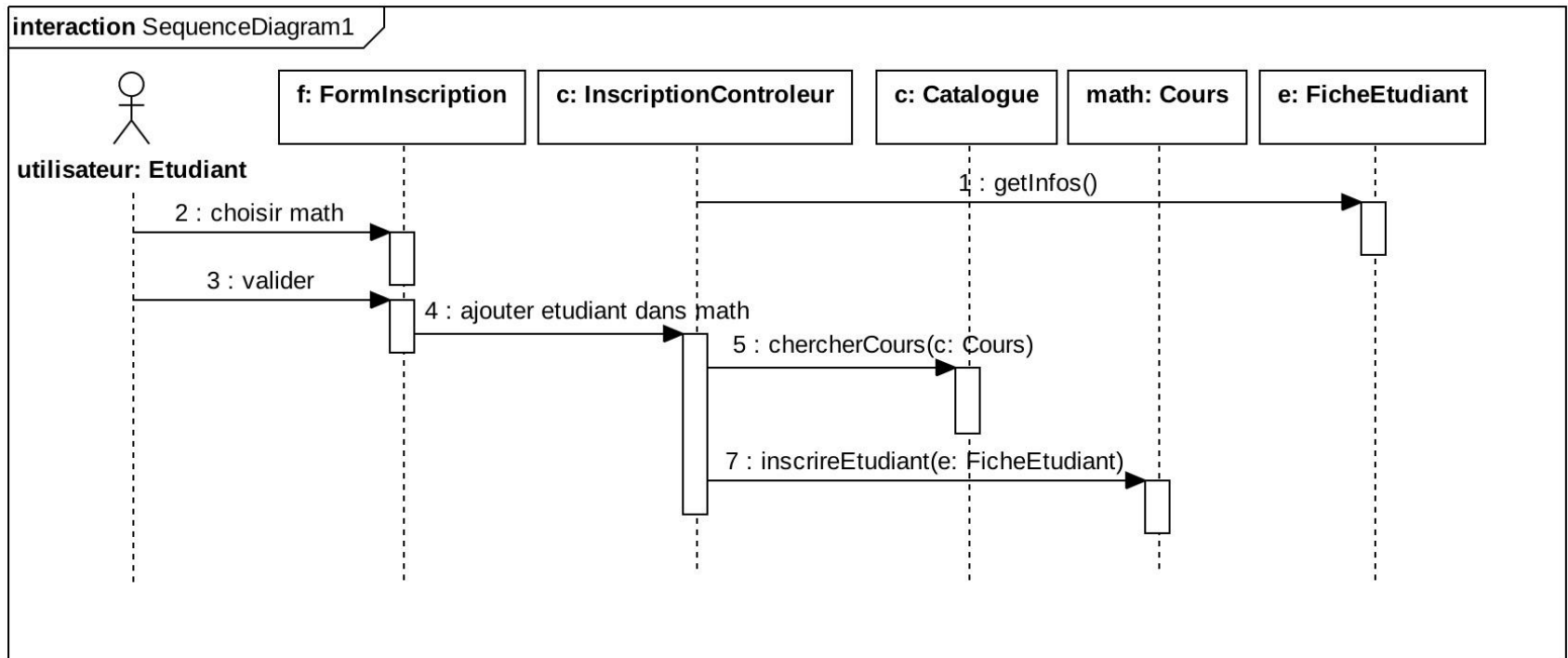


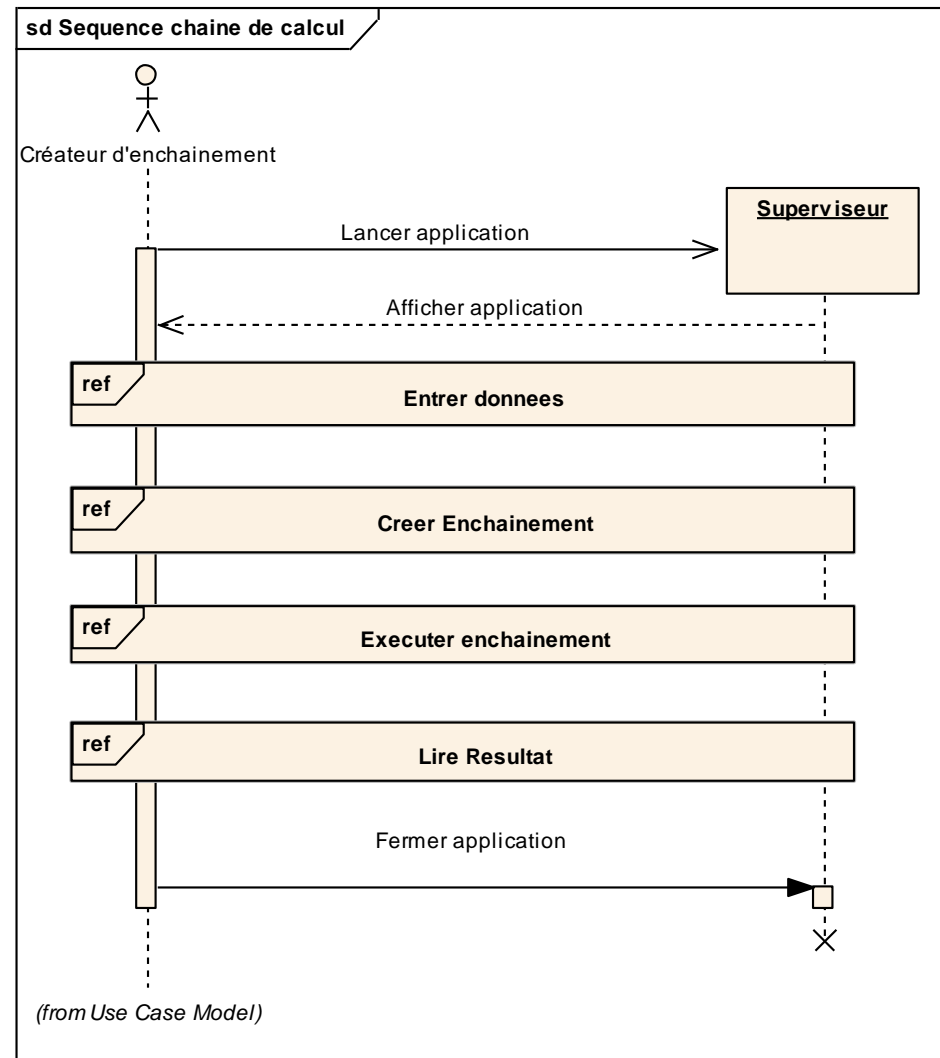
Diagramme de séquence du scénario Inscrire un élève au cours

<https://manurnx.wp.imt.fr/2017/01/26/modelisation-uml-dapplications-logicielles/>

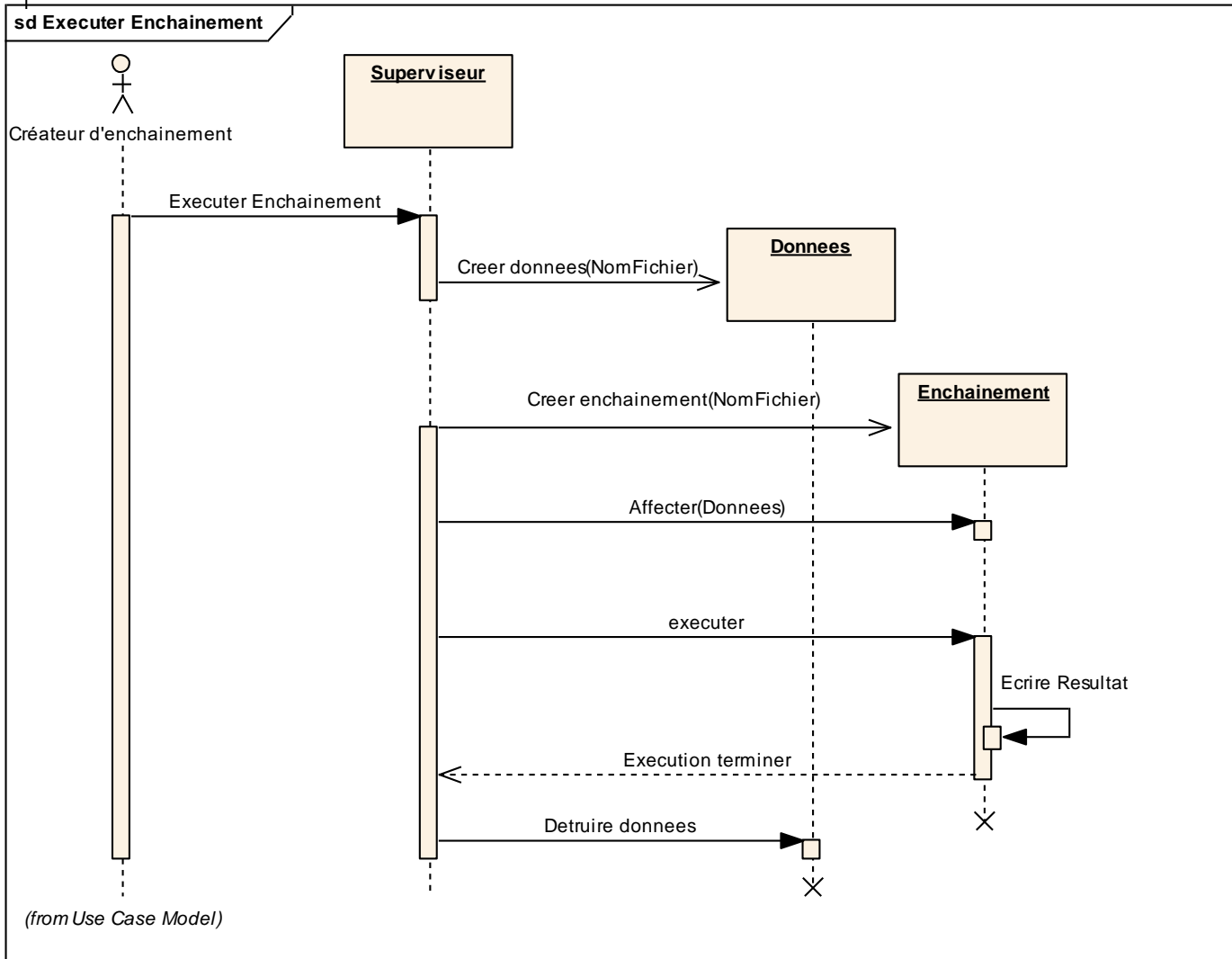


- Exemple : Séquence chaîne de calcul
- Exemple : Exécuter enchaînement
- Exemple : Créer enchaînement

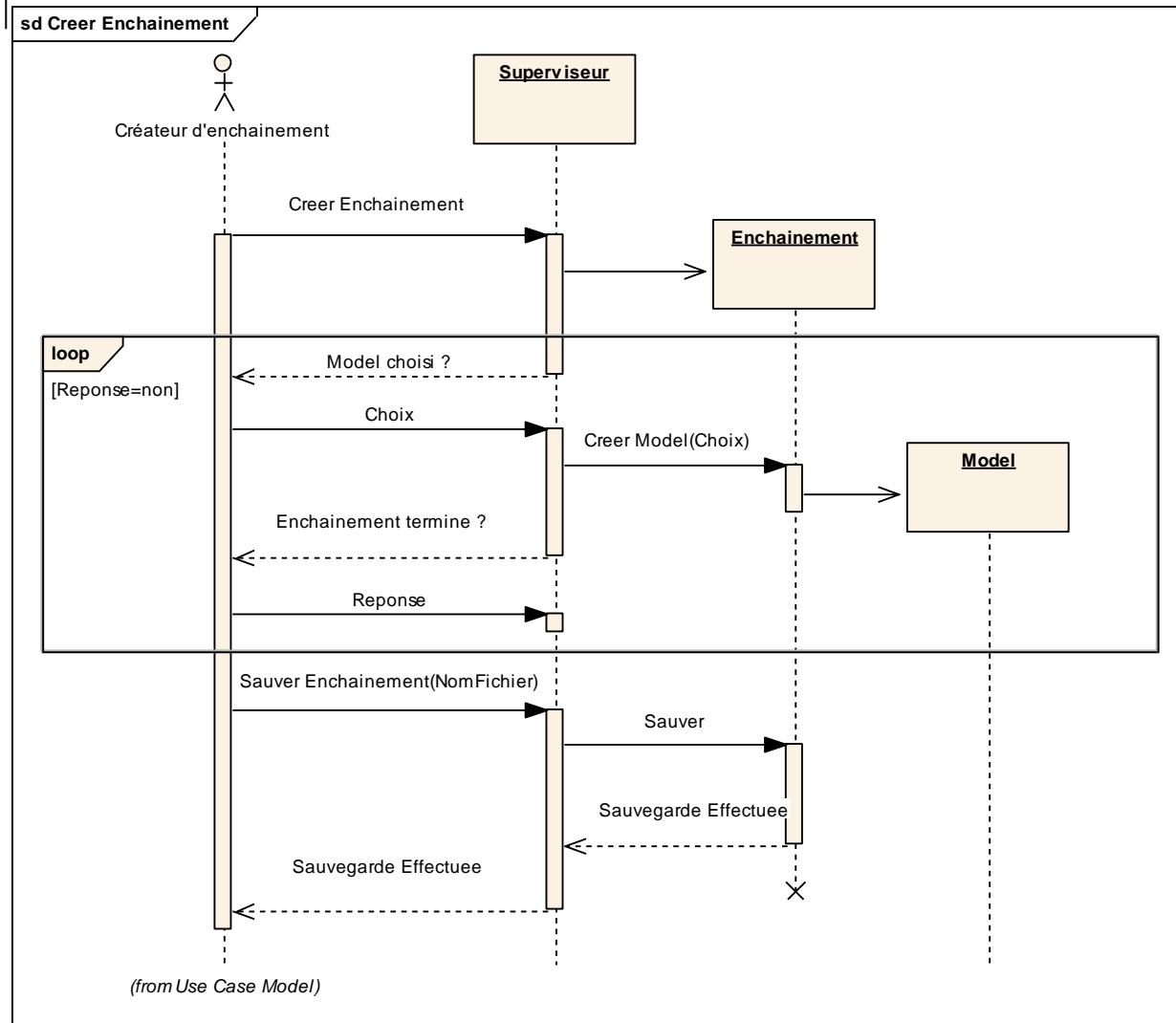
2.5.4.4a – Exemple : Séquence chaîne de calcul



2.5.4.4b – Exemple : Exécuter enchaînement



2.5.4.4c – Exemple : Créer enchaînement



3 – Conclusion



3.1 - Avantages et Inconvénients de l'approche objet

3.2 - Comment utiliser UML



Avantages d'UML

UML est un langage formel et normalisé

- gain de précision
- gage de stabilité
- accélère et structure les développements logiciels
- encourage l'utilisation d'outils

UML est un support de communication performant

- Il cadre l'analyse.
- Il facilite la compréhension de représentations abstraites complexes.
- Son caractère polyvalent et sa souplesse en font un langage universel.

Inconvénients d'UML

La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.

Le processus (non couvert par UML) est une autre clé de la réussite d'un projet.

.



Multipliez les vues sur vos modèles !

Un diagramme n'offre qu'une vue très partielle et précise d'un modèle.
Croisez les vues complémentaires (dynamique / statique)

Restez simple !

Utilisez les niveaux d'abstraction pour synthétiser vos modèles
(ne vous limitez pas aux vues d'implémentation).

Ne surchargez pas vos diagrammes.

Commentez vos diagrammes (notes, texte...).

Utilisez des outils appropriés pour réaliser vos modèles !

4 – Bibliographie

