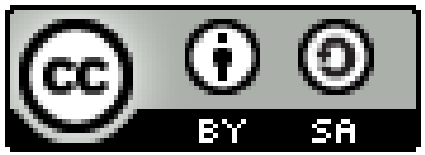
UML

Christine Plumejeaud
IR CNRS à Poitiers, UMR 7301 Migrinter

ANF RBDD – UML pour les bases de données

Ce(tte) oeuvre est mise à disposition selon les termes de la Licence Creative Commons



Programme de l'ANF

8h30-12h30 / 10h

14h-17h50 / 16h (20 min)

1. Présentation RBDD - Introduction
2. Diagramme de cas d'utilisation
3. Diagramme d'activité/séquence

Lundi après-midi

4. Fil rouge : analyse fonctionnelle

Mardi matin

5. Diagramme de classe

Mardi après-midi

6. Persistance BDD
7. Héritage et POSTGRES

Mercredi

8. QCMs pour vous tester

Jeudi matin

9. Le patron composite
10. Lire UML - Evaluation en ligne.

Plan Lundi

14h-17h50
Pause à 16h (20 min)

1. Présentation RBDD : 15 min
2. Introduction : 30 min
3. Les cas d'utilisation : 30 min
   - Un exercice : 30 min

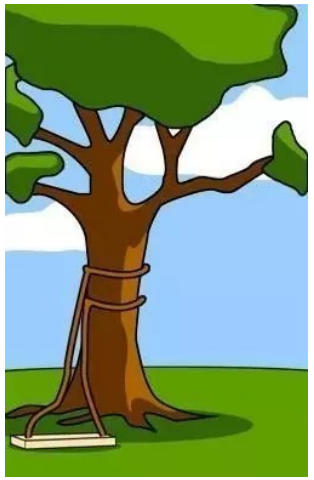
4. Diagramme d'activité : 15 min
   - Un exercice : 30 min
5. Diagramme de séquence : 15 min
   - Un exercice : 30 min

MOTIVATIONS POUR ENSEIGNER UML

Introduction

Une histoire bien connue - 1/5



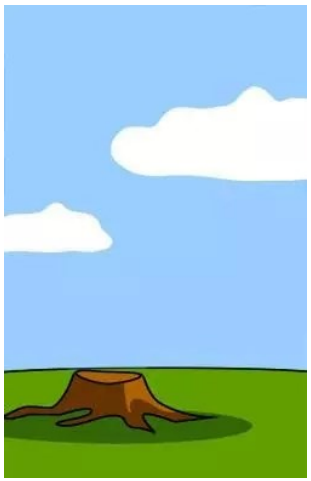
1. **Ce que le client a expliqué**
2. **Ce que le chef de projet a compris**
3. **La première d'analyse des ingénieurs**
4. **Ce que les ingénieurs ont conçu**

https://www.astuces-pratiques.fr/electronique/analyse-du-besoin-ce-que-le-client-voulait

Une histoire bien connue - 2/5

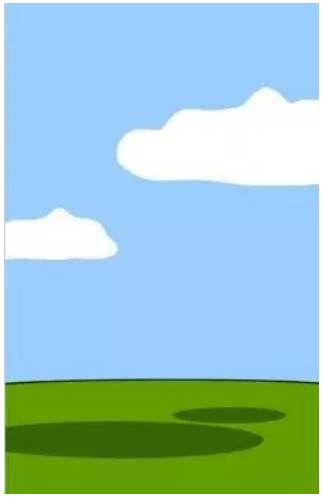
5. La présentation de la maquette au client
6. La première version : prototype
7. Les délais
8. Le soutien



https://www.astuces-pratiques.fr/electronique/analyse-du-besoin-ce-que-le-client-voulait

Une histoire bien connue - 3/5

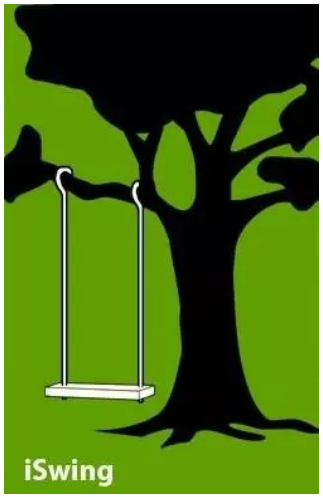


9. Les rapports

La documentation est essentielle pour capitaliser le savoir et le transmettre aux personnes arrivant dans l'entreprise.

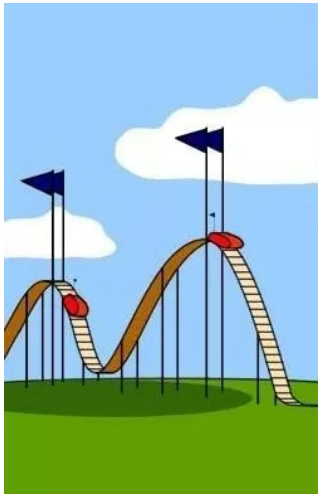
10. La promesse du commercial

Le commercial met en avant sa solution. La balançoire sera luxueuse, un véritable canapé connecté et intelligent. Le soleil levant est en prime !



11. La publicité de l'équipe marketing

Le marketing propose quelque chose d'innovant, suscitant l'émotion et l'adéquation à des valeurs telles que l'écologie, ou simplement une expérience de plaisir !



12. Ce que le client a payé

En termes de prix, le client a payé au delà de son budget.

https://www.astuces-pratiques.fr/electronique/analyse-du-besoin-ce-que-le-client-voulait

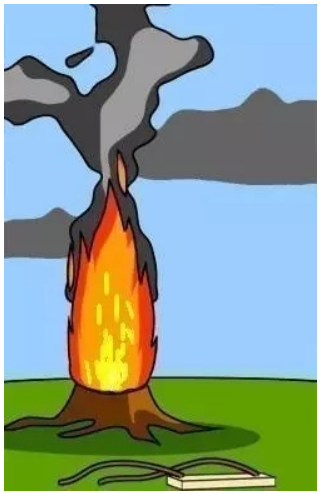
Une histoire bien connue - 4/5

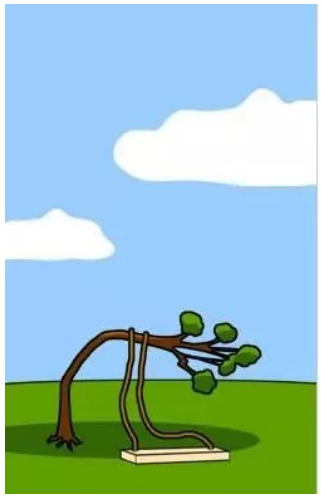
13. Les tests

14. La mise en place initiale

15. Le plan de reprise d'activité

16. La tierce maintenance applicative





https://www.astuces-pratiques.fr/electronique/analyse-du-besoin-ce-que-le-client-voulait

Une histoire bien connue – 5/5

Ce que le client voulait vraiment

Le gaspillage dans les bases de données

1. La production d'une base de données correspond à un besoin, un usage, voire à une question d'analyse
2. Le gaspillage peut être :
   - En amont : ne pas bien réfléchir à ses besoins, et choisir de faire une usine à gaz pour répondre à un besoin basique
   - En aval : récupérer un historique technique et fonctionnel, sans documentation appropriée

A propos de la documentation

A l'aire de l'open-data et de l'open-source

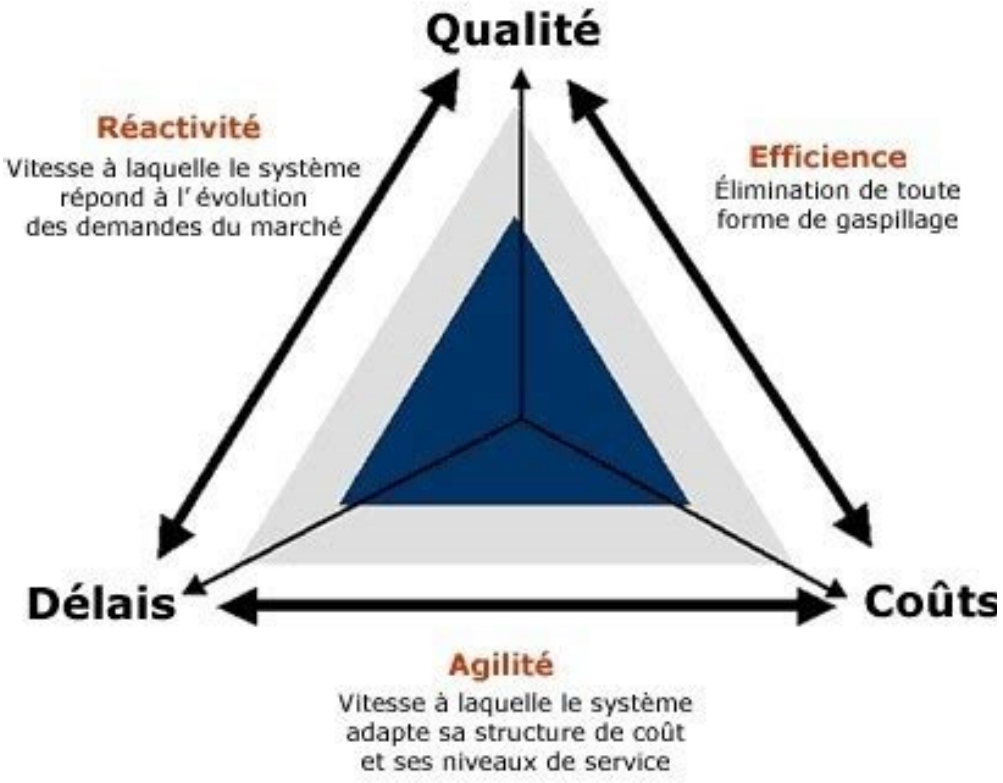
Les diagrammes et la documentation coûtent un temps précieux aux développeurs et deviennent rapidement obsolètes.

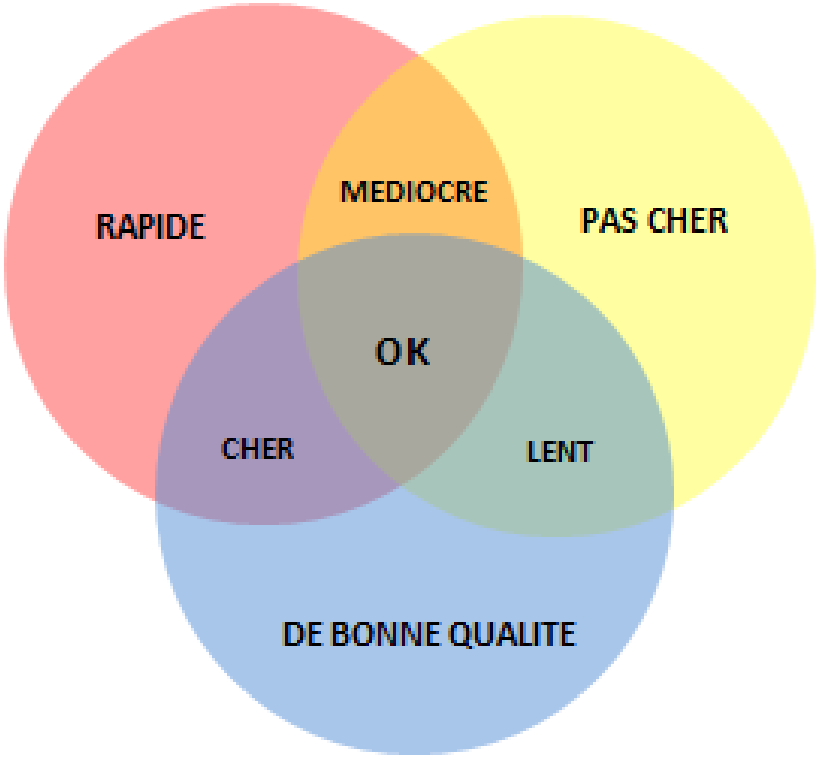
Mais l'absence de diagrammes ou de documentation ruine la productivité et nuit à l'apprentissage organisationnel.

https://mermaid.js.org/intro/

Mermaid répond à ce problème en permettant aux utilisateurs de créer des diagrammes facilement modifiables, qui peuvent également être intégrés à des scripts de production (et à d'autres éléments de code).

Gestion de projet





http://blog.ametragroup.com/excellence-operationnelle-demarche-damelioration-continue-triptyque-qualitedelaicout

https://www.la-rache.com/

- Accueil / News
- Présentation
- La F.A.Q
- Lexique
- Publications
- Consulting
- Formations
- Certificat de reussite
- En images
- Témoignages
- Liens

LA RACHE

/lamethoderache

/larache

Bienvenue sur le site de l'IILaR

(l'International Institute of La RACHE)

Le but de l'IILaR est de promouvoir la méthodologie de La RACHE. La RACHE, solution globale de génie logiciel, est un ensemble de techniques, de méthodes et de bonnes pratiques décrivant - des spécifications à la maintenance - comment produire du logiciel dans des conditions à peu près satisfaisantes et approximativement optimales.

Nous ne prétendons certes pas qu'il s'agisse d'une nouvelle méthode révolutionnaire, bien au contraire : La RACHE est mise en pratique depuis les débuts de l'informatique par de nombreux particuliers, étudiants et entreprises. Il semblerait même que la métempsychose récurrente du concept de la RACHE soit intrinsèquement axiomale chez certains grands éditeurs. Cependant, nous considérions que cette méthode nécessitait une formalisation ainsi qu'une normalisation rigoureuse afin d'être enfin utilisée fièrement et officiellement.

La RACHE est particulièrement adaptée à l'informatique, industrie qui a fait émerger le concept et l'a vulgarisé. Nonobstant, le corpus méthodologique, extrêmement flexible, est facilement transposable à de nombreux autres domaines.

News

01/01/2024

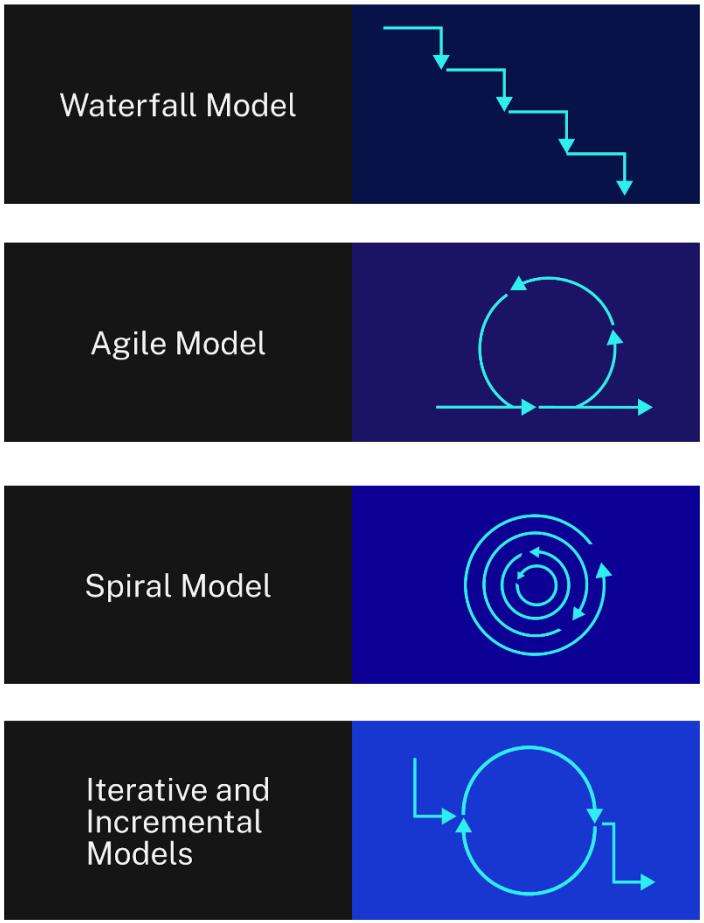
- 🎉🎈🎊 API NIOÙ HYÈRE et bons Jeux Olympiques à La RACHE !!! 🥇🥈🥉

01/05/2023

- Création de notre certification à La RACHE

Des méthodes de développement

https://www.door3.com/fr/blog/different-software-development-models



+ RUP

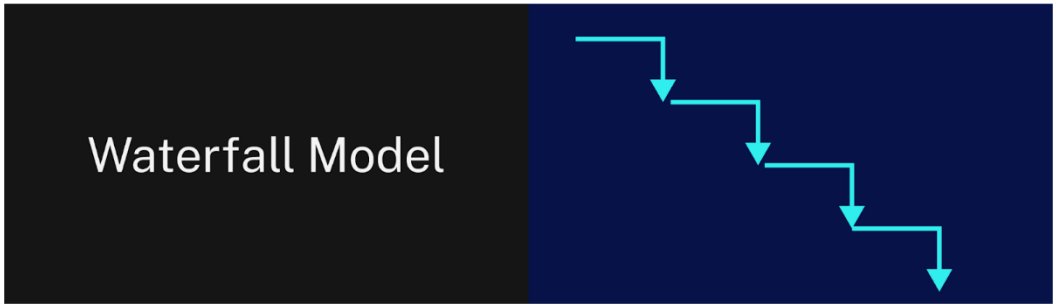
- limiter les effets de la contingence d'un projet informatique,
- à maîtriser au maximum le tryptique qualité/délai/coût

La rationalité contre les hasards

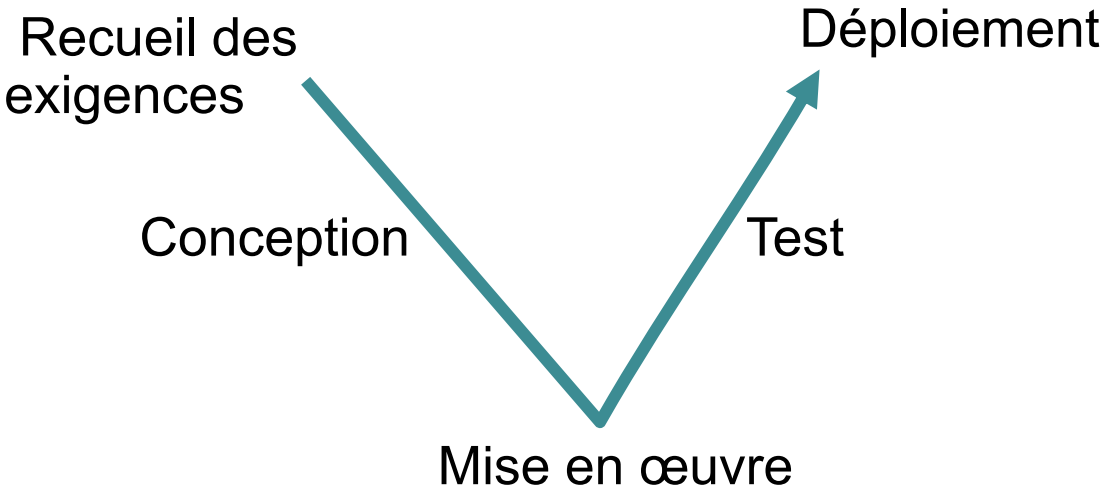
La coordination contre la spontanéité

Le respect

CHUTE d'EAU / V

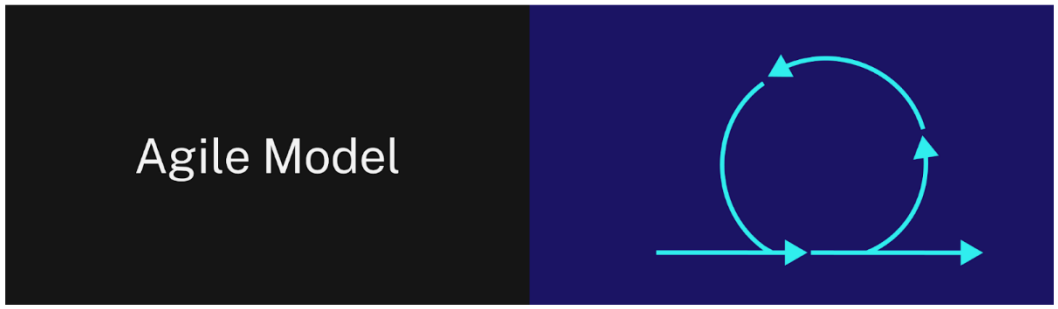


- **Recueil des exigences**
- Conception
- Mise en œuvre
- Test : unitaires, d'intégration, de recette
- Déploiement



Et après, c'est terminé ?

AGILE

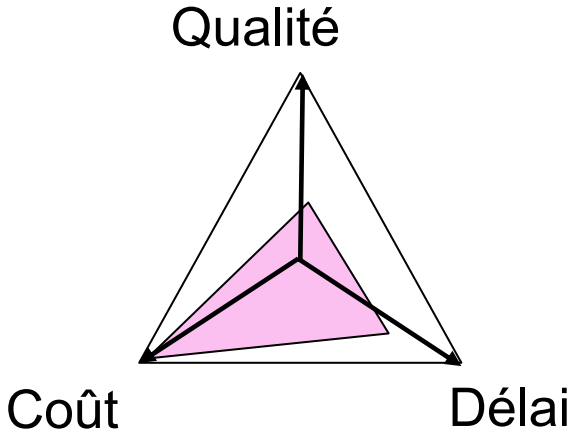


Sprint = une itération

- **Planification du sprint**
- Exécution du sprint
- Intégration et test continus
- **Révision et retour d'information** (implication des clients / utilisateurs)
- Rétrospective du sprint

« idéal pour les projets dynamiques dont les exigences évoluent

et il encourage la livraison rapide de solutions logicielles »



SPIRALE

- **Planification**
- Analyse des risques
- Prototypage
- Évaluation

Dans le modèle en spirale, la phase de planification commence par l'identification des objectifs, des exigences et des contraintes du projet. L'équipe de développement collabore avec les parties prenantes pour recueillir les exigences initiales et définir la portée globale du projet.

- Développement
- Test et intégration
- Déploiement
- Maintenance

La maintenance suppose une permanence du service informatique (vous ?) dans le temps…

https://www.door3.com/fr/blog/different-software-development-models

Itératives et incrémentales

Engagement des utilisateurs ➔ disponibilité ?

- Planification initiale
- Itération 1
- Tests et retours d'information
- Itération 2
- Test et retour d'information
- Itération 3
- Test et retour d'information

*Les modèles itératif et incrémental sont des modèles de développement de logiciels qui permettent à l'équipe de développement de livrer un logiciel fonctionnel de manière incrémentale au fil du temps. Chaque itération ajoute de la valeur au projet, en s'appuyant sur les bases existantes et en intégrant les commentaires des utilisateurs et des parties prenantes. Cette approche permet un développement rapide, **un engagement précoce des utilisateurs** et une flexibilité permettant de s'adapter à l'évolution des besoins.*

La méthode RUP

RUP comme Rational Unified Process

Processus unifié rationnel

RUP est un processus de développement logiciel de *Rational*, une division d'IBM en 4 phases :

1. **Début** - L'idée du projet est énoncée. L'équipe de développement détermine si le projet mérite d'être poursuivi et quelles ressources seront nécessaires.
2. **Élaboration** - L'architecture du projet et les ressources requises sont ensuite évaluées. Les développeurs considèrent les applications possibles du logiciel et les coûts associés au développement.
3. **Construction** - Le projet est développé et complété. Le logiciel est conçu, écrit et testé.
4. **Transition** - Le logiciel est mis à la disposition du public. Les derniers ajustements ou mises à jour sont effectués en fonction des commentaires des utilisateurs finaux.

Il doit **éviter le gaspillage** des ressources et réduit les coûts de développement inattendus.

https://techlib.fr/definition/rup.html#:~:text=Signifie%20%C2%ABProcessus%20unifi%C3%A9%20rationnel%C2%BB,les%20tests%20et%20le%20d%C3%A9ploiement

Un bon cahier des charges

1. Définit le contexte autour du projet : évoquer la situation prévalant au moment donné : décrire l'état du parc logiciel et informatique, les motivations pour le nouveau projet (**POURQUOI**)
2. **Établit les besoins** et les contraintes liées au projet

   compatibilité informatique de la solution avec l'architecture de l'entreprise et les compétences des utilisateurs
3. Mentionne les résultats attendus

   - livraison de quoi (code, données, graphiques),
   - et comment (fichier/URL/plateforme), et où (dispositif fixe/mobile)
4. Définit un calendrier, des échéances
5. Définit les utilisateurs finaux

   nombre et caractéristiques des utilisateurs : ergonomie adaptée (accessibilité), Langues (FR, EN, ES, …), puissance de calcul…

UML

Le **Langage de Modélisation Unifié**, de l'anglais *Unified Modeling Language* (**UML**), est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet.

Les objectifs de UML

1. UML
   - notation graphique de modélisation à objets (9 types de diagrammes)
   - UML n'est PAS une méthode de développement, ni un langage de programmation
2. Objectifs :
   - Représenter des systèmes entiers par des concepts objets (pas juste les logiciels)
   - Lier explicitement les concepts et le code
   - Modéliser à différents niveaux de granularité
   - Langage utilisable à la fois par les humains et les machines par un formalisme bien établi ($\rightarrow$ génération de code)
   - Encourager l'utilisation des outils Orientés Objets
   - Offrir des mécanismes d'extension et de spécialisation pour pouvoir étendre les concepts de base

Intérêts de l'approche objet

1. « *Naturelle* » car basée sur le domaine d'application. Facilite la communication avec les utilisateurs.
2. Meilleur support pour *l'évolution* des besoins
   - Modélisation plus stable
   - Les évolutions du système ne remettent pas l'architecture en cause
3. Facilite la *réutilisation* de composants

•Encapsulation	→ systèmes modulaires
•Héritage	→ système extensibles
•Association	→ systèmes complexes

L'unification des méthodes objet

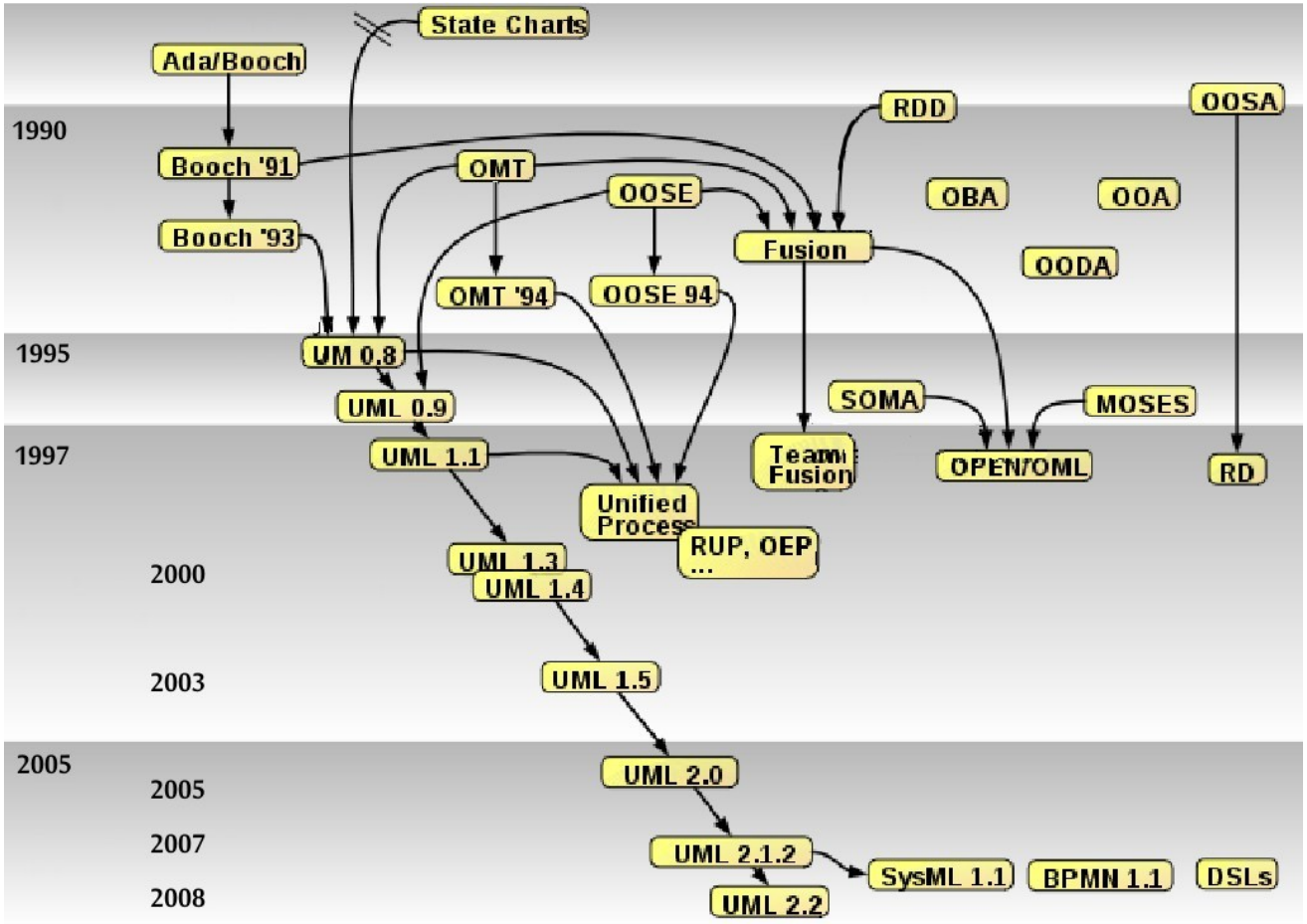
OMG (*Object Management Group*) (http://www.omg.org/) fondé en 1989. Consortium d'industriels pour la standardisation des méthodes objets, à l'origine du standard CORBA

Objectifs : maximiser la portabilité et la portabilité des logiciels par la définition de standards industriels, pour le développement d'applications commerciales orientées objet.

570 membres en 1995

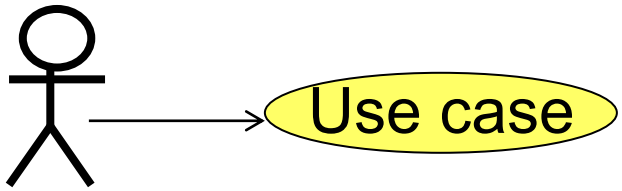
La genèse d'UML

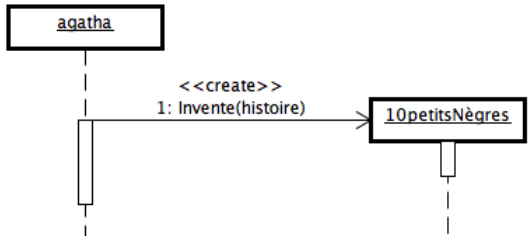
1. Foisonnement
2. Unification - 1995
3. Standardisation -
4. UML 2.0 - 2005



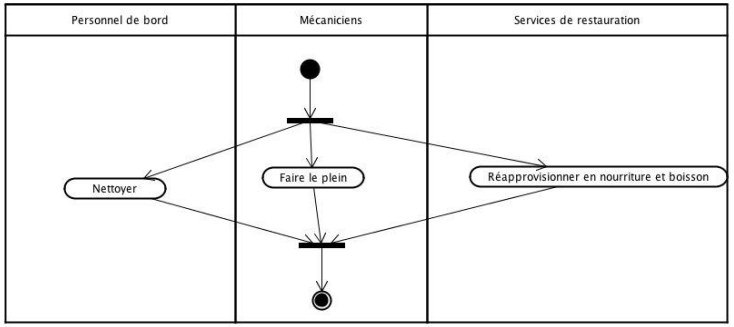
Les 9 diagrammes UML

1. Aspects fonctionnels:
   - cas d'utilisation
   - (diagrammes de séquence)
   - (diagrammes d'activité)

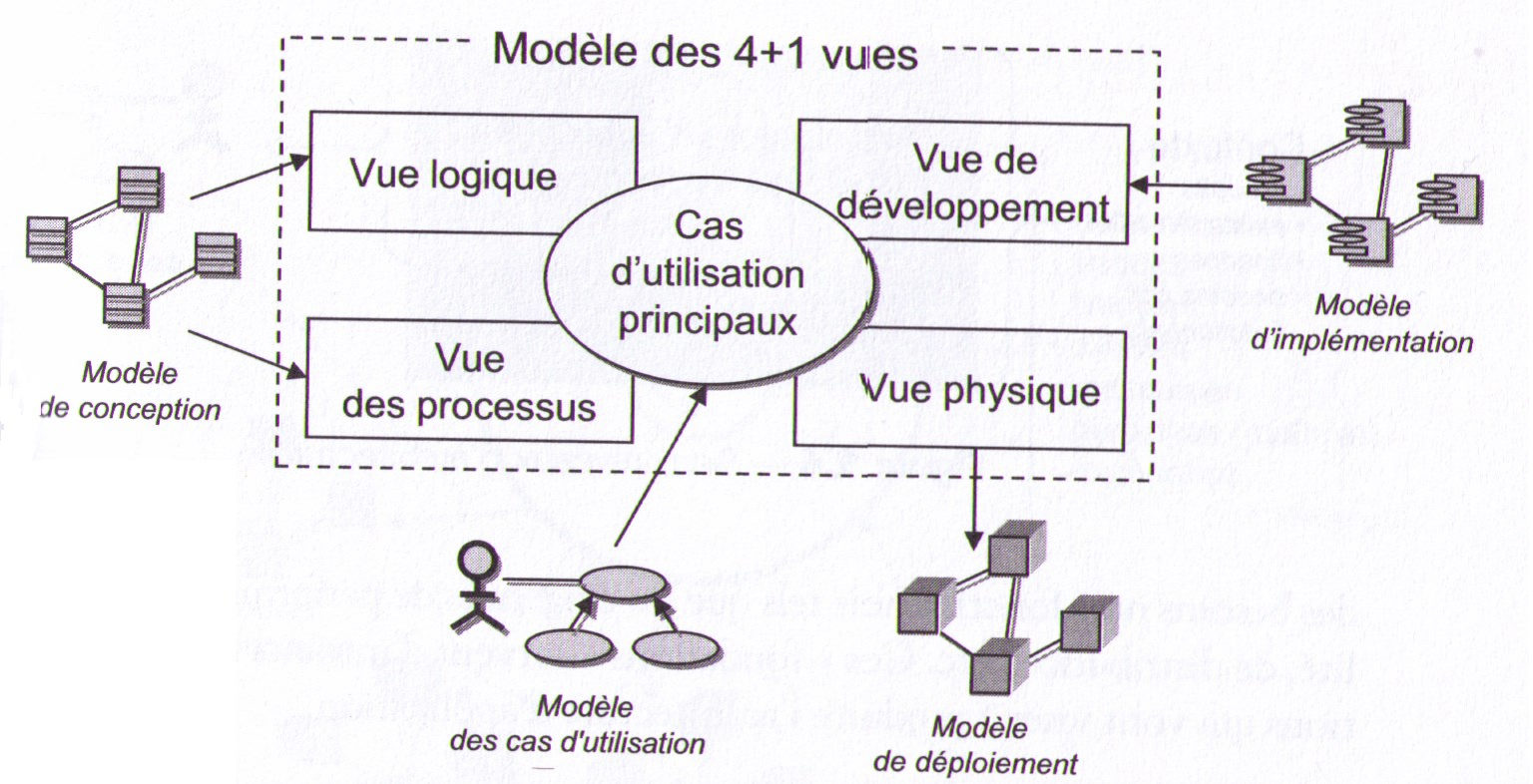




2. Aspects statiques
   - diagrammes de classes
   - diagrammes d'objets
   - Les diagrammes de composants / packages
   - Les diagrammes de déploiement
3. Aspects dynamiques
   - diagrammes de séquence
   - diagrammes d'activités
   - automates d'états-transitions
   - diagrammes de collaboration



Intégration d'UML au RUP



- Piloté par les cas d'utilisation
- Centré sur l'architecture (la structure du système)
- Itératif et incrémental

Modéliser les interactions autour d'une base de données

Décrit les interactions entre objets, et la façon dont ils collaborent ➔ comportement global du système

1. Cas d'utilisation

   capturent les besoins, et les interactions du système avec l'extérieur

2. Les diagrammes de séquence

   représentent la séquence des interactions entre les objets dans le temps

3. Les diagrammes d'activité

   illustrent le flux de contrôle entre les étapes d'un traitement

Diagramme de cas d'utilisation

Acteur/système/use case/include/extend

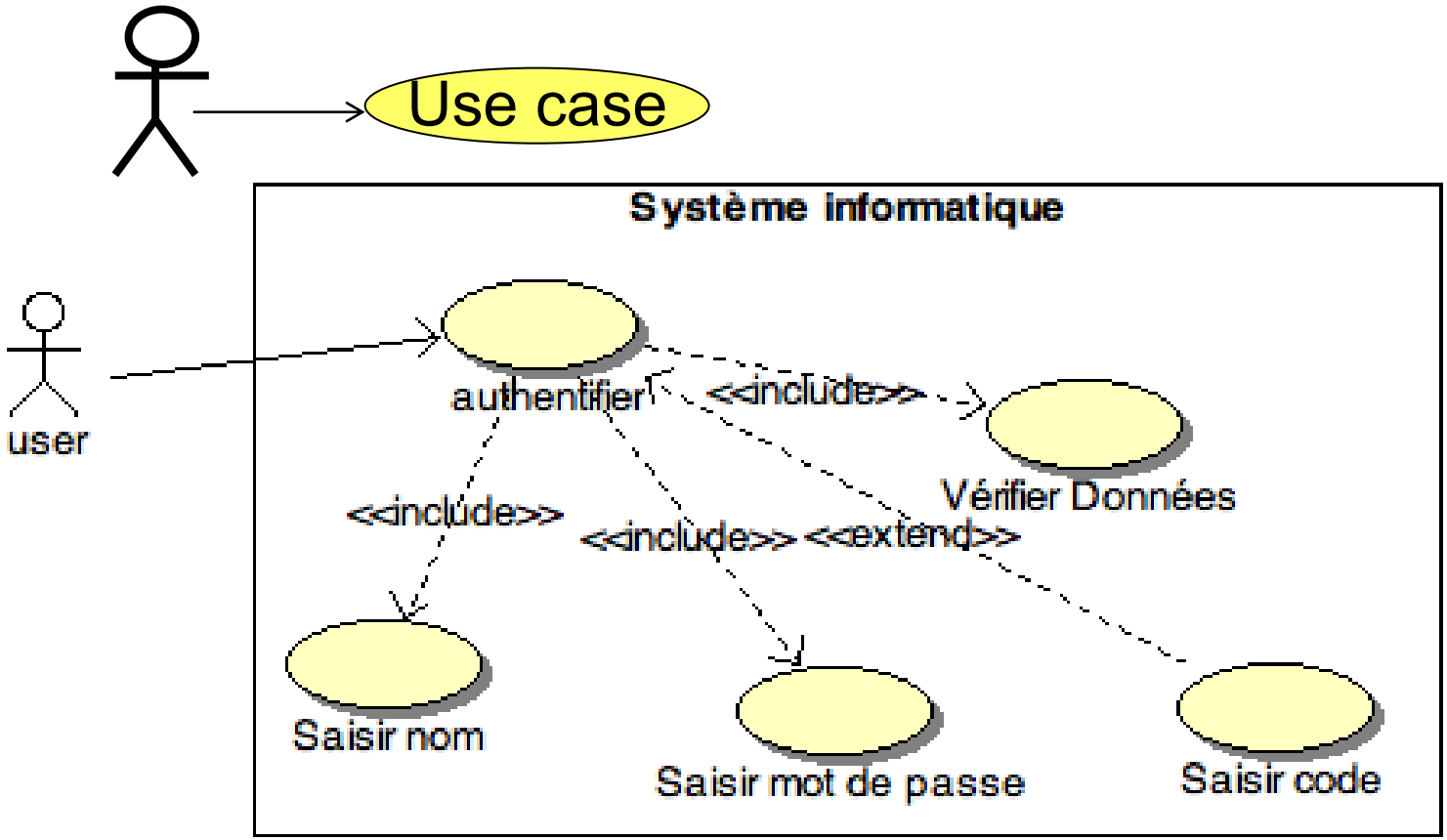


Diagramme de séquence

Objet/fil de vie/fin de vie/loop/opt
message/synchrone/asynchrone/retour

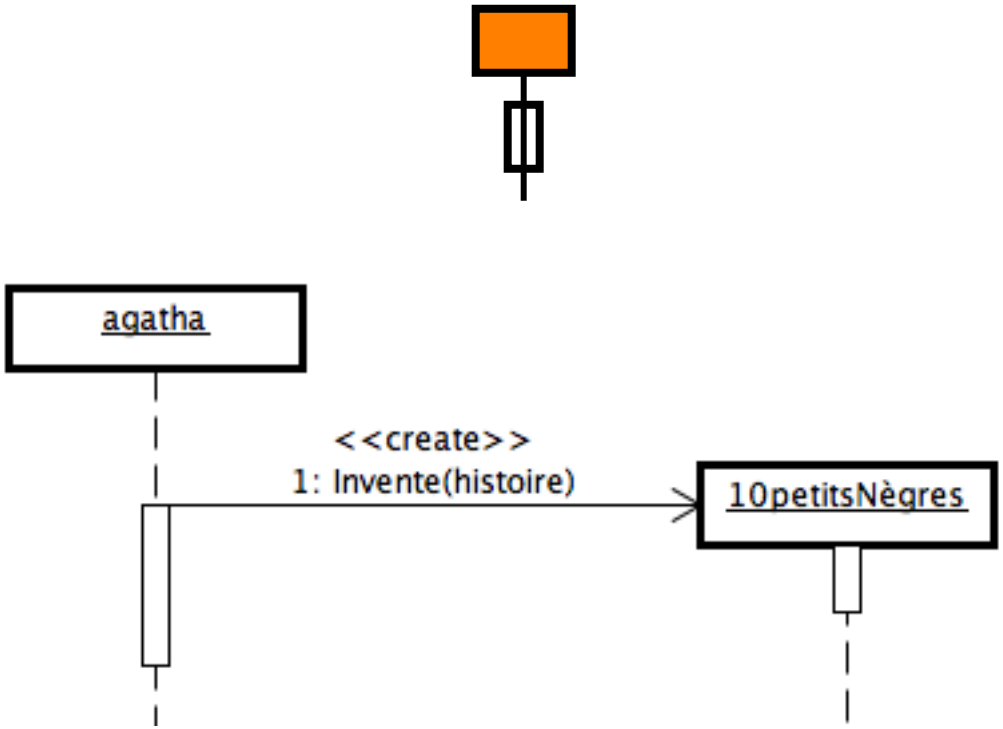
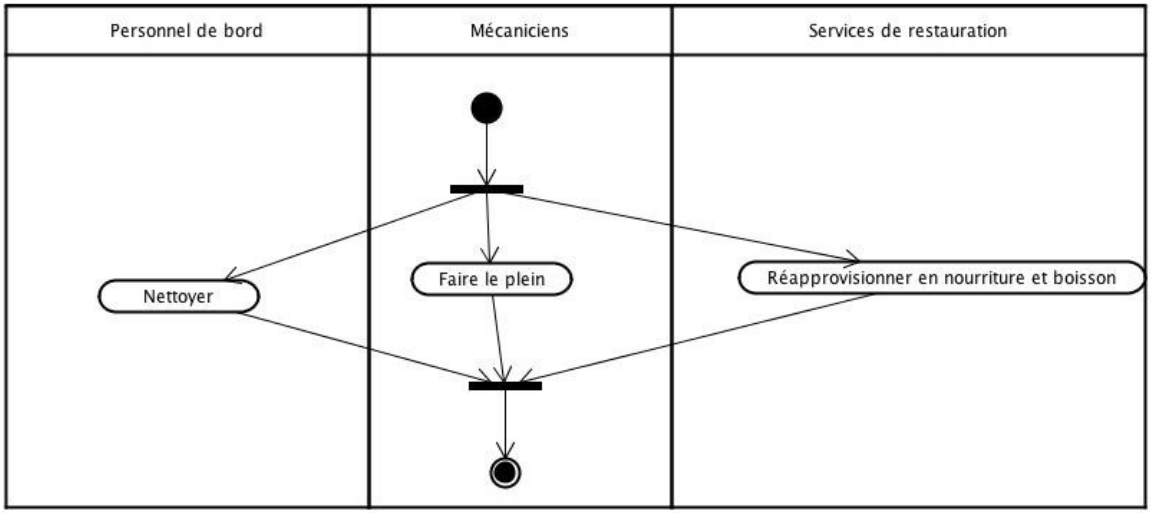
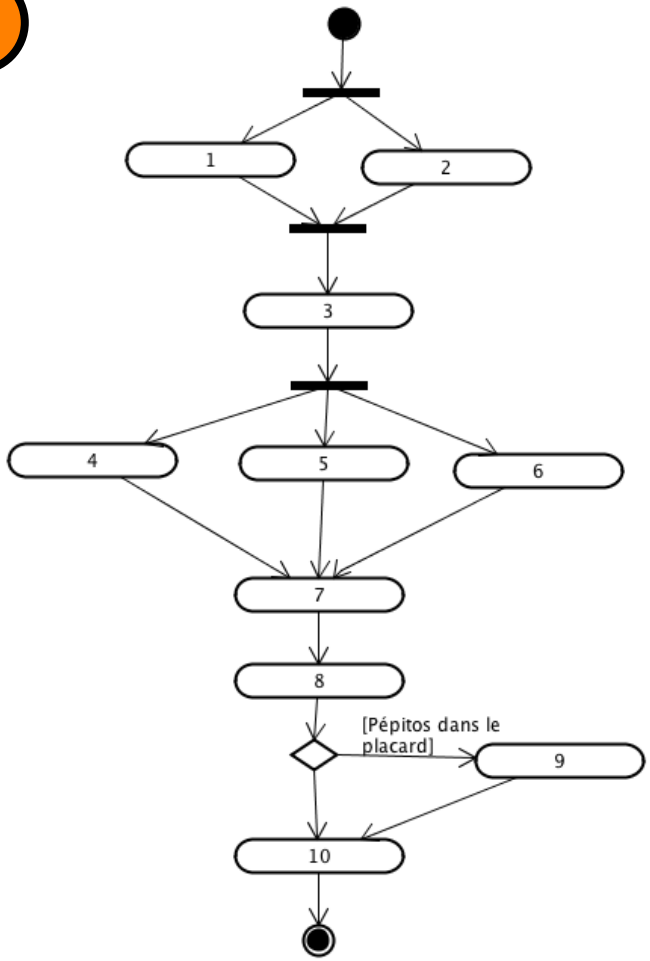


Diagramme d'activités

Départ/activité/fin/embranchement/garde/fourche/synchronisation

Couloir/responsabilité





Avec quels outils

1. Du papier et un crayon
2. Sinon…
   - UMLet https://www.youtube.com/watch?v=3UHZedDtr28
   - BOUML
   - ArgoUML
   - plantUML : https://github.com/plantuml/plantuml
   - MermaidJS

Références UML

https://laurent-audibert.developpez.com/Cours-UML/

Pascal Roques. **UML2 par la pratique (étude de cas et exercices corrigés)**. Eyrolles, 5e édition, 2006.

James Rumbaugh, Ivar Jacobson, and Grady Booch. **UML 2.0 Guide de référence**. CampusPress, 2004.

DÉCRIRE LES CAS D'UTILISATION

Avec UML

Mardi 30 Aout - 14H45-15h15

Capturer les besoins des utilisateurs

Un SI doit répondre aux besoins des utilisateurs. Leur capture nécessite :

- l'étude du système existant
- l'acquisition des nouveaux besoins
  - Fonctionnels : que doit faire le futur SI ?
  - Non fonctionnels : performances, sécurité, etc.
  - Utilisabilité : contexte de l'usage
- L'utilisation de techniques d'acquisition
  - Analyser la documentation existante
  - Questionner, observer les utilisateurs

La vue des cas d’utilisation

1. Cette vue définit les besoins des clients du système à l'aide de scénarios et de cas d'utilisation
2. Elle justifie l'architecture du système sur la satisfaction (la réalisation) de ces besoins.
3. Elle est prioritaire et accompagne toutes les autres vues pour leur donner leur cohérence.

Décrire les besoins des utilisateurs

Spécifier avec UML avec les diagrammes de cas d'utilisation pour modéliser et documenter les besoins

- Un cas d'utilisation n'est pas un besoin, mais une fonctionnalité du système devant répondre à un besoin
- Les diagrammes de cas d'utilisation ne permettent pas de répondre à des besoins non fonctionnels
- Il est nécessaire d'établir la liste des besoins non-fonctionnels pour compléter la spécification des besoins

Diagramme de cas d'utilisation (*use case*)

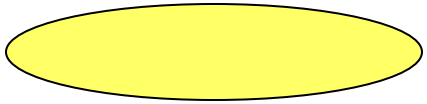
1. Formalisé par Ivar Jacobson
2. Décrivent sous la forme d'actions et de réactions le comportement d'un système du *point de vue utilisateur*
3. Définissent les *limites du système* et les relations entre celui-ci et son environnement.
4. Manière spécifique d'utiliser un système : image d'une fonctionnalité du système déclenchée en réponse à la *stimulation d'un acteur externe*.

Exemple de besoin

Gestion des malades dans un cabinet médical

1. Un médecin doit pouvoir
   - Créer des dossiers médicaux des patients, les modifier, consulter, supprimer.
   - Créer des ordonnances, modifier, imprimer et supprimer
   - Créer et modifier des instructions à l'attention du secrétariat
   - Créer et modifier les informations sur le patient (informations non médicales)
2. Une secrétaire doit pouvoir
   - Créer et modifier des informations sur un patient (si informations non médicales)
   - Editer une feuille de soin
   - Imprimé un récépissé lors de l'encaissement d'un paiement
   - Etc.

Diagramme d'utilisation : concepts

- Il comprend les acteurs, le système et les cas d'utilisation eux-mêmes.
- Les acteurs  déclenchent des cas d'utilisation 
- les cas d'utilisation sont contenus dans le système

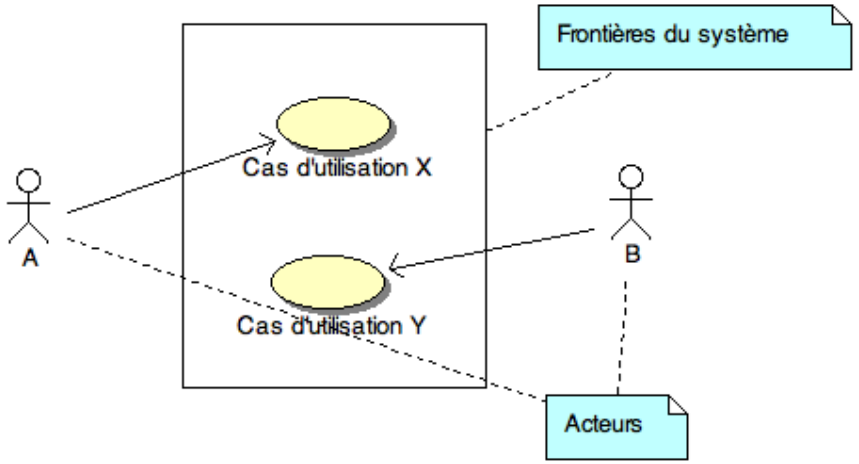


Diagramme d'utilisation : l'acteur

1. Un acteur représente tout ce qui est *externe au système*, humain ou non, qui *interagit* avec le système et qui correspond à une catégorie d'utilisateurs (plus précisément à un *rôle*).
2. Les acteurs se déterminent en observant les utilisateurs directs du système ainsi que les autres systèmes interagissant avec le système en question
3. La même personne physique peut jouer le rôle de plusieurs acteurs (vendeur, client). Plusieurs personnes peuvent jouer le même rôle et donc agir comme le même acteur (tous les clients). Le nom de l'acteur décrit le rôle joué par l'acteur.

Diagramme d'utilisation : l'acteur

Dans le cas du cabinet médical, nous avons 2 acteurs :

- la médecin
- le secrétaire

Diagramme d'utilisation : les cas d'utilisation

Un cas d'utilisation

- représenté graphiquement par une ellipse avec le nom du cas en dessous
- décrit une séquence d'action effectuée par le système pour livrer un résultat à l'acteur

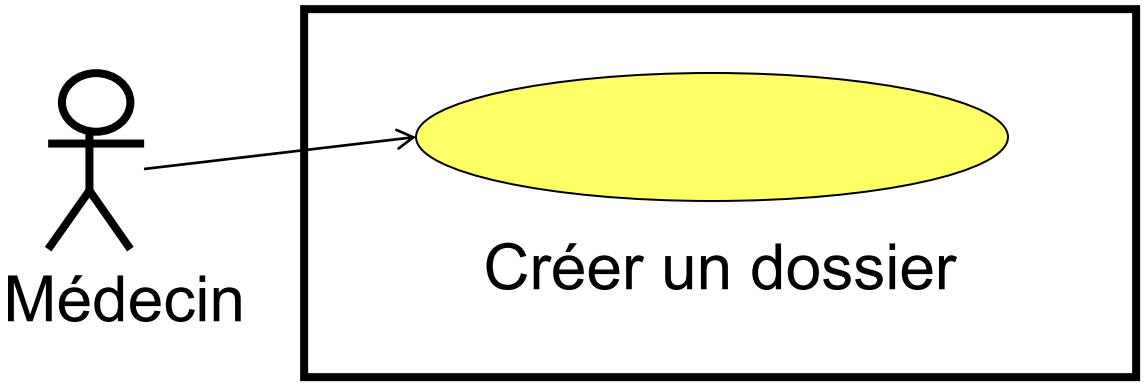


Diagramme d'utilisation : les cas d'utilisation

- Les cas d'utilisation se déterminent en observant et en précisant, acteur par acteur, les séquences d'interaction - les scénarios - du point de vue des l'utilisateur.
- La participation de l'acteur est signalée par un lien de communication entre l'acteur et le cas d'utilisation. Ce lien peut être orienté pour signaler l'initiateur de l'interaction.

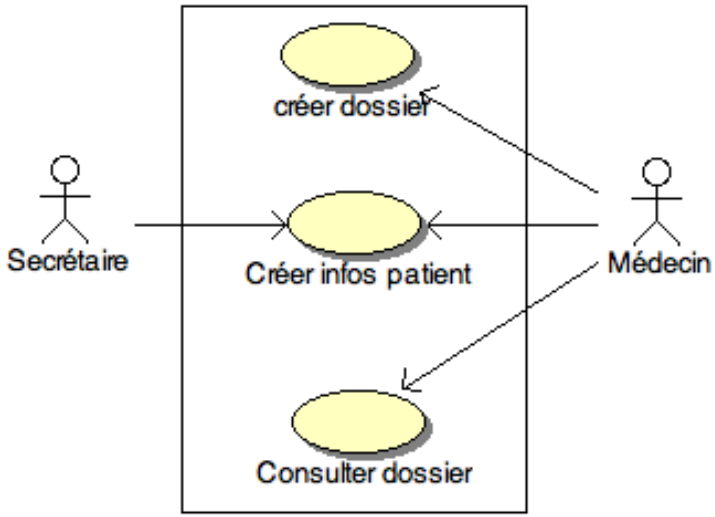
Diagramme d'utilisation : les cas d'utilisation

1. Les cas d'utilisations doivent être vus comme des classes dont les instances sont des *scénarios*. Chaque fois qu'un acteur interagit avec le système, le cas d'utilisation instancie un scénario; ce scénario correspond au flot de messages échangés par les objets durant cette interaction.
2. Les cas d'utilisation servent de fil conducteur pour l'ensemble du projet

Le modèle de cas d'utilisation

Le **modèle de cas d'utilisation**

- Comprend une collection de cas d'utilisation
- Caractérise le comportement de l'ensemble du système et des acteurs externes dans leurs interactions.

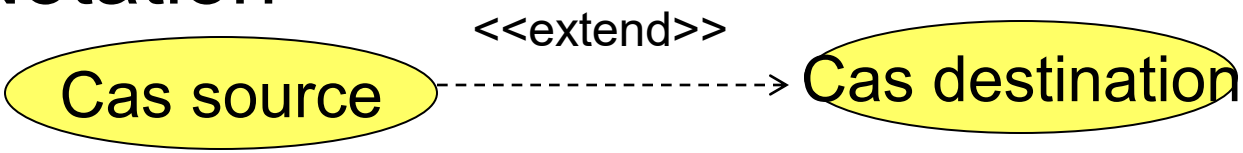


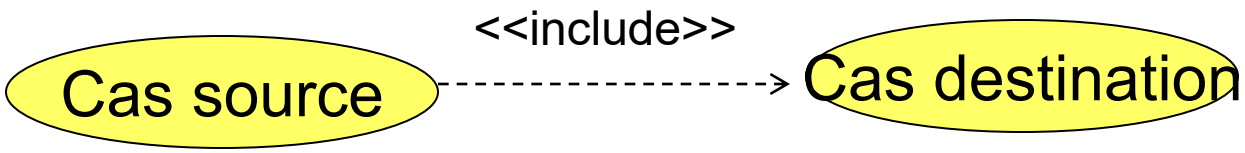
Raffinement des cas d'utilisation

Relations entre cas d'utilisation

- Extend : stéorotype <<extend>>
- Include : stéorotype <<include>>

Notation





La relation « *include* »

L'inclusion entre 2 cas d'utilisation signifie que toute instance du cas source comprend nécessairement le cas destination.

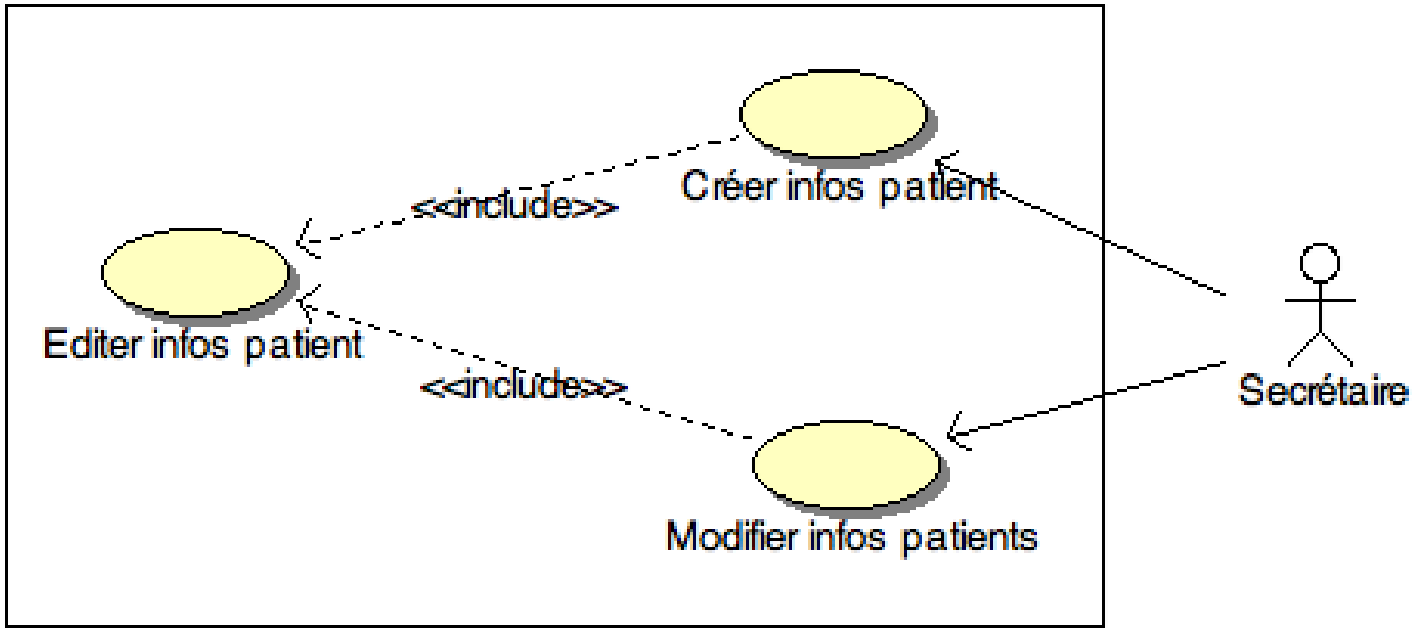
Quand l'utiliser ?

Lorsqu'on veut réutiliser un ensemble d'interactions communes dans plusieurs cas d'utilisation : on regroupe alors les interactions communes dans un cas d'utilisation (destination), lié aux cas d'utilisation qui l'emploient (les cas sources)

Avantage : réutilisation

La relation « *include* »

Exemple



•2008

La relation « *extend* »

L'extension entre 2 cas d'utilisation signifie que le cas source étend le comportement du cas destination

Quand l'utiliser ?

- Lorsque l'on veut enrichir un cas de base avec une possibilité de fonctionnement non décrit dans le cas source.
- Cette extension est décrite dans un cas à part, et n'a pas de sens SEULE (sans son cas source).
- Les 2 cas sont ensuite reliées par la relation d'extension

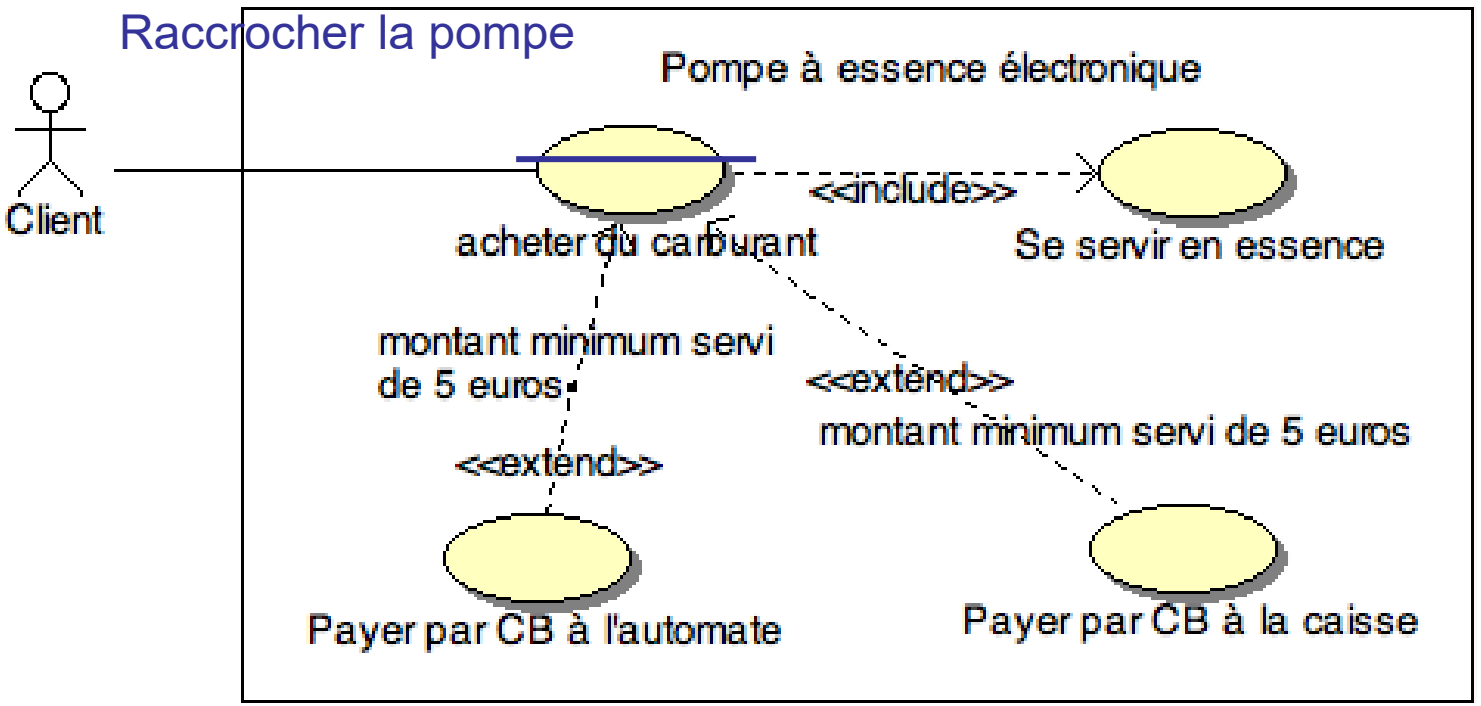
Les points d'extension montrent à quel moment survient l'extension

Une condition peut-être rajoutée à la relation d'extension pour la préciser.

La relation « *extend* »

Exemple : pompe à essence

Point d'extension :
Raccrocher la pompe



Scénarios des cas d'utilisation

- Un cas d'utilisation
  - Est un ensemble complet d'actions (avec des évènements de début et de fin, les acteurs impliqués dans les actions) et de règles qui régissent l'enchaînement des actions.
  - Il définit les interactions entre le système et les acteurs
  - Il inclut le déroulement normal et tous les déroulements alternatifs
- Un scénario
  - Est un *déroulement spécifique* des évènements. Ce déroulement dépend des évènements à l'origine et à l'issue de chaque action spécifié dans le cas d'utilisation et dont dépend l'enchaînement des actions.
  - Il est possible de dériver plusieurs scénarios à partir d'un cas d'utilisation. Il suffit pour cela que l'enchaînement des actions ne soit pas une simple séquence.

Description des cas d'utilisation

La description peut-être sous une forme textuelle et peu structurée

Exemple :

« le médecin cherche le patient dans la liste des patients. Si celui-ci n'existe pas il le crée, sinon il crée son dossier. Le médecin introduit les informations sur les antécédents du patient, ses allergies, la liste des substances auxquelles il est allergique, la liste des traitements qu'il suit et la durée de ces traitements.

Etc. »

Description des cas d'utilisation

Elle peut être décomposée en précisant les interactions entre le système et l'acteur en distinguant le déroulement de base des déroulements alternatifs.

Créer ordonnance

Acteur	**Réponse du système**
1. Le médecin demande à créer une ordonnance 3. Le médecin choisit le patient souhaité 5. Le médecin sélectionne un médicament dans une liste …	2. Le système lui demande de choisir un patient 4. Le système édite une ordonnance vierge 6. Le système demande à saisir les doses et la fréquence des prises
Déroulement alternatif 5-6 Le médecin entre le nom du médicament, et la posologie	

Description des cas d'utilisation

La description d'un cas d'utilisation comprend les éléments suivants :

- Le *début* : « le cas débute quand X se produit »
- La *fin* : « le cas se termine quand Y s'est produit »
- *L'interaction* entre le cas et les acteurs
- Les échanges *d'informations* : par exemple, « l'utilisateur se connecte au système et donne son nom et son mot de passe »
- La *chronologie* et l'origine des informations
- Les *répétitions de comportement* qui peuvent être décrites au moyen de pseudo-code, avec des constructions du type :

```
Boucle
…faire quelque chose
Fin de boucle
```

```
Tant que
…faire quelque chose
Fin tant que
```

Règles de mise en œuvre des cas d'utilisation

- Un cas d'utilisation décrit une fonctionnalité ou une motivation et aussi une interaction entre un acteur et un système sous la forme d'un flot d'évènements
- La description de l'interaction se concentre sur ce qui doit être fait
- Un cas d'utilisation doit être simple
- Il ne faut pas mélanger les cas d'utilisation
- Un cas d'utilisation doit éviter d'employer des expressions floues et imprécises.

Règles de mise en œuvre des cas d'utilisation

Il est primordial de trouver le *bon niveau d'abstraction*. Les réponses apportées aux 2 interrogations suivantes peuvent servir de gabarit :

- Est-il possible d'exécuter une activité donnée indépendamment des autres, ou faut-il toujours l'enchaîner avec une autre activité :
  - « passer un appel téléphonique »,
  - « composer un numéro de téléphone »
- Est-il judicieux de regrouper certaines activités en vue de les documenter, tester ou modifier ?

Construction des cas d'utilisation

- En règle générale, il n'y a qu'un acteur par cas d'utilisation
- Lors de la construction, il faut se demander :
  - Quelles sont les tâches de l'acteur
  - Quelles informations l'acteur doit-il créer, sauvegarder, modifier, détruire ou lire ?
  - L'acteur devra-t-il informer le système des changements externes
  - Le système devra-t-il informer l'acteur des conditions internes
- Les cas d'utilisation peuvent :
  - être présentés au travers de vues multiples
  - être groupés selon leurs séquences de déclenchement types ou en fonction des différents points de vue

Processus d'élaboration des cas d'utilisation

- Définir un guide de style pour la rédaction
- Définir grossièrement les cas d'utilisation
- Approfondir la compréhension et la description d'un cas d'utilisation particulier. Identifier les scénarios
- Un scénario est un chemin particulier au travers de la description abstraite et générale fournie par le cas d'utilisation

UN EXERCICE USE-CASE

1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

DIAGRAMMES D'ACTIVITÉ

Analyser les processus

Mardi 08H30-10H00 : Cours : Diagramme activité / séquence (Christine Plumejeaud)

Les diagrammes d'activité

Le diagramme d'activité est utilisé pour

- Modéliser une tâche (dans la modélisation métier)
- Décrire une fonctionnalité du système modélisée par un diagramme d'utilisation
- Décrire la logique d'une *opération* ou d'un algorithme.
- Décrire les activités et leur enchaînement dans un processus

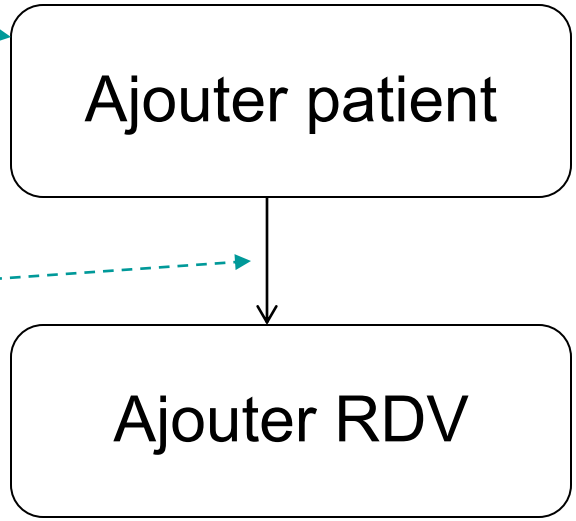
Le diagramme d'activités : concepts

Diagramme d'activités =

- Ensemble d'activités liées par :
  - Transitions (séquentielle)
  - Transitions alternatives (conditionnelle)
  - Synchronisation (disjonction et conjonction d'activités)
- Deux états : initial et final
- Couloirs d'activité : représente le responsable de l'activité.

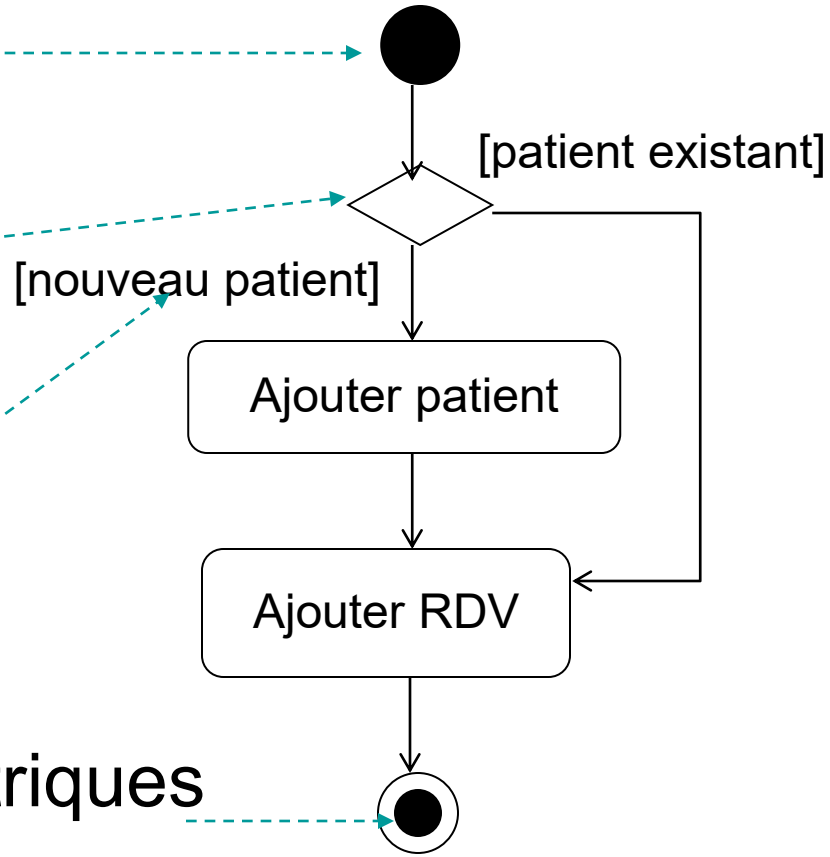
Notations du diagramme d'activité

- Activités
  - Un rectangle arrondi
  - Un nom significatif
- Transitions
  - flèche



Notations du diagramme d'activité

- Etat initial
  - Cercle noirci
- Point de décision
  - Losange
- Condition (garde)
  - Entre crochets
- Etat final
  - Deux cerce concentriques dont un noir

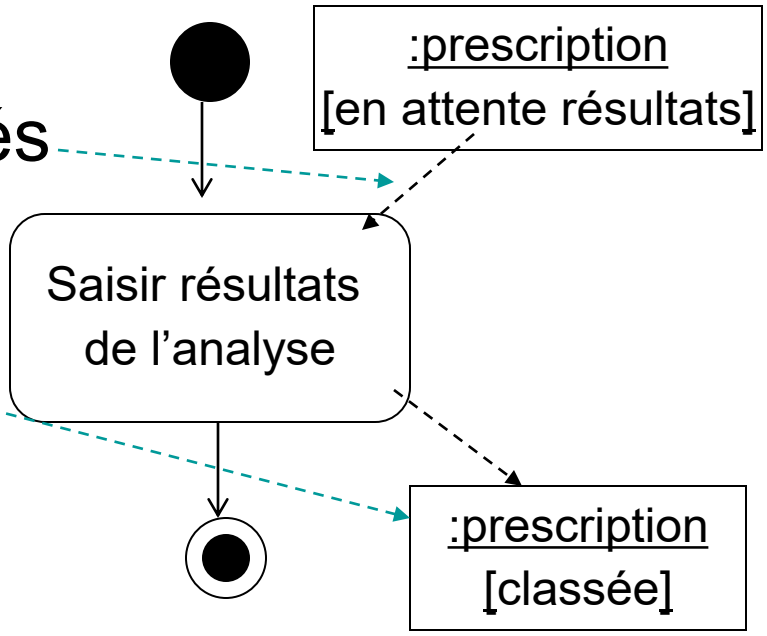


Notations du diagramme d'activité

- Flot d'objets
  - Des flèches en pointillés
- Objets
  - Rectangle
  - Le nom de l'objet

  est souligné
  - On peut montrer l'état

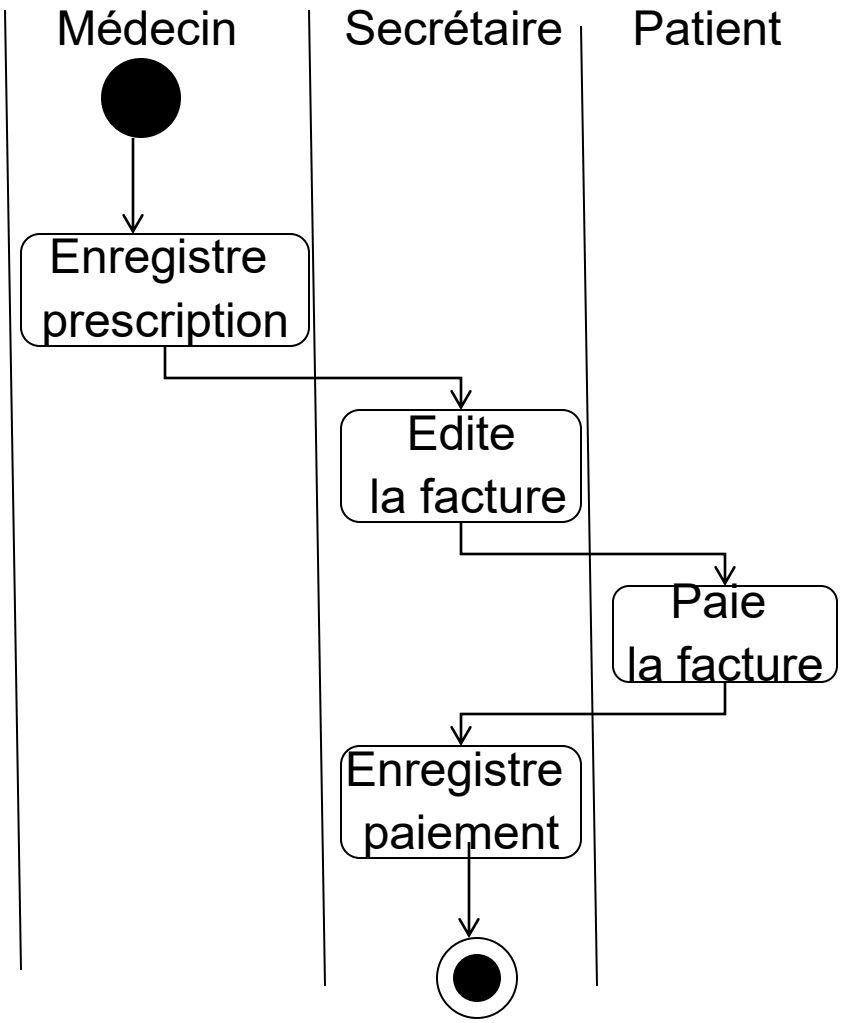
  de l'objet (noté entre crochets)



Notations du diagramme d'activité

1. Couloirs d'activité
   - Colonnes verticales
   - Etiquetés des noms

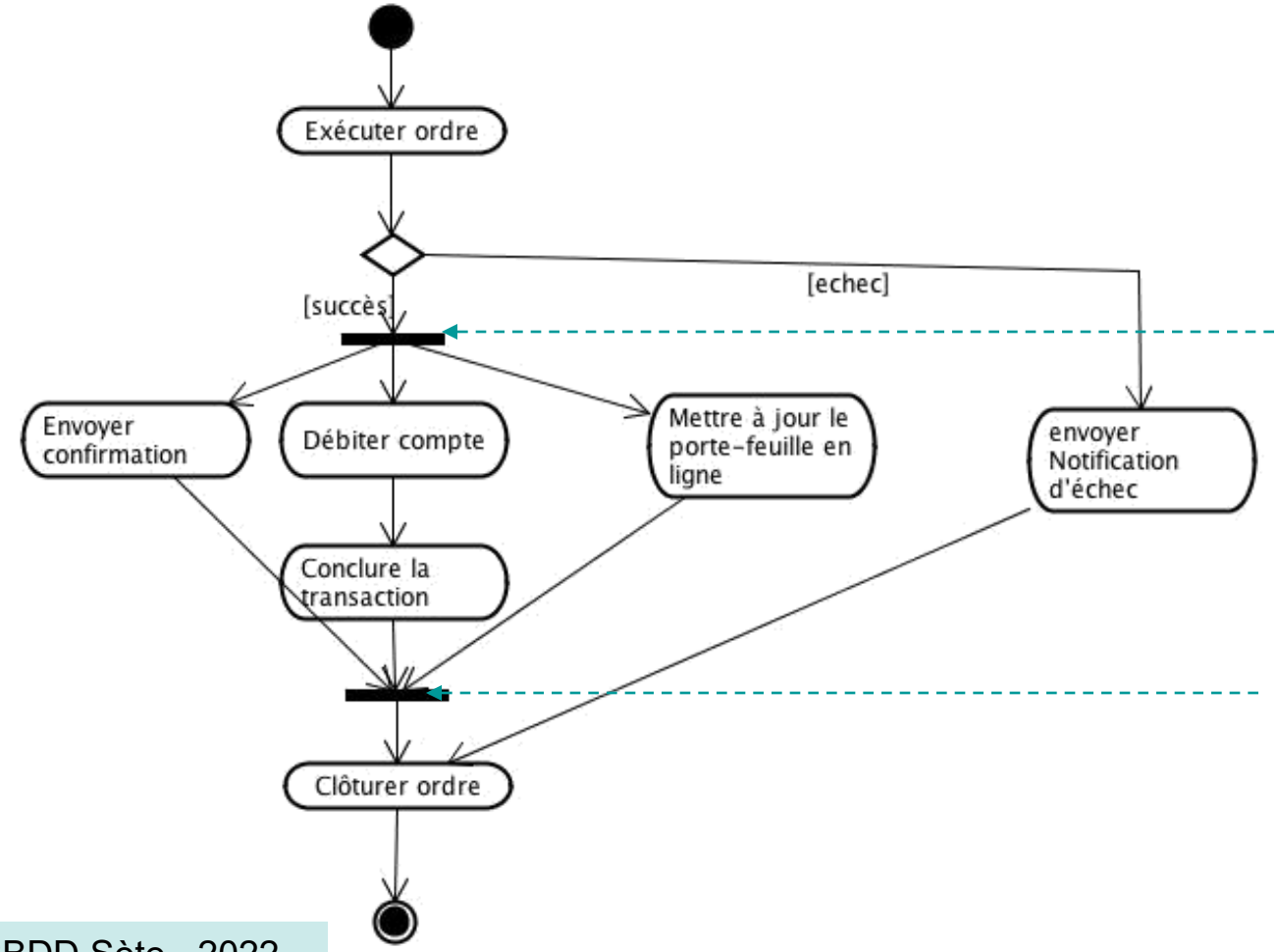
de personnes,
département, etc.,
responsables
de cette activité



Référence :
https://fr.acervolima.com/breve-note-sur-le-diagramme-d-activite-et-de-couloir/

Notations du diagramme d'activité

Fils d'exécutions concurrents

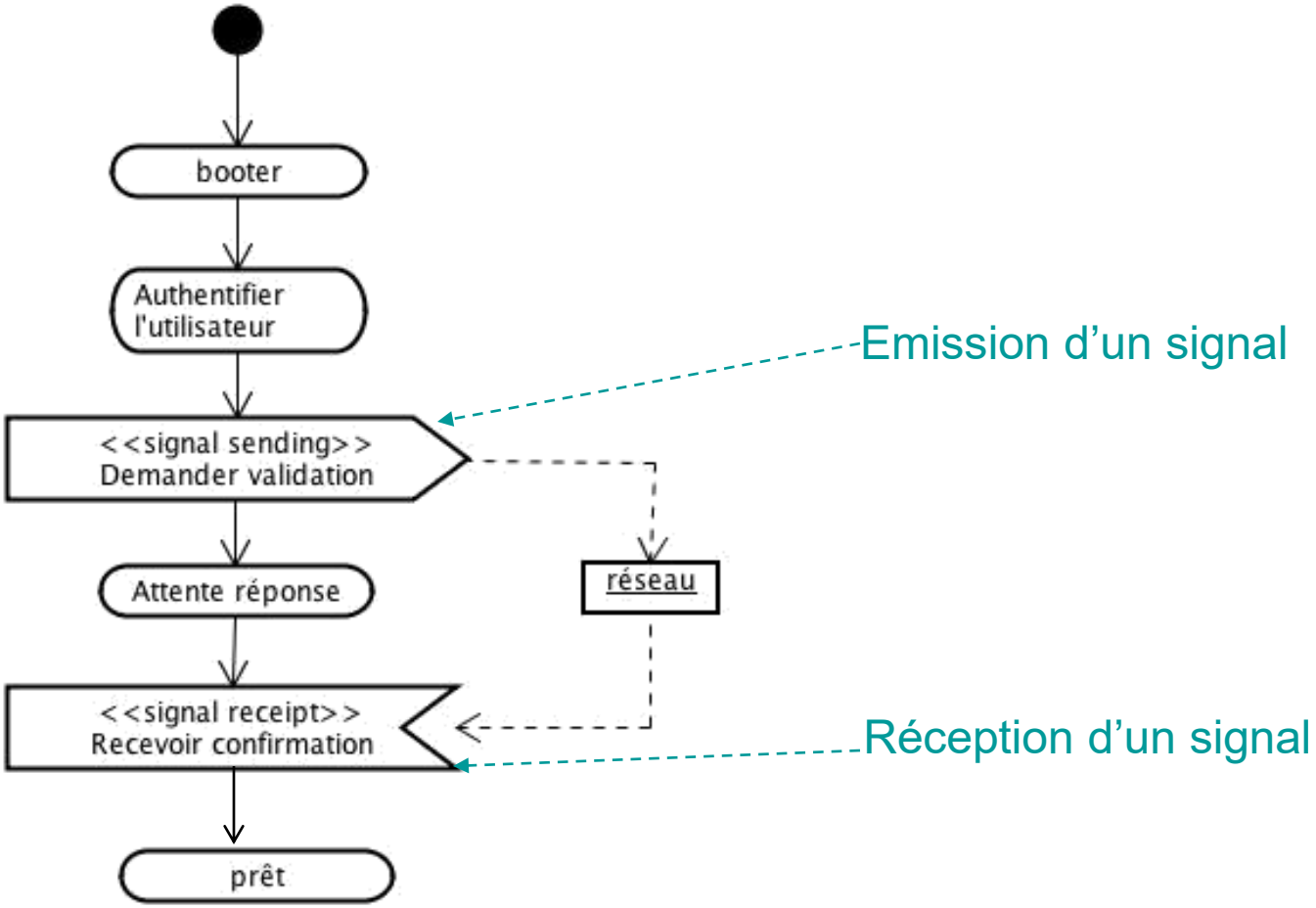


Barre de parallélisation

Barre de synchronisation

Diagramme d'activité

Emission et réception de signaux



Construire un diagramme d'activité

1. Identifier les activités
   - Qu'arrive-t-il lorsqu'un médecin souhaite saisir une prescription ?
     - Créer dossier médical du patient
     - Créer un dossier pour les antécédents
     - Saisir la prescription
2. Ordonner les activités en utilisant les transitions

Attention, le niveau de détails des activités d'un même diagramme doit être identique.

Diagramme d'activité : exemple

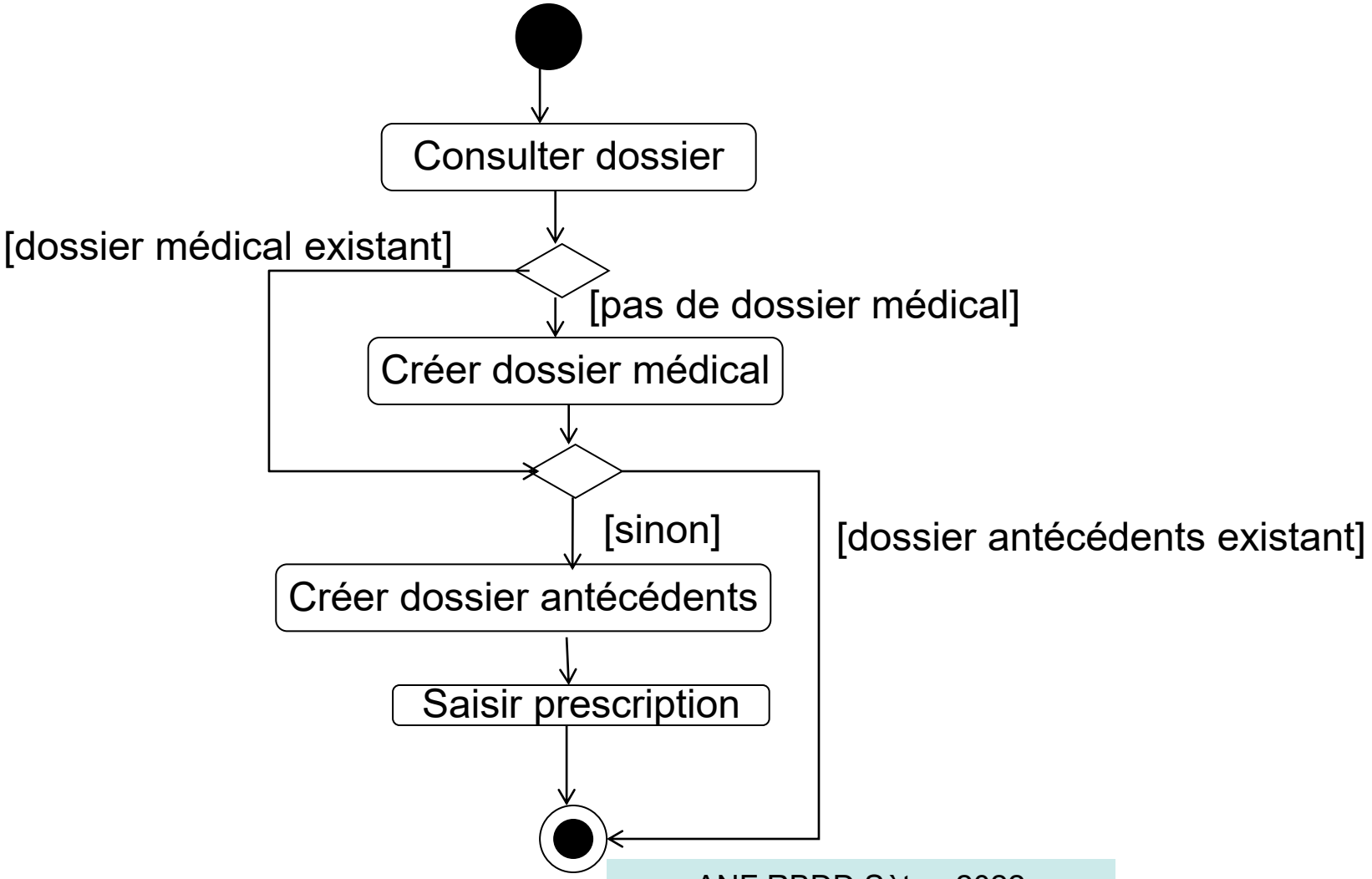


Diagramme d'activité : conseils

1. L'objectif ?
   - Décrire un workflow, des opérations, un algorithme, un flux d'objets.
2. Les critères de qualité ?
   - Un niveau de détails équilibré dans chaque branche
   - Les branchements sont tous documentés et couvrent le domaine. Usage du [sinon]
3. Le niveau de détail souhaité ?
   - Haut niveau d'abstraction ou plutôt détaillé ?

Bilan diagramme d'activité

Nous avons vu :

- L'utilité d'un diagramme d'activité
- Quelles sont les notations associées à un diagramme d'activité
- Comment construire un diagramme d'activité

UN EXERCICE ACTIVITÉ

1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

INTERACTIONS ET SEQUENCES

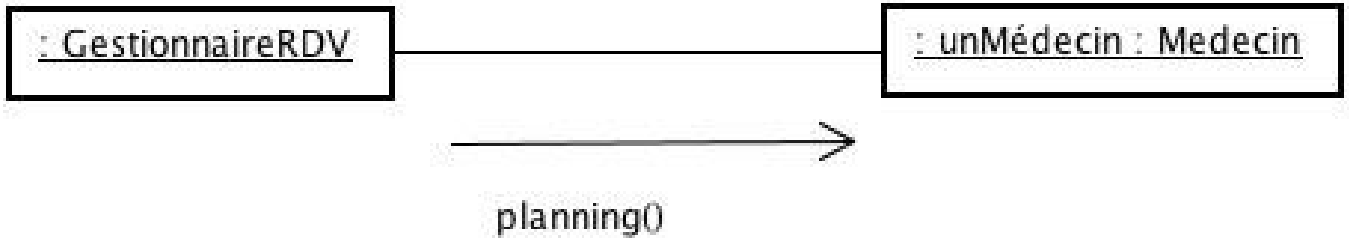
La communication dans un SI : déroulement temporel des actions.

Les interactions entre objets

1. L'envoi de message
   - Les objets communiquent par envoi de messages

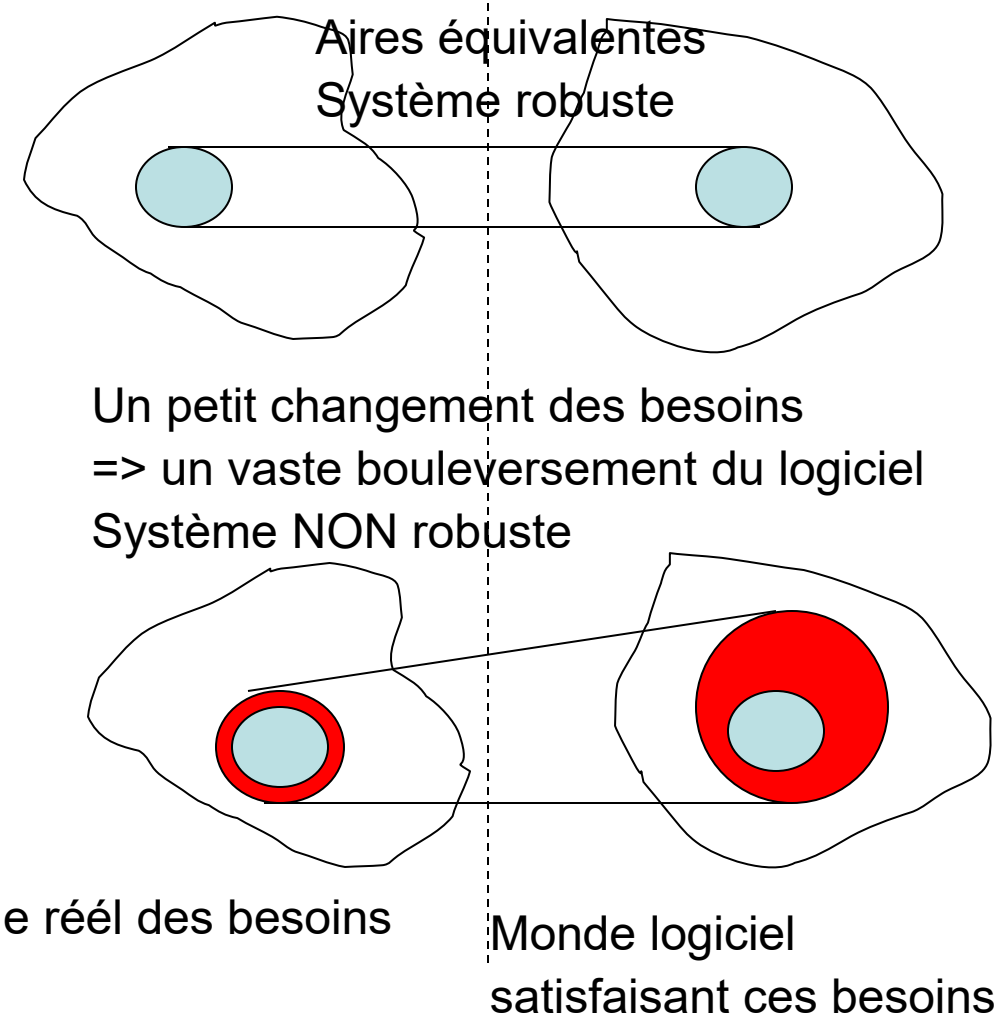
     Envoyer le message **`fournir_planning()`** à un objet **`UnMédecin`**, doit utiliser la syntaxe suivante :

     planning = UnMédecin.planning()



L'envoi de messages

- Les objets communiquent à travers l'envoi de messages
- L'objectif des diagrammes d'interaction est de préciser les responsabilités des objets pour minimiser l'effet de bord résultant des changements des besoins.



Interaction et collaboration

- On appelle *collaboration* l'ensemble formé par:
  - la structure des instances participant à un comportement
  - les liens qu'elles entretiennent.
- Une *interaction* est définie dans le contexte d'une collaboration.

  Elle spécifie le canevas de communication entre les rôles dans la collaboration. Elle contient un ensemble de *messages* partiellement ordonné. Chaque message spécifie une communication comme par exemple le signal à envoyer ou l'opération à invoquer, ainsi que les rôles joués par la source et la cible du message.

Diagramme de séquence

- Les interactions entre objets selon un point de vue *temporel*
- Le contexte des objets n'est pas représenté de façon explicite. La représentation se concentre sur l'expression des interactions
- Peuvent être utilisés à différentes phases du cycle de vie avec +/- de détails
- Peuvent être utilisés de façon :
  - Informelle : documentation des cas d'utilisation en illustrant les scénarios
  - Informatique : le concept de MESSAGE unifie toutes les formes de communication entre objets (appel de procédure, événement discret, signal entre flots d'exécution)

Diagramme de séquence

- La dimension verticale représente le *temps*
- Les *objets* impliqués sont disposés horizontalement. Ils sont représentés par des *lignes de vie* verticales
- Les *messages* sont représentés par des flèches pleines horizontales
- L'exécution d'une opération est matérialisée par une *activation*

Diagramme de séquence : notation

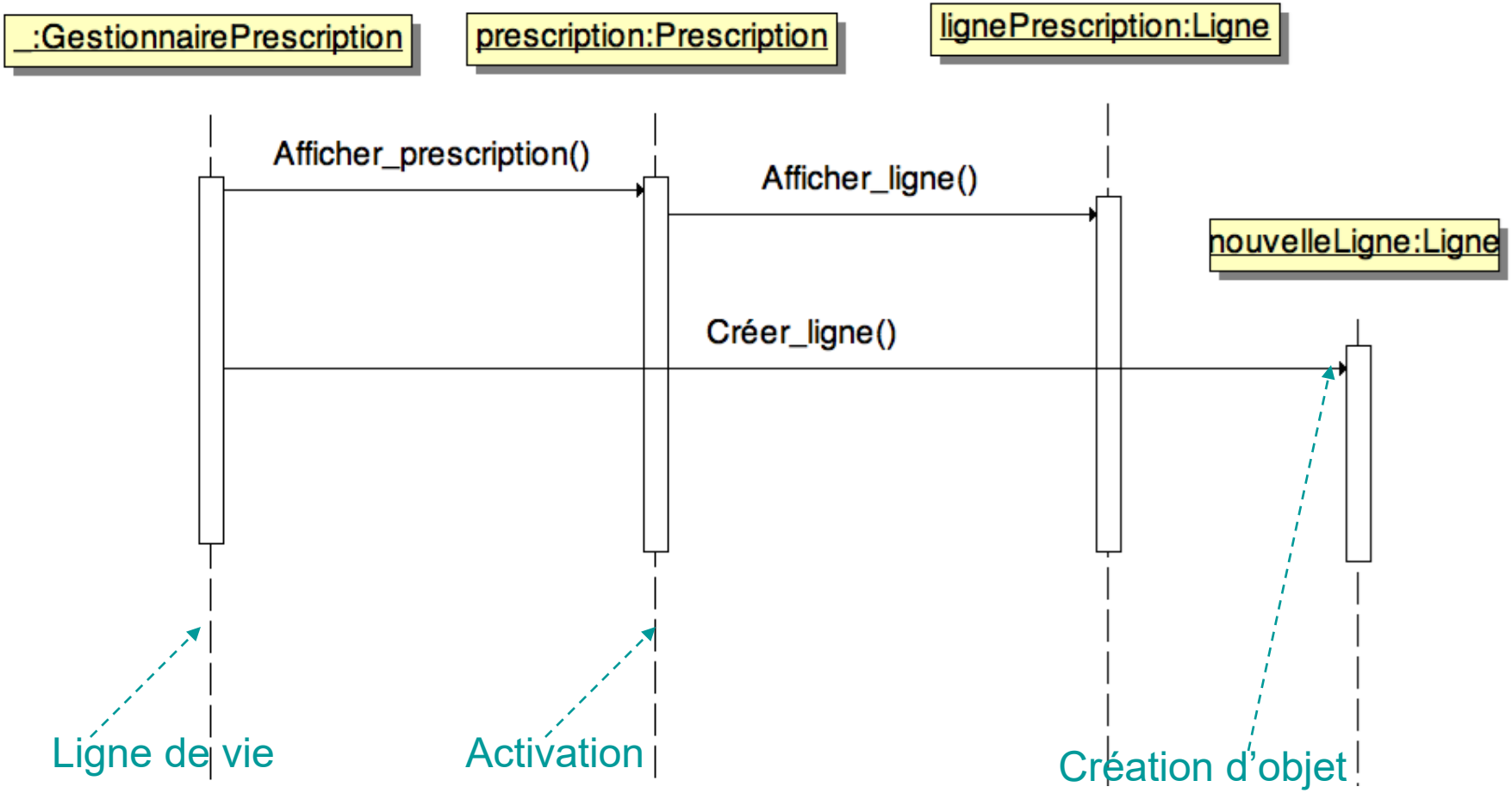


Diagramme de séquence : création/destruction d'objets

1. Création : pointer le message de création sur le rectangle symbolisant l'objet créé.
2. Destruction : fin de la ligne de vie, et un croix X.

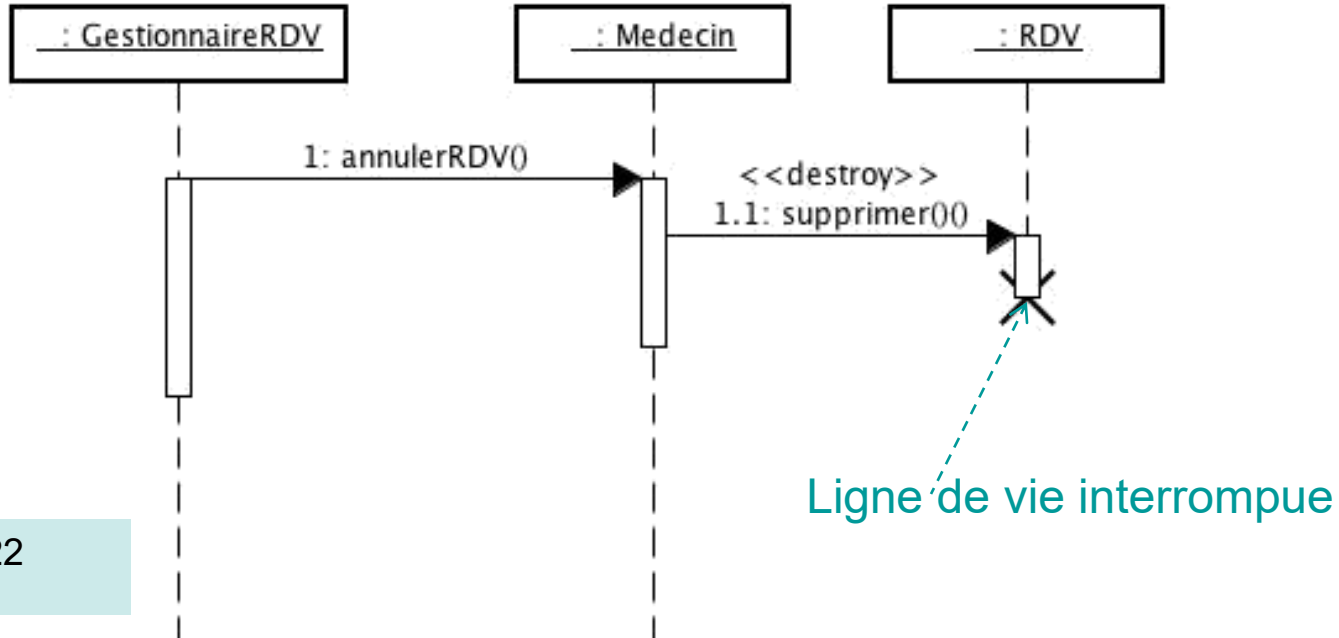


Diagramme de séquence : les différents types de message

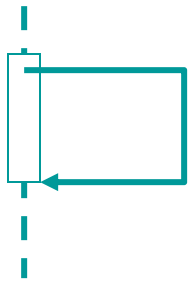
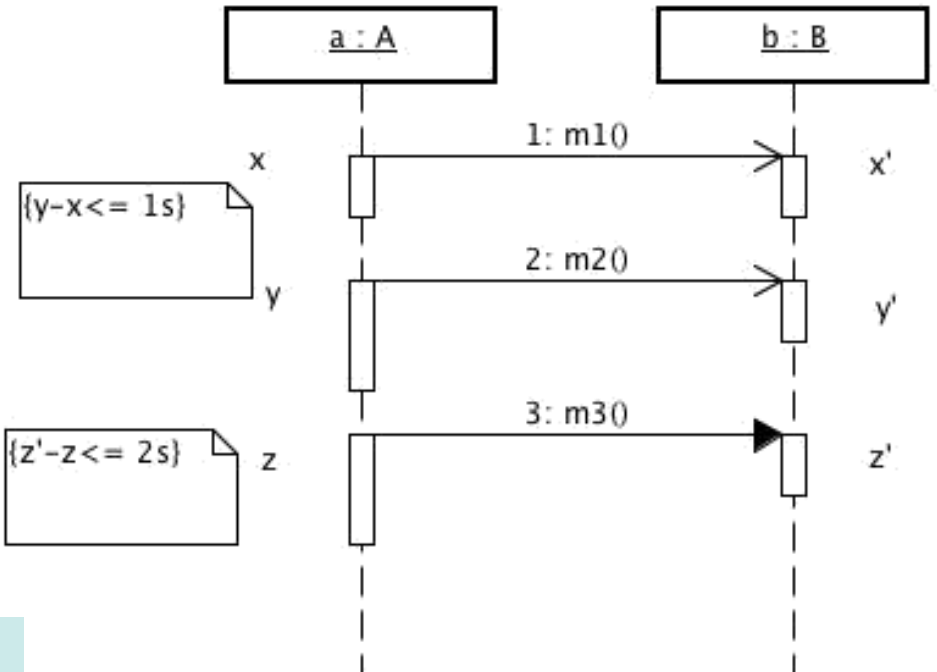
1. Message synchrone : l'appelant est bloqué en attente du retour de message 
2. Message asynchrone : l'appelant continu son flot d'exécution après l'émission. La réception se fait ultérieurement. 
3. Retour de procédure : le retours des messages sont le plus souvent implicite, mais parfois ils sont indiqués (cas asynchrone)
4. Message réflexif : les objets peuvent produire des messages pour eux-mêmes. 

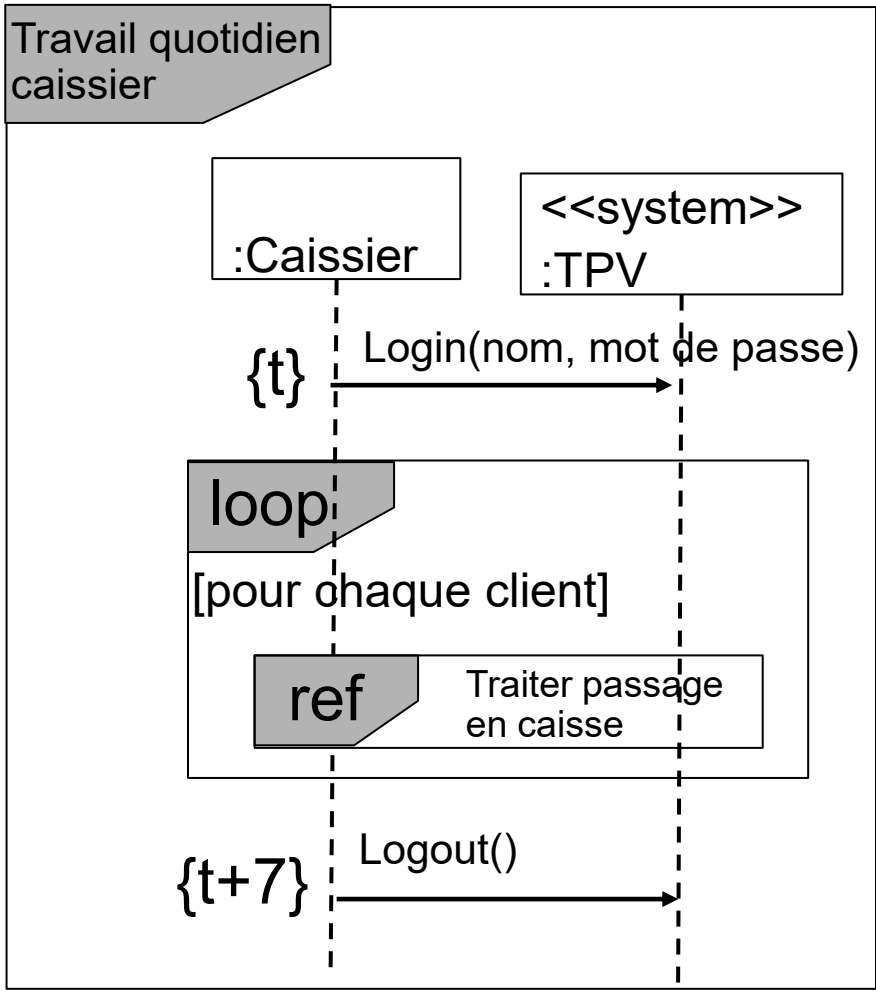
Diagramme de séquence : les contraintes temporelles

- Instant d'émission x
- Instant de réception x'
  - Exemples :
    - Il s'écoule moins d'une seconde entre l'émission de m1 et m2
    - Il s'écoule au moins 2 secondes entre l'envoi et la réception de m3

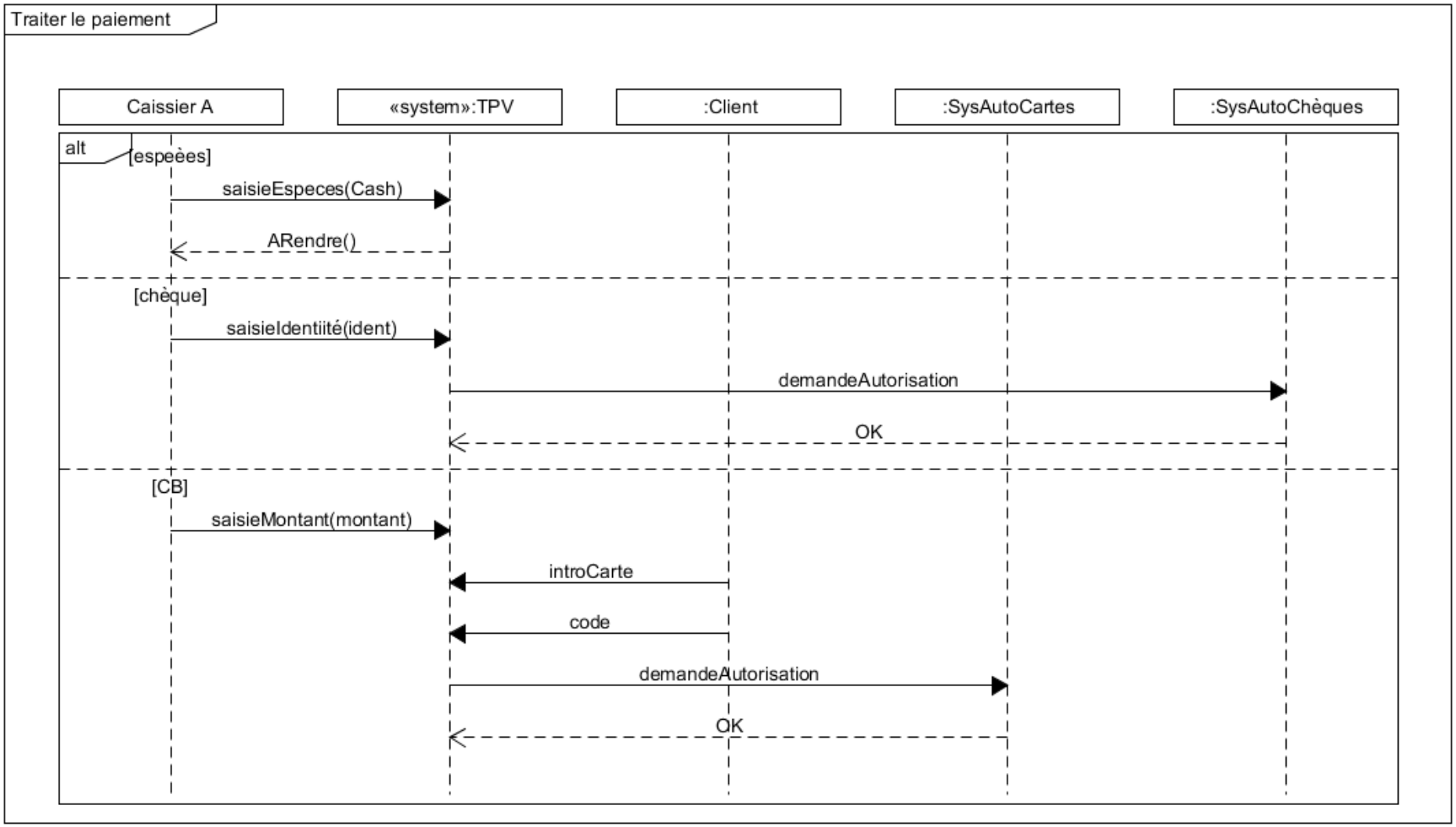


Les fragments : pour gérer la complexité

1. LOOP
   - Enchaînement qui se répète
2. OPT
   - Une action qui se fait sous condition
3. REF
   - Référence à un autre diagramme de séquence
4. ALT
   - 2 ou plus alternatives qui se réalisent
5. PAR
   - Plusieurs actions qui se déroulent en parallèle



Représentation des ALT ou PAR



Les diagrammes de séquence : conseils pour la description des besoins

1. Un scénario par cas d'utilisation : évitez les branchements conditionnels
2. Subdiviser les interactions complexes : décrivez dans un second diagramme une sous-tâche.
3. Un diagramme de séquence par condition d'erreur possible.

Diagramme de séquence : les branchements conditionnels

A émet un message

soit vers B si la condition 1 est remplie,

soit vers C si la condition 2 est remplie

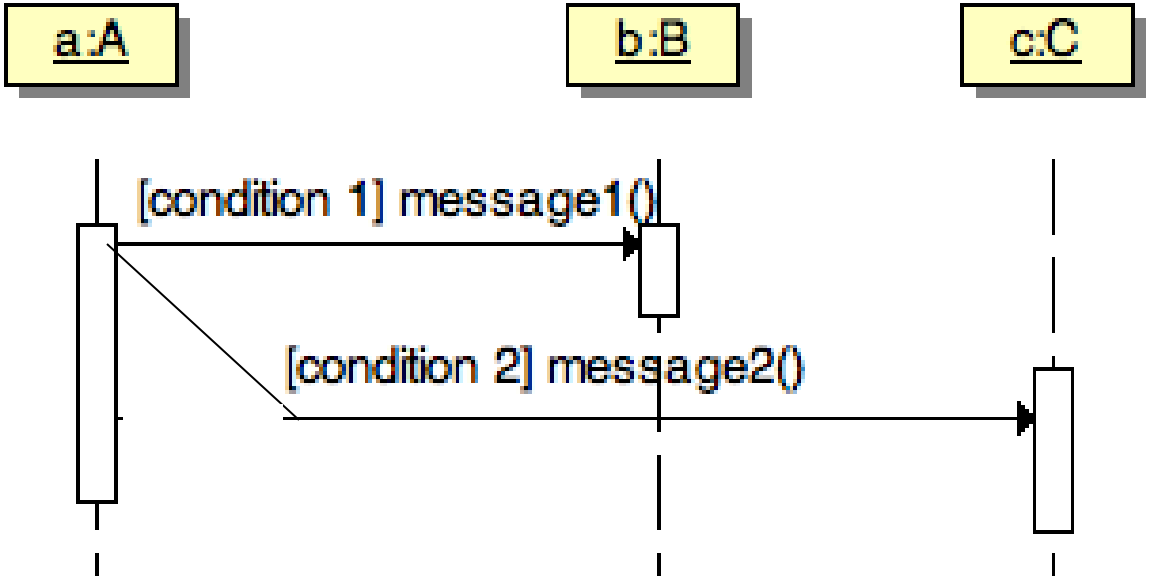


Diagramme de séquence : gérer la complexité

Les diagrammes complexes peuvent être décomposés en plusieurs diagrammes plus simples en précisant convenablement cette décompositions par des annotations.

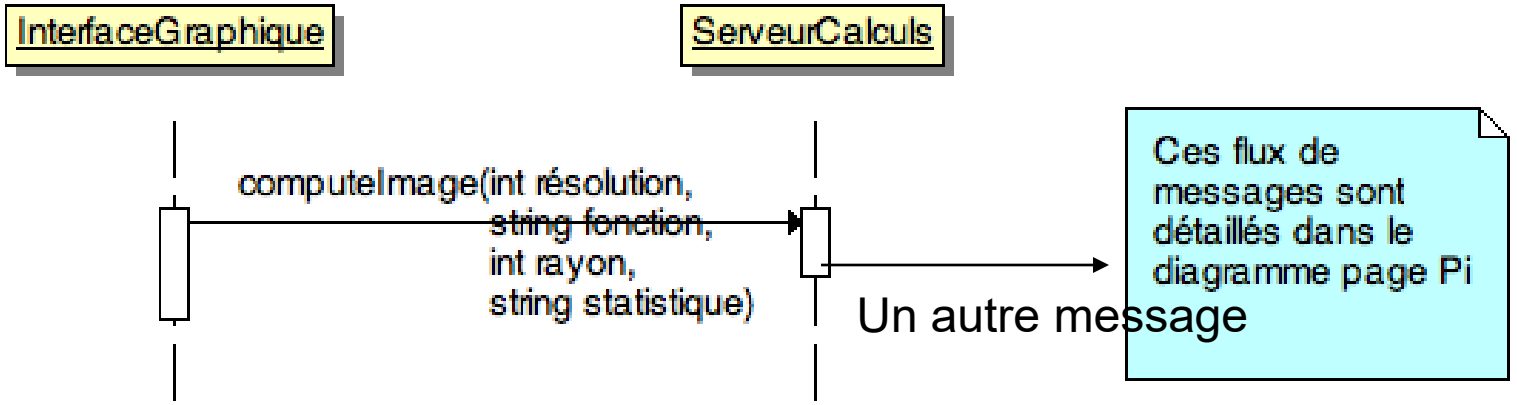
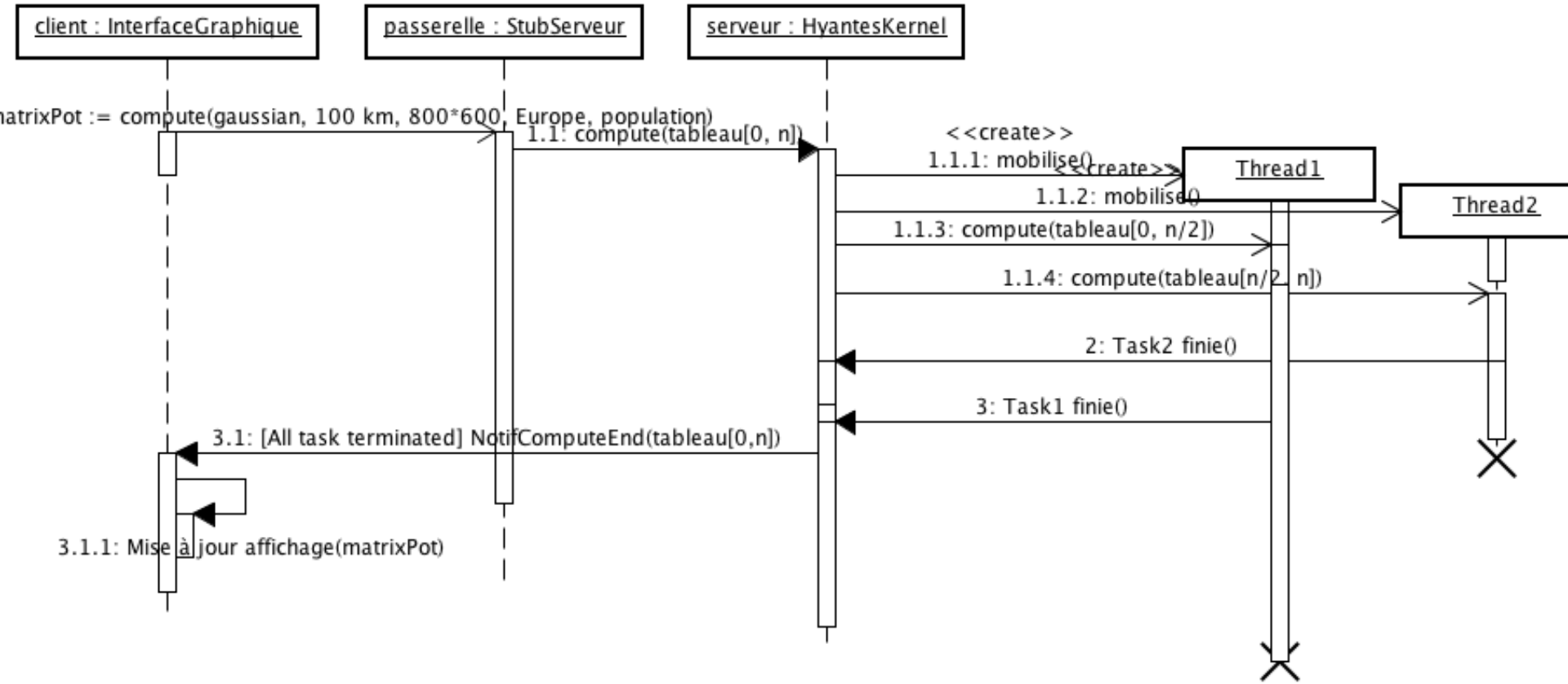


Diagramme de séquence

Exemples de messages synchrones/asynchrones



UN EXERCICE SÉQUENCE

1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

Etude de cas

Guichet Automatique de Billets (GAB)

Source : UML par la pratique,
P. Roques

Enoncé simplifié

1. Le GAB offre les services suivants :
   - Distribuer de l'argent à tout porteur de Carte Bancaire (CB) de type VISA ou de la banque
   - Pour un client porteur d'une carte de la banque : consulter le solde du compte, déposer des espèces ou des chèques
2. Il faut savoir que :
   - Toutes les transactions sont sécurisées
   - Il est nécessaire d'alimenter le distributeur en argent, papier pour l'impression des tickets, récupérer les chèques déposés, les cartes avalées, et les espèces déposées.

Identifier les acteurs du GAB

1. Un acteur représente une catégorie d'individus externes au système et qui ont le même comportement vis à vis de celui-ci : attendre un service
2. Un acteur peut-être un agent humain ou un autre système
3. Un acteur est dit *primaire* lorsqu'il est l'initiateur du cas d'utilisation. Il est dit *secondaire* lorsque le déroulement du cas d'utilisation nécessite son intervention.

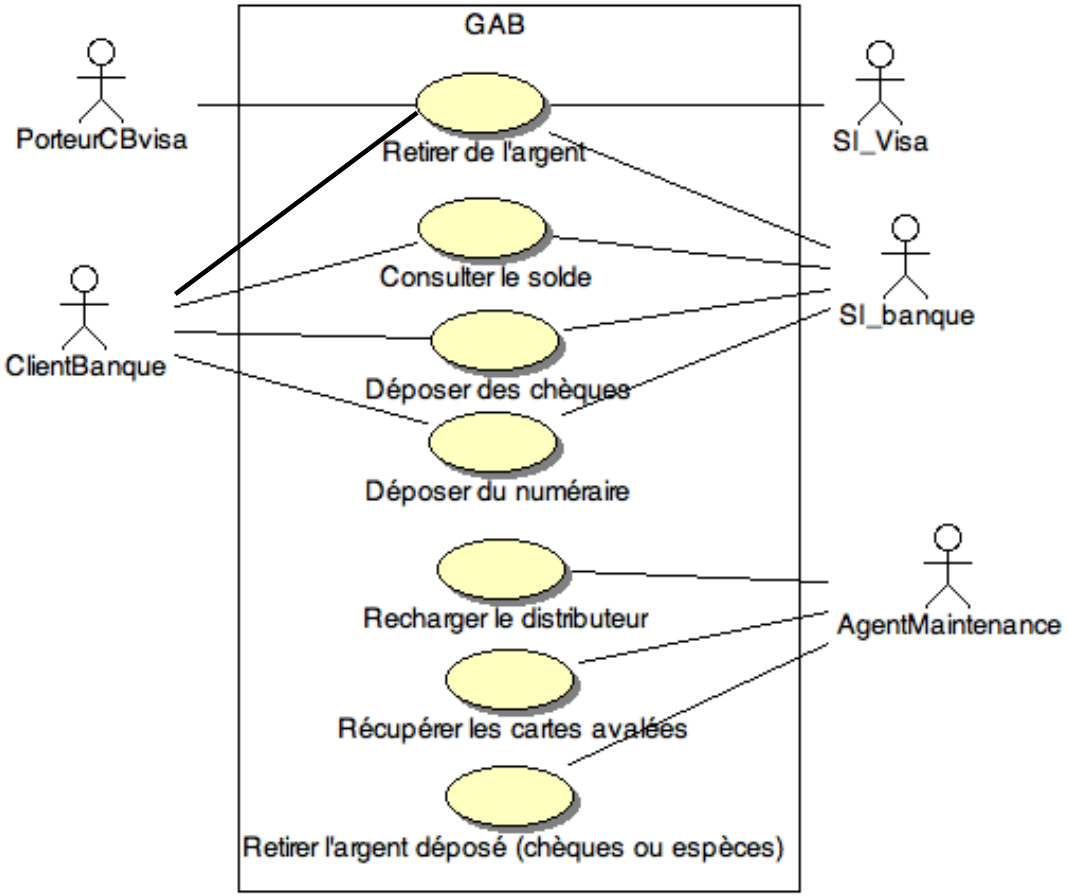
Identifier les acteurs du GAB

1. Acteurs principaux
   - Porteur de CB visa
   - Client de la banque
   - Agent de maintenance
2. Acteurs secondaires
   - SA visa (organisme de carte bancaire)
   - SI banque (Système d'Information de la banque)

Identifier les cas d'utilisation

1. Pour un porteur de CB visa
   - Retirer de l'argent
2. Pour un client de la banque
   - Retirer de l'argent
   - Consulter le solde
   - Déposer des chèques
   - Déposer des numéraires
3. Pour l'agent de maintenance
   - Retirer des chèques et numéraires
   - Récupérer les cartes avalées
   - Recharger le distributeur

Diagramme de cas d'utilisation du GAB



Documenter les cas d'utilisation

- Un cas d'utilisation ne se limite pas à sa représentation graphique, sa documentation n'est pas optionnelle mais *nécessaire* et obligatoire pour la définition du cas
- La documentation doit expliciter les activités à dérouler, le déroulement *nominal* et le déroulement *exceptionnel*, les déroulements *alternatifs* (interruption du déroulement nominal du cas initial par un autre cas d'utilisation pour les besoins du cas initial) et les *pré et post conditions* pour s'assurer du bon déroulement du cas.

Documenter les cas d'utilisation

Titre : retirer de l'argent avec une carte visa

Résumé : ce cas permet au porteur de CB qui n'est pas un client de la banque de retirer de l'argent, si son crédit hebdomadaire le permet

Date de création : 02/02/2008

Date de mise à jour : 03/03/2022

Version : 2.3

Responsable : C. Plumejeaud

Pré conditions :

- La caisse au GAB est alimentée
- Aucune CB ne se trouve dans le lecteur

Documenter les cas d'utilisation

Scénario nominal

1. Le porteur de CB introduit la carte dans le lecteur du GAB
2. Le GAB vérifie que la carte est de type Visa
3. Le GAB demande au porteur de saisir son code d'identification
4. Le porteur saisir son code d'identification
5. Le GAB compare le code saisi avec celui de la puce de la carte
6. Le GAB demande une autorisation au SI_visa
7. SI_visa donne son accord en indiquant le solde hebdomadaire
8. Le GAB demande au porteur de CB d'indiquer le montant souhaité
9. Le porteur saisit le montant
10. Le GAB contrôle le montant demandé par rapport au solde hebdomadaire
11. Le GAB demande au porteur de CB s'il veut un ticket
12. Le porteur de CB demande un ticket
13. Le GAB rend la carte au porteur de CB
14. Le porteur de CB reprend sa carte
15. Le GAB délivre les billets et un ticket
16. Le porteur de CB prend les billets et le ticket.

Documenter les cas d'utilisation

Enchaînement alternatifs

A1 : code d'identification provisoirement erroné. L'enchaînement A1 commence au point 5 du scénario nominal

- Le GAB indique au porteur que le code est erroné, pour la première ou seconde fois
- Le GAB enregistre l'échec sur la carte

Le scénario reprend au point 3

A2 : montant demandé supérieur au solde hebdomadaire. L'enchaînement A2 commence au point 10 du scénario nominal

- Le GAB indique que le montant est supérieur au solde hebdomadaire

Le scénario reprend au point 8

A3 : ticket refusé. L'enchaînement A3 commence au point 11 du scénario nominal

- Le porteur refuse les billets

Le scénario reprend au point 13

Documenter les cas d'utilisation

Enchaînement d'exception

E1 : carte non valide. L'enchaînement E1 commence au point 2 du scénario nominal

- Le GAB indique au porteur que la carte n'est pas valide et la confisque

Le cas d'utilisation est terminé

E2 : le code d'identification est faux pour la 3eme fois. L'enchaînement E2 commence au point 5 du scénario nominal

- Le GAB indique que au porteur que le code est erroné pour la troisième fois
- Le GAB confisque la carte
- Le SI_visa est informé

Le cas d'utilisation est terminé

E3 : Retrait non autorisé. L'enchaînement E3 commence au point 6 du scénario nominal

- SI_visa interdit tout retrait
- Le GAB éjecte la carte

Le cas d'utilisation est terminé

Documenter les cas d'utilisation

Enchaînement d'exception (suite)

E4 : carte non reprise. L'enchaînement E4 commence au point 13 du scénario nominal

- Le GAB confisque la carte au bout de 15 secondes
- Le SI_visa est informé

Le cas d'utilisation est terminé

E5 : Billets non pris. L'enchaînement E5 commence au point 15 du scénario nominal

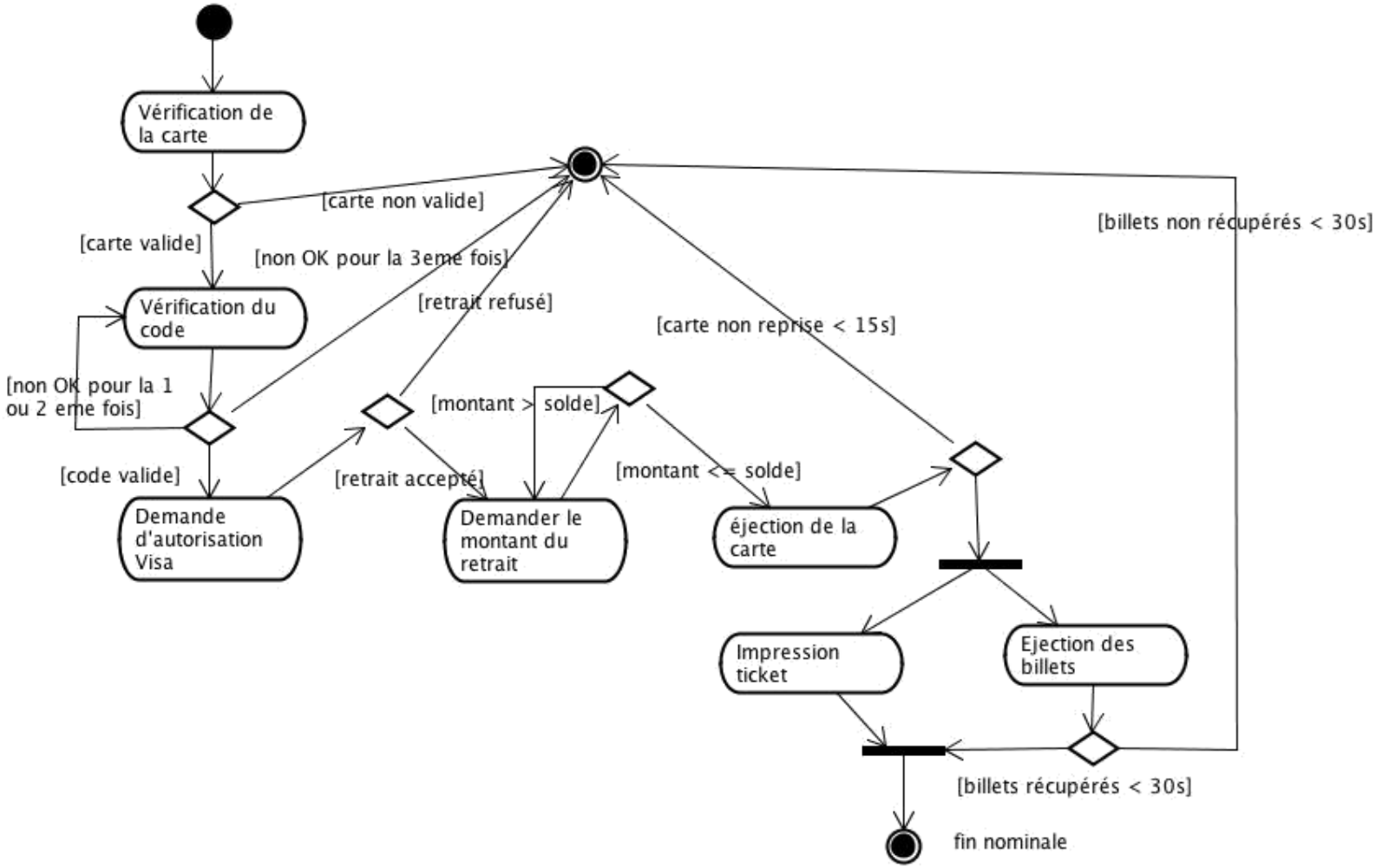
- Le GAB reprend les billets au bout de 30 secondes
- Le SI_visa est informé

Le cas d'utilisation est terminé

Postconditions :

La caisse du GAB est mise à jour en fonction du montant des retraits effectués.

Diagramme d'activité du GAB



Le fil rouge: différents points de vue sur « Réaliser la campagne »

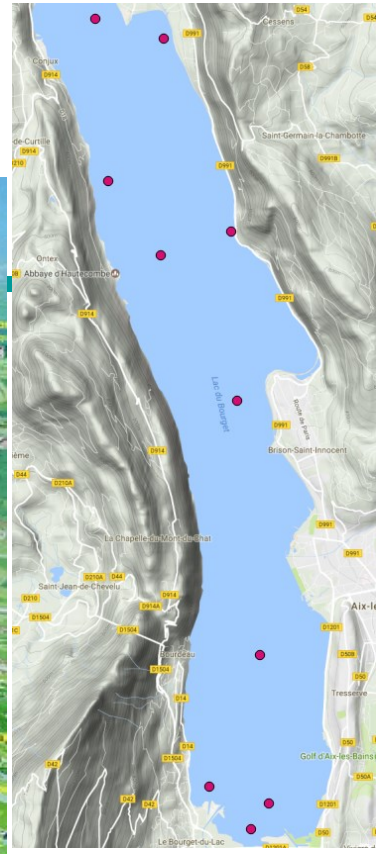
Reprendre l'énoncé du problème et réaliser par groupes l'analyse fonctionnelle de cette base de données

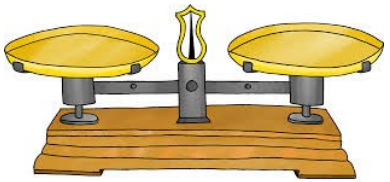
- Diagrammes de cas d'étude
- Diagrammes d'activité et de séquence

Application

1. Introduction de ce fil rouge (5 min)
2. Lire le fil rouge – 10 min attentivement
3. Déterminer au moins 3 cas d'utilisations autour du SI « friture du lac »
   - Préparer la campagne
   - Faire la campagne
   - Exploiter les données de la campagne

Un vrai protocole





Carte

Les espèces pêchées

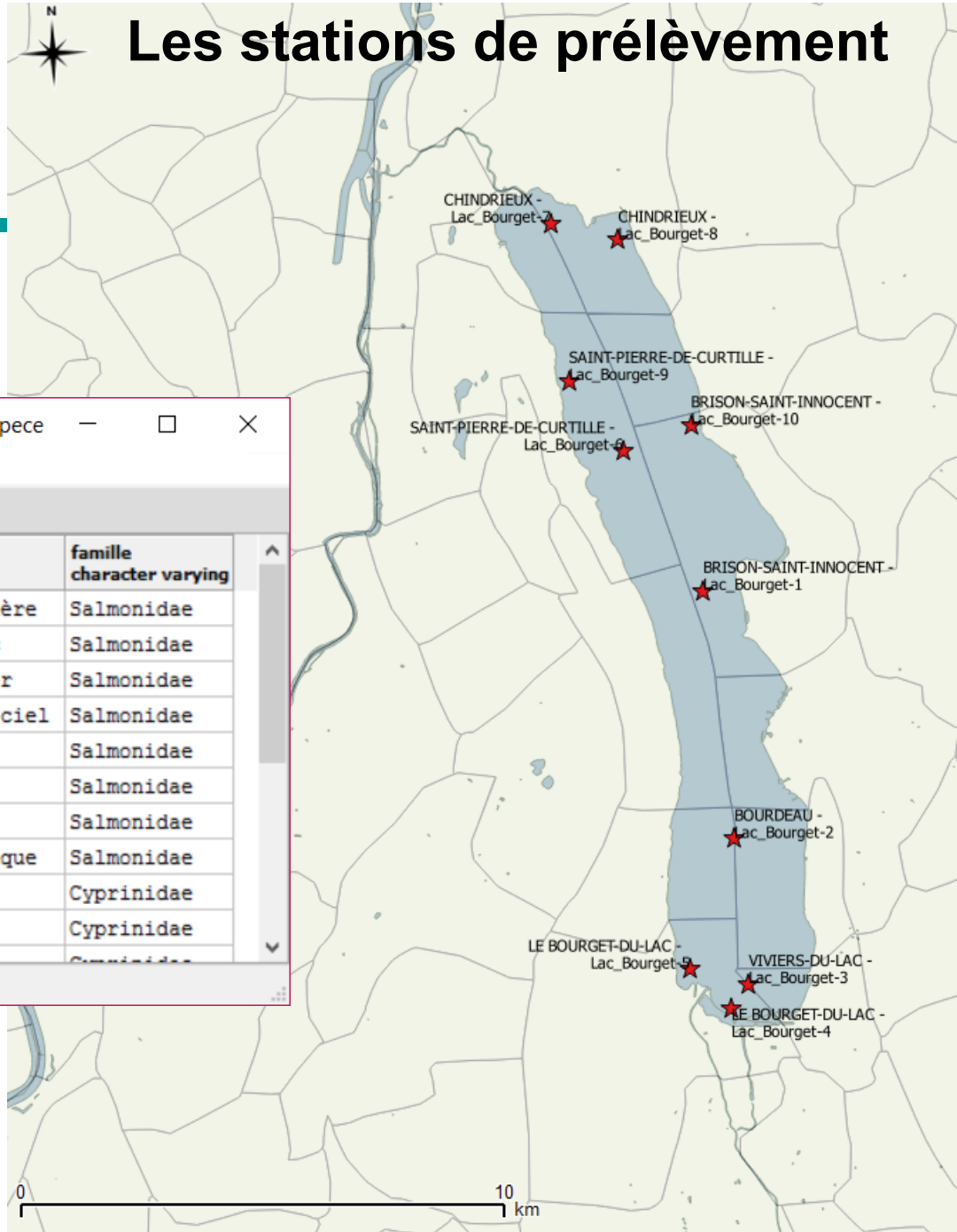
Édition des données - tp_anf (194.214.86.66:5432) - savoie - poisson.espece

Fichier Édition Affichage Outils Aide

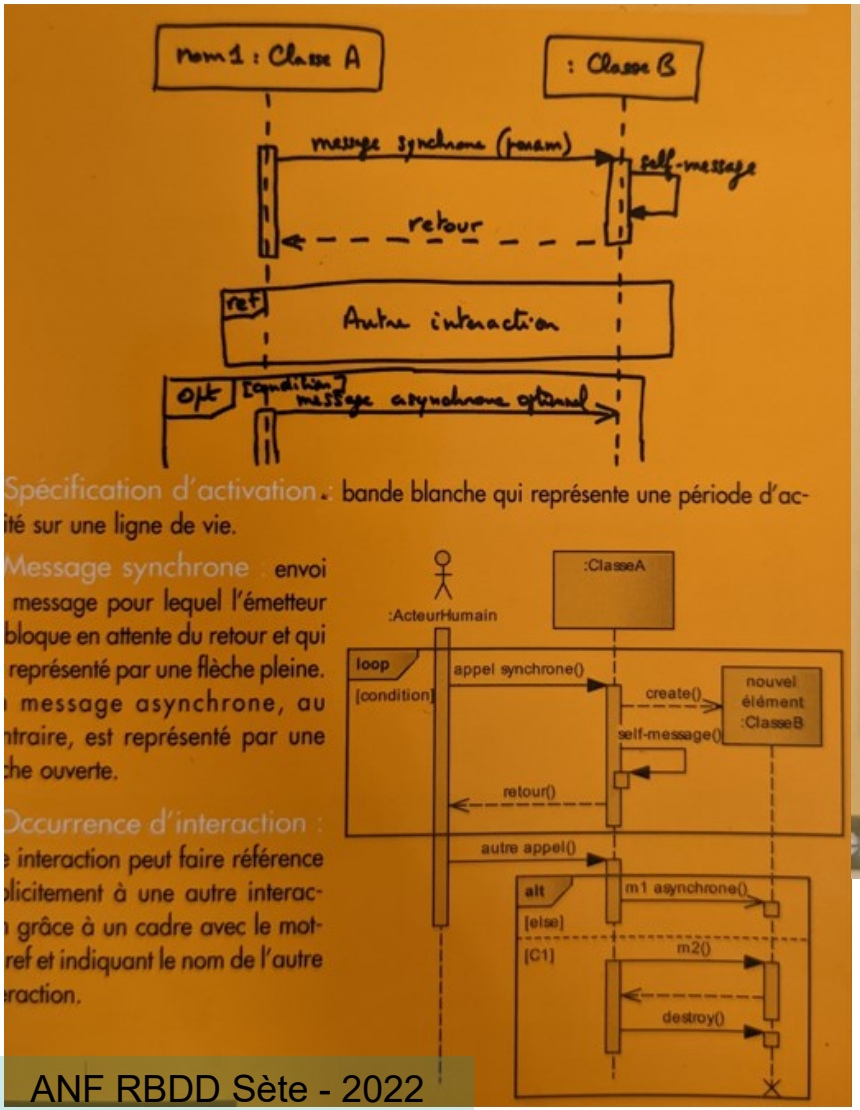
Pas de limite

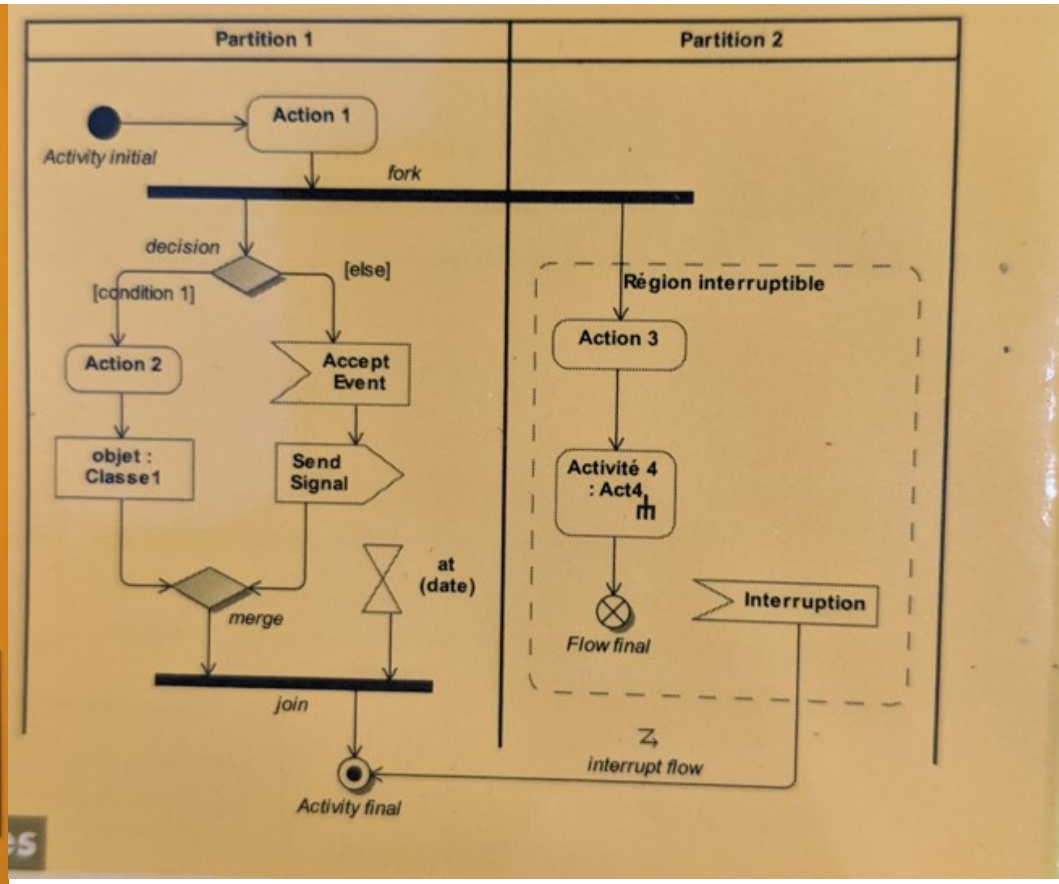
	cd_nom [PK] integer	nom_scientifique character varying	nom_vernaculaire character varying	famille character varying
1	67778	Salmo trutta fario	Truite de rivière	Salmonidae
2	67854	Coregonus albula	Corégone blanc	Salmonidae
3	67812	Salvelinus alpinus	Omble chevalier	Salmonidae
4	67804	Oncorhynchus mykiss	Truite arc-en-ciel	Salmonidae
5	67819	Salvelinus namaycush	Cristivomer	Salmonidae
6	67759	Thymallus thymallus	Ombre commun	Salmonidae
7	67794	Hucho hucho	Huchon	Salmonidae
8	67765	Salmo salar	Saumon atlantique	Salmonidae
9	67111	Alburnus alburnus	Ablette	Cyprinidae
10	67466	Scardinius erythrophthal	Rotengle	Cyprinidae

17 lignes.



Recap Séquences et Activité (P. Roques)





14h-17h50
Pause à 16h (20 min)

Plan Mardi après-midi

1. Classes et objets : 15 min
   - Un exercice (TD 4 SPECTACLES) : 30 min
2. Les associations : 15 min
   - Un exercice (TD 4 SPECTACLES) : 30 min
   - Un exercice (TD5 RECETTE) : 30 min
3. Héritage : 10 min
   - Un exercice (TD6 GARAGE) : 30 min
4. Le fil rouge : 45 min

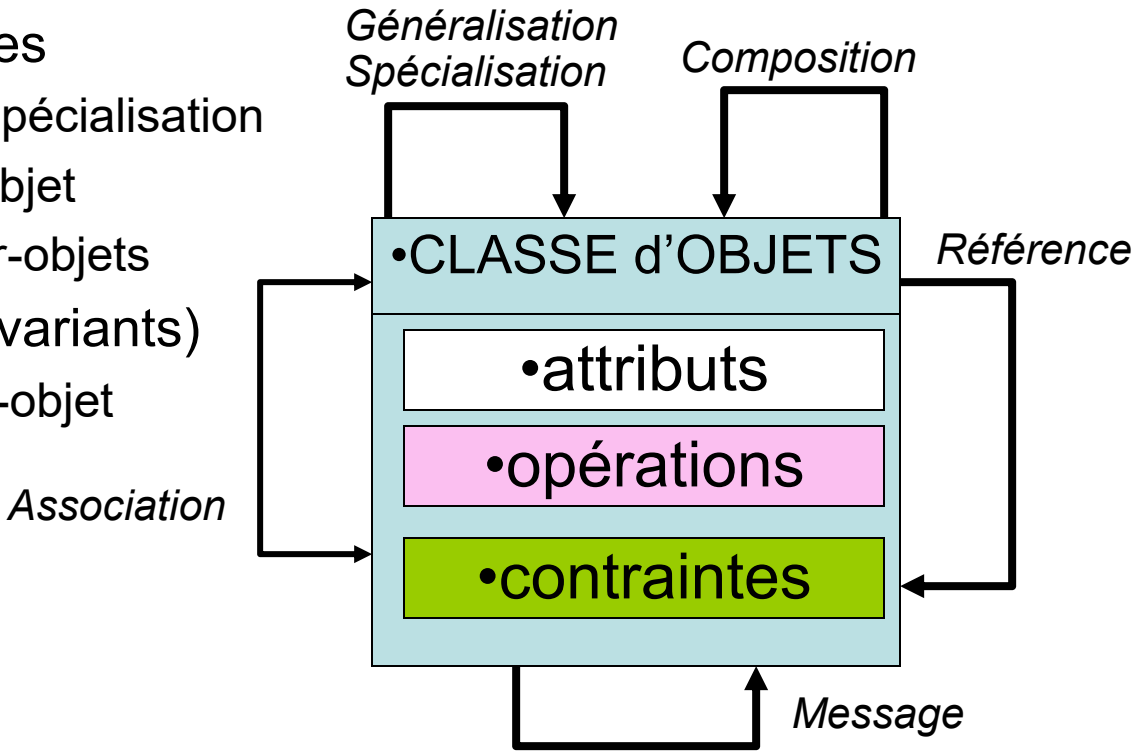
DIAGRAMMES DE CLASSE

Cours du Mardi après-midi

https://www.uml-diagrams.org/class-reference.html

Aspects statiques d'un système

- Identification de l'objet
- Valeurs de l'objet
- Structuration complexe des données : produit cartésien, ensemble, liste
- Graphes de classes
  - Généralisation/spécialisation
  - Composition d'objet
  - Association inter-objets
- Contraintes (ou invariants)
  - Intra-objet, Inter-objet



Paradigme Conception Orientée-Objet

OBJET = Entité **pertinente** du domaine technique

« Une entité capable de mémoriser un état (information) et qui offre des opérations (comportement) pour à la fois explorer et modifier cet état. » *(Jacobson, 1992)*

Exemples :

- Objets matériels : table, chaise, voiture
- Objets immatériels : compte en banque, voyage, équation, point, ligne
- Objets conceptuels : groupe de travail, idée politique

Attention à la **subjectivité** des points de vue :

L'objet étant une abstraction faite par un humain d'un phénomène réel observé, 2 personnes peuvent donner 2 modèles différents du même phénomène réel.

L’encapsulation

«L’encapsulation est le processus qui consiste à compartimenter des éléments d’une abstraction pour constituer sa structure et son comportement » *(Booch, 1995)*

L’abstraction se focalise sur les caractéristiques observables et le comportement de l’objet

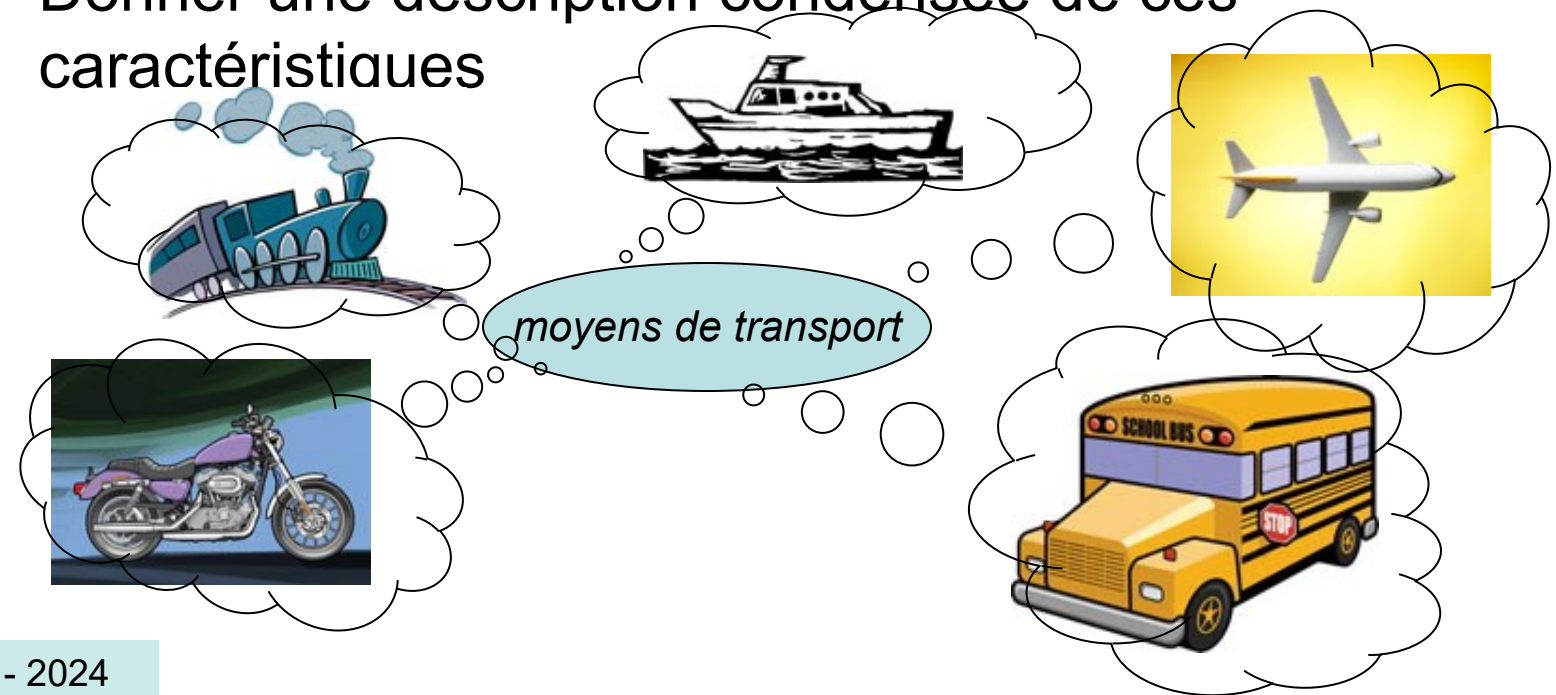
Encapsulation : masquer les détails de la réalisation

- Masquer la structure sous-jacente et l’implémentation des opérations
- Les plus gros bénéfices obtenus par une approche orientée objet proviennent du respect du principe d’encapsulation.

Comment abstraire ?

Regrouper des objets ou concepts qui se rassemblent

- Identifier des caractéristiques communes à un ensemble d'éléments
- Donner une description condensée de ces caractéristiques



La classe

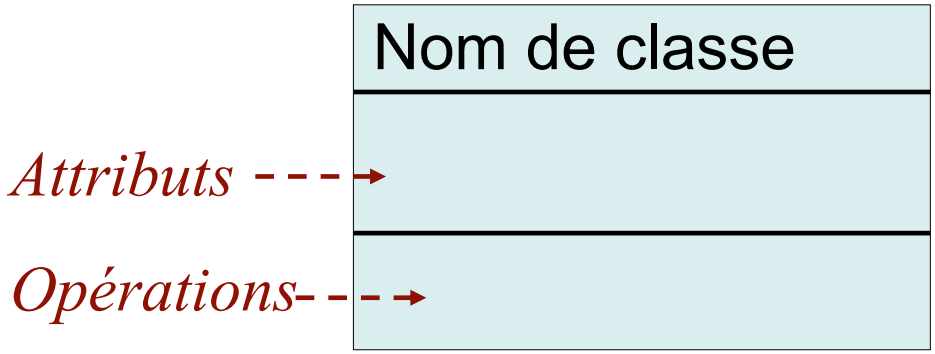
1. Spécification :
   - décrit le domaine de définition et les propriétés des instances de cette classe (équivalent de la notion de type dans les langages classiques)
2. Réalisation :
   - décrit comment le spécification est réalisée et qui contient le corps des opérations et les données nécessaires à leur fonctionnement.
3. Catégories :
   - Classes instanciables
   - Classes non instanciable : *abstract*
   - Classes dérivables
   - Classes utilitaires : *System*

Notation UML d'une classe

Une classe est une description abstraite d'un ensemble d'objets ayant des propriétés similaires, un comportement commun, des relations communes avec d'autres objets, et des sémantiques communes

Nom de classe

Personne



Avec UMLet

```
Personne
--
nom
prénom
dateNaissance
--
age()
```

Personne
nom prénom dateNaissance
age()

Passer des classes aux objets

- **Tout objet est une instance de classe**
- La classe définit les comportements et les structures possibles de ces instances qui sont les objets
- Différentes instances (objets) peuvent avoir leurs opérations invoquées à différents moments, de différentes manières, et être dans différents états.
- Le mécanisme d'instanciation permet de générer des objets (instances) d'une classe.

  *Point p = **new** Point(10, 5);*

Objet

Définition OO : « une structure de données et un ensemble de fonctions associées à cette structure de données »

Caractérisé par :

- Une *identité* (ou un nom)
- Un *état* interne: la valeur de ses attributs
- Un *comportement* : ses méthodes ou opérations

Dans ce cours, je passe sous silence la modélisation du comportement car je ne m'intéresse qu'à la partie statique

L'état interne d'un objet

Les attributs représentent les caractéristiques de l'objet (son *état*)

Un attribut est constitué du couple (nom, valeur)

Exemple :

Sac à main
Couleur : {rose, rouge, noir, marron, blanc, …}
Texture : {Cuir, plastique, tissu, …}
Forme : {carrée, enveloppe, sac à dos, ronde, …}
Marque : {Gucci, Dior, Hermes, …}

On définit un **domaine** pour la valeur de l'attribut.

Le plus souvent, c'est un type de base (entier, réel, chaîne de caractères, booléen) mais cela peut aussi être un type énuméré (*enum*) pour un ensemble fini de valeurs

Identité de l’objet

Elle caractérise l’existence propre de l’objet

- Immuable
- Permanente : permet de distinguer tout objet de façon non ambiguë indépendamment de son état
- Concept différent de l’égalité dans les approches orientés objet
- Deux objets peuvent être égaux, mais correspondrent chacun à 2 instances différentes.

Sac à main
couleur texture forme marque

Christine

Sylvie

L'objet est une instance d'une classe

Représentation UML d'un objet

Nom de l'objet : nom de sa classe

Nom des attributs : valeur

Un objet n'a pas d'opération

Avec UMLet


```
__sacDeChristine : SacAMain__
--
couleur : bleue
texture : « toile »
forme : « enveloppe »
marque : « DIY »
```

Comportement de l'objet

Il regroupe toutes les compétences de l'objet et il est décrit par l'ensemble des *opérations* applicables à l'objet.

Décrit les actions et les réaction d'un objet (sa *dynamique*)

- Les méthodes (opérations)
- Les messages (évènements)

Une opération est déclenchée suite à une stimulation externe.

Les opérations

Opérations (spécifie un service) ≠Méthodes (implémente l'opération)

Une opération :

- des paramètres (avec un type, un mode, une valeur par défaut). Ils sont optionnels.
- un résultat (typé)

Operation (mode$_1$ param$_1$: Type$_1$ = val$_1$, …,
mode$_n$ param$_n$: Type$_n$ = val$_n$) : TypeRésultat

Le mode peut-être :

- In : paramètre d'entrée
- Out : paramètre de sortie (modifié en sortie d'opération)
- Inout : paramètre d'entrée-sortie (lu en entrée et modifié en sortie d'opération)

Les attributs d'une classe

- Définissent les propriétés des objets d'une classe
- Chaque nom d'attribut est unique dans une classe
- L' identification est fournie par le système, elle n'est pas à considérer dans le modèle objet
- Chaque objet a ses propres valeurs pour chacun des attributs de sa classe.

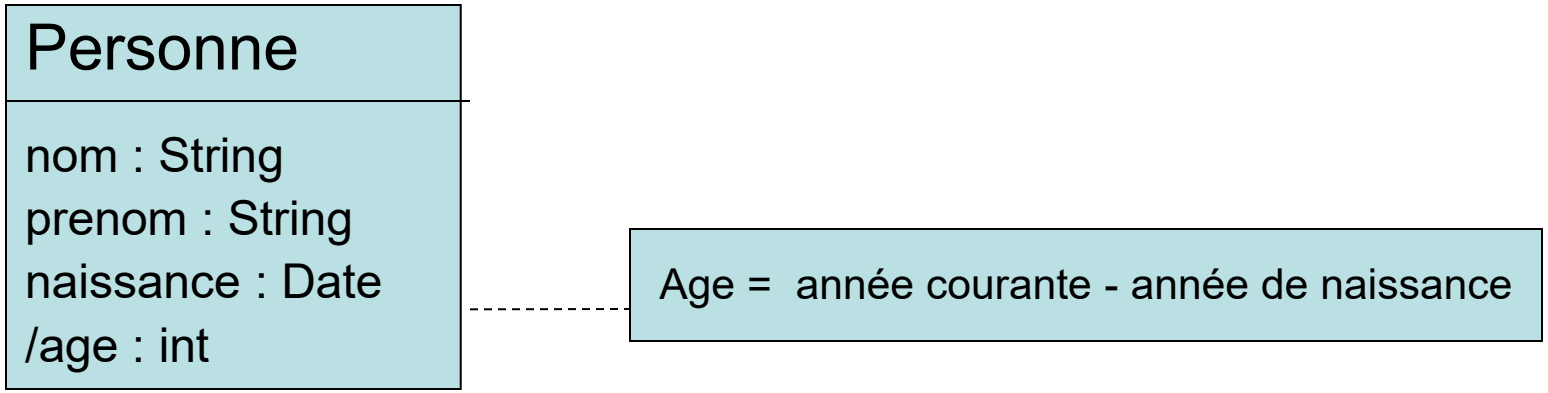
Nom de classe
Nom d'attribut : type = valeur initiale

Personne
nom : String prenom : string naissance : Date

Personne
ID_pers : ID nom : String prenom : string naissance : Date

Décrire les attributs

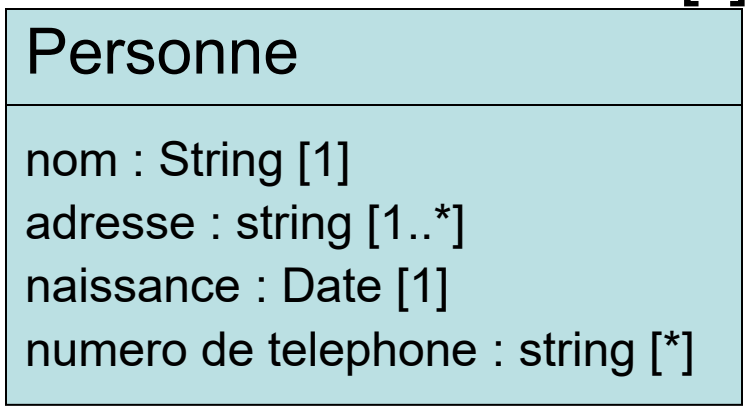
- Le type n'est pas obligatoire
- Les types possibles ne sont pas spécifiés en UML: int, char, String, boolean, Date, etc.
- Les attributs dérivés notés /nom d'attribut peuvent se déduire des autres attributs (avec parcimonie !)
- Un attribut de classe est un attribut partagé par toutes les instances de la classe (*static*)



Cardinalité des attributs

La multiplicité d'un attribut spécifie le nombre de valeurs possibles pour chacunes des instanciations.

- Valeur obligatoire [1] (par défaut)
- Une seule valeur optionnelle [0..1]
- Plusieurs valeurs [*]



Une personne a une ou plusieurs adresses,

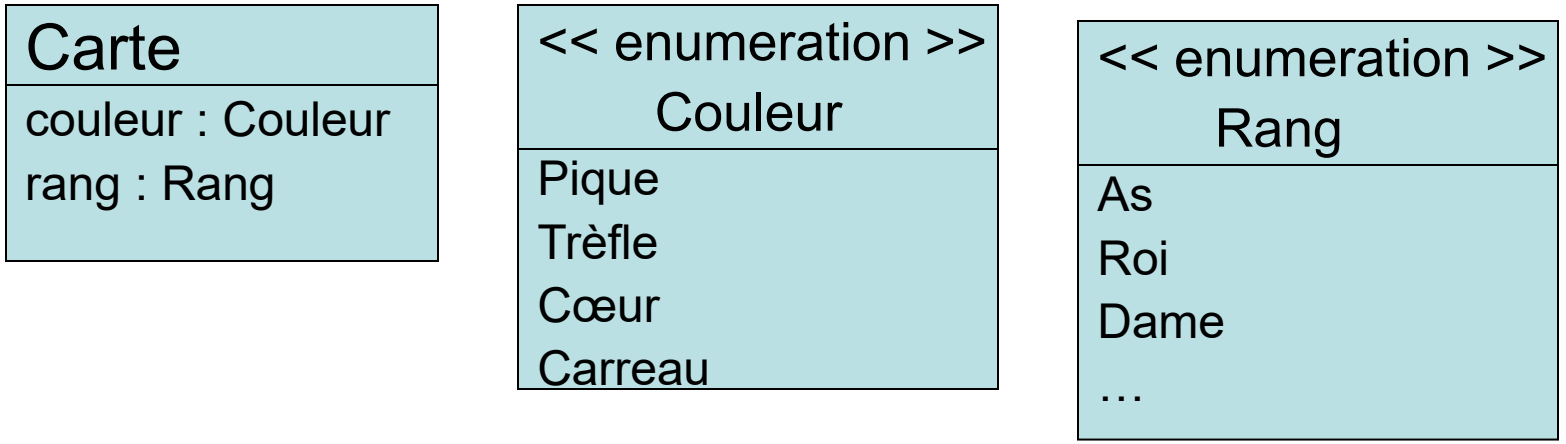
zéro ou plusieurs numéros de téléphone,

et une seule date de naissance

Les énumérations

Une énumération est un type de données qui a un ensemble fini de valeurs.

Noté avec le stéréotype <<enumeration>>, l'ensemble des valeurs possibles est listé sous le nom de l'énumération



Règles de la classification

1. Identification de chaque objet
2. Appartenance EXPLICITE de chaque objet à UNE classe. Implicite à plusieurs classes.
3. Importances des objets PERSISTANTS : mécanismes de sérialisation
4. UNICITE des noms de classe
5. Au moins UNE propriété (attribut ou méthode) par classe
6. Objet ≠ valeur

Valeur : entité mathématique, éternelle, immuable, jamais créée ni détruite, ni mise à jour ! 2+5 n'implique pas que 7 vient d'être créé, ni que 2 et 5 ont été consommés

Objet : entité modifiable tout en gardant son identité ; son état peut changer au cours du temps. Il peut être créé, détruit, abandonné, partagé, mis à jour.

Exercice TD 4 Spectacle

1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

TD 4 énoncé

•Un abonné est caractérisé par un numéro (unique), un nom, une adresse, un téléphone et une carte de crédit (avec son type, son numéro et sa date d'expiration).

• Un abonné est une personne inscrite auprès de la société et susceptible de réserver des places pour les représentations proposées par la société.

• Une représentation concerne un spectacle et elle est caractérisée par une date de représentation, un nombre de places (encore) disponibles et le prix d'une place.

• Il n'existe pas deux représentations d'un même spectacle à une même date de représentation.

• Pour chaque spectacle, on connaît en plus de son titre, qui est unique pour l'ensemble des spectacles mémorisés, la troupe qui le représente et son producteur.

• Un abonné peut obtenir plusieurs réservations pour une même représentation.

• Chaque réservation enregistrée, rattachée à un seul abonné, se voit attribuer un numéro unique en plus de sa date d'enregistrement et du montant total à payer.

• La date à laquelle l'abonné passe la réservation doit être antérieure à celle d'expiration de sa carte de crédit.

• ~~Une réservation comporte entre une et dix demandes par abonné.~~ Chaque demande précise le nombre de places demandées. *Un abonné ne peut pas émettre plus de 10 demandes. Une demande donne lieu à une réservation.*

• Le nombre de places réservées à un moment donné pour une représentation doit toujours rester inférieur au nombre de places disponibles. Cette exigence peut amener à devoir ajuster le nombre de places effectivement demandées par un abonné.

Les associations

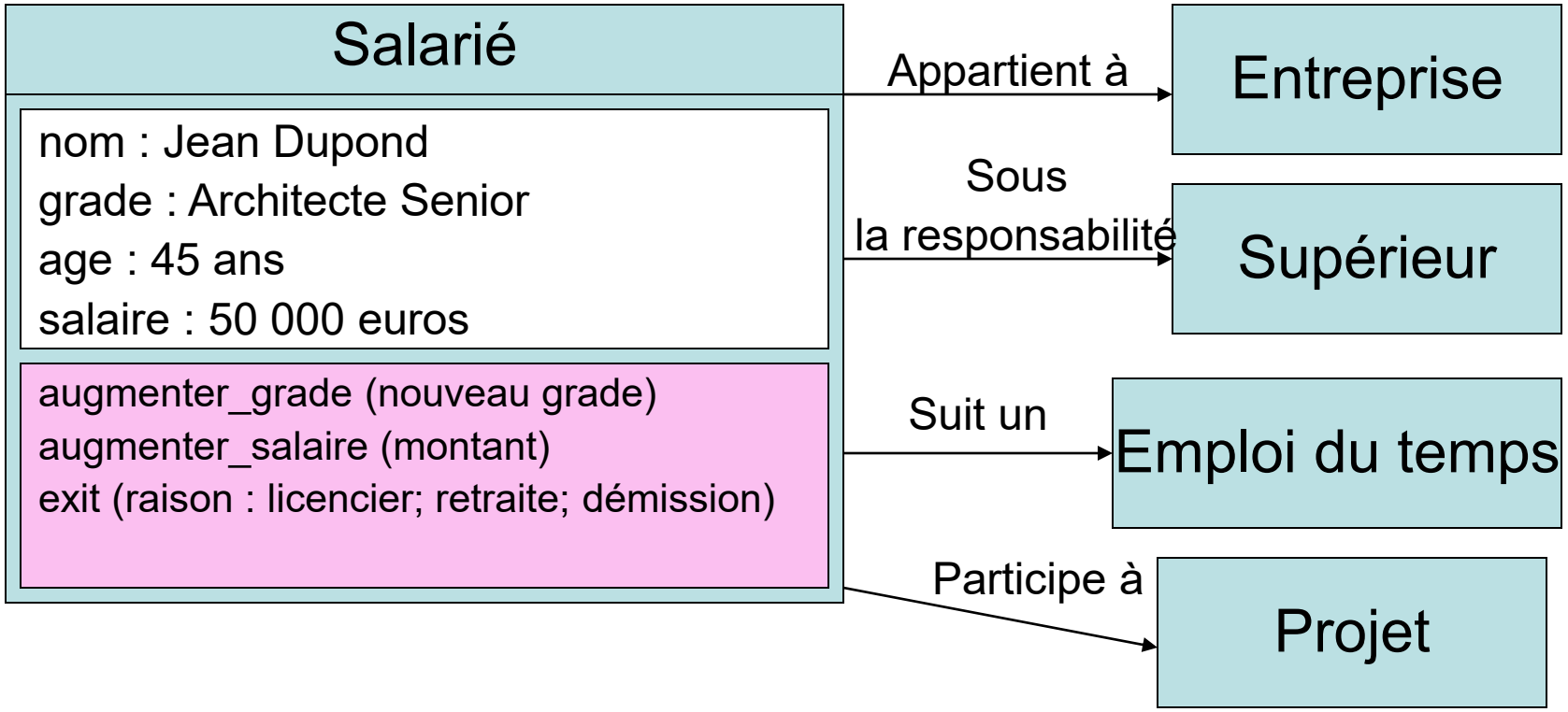
1. L'association est une connexion sémantique bidirectionnelle entre classes
2. C'est une abstraction des liens qui existent entre les objets des classes associées.
3. Elle symbolise une information dont la durée de vie n'est pas négligeable par rapport à la dynamique générale des classes instanciées.
4. Dès lors qu'un objet a un lien vers un autre objet, il peut émettre un message pour déclencher une de ses opérations.

Si Entreprise E est une composition de Salarié S_i. Une augmentation globale se traduit ainsi:

```
Liste salariés = E.listeDesSalariés();
Pour i de 1 à E.nombreSalariés,
    Salarié Si = salariés.getElement(i);
    Si.Augmenter_salaire (100);
```

Exemple

Un salarié dans le système entreprise



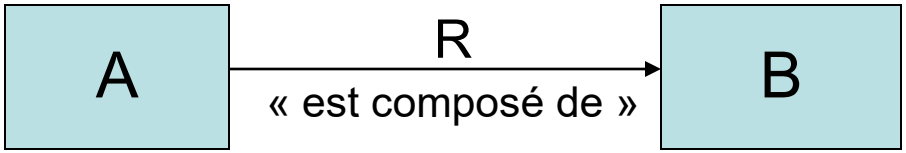
Propriétés des associations

Relations de classes bidirectionnelle dissymétriques

- Un mariage concerne un homme et une femme durant une période
- Un train est composé de voitures : composition
- Une course agrège des participants : agrégation

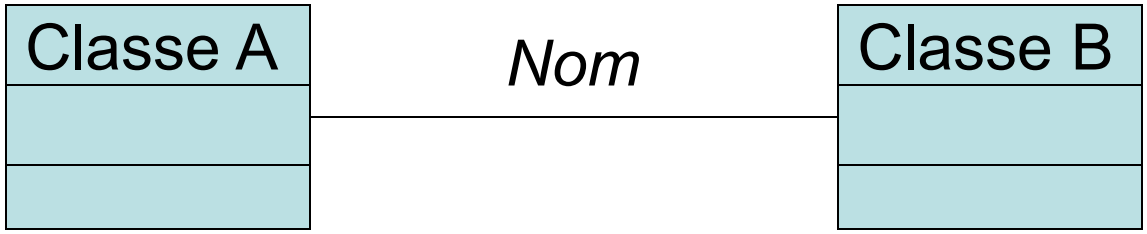
Ces relations possèdent une arité (ou multiplicité) exprimé par un entier : $n \in \mathbb{N}$

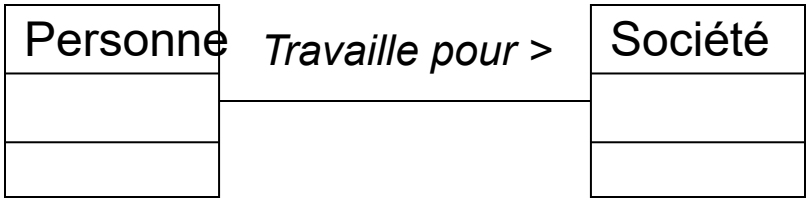
Une association peut avoir une orientation.

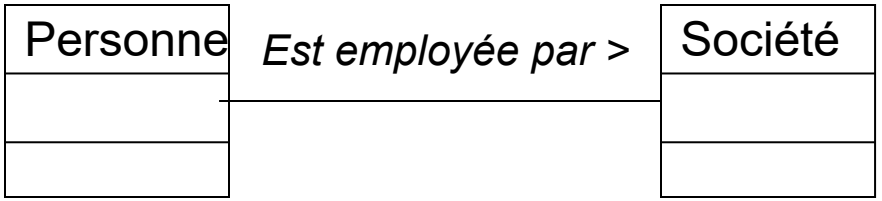


Nommage des associations

- Le nom d'une association apparaît en italique, au milieu du lien
- Nommer l'association par une forme verbale (active ou passive)
- On peut noter le sens de navigation (lecture)

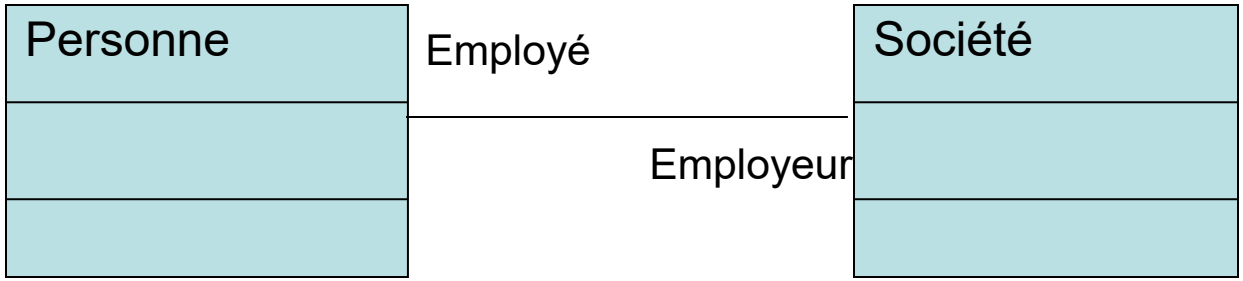






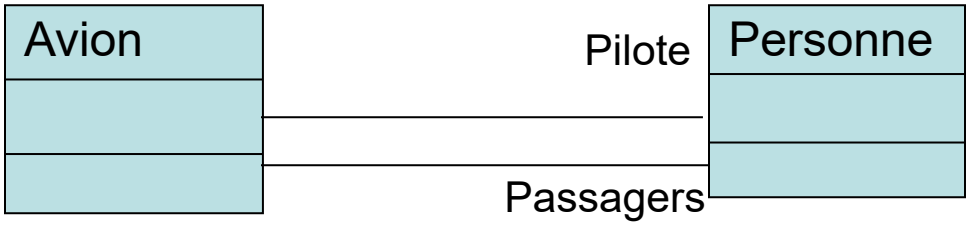
Nommage des rôles

- Chaque association binaire possède 2 rôles
- Le rôle décrit comment une classe voit une autre classe au travers d'une association
- Le nommage des associations et des rôles ne sont pas exclusifs l'un de l'autre

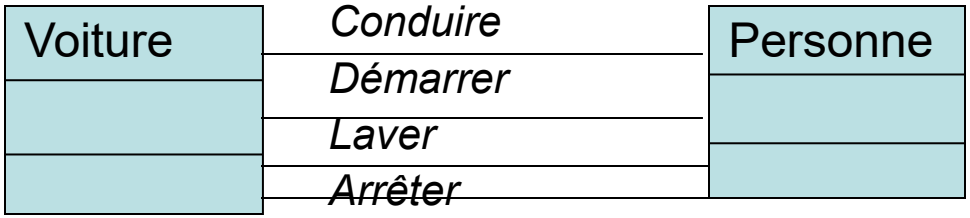


Nommage des rôles

- Le nommage des des rôles prend tout son intérêt lorsque plusieurs associations relient 2 classes. Chaque association exprime un concept distinct



- La présence d'un grand nombre d'associations entre 2 classes est suspecte : signe d'une mauvaise décomposition



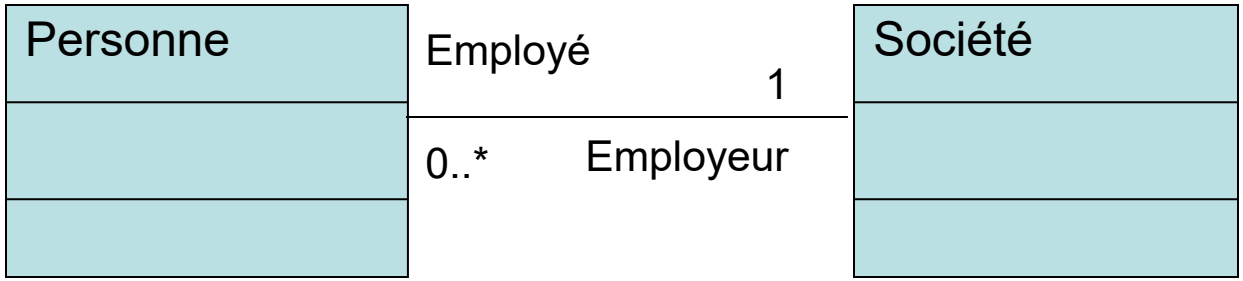
Multiplicité des associations

La *multiplicité* est une information portée par le rôle, sous la forme d'une expression entière bornée.

1	Un et un seul
0..1	Zéro ou un
M..N	De M à N, entiers positifs
* 0..*	De zéro à plusieurs
1..*	De un à plusieurs

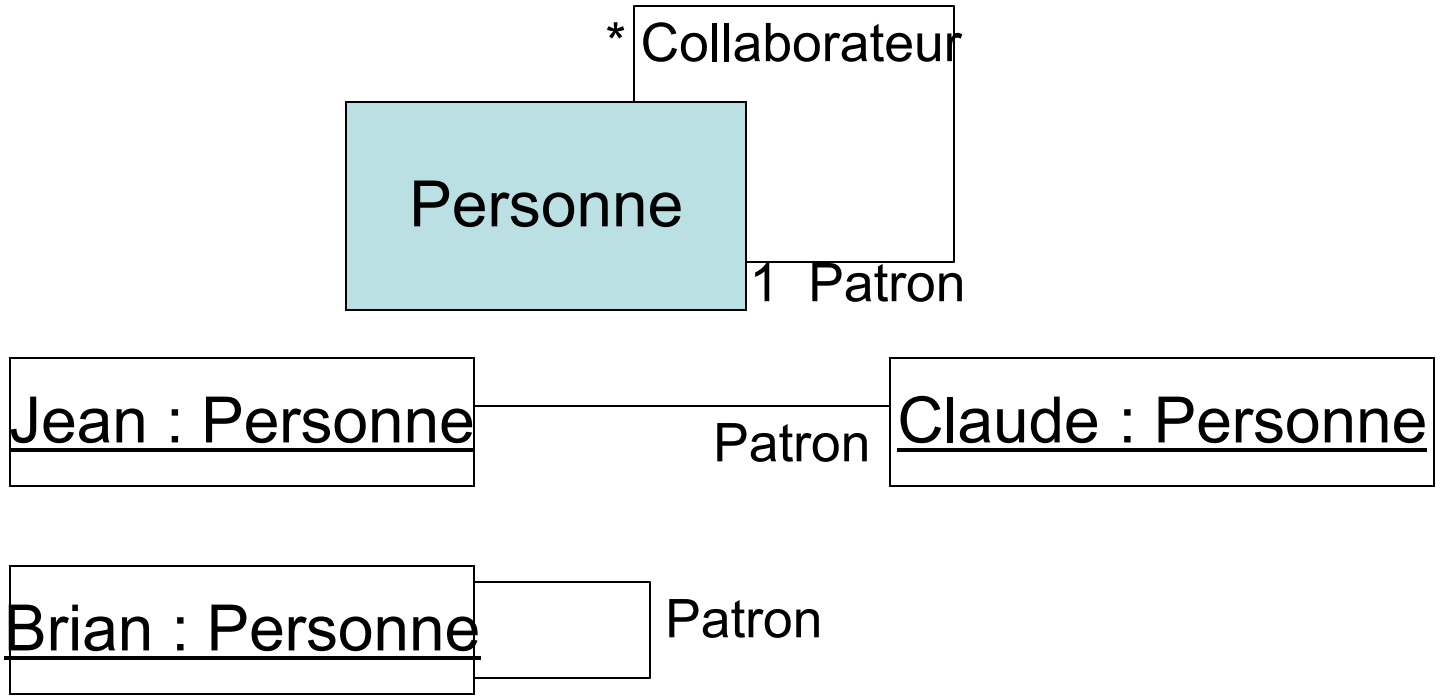
Multiplicité des associations

- Une valeur de multiplicité > 1 implique une collection d'objets.
- Les valeurs de multiplicité sont des contraintes liées au domaine de l'application, valables durant l'existence des objets. A ignorer durant les régimes transitoires



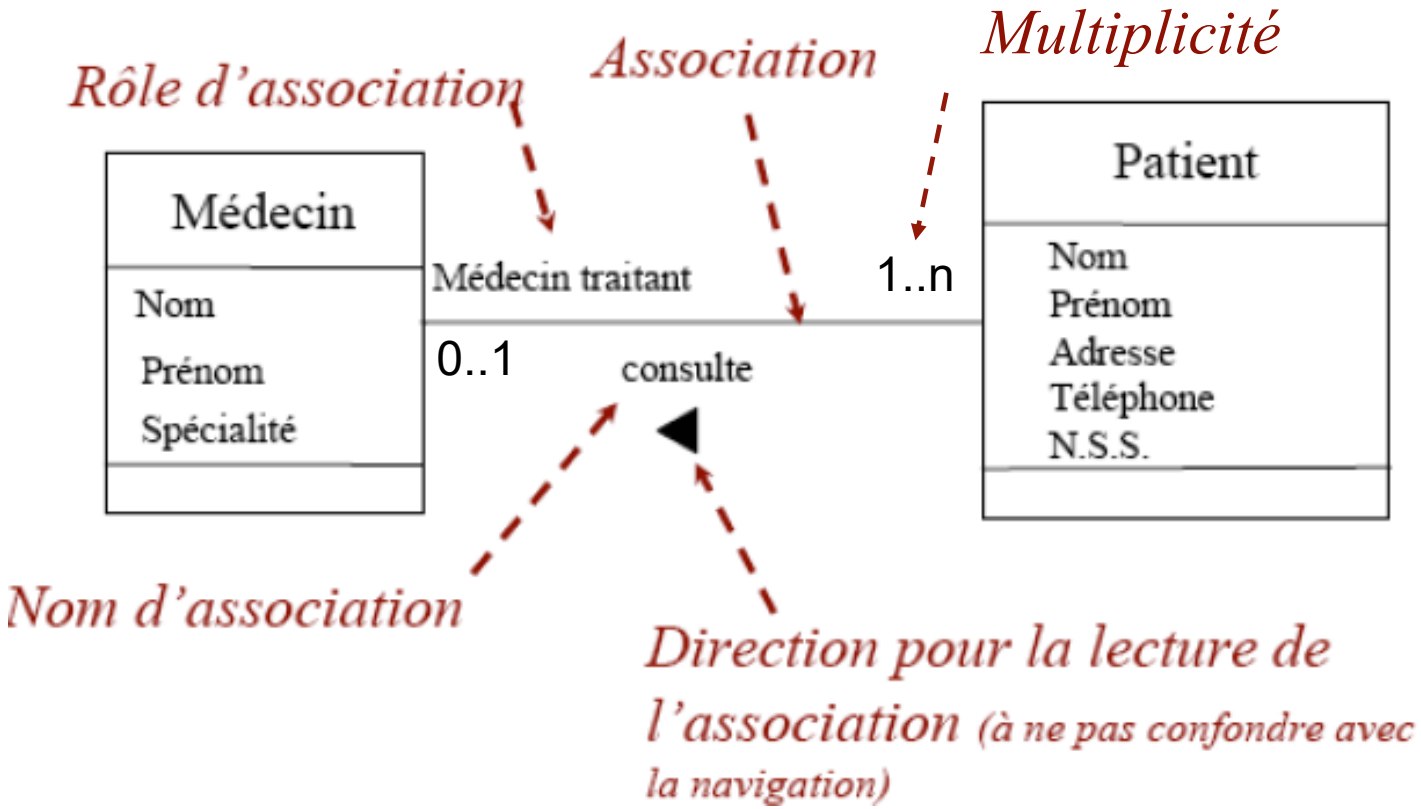
Associations réflexives

Les liens instances des associations réflexives peuvent lier un objet à lui-même.



Les association bi-naires

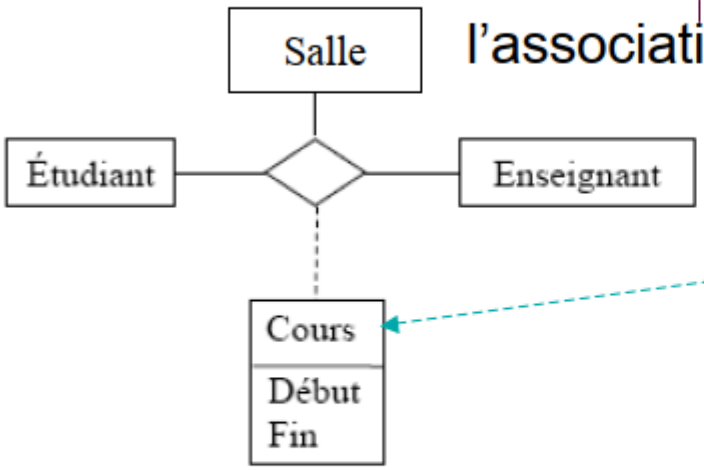
Le cas le plus général



Les associations n-aires

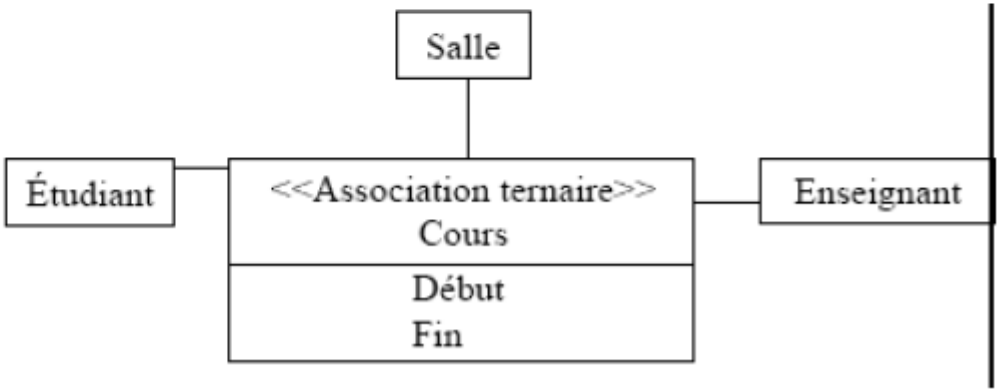
Avec réification des associations

Exemple : une association ternaire avec réification (promotion au rang de classe de l'association)



Classe associative

ou classe-association



Placements des attributs en fonction des valeurs de multiplicité

1. La réification des associations prend tout son sens pour les associations **N vers N**
2. Pour les associations **1 vers 1**, les attributs de l'association peuvent toujours être déplacés dans une des classes qui participent à l'association.
3. Pour les associations **1 vers N**, le déplacement est généralement possible vers la classe côté N; il est fréquent de promouvoir l'association au rang de classe pour augmenter la lisibilité ou en raison de la présence d'associations vers d'autres classes.

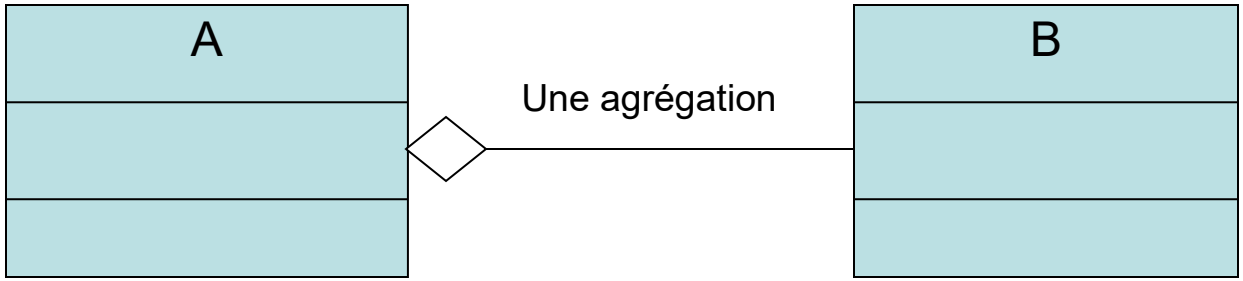
•2008

Agrégation

Représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité. Quelle que soit l'arité, l'agrégation ne concerne qu'un seul rôle de l'association.

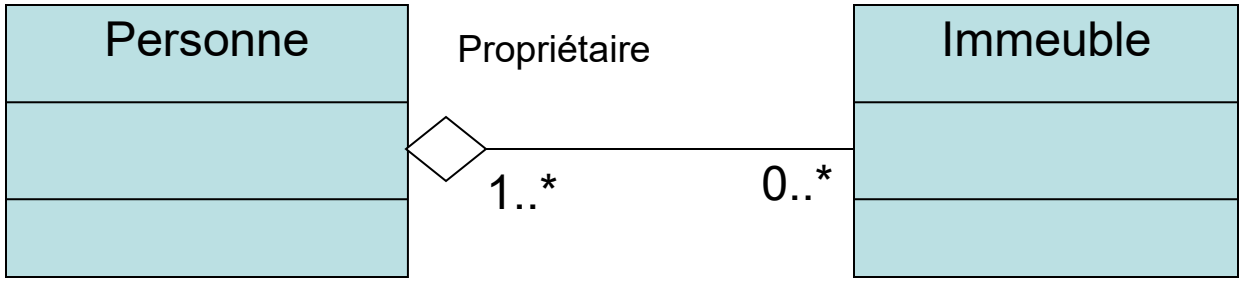
Les critères suivant impliquent une agrégation :

- Une classe fait partie d'une autre classe
- Les valeurs d'attribut d'une classe se propagent dans les valeurs d'attribut d'une autre classe
- Une action sur une classe implique une action sur une autre classe
- Les objets d'une classe sont subordonnés aux objets d'une autre classe



Agrégation

L’agrégation peut-être multiple, comme l’association



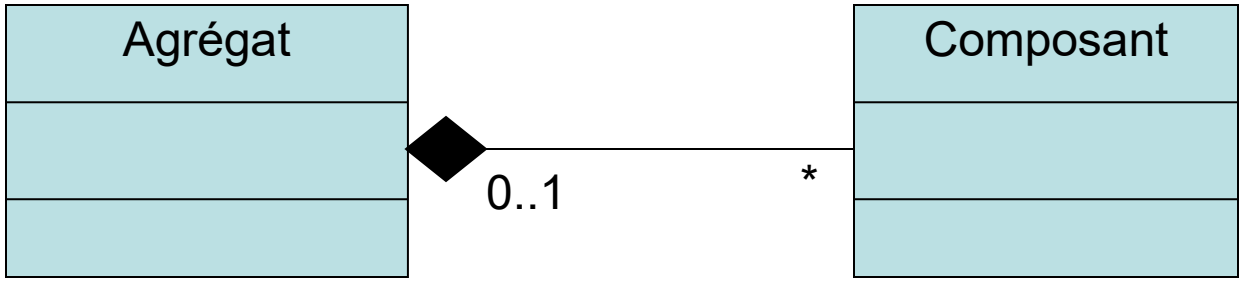
Composition

La composition est une agrégation réalisée par valeur sur les attributs.

Les attributs sont physiquement contenus dans l'agrégat.

La composition implique une contrainte sur la valeur de la multiplicité du coté de l'agrégat : elle ne peut prendre que les valeurs 0 ou 1.

La valeur 0 du côté du composant correspond à un attribut non renseigné.



Agrégation versus Composition

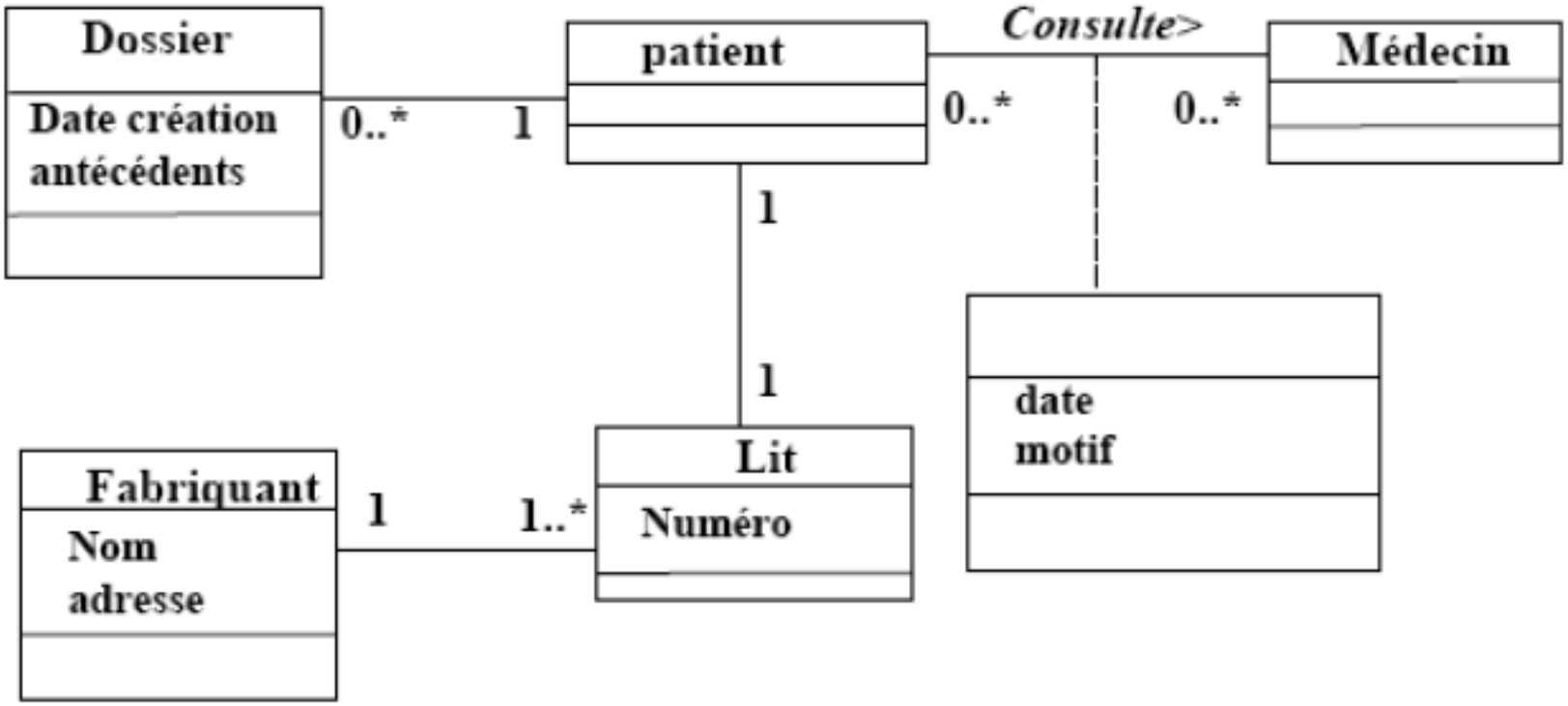
L'agrégation représente une relation de partie-de. La sémantique peut être imprécise

Composition est une relation plus « forte » :

- Chaque partie peut appartenir à une et un seul tout à un instant donné
- Lorsque le tout est détruit (n'a plus d'existence) il en est de même pour toutes les parties qui le composent.

Exemple : une voiture est composée avec 4 roues. Une roue ne peut pas être partagée par plusieurs voiture. Et lorsque la voiture brûle, ses roues aussi.

Exemple et exercices



Exercice TD 4 Spectacle

1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

TD 4 énoncé

•Un abonné est caractérisé par un numéro (unique), un nom, une adresse, un téléphone et une carte de crédit (avec son type, son numéro et sa date d'expiration).

• Un abonné est une personne inscrite auprès de la société et susceptible de réserver des places pour les représentations proposées par la société.

• Une représentation concerne un spectacle et elle est caractérisée par une date de représentation, un nombre de places (encore) disponibles et le prix d'une place.

• Il n'existe pas deux représentations d'un même spectacle à une même date de représentation.

• Pour chaque spectacle, on connaît en plus de son titre, qui est unique pour l'ensemble des spectacles mémorisés, la troupe qui le représente et son producteur.

• Un abonné peut obtenir plusieurs réservations pour une même représentation.

• Chaque réservation enregistrée, rattachée à un seul abonné, se voit attribuer un numéro unique en plus de sa date d'enregistrement et du montant total à payer.

• La date à laquelle l'abonné passe la réservation doit être antérieure à celle d'expiration de sa carte de crédit.

• ~~Une réservation comporte entre une et dix demandes par abonné.~~ Chaque demande précise le nombre de places demandées. *Un abonné ne peut pas émettre plus de 10 demandes. Une demande donne lieu à une réservation.*

• Le nombre de places réservées à un moment donné pour une représentation doit toujours rester inférieur au nombre de places disponibles. Cette exigence peut amener à devoir ajuster le nombre de places effectivement demandées par un abonné.

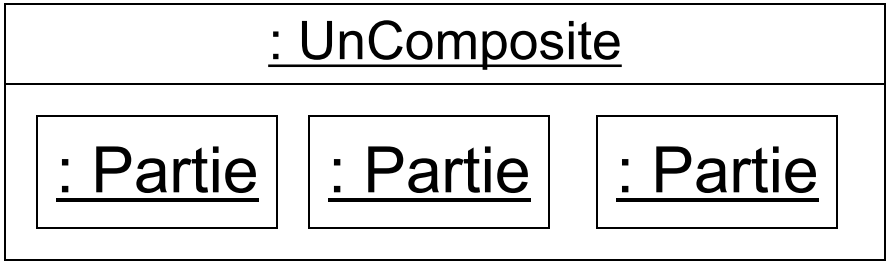
EXERCICE TD 5 RECETTE

1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

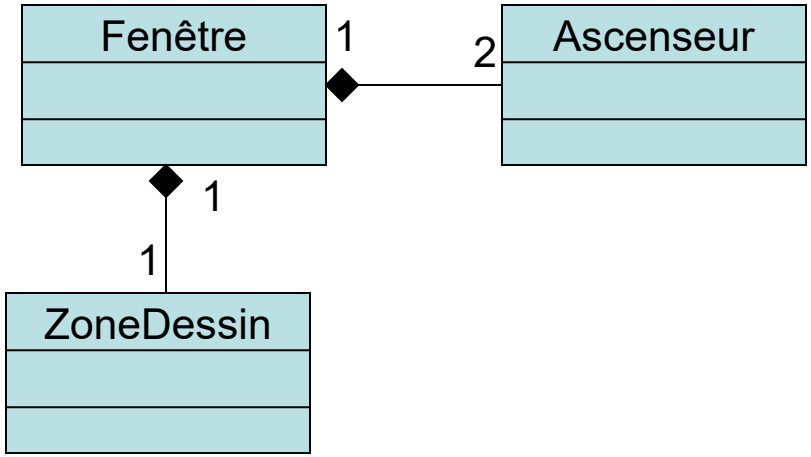
Pour votre culture

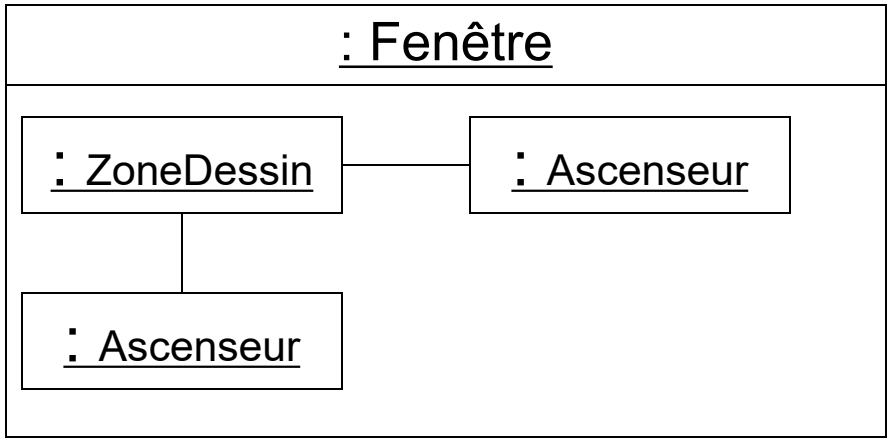
Objet composite

Les objets composés de sous-objets peuvent être représentés au moyen d'un objet composite.



Les objets composites sont instances de classes composites.

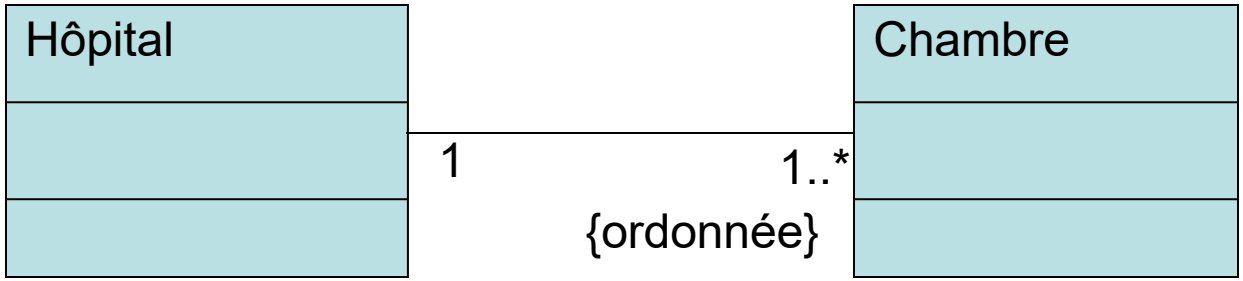




Contraintes sur les associations

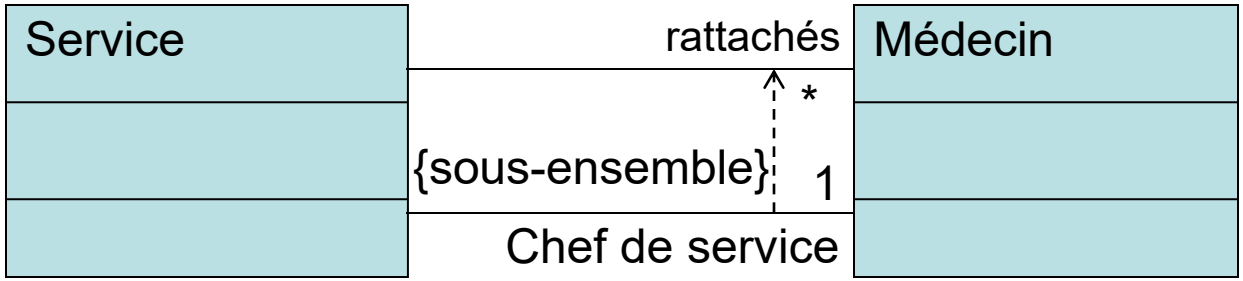
Toutes sortes de contraintes peuvent être définies sur une relation ou un groupe de relations. La multiplicité est une sorte de contrainte.

- Elles se représentent par des expressions placées entre accolades
- La contrainte {ordonnée} peut être placée sur le rôle pour spécifier qu'une relation d'ordre décrit les objets placés dans la collection.

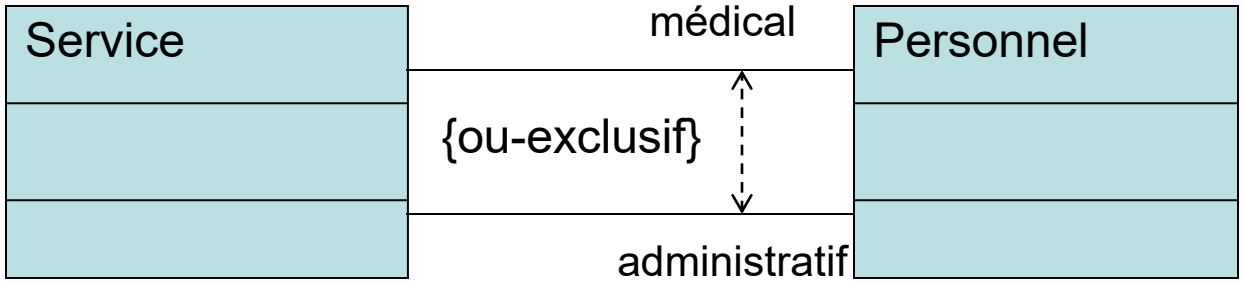


Contraintes sur les associations

- La contrainte {sous-ensemble} indique qu'une collection est incluse dans une autre collection

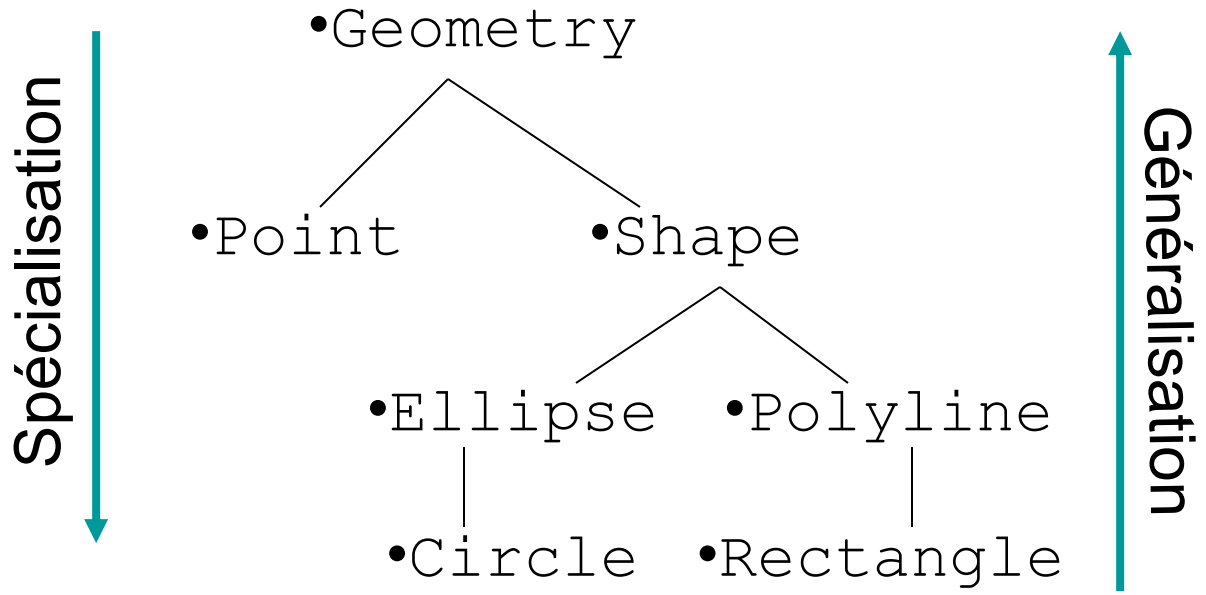


- La contrainte {ou-exclusif} précise que pour un objet donné, une seule association est valide



L'héritage / la généralisation

L'héritage est un mécanisme de spécialisation d'objets plus « généraux » vers des objets plus « spécifiques ». Les objets qui héritent sont une sous-catégorie de leur classe mère.



- `Circle` hérite
- de `Ellipse`
- `Rectangle` est
- une spécialisation
- de `Polyline`
- `Geometry`
- est l'abstraction
- la plus générale
- de tous
- ces objets

Hiérarchie de classes

Avec l'héritage, on met en évidence des relations de sous-typage.

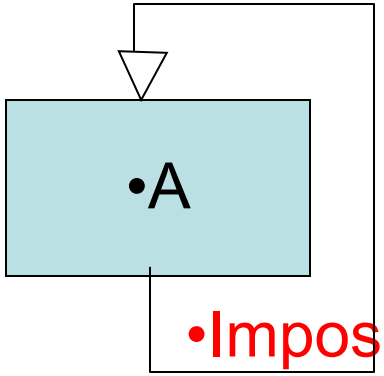
« Si une classe B hérite d'une classe A, alors aussi bien les opérations que la structure (=attributs) de A deviennent ceux de B » *(Jacobson, 1992)*

La généralisation signifie toujours « est un » ou « est une sorte de ». Ainsi, si Y hérite de X, on peut substituer une instance de X par unes instance de Y.

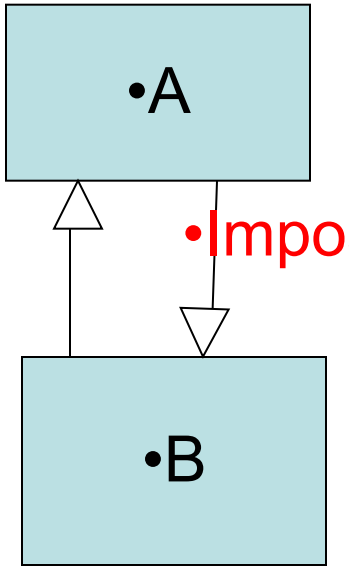
```
Geometry x = new Point();
```

Propriétés de la généralisation

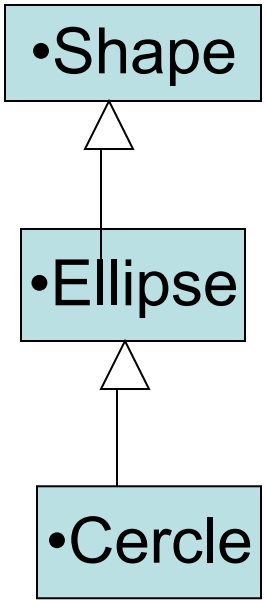
- Relation
- non réflexive



- Relation
- non symétrique



- Relation transitive

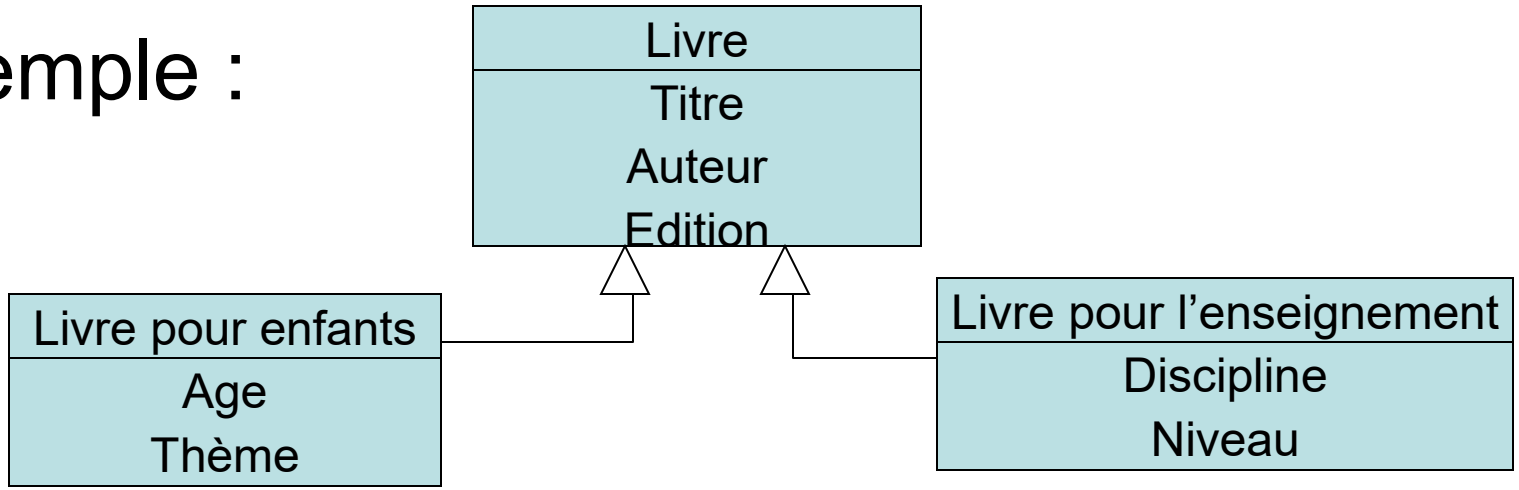


- Cercle est une Ellipse,
- puisque qu'il hérite de Ellipse.
- C'est aussi une forme
- géométrique (*Shape*)
- par transitivité

Des ensembles aux classes

- La généralisation des classes correspond à la relation d'inclusion des ensembles.
- Les objets instances d'une classe donnée sont décrits par la propriété caractéristique de leur classe, mais également par les propriétés caractéristiques de toutes les classes parents de leur classe.

Exemple :



Niveau de visibilité

Les membres d'une classe (attributs ou opérations) ont un niveau de visibilité

- Public (+) : visible pour tous les clients de la classe
- Protégé (*protected*) (#) : accessible uniquement pour les sous-classes
- Paquetage (*package*) (~) : accesible uniquement pour les classes du même paquetage (par défaut dans java)
- Privé (*private*) (-) : seulement la classe qui le définit peut l'utiliser

Critères pour choisir le niveau de visibilité :

- Compréhension
- Extensibilité
- Contexte

La classe abstraite

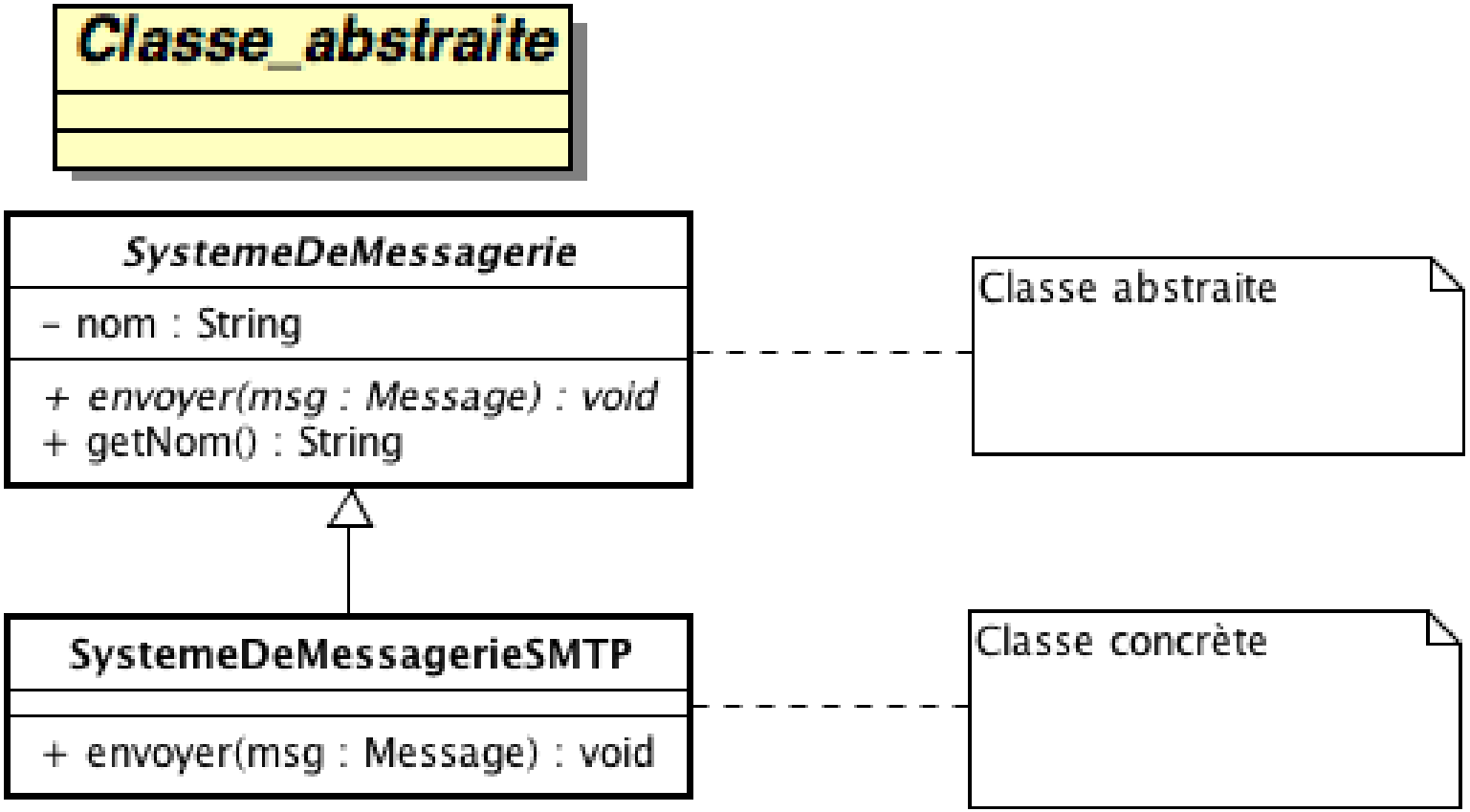
- Une classe abstraite (<< *abstract* >>) ne peut être instanciée.
- Une opération abstraite est une méthode déclarée sans implémentation.

Conséquences :

- Les instances d'une classe abstraite sont forcément des instances de ses sous-classes.
- Une classe abstraite ne donne pas directement des objets : il est nécessaire de la spécialiser pour obtenir des instances.
- Utilisée pour regrouper des attributs et opérations communes à plusieurs classes.

NB : en Java, l'*interface* est l'équivalent d'une classe abstraite avec seulement des opérations

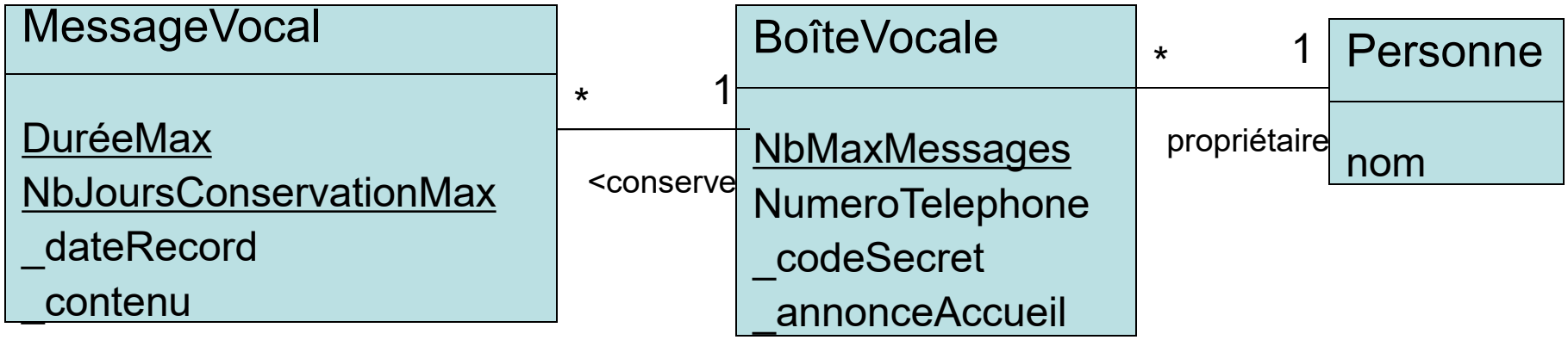
Classe abstraite en UML



Portée des membres d'une classe

La *portée* indique si une propriété s'applique à un seul objet, ou est partagée par toutes les instances de la classe.

- Si la portée est la classe, on dit que le membre (attribut ou opération) est statique.
- Statique s'utilise pour contenir l'extension d'une classe (l'ensemble d'objets de la classe)



Classe utilitaire

Une classe utilitaire est une sorte de boîte à outils : elle ne s'instancie pas, et tous ses membres publics (attributs et méthodes) sont statiques (de portée de classe)

Permet de regrouper un ensemble de valeurs et d'opérations.

<<utilitaire>> Math
+pi : float = 3.14159
+sin(x:float):float +cos(x:float):float +sqrt(x:float):float +pow(x:float, n:int):float

Math ne peut être instanciée.

On l'utilise ainsi :

pour un disque de rayon R

Aire = Math.pi * Math.pow(R, 2)

Interface

- Déclarer des méthodes que des classes concrètes doivent implémenter
- Eviter l'héritage

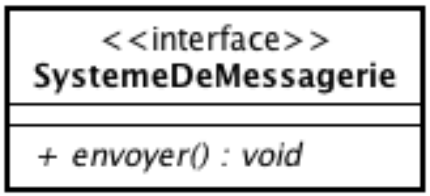
$\Rightarrow$ solution = l'interface

Mieux que l'héritage de classe abstraite : en Java, une classe peut implémenter (*implements*) n interfaces, n≥0, mais n'hérite (*extends*) que d'une classe directement

Une interface peut contenir des attributs : ils sont statiques, le plus souvent des constantes

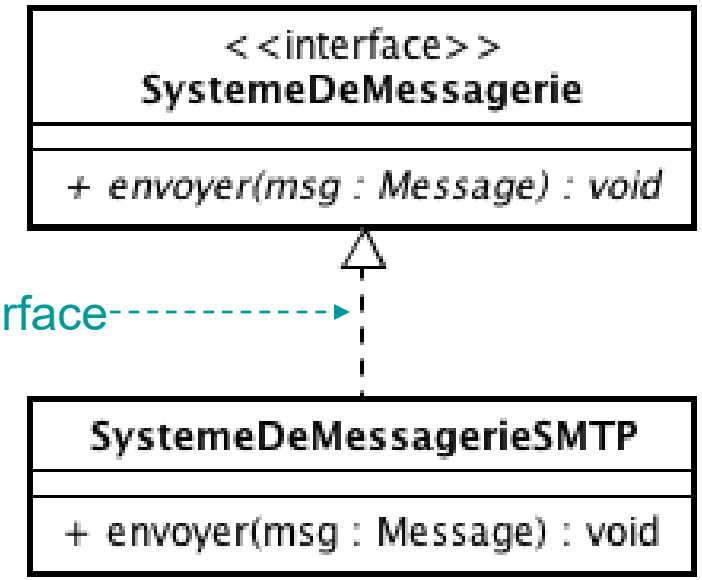
•2008

Interface : notation



•Notation de stéréotype

•Notation « à rotules »



•Réalisation de l'interface

•2008

Concepts à retenir pour les BDD

- **Objet** : entité qui combine les caractéristiques abstraites à partir du domaine système
- **Classe** : expression des caractéristiques communes aux objets
- **Encapsulation** : mécanisme permettant de masquer les détails de l'implémentation
- **Association** (par agrégation/composition) permet de construire des objets plus complexes et l'échanges de messages
  - Nom, rôles et multiplicités, sens
  - Classe d'association, association n-aire
  - Différence entre Composition et Agrégation
- **Héritage** : relations entre classes et sous-classes décrivant des ressemblances, et permettant de factoriser du code
- ~~Polymorphisme : mécanisme permettant l'invocation « aveugle » de méthodes sur un objet~~

EXERCICE TD 6
GARAGE

1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

8h30 – 12h20
Pause à 10h (20 min)

Plan Mercredi matin

1. Diagrammes de classes : 60 min
   - Exercices rapides (EXERCICES-5-UML.pdf)
2. Persistance : 30 min
   - Cours : 15 min
   - Démonstration sur fg1 : 15 min

3. Le fil rouge : 2h
   - Persistance

Exercices (de Bruno Bouzy)

Exercice 1 - 30 min

Dessiner les diagrammes (d'objets, de classes) correspondant aux situations suivantes :

(a) La France est frontalière de l'Espagne. Le Canada est frontalier des Etats-Unis.

(b) Un polygone est constitué de points. Un point possède une abscisse et une ordonnée.

(c) Une médiathèque possède des médias, empruntables par les abonnés de la médiathèque.

Exercices (de Bruno Bouzy)

Exercice 2 – 30 min

Classer les relations suivantes en généralisation, instanciation, agrégation ou association.

Argumenter les réponses.

(a) Un pays possède une capitale.

(b) Un philosophe qui dîne utilise une fourchette.

(c) Un joueur de rugby est un avant, un demi ou un arrière.

(d) Une équipe de rugby est composée de 8 avants, 2 demis et 5 arrières.

(e) Dédé programme son simulateur de vol en Java sur son PC.

(f) Java, C++, Eiffel sont des langages orientés objet.

(g) La Tour Eiffel a 3 étages et 3 millions de boulons.

(h) L'agrégation est un examen.

Mercredi matin
8h30-12h30

PERSISTANCE DU MODÈLE UML OBJET EN BDD RELATIONNELLE

De UML vers SQL

Rappels : base de données

- Les données sont stockées dans des relations.
- Une relation est un ensemble de T-uple, qui est défini par un ou plusieurs attributs.
- Dans la pratique, la relation est en fait la table, un
- T-uple est une ligne (ou enregistrement), et les attributs sont les colonnes.
- Il n'y a pas de doublon dans une relation.

id_newsletter	Sujet	DateEnvoie	Contenu	id_rubrique
25	Newsletter N°25	11/01/2001	Texte 25	10
26	Newsletter N°26	21/01/2001	Texte 26	20

- Reference une autre table

Exemple de relation

1. La table PRODUIT

PRODUIT	NPRO	NOMP	QTES	COULEUR
	7	parapluie	10	noir
	13	chapeau	5	vert
	4	valise	10	noir
	8	valise	12	marron

2. La relation Produit (**NPRO**, NOMP, QTES, COULEUR) a 4 attributs :
   - NPRO : numéro identifiant le produit uniquement
   - NOMP : nom du produit
   - QTES : quantité en stock
   - COULEUR : couleur (type énuméré)

Les clés

1. Primaire (PK) : c'est un ensemble d'attributs(colonnes) qui identifient uniquement le tuple (la ligne) dans la relation (table). Cela peut n'être qu'un attribut. Elle est toujours soulignée.

2. Etrangère (FK) : c'est un attribut (ou un ensemble d'attributs) qui référence d'autres tuples (lignes) dans une autre relation (table) : l'attribut doit être clé dans la table cible

Exemple de clé étrangère (FK)

Des produits en stocks sont achetés :

- NPRO : PK sur Produit(NPRO, NOMP, QTES, COULEUR)
- NACH : PK sur Achat(NACH, NPRA, QTEA, FOURNISSEUR)
- NPRA : FK de Achat vers Produit(NPR)

•Table cible

PRODUIT	NPRO	NOMP	QTES	COULEUR
	7	parapluie	10	noir
	13	chapeau	5	vert
	4	valise	10	noir
	8	valise	12	marron

•→

•référence

•Table source

ACHAT	NACH	NPRA	QTEA	FOURNISSEUR
	1	7	10	Labaleine
	2	4	5	Labaleine
	5	8	6	Lenoir

Passage objet - relationnel (1/4)

1. Règle 1 : toute entité (classe) 1 se transforme en une relation (table)
   - Chaque propriété de l'entité devient un attribut de cette relation, et donc une colonne de la table correspondante.
   - L'identifiant de l'entité devient la clé primaire (PK) de la table correspondante.
   - On ignore les méthodes de la classe

Exemple

Le cheval et une personne

Personne
nom
prénom
téléphone
adresse

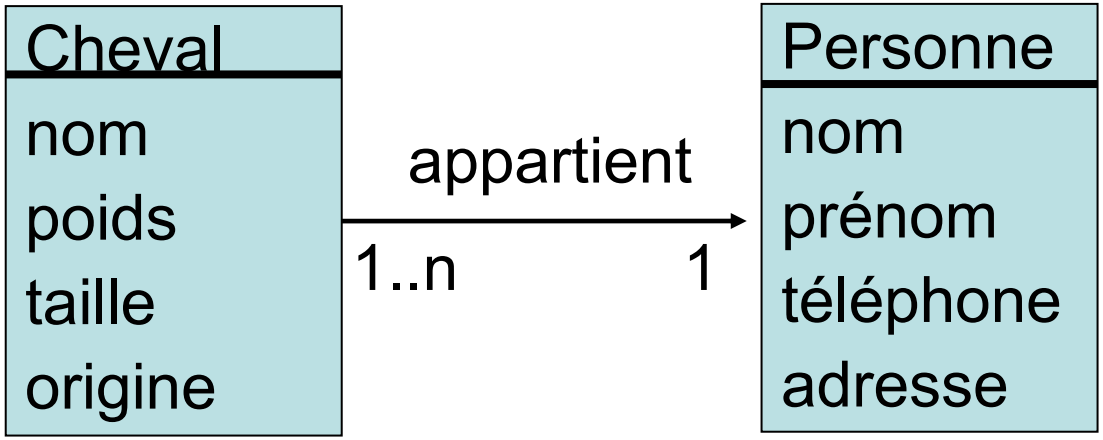
Personne	nom	prénom	téléphone	adresse
	Tourte	Madelaine	0235728945	20 rue des Lilas, Rouen
	Tourte	Louis	0231442382	13 rue des abattoirs, Caen
	Luz	Maria	0142224964	7 rue Denfert, Paris

Cheval
nom
poids
taille
origine

Cheval	nom	poids	taille	origine
	Blanco	670	137	Arabe
	2	580	125	Islande
	5	530	120	Bretagne

Passage objet - relationnel (2/4)

1. Règle 2: pour les associations binaires de cardinalité (X,1) - (X,n), X=0 ou X=1, entre 2 classes, la clé primaire de la table à la cardinalité (X,n) devient une clé étrangère dans la table à la cardinalité (X,1)



Exemple

Une personne possède un cheval

Personne	nom	prénom	téléphone	adresse
	Tourte	Madelaine	0235728945	20 rue des Lilas, Rouen
	Tourte	Louis	0231442382	13 rue des abattoirs, Caen
	Luz	Maria	0142224964	7 rue Denfert, Paris

reference

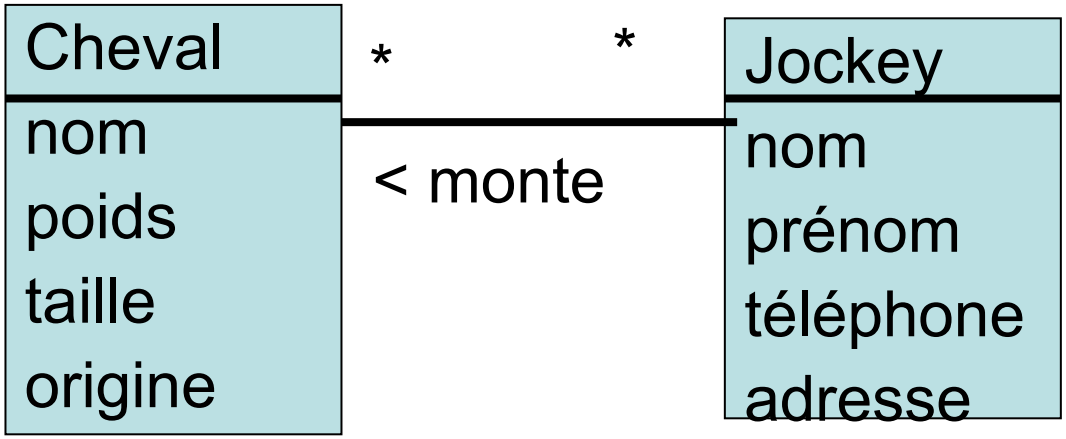
Cheval	nom	poids	taille	origine	Propriétaire_nom	Propriétaire_prénom
	Blanco	670	137	Arabe	Tourte	Madelaine
	2	580	125	Islande	Tourte	Louis
	5	530	120	Bretagne	Luz	Maria

Passage objet - relationnel (3/4)

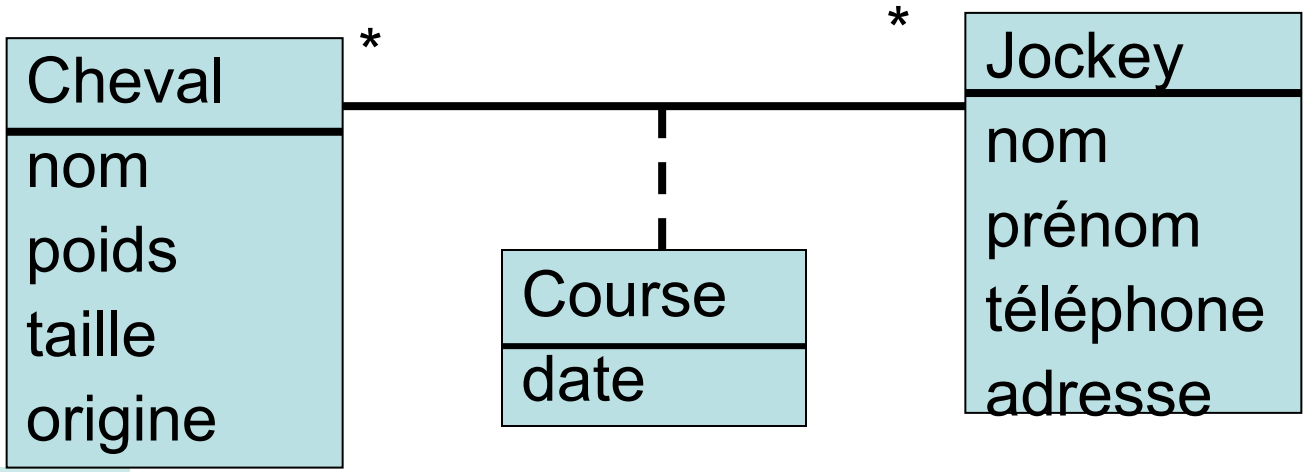
1. Règle 3 : pour les associations binaires aux cardinalités (X,n) - (X,m), X=0 ou X=1
   - Il y a création d'une table supplémentaire (table d'association) ayant comme clé primaire une clé composée des identifiants des 2 entités (classes). On dit que la clé primaire de la nouvelle table est la concaténation des clés primaires des deux autres tables.
   - Si l'association porte des attributs propres (classe d'association), ils deviennent des attributs pour la nouvelle table.

Exemple

Sans classe d'association



Avec classe d'association



Un jockey monte un cheval

Jockey	nom	prénom	téléphone	adresse
	Tourte	Madelaine	0235728945	20 rue des Lilas, Rouen
	Tourte	Louis	0231442382	13 rue des abattoirs, Caen
	Luz	Maria	0142224964	7 rue Denfert, Paris

•reference

Monte	nom	prénom	chevalNom
	Tourte	Madelaine	Blanco
	Tourte	Louis	Tornade
	Luz	Maria	Blanco

•reference

Cheval	nom	poids	taille	origine
	Blanco	670	137	Arabe
	Tornade	580	125	Islande
	Panache	530	120	Bretagne

Un jockey et un cheval participent à une course

Jockey	nom	prénom	téléphone	adresse
	Tourte	Madelaine	0235728945	20 rue des Lilas, Rouen
	Tourte	Louis	0231442382	13 rue des abattoirs, Caen
	Luz	Maria	0142224964	7 rue Denfert, Paris

•reference

Course	nom	prénom	chevalNom	Date
	Tourte	Madelaine	Blanco	12 janvier 2003
	Tourte	Louis	Tornade	24 Juillet 2008
	Luz	Maria	Blanco	3 Mars 2004

•reference

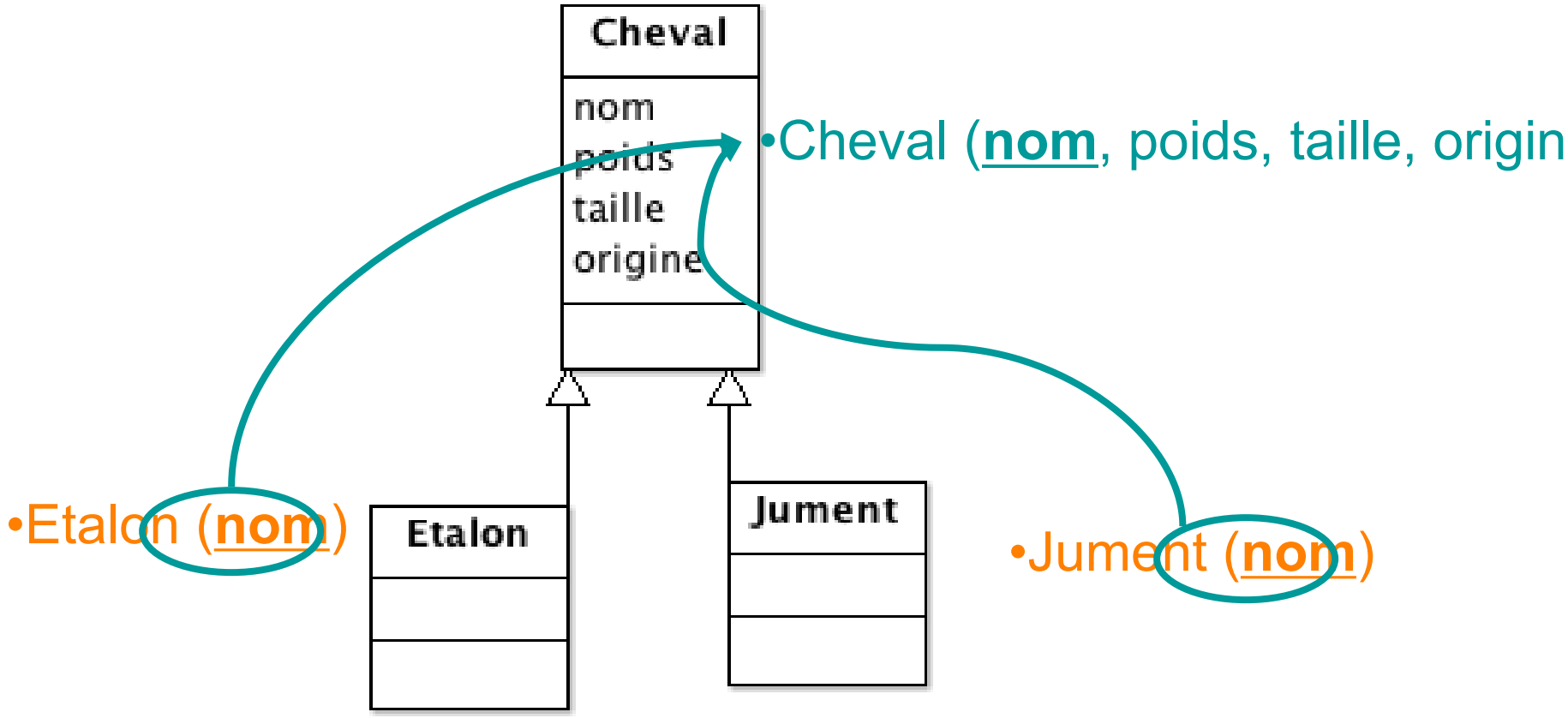
Cheval	nom	poids	taille	origine
	Blanco	670	137	Arabe
	Tornade	580	125	Islande
	Panache	530	120	Bretagne

Passage objet - relationnel (4/4)

- Règle 4 : La relation **d'héritage** est transformée en une association dont la clé primaire se situe au niveau de la table correspondant à la surclasse et les clés étrangères correspondantes au niveau des sous-classes

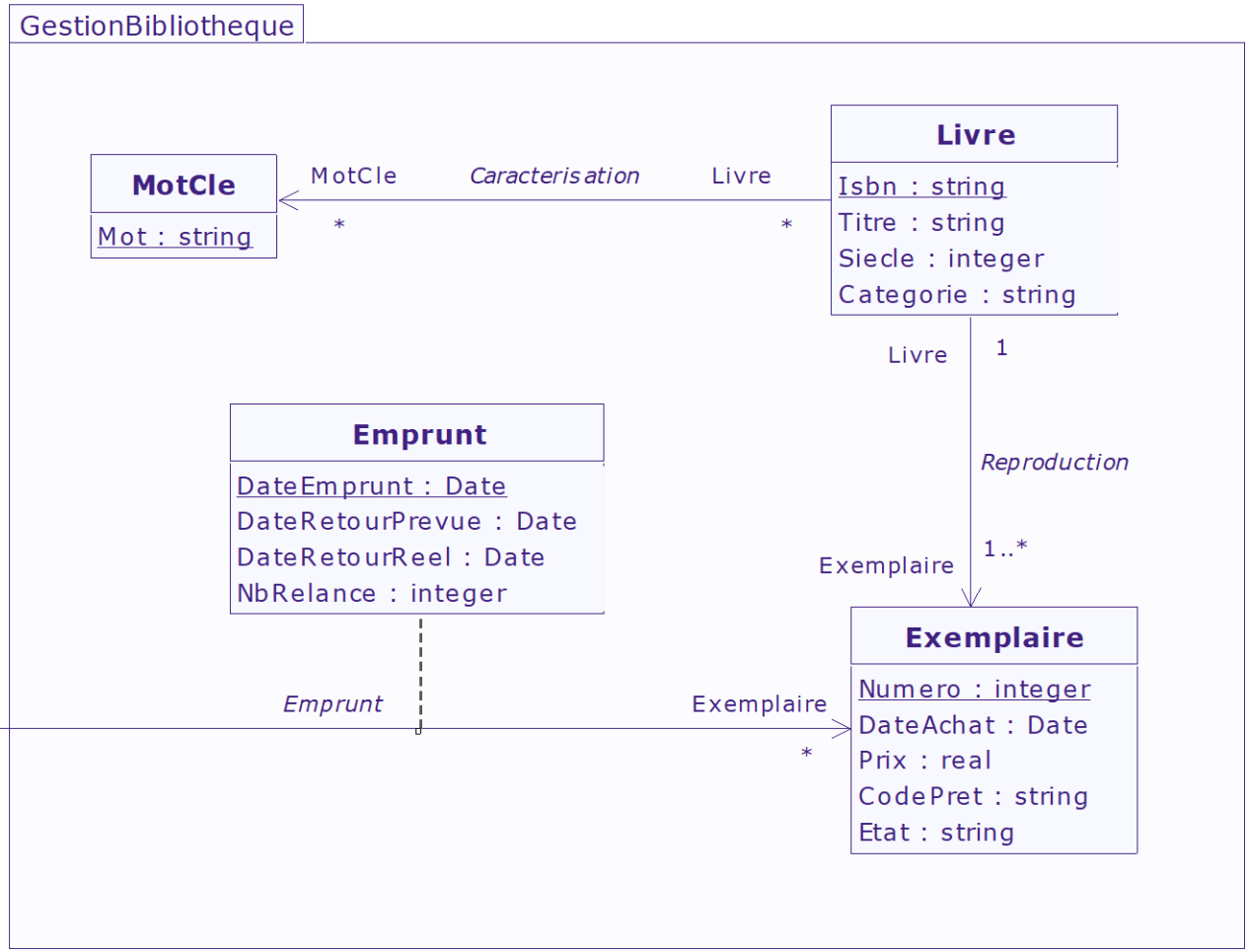
Exemple héritage

Cheval surclasse Etalon et Jument



La bibliothèque

Les ***attributs soulignés sont les clés primaires*** des tables auxquelles ils sont attachés.



Règles

1. Une classe = une table
2. Ses attributs = Ses colonnes
3. Trouver la clé primaire (PRIMARY KEY)
    1. Ou créer un identifiant unique pratique (SERIAL)
4. Une relation ?
    1. Une table nouvelle si N…M ou si classe d'association
    2. Sinon, une contrainte
        1. FOREIGN KEY
        2. CHECK

Démo

1. Connexion sur Postgres avec Dbeaver
2. Déroulé du script de création
   - 01 - 2.3 - Fg1 - Script de Creation Base de données
3. Déroulé du script d'import de données
   - 01 - 2.4 - Fg1 - Script de Chargement données

Attributs	Contraintes
Age	0<Age<120
Siecle	0<Siecle<21
Etat	Etat = 'BON','ABIME' ou 'EN_REPARATION'
Nb_Relance	Nb_Relance = 1, 2 ou 3
Code_Pret	Code_Pret = 'EXCLU', 'EMPRUNTABLE' ou 'CONSULTABLE'

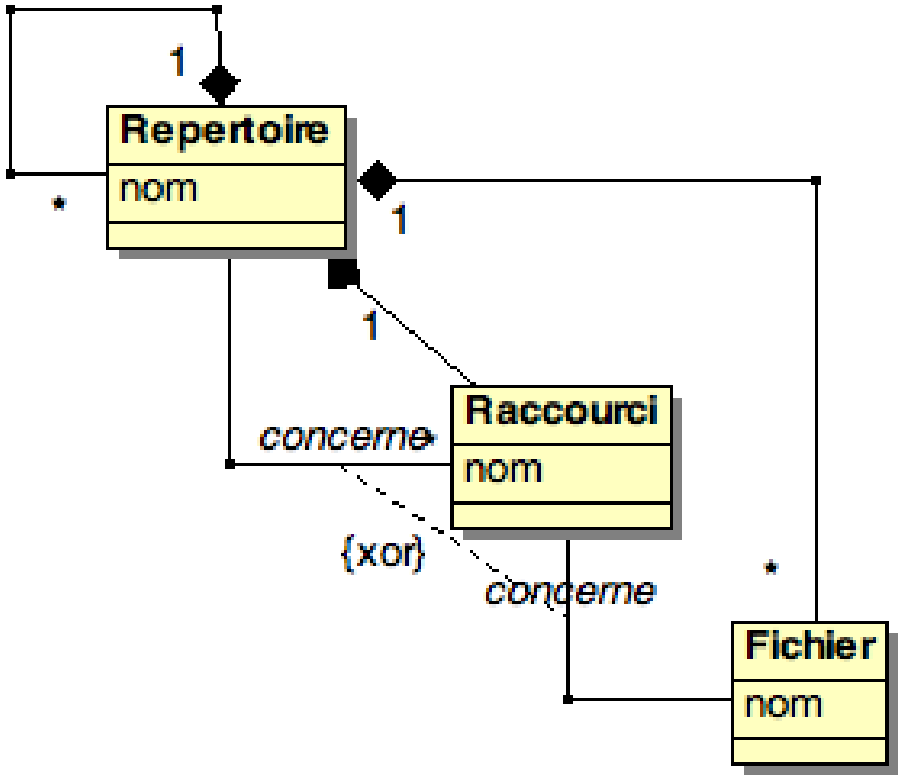
Fil rouge – 2H

Proposer l'implémentation SQL correspondante

Et pour une association réflexive ?

Le système de gestion de fichier

- Les fichiers, les raccourcis et les répertoires sont contenus dans des répertoires et possèdent un nom
- Un raccourci peut concerner un fichier OU un répertoire.



Mercredi 14h – 17h50
Pause à 15h30 (20 min)

HÉRITAGE DANS POSTGRES

Cours, démonstration et pratique avec André Mirallès

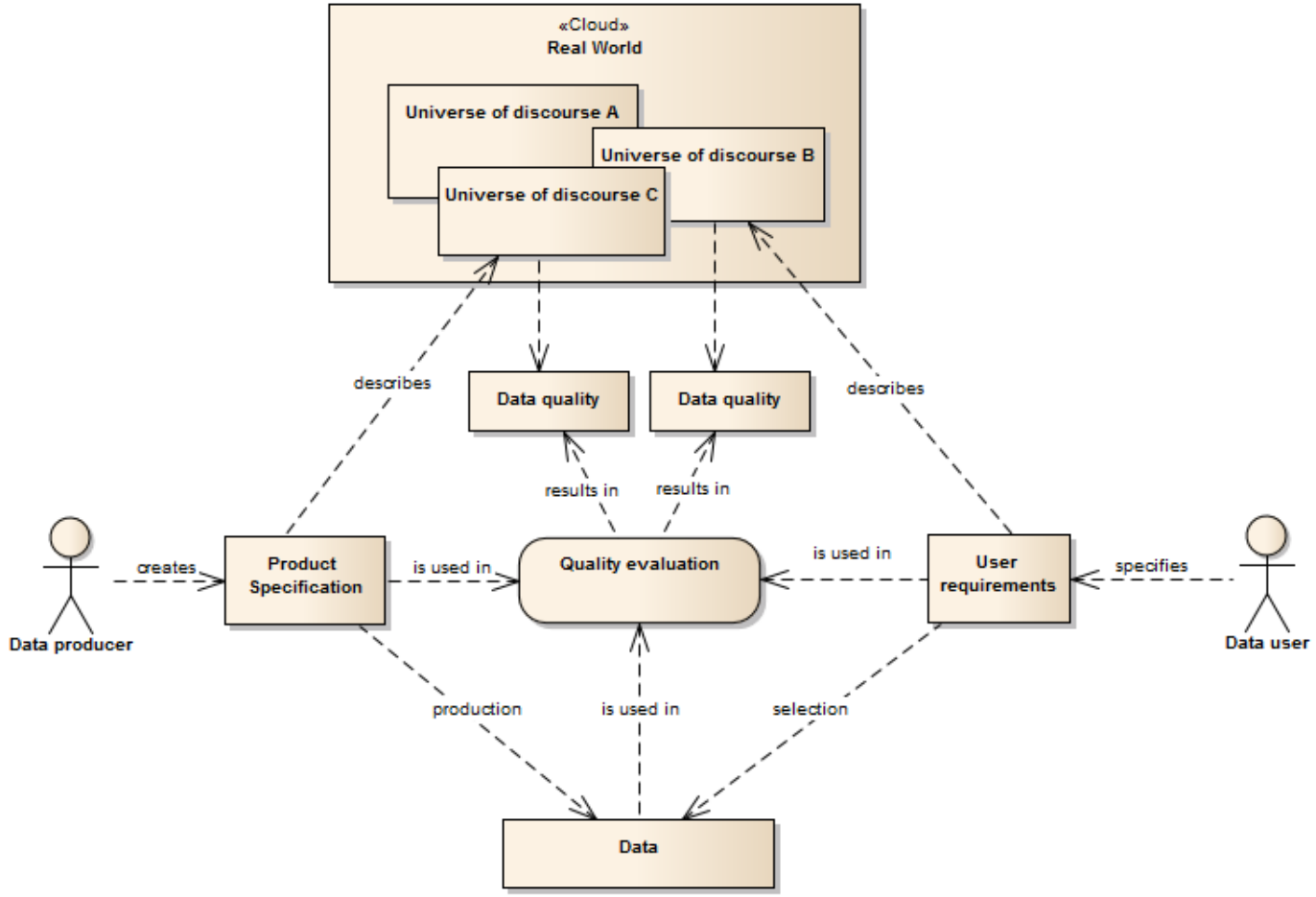
Jeudi 8h30 – 12h
Pause à 10h (15 min)

LIRE UML

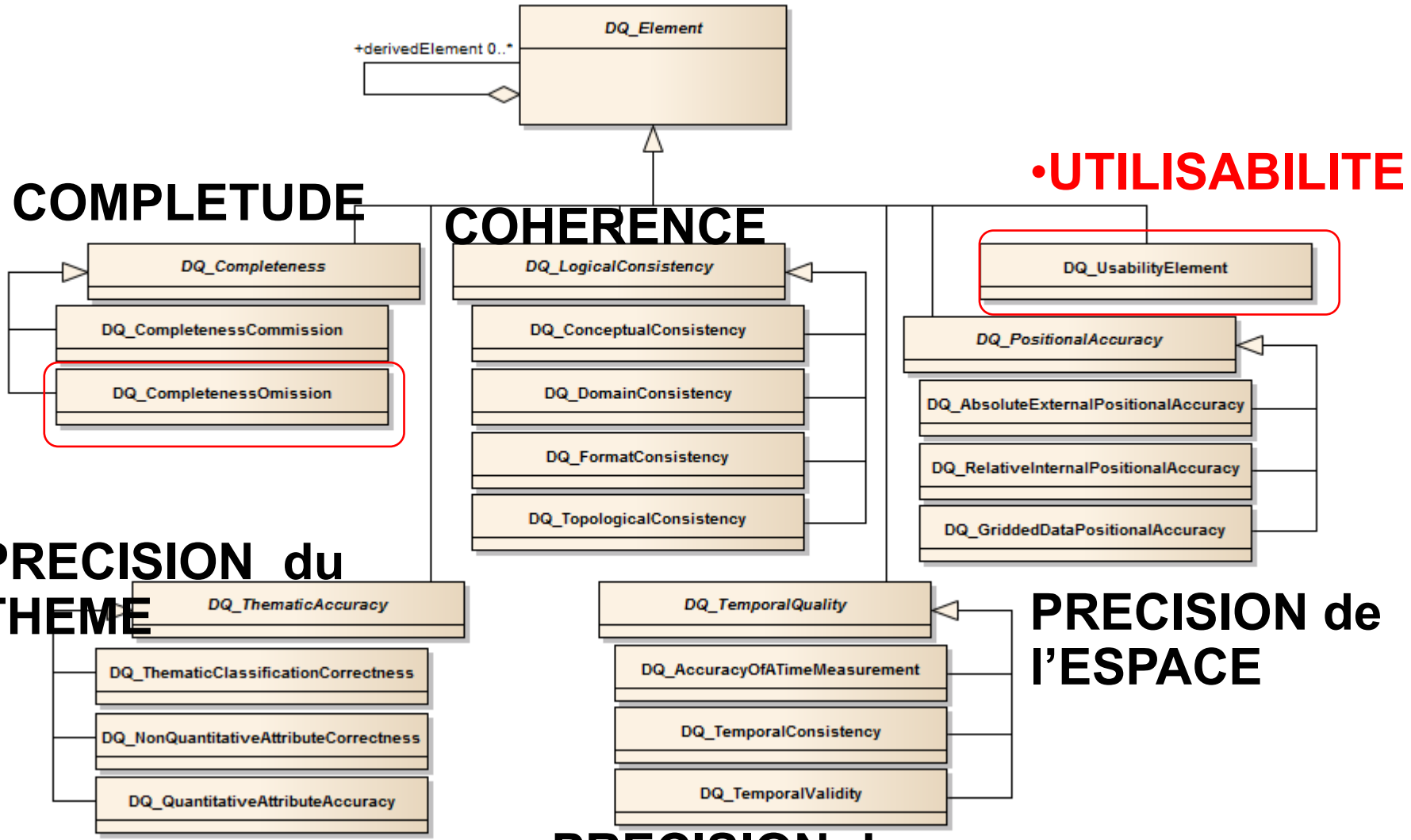
Souvent dans les normes ISO par exemple

Norme ISO 19157

http://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/



Les critères qualité en 2018
Norme ISO 19157

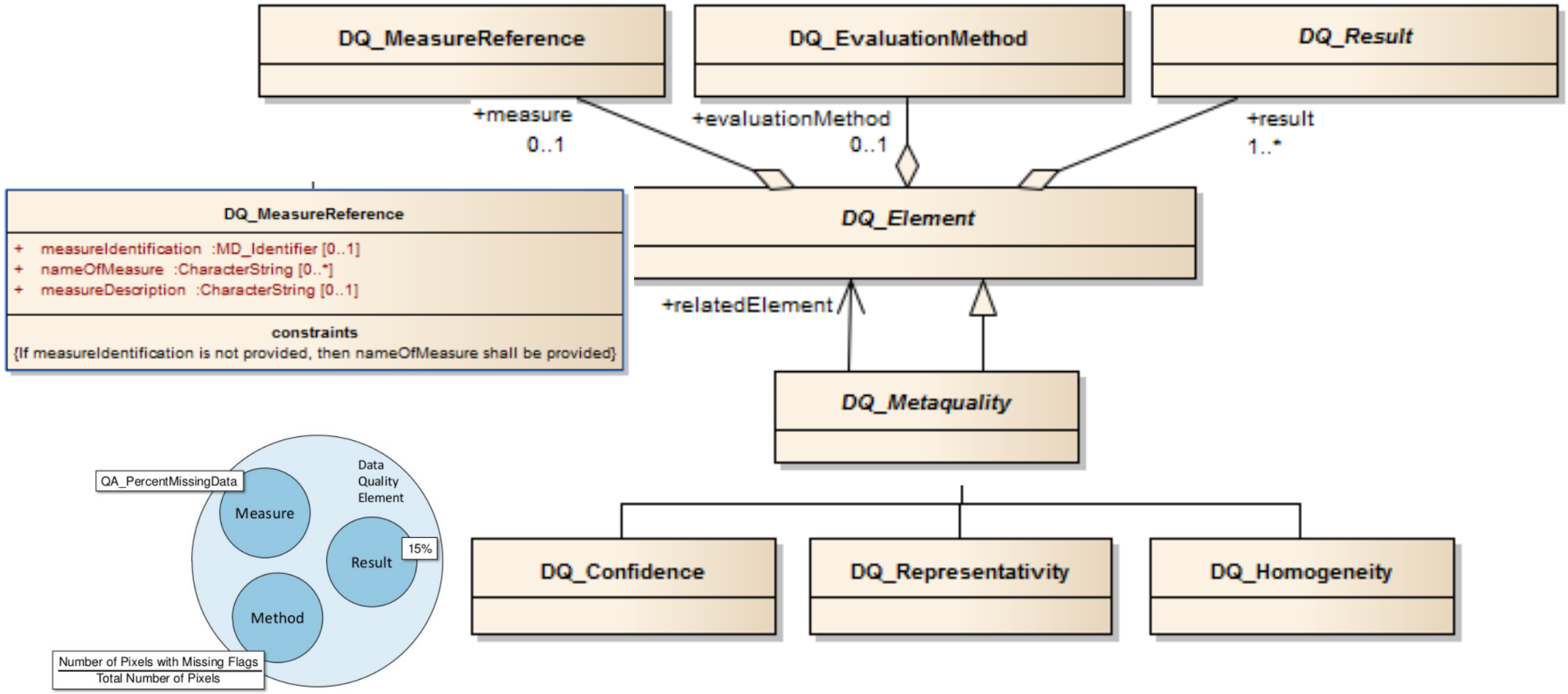


2013 : Norme ISO 19157

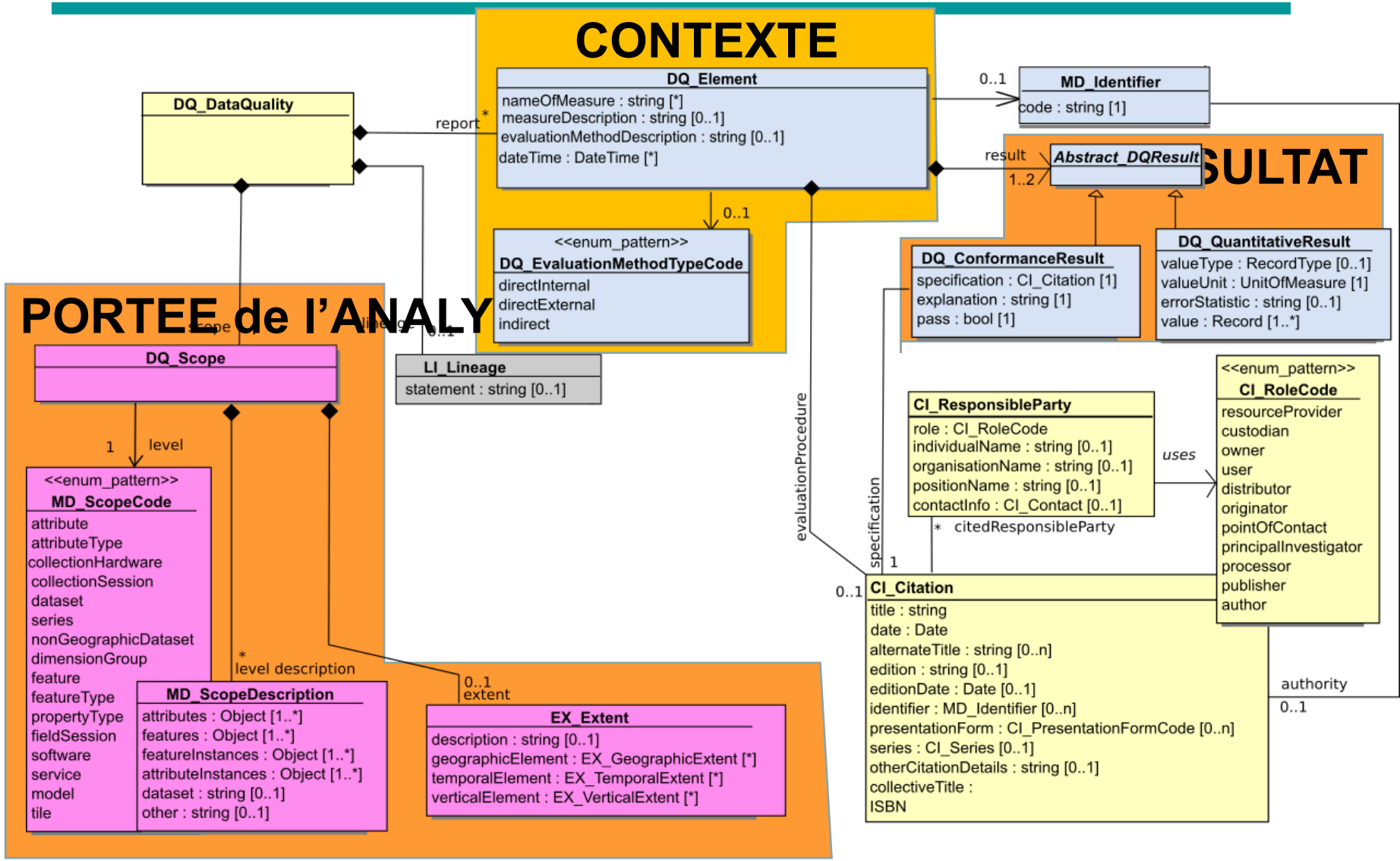
http://inspire.ec.europa.eu/data-model/approved/r4618-ir/html/

https://www.iso.org/fr/standard/32575.html

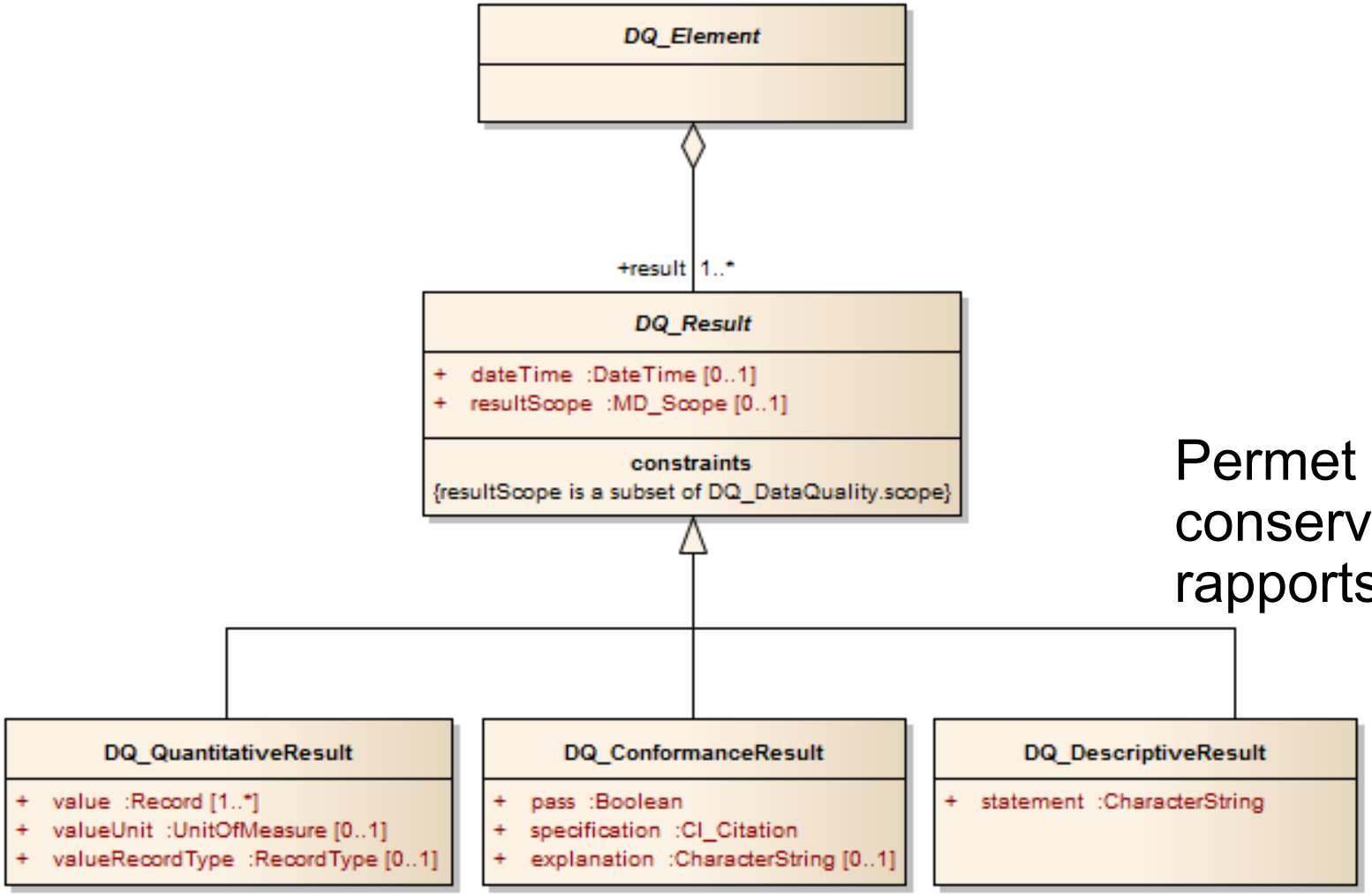
DQ_Element



L'élément qualité ISO 19115



DQ_Result



Permet de conserver des rapports écrits

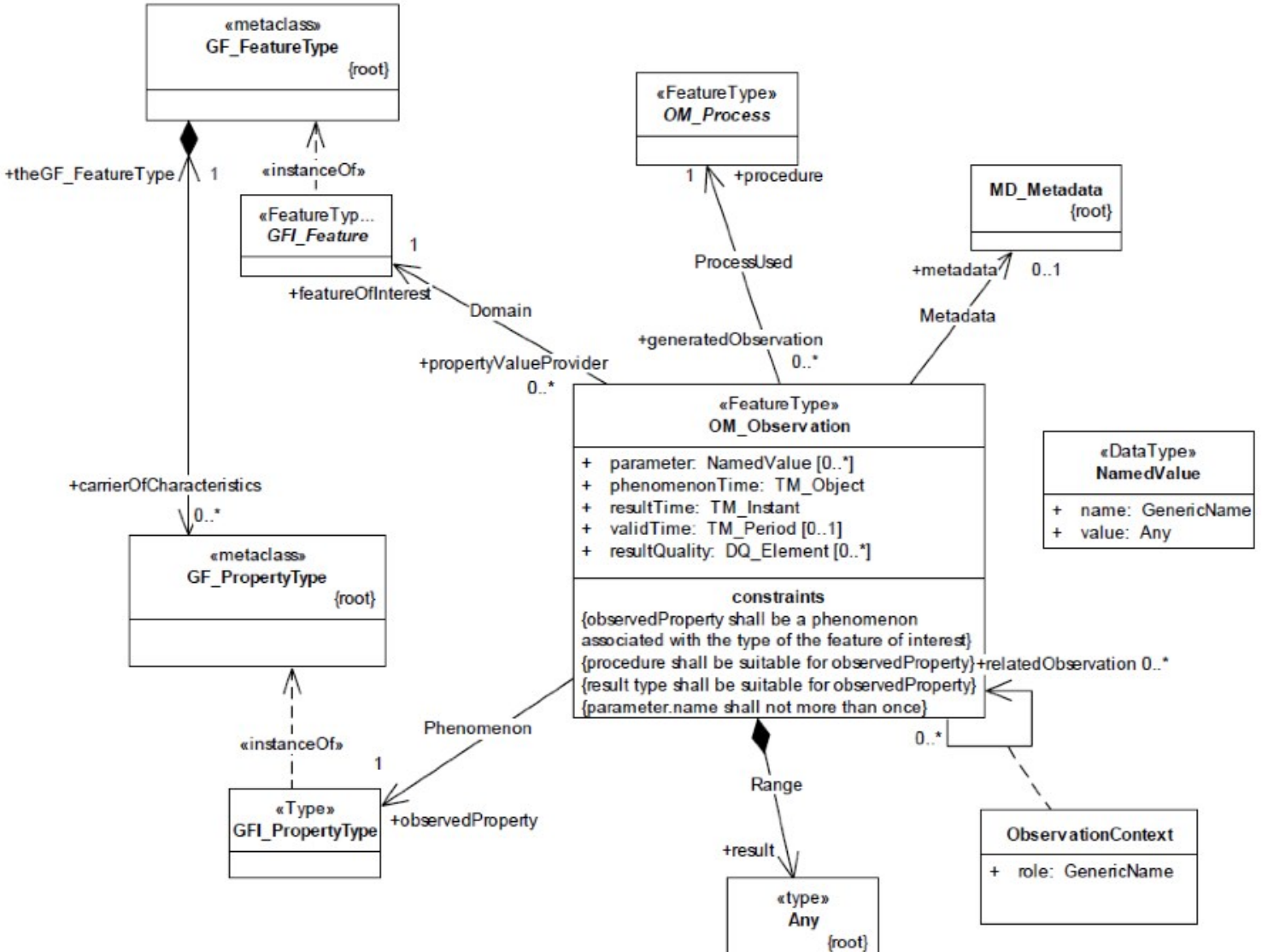
O&M

Observation and Measurement - norme ISO 19156

Spécifications

1. **Earth Observation Metadata profile of Observations & Measurements**
   http://docs.opengeospatial.org/is/10-157r4/10-157r4.html
2. http://schemas.opengis.net/gml/3.1.1/base/gml.xsd
3. http://www.mediamaps.ch/ogc/schemas-xsdoc/
   - http://www.mediamaps.ch/ogc/schemas-xsdoc/sld/1.1.0/
   - http://www.mediamaps.ch/ogc/schemas-xsdoc/sld/1.2/
4. https://3d.bk.tudelft.nl/hledoux/fieldgml/doc/withgml/dynamicFeature.xsd.html
5. https://3d.bk.tudelft.nl/hledoux/fieldgml/doc/withgml/fieldgml.xsd.html
6. http://www.datypic.com/sc/niem21/ss.html
7. http://www.datypic.com/sc/niem21/s-dynamicFeature.xsd.html
8. https://schemas.wmo.int/wmdr/1.0RC4/documentation/schemadoc/schemas/dynamicFeature_xsd/schema-overview.html
9. http://schemas.opengis.net/gml/3.2.1/

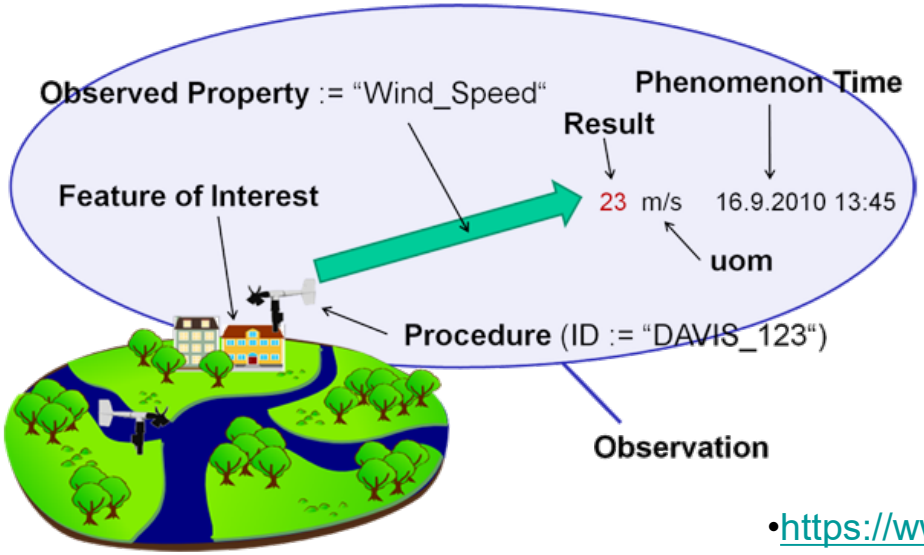
An ***observation*** is an event that estimates an ***observed property*** of some ***feature of interest*** using a specified ***procedure*** and generates a ***result***.



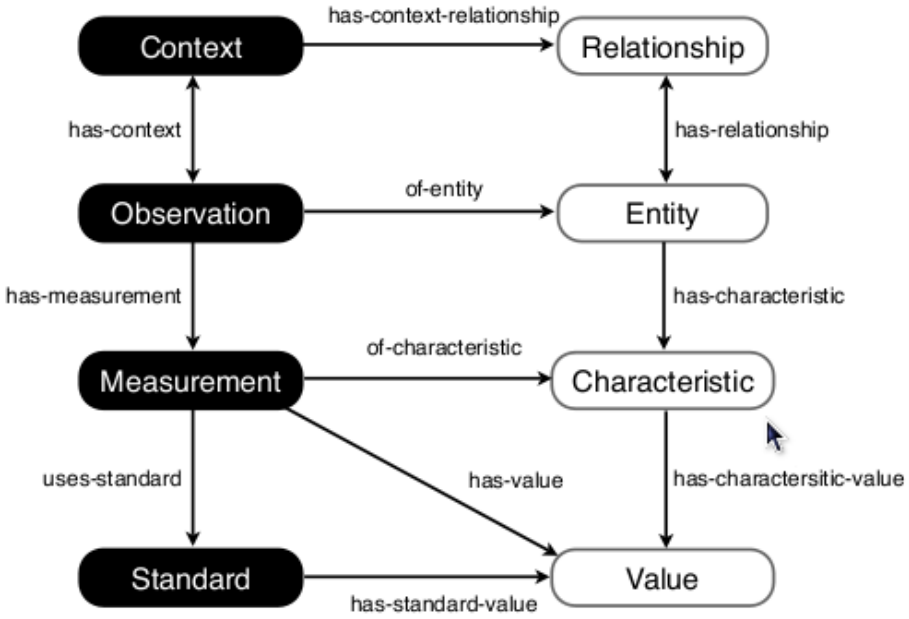
O&M / OBOE

1. O&M

Une observation est un événement dont le résultat est une estimation de la valeur d'une propriété de l'entité d'intérêt, obtenu à l'aide d'une procédure.



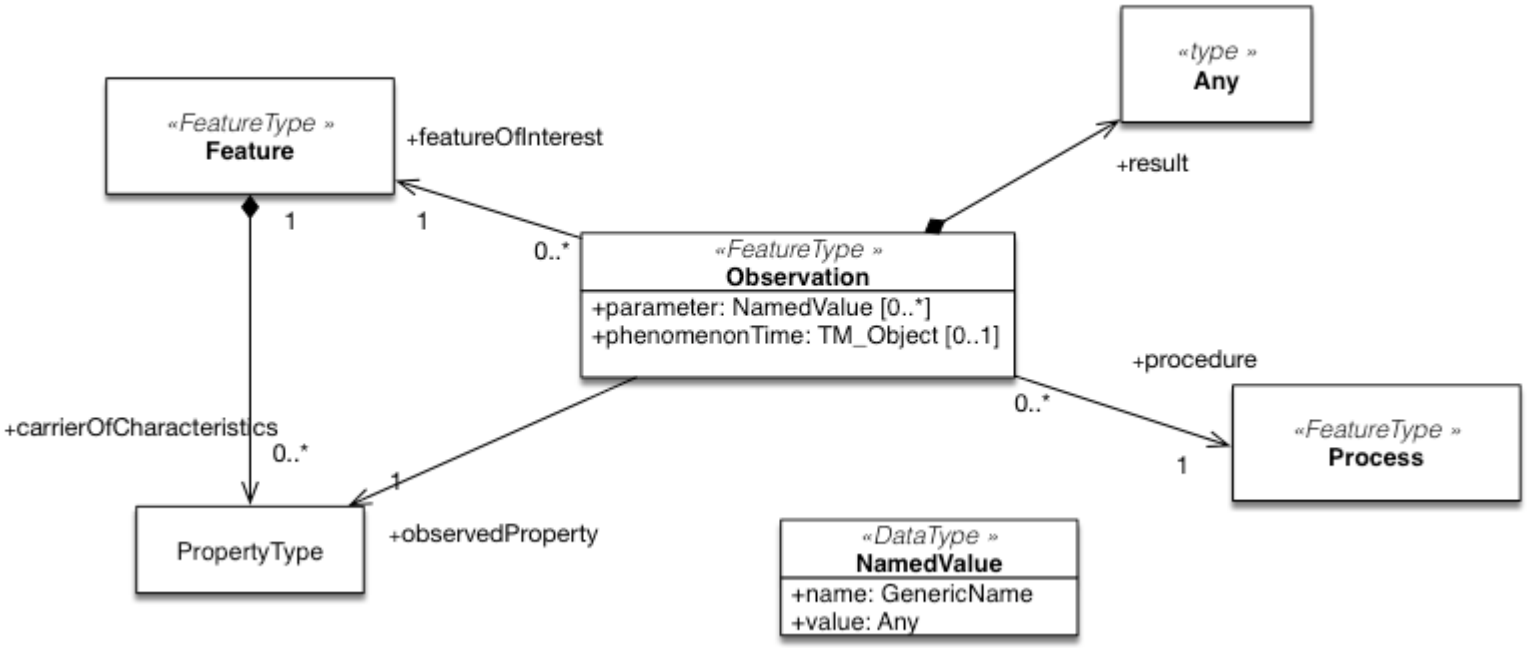
• OBOE



- https://www.researchgate.net/publication/223833893
- *An ontology for describing and synthesizing ecological observation*
- Article in Ecological Informatics · October 2007
- DOI: 10.1016/j.ecoinf.2007.05.004 · Source: DBLP

O&M

Une observation est un événement dont le résultat est une estimation de la valeur d'une propriété de l'entité d'intérêt, obtenu à l'aide d'une procédure.

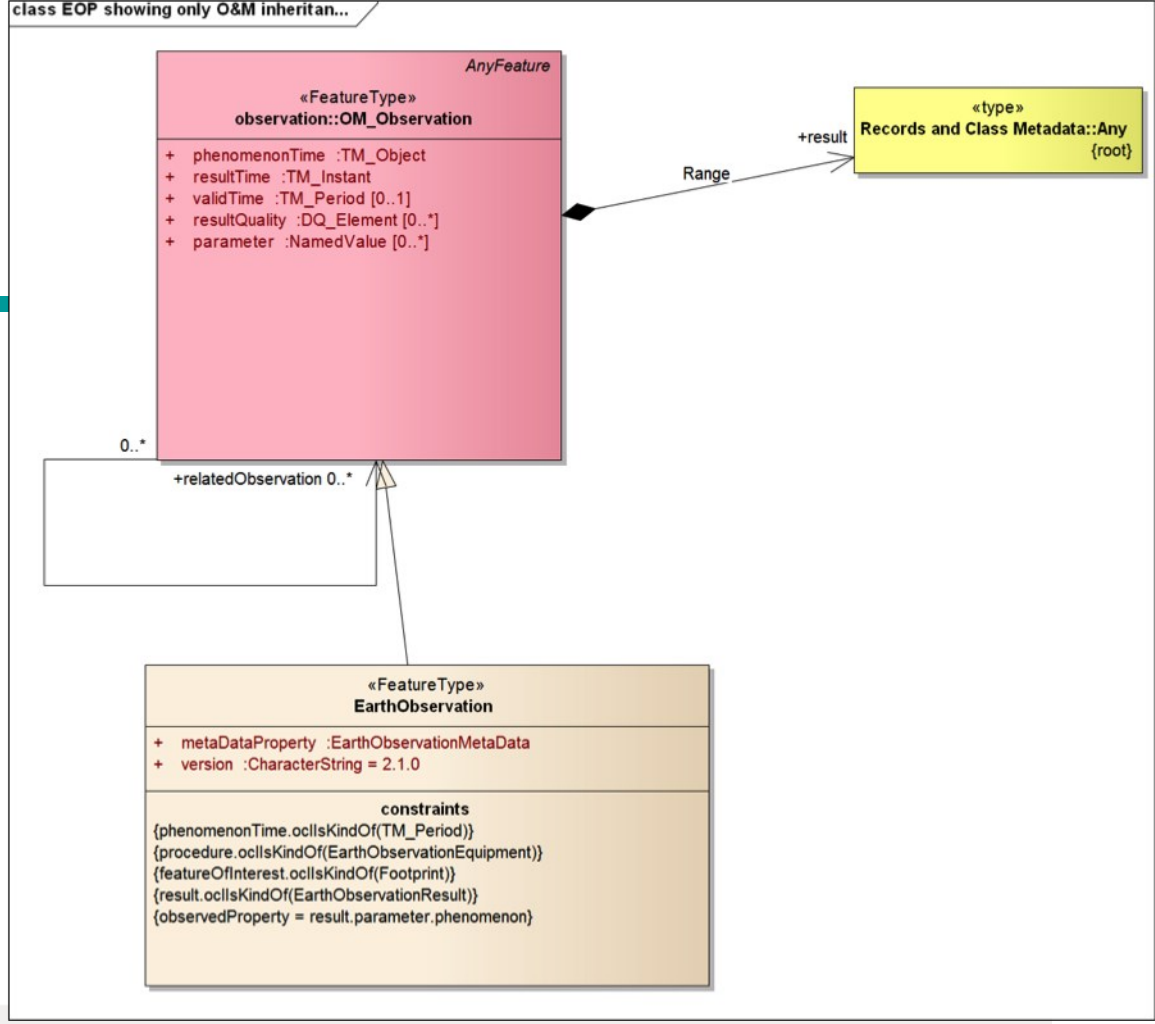


Source : POPS AP Observatoire

https://www-iuem.univ-brest.fr/pops/attachments/1616

Etendre le schéma

Un type d'observation par exemple




```
<element name="EarthObservation" substitutionGroup="om:OM_Observation" type="eop:EarthObservationType"/>
    <complexType name="EarthObservationType">
        <complexContent>
            <extension base="om:OM_ObservationType">
                <sequence>
                    <element name="metaDataProperty" type="eop:EarthObservationMetaDataPropertyType"/>
                    <element name="version" type="string"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

UTILISER UML POUR MODÉLISER DES CONNAISSANCES

Utile aussi pour réfléchir à la formalisation d'ontologies

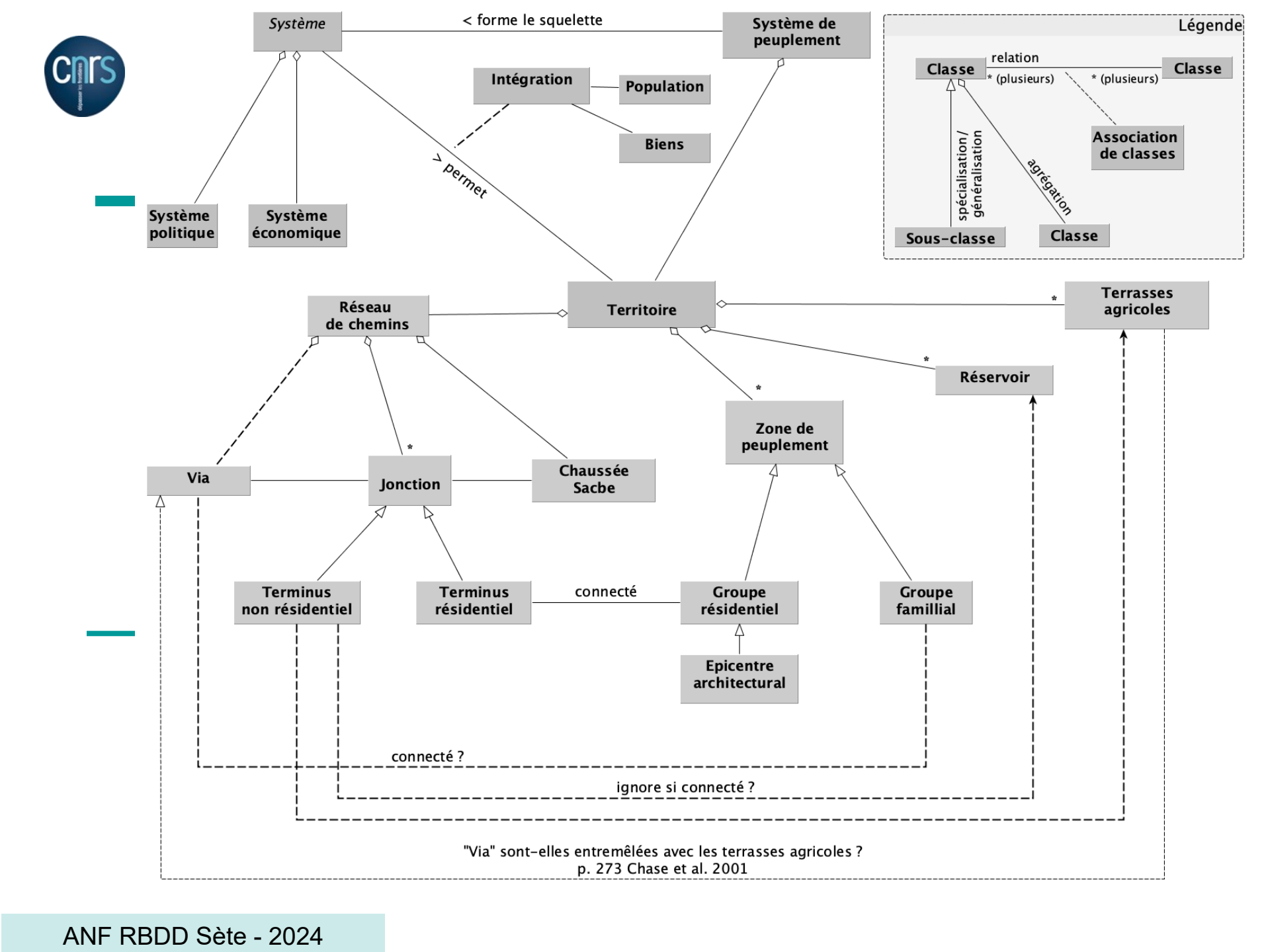
Explorer les processus de mobilité passée

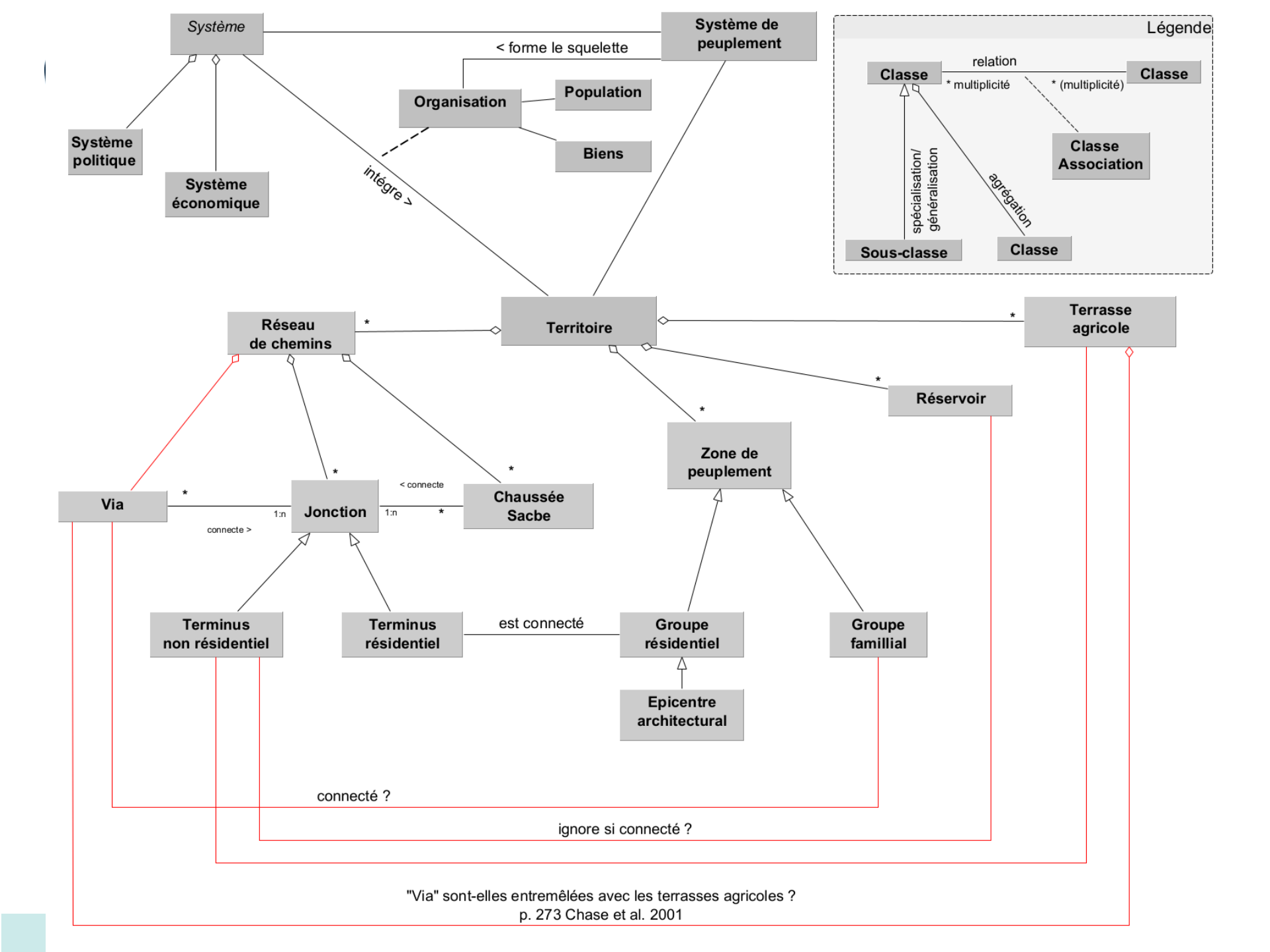
Laure Nuninger, T. Libourel, X. Rodier et al., à paraitre…

Dans la région de Caracol, de nombreuses voies ont été découvertes grâce à l'analyse du MNT dérivé du LiDAR. De nouveaux segments de chaussée s'ajoutent ainsi à l'ensemble de données compilées à partir d'observations précédentes réalisées sur le terrain et à partir d'images satellites (A.F. Chase & Chase, 2001). L'étude des données altimétriques LiDAR a révélé une grande partie du **système viaire** principal reliant l'épicentre principal de Caracol à d'autres sites, appelés *termini* et compris comme des centres secondaires, potentiellement résidentiels (Arlen F. Chase et al., 2011).

Le MNT LiDAR montre également qu'il existe une continuité d'habitats et de terrasses entre l'épicentre et ces *termini*. En effet, cette vaste zone de peuplement est caractérisée par de petits groupes de maisons organisés autour d'une petite place, qui sont intégrés dans la zone agraire, selon un modèle reconnu dans d'autres régions mayas. Ces unités de peuplement ne sont pas directement reliées à l'épicentre ou aux centres secondaires par des chaussées (appelées *sacbes*). Par ailleurs, il est fort plausible que les éléments composant le système agraire ont permis d'établir des liens solides entre ces habitations et les principaux centres résidentiels (Arnauld & Michelet, 2004). Dans l'étude de cas de Caracol, on a pu noter que les auteurs font référence à des entités qu'ils appellent « vias » et caractérisent comme étant des « routes plus informelles et plus courtes » (A.F. Chase & Chase, 2001), qui relient des groupes familiaux (ménages) aux principales chaussées, ou qui relient « d'importants groupes résidentiels directement à divers *termini* non résidentiels ». Les auteurs notent qu'à Caracol, « de nombreuses chaussées (...) sont enchevêtrées avec des terrasses agricoles » et que les vias peuvent être principalement identifiées grâce à une cartographie détaillée de ces terrasses. Alors que « seulement une demi-douzaine de vias ont été formellement identifiées comme rattachées aux longues chaussées [du réseau viaire] de Caracol », ils soulignent qu'« une cartographie détaillée des terrasses permettrait sans aucun doute de trouver d'autres exemples » (A.F. Chase & Chase, 2001). Puisque les vias identifiées semblent fournir un accès direct aux principales chaussées, les auteurs suggèrent que ces voies étaient en partie utilisées à des fins quotidiennes, pour faciliter les déplacements et la communication des groupes familiaux.

Ce système de peuplement crée le squelette de l'organisation des populations et des ressources (biens) intégrées dans le territoire par le biais de systèmes politiques et économiques. Le territoire de cette population est composé de ressources agricoles (terrasses agricoles), de ressources en eau (réservoirs), de zones de peuplement (groupes résidentiels et groupes ménages isolés) et de zones vides dites « non résidentielles » qui structurent un paysage particulier.


Système
< forme le squelette
Système de peuplement
Légende
Classe
relation
* (plusieurs)
* (plusieurs)
Classe
Association de classes
Intégration
Population
Biens
> permet
spécialisation / généralisation
agrégation
Sous-classe
Classe
Système politique
Système économique
Territoire
Réseau de chemins
Terrasses agricoles
Réservoir
Zone de peuplement
Via
Jonction
Chaussée Sacbe
Terminus non résidentiel
Terminus résidentiel
connecté
Groupe résidentiel
Groupe famillial
Epicentre architectural
connecté ?
ignore si connecté ?
"Via" sont-elles entremêlées avec les terrasses agricoles ?
p. 273 Chase et al. 2001


Système
Système de peuplement
< forme le squelette
Organisation
Population
Biens
Système politique
Système économique
intégre >
Légende
Classe
relation
* multiplicité
* (multiplicité)
Classe
Classe Association
spécialisation/ généralisation
agrégation
Sous-classe
Classe
Réseau de chemins
Territoire
Terrasse agricole
Réservoir
Zone de peuplement
Via
Jonction
< connecte
Chaussée Sacbe
connecte >
Terminus non résidentiel
Terminus résidentiel
est connecté
Groupe résidentiel
Groupe famillial
Epicentre architectural
connecté ?
ignore si connecté ?
"Via" sont-elles entremêlées avec les terrasses agricoles ?
p. 273 Chase et al. 2001

Le texte qui décrit le diagramme

Ce système de peuplement crée le squelette de l'organisation des populations et des ressources (biens) intégrées dans le territoire par le biais de systèmes politiques et économiques. Le territoire de cette population est composé de ressources agricoles (terrasses agricoles), de ressources en eau (réservoirs), de zones de peuplement (groupes résidentiels et groupes ménages isolés) et de zones vides dites « non résidentielles » qui structurent un paysage particulier. Dans ce paysage, il y a un *épicentre* (groupe architectural et résidentiel) et plusieurs endroits appelés *termini* qui peuvent être résidentiels. Chaque composante de ce paysage, décrite ci-dessus, est reliée spatialement par un réseau de chemins composé de chaussées très formelles appelées « sacbes » et de chemins plus informels appelés « vias ». Les tracés de ces vias sont enchevêtrées dans des terrasses agricoles et reliées au réseau viaire formel par des jonctions avec les principales chaussées formelles, les sacbes. Ces chaussées organisées en réseau relient le groupe résidentiel de l'épicentre à d'autres *termini* résidentiels et, dans certains cas, à des *termini* non résidentiels (réservoirs, terrasses agricoles ou autres éléments) par des jonctions.

FIN

Evaluation et au-revoir