Analyse et Conception Orientées Objet avec UML.2

Unified Modeling Language

Bouchra BERRADA
email : berrada.bouchra@gmail.com

Sommaire

- Genèse d'UML
- Qu'est-ce UML ?
- Pourquoi UML ?
- Les Diagrammes UML 2
- Les point de vues UML 2

Genèse UML…

1998-2013 : - UML V1.2 en 98, 1.3 en 99, 1.4 en 2001, 1.5 en 2002, UML 2 en 2003, …UML 2.5 en 2013

1997 : - UML, consacrée par l'OMG et diffusée sous UML V1.1 , devient standard international

1995 : - Présentation de la description commune publique (UM) de la version 0.8 à la Conférence OOPSLA'95, puis 1.0 notation Universelle

1988 -1992 :



J. Rumbaugh 1991 OMT I. Jacobson 1992, OOSE **G. Booch 1994, OOAD**

1980 : Concepts Objets recherche (Smalltalk, C++, …)

Réflexion sur les langages de conception graphique OO … plus de 50 Méthodes OO

Qu'est-ce qu'UML ?

- Un langage graphique de modélisation objet
- Une famille de notation graphique s'appuyant sur un méta-modèle unique
- Un support méthodologique pour les concepteurs et développeurs pour formaliser l'analyse et la conception technique du logiciel orienté objet en particulier
- Un standard car s'appuyant sur une norme structurante.
- Une méthode ?

UML, Une Méthode ..?

METHODE = Langage de modélisation + ~~PROCESSUS~~

Par conséquent :

- UML est un langage de modélisation, pas une méthode
- UML est adaptable car peut être utilisé avec un processus de développement existant
- Deux processus : Unified Process (UP) et Rational Unified Process (RUP) sont associés à UML :
  - Guidé par les besoins des utilisateurs du système,
  - Itératif et incrémental
  - Centré sur l'architecture logicielle,
  - Orienté par la réduction des risques
- Les processus étant fondés sur quatre concepts et deux dimensions

Pourquoi UML ?

- Le plus connu et utilisé dans le monde
- Le marché est important et s'accroît
- UML est adaptable
- UML, étant un standard ouvert, contrôlé par l'OMG, prône l'interopérabilité, et spécifiquement entre systèmes orientés objet
- UML est dans le domaine public

UML est un langage standard de modélisation orientée objet

Présentation Générale

Diagrammes UML

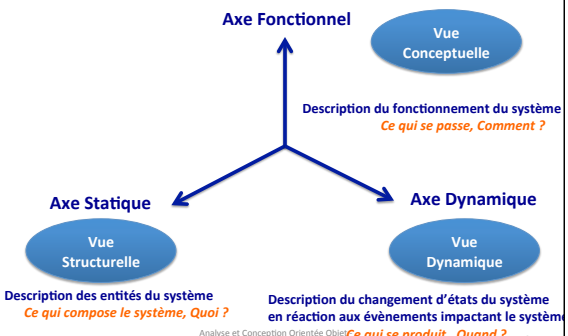
UML, dans sa version 2, propose treize diagrammes dépendants hiérarchiquement et qui se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie.

Ces diagrammes sont regroupés dans deux grands ensembles :

1. **STRUCTURELS** : *représentant l'aspect statique (Structure Diagram)*
1. **COMPORTEMENTAUX** : *représentant l'aspect dynamique (Behavior Diagram)*

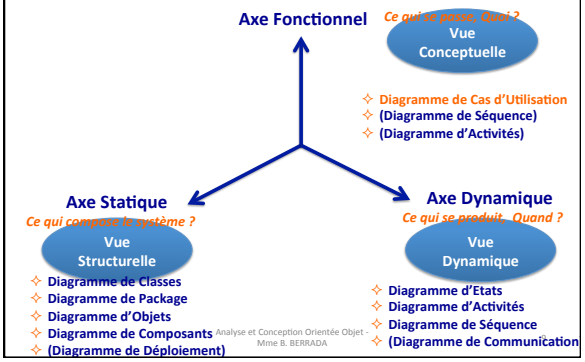
Structure du Cours

3 points de vues de modélisation :



Structure du Cours

3 points de vues de modélisation :



Présentation Générale

Description des Diagrammes UML 2

Diagramme	Finalité	Historique
Axe fonctionnel		
Cas d'utilisation (DCU) ou Use Case (UCD)	Interactions entre les utilisateurs et le Système, illustrant les différentes fonctionnalités (scénarios). Il constitue l'un des diagrammes des plus structurants dans l'analyse d'un système.	UML 1
Séquence (DSE)	Il décrit les Interactions de chaque scénario d'un CU en mettant l'accent sur la chronologie des opérations en interactions avec les objets.	UML 1, 2
Activité (DAC)	Il donne une vision des enchaînements des activités propres à une opération ou à un CU. Il permet aussi de représenter les flots de contrôle et les flots de données.	UML 1, 2

Présentation Générale

Suite - Description des Diagrammes UML 2

Diagramme	Finalité	Historique
Axe Statique (6 Diagrammes Structurels)		
Classes (DCL)	Description statique du système, intégrant dans chaque classe, ses caractéristiques et les relations statiques.	UML 1
Objets (DOB)	Représentation des Instances de Classes et de leurs liens.	UML 1
Packages (DPA)	Vue d'ensemble du système, constitué d'éléments homogènes du système (Classes, composants)	UML 2
Composant (DCP)	Représentation des constituants du logiciel au niveau implémentation d'un système.	UML 1, 2
Structure composite (DSP)	Description des collaborations d'instances constituants les fonctions du système à réaliser	UML 2
Déploiement (DPL)	Description de l'architecture physique avec vue centrée sur la répartition des composants - nœuds de traitements (device), ou nœud d'environnement d'exécution)	UML 2

Présentation Générale

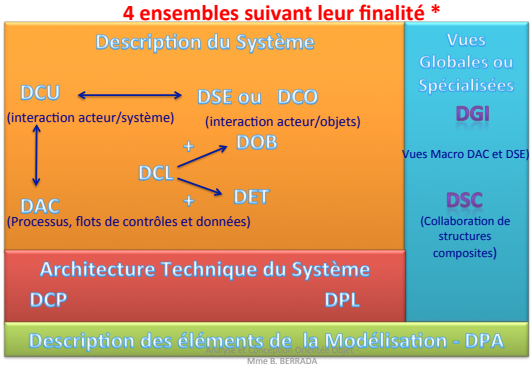
Suite - Description des Diagrammes UML 2

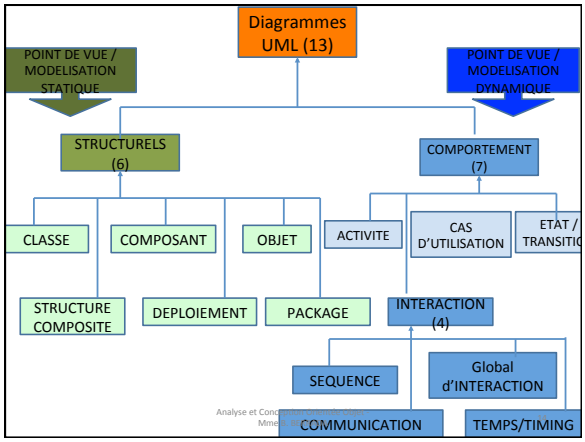
Diagramme	Finalité	Historique
Axe Dynamique (4 Diagrammes Comportementaux)		
Etat-Transition (DET) (Machine d'état)	Description des différents états des objets en réaction aux évènements.	UML 1
Communication (DCO)	Représentation des scénarios de communications mettant l'accent sur les objets et les messages échangés.	UML 1
Global d'Interaction (DGI)	Vue générales des interactions, décrites dans les DSE et flots de contrôles du DAC	UML 2
Temps ou Timing (DTP)	Représentation des états et les interactions d'objets dans un contexte où le temps a une forte influence sur le système à gérer.	UML 2

Présentation Générale

Suite - Description des Diagrammes UML 2

4 ensembles suivant leur finalité *





Structure du Cours

Construction privilégiant :

- **Trois points de vues classiques, en insistant sur le ou les diagrammes prépondérant.s**
  - **1. AXE FONCTIONNEL**
    - **DIAGRAMME DES CAS D'UTILISATION (DCU)**
    - DIAGRAMME DE SEQUENCE (DES)
    - DIAGRAMME D'INTERACTION (DAC)
  - **2. AXE STATIQUE**
  - **3. AXE DYNAMIQUE**
- **Une présentation détaillée d'UML 2, illustrant les concepts essentiels et les diagrammes les plus structurants par des exemples et études de cas.**

Modélisation Fonctionnelle

- Elle comprend 3 diagrammes dans la modélisation dynamique :
  - **DIAGRAMME DES CAS D'UTILISATION (DCU)**
  - DIAGRAMME DE SEQUENCE (DSE)
  - DIAGRAMME D'INTERACTION (DGI)

DIAGRAMME DE CAS D'UTILISATION ou USE CASES : *Notions de base*

- **Présentation et notions de base** :
  - Les diagrammes de cas d'utilisation ont été définis initialement par Ivar JACOBSON en 1992 dans sa méthode OOSE (Addison-Wesley, 92)
  - Ils constituent un moyen de recueillir et de décrire les besoins des acteurs du système
  - Ils décrivent l'interaction entre les acteurs (utilisateurs du cas) et le système, ***selon le point de vue utilisateur***
  - Ils mettent en jeu 3 concepts importants :
    - ✓ **Acteur**,
    - ✓ **Cas d'utilisation** ,
    - ✓ **Interaction entre acteur et cas d'utilisation**

Exercice 1 - Etape 1.2
Diagramme de Contexte

- **Ses composants sont :**
  - ✓ Les acteurs externes.
  - ✓ Un processus unique (Boîte noire*) symbolisant le Système Information étudié(SI) ;
  - ✓ L'Echange entre le système étudié et son environnement. Autour du système, nous devons donc relier les acteurs principaux et secondaires correspondants.

Remarque : **On représente les interactions des acteurs avec le système étudié. Mais pas les interactions entre acteurs.**

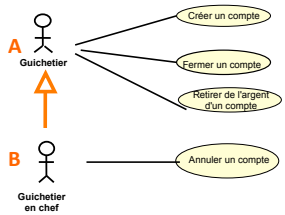
* : représentation d'un système sans considérer son fonctionnement interne

Etude de Cas

Relations entre acteurs

Relation acteur <-> acteur

- **Une seule relation** la généralisation/spécialisation



Les acteurs spécialisés héritent alors des associations de l'acteur ancêtre

CAS D'UTILISATION : *Identification*

Cas d'Utilisation : Comment les identifier ?

L'objectif est le suivant :

- L'ensemble des cas d'utilisation doivent décrire exhaustivement les exigences ***fonctionnelles*** du système.
- Chaque cas d'utilisation correspond donc à une fonction métier du système, selon un point de vue d'un de ses acteurs.
- **Pour chaque acteur, il convient de :**
  - ✓ Rechercher les différentes fonctions métier utiles à l'utilisation du système (tâches utilisateur)
  - ✓ Déterminer dans le cahier des charges, les services fonctionnels attendus du système

CAS D'UTILISATION : *Définition et Notation*

Cas d'Utilisation : Définition

- Un Cas d'Utilisation (use case) correspond à un ***nombre d'actions*** ou fonctionnalités *(scenario)* que le système devra exécuter en réponse à un ***besoin d'un acteur***.
- Un cas d'utilisation doit produire un résultat observable et intéressant pour un ou plusieurs acteurs particuliers.

Le cas d'utilisation est représenté par un ovale dans lequel figure son nom.

- Un cas d'utilisation doit être relié à au moins un acteur.
- Un cas d'utilisation est Nommé par un verbe à l'infinitif, suivi d'un complément, du point de vue de l'acteur, par exemple : Envoi d'un message

CAS D'UTILISATION : *Notation*

Cas d'Utilisation : Notation

Nom du CU

Exemples

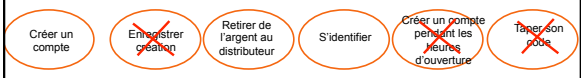


DIAGRAMME DE CAS D'UTILISATION : *Intérêt (1)*

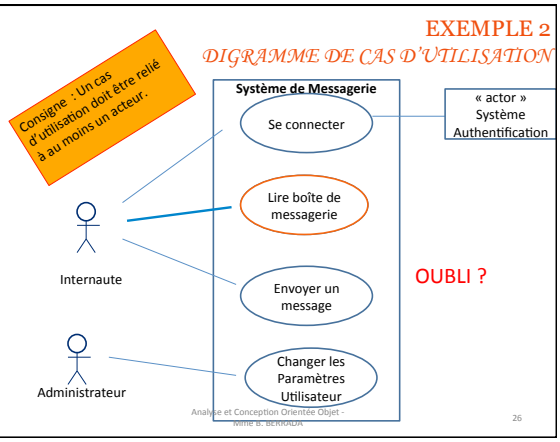
- **Intérêt** :
  - Déterminer les besoins fonctionnels du système
  - Permettre aux utilisateurs d'exprimer leurs attentes et besoins
  - Permettre d'impliquer les utilisateurs dès les premiers stades de développement
  - Constituer une base pour les tests fonctionnels

CAS D'UTILISATION : *Intérêt (2)*

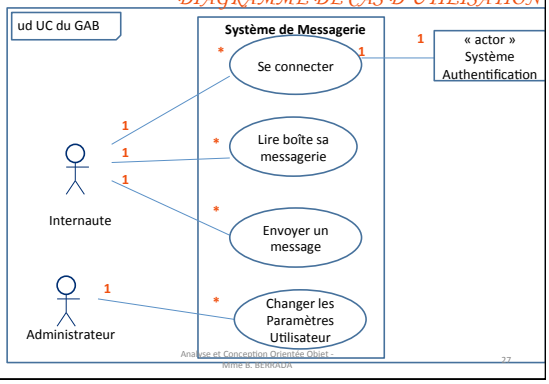
- Les cas d'utilisation (CU) représentent le dialogue ***entre l'acteur et le système*** de manière abstraite
- Un CU correspond à une fonction du système ***visible par l'acteur***
- Un CU est une ***séquence d'actions*** réalisées par le système produisant un résultat observable
- Ensemble de scénarios au sein d'une description unique

Regroupe un ensemble de scénarios correspondant à un même but

EXEMPLE 2
DIGRAMME DE CAS D'UTILISATION



EXEMPLE 2
DIAGRAMME DE CAS D'UTILISATION



CAS D'UTILISATION
Relations entre éléments de DCU

- **3 types de relations** :
  - Communication
  - Dépendances stéréotypées (inclusion et extension)
  - Généralisation/Spécialisation

CAS D'UTILISATION

Relations entre éléments de DCU

Relation acteur <-> cas d'utilisation : communication

- Point de vue **besoin** : représente la possibilité d'atteindre un but
- Point de vue **système** : représente un échange de messages

CAS D'UTILISATION :

Relation d'inclusion

Relations cas d'utilisation <-> cas d'utilisation

- **inclusion** (« *include* »)
  - Le cas d'utilisation incorpore ***explicitement*** et de manière ***obligatoire*** un autre cas d'utilisation à l'endroit spécifié

Signifiant qu'une instance de A contient le comportement décrit par B ou encore B est une partie *obligatoire* de A

- Cette relation permet ainsi de décomposer des comportements et de définir des comportements partageables entre plusieurs cas d'utilisation.

CAS D'UTILISATION :

Relation d'extension

Relations cas d'utilisation <-> cas d'utilisation

- **extension** (« *extend* »)
  - Le cas d'utilisation incorpore ***implicitement*** et de manière ***optionnelle*** un autre cas d'utilisation à l'endroit spécifié

Signifiant qu'une instance de A peut être étendue par le comportement décrit par B

- Si une condition d'extension existe, il est possible de la préciser à proximité du mot clé.

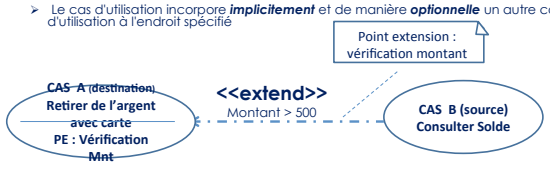
Consultation de solde n'intervient que si le retrait dépasse 500 dirhams

CAS D'UTILISATION :

Relation d'extension

Relations cas d'utilisation <-> cas d'utilisation

- **extension** (« *extend* »)
  - Le cas d'utilisation incorpore ***implicitement*** et de manière ***optionnelle*** un autre cas d'utilisation à l'endroit spécifié



Signifiant qu'une instance de A peut être étendue par le comportement décrit par B

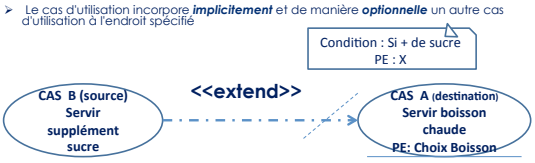
- Si une condition d'extension existe, il est possible de la préciser à proximité du mot clé.

Consultation de solde n'intervient que si le retrait dépasse 500 dirhams

CAS D'UTILISATION :
Relation d'extension

Relations cas d'utilisation <-> cas d'utilisation

- **Relations d'extension** (« extend »)
  - Le cas d'utilisation incorpore ***implicitement*** et de manière ***optionnelle*** un autre cas d'utilisation à l'endroit spécifié



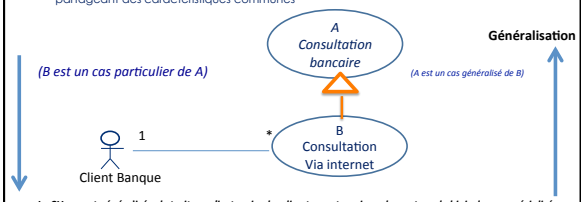
Signifiant qu'une instance de A peut être étendue par le comportement décrit par B

- Si une condition d'extension existe, il est possible de la préciser à proximité du mot clé.

CAS D'UTILISATION :
Relation généralisation /spécialisation

Relations cas d'utilisation <-> cas d'utilisation

- **Relations de généralisation / spécialisation**
  - la généralisation est la relation entre un cas d'utilisation et un ou pls autres cas d'utilisation partageant des caractéristiques communes



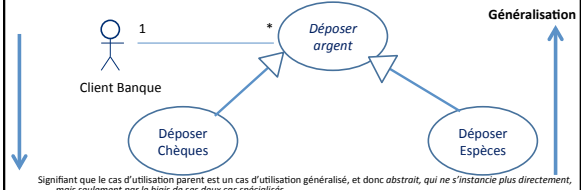
Le CU parent généralisé, *abstrait, ne s'instancie plus directement, mais seulement par le biais du cas spécialisé.*

Spécialisation

CAS D'UTILISATION :
Relation généralisation /spécialisation

Relations cas d'utilisation <-> cas d'utilisation

- **Relations de généralisation / spécialisation**
  - Conformément aux principes de spécialisation-généralisation présentée pour les classes, la généralisation est la relation entre un cas d'utilisation et de deux autres cas d'utilisation partageant des caractéristiques communes



Signifiant que le cas d'utilisation parent est un cas d'utilisation généralisé, et donc *abstrait, qui ne s'instancie plus directement, mais seulement par le biais de ses deux cas spécialisés.*

Spécialisation

CAS D'UTILISATION :
Description textuelle des cas d'utilisation

- **La fiche de description textuelle d'un cas d'utilisation**
  - La structuration suivante est souvent préconisée :

Sommaire d'identification (obligatoire)	Titre, Résumé, Dates de création et de modification, Version, Responsables, Acteurs, …
Description des scénarios	le scenario nominal, les scenarios (enchaînements) alternatifs, les scénarios (enchaînements) d'erreur, mais aussi les pré-conditions et les post-conditions.
Exigences non-fonctionnelles (optionnel)	Fréquence, volumétrie, disponibilité, fiabilité, intégrité, confidentialité, performances, concurrences, etc. Précise aussi les contraintes IHM, comme les règles d'ergonomie, une charte graphique, etc.

CAS D'UTILISATION :
Description textuelle des cas d'utilisation

2. Description des scénario

- **SCENARIO ? Il décrit le déroulement d'un CU**

Le scénario est au cas d'utilisation ce qu'est l'objet à la classe

- ✓ Un cas d'utilisation propose un comportement nominal (scénario nominal ou idéal)
- ✓ Un cas d'utilisation propose aussi un ou pls cas alternatifs représentant un cheminement particulier dans le cas d'utilisation (que se passe t-il si ..?)
- ✓ Un cas d'utilisation propose aussi des situations exceptionnelles
- ✓ L'idéal est de créer suffisamment de scénarios pour couvrir l'essentiel du cas d'utilisation

CAS D'UTILISATION :
Description textuelle des cas d'utilisation

2. Description des scénario

- **Pré-Condition :** Si certaines conditions particulières sont requises avant l'exécution du cas
- **Scénario Nominal :** Il s'agit du scénario principal qui doit se dérouler sans incident et qui permet d'aboutir au résultat souhaité
- **Scénarios Alternatifs :** Les autres scénarios, secondaires ou correspondant à la résolution d'anomalies. Le lien avec le scénario principal se fait par une numérotation hiérarchisée (1.1a, 1.1b,...) pour rappeler le numéro de l'action concernée

- **Scénario d'erreur :** Il termine brutalement le cas 'utilisation en échec
- **Post-Condition : P**ar symétrie, si certaines conditions doivent être réunies après l'exécution du cas.

CAS D'UTILISATION :
Description textuelle des cas d'utilisation

La description textuelle des cas d'utilisation est indispensable car elle seule permet de :

- Communiquer facilement avec les utilisateurs
- S'entendre sur la terminologie métier employée.

En revanche, le texte présente des désavantages car difficile de montrer :

- La succession des enchaînements
- Le moment où les acteurs secondaires sont sollicités.

De même que la maintenance des évolutions est aussi fastidieuse.

Quelques rappels
Description textuelle des cas d'utilisation **:**

Une Description textuelle fait référence au déroulement d'un ou pls scénario.s lié.s à un CU.

Généralement :

- Un cas d'utilisation propose un comportement idéal qui se traduit par un **scénario nominal**
- Un cas d'utilisation propose aussi un ou pls **scénario.s alternatifs** représentant un cheminement particulier dans le cas d'utilisation (que se passe t-il si ..?)
- Un cas d'utilisation propose aussi des **scénario.s d'erreur** traduisant des situations exceptionnelles

L'idéal est de créer suffisamment de scénario.s pour couvrir l'essentiel du cas d'utilisation

Quelques rappels
Relations entre CU

- Les inclusions et les extensions sont représentées par des dépendances.
- Lorsqu'un cas B inclut un cas A, B **dépend** de A.
- Lorsqu'un cas B étend un cas A, B **dépend** aussi de A.
- On note toujours la dépendance par une **flèche pointillée** **B --- > A** qui se lit « B dépend de A ».
- Lorsqu'un élément A dépend d'un élément B, toute modification de B sera susceptible d'avoir un impact sur A.
- Les « include » et les « extend » sont des **stéréotypes** (entre guillemets) des relations de dépendance.

DIAGRAMME DE CAS D'UTILISATION

Exercices d'Entraînement (EE)

Description dynamique des cas d'utilisation

Pour les cas d'utilisation, on peut utiliser une description dynamique :

1. **Diagramme de séquence**
2. **Diagramme d'activité (43 – 81)**

Diagramme dynamique
Diagramme d'Activité (DAC)

Activité :

- Une exécution d'un mécanisme ;
- Un déroulement d'étapes séquentielles.

Diagramme d'Activité :

Une technique intéressante pour :

- Représenter la logique comportementale (de méthode)
- Décrire le processus métier
- Représenter les enchaînements d'activités (workflow ou ordre d'exécution, ou encore règles de séquencement)
- La vision des diagrammes d'activités se rapproche des langages de programmation impérative (C, C++, Java)

Le **Diagramme d'activité** décrit le ***comportement*** interne des opérations ou des CU, en se concentrant sur les opérations plutôt que sur les objets.

Diagramme d'activité (DAC)
Concepts de base

Les concepts du diagramme d'activité (DAC)

Au cœur du DAC, 2 concepts essentiels :

1. **ACTION** : Elle correspond à :
   - un **traitement** qui modifie l'état du système
   - une opération instantanée (conceptuellement) et atomique (ne peut être interrompue)

Formalisme Nom de l'action

N.B. : Un rectangle avec des coins arrondis
(comme pour les états-transitions)

Exemple Saisir commande

Diagramme d'activité
Action

1. Action : Transition et flot de contrôle

- Dès qu'une action est achevée, une **transition** automatique peut être déclenchée vers l'action suivante.
  Il n'y a donc pas d'événement* associé à la transition

- **La transition s'effectue lorsque l'opération est terminée**
- **La transition ne nécessite pas de déclenchement par événement asynchrone**
- **Les transitions spécifient l'enchaînement des traitements, sans durée perceptible**
- L'enchaînement des actions constitue le **flot de contrôle**

Diagramme d'activité
Activité

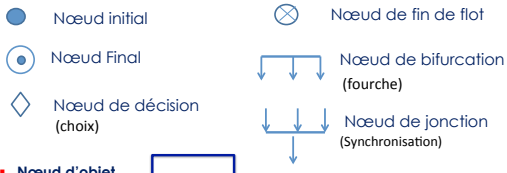
2. Activité : Elle représente le comportement d'une partie du système **en terme d'action et de transitions**

Mis à part les transitions, combien de types de symboles voit-on ?

Diagramme d'activité
Activité : Type de Nœuds

Une activité est composée de 3 types de nœuds :

- **Action**, symbolisée dans un nœud d'exécution Action 1
- **Nœud de contrôle** , symbolisé dans un nœud initial, final, bifurcation, jonction, fusion, décision, flux de sortie, pin d'entrée et de sortie)



- **Nœud d'objet.** objet

Diagramme d'activité
Activité : Type de Nœuds

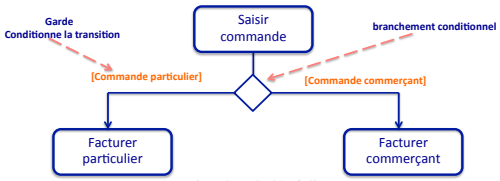
Nœud initial : marque le début d'une activité

Nœud Final : marque la fin d'une activité

Diagramme d'activité
Activité : Type de Nœuds

Nœud de décision

- Il n'a qu'un seul flot en entrée
- Il permet de faire un choix entre plusieurs flots sortants en fonction des conditions de chaque flot.
- On peut aussi utiliser que 2 flots de sortie :
  - ✓ Le premier correspondant à la condition vérifiée
  - ✓ Le second traitant le cas contraire (condition non vérifiée)



Description dynamique
Diagramme d'activité

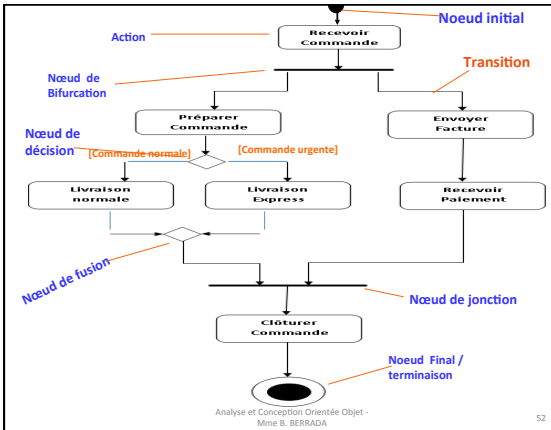
Exemple de Diagramme d'activité :

Prenons en exemple le processus métier de traitement d'une commande, depuis sa réception jusqu'à la clôture de cette commande, et impliquant les actions suivantes :

- Recevoir commande ;
- Préparer commande ;
- Envoyer Facture ;
- Livrer Express ou Normalement selon choix
- Recevoir Paiement ;
- Clôturer Commande.

Précisons que :

- Quelques actions peuvent se dérouler en parallèle : Préparer commande et Envoyer Facture ;
- Les actions suivantes peuvent aussi se dérouler en parallèle



Description dynamique
Diagramme d'activité

Interprétation du Diagramme d'activité :

- Préparer commande et Envoyer Facture sont parallèles ainsi que les actions qui suivent.

Il est donc possible de lire les cheminements :

1. Préparer commande
2. Envoyer Facture,
3. Livrer commande
4. Recevoir Paiement

Ou

1. Envoyer Facture
2. Recevoir Paiement
3. Préparer commande
4. Livrer commande

Enfin

5. Clôturer Commande.

Description dynamique
Diagramme d'activité

Caractéristiques du Diagramme d'activité :

- **Parallélisme :**
  - ✧ **Le DAC permet au responsable du processus de choisir l'ordre d'exécution.**

- ***Important*** *pour la modélisation métier, puisque les processus se déroulent toujours en parallèle.*
- ***Utiles*** *dans les cas d'algorithmes concurrents (threads)*

Description dynamique
Diagramme d'activité

Caractéristiques du Diagramme d'activité :

- **Bifurcation / Débranchement**
  - ✧ **Un flot unique entrant peut créer plusieurs flots sortants concurrents.** *(en sortie de la barre de synchronisation)*

- **Synchronisation / Jonction**
  - ✧ **Le flot sortant ne peut être exécuté que lorsque toutes les actions entrantes sont terminées.**
- *Elle marque la fin d'un comportement conditionnel. Initialisé par une décision*
- *L'action ne s'exécute que si tous les flots aboutissent (F1et F2 et Fn).*

Cas de la clôture de la commande

Le nœud de Jonction est le symétrique du nœud de bifurcation

Description dynamique
Diagramme d'activité

Caractéristiques du Diagramme d'activité :

- **Branchement / Décision – test**
  - ✧ **A un seul flot entrant, plusieurs flots sortants *gardés.***

Une garde est une condition booléenne placée entre crochets.

Le Branchement est donc associé à un comportement conditionnel.

- *Chaque fois qu'une décision est atteinte, il n'est possible de s'orienter que vers l'un des flots sortants gardés (l'un **ou** l'autre)*
- *Les conditions **doivent** donc être mutuellement exclusives. (Employer else)*

- **Fusion – Test**
  - ✧ **Plusieurs flots entrants possibles et un seul flot sortant.**

Le flot sortant est exécuté dès qu'un des flots est activé.

Description dynamique
Diagramme d'activité

Autre exemple :

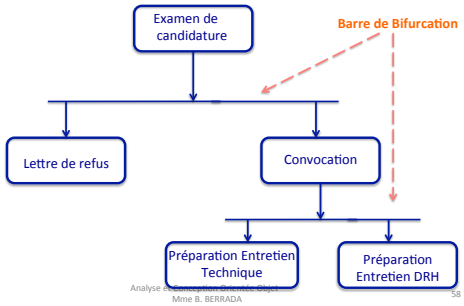
Représenter le traitement lors du dépôt de candidature pour embauche.

La candidature est examinée. Il donne lieu soit à une lettre de refus, soit à une convocation.

La convocation exige une préparation à deux entretiens : technique et celui avec le DRH.

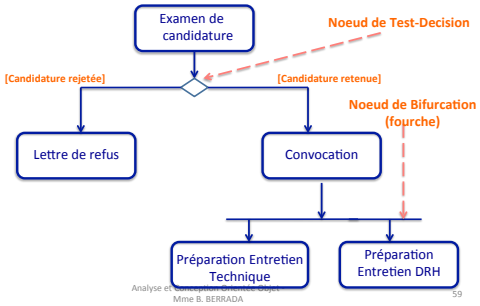
Description dynamique
Diagramme d'activité

Solution 1 pour le traitement lors du dépôt de candidature pour embauche.



Description dynamique
Diagramme d'activité

Solution 2 pour le traitement lors du dépôt de candidature pour embauche.



Description dynamique
Diagramme d'activité

Exercice 3 :

Réalisation du diagramme d'activité qui décrit la dynamique du cas d'utilisation RETIRER ARGENT en utilisant les principaux symboles graphiques.

Description dynamique
Diagramme d'activité

1. **Un DAC peut-être utilisé pour décrire un workflow**

Oui. Un DAC est un bon candidat pour décrire les enchaînements constituant un workflow.

2. **Un DAC peut servir à découvrir des activités parallèles**

Oui. Il est possible que les actions soient exécutées simultanément. Donc, lors de l'élaboration d'un DAC, la modélisation peut découvrir des actions // .

3. **Une activité représente une tâche automatisée ou manuelle**

Oui. Une activité est constituée d'actions élémentaires. Une action consiste à affecter une valeur à un attribut, créer ou détruire un objet, envoyer un signal à un autre objet ou à soi-même. Dans un workflow, il peut s'agir s'une tâche manuelle.

4. **Une activité peut être implantée par une méthode d'un objet**

Oui. Une méthode peut tout à fait décrire le comportement d'une activité

5. **Il n'est pas permis de faire suivre une activité d'une autre activité**

Non. Une activité peut suivre une autre, par enchaînement. Dès que l'activité est terminée, l'enchaînement est franchi et la seconde activité est enclenchée.

Description dynamique
Diagramme d'activité

7. **Un DAC peut décrire des activités // en fixant des priorités**

Non. Un DAC peut décrire des activités enclenchées en // mais sans fixer ni ordre ni priorités.

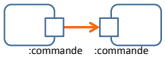
8. **A quoi sert un enchaînement de type fourche ainsi qu'un enchaînement de type synchronisation ?**

FOURCHE : il possède plusieurs activités de destination. Après son franchissement, toutes les activités sont enclenchées en parallèle.

SYNCHRONISATION : Plusieurs activités d'origine, après exécution, doivent être terminées pour que l'enchaînement soit franchi. Les activités parallèles sont synchronisées.

Description dynamique
Diagramme d'activité

1. **Diagramme d'activité et autres concepts particuliers :**

- Pin d'entrée et de sortie : 
- Flots de données et Nœuds d'objet : 

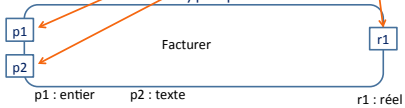
- Décomposition des actions
- Partitions
- Signaux

Description dynamique
Diagramme d'activité

- Pin d'entrée et de sortie :  (Paramètre)

Un pin d'entrée représente un paramètre que l'on peut spécifier en entrée ou en sortie d'une action

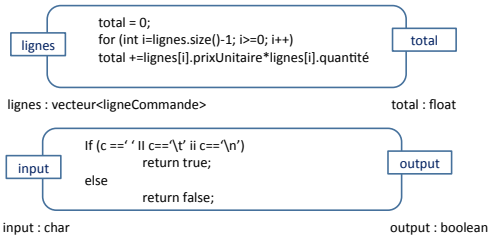
Un nom de donnée et un type peuvent être associés au PIN



Un pin représente un point de connexion pour une action.
L'action ne peut débuter que si l'on affecte une valeur à chacun de ses pins d'entrée.
Les valeurs sont passées par copie.
Quand l'activité se termine, une valeur doit être affectée à chacun des pins de sortie

Description dynamique
Diagramme d'activité

- Pin d'entrée et de sortie : (Paramètre)



Description dynamique
Diagramme d'activité

- **Flot de données : *Définition***

UML 2 emploie le terme Flot *(Flow)* ou Arc pour décrire les connexions entre deux actions.

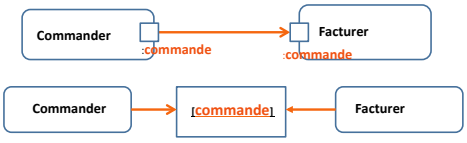
La forme d'arc la plus élémentaire est ***une simple flèche*** entre deux actions.

Description dynamique
Diagramme d'activité

- **Flots de données et Nœud d'objet**

Un nœud d'objet permet de représenter le Flot de données véhiculé entre les actions.

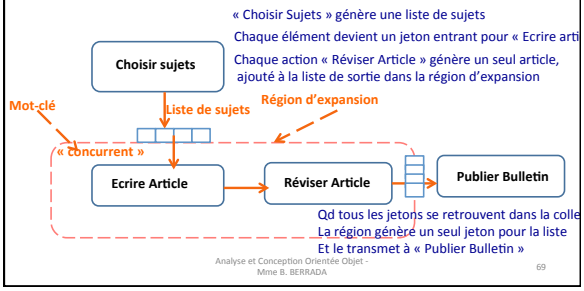
Les objets peuvent se représenter soit en utilisant le pin d'objet soit en représentant explicitement un objet



Description dynamique
Diagramme d'activité

- **Région d'expansion**

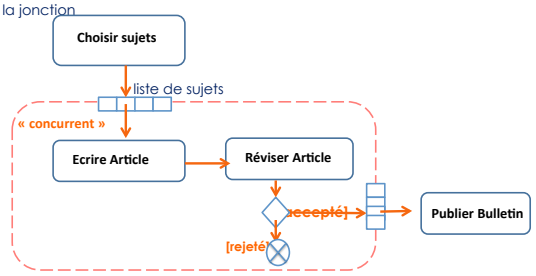
Elle marque une zone dans laquelle des actions se déroulent une fois pour chaque élément d'une collection. Exemple d'un journal :



Description dynamique
Diagramme d'activité

- **Région d'expansion avec spécification de jonction**

Il est peut être utile d'émettre des règles : expression booléenne associée à la jonction

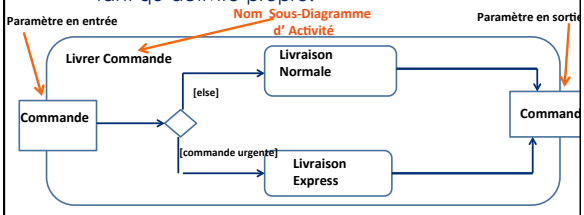


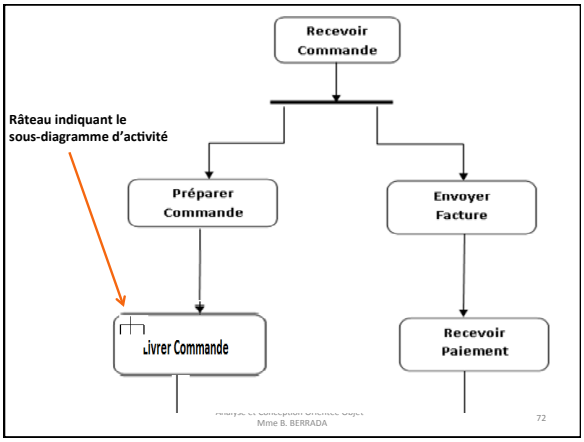
Description dynamique
Diagramme d'activité

- **Décomposition d'actions**

Les actions peuvent être décomposées en sous-activités.

La logique de la livraison peut peut-être définie en tant qu'activité-propre:

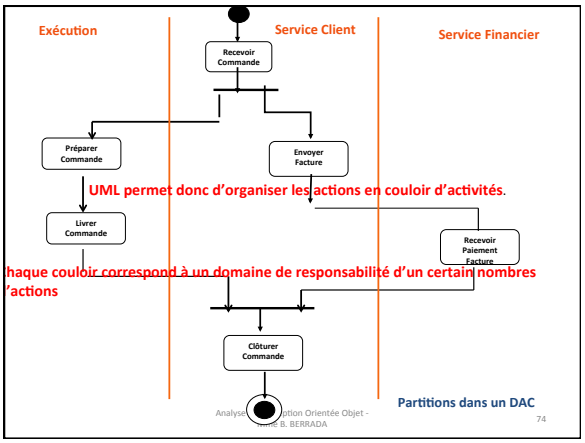




Description dynamique
Diagramme d'activité

Partitions : Qui fait quoi ?

- Diviser un diagramme d'activité en partitions montrant quelles actions sont exécutées par une classe ou une unité organisationnelle.



Description dynamique
Diagramme d'activité

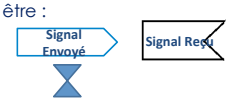
Signaux temporels : Quand ?

Les diagrammes d'activités, comme les diagrammes de temps, peuvent représenter les actions de communications liées à certains évènements.

Ils représentent les évènements auxquelles les actions peuvent être liées.

Les types d'évènements peuvent être :

- Signal
- Écoulement de temps



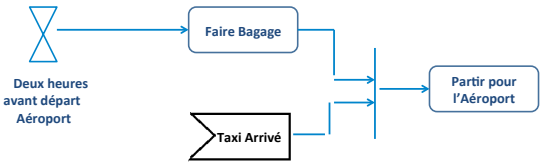
Description dynamique
Diagramme d'activité

Exemple :

1. Nous devons avoir fait nos bagages deux heures avant de partir pour l'aéroport.
2. Nous ne pouvons pas partir avant que le taxi ne soit arrivé.
3. Si le taxi arrive avant que les bagages soient faits, il doit attendre avant de partir.

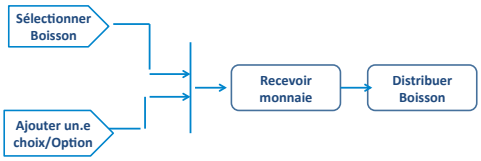
Description dynamique
Diagramme d'activité

Résolution exemple :



Description dynamique
Diagramme d'activité

Exemple 2 : Distributeur de boisson



Description dynamique
Diagramme d'activité

Quand utiliser des diagrammes d'activités ?

La grande force des diagrammes réside dans le fait qu'ils permettent la représentation du :

- ✓ Parallélisme : Workflow
- ✓ D'Organigramme

Technique similaire : Réseau de pétri

- ✓ Programmes concurrents

4. Test d'Entraînement (TE)
sur les Diagramme d'Activité

- ✓ Réaliser un diagramme d'Activité qui décrit le cas d'utilisation RETIRER ARGENT pour un porteur de carte non client.

- ✓ Utiliser les conventions graphiques :
  - Nœud initial, Terminal
  - Nœud de bifurcation, fusion, jonction
  - Nœud de décision ...

Description dynamique
des cas d'utilisation

Pour décrire le fonctionnement des cas d'utilisation, on peut utiliser une description dynamique :

1. **Diagramme de séquence**
2. **Diagramme d'activité**

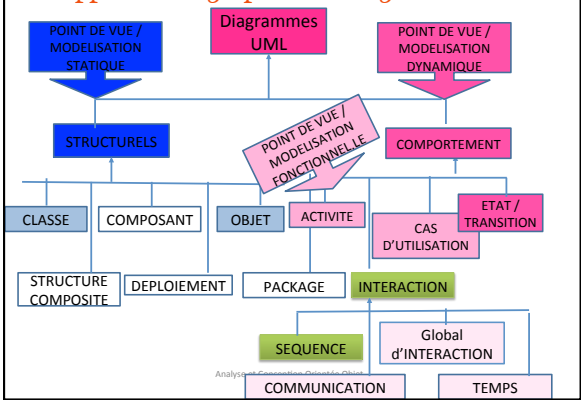
Diagrammes d'Interactions

La norme 2.0 reclasse les diagrammes de la norme 1.X et met les 4 diagrammes suivants dans la catégorie des diagrammes d'interaction :

- **Diagrammes de séquence**
- **Diagramme de communication**
- Diagramme d'interaction globale
- Diagrammes de temps

Ces types de diagramme tendent à mettre l'accent spécifiquement sur la séquence des opérations plutôt que sur les données qui sont véhiculées,

Rappel - Cartographie des Diagrammes UML 2



Description dynamique
Diagramme de Séquence (DSE)

Le DSE est une forme de diagramme comportemental, dérivé des scénarios de OMT

L'objectif du DSE est de représenter les interactions entre objets en indiquant la ***chronologie des échanges***.

Grâce à ces informations, vous pouvez déterminer plus précisément pourquoi deux objets sont liés et voir comment ils s'utilisent mutuellement.

Cette représentation est réalisée par ***cas d'utilisation*** en considérant :

- Forme générique : décrivant ***tous*** les déroulements possibles d'un scénario et contenant des ***branchements***, des **conditions**, et des ***boucles***;
- Forme d'instanciation : décrivant ***un aspect spécifique*** d'un scénario et ne contenant aucun ***branchements***, et aucune ***condition*** ou ***boucle***.

En général, un DSE capture le comportement d'un seul scénario.

Description dynamique
Diagramme de Séquence (DSE)

Autrement dit :

- Les diagrammes de séquences permettent de décrire **COMMENT** les éléments du système interagissent entre eux et avec les acteurs.
- Les DSE illustrent :
  - Les objets, placés au coeur d'un système interagissant en s'échangeant des messages chronologiquement pour la réalisation du processus de l'application (enchaînement ou chronologie des traitements) ;
  - Les contraintes de temps; ce dernier étant représenté comme une dimension explicite (apsects de temps réel);
  - Les acteurs interagissant avec le système au moyen d'IHM (Interfaces Homme-Machine).

Diagramme de Séquence (DSE)

Stimuli et Message

Quelques concepts de base :

Stimulus

- Un **stimulus** est une communication entre deux instances dans le but de déclencher une action.

Message

- Un **message** est une spécification du Stimulus, qui définit les rôles de l'émetteur et du récepteur.
  Un message est émis par un nom_objet_1 vers un nom_objet_2

Diagramme de Séquence (DSE)

Formalisme

Formalisme général du cadre d'un diagramme de séquence :

sd : abréviation de « sequence diagram »

Diagramme de Séquence (DSE)

Formalisme

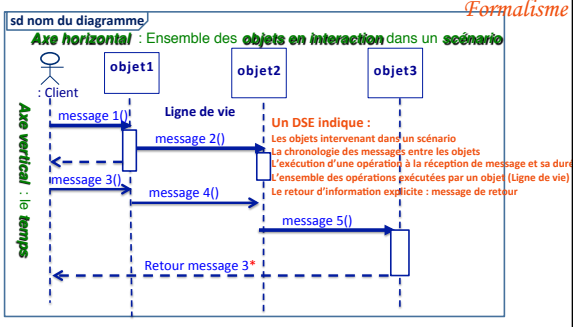


Diagramme de Séquence (DSE)

Concepts de base

Pour illustrer les concepts de base et les opérations d'un DSE, prenons l'exemple d'un scenario de traitement d'une commande, spécifiant :

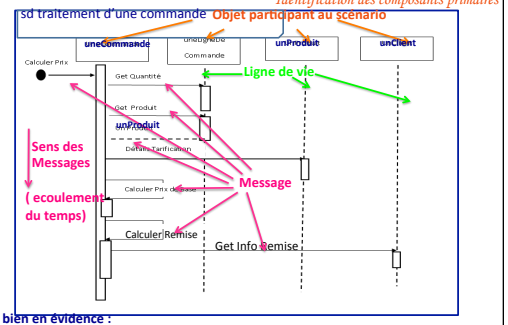
Pour calculer le prix d'une commande, l'objet commande doit :

- ✓ Lire toutes les lignes de la commande
- ✓ Déterminer leur prix, en fonction des tarifications des produits.

Après avoir traité chaque ligne, il doit ensuite :

- ✓ Calculer une remise globale, qui dépend des règles associées à chaque client.

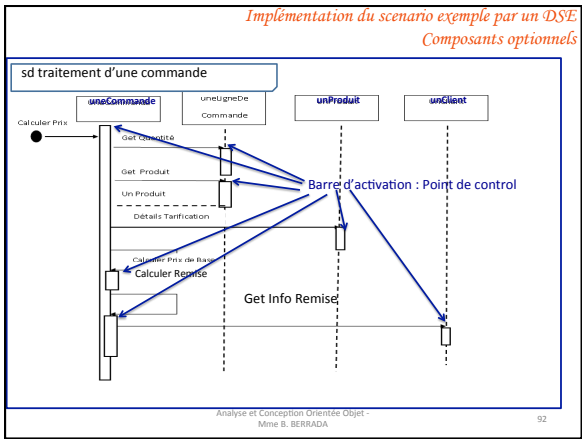
Implémentation du scenario exemple par un DSE
Identification des composants primaires



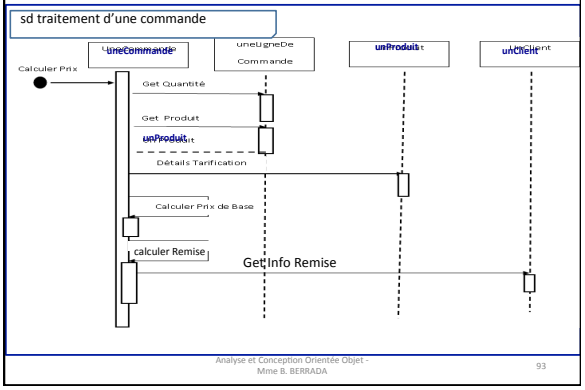
Le DES met bien en évidence :

- **L'interaction d'un objet participant au scénario**
- **Chaque participant ayant une ligne de de vie, qui parcourt verticalement la pag**
- **L'ordre des messages se faisant de haut en bas.**

Implémentation du scenario exemple par un DSE
Composants optionnels



Implémentation du scenario exemple par un DSE
Composants optionnels



Implémentation du scenario exemple par un DSE
Composants optionnels

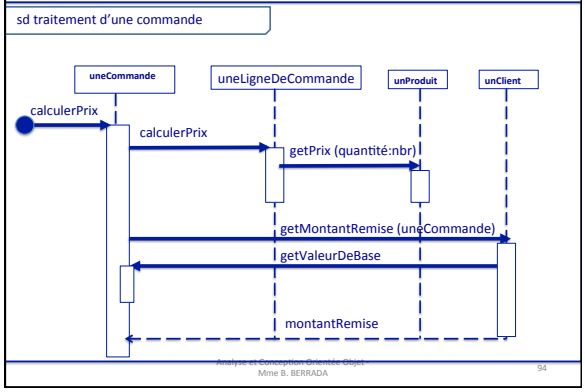


Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

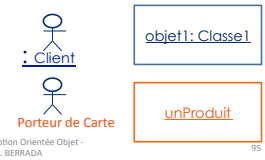
Définition des composants primaires d'un DSE

➢ ***1. Les objets (Object)*** : participants à l'interaction

Les objets apparaissent toujours dans la partie supérieure, ce qui facilite l'identification des classes (en supposant que la notation des classes est utilisée).

Représentation graphique :

UML



UML 2

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

Les composants primaires (suite) :

➢ ***2. La ligne de vie de l'objet (Object lifeline) :*** représentant

✓ la vie d'un objet dans le contexte de la séquence **d'événements**

Formalisme :

Un événement *se produit généralement lors de la réception explicite d'un signal ou d'un message, lorsqu'une condition devient vraie, écoulement d'une période de temps (expression temporelle)*

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

Les composants primaires (suite) :

➢ ***2. La ligne de vie de l'objet (Object lifeline) :***

✓ Chaque ligne de vie comporte une activation : point de contrôle (durée de vie de l'activité du participant dans l'interaction)

Formalisme :

- Le point de contrôle correspond au temps pendant lequel l'une des méthodes du participant est au sommet de la pile ;
- Aussi appelées ***barres d'activation, elles sont facultatives, mais elles clarifient le comportement.***

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

Les composants primaires (suite) :

➢ ***3. Les messages*** définissent une communication particulière entre deux instances, présentes sur des lignes de vie.

- Représentés par des flèches directionnelles véhiculant une information (message) envoyée entre les objets ;
- Les messages anticipent qu'une action sera entreprise ;
- Dans la plupart des cas, la réception d'un message est suivie de l'exécution d'une méthode d'une classe (classe réceptrice) ;
- L'ordre relatif des messages est matérialisé par l'axe vertical qui représente l'écoulement du temps ;
- Le retour de message doit être matérialisé, lorsqu'il existe.

Plusieurs types de messages existent, les plus communs sont :

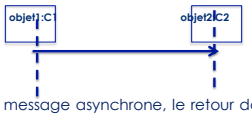
- L'envoi d'un signal ou d'un événement ;
- L'invocation d'une opération ;
- La création ou la destruction d'une instance.

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ ***3.1 Message asynchrone*** :

- ✓ L'émetteur n'attend pas la réponse à son message, il poursuit l'exécution de ses opérations.
- ✓ L'émetteur ne sait pas si le message arrivera à destination, le cas échéant quand il arrivera et s'il sera traité par le destinataire.
- ✓ Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.



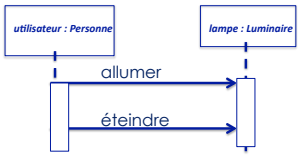
- Lors de l'envoi de message asynchrone, le retour doit être représenté s'il existe
- Les messages asynchrones correspondent à des signaux *(dans le diagramme de classes).*
- Utilité :
  - Affecter, initialiser ou afficher une valeur d'un paramètre spécifique : nomParametre = valeurParametre

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ ***3.1 Exemple de message asynchrone :***

- ✓ Représentation de l'allumage et de l'extinction de la lampe par un utilisateur



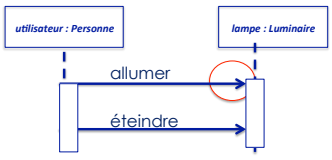
- Ces deux messages sont asynchrones :
  - L'utilisateur n'attend pas un message de retour de la lampe lui indiquant si elle est allumée (ou éteinte dans le second cas).

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ ***2.1 Exemple de message asynchrone :***

- ✓ Représentation de l'allumage et de l'extinction de la lampe par un utilisateur



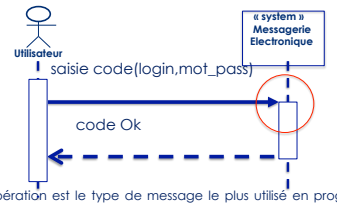
- Ces deux messages sont asynchrones :
  - L'utilisateur n'attend pas un message de retour de la lampe lui indiquant si elle est allumée (ou éteinte dans le second cas).

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ ***3.2 Message synchrone*** :

- ✓ L'émetteur reste en attente de la réponse à son message, avant de poursuivre l'exécution de ses opérations.



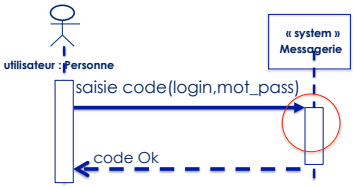
- L'invocation d'une opération est le type de message le plus utilisé en programmation objet ;
- Dans la pratique, la plupart des invocations sont synchrones, l'émetteur reste alors bloqué le temps que dure l'invocation de l'opération ;
- Le message de retour peut ne pas être représenté car il est inclut dans la fin d'exécution de l'opération de l'objet destinataire du message; le retour de message est implicite.

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ **3.2 Message *synchrone*** :

✓ L'émetteur reste en attente de la réponse à son message, avant de poursuivre l'exécution de ses opérations.



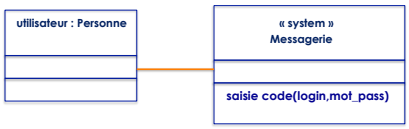
- Les méthodes correspondant aux messages synchrones seront définies dans un diagramme de classes.
- Les méthodes sont définies dans la classe du récepteur, et pas de l'émetteur du message.
- **On peut** associer aux messages d'appel de méthode un message de retour (en pointillés) marquant la reprise du contrôle par l'objet émetteur du message synchrone.

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ **3.2 Message *synchrone*** :

✓ L'émetteur reste en attente de la réponse à son message, avant de poursuivre l'exécution de ses opérations.



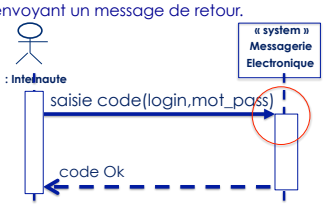
- **Envoyer un message et attendre la réponse pour poursuivre son activité revient à invoquer une méthode et attendre le retour pour poursuivre ses traitements.**
- *Si un objet A invoque une méthode d'un objet B, A reste bloqué tant que B n'a pas terminé.*

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ **3.2.1 Message *de retour***

✓ Le récepteur d'un message synchrone rend la main à l'émetteur du message en lui envoyant un message de retour.



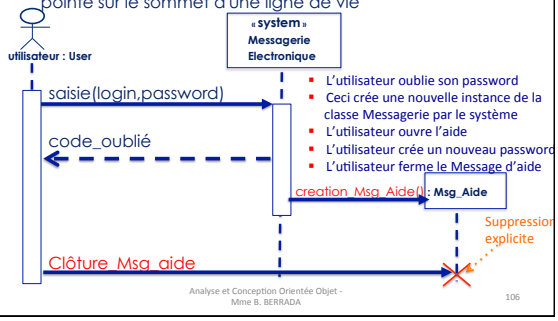
- Les messages de retour sont optionnels : ***la fin de la période d'activité marque également la fin de l'exécution d'une méthode***.
- Ils sont utilisés pour spécifier le résultat de la méthode invoqué.
- *Les messages de retour sont représentés en pointillés.*

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ **3.3 Message *de création et de suppression d'instance*** :

✓ La création d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie



- L'utilisateur oublie son password
- Ceci crée une nouvelle instance de la classe Messagerie par le système
- L'utilisateur ouvre l'aide
- L'utilisateur crée un nouveau password
- L'utilisateur ferme le Message d'aide

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ **3.3 Message de suppression d'instance** :

- La suppression d'un objet (participant) est indiquée par un **X** ;
- Une flèche de message entrant signifie qu'un objet en supprime explicitement un autre ;
- Un **X** à la fin d'une ligne de vie, montre que l'objet se supprime lui-même.
- Il est toujours utile d'utiliser un X pour indiquer qu'un objet n'est plus nécessaire, et qu'il est prêt à être collecté dans un environnement disposant d'un ramasse-miette ;
- Un **X** est également approprié aux opérations de clôture, pour signifier également que l'objet n'est plus utilisable.

Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ **3.4 Message *et événements*** :

UML permet de séparer clairement l'envoi du message, sa réception, ainsi que le début de l'exécution de la réaction et sa fin

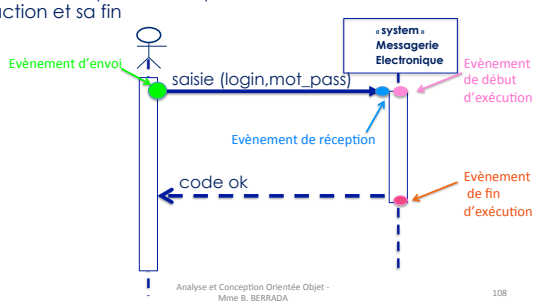


Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ **3.5 Message *trouvé :***

Un message trouvé est tel que l'événement de réception est connu, mais pas l'événement d'émission.

Une flèche partant d'une petite boule noire représente un message trouvé, comme illustré :

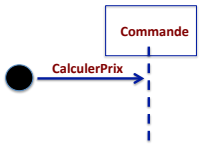


Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ **3.6 Message *perdu :***

Un message perdu est tel que l'événement d'envoi est connu, mais pas l'événement de réception.

Il se représente par une flèche qui pointe sur une petite boule noire tel qu'illustré ci-dessous :

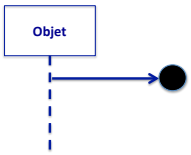
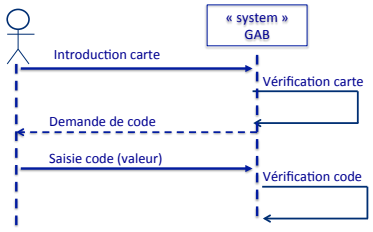


Diagramme de Séquence (DSE)

Sémantique des composants d'un DSE

➢ **3.7 Message réflexif :**

Il représente un message que s'envoie un objet à lui-même (self-call):



Il représente une activite interne à l'objet ou une abstraction d'une autre interaction (qu'on peut détailler dans un autre diagramme de séquence)

Description dynamique

DSE : EXERCICE 1

Représenter un diagramme de séquence, décrivant le traitement suivant :

1. Un gestionnaire reçoit une requête;
2. Le gestionnaire formule une requête à l'aide d'une instruction dans la base de données ;
3. L'instruction retourne un résultat ;
4. Après extraction des informations utiles, l'instruction est clôturée et la requête est terminée.

Solution 2

Description dynamique

DSE : EXERCICE 2

Exemple de diagramme de séquence qui décrit le scenario nominal du cas d'utilisation RETIRER ARGENT, proposé dans le TD 1 traitant de l'étude d'un Guichet Automatique de Banque (GAB).

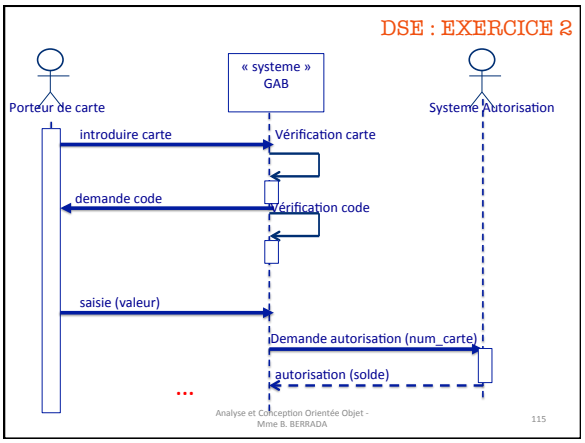
1. Il suffit de transcrire sous forme de diagramme de séquence les interactions citées dans la description textuelle du scenario nominal du cas d'Utilisation RETIRER ARGENT (pour un porteur de carte non client de la banque), précisée dans la diapositive suivante.

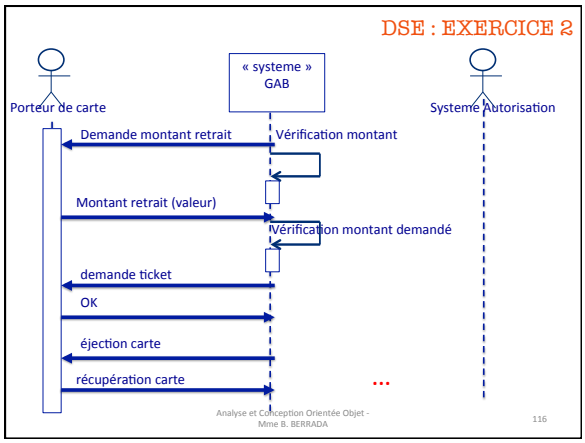
2. Utiliser les conventions graphiques :
   - Acteur Principal Porteur de carte, à gauche
   - Un participant représentant le GAB au milieu
   - L'acteur secondaire Sys. Auto à droite du GAB

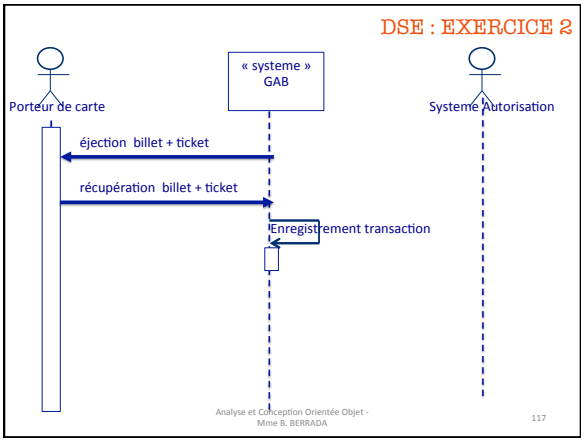
Description textuelle

Cas d'Utilisation RETIRER ARGENT

1. Le **Porteur de carte introduit sa carte dans le lecteur de cartes du GAB**
2. **Le GAB vérifie que la carte introduite est bien une carte bancaire**
3. **Le GAB demande au Porteur de carte de saisir son code d'indentification**
4. **Le Porteur de carte saisit son code d'identification**
5. **Le GAB compare le code d'identification avec celui qui est codé sur la puce de la carte**
6. **Le GAB demande une autorisation au Système d'autorisation**
7. **Le Système d'Autorisation donne son accord et indique le crédit hebdomadaire**
8. **Le GAB demande au Porteur de carte de saisir le montant désiré du retrait**
9. **Le Porteur de carte saisit le montant du retrait**
10. **Le GAB contrôle le montant demandé par rapport au crédit hebdomadaire**
11. **Le GAB demande au Porteur de carte s'il veut un ticket**
12. **Le Porteur de carte demande un ticket**
13. **Le GAB rend sa carte au Porteur de carte**
14. **Le Porteur de carte reprend sa carte**
15. **Le GAB délivre les billets et un ticket**
16. **Le Porteur de carte prend les billets et le ticket.**







TP 1

Exercice :

1. le Diagramme de Contexte préliminaire de l'utilisation du GAB, montrant les cas d'utilisation (ovales) reliés par des associations (lignes) avec leurs acteurs principaux (icônes ou « stick-man ») et secondaires.
2. Le diagramme de Séquence pour le scénario étudié
3. *Le diagramme de séquence en ajoutant des relations d'inclusion, d'extension pour illustrer les fragments d'interaction*

TP 3.1

Suite des Exercices :

Pour l'exercice 3,

3. Le diagramme de séquence en ajoutant des relations d'inclusion, d'extension pour illustrer les fragments d'interaction

3.1 Abordons tout d'abord la relation d'inclusion : le cas d'utilisation de base en incorpore explicitement un autre de façon obligatoire, à un endroit spécifique dans ses enchaînements.

On utilise cette relation pour décrire plusieurs fois le même enchaînement, et donc permettre de factoriser le comportement commun dans un cas d'utilisation à part.

TP 3.1

Suite des Exercices :

Le début de la description textuelle du scenario nominal du cas d'utilisation ***Retirer de l'argent***, (étapes de 1..5) va être similaire à tous les cas d'utilisation du client de la banque, en substituant « Porteur de carte » par « Client de la banque ».

1. Le Client de la banque introduit sa carte dans le lecteur de cartes du GAB
2. Le GAB vérifie que la carte introduite est bien une carte bancaire
3. Le GAB demande au Client de la banque de saisir son code d'indentification
4. Le Client de la banque saisit son code d'identification
5. Le GAB compare le code d'identification avec celui qui est codé sur la puce de la carte

Cet enchaînement est en outre complété par enchaînements alternatifs ou d'erreur A1 (code provisoirement erroné), E1 (carte non valide) et E2 (code d'identification définitivement erroné)

TP 3.1

Suite des Exercices :

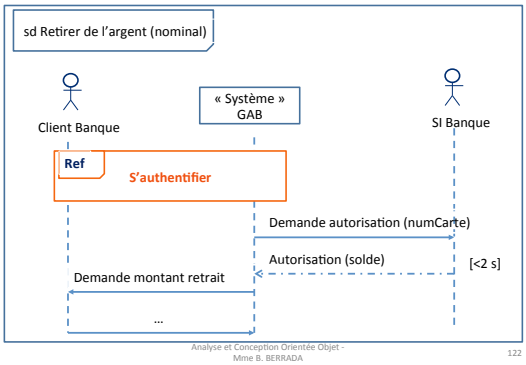
Un nouveau cas d'utilisation : S'Authentifier, inclus dans les précédents cas d'utilisation, entre autres Consulter Solde, Déposer Argent…

Avantages :

- ***S'Authentifier*** va contenir tous les enchaînements de 1 à 5
- Les descriptions textuelles des autres cas d'utilisation vont être simplifiées et focalisées sur les spécificités fonctionnelles
- Les descriptions redondantes seront éliminées.

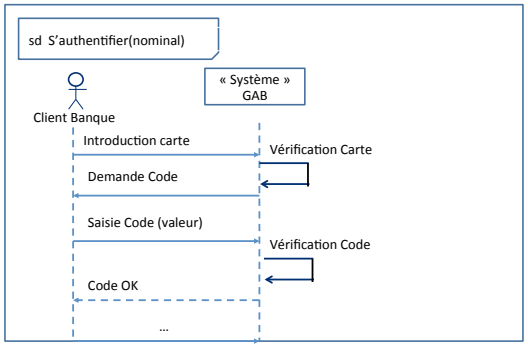
TP 3.1

Nouveau diagramme de séquence du système avec référence au cas inclus



TP 3.1

Le diagramme de séquence du fragment référencé



TP 3.2

Suite des Exercices :

Pour l'exercice 3,

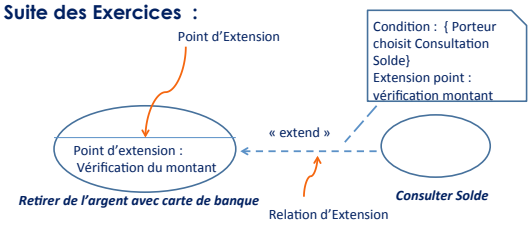
3. Le diagramme de séquence en ajoutant des relations d'extension

3.1 Rappelons la relation d'extension : le cas d'utilisation de base en incorpore implicitement un autre de façon optionnelle, à un endroit spécifique dans le cas d'utilisation qui procède à l'extension.

On utilise cette relation pour séparer un comportement optionnel ou rare du comportement obligatoire.

TP 3.2

Suite des Exercices :



Ce point d'extension doit être déclaré dans la description textuelle, par exemple en modifiant L'enchaînement nominal :

...
7. Le SI banque donne son accord et indique le solde hebdomadaire
8. Le GAB demande au Client banque de saisir le montant désiré du retrait
Point d'extension : Vérification montant
9. Client banque saisit le montant désiré
10. Le GAB contrôle le montant demandé par rapport au solde hebdomadaire.
...

TP 3.2

Nouveau diagramme de séquence du système avec référence au cas étendu

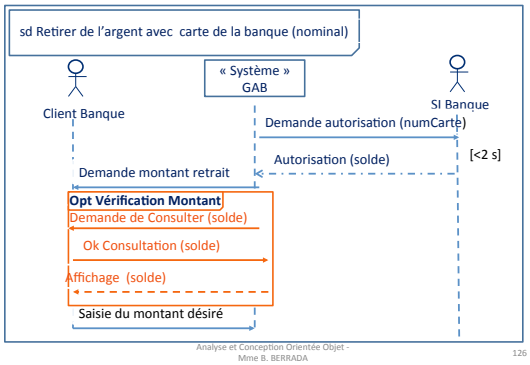


Diagramme de Séquence
Fragment d'interaction

Les cadres d'interactions sont utilisés comme ceux représentés, pour distinguer des sous-ensembles qui constituent des Fragments d'interactions.

Une indication dans le coin gauche, dans les cas précédents, ref ou opt portant le nom de l'interaction.

Treize opérateurs de fragments d'interaction (combiné) existent :

ref, opt, loop, alt, par, strict/weak, break, ignore/consider, critical, et assertion

Diagramme de Séquence
Fragment d'interaction

A l'image des deux premiers, l'opérateur :

alt : correspond à une instruction de test avec une ou plusieurs alternatives possibles. Seul le fragment dont la condition est vraie s'exécutera

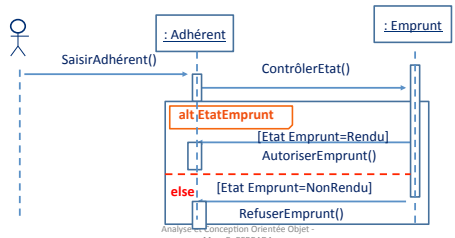


Diagramme de Séquence
Fragment d'interaction

A l'image des deux premiers, l'opérateur :

loop : correspond à une instruction de boucle qui permet d'exécuter une séquence d'interaction tant qu'une condition n'est pas satisfaite. Une garde indique la condition de l'iteration.

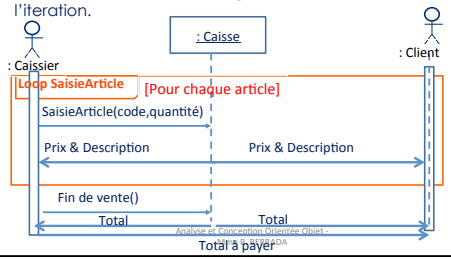


Diagramme de Séquence
Fragment d'interaction : par

par : correspond à une instruction qui permet de représenter deux séries d'interactions qui se déroulent en parallèle.

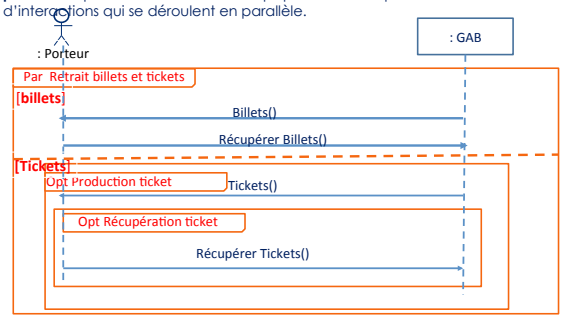


Diagramme de Séquence
Fragment d'interaction

Les autres opérateurs :

strict/weak sequencing : permettent de préciser que l'ordre des opérations(strict) ou pas important

Break : permet de représenter une situation exceptionnelle correspondant à un scénario de rupture (avec une condition de garde, par rapport au scénario général.

Consider/ignore : expriment respect. que des messages doivent être soit obligatoirement présents (consider), soit absents (ignore), sans incidence sue le déroulement des interactions.

Critical : indique une séquence d'interaction critique, donc **ne pouvant être interrompue** de l'importance des opérations traitées.

Diagramme de Séquence
Fragment d'interaction

Suite des autres opérateurs :

assertion : indique qu'une séquence d'interaction est l'unique séquence possible en considérant les messages échangés dans le fragment d'interaction. Toute autre configuration de message est invalide.

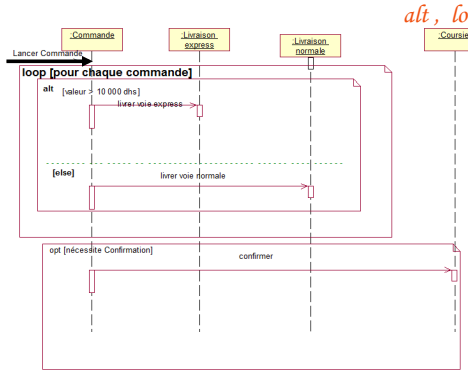
négative : indique le fragment représente une interaction invalide.

Fragment d'interaction
Exercice

Représenter, à l'aide d'un diagramme de séquence, le fragment d'interaction de la livraison d'une commande, soit par voie expresse soit par voie normale.

Voir Diapositive sur le sous-diagramme d'activité du traitement de la commande avec les 2 choix possibles de livraison

Solution – fragment combiné d'interaction alt , loop& opt



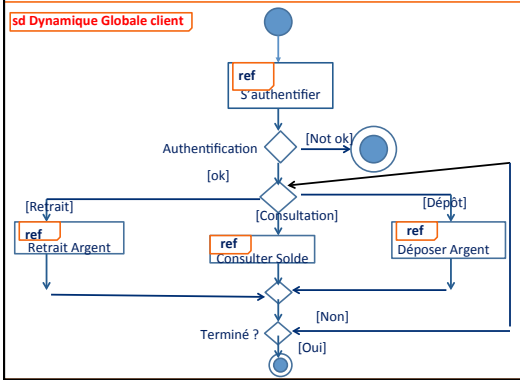
Dynamique globale
« Interaction Overview Diagram »

Diagramme de de vue globale/d'ensemble d'interaction

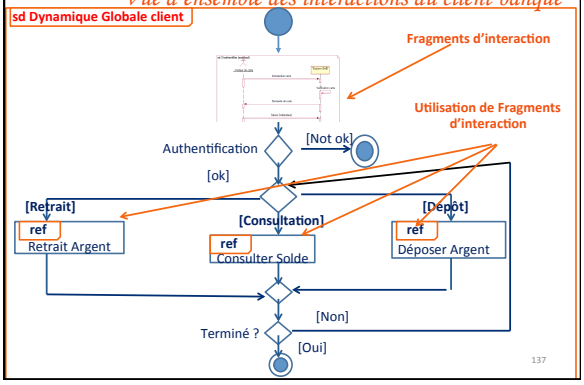
Ce diagramme est une fusion / combinaison du diagramme d'activité et du diagramme de séquence.

Il permet de représenter une vue générale des interactions décrites dans le diagramme de séquence et des flots de contrôle décrits dans les diagrammes d'activités.

Global Interaction Overview
Vue d'ensemble des interactions du client banque



Global Interaction Overview
Vue d'ensemble des interactions du client banque



Global Interaction Overview
Vue d'ensemble des interactions

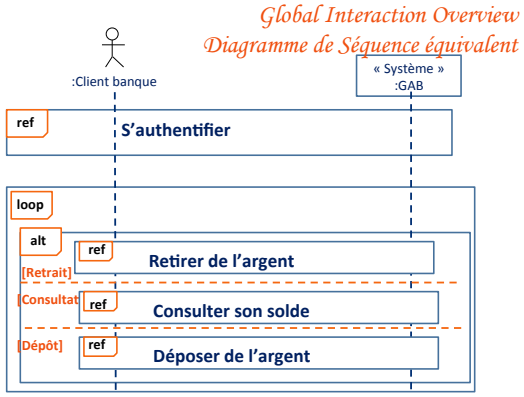
Le diagramme global d'interaction est un diagramme d'activité dans lequel sont représentés

- des **fragments d'interaction** et/ou
- des **utilisations de fragments d'interactions**

Ainsi, il est possible de voir :

- ***des choix de fragments d'interactions*** (fusion)
- des déroulements parallèles de fragments d'interaction
- ***des boucles de fragments d'interaction***.

Global Interaction Overview
Diagramme de Séquence équivalent



Global Interaction Overview

Les concepts manipulés ?

Le diagramme global d'interaction utilise les concepts du diagramme d'activité auquel on ajoute deux compléments:

➢ *Fragments d'interaction du diagramme de séquence.*
Il s'agit de la notion de fragment d'interaction vue dans le diagramme de séquence mais qui n'est pas détaillé à ce niveau :

➢ *Utilisations de fragments d'interaction* à l'aide de l'opérateur ref, comme le montre l'exemple ci-dessous :

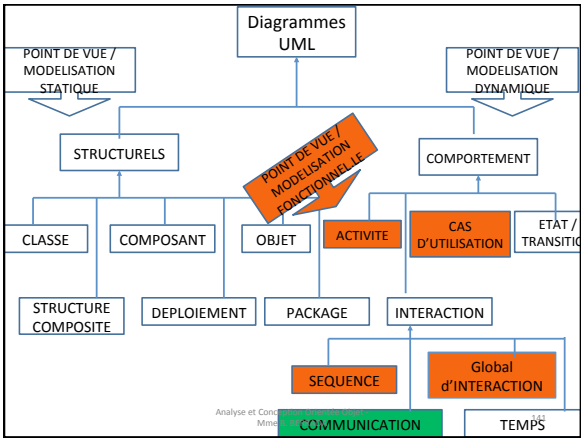


Diagramme de Communication

Le diagramme de communication (ex collaboration), un diagramme particulier de diagramme d'interaction, met l'accent sur :

➢ les liaisons de données entre participants à l'interaction

A la différence du DS, le DC permet de :

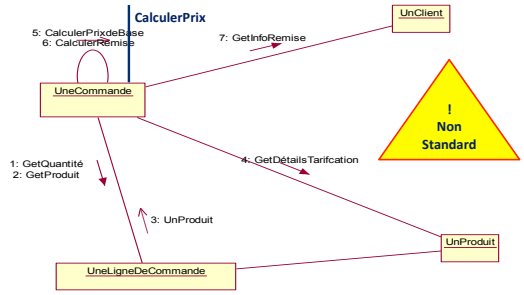
➢ Placer librement les participants;
➢ Tracer des liens pour montrer comment ils sont connectés
➢ Numéroter les messages pour suivre les séquences d'appel.

Le DC met davantage l'accent sur l'aspect spatial des échanges que sur l'aspect temporel

NB : Le DS est transformé en DC à l'aide de la fonction F5

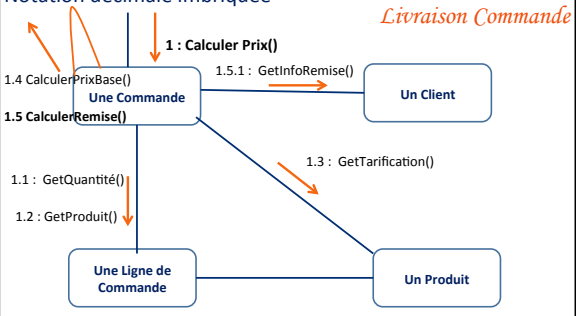
Diagramme de Communication
Livraison Commande

Notation couramment utilisée



Notation décimale imbriquée

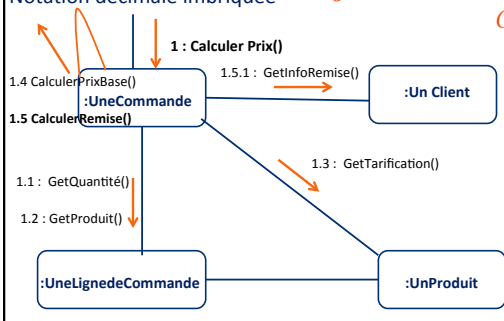
Diagramme de Communication
Livraison Commande



Raison d'être : Résolution de l'ambiguïté liée aux auto-appels

Notation décimale imbriquée

Diagramme de Communication
Concepts



1. **Rôle :** Participant à un échange correspondant à une ligne de vie dans le DS
2. **Message :** Appel à un appel d'opération effectuée par un rôle émetteur vers un Rôle récepteur

Diagramme de Communication
Concept : Rôle

1. Rôle :

- Chaque participant ayant un échange de message correspond à une ligne de vie du DSE
- Identifié par : **<nom de rôle> : type du message**

Exemple : **<nom de rôle> :UneCommande**, où l'un des deux est obligatoire avec le « : »

Le nom de rôle correspond au nom de l'objet et le nom du type à la classe

Diagramme de Communication
Concepts : Message

2. Message : Appel à une opération effectuée par un Rôle émetteur vers un Rôle récepteur

- Identifié par **<numéro> : nom ()**
- Syntaxe : [n° du message préc. reçu] . n° du message [itération] [condition] : nom du message

Exemple : 1.1.5.1σ [si commande>10.000dhs] : CalculerRemise()

Condition indiquant message à envoyer si commande>10000

Message précédent reçu Indiquant la chronologie du message

Message hiérarchique du message avec utilisation de lettre Indiquant la simultanéité d'envoi de message

Diagramme de Communication

Quand utiliser des diagrammes de communication ?

Ils sont à privilégier lorsque les liens entre participants doivent être mis en évidence.

De plus,
- ils sont plus faciles à modifier
- Ils constituent un bon moyen pour explorer des alternatives.

Les DSE sont par contre préférables lorsque il est besoin de mettre l'accent sur les séquences d'appels.

Diagramme de Communication
Exemple : création d'une agence

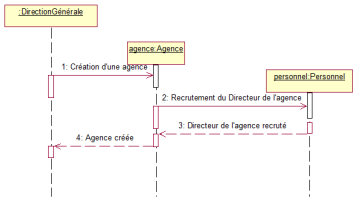


Diagramme de Communication
Exemple : création d'une agence

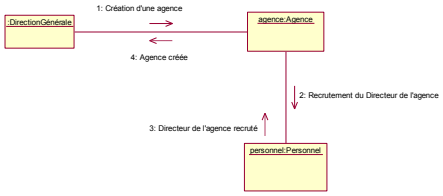


Diagramme de Communication
Exercice : Livraison Commande

Reprendre la Diapositive 70

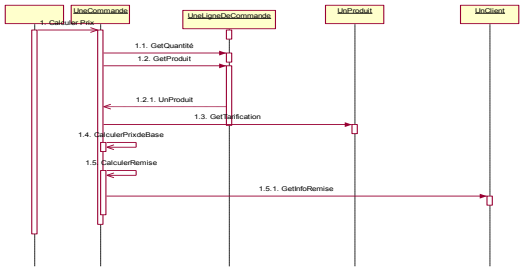
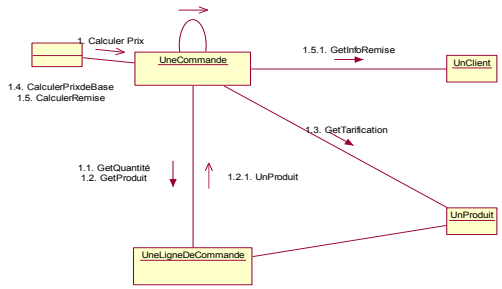


Diagramme de Communication
Exercice : livraison commande



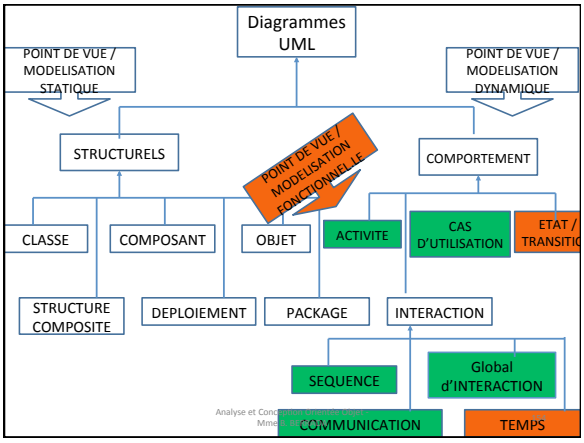


Diagramme de Temps (DTP)

Présentation générale :

Représente :

- Etats et interactions d'objets dans un contexte où le temps a une forte influence sur le comportement du système à gérer
- Changements d'états et des interactions entre objets liés à des contraintes de temps

Diagramme de Temps (DTP)

Concepts de base :

Trois concepts de base :

- **Ligne de vie :** Elle représente l'objet que l'on veut décrire. Elle se dessine de manière horizontale, et plusieurs lignes peuvent figurer dans un DTP
- **Etat ou ligne de temps conditionnée :** Différents états que peut prendre l'objet d'étude, listés en colonne et permettant de suivre le comportement de l'objet en ligne
- **Etats linéaires :** représentation de la succession des états est faite de manière linéaire à l'aide d'un graphisme particulier

Diagramme de Temps Exemple 1

Représentation et exemples:

Soit à représenter le dispositif de chauffe d'un fer à repasser à vapeur au moment de sa mise en service selon les règles suivantes :

- La pompe à eau qui remplit la chambre de chauffe s'active dès que le témoin d'eau interne le demande ;
- La pompe à eau se désactive dès que le niveau d'eau nécessaire est atteint ;
- le chauffage de l'eau permettant de produire la vapeur, se met en action à la première mise en service du fer à repasser dès que le niveau d'eau de la chambre de chauffe est suffisant ;
- Le chauffage initial de l'eau dure 3 s permettant de produire la vapeur

Diagramme de Temps Exemple de DT

Représentation en escalier ou sous forme de ligne:

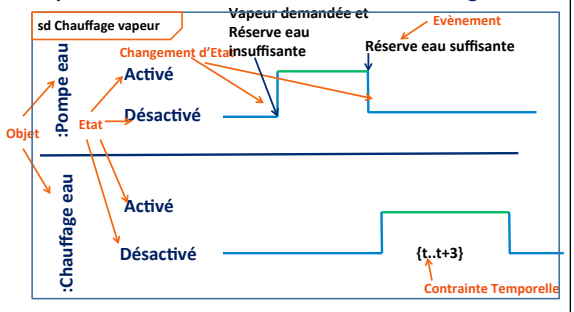


Diagramme de Temps Exemple de DT

Représentation linéaire ou sous forme de zone:

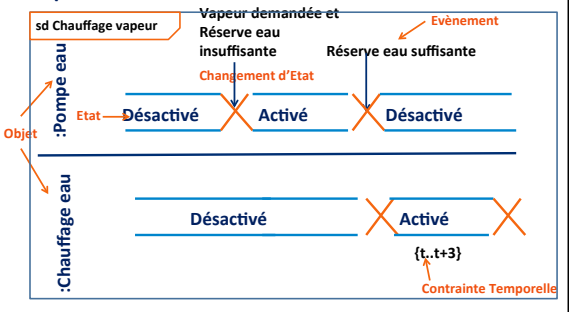


Diagramme de Temps
Exemple 2

Exemple 2:

Soit à représenter le dispositif d'une pompe et plaque chauffante d'une cafetière électrique.

Règles :

- Dix secondes au minimum doivent s 'écouler entre l'activation de la pompe et celle de la plaque chauffante ;
- Quand le réservoir d'eau est vide, la pompe s'arrête mais la plaque peut continuer à fonctionner 15 minutes de plus.

Diagramme de Temps
Exemple de DT

Représentation en escalier ou sous forme de ligne:

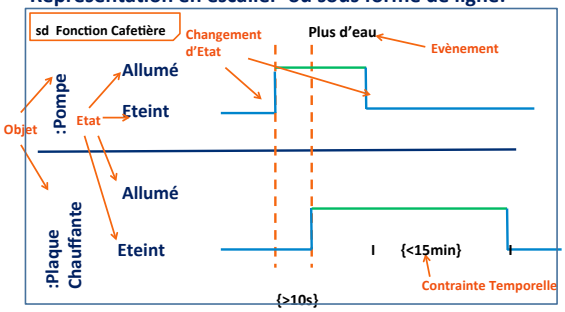
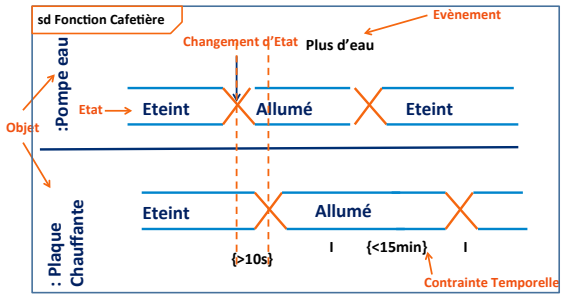


Diagramme de Temps
Exemple de DT

Représentation linéaire ou sous forme de zone:



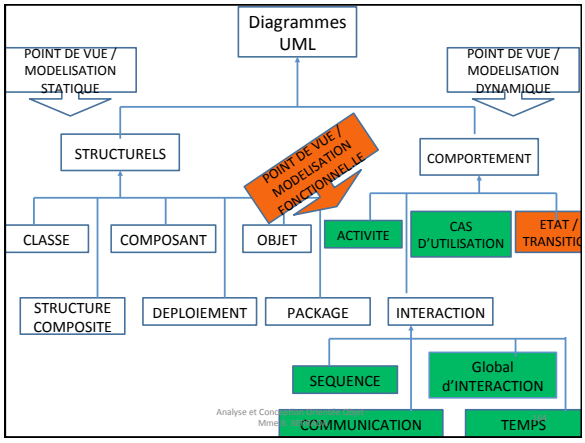


Diagramme d'Etat-Transition (DET)

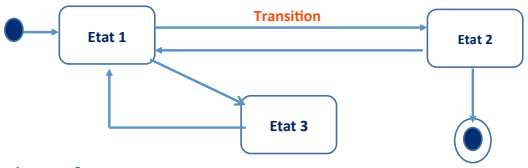
Présentation générale et concepts de base :

Les DET, depuis les années 60, constituent une technique courante permettant de :

- Décrire le comportement d'un système ;
- En particulier, pour mettre en évidence le comportement nominal d'un seul objet durant son cycle de vie au travers d'un enchaînement d'états caractéristiques de l' objet.

Diagramme d'Etat-Transition

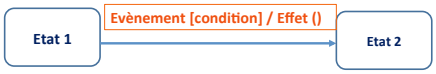
Formalisme d'Etat/Transition :



- **Etat ? Une situation durant la vie d'un objet pendant laquelle :**
  - ✓ Il attend un certain évènement
  - ✓ Il satisfait une certaine condition
  - ✓ Il exécute une certaine activité
- **Un objet passe par une succession d'états durant son existence**
- **Un état a une durée finie**
- **Le DET comporte un Etat initial (création de l'instance) et un Etat final (destruction de l'instance)**

Diagramme d'Etat-Transition

Formalisme d'Etat/Transition :



- La transition décrit la réaction d'un objet lorsqu'un évènement se produit
- Une transition possède :
  - ✓ Un évènement déclencheur (réception de message, appel opération, passage du temps, chgt dans la satisfaction d'une condition)
  - ✓ Une condition de garde
  - ✓ Un effet (action ou activité)
  - ✓ Un Etat cible

Diagramme d'Etat-Transition

Formalisme d'Etat/Transition :

- La transition décrit la réaction d'un objet lorsqu'un évènement se produit
- Une transition possède :
  - ✓ Un évènement déclencheur (réception de message, appel opération, passage du temps, chgt dans la satisfaction d'une condition)
  - ✓ Une condition de garde
  - ✓ Un effet (action ou activité)
  - ✓ Un Etat cible

Diagramme d'Etat-Transition
Exemple 1

Gestion commerciale de l'objet *Client* :

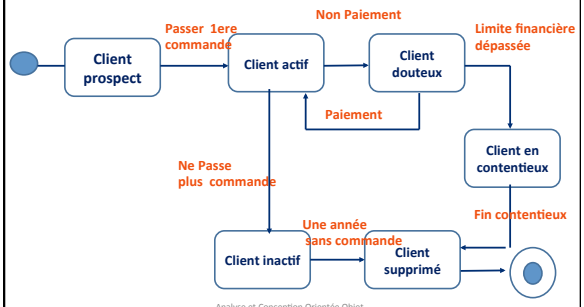


Diagramme d'Etat-Transition
Exemple 2

Ouverture/Fermeture d'un Coffre dans un château médiéval :

Dans ce château, les trésors sont conservés dans un coffre difficile à trouver.

Pour découvrir la serrure du coffre :

- Ôter une chandelle stratégique de son chandelier ;
- Pour révéler la serrure, la porte doit être fermée ;
- Dès que la serrure est visible, insérer la clef ;
- Le coffre n'est ouvert que si la chandelle est remise à sa place.

Diagramme d'Etat-Transition
Exemple

Ouverture/Fermeture d'un Coffre dans un château médiéval :

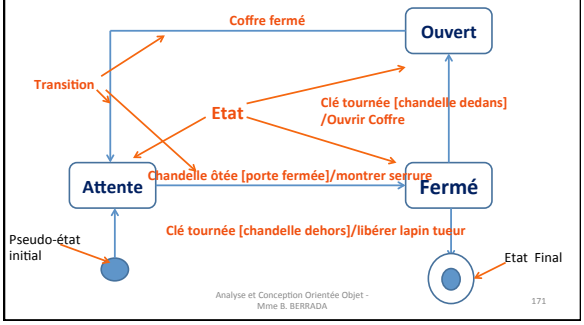


Diagramme d'Etat-Transition

- L'enchaînement de tous les états caractéristiques d'un objet constitue le diagramme d'état
- Un diagramme d'état débute toujours par un état initial. Et se termine par un ou plusieurs état.s final.ux
- Sauf dans le cas où le diagramme d'états représente une boucle.

Diagramme d'Etat-Transition
Compléments

- **Composition et décomposition**
- **Point d'entrée et de sortie**
- **Point de jonction**
- **Point de choix**
- **Etat historique**
- **Etat d'activité**

Diagramme d'Etat-Transition
Composition et décomposition d'Etat

Il est possible de décrire un DET à plusieurs niveaux.

- **Premier niveau :** Etats élémentaires + Etats composites
- **Second niveau** : Description des Etats composites dans un autre diagramme

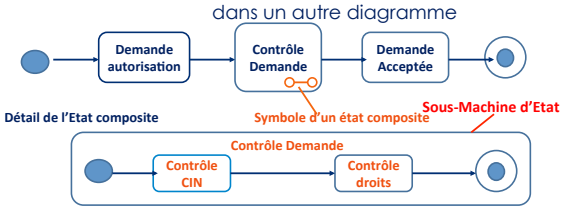


Diagramme d'Etat-Transition
Point d'Entrée et de Sortie

Sur une sous-machine d'Etat, il est possible de repérer un **point d'entrée** et un **point de sortie particuliers**.

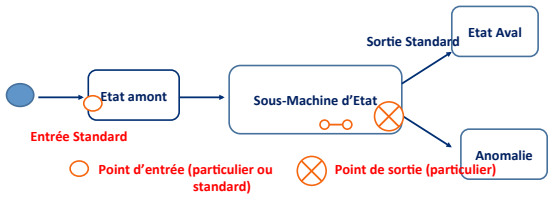


Diagramme d'Etat-Transition
Point d'Entrée et de Sortie

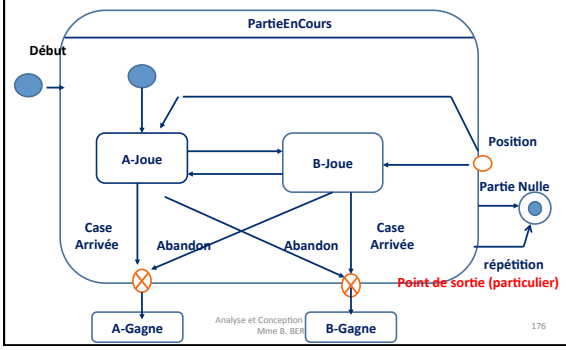
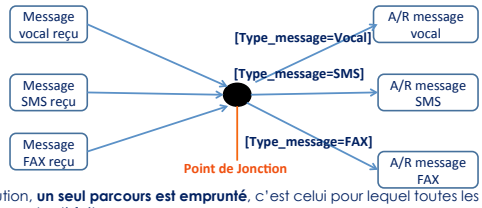


Diagramme d'Etat-Transition

Point de Jonction

Lorsque l'on veut relier plusieurs états vers d'autres, le point de jonction permet de décomposer une transition en deux parties, en indiquant si nécessaire, les gardes propres à chaque segment de la transition.



A l'exécution, **un seul parcours est emprunté**, c'est celui pour lequel toutes les conditions sont satisfaites.

Diagramme d'Etat-Transition

Point de Choix

Le point de choix se comporte comme un test de type :
si condition faire action 1 sinon faire action 2

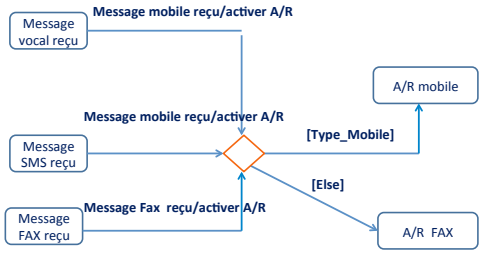


Diagramme d'Etat-Transition

Etat Historique

La mention de l'historisation d'un état composite permet de pouvoir indiquer **la réutilisation du dernier état historisé en cas de besoin.**

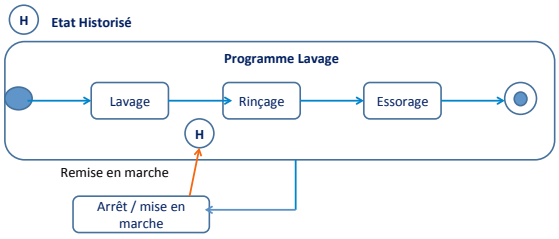


Diagramme d'Etat-Transition

Etat d'activité

Dans les états décrits jusqu'à présent, l'objet est inactif et attend l'évènement suivant avant de faire qqchose.

Il est possible de rencontrer des états dans lesquels l'objet exécute une activité durable.

Une activité durable à l'intérieur d'un état peut être soit :

- **Continue** : elle ne cesse que lorsque se produit un évènement qui fait sortir l'objet de l'état
- **Finie / automatique**: elle peut être interrompue par un évènement mais elle cesse d'elle-même au bout d'un certain temps

Diagramme d'Etat-Transition

Etat d'activité

Considérons un réveil du matin simplifié :

- Phrase 1 : On peut mettre l'alarme « on » ou « off »;
- Phrase 2 : Quant l'heure courante devient l'heure de l'alarme, le réveil sonne sans s'arrêter ;
- Phrase 3: On peut interrompre la sonnerie.

Diagramme d'Etat-Transition

Etat d'activité

Considérons un réveil du matin simplifié :

- Phrase 1 : On peut mettre l'alarme « on » ou « off »

Le réveil a 2 Etats distincts : Activé et Desactivé

Une simple action de l'utilisteur permet de passer d'un Etat à l'autre :

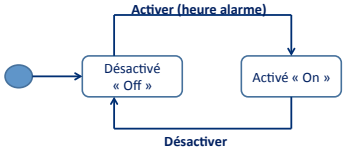


Diagramme d'Etat-Transition

Etat d'activité

Considérons les deux autres phrases :

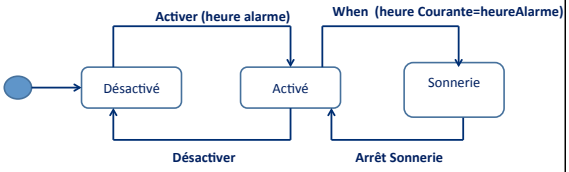
- Phrase 2 : Quant l'heure courante devient l'heure de l'alarme, le réveil sonne sans s'arrêter ;
- Phrase 3: On peut interrompre la sonnerie.

Le fait de sonner constitue un nouvel état pour le réveil. Il s'agit bien d'une période de temps durant laquelle le réveil effectue une activité (sonner) qui dure jusqu'à ce qu'un évènement vienne l'interrompre.

Diagramme d'Etat-Transition

Etat d'activité

Le passage de l'Etat Activé à l'Etat Sonnerie est déclenché par une transition « when » dite interne



En revanche, le retour de l'Etat sonnerie à l'Etat Activé s'effectue sur un évènement utilisateur.

Diagramme d'Etat-Transition
Etat d'activité

- Il y a une autre possibilité de sortie de l'Etat de sonnerie : quand le réveil arrête de sonner par lui-même, au bout d'un certain temps.
- Il suffit donc d'ajouter une activité durable Sonner à l'Etat sonnerie et une transition automatique en sortie de cet Etat.
- Le diagramme d'Etat complété est représenté dans le schéma suivant :

Diagramme d'Etat-Transition
Etat d'activité

Exemple d'un réveil matin

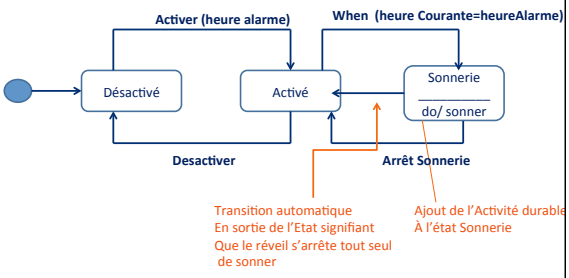


Diagramme d'Etat-Transition
Etat d'activité

- Autre question : l'utilisateur a-t-il le droit de désactiver le réveil (pas seulement la sonnerie) pendant qu'il sonne ?
- Solution :

Il faudra ajouter une transition déclenchée par Désactiver allant directement de Sonnerie à Désactivé.

Rappel

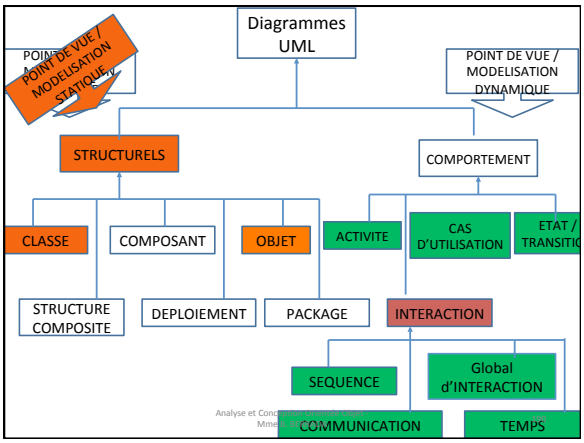
Après la description dynamique du langage UML 2 :

- Diagrammes de Cas d'Utilisation, modélisant à **Quoi** sert le système ;
- Diagrammes de séquence et d'activités, modélisant **Comment** les objets interagissent…

Dans ce qui suit, une description statique modélisant :

QUI intervient à l'intérieur du système, décrite par les Diagrammes de Classes et d'Objets

Cette représentation est située dans la partie de la modélisation statique, puisque le facteur temporel dans le comportement du système n'est pas considéré



Concepts abordés

- Classe et Diagramme de Classe ;
- Objet et Diagramme d'Objets ;
- Association pertinentes entre concepts ;
- Multiplicités aux extrémités des associations ;
- Classes d'associations, contraintes et qualificatifs ;
- Structuration d'un modèle en package

Diagramme de Classe

Le Diagramme de Classe constitue :

- Un point central dans le développement objet
  - En analyse : Structure des Entités manipulées par les utilisateurs ;
  - En conception : Structure d'un code orienté objet ou les modules du langage de programmation.

Cette représentation est centrée sur les notions :

- Objet
- Classe
- Types d'association entre classes.

Concept de Classe

Une Classe est une abstraction d'objets, c'est à dire :

- Une description d'un groupe d'objets ayant :
  - même structure (même ensemble d'attributs)
  - même comportement (mêmes opérations*)
  - une sémantique commune (domaine de définition)

Une classe déclare donc des propriétés communes à un ensemble d'objets

Exemples concrets

Exemples abstraits

Stratégie d'ordonnancement de processus

Concept de Classe
Attribut

Attribut :

- **Une propriété élémentaire dans une classe**
  Exemples : Pour une voiture, les attributs peuvent être :
  - marque,
  - vitesse courante,
  - numéro d'immatriculation,
  - puissance (fiscale),
- **Formalisme:**

visibilité nom: type [multiplicité] = valeurParDéfaut {propriété}

Exemple pour une personne :
prénom: String [3] {interdit}

Concept de Classe
Attribut particulier

- **Identifiant de la classe permet de repérer de façon unique les objets de la classe**

L'identifiant doit avoir une utilité pour le gestionnaire en lui permettant de se repérer sans ambiguïté dans un ensemble d'objets analogues

Rappels :

Chacun des autres attributs de la classe est relié à l'identifiant par une relation appelée **dépendance fonctionnelle**

- *Soit deux attributs a et b, on dit que b dépend fonctionnellement de a si connaissant la valeur de a, on peut trouver la valeur de b*

- Exemples :

1. Connaissant la référence d'un article, on peut trouver sa désignation, son prix unitaire, la quantité disponible en stock,...

2. Connaissant le numéro du client, on peut trouver son nom, son adresse, son téléphone, ses conditions commerciales,...

Concept de Classe
Opération

- **Une fonction, un service ou un traitement applicable aux objets de la classe :**
  (*définies de manière globale au niveau de la classe)*

  Exemples : Pour une voiture, les opérations peuvent être **:**
  rouler(vitesse), freiner(), arrêter() [1]
- **Formalisme :**

visibilité nom: type (liste_paramètres) : type_de_retour{propriété}

+ consulterSolde (date : Date) : Monnaie

La description de la liste des paramètres est semblable à celle des attributs :
direction* nom: type = valeur_par_défaut

*par défaut in, sinon, out, ou inout**.**

[1] *Le niveau de détail pour décrire les opérations dépend du niveau d'avancement dans l'étude. En phase d'analyse on ne s'intéresse qu'aux opérations qui définissent des comportements pertinents et on évite les opérations évidentes (consulter, modifier, supprimer,...)*

Concept de Classe
Distinction Opération / Méthode

Une opération est invoquée sur un objet, lors de l'appel de procédure

Une méthode est le corps de la procédure

La distinction entre les deux est pertinente en cas de redéfinition de l'opération de la classe par des sous-classes.

Dans ce cas :

Une opération et plusieurs méthodes qui l'implémentent.

Concept de Classe
Représentation

Chaque classe est divisée en 4 compartiments :

- Le nom de la classe,
- ses propriétés,
- ses opérations ,
- ses responsabilités et exceptions (en option)

NomDeLaClasse
attributs
operations
responsabilités/exceptions

Compte
numéro:string solde : reel decouvertmax : entier
consulterSolde() : reel *créditer()* *Débiter(montant : reel)*

Compte
numéro:string solde : reel decouvertmax : entier

Description réduite à 2 compartiments

Compte

Description réduite à la désignation de la classe

Conventions : Contrainte {montant <= solde }

Les noms de classes commencent par une majuscule
Les noms d 'attributs et d'opérations commencent par une minuscule
Pas de représentation particulière pour les identifiants

Concept de Classe
Représentation

NomDeLaClasse
nomAttribut1 : typeDeDonnées1 = valeurParDéfaut1 **nomAttribut2 : typeDeDonnées2 = valeurParDéfaut2** **...**
nomOperation1 (listeDesArguments1) : typeDuRésultat1 ***nomOperation2 (listeDesArguments2) : typeDuRésultat2*** **...**

Signature d'une opération : nom, arguments, résultat.

Pour chaque argument d'une opération on peut noter de manière optionnelle le sens de flux (in, out, inout), le type et la valeur par défaut.
sensDeFlux nomArgument : type = valeurParDéfaut

Concept de Classe
Visibilité des attributs et opérations

- **Signification du type de visibilité autorisé aux autres classes**
- Les droits d'accès associés à chaque niveau de confidentialité sont :
  - **Public (+)** : Attribut / Opération accessible par tous
  - **Privé (-)** : Attribut / Opération seulement visible à l'intérieur de la classe
  - **Protégé (#)** : Attribut / Opération visible seulement à l'intérieur de la classe et pour toutes les sous-classes de la classe
  - **Paquetage (~) ou rien** : Attribut / Opération seulement visible à l'intérieur du paquetage où se trouve la classe.

- ***Utile lors de la conception et de l'implémentation, pas avant !***
- ***N'a pas de sens dans un modèle conceptuel (abstrait)***
- ***N'utiliser que lorsque nécessaire***
- ***La sémantique exacte dépend du langage de programmation !***

Concept de Classe
attributs et opérations de Classe

Un attribut ou une opération peut être défini.e non pas au niveau des instances de Classe, mais au niveau de la classe.

Il s'agit soit :

- Un attribut, une **constante** pour toutes les instances d'une Classe
- Une opération d'une Classe abstraite, *dite statique (en langage de prg)*

Formalisme :

- **Soulignement de l'attribut ou de l'opération (créer, par exemple)**

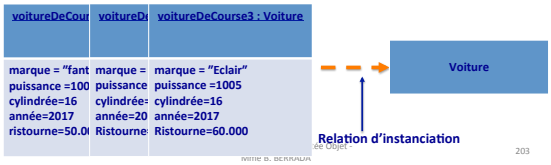
Concept de Classe
attributs et opérations de Classe

Voiture
- marque - puissance - cylindre - année - chiffreAffaires - ristourne
- modifier () *+ démarrer ( )* *- rouler()* *+ freiner* *# arrêter()*

Formalisme :
- **Soulignement de l'attribut ou de l'opération (créer, par exemple)**

Notion d' Objet

- **Une instance ou occurrence d'une classe, c'est à dire :**
  - Il possède une référence implicite à sa classe ;
  - Il représente une valeur particulière de la classe.
- **Il est conforme à la description fournie par la classe**
  - Il admet au plus une valeur pour chaque attribut de la classe
  - Il est possible de lui appliquer toute opération définie dans la classe



Notion d'Objet
Identité

- Chaque objet possède une identité intrinsèque.
- Deux objets sont distincts même si toutes les valeurs de leurs attributs sont identiques.

iphone1 : Smartphone	iphone2 : Smartphone
IMEI = « 46 578976 598876 6 » Désignation =« iphone 6S » Modèle = « MKQR2QL/A » Version = « 10.0.2(14A548) » Couleur = « blanc » Capacité = 60 Go Disponible = 40 Go	IMEI = « 87 774433 779900 5 » Désignation =« iphone 6S » Modèle = « MKQR2QL/A » Version = « 10.0.2(14A548) » Couleur = « blanc » Capacité = 60 Go Disponible = 40 Go

L'identité permet de distinguer tout objet de façon non ambiguë

Notion d'Objet

Il est possible de :
- Mentionner des valeurs pour des attributs significatifs et
- Nommer des états dans lesquels sont les objets.

* L'objet admet au plus une valeur pour chaque attribut

Notion d'Objet

Des Objets peuvent être ajoutés ou détruits pendant l'exécution, la valeur des attributs peut changer

Notation pour les Objets

leCompteDeSalim
numéro = 12122012 solde = 10000,50 découvertMax = -1000

La structure d'une classe est constante

Notation pour une Classe

Compte
numéro : entier solde : reel; découvertMax : entier
consulterSolde() *créditer()* *débiter()*

Notion d'Objet
Notation

Différentes représentations des objets :

nom_objet

Désignation directe

nom_objet : Nom_classe

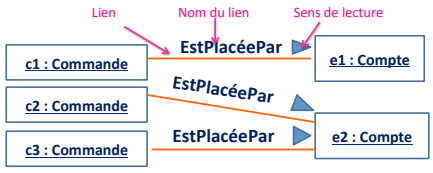
Désignation avec nom de classe

: Nom_classe

Désignation anonyme

Notion d'Objets
Lien

Un lien indique une connexion **entre deux objets**



MODÈLE DE DIAGRAMME D'OBJETS

Une association entre classes exprime les liens structurels entre leurs instances

Association simple vs Lien

Une association décrit un **ensemble de liens** entre 2 classes(binaires ou plusieurs classes (n-aires)

MODÈLE DE DIAGRAMME DE CLASSES

- L'association est symbolisée par une ligne qui lie deux classes :
  - les associations sont généralement bidirectionnelles.
  - les associations peuvent être réflexives.
- La navigation est symbolisée par une flèche qui indique le sens
- de lecture du nom de rôle.

*Conventions :
Les noms des liens sont des formes verbales et commencent par une majuscule.

Récapitulatif
Association vs. Lien

- Un lien lie deux objets
- Une association lie deux classes
- Un lien est une instance d'association
- Une association décrit un ensemble de liens
- Des liens peuvent être ajoutés ou détruits pendant l 'exécution, (ce n 'est pas le cas des associations)

Le terme "relation" ne fait pas partie du vocabulaire UM

Concept de Classe
Multiplicité

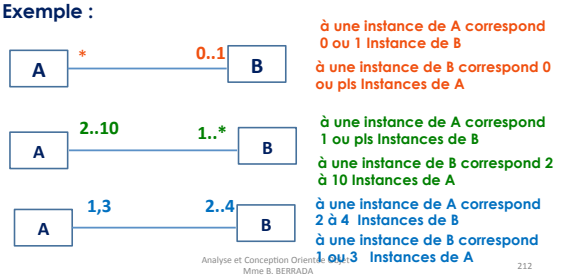
La **multiplicité** spécifie le nombre d'instances d'une classe pouvant être liées à une seule instance d'une autre classe associée.

- **On en place de chaque côté des associations**
  - Une multiplicité d'un côté spécifie combien d'objets de la classe du côté considéré sont associés à un objet donné de la classe de l'autre côté.
- **Syntaxe : min..max,** où :
  - min et max sont des nombres représentant respectivement les nombre minimaux et maximaux d'objets concernés par l'association.

Concept de Classe
Multiplicité

La **multiplicité/Cardinalité** d'une propriété est une indication du nombre d'objets qui peuvent participer à l'association.

Exemple :



Concept de Classe
Ecritures des multiplicités

La **multiplicité** spécifie le nombre d'instances d'une classe pouvant être liées à une seule instance d'une autre classe associée.

- **Certaines écritures possibles :**
  - * à la place de max signifie *plusieurs* sans préciser de nombre
  - n..n se note aussi n pour *exatement n*
  - 0..* se note aussi *
  - n1, n2 se lit n1 ou n2
- **Exemples** :
  - 1..* : au minimum 1 mais sans maximum
  - 2..4 : entre 2 et 4 (mais jamais 0, par exemple)
  - 2,4 : 2 ou 4
  - * : autant qu'on le souhaite

Concept de Classe
Exemple de Multiplicité

La **multiplicité/Cardinalité** d'une propriété est une indication du nombre d'objets qui peuvent participer à l'association.

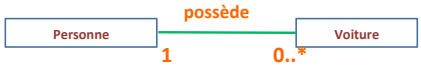
Comment interprétez-vous la multiplicité 1..5 ?

Ici, le 1..5 s'interprète comme suit :

à un objet donné de la classe Article, on doit associer :
au minimum 1 objet de la classe Catégorie
et on peut en associer :
au maximum 5.

Concept de Classe
Exemple 2 de Multiplicité

La **multiplicité/Cardinalité** d'une propriété est une indication du nombre d'objets qui peuvent participer à l'association.



Comment interprétez-vous cette association ?

Cette association s'interprète comme suit :

« Une personne possède 0 ou plusieurs voitures »
« Une voiture a 1 et 1 seul propriétaire »

Concept de Classe
Multiplicité

La **multiplicité** spécifie le nombre d'instances d'une classe pouvant être liées à une instance d'une autre classe associée.

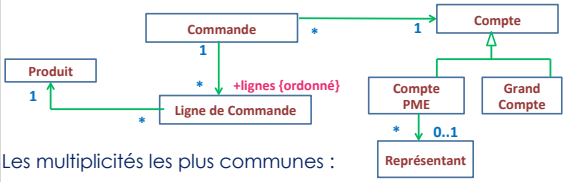
Elle contraint le nombre d'objets liés.

Les différents termes se rapportant à la multiplicité :

- **Optionnel** implique une borne inférieure de 0
- **Obligatoire** implique une borne inférieure de 1 ou plus
- **Monovalué** implique une borne supérieure de 1
- **Multivalué** implique une borne supérieure au-delà de 1 : généralement *

Concept de Classe
Multiplicité

La **multiplicité (cardinalité)** d'une propriété est une indication du nombre d'objets qui peuvent participer à l'association.



Les multiplicités les plus communes :

1 Exemple : Une commande doit avoir exactement un Client

0..1 Exemple : Un Client professionnel peut avoir ou ne pas avoir de repr

***** Exemple : Un Client peut ne pas passer commande et il n'y a pas de limite au nombre de commandes qu'il peut passer

Récapitulatif - Classe
Multiplicité

Plus généralement, les multiplicités sont définies avec :

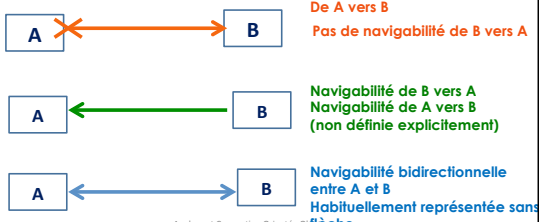
- Une borne inférieure et une borne supérieure
  - ✓ Exemple : 2..4
- La borne inférieure est zéro ou nombre positif
- La borne supérieure est un nombre strictement positif ou *
- Si les deux bornes sont identiques, utiliser un seul nombre
  - ✓ Exemple : 1..1 équivaut à 1
- 0..* équivaut à *

Concept de Classe
Navigabilité

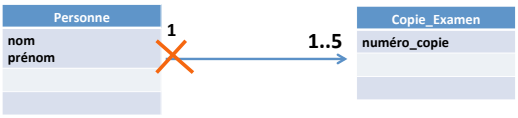
Par défaut, une association est navigable dans les deux sens.

Cependant, la navigabilité indique si l'association fonctionne **de** manière **unidirectionnelle** ou **bidirectionnelle**

Exemple :



Concept de Classe
Navigabilité – Exemple 1



Quelle est votre interprétation ?

1. A une personne sont associées ses copies d'examen

L'inverse est-il pas possible ?

2. Il n'est pas possible d'associer à une copie d'examen, une personne (auteur) .

Concept de Classe
Navigabilité – Exemple 2

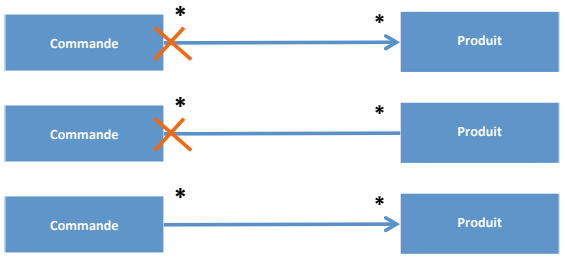
Quelle est votre interprétation ?

1. La terminaison du côté de la classe *Commande* n'est pas navigable : les instances de la classe *Produit* ne stockent pas de liste d'objets(*) du type *Commande*.

L'inverse est-il pas possible ?

2. Inversement, la terminaison du côté de la classe *Produit* est navigable : chaque objet commande contient une liste(*) de produits.

Concept de Classe
Navigabilité



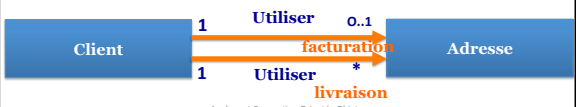
La navigabilité est représentée uniquement sur l'une des extrémités d'une association,
implicitement : l'association ne peut être traversée que dans un seul sens.

Ces trois notations ont la même sémantique.

Concept de Classe
Notion de Rôle / Nom d'extrémité d'association

1. Pour améliorer la lisibilité des associations

2. Pour préciser des modalités différentes, impliquant les mêmes objets : associations multiples



Concept de Classe
Notion de Rôle / Nom d'extrémité d'association

Exemple 1 : Représenter ce qui suit

1. Une liste chaînée est une structure informatique représentant une collection ordonnée d 'éléments.

2. L'accès aux éléments d'une liste s'effectue d'une manière séquentielle : chaque élément de la liste permet l'accès au suivant par un pointeur.

3. Un exemple de la liste composée de 3 éléments

Concept de Classe
Notion de Rôle / Nom d'extrémité d'association

Question : Proposer un diagramme de classe pour modéliser cette structure

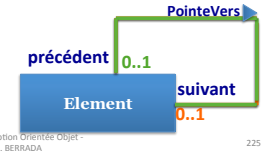
R1. Les éléments de la liste vont être représentés par une classe Éléme

R2. Un élément a au plus un précédent

R3. Un élément a au plus un suivant

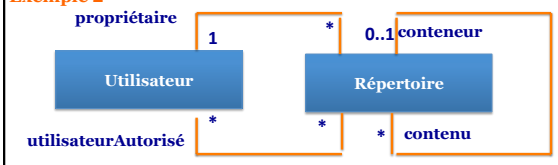
R4. Puisque l 'association PointeVers relie la classe classe à elle-même : elle est réflexive

R5. Ses rôles sont appelés :



Concept de Classe
Notion de Rôle / Nom d'extrémité d'association

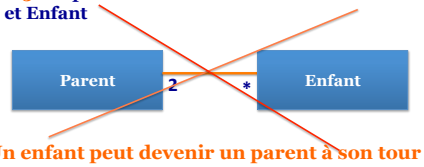
Exemple 2



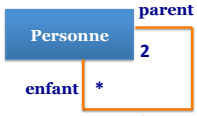
3. Les noms d'extrémités sont nécessaires pour le associations entre deux objets de la même classe associations réflexives.

Concept de Classe
Notion de Rôle / Nom d'extrémité d'association

Exemple 3 : Représentons l'association entre 2 Classes : Parent et Enfant



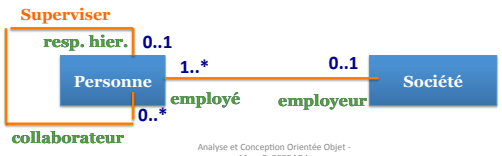
Un enfant peut devenir un parent à son tour …



Les noms d'extrémités sont nécessaires pour modéliser plusieur références à la même classe.

Concept de Classe
Notion de Rôle / Nom d'extrémité d'association

1) Dans toute société, il y a au moins une personne qui n 'a pas de responsable hiérarchique (le directeur de la société).
2) Une personne ne peut pas avoir plus d'un responsable hiérarchique.
3) Une personne peut avoir zéro ou plusieurs collaborateurs.
4) Une personne ne peut pas être employée par plusieurs sociétés.
5) Une société emploie une ou plusieurs personnes.



Concept de Classe
Association Réflexive

Une association réflexive, est une association, dont les deux extrémités sont liées à une même classe.

- La signification des extrémités d'une association réflexive doit être clarifiée par l'indication d'un rôle.

Concept de Classe
Contrainte

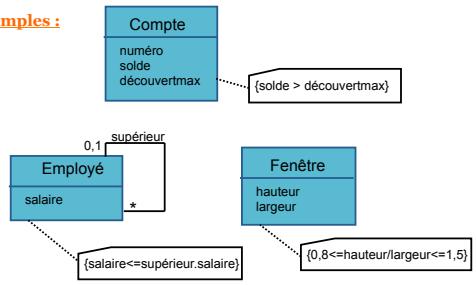
- Les contraintes montrent des règles de gestion qui devront être implémentées dans le système final.
- Il est possible d'en trouver plusieurs, au risque de surcharger le modèle, et le rendre illisible.
- Il faudra donc les répertorier dans le texte accompagnant le modèle, et choisir les plus importantes.

Concept de Classe
Exemples de contraintes

- Contraintes sur attributs
- Contraintes entre associations
- Contraintes sur l'extrémité d'une association
- Contraintes entre classes (ou contraintes de spécialisation), à voir dans la partie « Généralisation / Spécialisation, plus loin)

Concept de Classe
Contraintes sur attributs

Exemples :

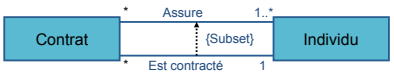


Concept de Classe
Contraintes entre associations

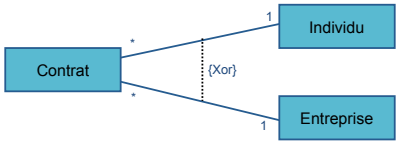
- Contrainte sur associations reliant deux même classes ;
- Contrainte sur associations reliant des classes différentes

☞ **Contraintes d'inclusion (subset)**
☞ **Contrainte d'exclusion (xor)**

Concept de Classe
Exemples de Contraintes entre associations



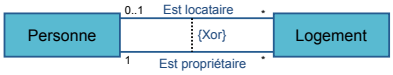
Le contractant d 'un contrat d'assurance fait obligatoirement partie des assurés



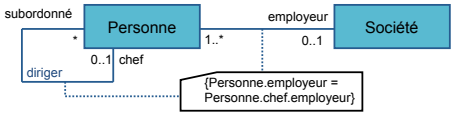
Un contrat assure un individu ou une entreprise mais pas les deux

235

Concept de Classe
Exemples de Contraintes entre associations



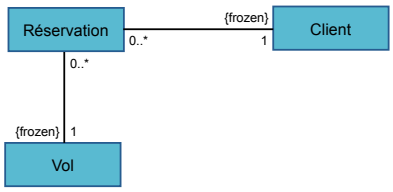
Une même personne est soit locataire, soit propriétaire du logement dans lequel elle habite (exclusion).



Le supérieur hiérarchique d'une personne a nécessairement le même employeur que cette personne.

236

Concept de Classe
Exemple de Contraintes sur l'extrémité d'une associatio



{frozen} : l'instance de la classe à cette extrémité d'association est non modifiable

237

Concept de Classe
Autres Contraintes

D'autres propriétés particulières sont proposées pour préciser la sémantique d'une association

- **Ordre de tri**
- **Mise à jour**

Une seule règle : Placer la contrainte entre { }

Langages :

- **OCL, ex : {ordered }**
- **Naturel, ex : {tarif exceptionnel : 50% aux étudiants }**

Concept de Classe
Autres Contraintes

Des propriétés particulières sont proposées pour préciser la sémantique d'une association

Ordre de tri

- Pour une association de **multiplicité supérieure à 1**, les liens peuvent être :
  - *Non ordonnés (valeur par défaut) ou* **{bag}**
  - *Ordonnés ou triés (sur une valeur interne)* **{ordered}* ou {sequence}****

Exemple 1 :

* ordered : doublons non autorisés
** sequence : doublons autorisés

Concept de Classe
Autres Contraintes

Des propriétés particulières sont proposées pour préciser la sémantique d'une association

Ordre de tri

- Pour une association de multiplicité **supérieure à 1**, les liens peuvent être :
  - *Non ordonnés (valeur par défaut) ou* **{bag}**
  - *Ordonnés ou triés (valeur interne)* **{ordered}* {sequence}****

Exemple 2 :

* ordered : doublons non autorisés
** sequence : doublons autorisés

Concept de Classe
Autres Contraintes

Des propriétés particulières sont proposées pour préciser la sémantique d'une association

Ordre de tri

- Pour une association de **multiplicité supérieure à 1**, les liens peuvent être :
  - *Non ordonnés (valeur par défaut) :* **{bag}**
  - *Ordonnés ou triés (valeur interne) :* **{sequence} ou {ordered}**

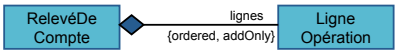
Exemple 3

Classe
Autres Contraintes

Il est possible d'indiquer des contraintes particulières relatives aux conditions de mise à jour :

- **{interdit}** : interdit l'ajout, la suppression ou la mise à jour des liens
- **{ajout seul}** : n'autorise que l'ajout de liens.

Concept de Classe
Contraintes sur l'extrémité d'une association

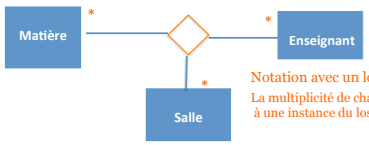


{ordered} : les éléments de la collection sont ordonnés
{addOnly} : il est impossible de supprimer un élément

243

Concept de Classe
Association n-aires

- La plupart des associations sont dites binaires car elles relient deux classes
- Des associations de dimension supérieure à 2 peuvent néanmoins exister : **Ternaire**



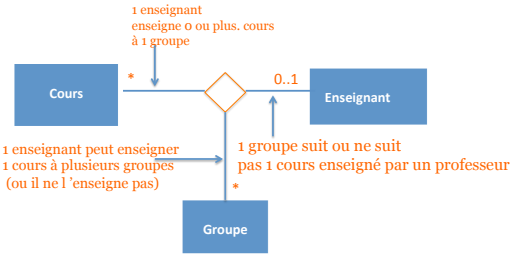
Notation avec un losange central
La multiplicité de chaque classe s'applique à une instance du losange.

Les associations n-aires sont peu fréquentes et concernent surtout les cas où les multplicités sont toutes « * ».

Dans la plupart des cas, on utilisera plus avantageusement des classes-association ou plusieurs relations binaires

Concept de Classe
Association n-aires

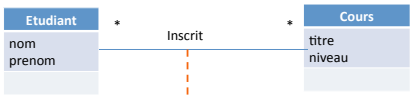
Pour un couple d'instances de la classe A et de la classe B, il y a au minimum r1 instances de la classe C et au maximum r2 instances.



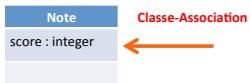
Dans la plupart des cas, on utilisera plus avantageusement des classes-association ou plusieurs relations binaires

Concept de Classe
Classe-association

Exemple 1



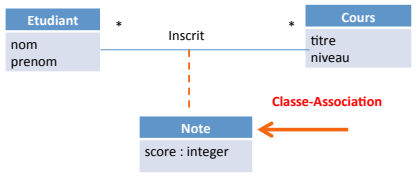
Il s'agit maintenant d'étendre le modèle ci-dessus de façon à représenter **la note obtenue par** un étudiant.



Classe Association : Permet de décrire soit des attributs, soit des opérations propres à l'association, donc qui ne sont disponibles dans aucune des classes qu'elle lie.

La Classe association est reliée par un trait en pointillé aux classes associées.

Concept de Classe
Classe-association

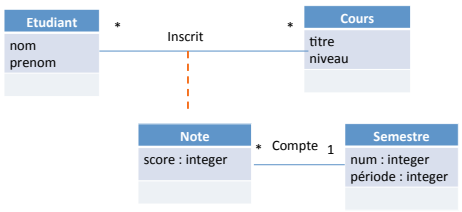


Comme les liens d'une association

- **les instances d'une classe-association dérivent leur identité des instances des classes constituantes**
- **La classe –association ajoute une contrainte : il ne peut y avoir qu'une seule instance pour un lien entre 2 objets qcqs.**
- **Une classe-association peut avoir des attributs et des opérations et peut participer à une association**

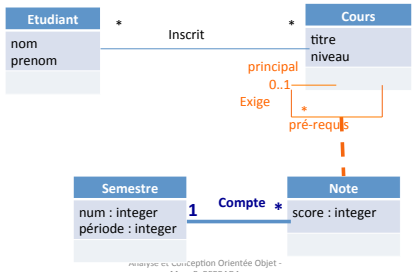
Concept de Classe
Classe-association

- **Une classe-association peut avoir des attributs et des opérations et peut participer à une association**

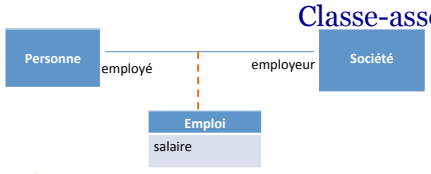


Concept de Classe
Classe-association

- **Une classe-association peut avoir des attributs et des opérations et peut participer à une association**



Concept de Classe
Classe-association



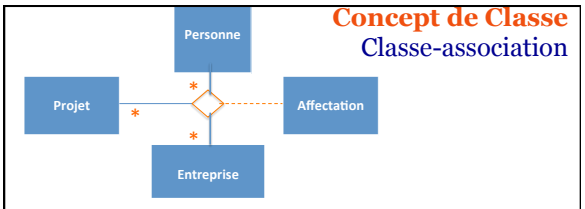
Interprétation ? Une personne ne peut pas avoir deux emplois dans une même société…

Comment modifier le DC pour qu'une personne ait pls emplois dans 1 même société ?

Une classe-association peut être transformée en classe à part entière

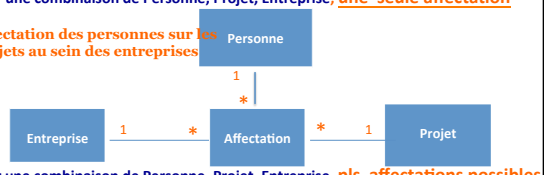
La transformation d'une classe-association en classe peut modifier la signification du modèle…

Concept de Classe
Classe-association



Pour une combinaison de Personne, Projet, Entreprise, une seule affectation

Affectation des personnes sur les projets au sein des entreprises

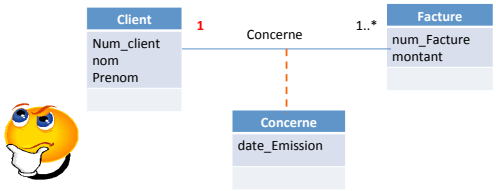


Pour une combinaison de Personne, Projet, Entreprise, pls affectations possibles

Concept de Classe
Classe-association

Attention

- Une association à multiplicité 1 (1..1) n'est jamais représentée à l'aide d'une classe association.



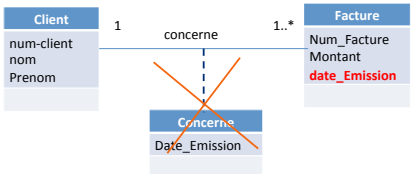
Effectivement, chaque facture ne possède qu'une et une seule date d'émission,

Concept de Classe
Classe-association

Attention :

- Une association à multiplicités 1 (1..1) n'est jamais représentée à l'aide d'une classe association.

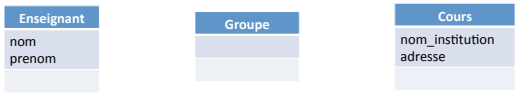
Dans ce cas, les attributs en question migrent dans la classe concernée par les multiplicités 1 (1..1).



Puisque la facture ne possède qu'une et une seule date d'émission, l'attribut DateEmission a migré dans la classe Facture

Concept de Classe
Classe-association ou Classe n-aire

un Enseignant enseigne à un ou pls groupes d'étudiants

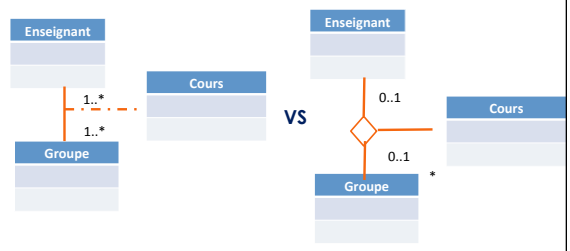


- Question : Quel lien faut-il privilégier entre les 3 différentes classes ?

Concept de Classe
Association n-aires

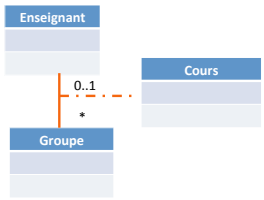
Modélisation suivante à considérer :

- un Enseignant enseigne à un ou pls groupes d'étudiants



Concept de Classe
Association n-aire

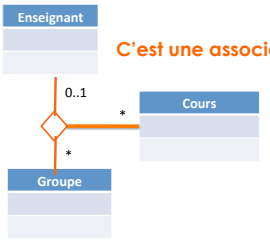
Interprétation :



Un cours ne peut exister que s'il existe un lien entre un objet Enseignant et un objet Groupe. Quand le lien est rompu (effacé), le cours l'est également.

Concept de Classe
Association n-aire

Un cours doit pouvoir exister indépendamment de l'existence d'un lien :
Exemple : on a pas encore trouvé d'enseignant pour ce cours, le cours n'est pas enseigné cette année mais le sera probablement l'année prochaine, …)



C'est une association ternaire

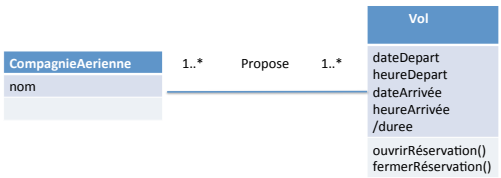
Etude de cas

Etude d'un système simplifié de réservation de vol pour une agence de voyages :

Considérer les phrases suivantes :

1. Des compagnies aériennes proposent différents vols
2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie

Modélisation statique – Etude de cas



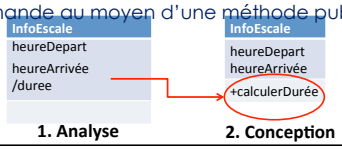
Etude de cas
Attributs dérivés

Les attributs dérivés permettent à l'analyse de ne pas faire de choix prématuré.

Plusieurs solutions :

1. Stocker un attribut normal en conception, qui aura ses méthodes (*get et set*) d'accès, comme les autres attributs, sans oublier de déclencher les mises à jour dès qu'une information dont il dépend est modifiée.

2. Ne pas stocker la valeur redondante, mais la calculer à la demande au moyen d'une méthode publique.



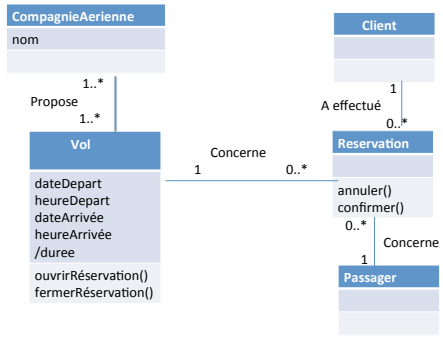
Etude de cas
Suite

3. Un client peut réserver un ou pls vols, pour des passagers différents

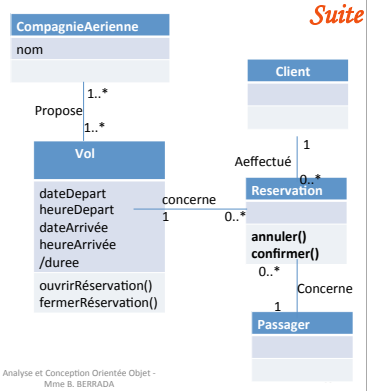
4. Une réservation concerne un seul vol et un seul passager

5. Une réservation peut être annulée ou confirmée

Etude de cas
Suite



Etude de cas
Suite

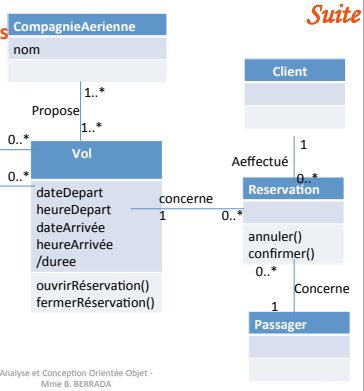


Etude de cas
Suite

3. Un vol a un aéroport d'arrivée et un aéroport de départ
4. Un vol a un jour et une heure de départ, et un jour et une heure d'arrivée

Etude de cas
Suite

Ajout de contraintes

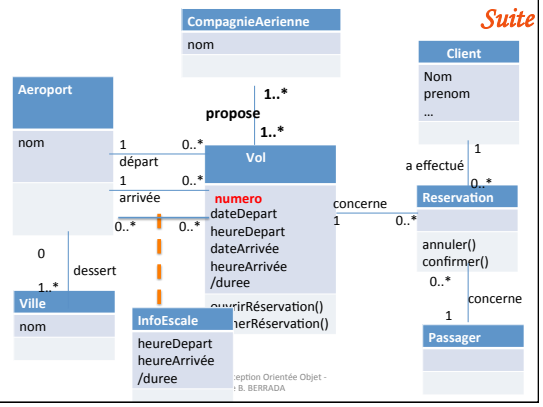


Etude de cas
Suite

Une escale a une heure d'arrivée et une heure de départ

Chaque aéroport dessert une ou plusieurs villes

Etude de cas Suite



Concept de Classe
Association qualifiée

Qualification :

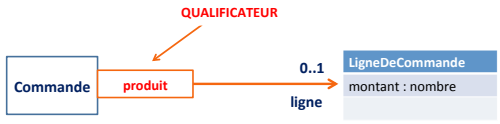
Une association qualifiée est une association dans laquelle un attribut nommé qualificateur lève l'ambiguité sur les objets situés à l'extrémité de multiplicité plusieurs.

Seules les instances possédant l'attribut indiqué dans la qualification sont concernées par l'association : une instance du couple {*Personne,compte*} est en association avec une instance unique de la classe *Banque*.

Concept de Classe
Association qualifiée

Qualification :

- Lien entre deux classes permettant de préciser la sémantique de l'association et de qualifier de manière restrictive les liens entre les instances.



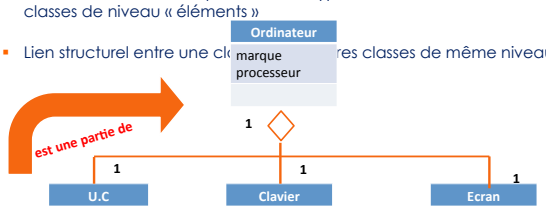
Le Qualificatif indique qu'il peut y avoir, en connexion avec une Commande, une LigneDeCommande pour chaque instance de produit.

Dans le cadre de la modélisation conceptuelle, le **Qualificatif** est utilisé pour représenter des contraintes telles qu'**une seule ligne de Commande par Produit de Commande.**

Concept de Classe
Agrégation

Association permettant de représenter un lien de type « ensemble »

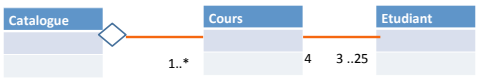
- Lien entre une classe représentant le type « ensemble » et 1 à n classes de niveau « éléments »
- Lien structurel entre une cl[illegible]res classes de même niveau.



Lien « *tout/partie* » où une classe constitue un élément plus grand (*tout, composé*) composé d'éléments plus petits (*partie ou composant*), il faut utiliser une agrégation.

Concept de Classe
Association vs Agrégation

- Si les objets des deux classes sont considérés indépendants => Association
- Si les objets des deux classes sont inséparables => Agrégation
- Exemples :



Concept de Classe
Quand utiliser une Agrégation ?

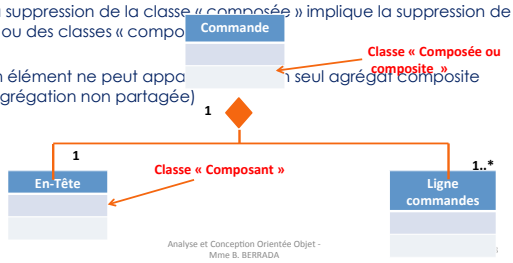
Indicateurs :

- Si l'expression « est un composant » est utilisée pour décrire un lien :
  Ex : La porte est un composant de la voiture
- S'il existe des opérations, s'appliquant sur le tout, elles s'appliquent aussi sur les composants
  Ex : La voiture se déplace, la porte aussi
- S'il existe des attributs dont les valeurs se propagent à ceux des composants
  Ex : La voiture est bleue; la porte est bleue
- Si la relation est asymétrique
  Ex : La porte est un composant de la voiture, mais la voiture n'est pas un composant de la porte.

Concept de Classe
Composition

Lien d'agrégation plus fort, dans lequel il existe une contrainte de durée de vie entre la classe « composant » et la ou les classes « composé.e.s».

- La suppression de la classe « composée » implique la suppression de la ou des classes « compo
- Un élément ne peut appa n seul agrégat composite (agrégation non partagée)



Concept de Classe
Composition

Exemple 2 : Considérons la phrase suivante :

- Un répertoire contient des fichiers. Est-ce une composition ou une agrégation ?
- Il s'agit au moins d'une agrégation. Voyons s'il s'agit d'une composition à travers les 2 critères établis précédemment :
- **1**. La multiplicité **ne doit pas être supérieure à un** du côté du composite. Est-ce le cas ?

Concept de Classe
Composition

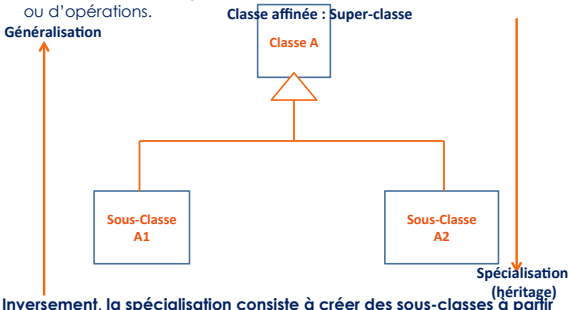
Exemple 2 : Considérons la phrase suivante :

- Un répertoire contient des fichiers. Est-ce une composition ou une agrégation ?
- Il s'agit au moins d'une agrégation. Voyons s'il s'agit d'une composition à travers les 2 critères établis précédemment :
- **2**. Le cycle de vie du composant doit dépendre de celui du composite. Est-ce le cas ?

Oui, puisque la destruction du répertoire entraîne celle de tous les fichiers qu'il contient

Concept de Classe
Généralisation et spécialisation

- La généralisation est la relation entre une classe et deux autres classes ou plus partageant un sous-ensemble commun d'attributs et/ ou d'opérations.



Inversement, la spécialisation consiste à créer des sous-classes à partir d'une classe

Concept de Classe
Généralisation et spécialisation

- La généralisation permet la création de super-classes regroupant les propriétés et les comportements communs aux sous-classes (factorisation)
- La spécialisation permet la création de sous-classes afin d'ajouter ou modifier certaines propriétés ou comportements définis dans la ou les super-classes.
- Elles permettent donc la création d'arborescences de classes d'abstraction croissantes

L'héritage est l'outil qui va nous permettre d'exprimer entre nos classes une relation de type "est une sorte de".

Concept de Classe
Généralisation et spécialisation

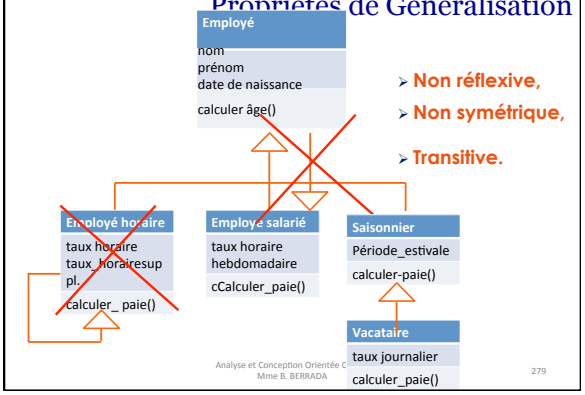
Cet outil est très puissant, car grâce à lui nous pourrons déclarer des classes d'objets très généraux, puis progressivement "spécialiser" ces classes d'objets.

Cette spécialisation est aussi une "extension" de la classe de base : si l'on reste dans les généralités, on n'a pas grand-chose à dire.

Plus les choses se précisent, plus on doit détailler.

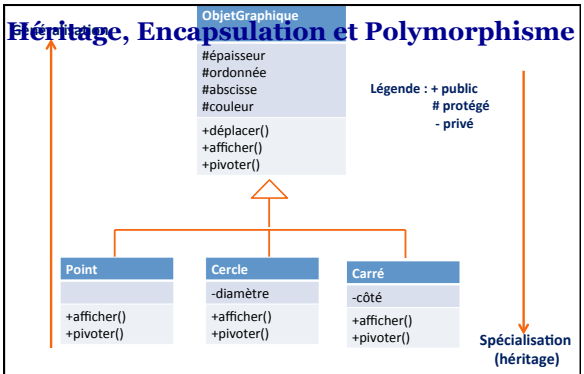
Donc à chaque étape du processus, on rajoutera du code. Le fait de partir d'une classe de base générale permet d'exprimer directement dans le code des concepts abstraits.

Concept de Classe
Propriétés de Généralisation



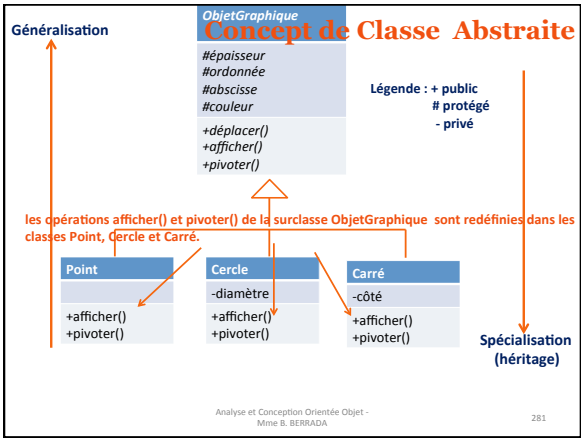
- Non réflexive,
- Non symétrique,
- Transitive.

Héritage, Encapsulation et Polymorphisme



L'héritage permet à une sous-classe de disposer des attributs et opérations de la classe dont elle dépend.

Concept de Classe Abstraite



Concept de Classe Abstraite

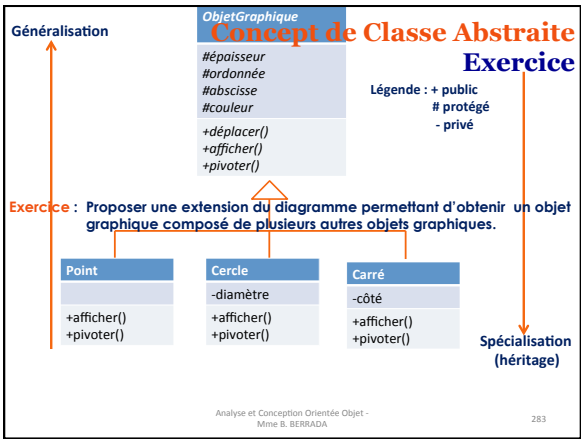
Classe abstraite est une classe qui ne peut posséder d'instances directes, et dont les classes descendantes ont des instances.

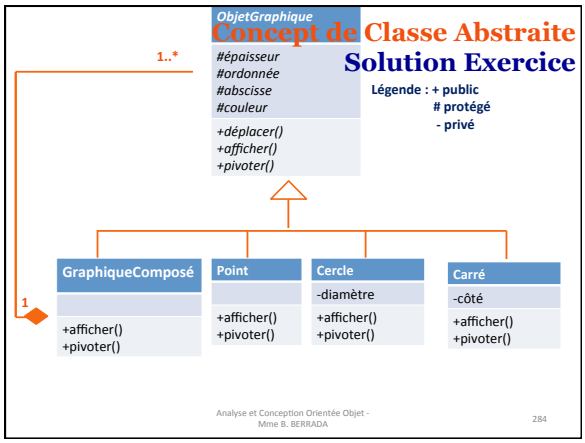
Classe abstraite s'oppose à Classe concrète.

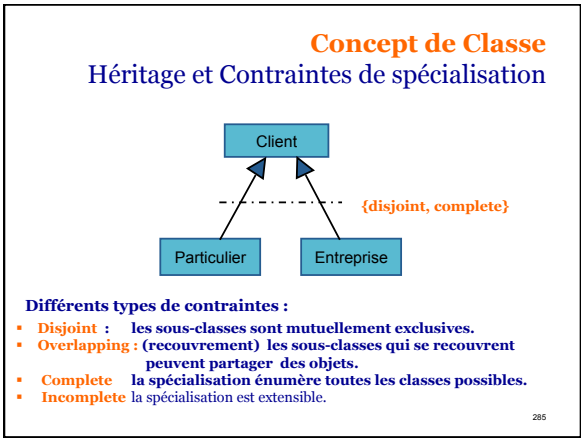
Dans une relation d'héritage, la superclasse est par définition une classe abstraite.

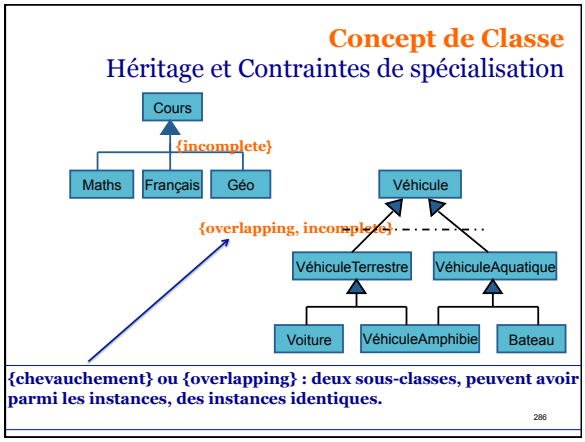
Dans les diagrammes de classe, les classes abstraites ont leur nom écrit en italique.

Dans l'exemple précédent, il est préférable d'éviter que la classe ObjetGrahique soit concrète sinon il est obligatoire de spécifier le comportement des méthodes afficher et pivoter, en donnant alors un comportement par défaut, ce qui n'est pas une solution satisfaisante.


Généralisation
ObjetGraphique
#épaisseur
#ordonnée
#abscisse
#couleur
+déplacer()
+afficher()
+pivoter()
Concept de Classe Abstraite
Exercice
Légende : + public
protégé
- privé
Exercice : Proposer une extension du diagramme permettant d'obtenir un objet graphique composé de plusieurs autres objets graphiques.
Point
+afficher()
+pivoter()
Cercle
-diamètre
+afficher()
+pivoter()
Carré
-côté
+afficher()
+pivoter()
Spécialisation
(héritage)
Analyse et Conception Orientée Objet - Mme B. BERRADA
283

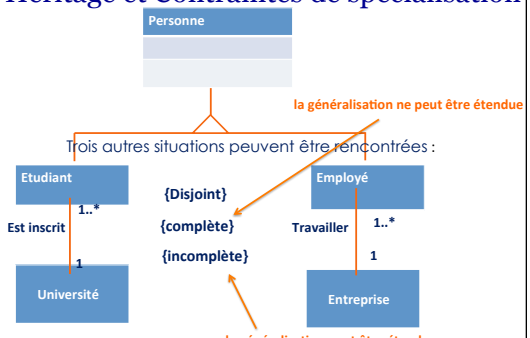

ObjetGraphique
#épaisseur
#ordonnée
#abscisse
#couleur
+déplacer()
+afficher()
+pivoter()
Concept de Classe Abstraite
Solution Exercice
Légende : + public
protégé
- privé
1..*
1
GraphiqueComposé
+afficher()
+pivoter()
Point
+afficher()
+pivoter()
Cercle
-diamètre
+afficher()
+pivoter()
Carré
-côté
+afficher()
+pivoter()
Analyse et Conception Orientée Objet - Mme B. BERRADA
284


Concept de Classe
Héritage et Contraintes de spécialisation
Client
{disjoint, complete}
Particulier
Entreprise
Différents types de contraintes :
Disjoint : les sous-classes sont mutuellement exclusives.
Overlapping : (recouvrement) les sous-classes qui se recouvrent peuvent partager des objets.
Complete la spécialisation énumère toutes les classes possibles.
Incomplete la spécialisation est extensible.
285


Concept de Classe
Héritage et Contraintes de spécialisation
Cours
{incomplete}
Maths
Français
Géo
Véhicule
{overlapping, incomplete}
VéhiculeTerrestre
VéhiculeAquatique
Voiture
VéhiculeAmphibie
Bateau
{chevauchement} ou {overlapping} : deux sous-classes, peuvent avoir parmi les instances, des instances identiques.
286

Classe

Héritage et Contraintes de spécialisation

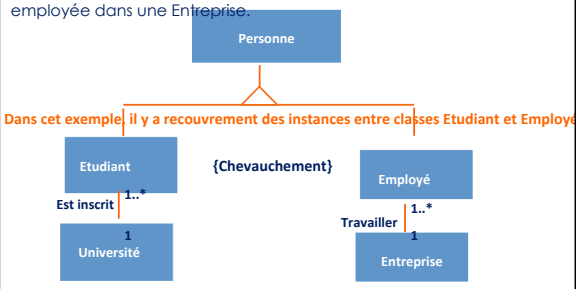


Concept de Classe

Héritage et Contraintes de spécialisation

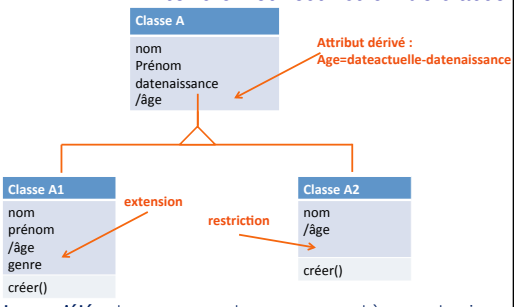
Autre exemple

Une même personne peut être à la fois Etudiant dans une université et employée dans une Entreprise.



Classe

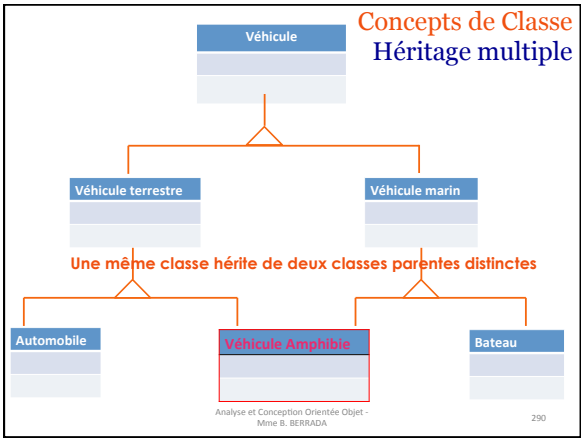
Extension et restriction de classe



L'ajout de propriété.s dans une sous-classe correspond à une extension.
Le masquage de propriété dans une sous-classe correspond à une restriction.

Concepts de Classe

Héritage multiple



Concept de Patterns

- La Classe Vol semble surchargée, et a deux types responsabilités différentes :
  - Première responsabilité : respect de l'agenda régulier d'une Compagnie Aérienne.
    Exemple : Vol tous les lundis matin
  - Deuxième responsabilité : Vol tous les lundis, mais nécessitant une réservation à une date et heure précises.
- La surcharge de responsabilités pour une classe enfreint le principe de la conception orientée objet : forte cohésion.

Concept de Patterns

- Première responsabilité : respect de l'agenda régulier d'une Compagnie Aérienne.
  Exemple : Vol tous les lundis matin

  Solution 1 : Création d'une classe générique ***VolGénérique***
- Deuxième responsabilité : Vol tous les lundis, mais nécessitant une réservation à une date et heure précises.

  Solution 2 : Restriction de la classe Vol aux propriétés du type2

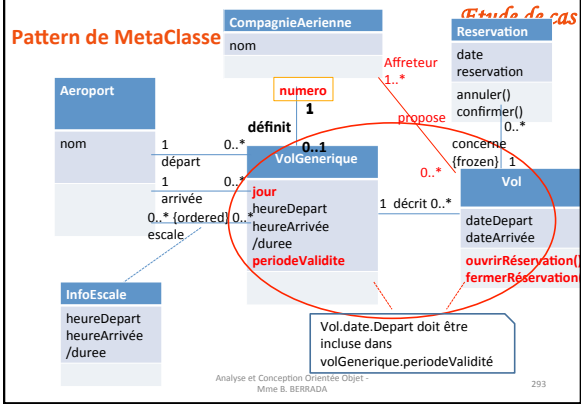
  Comment ?

  Il suffit de répartir les attributs, les opérations et les associations de l'ancienne classe Vol entre les deux classes VolGenerique et Vol.

Concept de Patterns

Pattern de MetaClasse

Etude de cas



Concept de Patterns

La séparation des responsabilités opérée peut être généralisée sous forme d'un « pattern d'analyse » et réutilisable dans des contextes différents.

On identifie une classe ***ABC*** qui possède plusieurs responsabilités, dont certaines ne sont pas propres à chaque instance.

On ajoute une classe ***TypeABC,*** *on répartit les propriétés sur les deux classes et on les relie par une association « *-1 »*

La classe TypeABC est qualifiée de « métaclasse », comme la classe VolGenerique, car elle contient des informations qui décrivent la classe ABC, à savoir Vol

Concept de Patterns

Ce pattern peut être reproduit dans l'exemple de Livres de bibliothèque, où :

- *la classe Livre serait la classe TypeXX avec des attributs tels que num_ISBN, auteur, dateEdition, dateParution, …*
- *la classe ExemplaireLivre serait la classe XX, avec comme attributs état (neuf, récupéré, abîmé)*
- *une association serait « emprunté par » liée à Adhérent, …*

Structuration en packages

La structuration s'appuie sur deux principes fondamentaux :

1. **cohérence** : *Regrouper les classes proches d'un point de vue, respectant les* **3** critères suivants :

   1.1 Finalité : les classes doivent rendre un service de même nature aux utilisateurs,

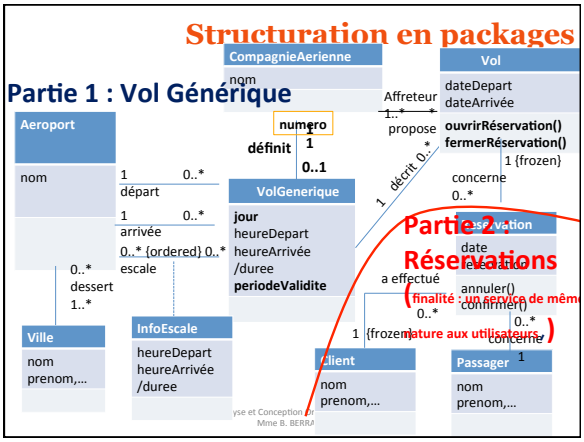
   1.2 Evolution : isoler les classes réellement stables de celles qui vont évoluer au cours du projet (classes métier et applicatives)

   1.3 Cycle de vie des objets : *Distinguer les classes dont les objets ont des durées de vie très différentes.*

Structuration en packages

2. **Indépendance** : *Renforcer ce découpage initial en s'efforçant de minimiser le dépendances entre packages.*

Structuration en packages



Structuration en packages

1. **Chaque package contient bien un ensemble de classes fortement reliées et les deux packages sont presque indépendants.**

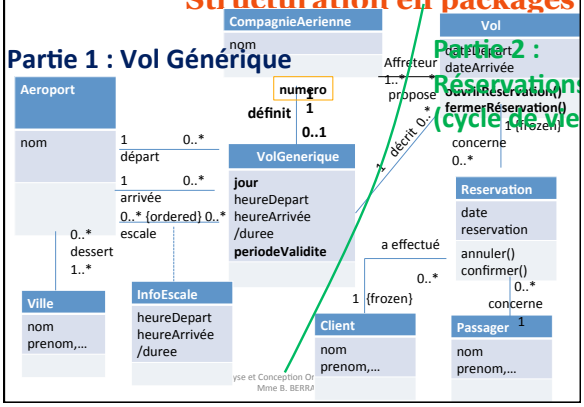
2. Il y a toute fois, une deuxième solution qui consiste à positionner la classe Vol dans même package que la classe Réservation.

   Le critère privilégié est la vie des objets, les vols se rapprochant plus des réservations que des vols génériques.

Structuration en packages

Partie 1 : Vol Générique

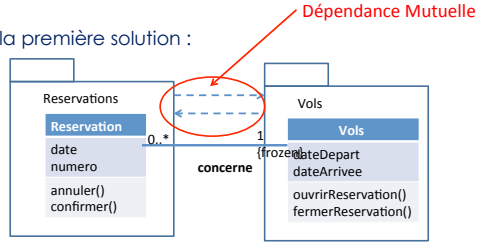
Partie 2 : Réservations (cycle de vie



Structuration en packages

1. **Réduction du couplage**

Dans la première solution :

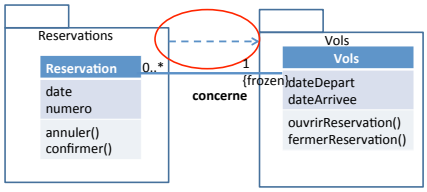


Une association entre deux Classes A et B permet par défaut de naviguer dans les deux sens entre des objets de la Classe A et la Classe B

Structuration en packages

1. **Réduction du couplage**

Dans notre exemple, nous essayons de réduire cette dépendance, en privilégiant un sens de navigation afin de d éliminer une des deux dépendances.

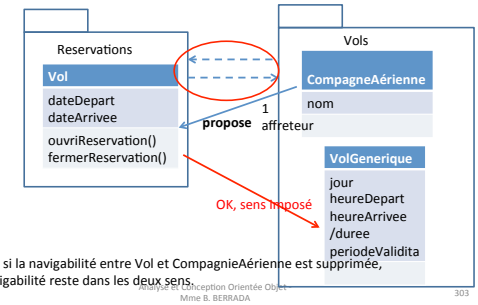


Une réservation ne va pas sans connaissance du vol concerné, mais un vol peut exister par lui-même, indépendamment de toute réservation

Structuration en packages

1. Réduction du couplage

Qu'en est-il de la deuxième solution ?

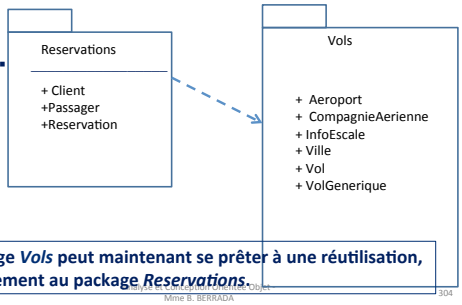


Même si la navigabilité entre Vol et CompagnieAérienne est supprimée, la navigabilité reste dans les deux sens.

Structuration en packages

1. Réduction du couplage

La balance penche vers la première Solution.



Le package *Vols* peut maintenant se prêter à une réutilisation, Contrairement au package *Reservations*.