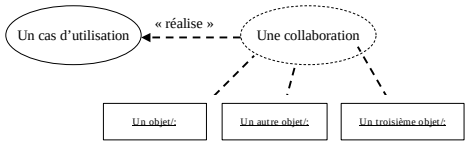
Les diagrammes UML (suite)

- Les vues statiques
  - *Diagrammes de classes* : classes et relations
  - ***Diagrammes d'objets* : liens et objets, diag. de collaborations simples**
  - *Diagrammes de cas d'utilisation* : fct du système du point de vue de l'utilisateur
  - *Diagrammes de composants* : composants physiques d'une application
  - *Diagrammes de déploiement* : déploiement des composants sur les matériels
- Les vues dynamiques
  - ***Diagrammes de séquences* : représentation temporelle des objets et des interactions**
  - ***Diagrammes de collaborations* : représentation spatiale des objets, des liens et des interactions**
  - *Diagrammes d'états-transitions* : comportement d'une classe ou d'une méthode en terme d'états
  - *Diagrammes d'activités* : comportement d'une méthode, d'un cas d'utilisatio ou d'un processus métier

Les diagrammes d'utilisation

fonctionnement du système du point de vue de l'utilisateur

La transition vers l'objet (suite)

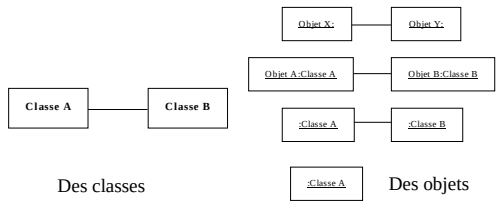


- Dans une approche objet un cas d'utilisation est réalisé au moyen d'une collaboration entre objets
- Les scénarios sont des instances du cas d'utilisation
- Ils sont représentés par :
  - des diagrammes de séquences
  - Des diagrammes de collaborations

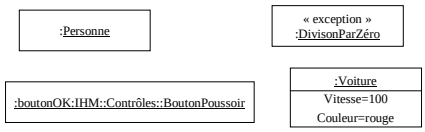
Les diagrammes d'objets

liens et objets, diagramme de collaborations simples

- Encore appelés diagrammes d'instances
- Une vue statique comme les diagrammes de classe



Des objets

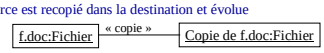


- L'état d'un objet est déterminé par les valeurs des attributs
- A un instant donné, l'objet est dans un état
- Celui-ci est la conséquence des opérations de modification

Etats d'un objet et stéréotypes des liens

- Représentation de l'état d'un objet (plusieurs : séparés par des virgules) :

- UML défini 2 stéréotypes pour les liens :
  - « copie » : l'objet source est recopié dans la destination et évolue indépendamment

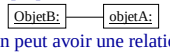


  - « devient » : pour modéliser un objet dans différents états

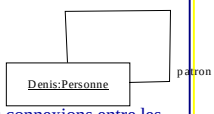
  - « local » : placé à l'extrémité d'un lien, indique que l'instance est locale
  - « global » : instance visible par tous
  - « paramètre » : paramètre d'une méthode
  - « self » : lien vers l'objet lui-même

Représentation des liens

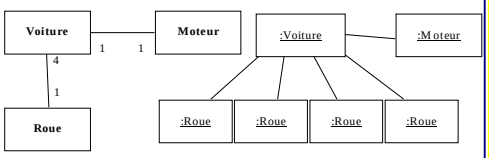
- La plupart des liens sont binaires :



- On peut avoir une relation réflexive :

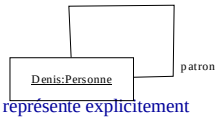


- Les liens entre objets représentent les connexions entre les instances des classes :



Similitude entre diagrammes d'objets et diagrammes de classes

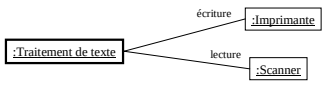
- On a vu les liens simples
- Toutes les décorations des classes peuvent être reportées sur les liens entre objets :
  - Le nom (souligné : objetA: ou :classeA ou objetA:classeA)
  - le nom du rôle
  - L'aggrégation
  - La composition
  - La naviguation



- L'exception : la multiplicité se représente explicitement par les liens

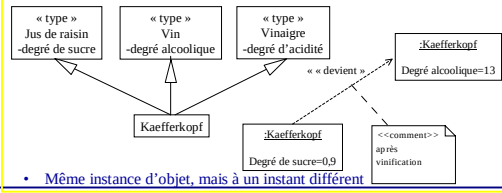
Les objets actifs

- Par défaut les classes sont passives donc par défaut les objets sont passifs
- Les objets instances de classes actives (qui ont leur propre flot d'exécution) sont dits des objets actifs
- Ils possèdent un flot de contrôle
- Un objet actif peut activer un objet passif, le temps d'une opération, le contrôle est à nouveau restitué à la fin
- Un objet actif est représenté par un cadre en gras et à la propriété {actif} peut être rajouté dans le symbole de l'objet



La classification dynamique

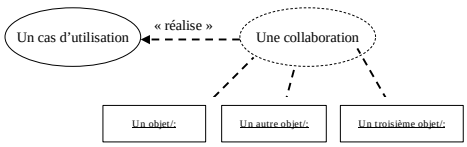
- Un concept qui permet de changer le type d'un objet de manière dynamique : on change son rôle au cours du temps
- Pour changer le type d'un objet, la classe dont il est l'instance doit définir différents types (spécification du comportement et de la structure) ou différents interfaces (spécification du comportement uniquement)



- Même instance d'objet, mais à un instant différent

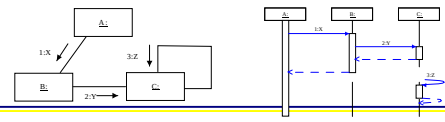
Les diagrammes de collaboration

- Les diagrammes d'objets : des collaborations simples



Les diagrammes de collaboration (suite)

- Aussi appelés diagrammes d'interactions avec les diagrammes de séquence
- Représente une vue dynamique du système
  - Présente un ensemble de rôles joués par des objets dans un contexte particulier
  - Ces objets sont reliés par liens
  - Sur ces liens circulent des messages
- Ces diagrammes insistent sur la structure spatiale qui permet la collaboration entre objets
- La dimension temporel : les messages sont numérotés



Une collaboration

- Une collaboration définit les éléments qui sont utiles pour atteindre un objectif particulier
  - On spécifie le rôle de l'élément dans le contexte de la collaboration
- Une collaboration est la *réalisation* d'une opération ou d'un classificateur (classe ou cas d'utilisation) dans un cas donné
- Une collaboration possède 2 types :
  - Une description générale *au niveau spécification* qui représente:
    - les rôles des classes et des associations
    - les interactions sous forme de messages
  - Une description spécifique *au niveau instance* qui représente :
    - Une instance particulière d'une interaction avec les objets et les liens qui se conforment aux rôles définis au niveau de la spécification. Plus les messages

Pourquoi deux types description?

- Pleinement décrire le comprotement
- Selon le niveau de détails : décrire des spécification et exprimer des réalisation
- Résumé des éléments de modélisation

Spécification	Classe	Opération	Cas d'utilisation
Réalisation	Instance	Méthode	Réalisation de cas d'utilisation

- Une collaboration peut-être décrite par une autre collaboration à un niveau de granularité plus faible
- Un même élément peut intervenir dans plusieurs collaboration

Les Rôles

- Dans le contexte d'une collaboration les classes et les associations jouent des rôles
  - Représenté par un symbole de classe
  - Syntaxe : **/nom_du_rôle:classes**
- Le nom du rôle peut-être omis s'il n'existe qu'un seul rôle
- Les instances de classificateurs sont des objets qui se conforment à ce rôle
  - Syntaxe : **nom_de_l'objet/nom_du_rôle:classes**

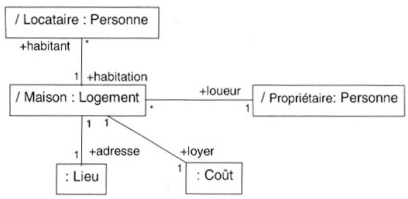
Exemples de rôles

- Un rôle anonyme de la classe C — :C
- Un rôle R de la classe C — /R:C
- Un rôle R — /R
- Objet anonyme, instance de la classe C — :C
- Objet anonyme, instance de la classe C, jouant le rôle R — /R:C
- Objet anonyme, jouant le rôle R — /R
- Objet O, instance de la classe C, jouant le rôle R — O/R:C

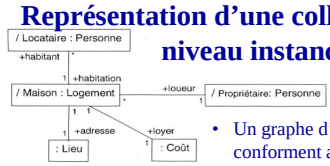
Objecteering inverse Objet et Rôle

Représentation d'un collaboration au niveau spécification

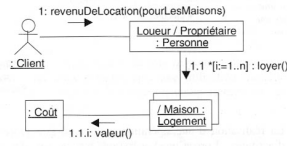
Un graphe de rôles pour les classes et les associations



Représentation d'une collaboration au niveau instance

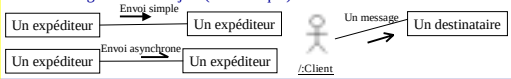


- Un graphe d'instances qui se conforment aux rôles définis au niveau de la spécification



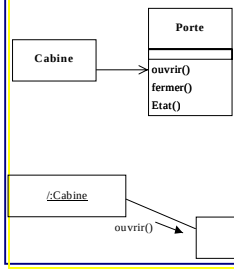
Les envoies de messages

- On parle de *stimulus* pour parler d'un message entre 2 objets
  - Déclenche une opération : émission d'un signal, la création ou la destruction d'un objet
  - Le message est la spécification du *stimulus* : il définit le rôle des objets émetteur et récepteur, ainsi que l'action qui envoie le stimulus
  - La plupart du temps on ne fait pas la distinction entre les deux
- Un message matérialise une communication au cours de laquelle se transmettent des informations (éventuellement aussi des informations retour)
- Plusieurs messages peuvent figurer sur le même lien
- Les diagrammes de collaboration (lien dynamique) sont des extension des diagrammes d'objets (lien statique)



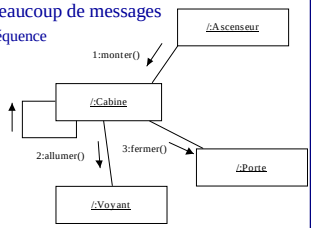
Les interactions au sein d'une collaboration

- Une interaction définit une communication entre instances
- Messages partiellement ordonnés
- Instances interagissent pour réaliser un certain comportement
- Seuls les objets nécessaires sont représentés
- On trouve les éléments suivants :
  - Les instances
  - Les liens
  - Les messages
  - Les rôles
- Deux points de vue pour exprimer des interactions :
  - Le temps (séquence)
  - L'espace (collaboration)



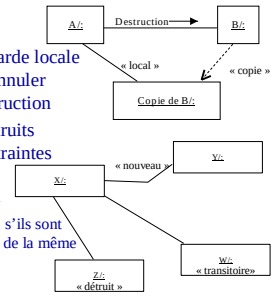
Les interactions (suite)

- Le temps n'est pas représenté de manière implicite
- Rajoute des messages numérotés
- Difficile à lire avec beaucoup de messages
  - Les diagrammes de séquence
- Exemple : ordre d'envoie des messages



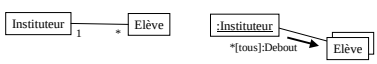
Les interactions (suite)

- A effectue une sauvegarde locale de B afin de pouvoir annuler éventuellement la destruction
- Les objets créés ou détruits peuvent porter les contraintes suivantes :
  - {nouveau} ou {détruit}
  - Ou encore {transitoire} s'ils sont créés et détruits au sein de la même interaction



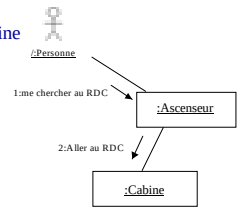
Représentation condensée d'une interaction

- UML permet de représenter de manière condensée une famille de liens, instances d'une même association, entre un objet et un groupe d'objet, instances d'une même classe
- Le groupe d'objets est traité de manière uniforme, tous sont destinataire d'un même message
- L'itération est symbolisée par le caractère *



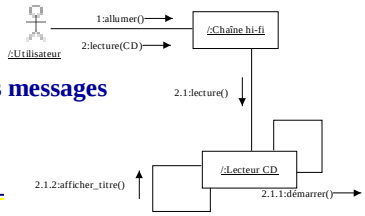
La place de l'utilisateur

- Il peut figurer dans le diagramme de collaboration
- Pour représenter une interaction avec un élément extérieur au système
- Ex : l'appel d'une cabine



Exemple avec Objecteering

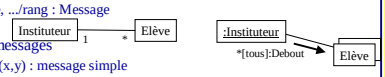
- Un message est représenté par une flèche
- La séquence de message commence en général par 1
- Les numéros des séquences suivantes sont incrémentés (2, 3, 4, ..)
- Un message déclenchant d'autres messages, la séquence peut-être numérotée de la manière suivante :
  - 2.1, 2.1.1, 2.2, …
- Les messages concurrents
  - 2.1a, 2.1b,..



Syntaxe des messages

Messages itératifs ou conditionnels

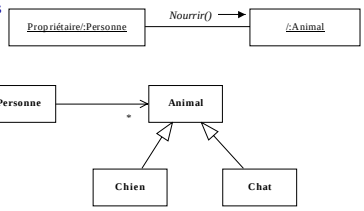
- Message itératif :
  - Envoi séquentiel des messages : *[i=1..n]:Message
  - Envoi parallèle ou diffusion : *| | [i=1..n]:Message
- Message conditionnel
  - [X>Y]:Message
- Message synchronisé avec d'autre flots (rang : 1, 1.2, 2.2a, 2.2b)
  - Rang, range, .../rang : Message
- Exemples de messages



  - 4 : Afficher(x,y) : message simple
  - 3.3.1 : Afficher(x,y) : message imbriqué
  - 4.2 :age = soustraire(aujourd'hui, dateDeNaissance) : valeur retournée
  - [age>18] 6.2 : Voter() : message conditionnel
  - 4.a, b.6/c.1 : Allumer(Lampe) : synchronisation avec d'autres flots
  - 1 * : Laver() : itération
  - 3.a, 3.b / 4*| | [i=1..n] : Eteindre() : itération parallèle

Les classes abstraites et les collaborations

- Les classes abstraites et interfaces bien que non instanciables peuvent figurer dans des diagrammes de collaboration
- Mettre en relief le lien polymorphe entre la classe abstraite et ses sous-classes concrètes

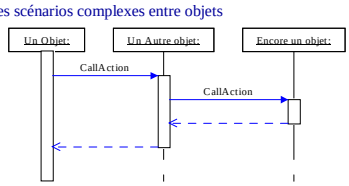


Les diagrammes de séquences

Représentation temporelle des objets et des interactions

Les diagrammes de Séquence

- Se concentre sur la séquence des interactions dans le temps
- Chaque objet à une ligne de vie
- Plus apte à modéliser les aspect dynamiques :
  - Des systèmes temps réels
  - Des scénarios complexes entre objets

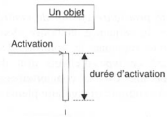


Les diagrammes de Séquence(suite)

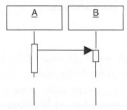
- Utilisation différente selon la phase du cycle de vie et le niveau de détails désiré :
  1. Documentation des cas d'utilisation : description des interactions, proche de l'utilisateur. Les fléches n'ont pas un sens au niveau programmation
  2. Usage plus informatique : représentation précis des interactions entre objets et la répartition du flot de contrôle

Les activations et envois de messages

- Activité représentée par les bandes rectangulaires : lui ou par l'intermédiaire d'un autre objet

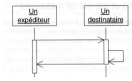


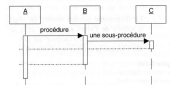
- A active B, l'activité de l'objet B recouvre celle de A
- L'objet A est bloqué jusqu'à ce que B lui redonne la main

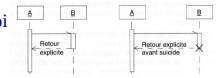


Trois grandes catégories d'envois

- Flot de contrôle à plat : message asynchrone, une flèche simple ou demi flèche symbolise cette catégorie de messages
- Appel de procédure ou flot de contrôle emboîté : la séquence emboîtée doit se terminer d'abord
- Retour d'un appel de procédure : le retour est implicite à la fin d'une activation. En cas d'envoi asynchrone le retour est symbolisé explictement

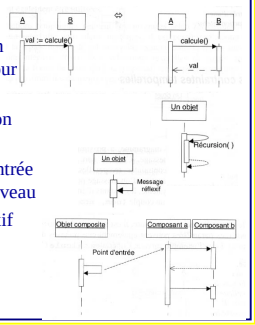






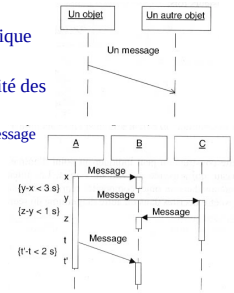
Retour des messages

- Il est possible de représenter un ou plusieurs paramètres de retour
- Le cas particulier de la récursion
- un message réflexif : point d'entrée pour une activité de plus bas niveau
- Utilisation d'un message réflexif comment entrée permanente



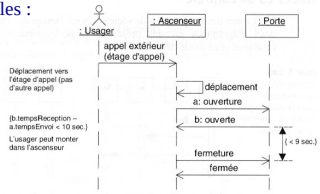
Les contraintes de temps

- Indiqué par la flèche en oblique
- Dans la marge ou à proximité des éléments concernés
  - L'instant d'émission d'un message peut-être indiqué
  - Si délai on utilise la notation **(nom, nom')**



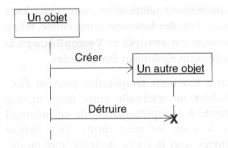
Les contraintes de temps (suite)

- Des annotation temporelles sous la forme :
  - Temps : message
- Des fonction temporelles :
  - TempsEnvoi
  - TempsRéception
  - D'autres fonctions peuvent être inventées
- Contrainte de temps sur interval
  - {<9sec.}

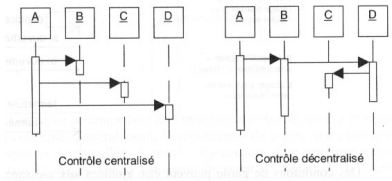


La ligne de vie des objets

- Une ligne verticale
- Précise l'existence
- La plupart du temps sur toute la hauteur
- Peut-être créer ou détruit



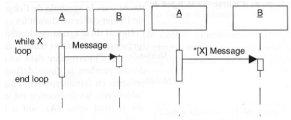
Forme des diagrammes



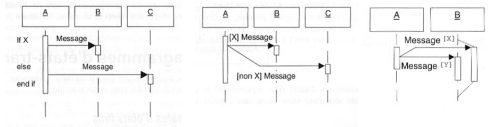
- Mode de contrôle centralisé : A active successivement les autres objets
- Mode de contrôle décentralisé : un objet ne domine pas

Structure de contrôle

- Une boucle **while** :



- Un branchement conditionnel **if then else**



Exemple de diagramme de séquence