Conception de Systèmes d'Information

Michel.Bigand@ec-lille.fr

Avertissement

- Certaines diapos sont issues d'autres cours (JP Bourey, D Corbeel, C Verzat…)

- Participation requise à toutes les séances
  - Absence prévisible : prévenir par mail préalablement
  - Absence non prévisible : prévenir par mail dès que possible
  - Dans tous les cas, fournir un travail de rattrapage

- Notation : à partir des travaux rendus
  - Par mail à Michel.Bigand@ec-lille.fr
  - Objet : *Nom formation* – séance 1
  - Document Word attaché (non zippé)
  - En fin de séance, ou avant la date convenue

Objectifs du cours

- Apprendre à utiliser des méthodes systématiques pour l'analyse des besoins
- Créer des modèles UML d'analyse couvrant 3 aspects :
  - Fonctionnel
  - Statique
  - Dynamique
- Connaître de nouveaux formalismes de modélisation
- Découvrir les principaux concepts des approches orientées objet

Plan d'une séance

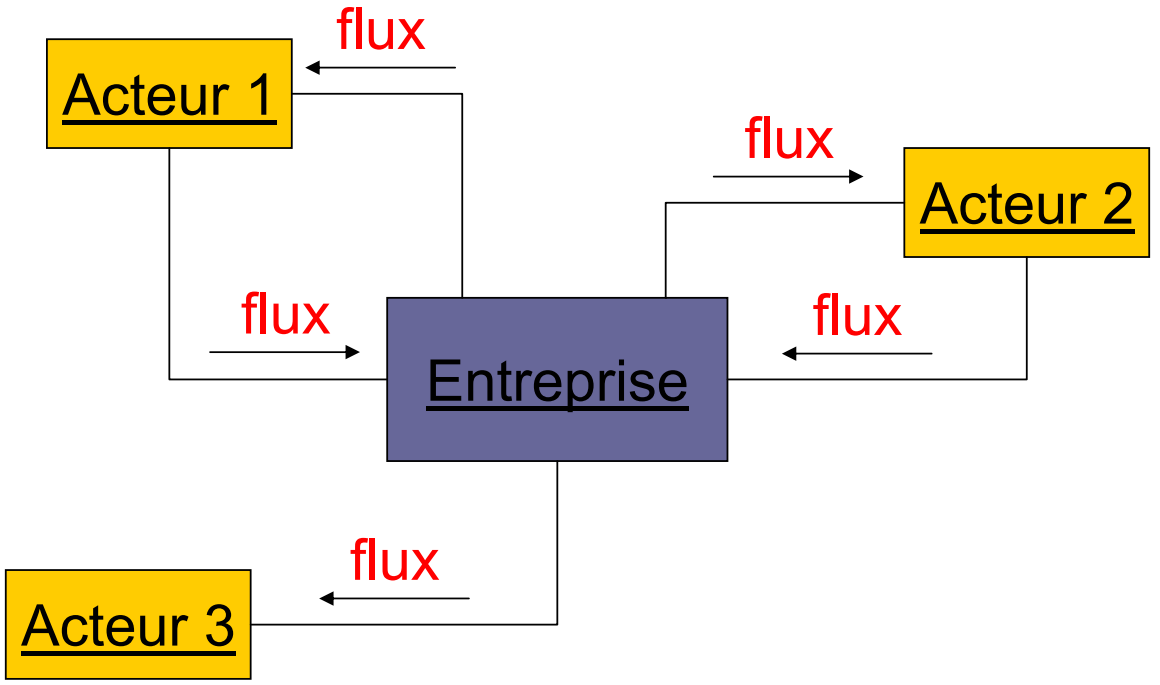
- Partie de cours
- Étude de cas guidée
- Travail en binôme à rendre
  - C'est en forgeant qu'on devient forgeron
  - Un problème peut avoir plusieurs solutions

Plan du cours

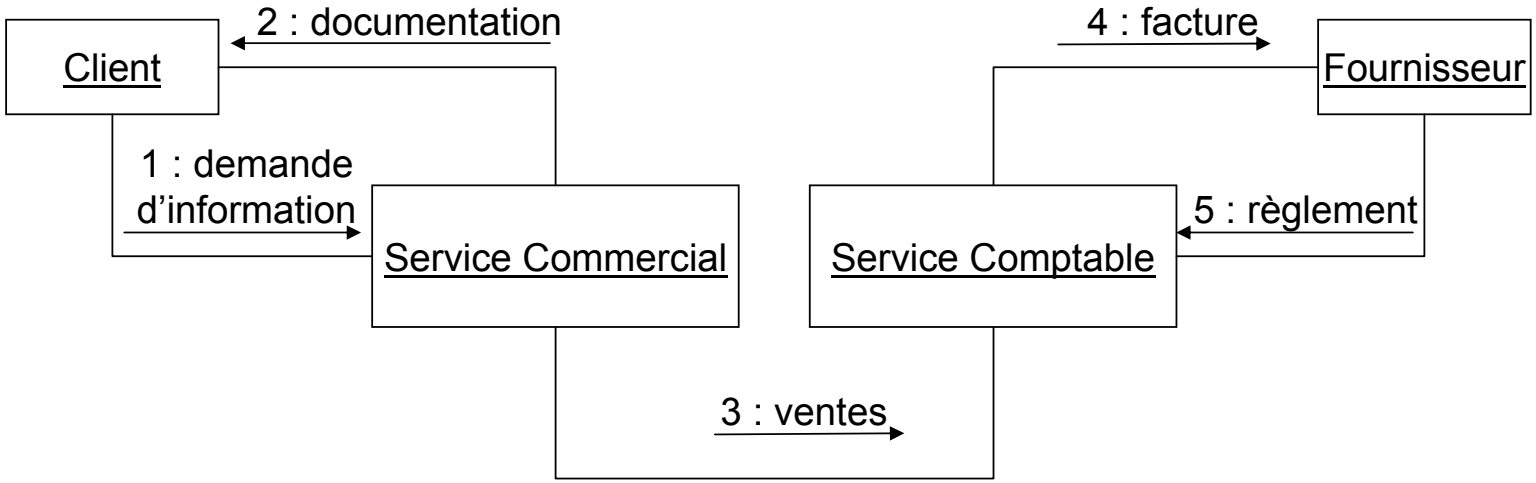
- Introduction
- Point de vue fonctionnel
- Point de vue statique
- Point de vue dynamique
- Étude de cas
  - Bien que la présentation semble linéaire, l'analyse et la conception abordent généralement les 3 points de vue en parallèle

- L'entreprise et son système d'information
- Problème actuels du génie logiciel
- Analyse vs conception
- Documentation et CASE
- Cycles de vie
- Modèle
- Notions sur les approches orientées objet
- Introduction à UML

L'entreprise et son environnement



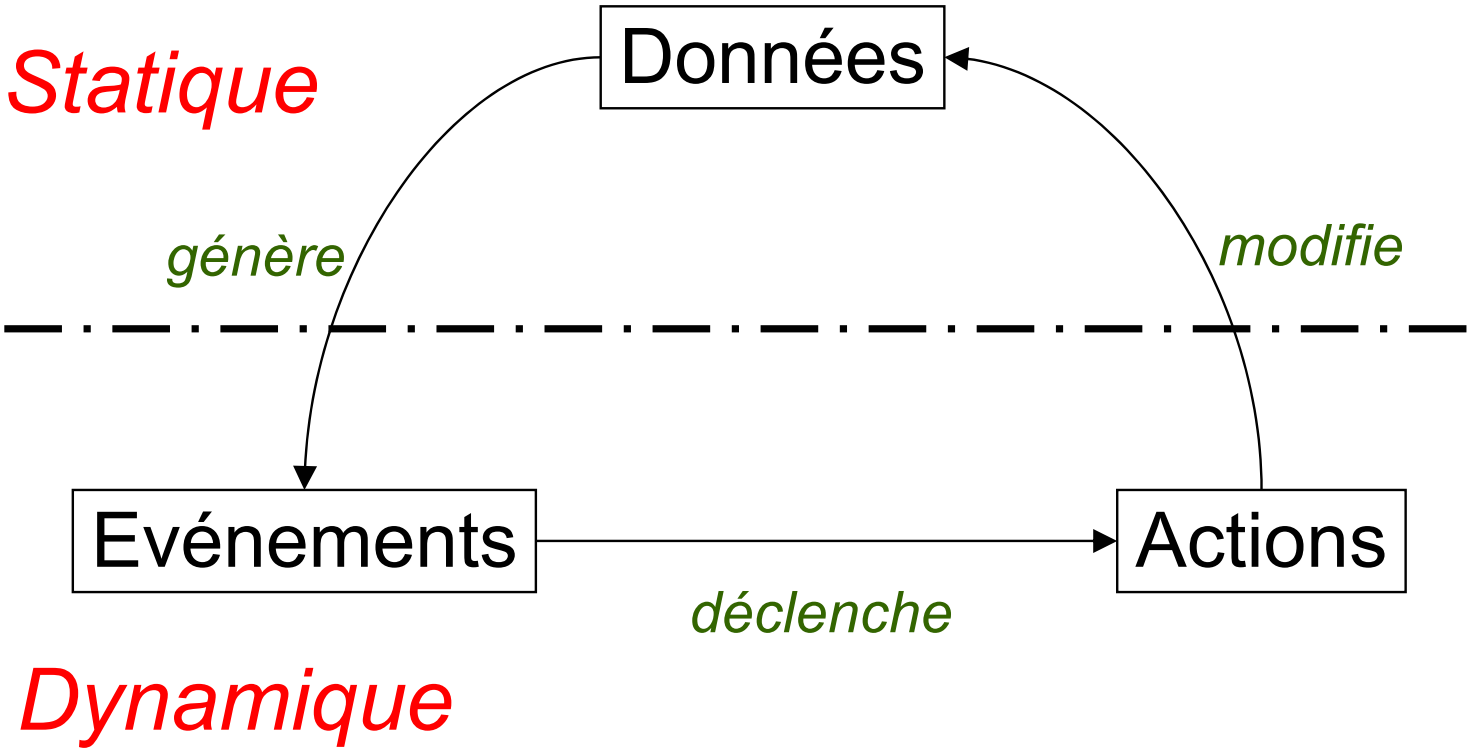
Les domaines de l'entreprise



Degrés d'invariance dans l'entreprise

- Ce qui est stable
  - processus liant l'entreprise à ses acteurs externes
  - données
- Ce qui est moins stable
  - traitements
- Ce qui est peu stable
  - techniques
  - nature de la demande
  - besoins en statistiques
  - organisation de l'entreprise

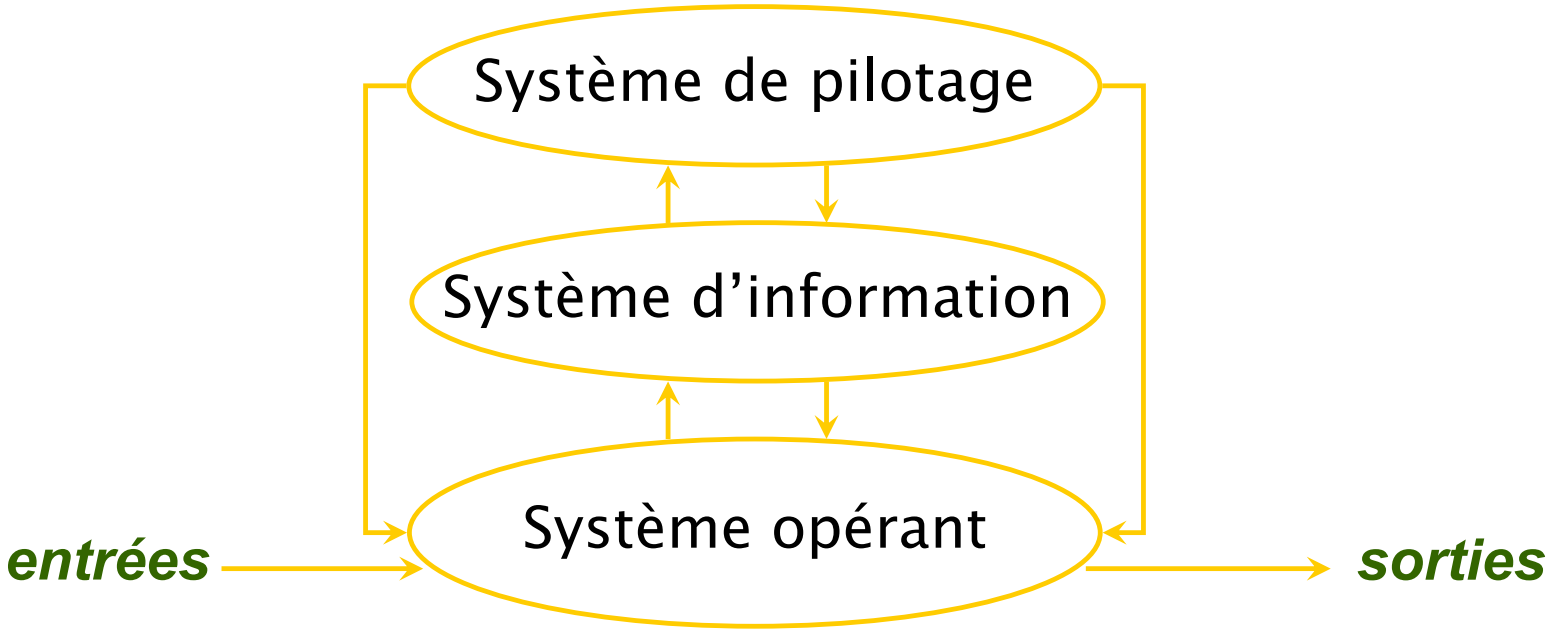
Statique et dynamique du système d 'information



A quoi sert l'information ?

- Support pour l'action
- Conserve une trace des activités
- Apporte une aide à la décision
- Technologie de l'information et de la communication
  - Mémorisation
  - traitement automatique
  - Diffusion

Le système d'information dans l'entreprise



Le schéma directeur

- Définit les orientations stratégiques de l'entreprise en matière de système d'information :
  - Politique d'investissement matériel et logiciel
  - Choix d'organisation des systèmes d'information (centralisation/répartition)
  - Rôle des différents acteurs de la politique informatique (utilisateurs, techniciens, exploitants)

- L'ensemble des actions ayant trait au traitement de l'information dans l'entreprise devra être cohérent avec le schéma directeur

- L'entreprise et son système d'information
- Problème actuels du génie logiciel
- Analyse vs conception
- Documentation et CASE
- Cycles de vie
- Modèle
- Notions sur les approches orientées objet
- Introduction à UML

Problèmes actuels du génie logiciel

- Taille et complexité du logiciel
  - Complexité fonctionnelle
  - Mutations technologiques perpétuelles
  - Complexité des architectures
  - Solutions :
    - Distinguer analyse et réalisation
    - Décomposer le système
    - Utiliser une approche de haut niveau

Problèmes actuels du génie logiciel

- Taille croissante des équipes
  - Compétences de + en + variées et pointues
  - Applications stratégiques orientées métier
  - Délais de + en + courts
  - Solutions :
    - Technologie unifiant le vocabulaire
    - Méthode, démarche de travail

Problèmes actuels du génie logiciel

- Évolution rapide des applications
  - Besoins du client
  - Activité du client
  - Environnement technique
  - Solution :
    - Cycle de vie itératif et incrémental

Problèmes actuels du génie logiciel

- Spécifications peu précises
  - Imprécision, incomplétude
  - Interface difficile entre domaine métier et informatique
  - Solution :
    - Utilisation de modèles, notamment graphiques

Exercice : l'argent de la caisse

Le dernier client venait de quitter le magasin. L'un des propriétaires ramassait le contenu d'une caisse enregistreuse quand un homme rentra. L'inconnu alla droit vers le gérant et lui demanda de l'argent.

La lumière s'éteignit brusquement. Quand elle revint, l'inconnu avait disparu. Toutes les caisses enregistreuses étaient vides.

L'inspecteur Lapreuve arriva immédiatement sur les lieux.

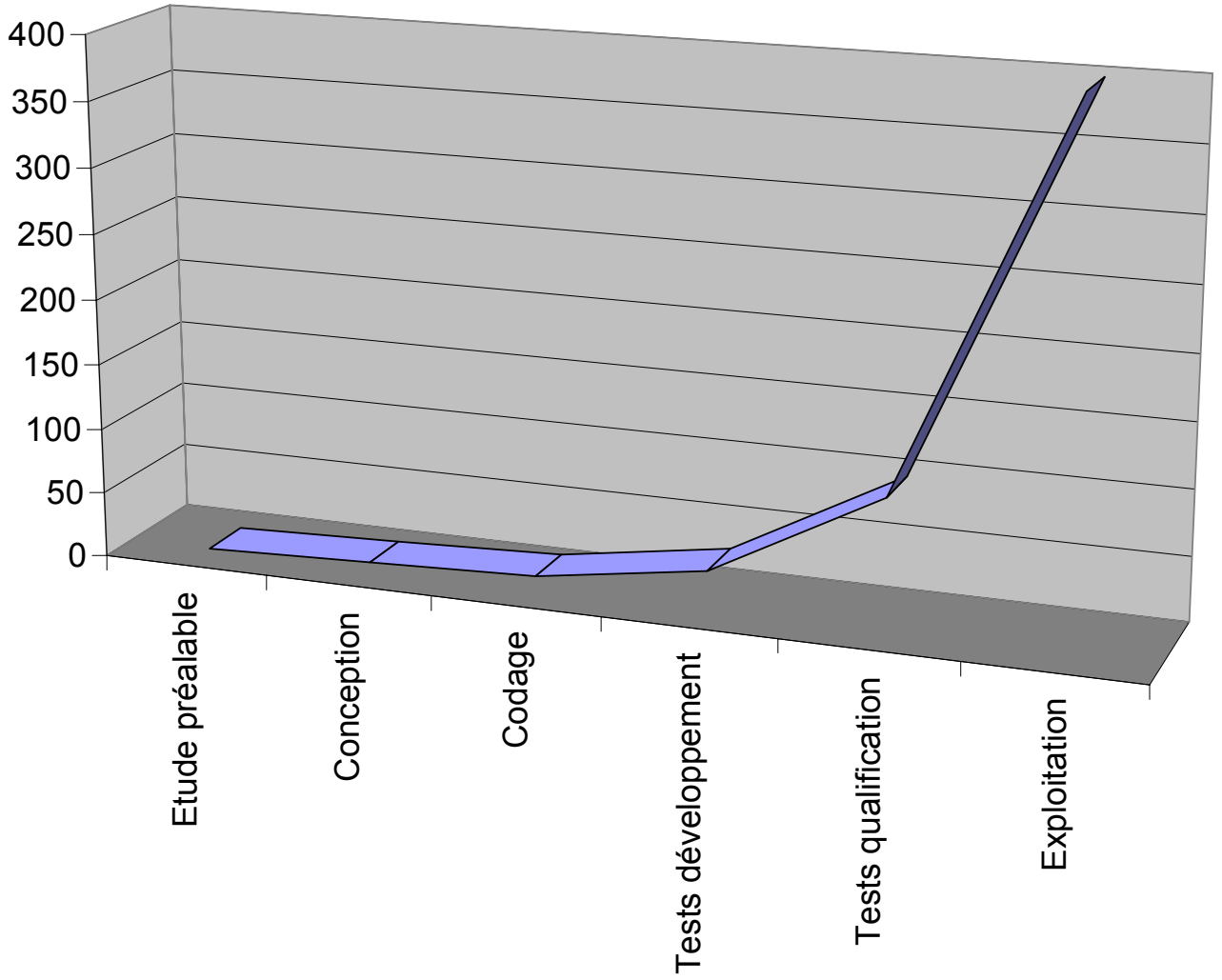
Exercice : l'argent de la caisse cocher en 2mn les cases appropriées

	oui	non	?
1. L'inconnu s'adressa au gérant			
2. Le voleur ne demanda pas d'argent			
3. Il n'y a qu'un propriétaire			
4. Le voleur a coupé l'électricité			
5. L'histoire ne précise pas combien d'argent a disparu			
6. Seules 2 personnes étaient présentes quand l'homme entra			
7. Le voleur voulait de l'argent			
8. Le gérant ramassait le contenu des caisses enregistreuses			
9. Le propriétaire a reconnu l'inconnu			
10. L'inspecteur Lapreuve recherchait l'inconnu			

Défauts du langage humain

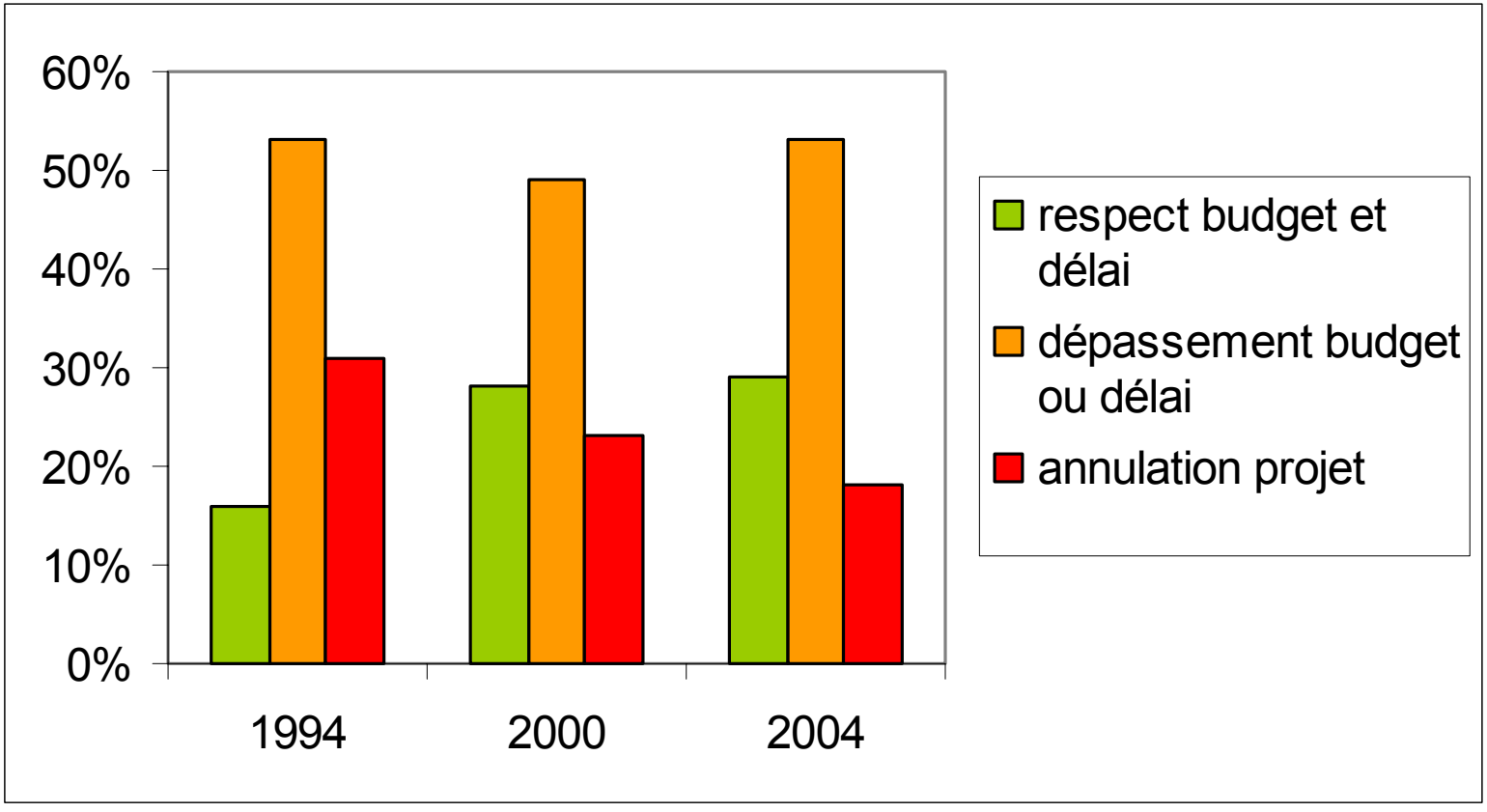
- Bruit
- Silence
- Sur-spécification
- Contradiction
- Ambiguïté
- Référence en avant
- Vœu pieu

Impact d'une erreur de spécification



- L'entreprise et son système d'information
- Problème actuels du génie logiciel
- Analyse vs conception
- Documentation et CASE
- Cycles de vie
- Modèle
- Notions sur les approches orientées objet
- Introduction à UML

Une amélioration lente



Source : Standish group, 2006

Nécessité de l'analyse et de la conception

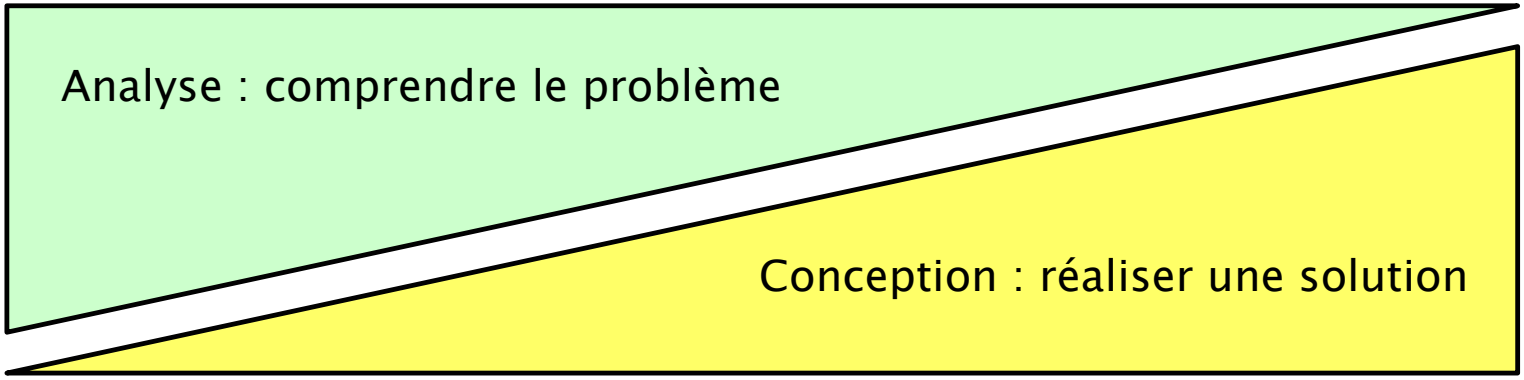
- Exemple de problème

- Proposer quelques solutions

Poser les bonnes questions

- QUEL est le problème ?
- POURQUOI le problème existe-t-il ?
- QUI est impliqué ?
- OU se situe le problème ?
- QUAND faut-il mettre en œuvre la solution ?
- QUELLE technologie est implicitement pressentie ?

Analyse vs conception



Domaine du problème

Domaine de la solution

- La conception est souvent un compromis

Analyse vs conception

- L'analyse prend en compte l'environnement du monde réel
  - QUOI
  - Conceptuel
  - Peut couvrir des besoins de l'utilisateur hors logiciel
- La conception consiste à trouver une solution techniquement possible
  - COMMENT
  - Organisation, implémentation physique

Analyse

- Comprendre le problème
  - Parfois, un cahier des charges textuel est rédigé par l'utilisateur
  - Poser des questions pour identifier les vrais besoins
  - Supprimer les ambiguïtés et incohérences, rendre précis, faire émerger les « non-dits »
  - Parler le langage de l'utilisateur (le client)
- Réaliser des modèles (abstractions)
  - L'ensemble des modèles constitue le modèle d'analyse
  - C'est une représentation + ou - formelle de l'information
- Le modèle d'analyse est contractuel
  - C'est un support de communication entre l'utilisateur et le concepteur
- En amont, il faut cerner les besoins par une étude préalable du domaine

- Résoudre le problème
  - Connaître
    - Les technologies
    - Les architectures appropriées
    - Les bonnes pratiques
  - Comparer les approches possibles (trade-off)
  - Appliquer des solutions standards quand c'est possible (design patterns, architectures de domaines (frameworks))
  - Parler le langage de l'utilisateur et de l'analyste
- Réaliser le modèle de conception
  - Adapter le modèle d'analyse
  - Allouer les composants du modèle d'analyse à des composants du modèle de conception

- L'entreprise et son système d'information
- Problème actuels du génie logiciel
- Analyse vs conception
- Documentation et CASE
- Cycles de vie
- Modèle
- Notions sur les approches orientées objet
- Introduction à UML

Importance de la documentation

- La réalisation de la documentation est perçue comme une contrainte
  - Elle sera faite plus tard (= jamais)
  - Elle n'est pas mise à jour
- Les spécifications et la conception doivent être documentées
  - Approbation des clients
  - Compréhension des développeurs
- Une bonne stratégie de tests peut être développée à partir des spécifications
- La documentation est indispensable pour la maintenance

Outils CASE

- Computer Aided Software Environment
  - Éditeur de diagrammes
  - Dictionnaire de données (repository)
  - Contrôle d'accès
  - Vérifications automatiques de la cohérence, complexité…
  - S'appuie sur une méthode
  - Génération automatique de documentation
  - Génération automatique de code
- Exemples
  - Magic Draw UML 11.5 (No Magic, Inc.)
  - Objecteering (Softeam)
  - Rose (Rational)
  - Mega (IBM)
  - Class Builder (*freeware*)
  - Outils gratuits de démonstration : UML Jude, visual paradigm (community)

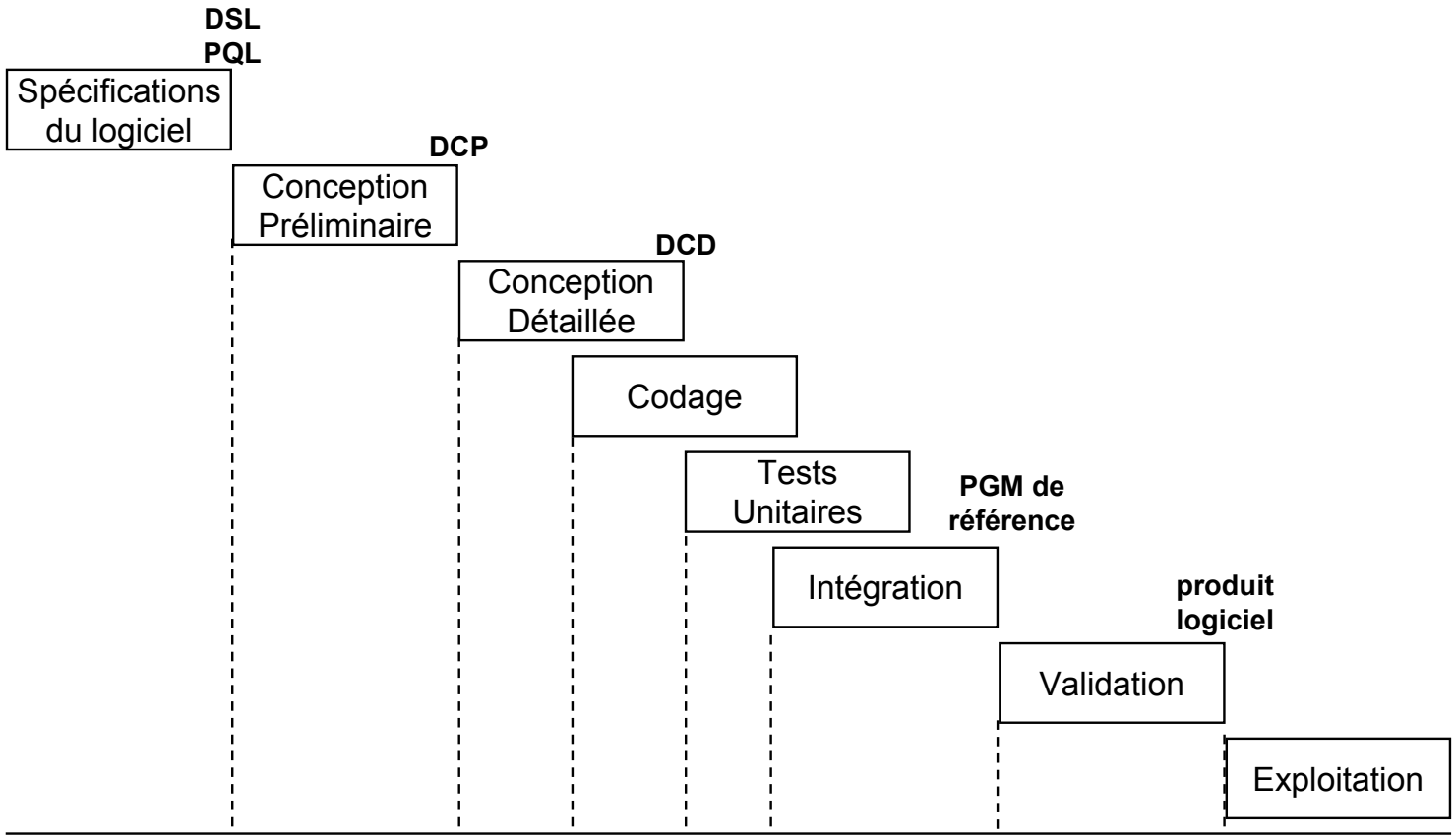
- L'entreprise et son système d'information
- Problème actuels du génie logiciel
- Analyse vs conception
- Documentation et CASE
- Cycles de vie
- Modèle
- Notions sur les approches orientées objet
- Introduction à UML

Démarche pour maîtriser et bâtir une application

- Déterminer les buts
- Définir le processus (cycle de vie)
- Utiliser une méthode
- S'appuyer sur un formalisme

Cycle de vie en cascade

(normalisé AFCIQ)



Cycle de vie en cascade

- **Spécification du logiciel**

  Ensemble des activités consistant à définir de manière précise, complète et cohérente ce dont l'utilisateur a besoin

- **Conception préliminaire**

  Ensemble des activités conduisant à l'élaboration de l'architecture du logiciel

- **Conception détaillée**

  Ensemble des activités consistant à détailler les résultats de la conception préliminaire, tant sur le plan algorithmique que sur celui de la structure des données, jusqu'à un niveau suffisant pour permettre le codage

- **Codage**

  Activité permettant de traduire le résultat de la conception détaillée en un programme à l'aide d'un langage de programmation donné

Cycle de vie en cascade

- **Tests unitaires**

  Activité ayant pour but de vérifier pour chaque composant du logiciel pris isolément que :
  - Tous les chemins logiques sont parcourus au moins une fois
  - La plage de validité des données d'entrée et de sortie a été explorée
  - Les résultats sont conformes au dossier de conception détaillée

- **Intégration**

  Activité consistant à assembler progressivement les composants du logiciel identifiés lors de la phase de conception préliminaire et contrôlés lors des tests unitaires

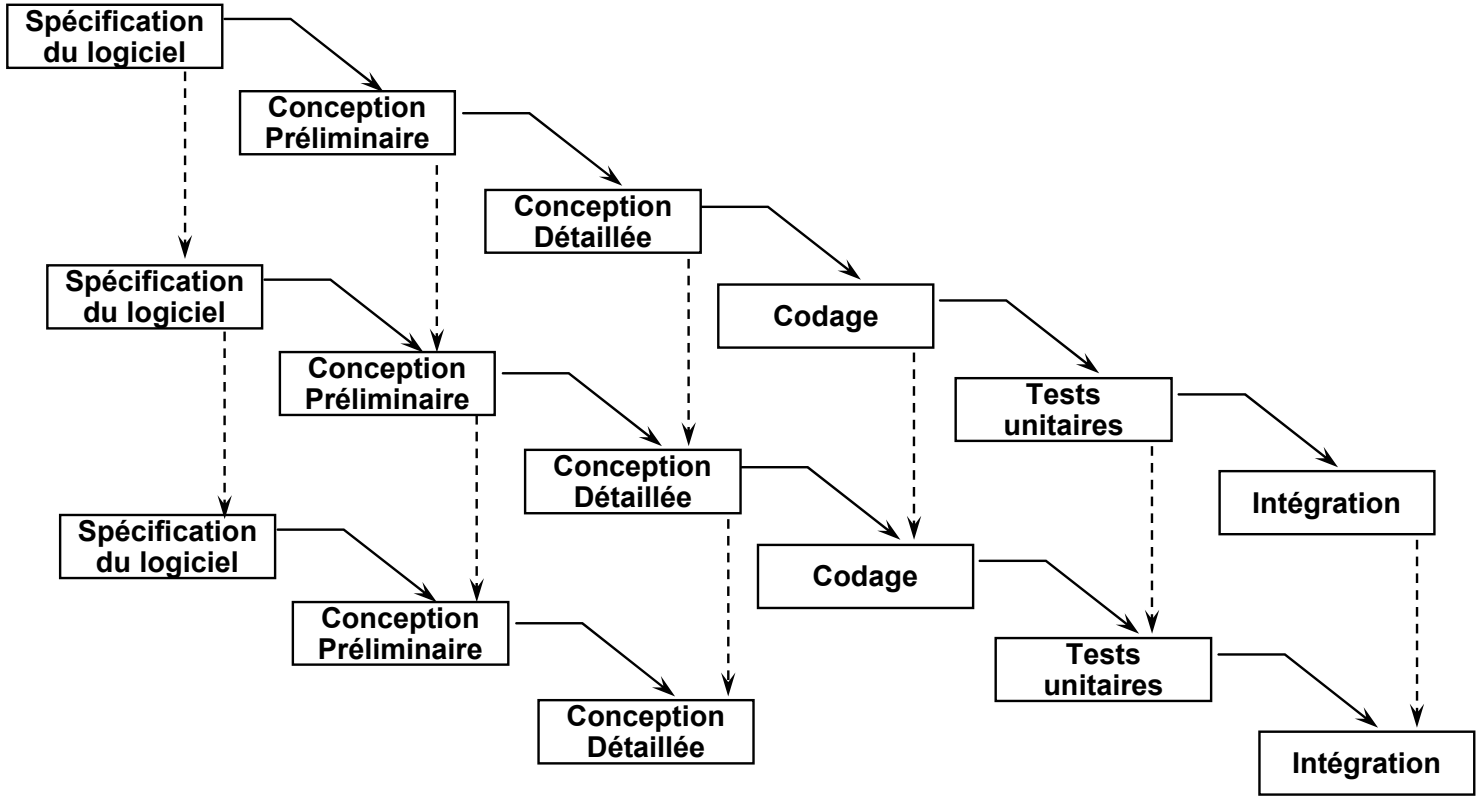
- **Validation**

  Activité conduisant à s'assurer, essentiellement au moyen de tests, qu'un logiciel est conforme au dossier de spécifications du logiciel

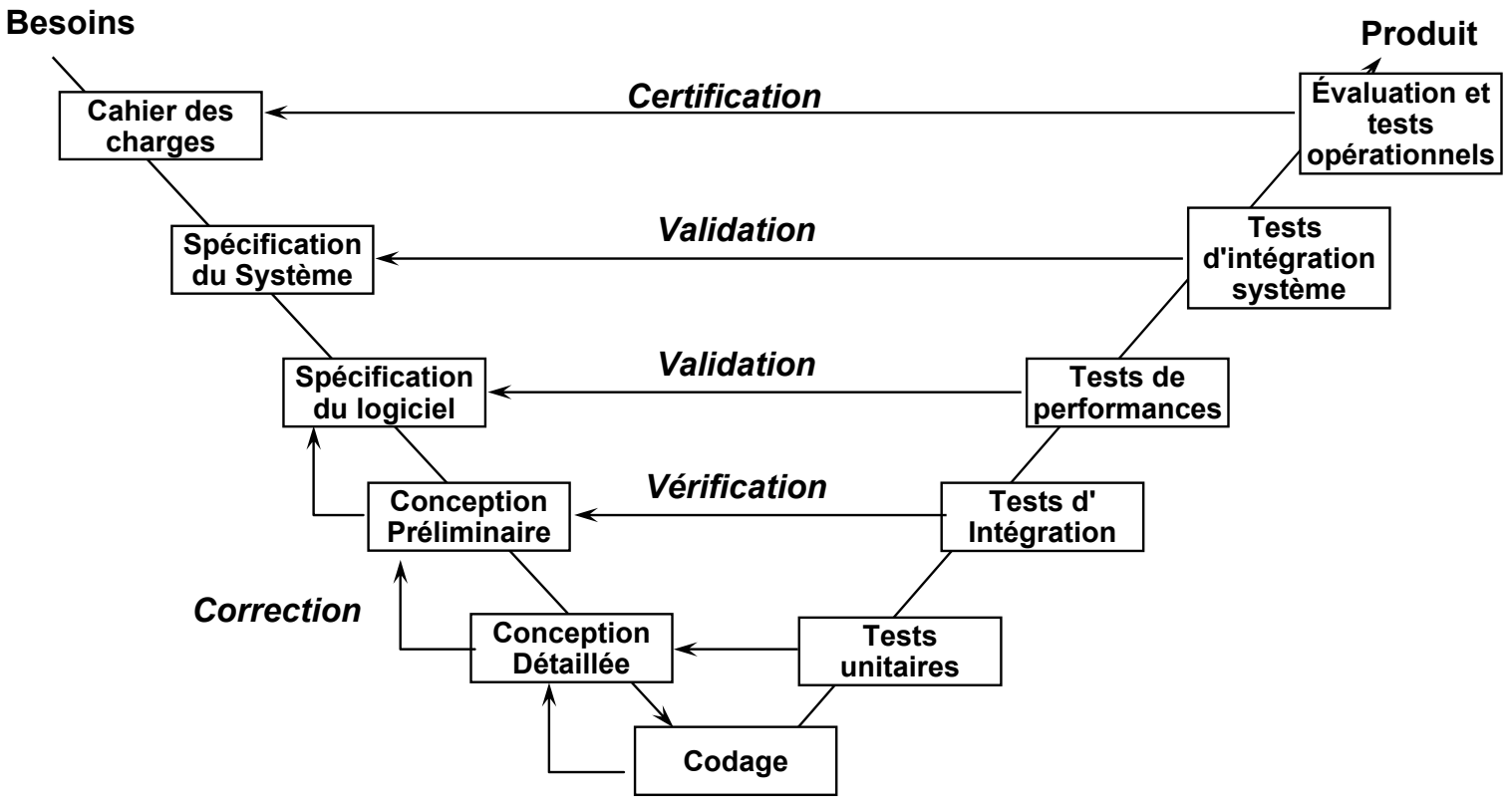
Cycle de vie en cascade

- Exploitation
  - Exploitation du logiciel
    - Ensemble des activités liées à la mise en œuvre opérationnelle d'un logiciel dans un environnement déterminé
  - Utilisation du logiciel
    - Ensemble des activités liées au besoin pour lequel le logiciel a été développé
  - Maintenance du logiciel
    - Ensemble des activités liées à la détection et la correction des défauts résiduels
  - Adaptation du logiciel
    - Ensemble des activités liées aux évolutions du dossier de spécification du logiciel

Cycle Incrémental



Cycle en V


Besoins
Produit
Cahier des charges
Certification
Évaluation et tests opérationnels
Spécification du Système
Validation
Tests d'intégration système
Spécification du logiciel
Validation
Tests de performances
Conception Préliminaire
Vérification
Tests d' Intégration
Correction
Conception Détaillée
Tests unitaires
Codage

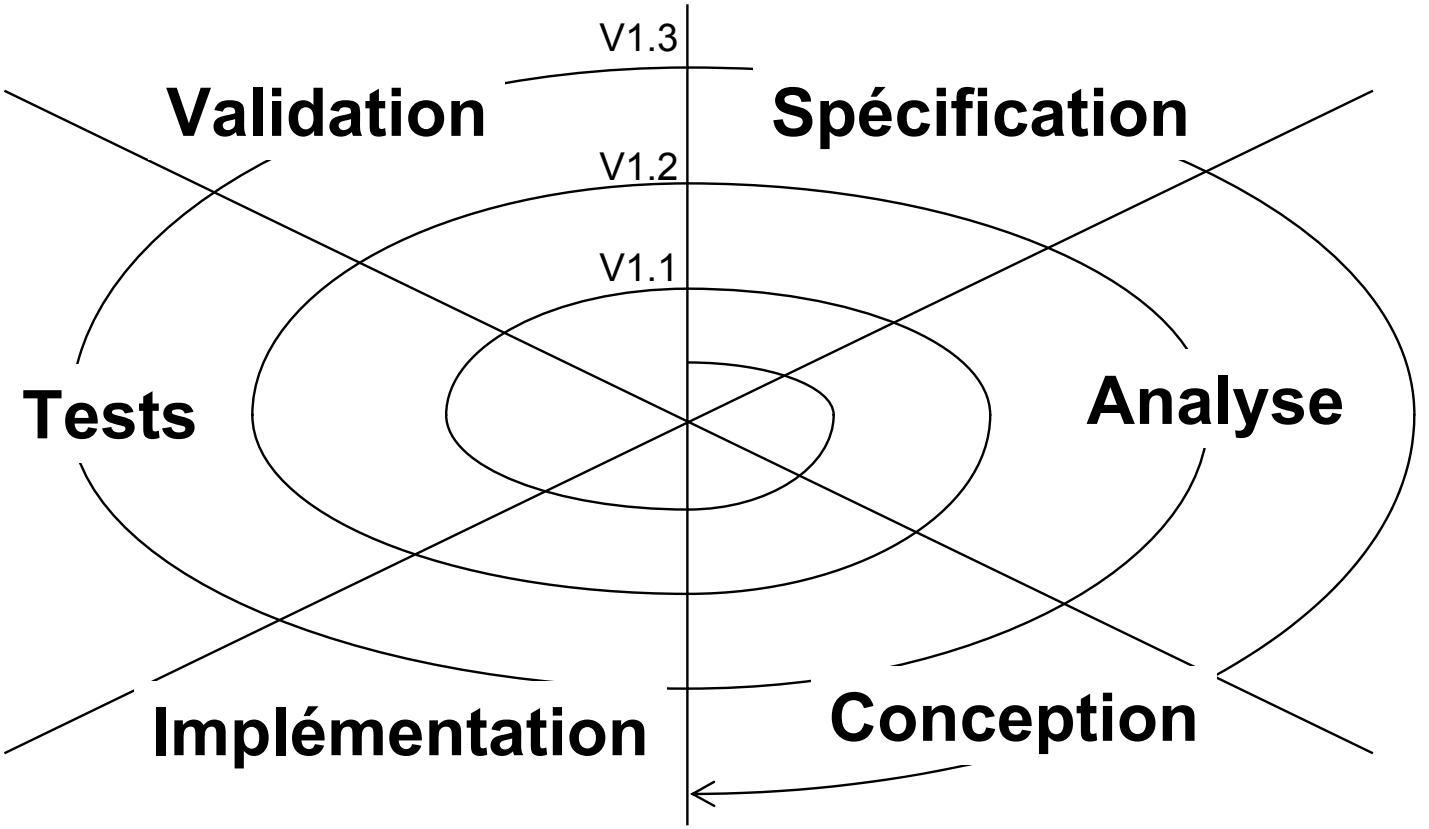
Avantages et limites du cycle en cascade ou en V

- A fait ses preuves sur de gros projets
- Le plus utilisé, le mieux compris
- Nombreuses opportunités de retour d'information
- N'est qu'un modèle dont la réalité s'écarte souvent
- Ne prend pas en compte l'aspect économique

Le cycle de vie objet

- 3 caractéristiques fondamentales :
  - Traçabilité entre les étapes
    - Correspondance aisée entre les éléments définis dans 2 phases successives
    - Une difficulté : passer des besoins (fonctionnels) à l'analyse (objets)
  - Caractère itératif
  - Caractère incrémental

Cycle en spirale

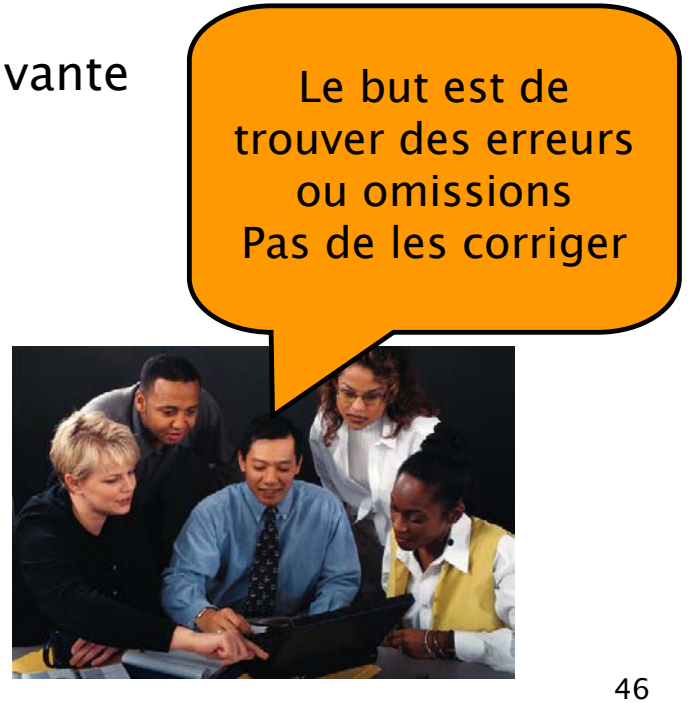


Avantages et limites du cycle en spirale

- Permet une grande flexibilité
- Très utile si les exigences initiales sont peu claires ou fortement évolutives
- Bien adapté aux technologies nouvelles
- Se prête difficilement aux plans à long terme
- Ne prend pas en compte l'aspect économique

Validation des étapes par des revues (recettes)

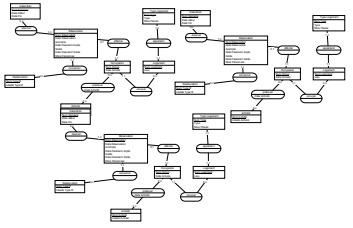
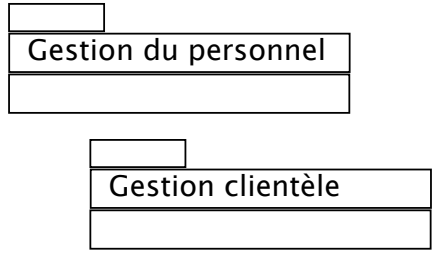
- Les revues doivent être courtes et bien ciblées
  - 1h maxi
- Ne pas entrer dans des discussions improductives
  - Règle des 3mn
    - Si le débat sur un point nécessite plus de temps, noter le problème
    - Réponse lors de la revue suivante
- Livrables
  - Fiches de recette
  - Modifications à apporter
  - Liste de questions



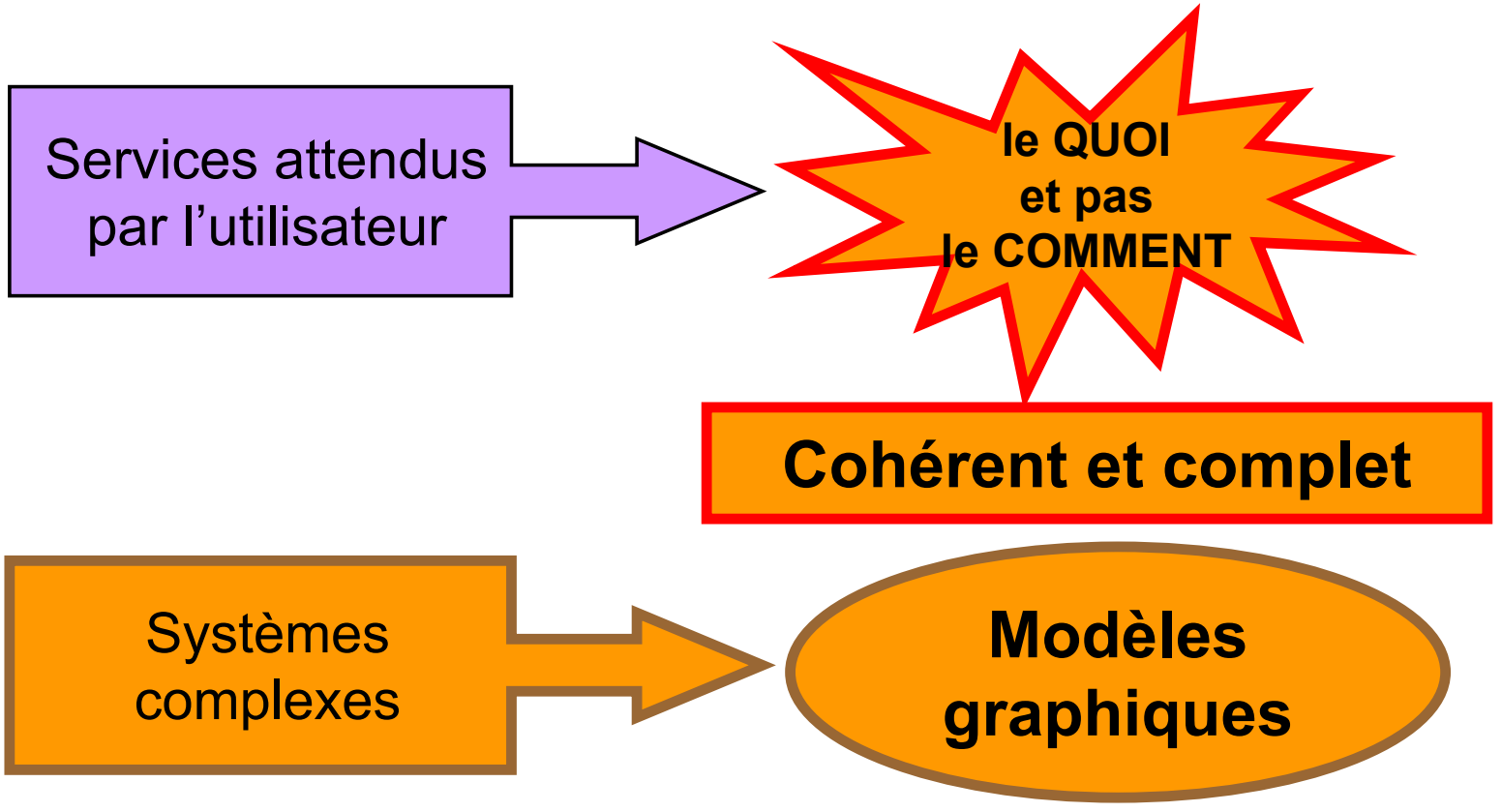
- L'entreprise et son système d'information
- Problème actuels du génie logiciel
- Analyse vs conception
- Documentation et CASE
- Cycles de vie
- Modèle
- Notions sur les approches orientées objet
- Introduction à UML

Modèle (cf cours JPB)

- Image simplifiée de la réalité
- Établi pour répondre à un type de questions
  - Abstraction pour montrer un aspect particulier
  - Inutilisable pour un autre aspect => plusieurs modèles
- Facilite la communication
  - Différentes vues permettent de mieux comprendre
  - Limites humaines (7 +- 2)

Peu lisible	Plus lisible
3,141592653589	3,14.15.92.65.35.89
1851159350103	1 85 11 59 350 103
	

Besoins


Services attendus par l'utilisateur
le QUOI et pas le COMMENT
Cohérent et complet
Systèmes complexes
Modèles graphiques

La carte conceptuelle : quoi, pourquoi ?
(cours C Verzat)

Analytique
logico-déductif
abstraction
langage

Synthétique
global
action
images

Fonctionne en mode

Cerveau gauche + droit

sollicite

Représentation graphique

est

Carte conceptuelle

Permet de

Structurer / hiérarchiser un grand nombre d'infos

dépasser les limites descriptives du langage parlé et écrit

Ce qui favorise

compréhensio
n
mémorisation
créativité

relations

interconnectés

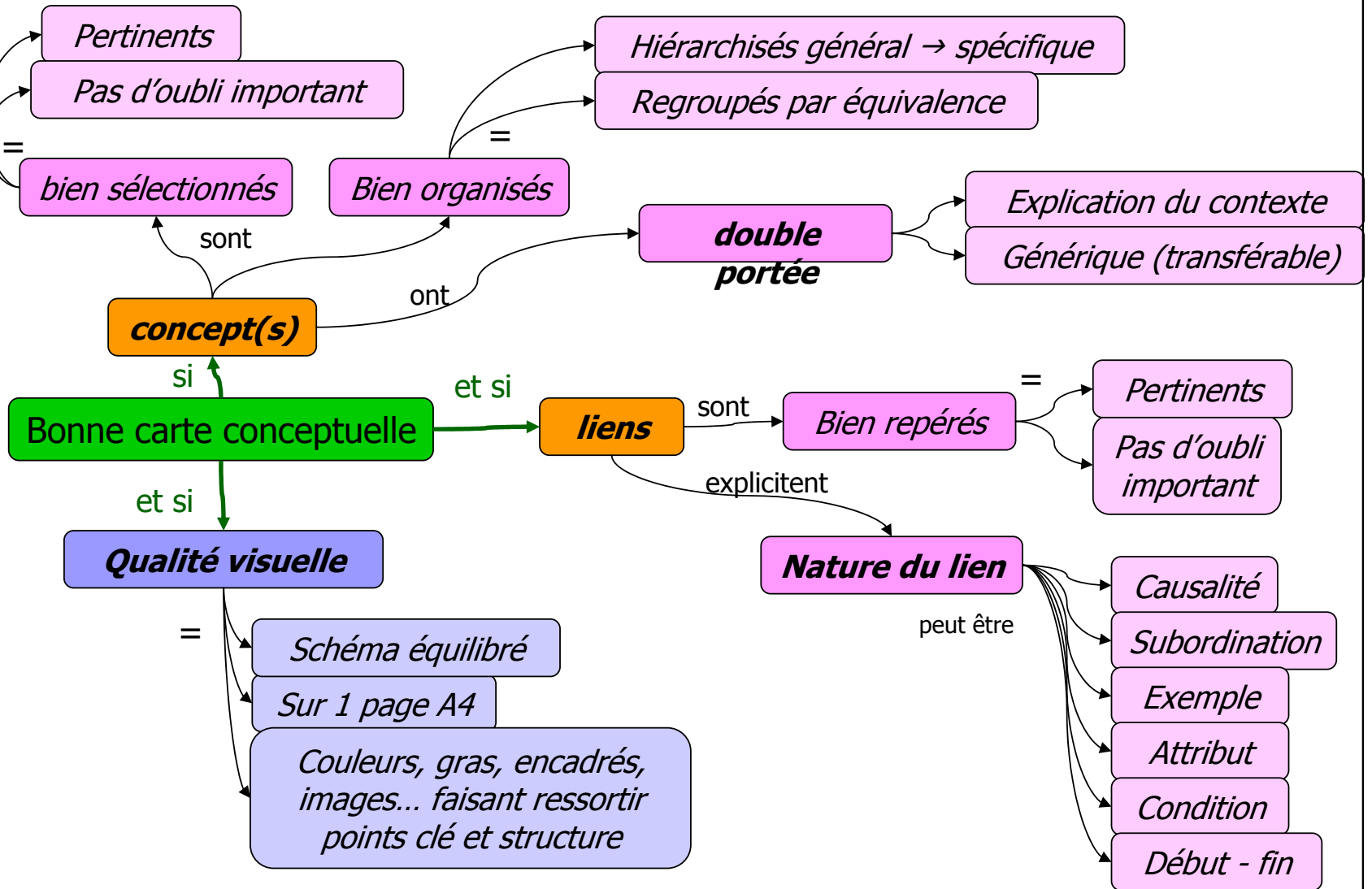
Concept(s)

composée de

Perçu(s) par

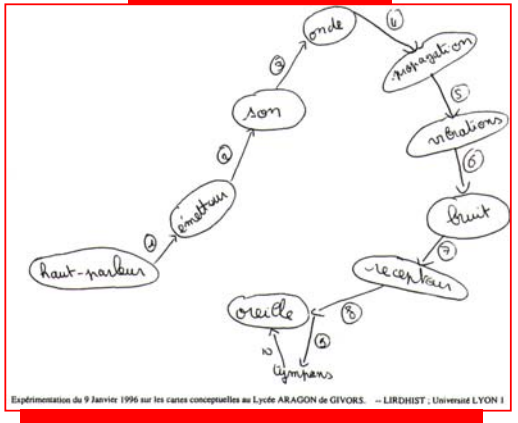
1 individu ou +

Qu'est-ce qu'une bonne carte conceptuelle ?
(cours C Verzat)



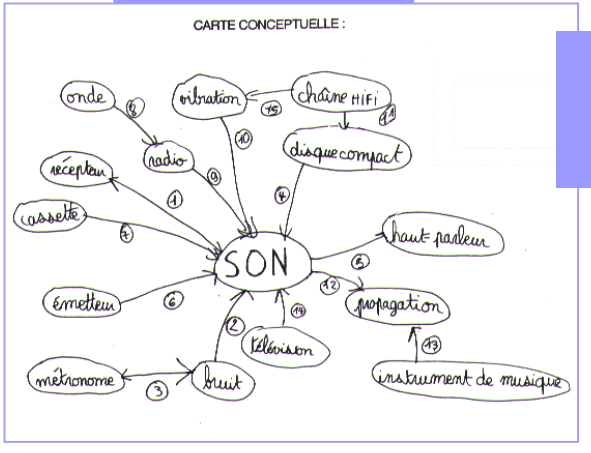
La structure des cartes reflète un type de théorie explicative (cours C Verzat)

« chaîne »



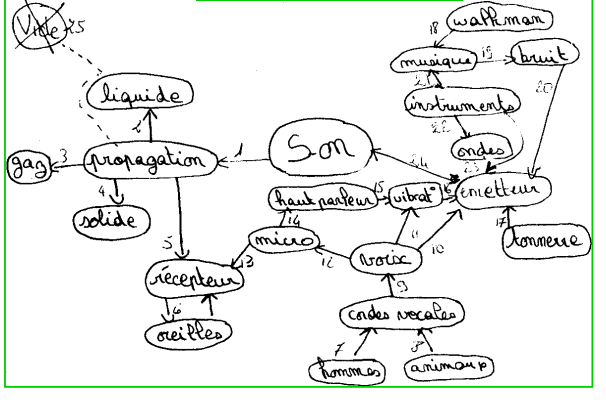
Raisonnement séquentiel et causal

« chardon »



Acquisition de connaissances cloisonnées

« réseau »



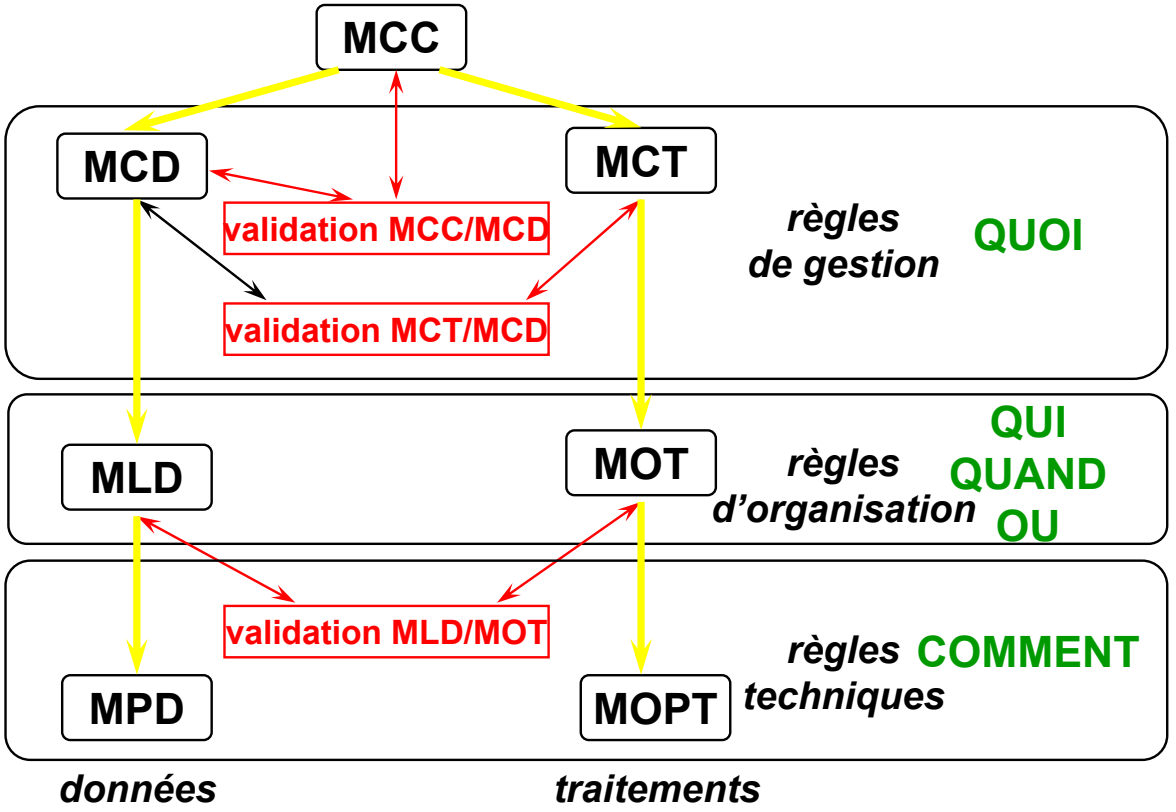
Meilleure interconnexion des concepts

- Pour en savoir plus
  - http://members.optusnet.com.au/~charles57/Creative/Mindmap/
- Logiciels de fabrication de cartes conceptuelles
  - http://www.licef.teluq.uquebec.ca/fr/index.htm
    - suivre le lien "MOT et· MOT Plus"
    - (ou http://rb.ec-lille.fr/l/CarteConceptuelle/mot23fr.exe)
  - IHMC CmapTools : http://cmap.ihmc.us/download/

Trois approches de modélisation

- Modélisation des données
  - Quoi
- Modélisation des comportements
  - Quand, comment
- Modélisation objet
  - Quoi, quand, comment

Cycle d'abstraction Merise



- L'entreprise et son système d'information
- Problème actuels du génie logiciel
- Analyse vs conception
- Documentation et CASE
- Cycles de vie
- Modèle
- Notions sur les approches orientées objet
- Introduction à UML

Pourquoi l'approche objet ?

- Stabilité de la modélisation par rapport aux entités du monde réel
- Construction itérative facilitée par le faible couplage entre composants
- Possibilité de réutiliser des éléments
- Simplicité du modèle :

  5 concepts : objets, messages, classes, généralisation, polymorphisme

Pourquoi l'approche objet ?

- Stabilité de la modélisation par rapport aux entités du monde réel
- Construction itérative facilitée par le faible couplage entre composants
- Possibilité de réutiliser des éléments
- Simplicité du modèle :
  - 5 concepts :
    - Objets
    - Classes
    - Messages
    - Généralisation
    - Polymorphisme

- Approche objet : encapsulation

  Objet = état + comportement + identité

- Exemple :

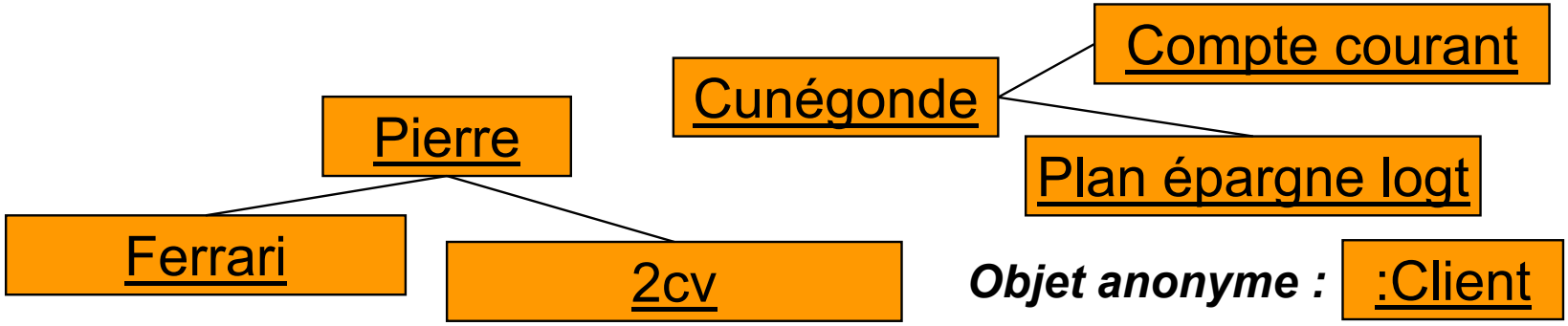
  Un objet Personne

Informations
nom = « BOND »
âge = 35 ans
société = « MI6 »

Comportements
nomEnMajuscule
changerAge
changerSociété

Caractéristiques d'un objet

- Objet : peut être concret ou abstrait

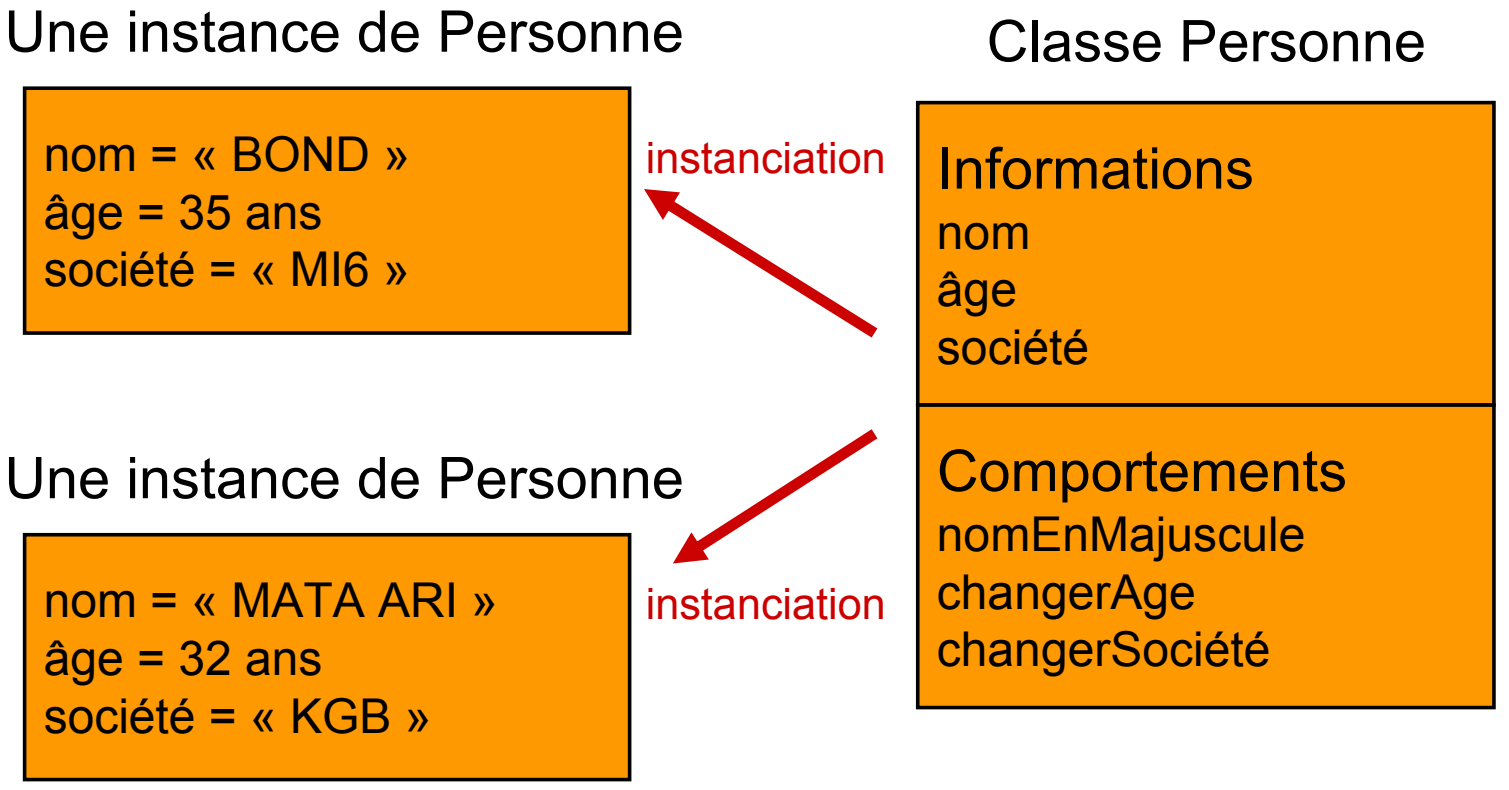


- Intérêt de la relation d'encapsulation :
  - Forte cohésion interne
  - Faible couplage avec l'extérieur
- Chaque objet possède une identité unique (non explicitée sur le modèle)
- État : regroupe les valeurs de tous ses attributs à un instant donné

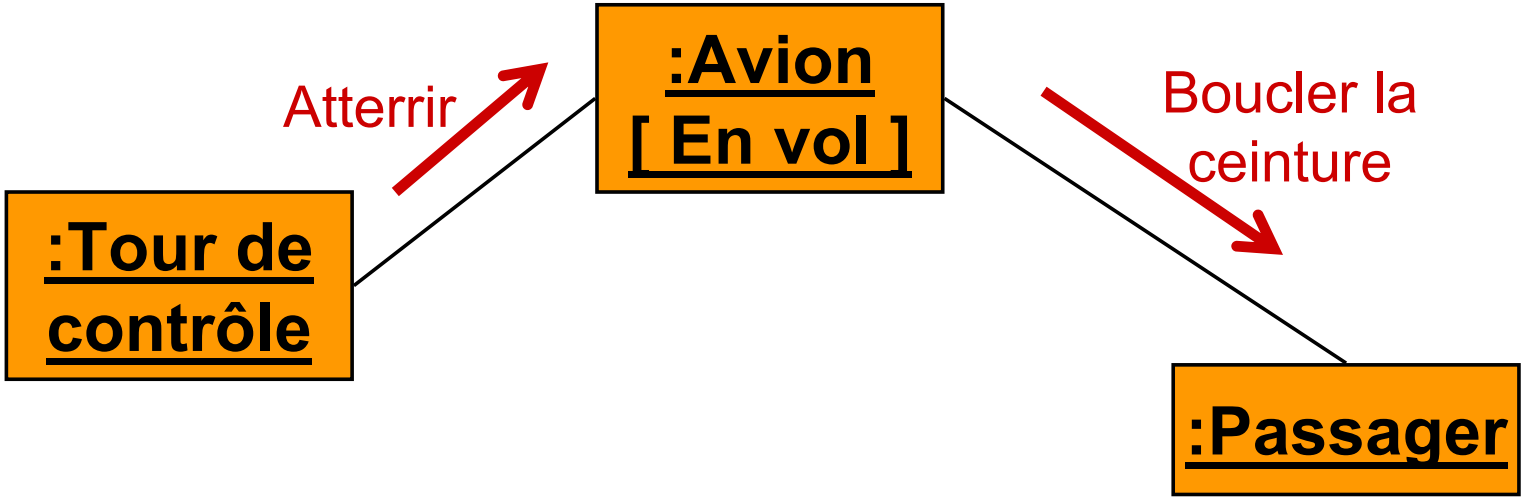
Notion de classe

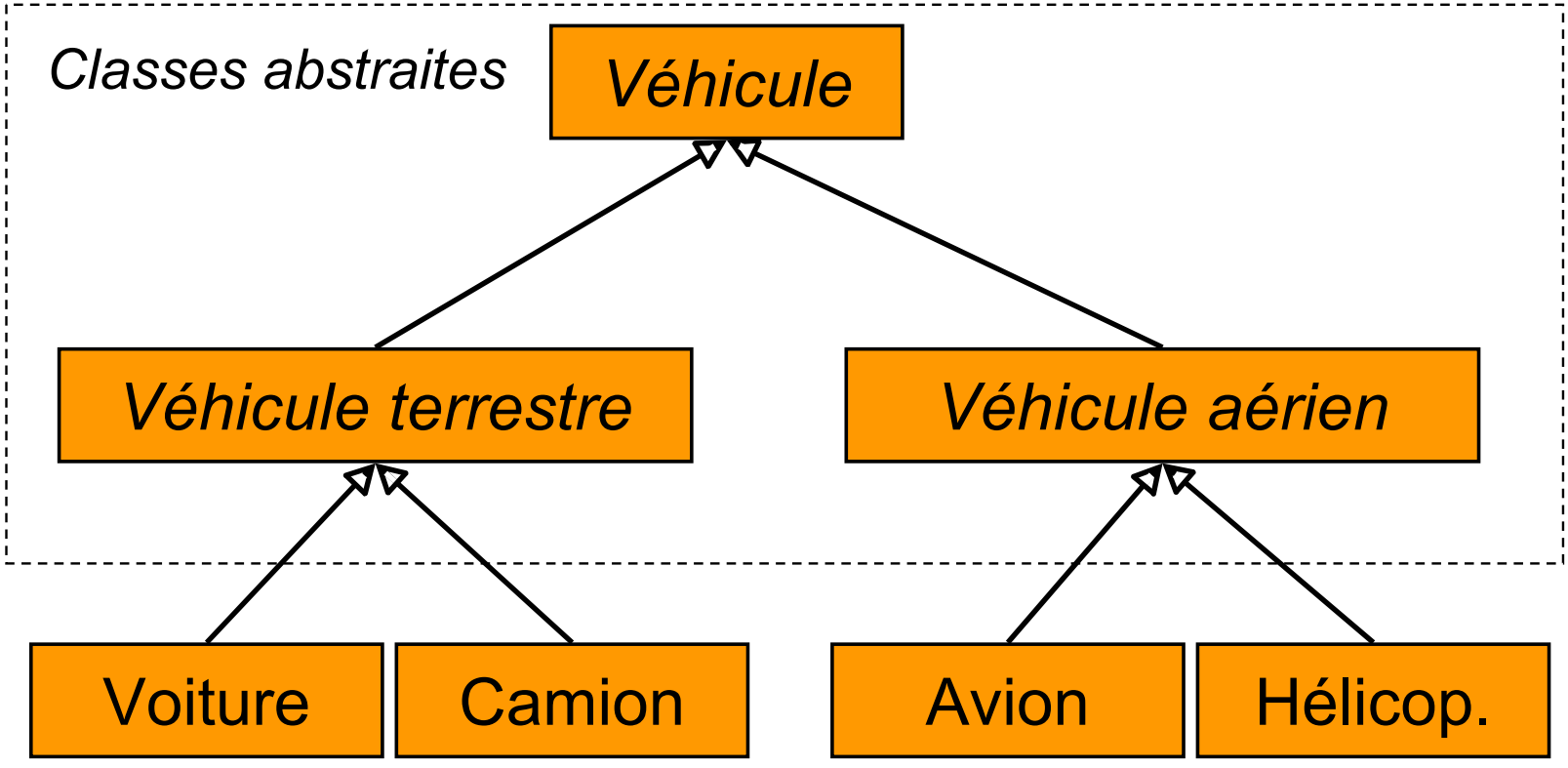
- Classe (d'objets) : abstraction d'objets ayant :
  - Des propriétés similaires
  - Un comportement commun
  - Des relations identiques avec les autres objets
  - Une sémantique commune
- Tout objet connaît sa classe
  - Propriété implicite d'un objet
  - Pb : changement de classe (transmutation)
- Classe : abstraction du problème, généralisation

Exemple de classe

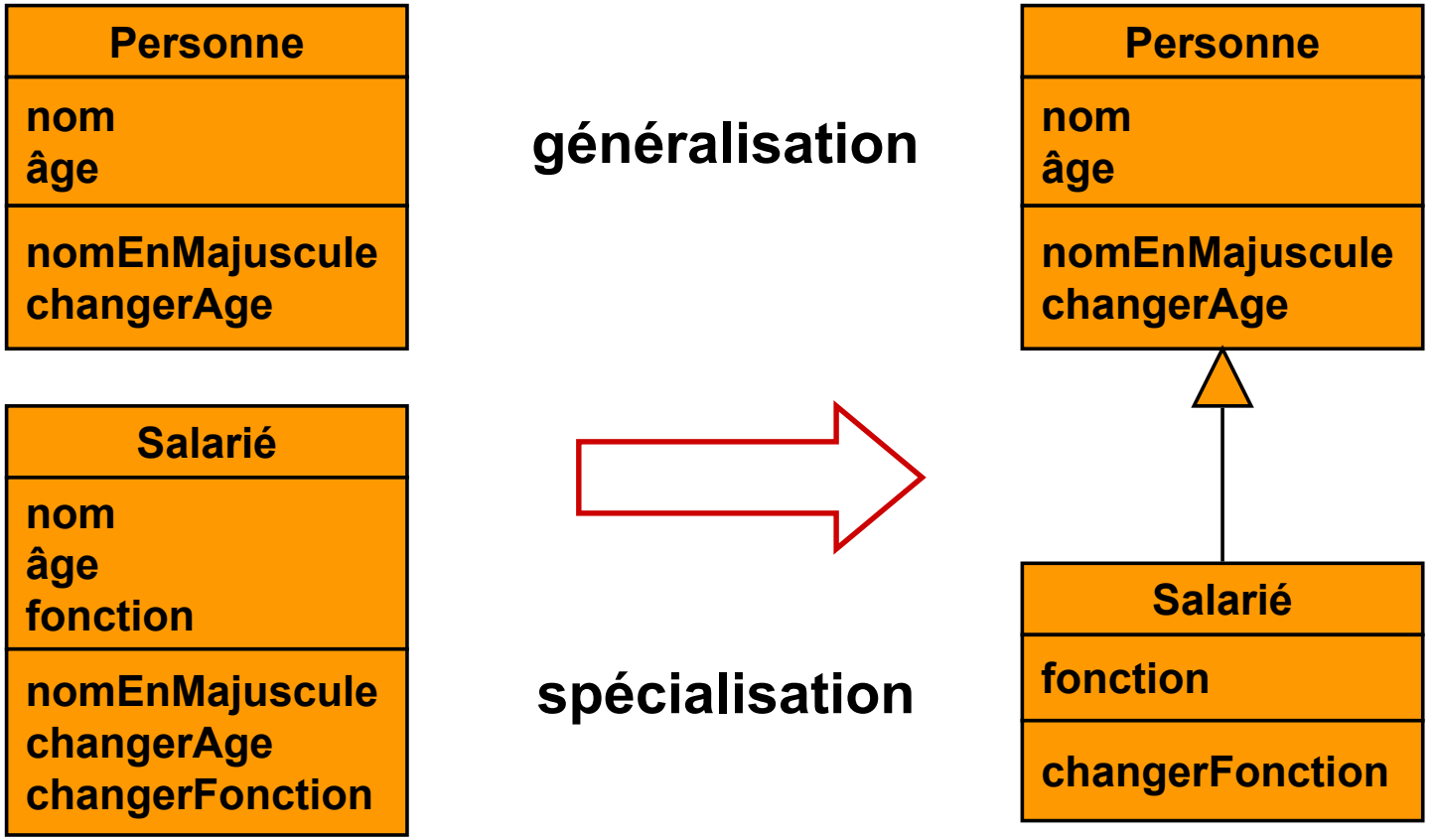


- Message : stimulation externe d'un objet par un autre
- Les opérations (actions et réactions) sont déclenchées suite à l'envoi d'un message par un autre objet

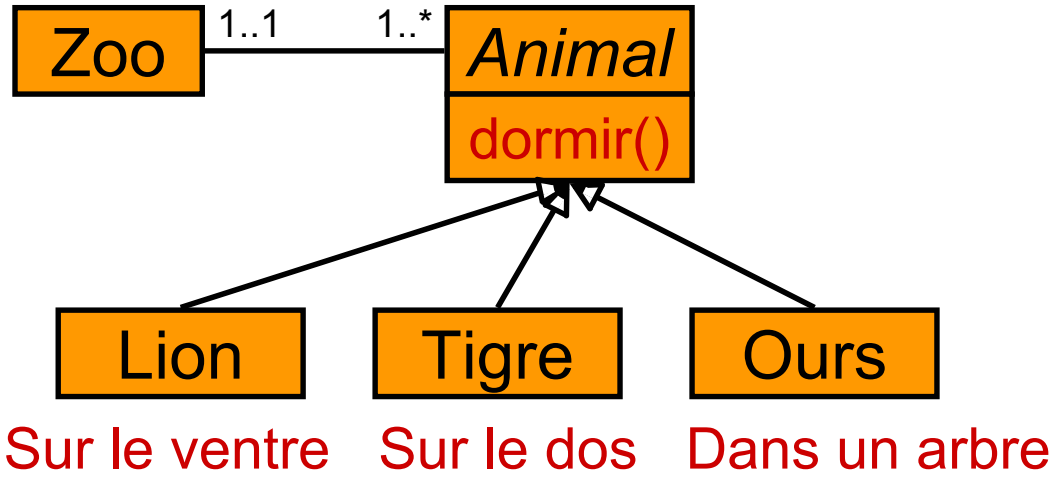


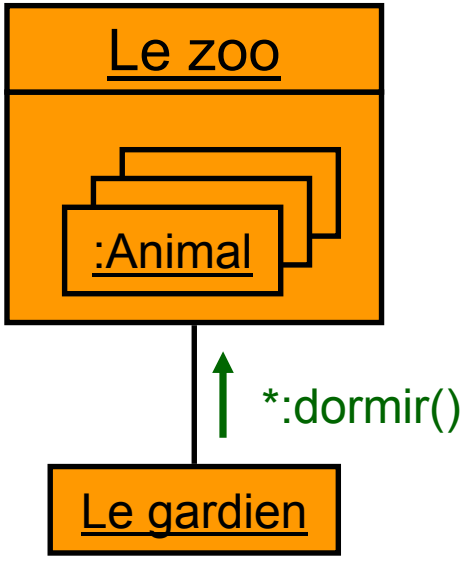

Classes abstraites
Véhicule
Véhicule terrestre
Véhicule aérien
Voiture
Camion
Avion
Hélicop.

Généralisation – spécialisation

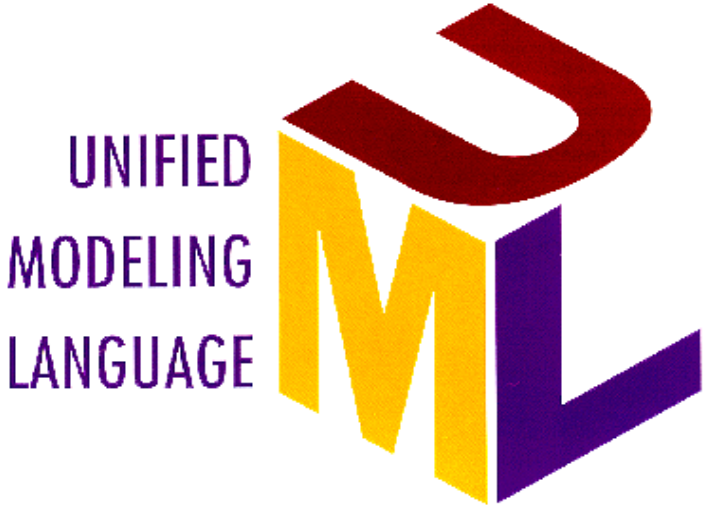


- Un nom d'objet peut désigner des instances de classes différentes issues d'une même arborescence
- L'objet peut réagir différemment à des opérations communes



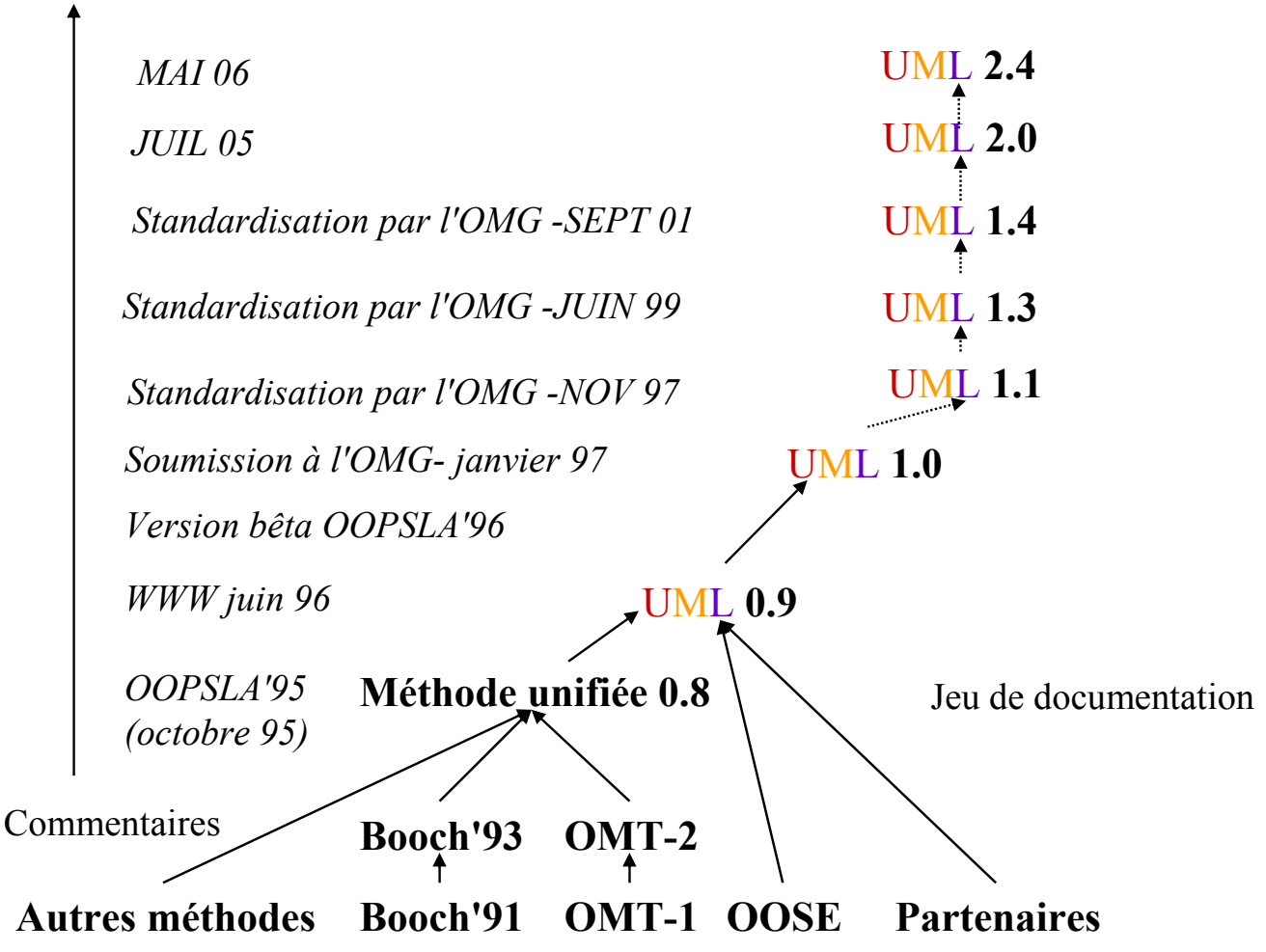


- L'entreprise et son système d'information
- Problème actuels du génie logiciel
- Analyse vs conception
- Documentation et CASE
- Cycles de vie
- Modèle
- Notions sur les approches orientées objet
- Introduction à UML

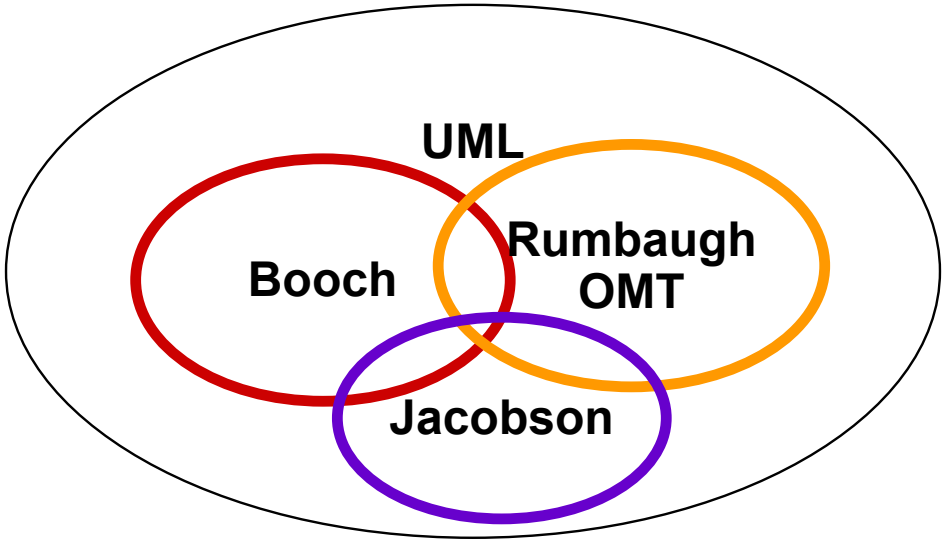


- Faire face à la prolifération des méthodes objets
- Rechercher un langage commun unique

Bref historique



En résumé



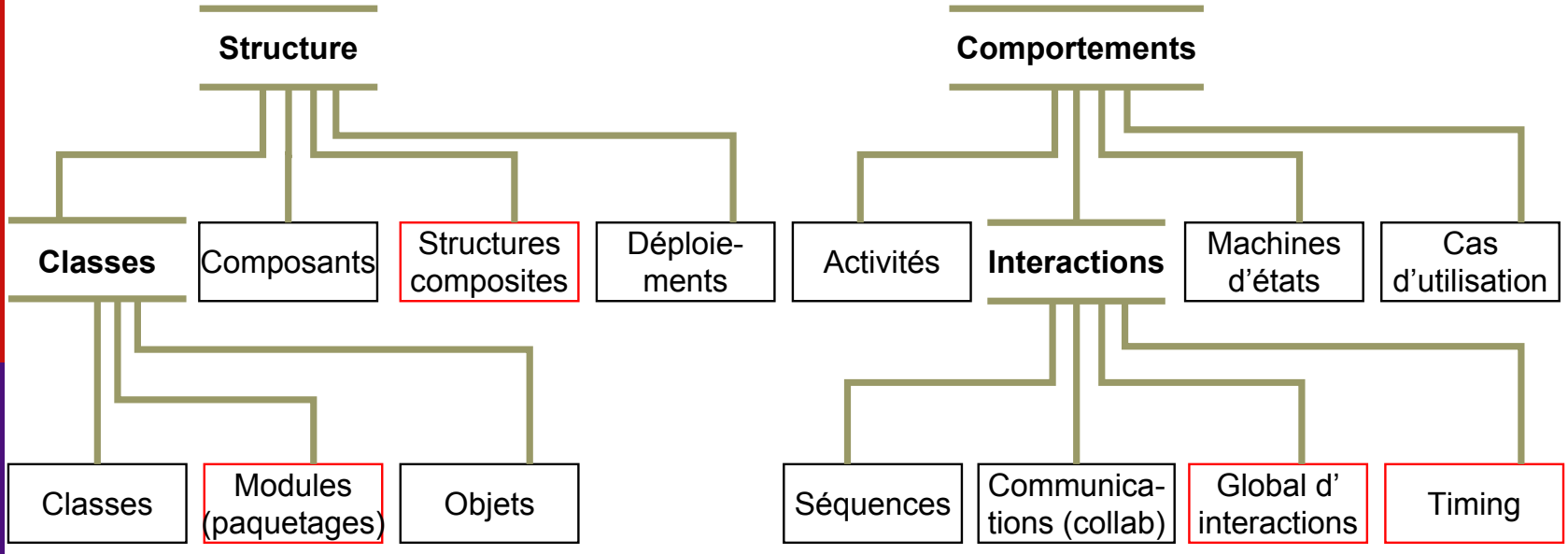
- UML est une notation, pas une méthode
- UML est un langage de modélisation objet
- UML convient à toutes les méthodes objet
- UML est dans le domaine public
- UML est riche
- UML est extensible

Informations sur UML

- Spécifications UML (voir sites web)
  - document au format HTML ou PDF
    - UML Summary, Notation Guide, Semantics, Glossary…
- Sites web
  - http://www.omg.org
  - http://www.uml.org

Hiérarchie des diagrammes d'UML 2

- 2 grandes catégories
  - Diagrammes de structure
  - Diagrammes de comportement



Diagrammes de structure

- Diagramme de classes
  - Représente la structure statique en termes de classes et de relations
- Diagramme d'objets
  - Représente les objets et leurs relations
- Diagramme de modules
  - Montre le regroupement des classes
- Diagramme de composants
  - Représente les composants logiciels d'une application
- Diagramme de déploiement
  - Représente le déploiement des composants sur les dispositifs physiques
- Diagramme de structures composites
  - Montre les liens entre diagramme de classes et diagramme de composants

Diagrammes de comportement

- Diagramme d'activité
  - Décrit le comportement en termes d'actions, de flux
- Diagramme de cas d'utilisation
  - Représente les fonctions du système du point de vue de l'utilisateur (besoins fonctionnels)
- Diagramme d'états-transitions
  - Représente le comportement des éléments en termes d'états et de changement d'états
- Diagrammes d'interactions

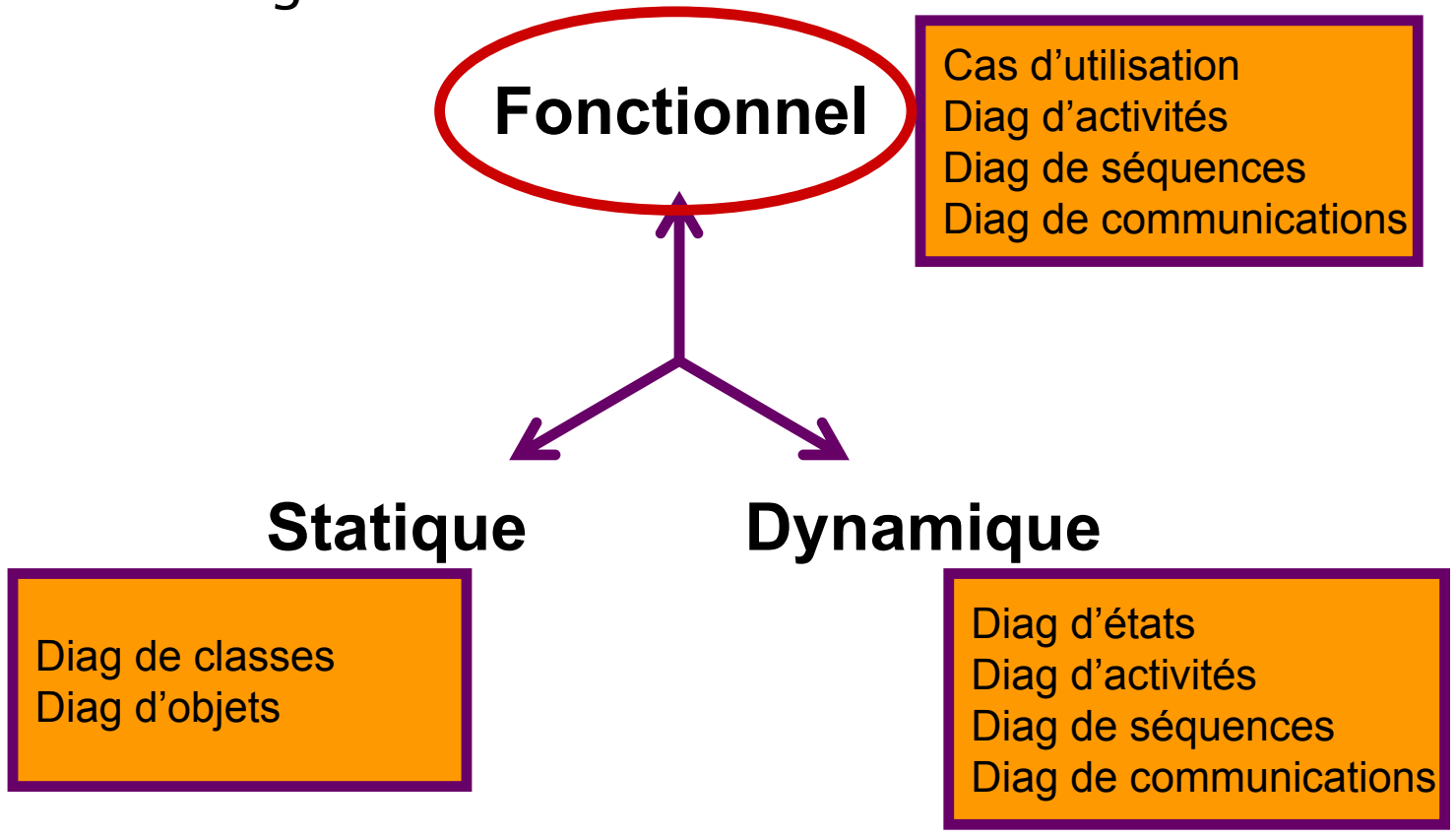
Diagrammes d'interactions

- Diagramme de séquence
  - Donne une représentation temporelle des objets et de leurs interactions
- Diagramme de communication
  - Donne une représentation spatiale des objets, des liens et des interactions
- Diagramme de timing
  - Décrit les spécifications temporelles (pour systèmes temps réel)
- Diagramme global d'interaction
  - Version simplifiée du diagramme d'activité qui se focalise sur les éléments qui interviennent dans la réalisation d'une activité plutôt que sur ses étapes

- Standard générique
  - Conçu pour « tout » modéliser
  - Se voit reprocher sa complexité
- Parfois concurrencé par les DSL
  - Domain Specific Language
- Ces approches se rejoignent autour du développement guidé par les modèles
  - Génération de code
- Nous n'étudierons pas tous les diagrammes

3 axes de modélisation…

…et les diagrammes étudiés

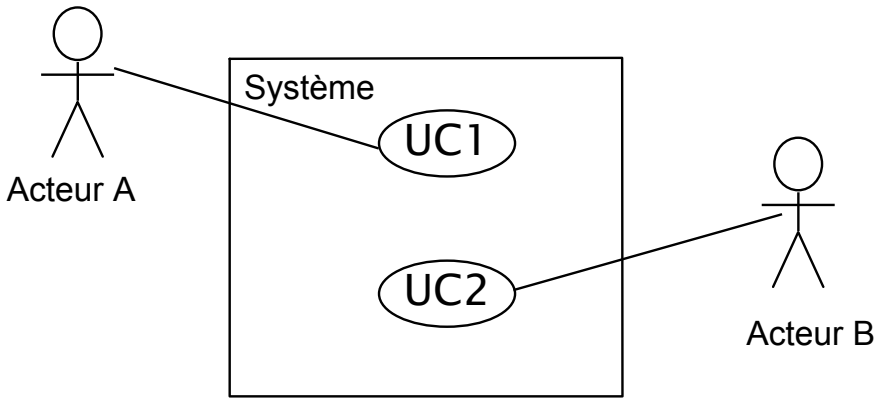


Rôle du diagramme des cas d'utilisation

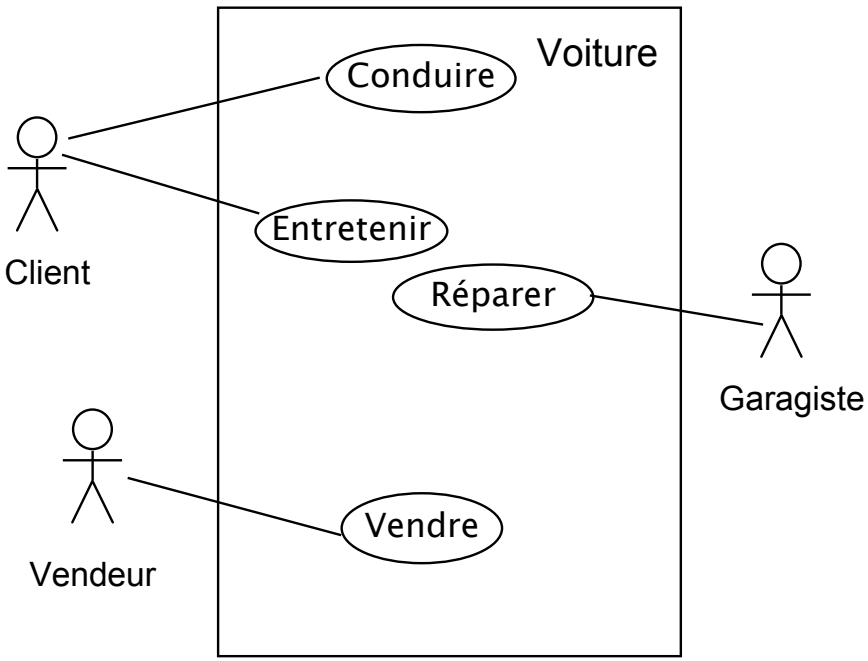
- Cas d'utilisation = use-case
- Permet l'analyse des besoins de l'utilisateur
- Sert à élaborer les jeux d'essais (tests)

Diagramme des cas d'utilisation

- Décrit le comportement du système vu de l'utilisateur : actions et réactions (fonctions)
- Un cas d'utilisation regroupe une famille de scénarii selon un critère fonctionnel
- Regroupement par catégories d'utilisateurs
- Comprend
  - Les acteurs
  - Le système
  - Les cas d'utilisations eux-mêmes

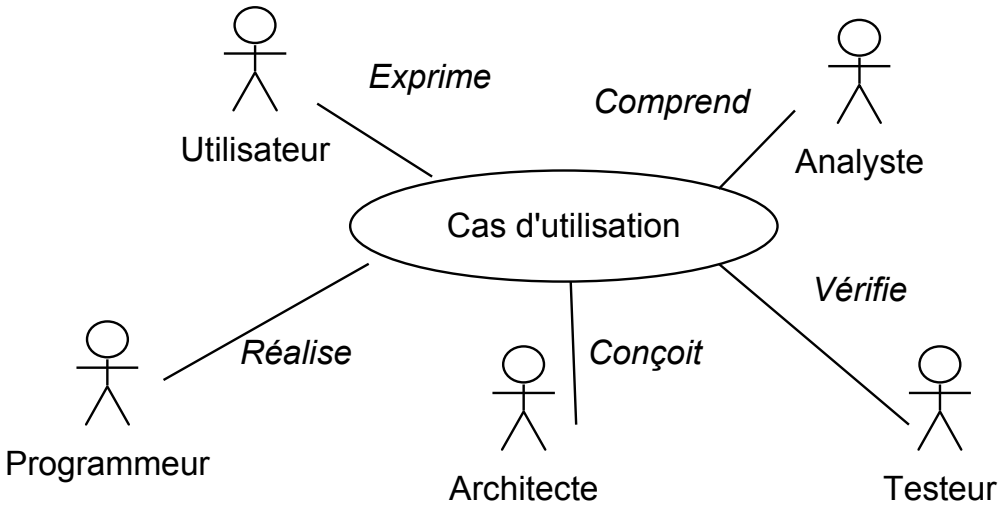


- Une même personne physique peut jouer le rôle de plusieurs acteurs

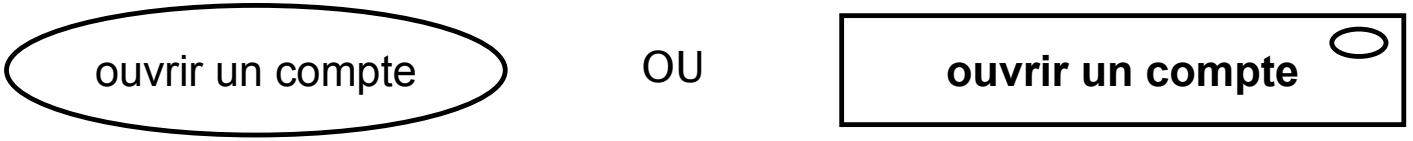


Catégories d'acteurs

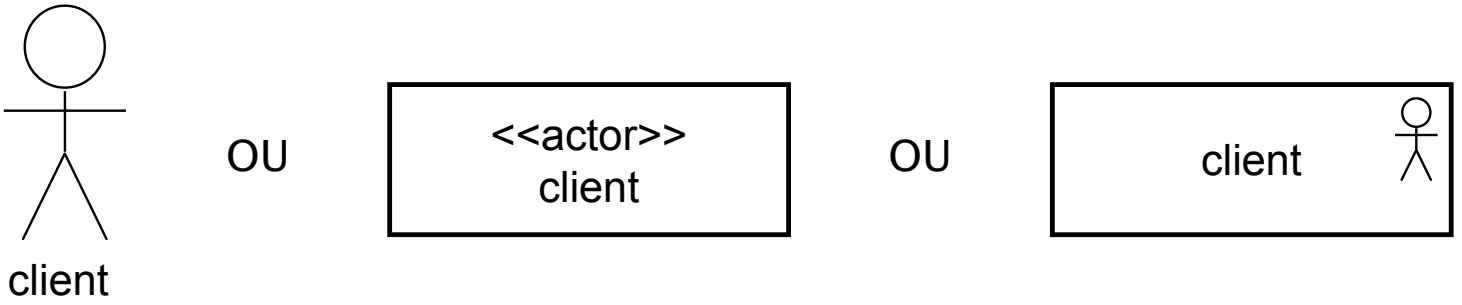
- Acteurs primaires : utilisent le système
- Acteurs secondaires : administrent le système
- Matériel externe
- Autres systèmes



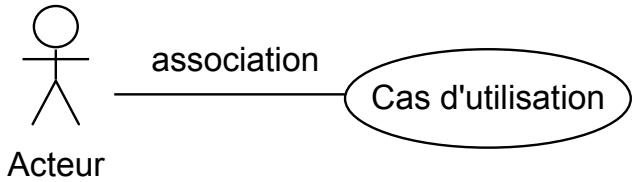
- Cas d'utilisation (ellipse ou notation en classificateur)



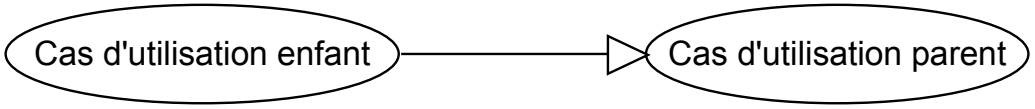
- Acteur (silhouette ou notation en classificateur)



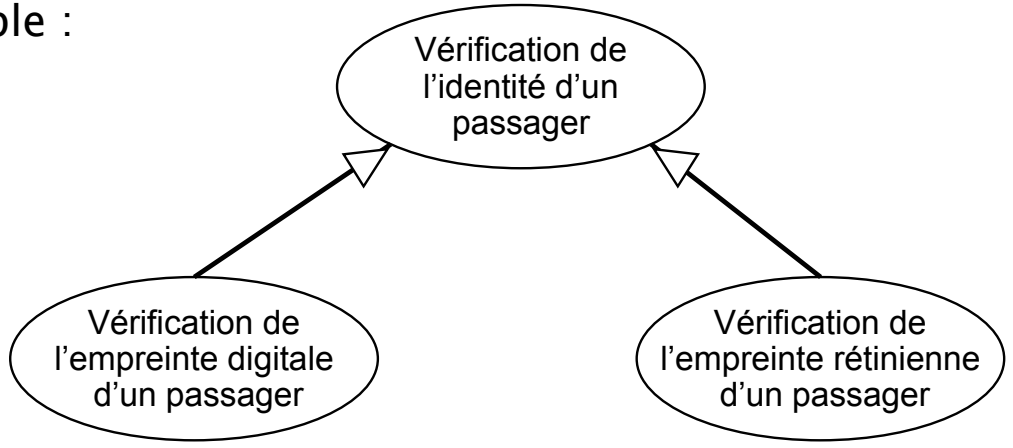
- Relation de communication entre acteur et cas d ’utilisation



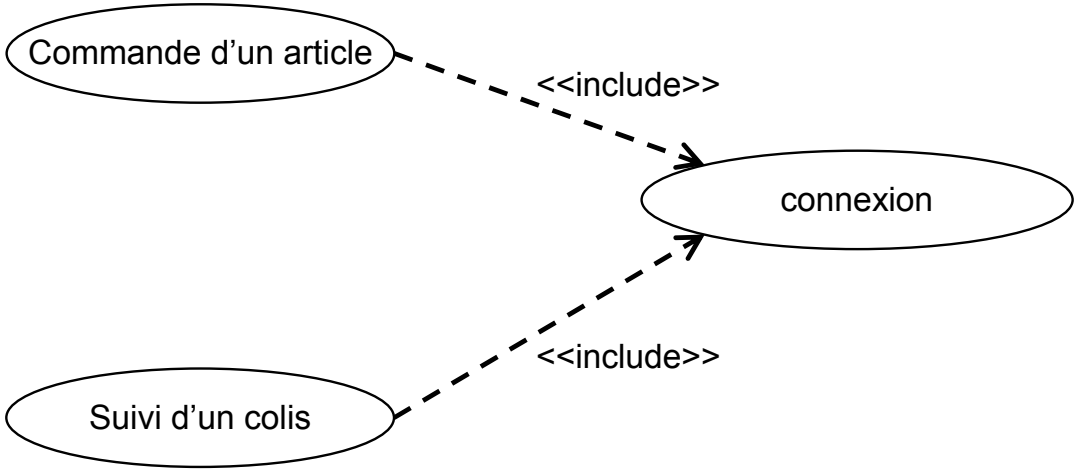
- Relation de généralisation
  - Le cas d'utilisation source est une généralisation du cas d 'utilisation destination



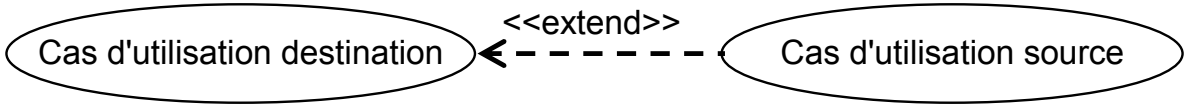
  - Exemple :



- Relation d'inclusion
  - On peut regrouper des fonctionnalités communes à plusieurs cas d'utilisation en créant un cas d'utilisation inclus, partagé
  - Le cas inclus n'est ni autonome ni complet : il constitue une partie nécessaire à la définition de cas d'utilisation plus larges

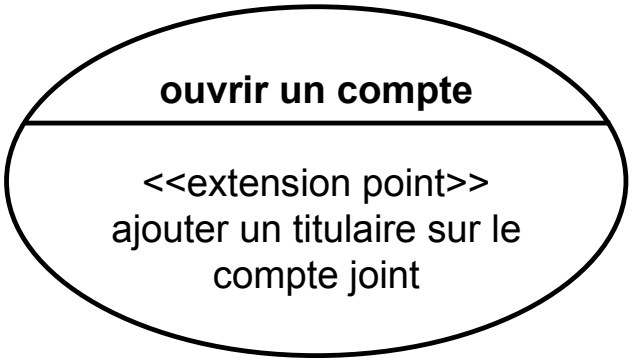


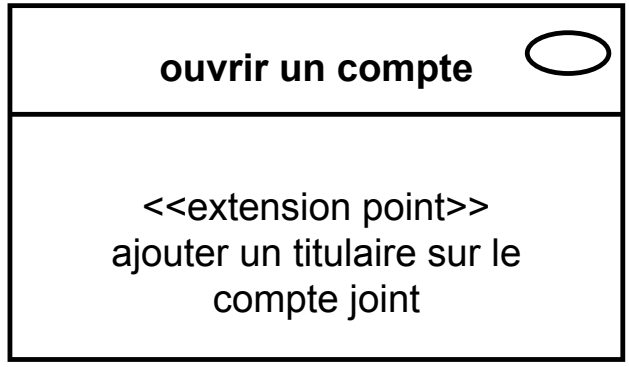
- Relation d'extension
  - Une instance du cas d'utilisation destination peut être étendu (sous certaines conditions) par le comportement spécifié par le cas source

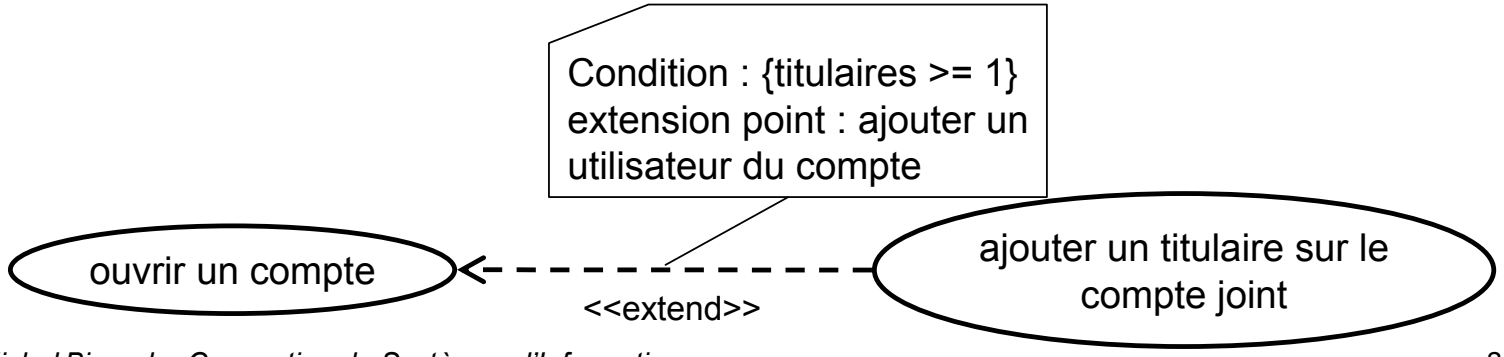


Exemple de relation d'extension

- Le cas « ajouter un titulaire sur le compte joint » étend le cas « ouvrir un compte » car il propose des fonctionnalités supplémentaires ; cependant, le cas de base est utilisable de façon autonome
- Les 3 notations ci-dessous sont possibles





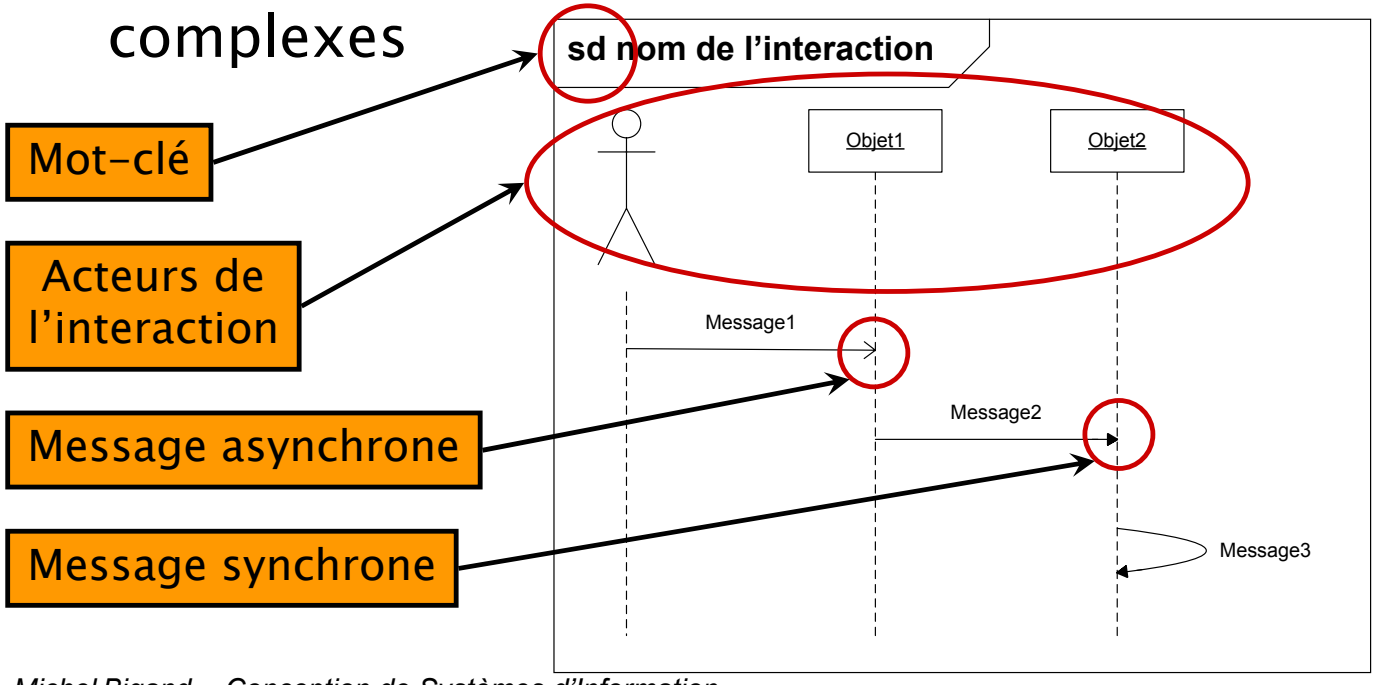


Notion de scénario (hors norme UML)

- Une fois les cas d'utilisation identifiés, il faut les décrire
- Un scénario représente une succession particulière de séquences d'actions qui s'exécutent du début à la fin du cas d'utilisation
- Description textuelle d'un scénario :
  - Titre
  - Résumé
  - Acteurs
  - Date de création / Date de MAJ / Auteur
  - Pré-conditions
  - Scénario nominal
  - Enchaînements alternatifs
  - Enchaînements d'exception
  - Post-conditions

Diagramme de séquences

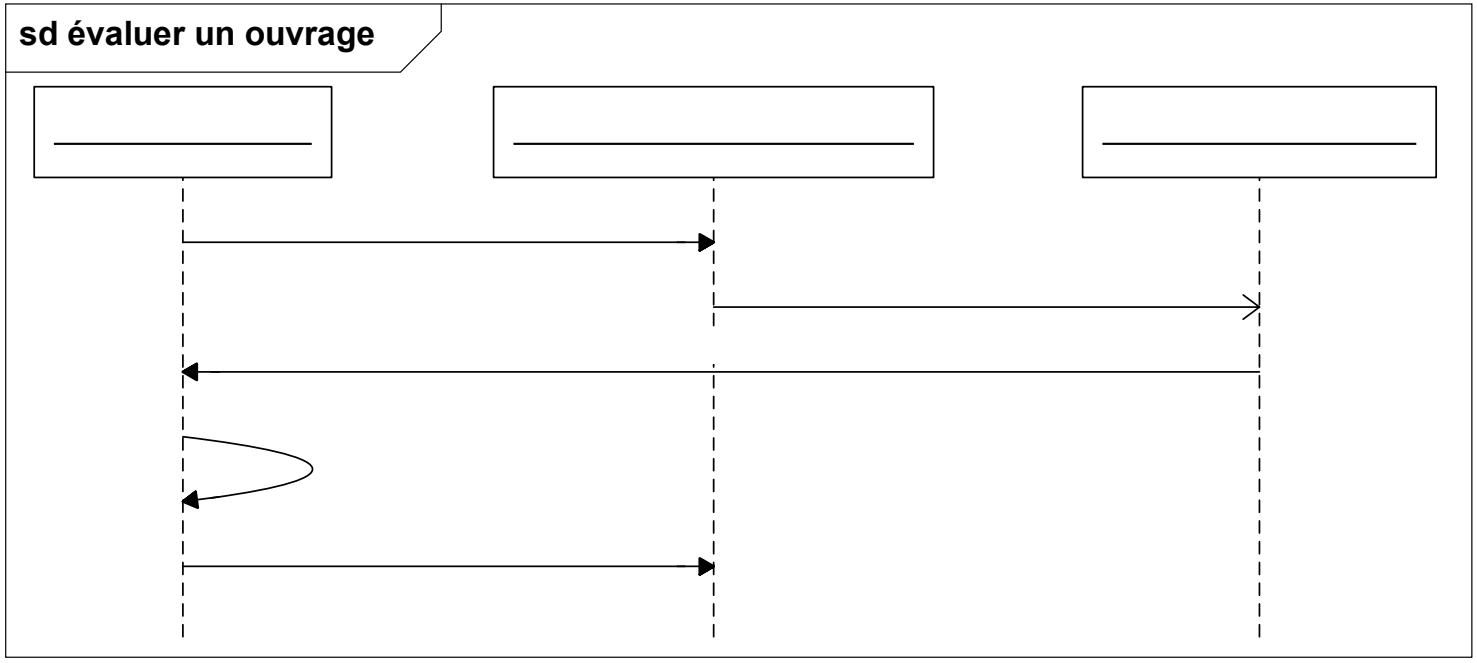
- Décrit le comportement d'un scénario
- Contient des objets et les messages transmis entre ces objets dans le cadre d'un cas d'utilisation
- Indique explicitement le temps
- Adapté aux systèmes temps réel et aux scénarios complexes



Message synchrone – asynchrone

- Si un appelant émet un message synchrone, il doit attendre que celui-ci ait terminé son travail
  - Notation 
  - Exemple : appel d'une routine
- S'il émet un message asynchrone, il peut continuer son traitement sans attendre la réponse
  - Notation 

Exemple : évaluer un ouvrage

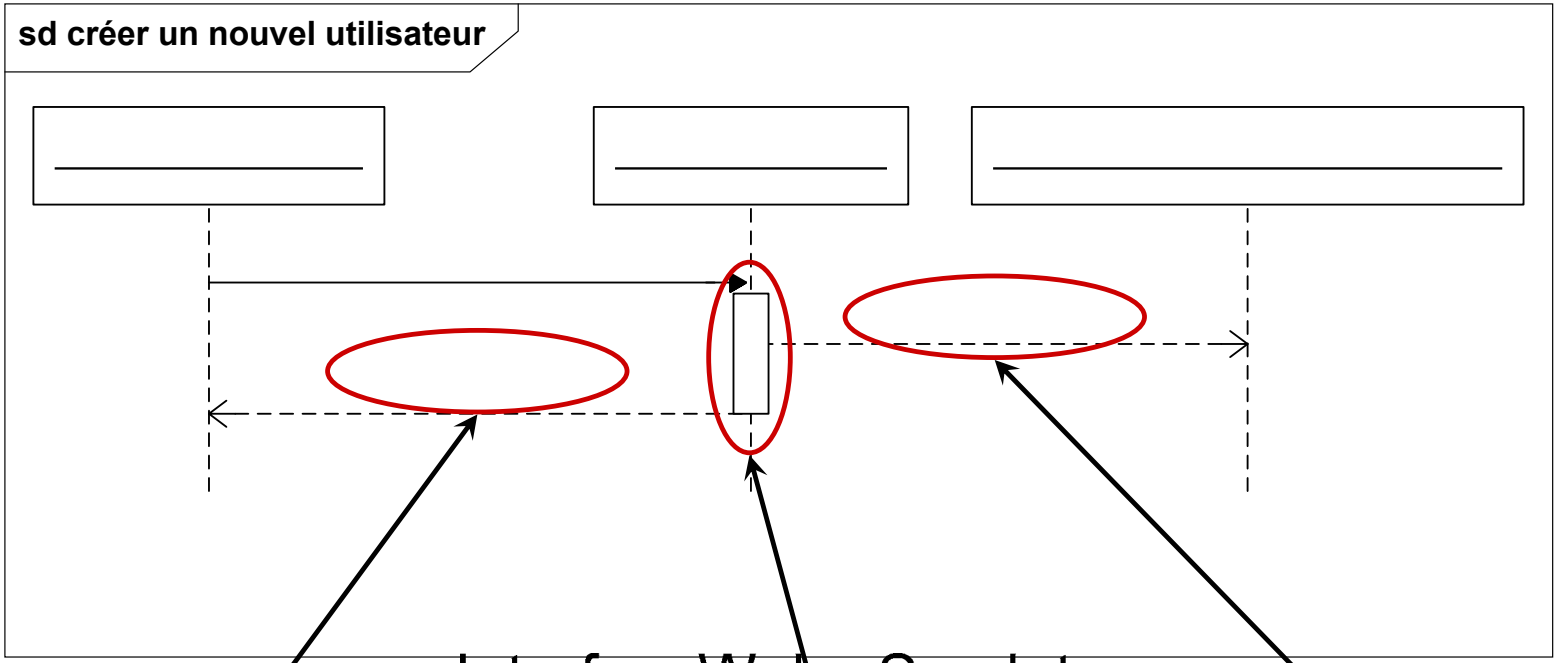


Evaluateur : Lecteur

Amazon.

commanderOuvrage("Conception

Exemple : créer un nouvel utilisateur



Quelques compléments

- D'autres éléments complètent la syntaxe des diagrammes d'interaction (cf doc de référence) :
  - Invariants d'état
  - Occurrences événementielles
  - Traces
  - Fragments combinés
  - Occurrences d'interaction
  - Décomposition
  - Continuations
  - Chronométrage de séquence

Exemple de diagramme d'interaction

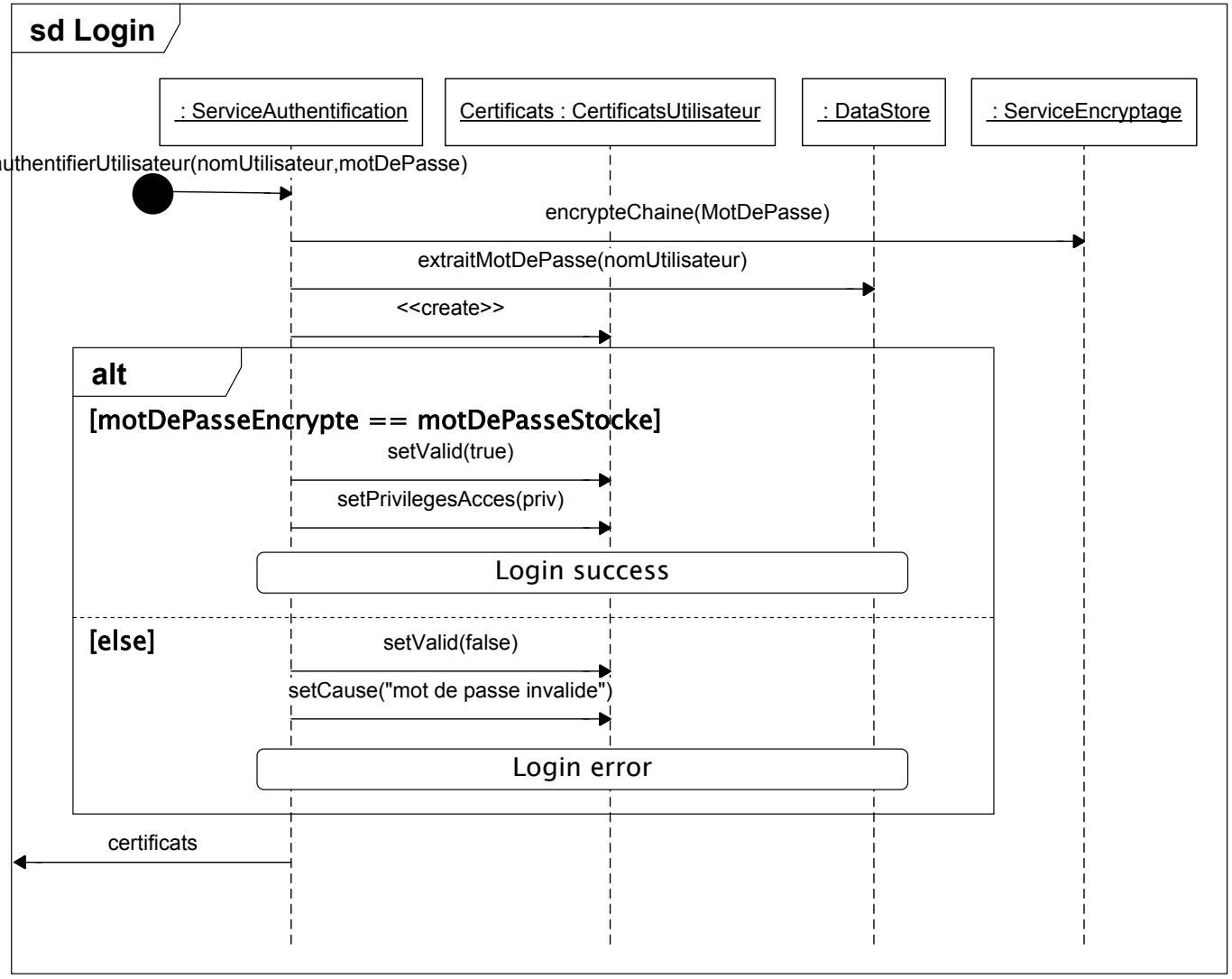
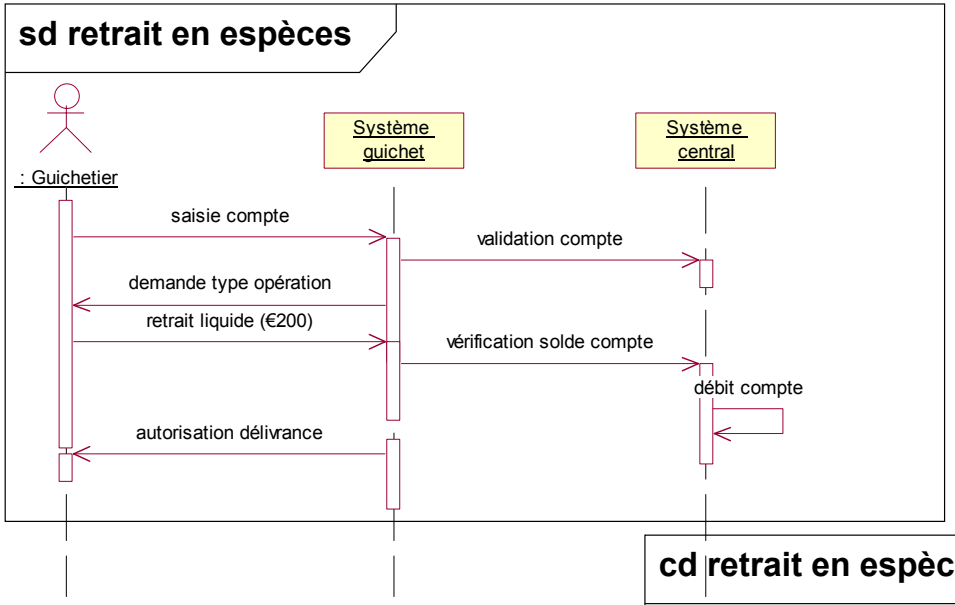
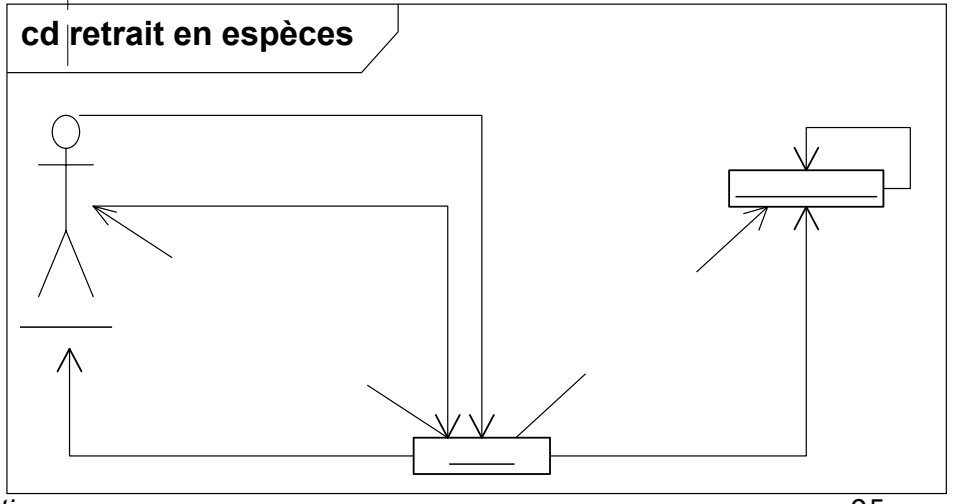


Diagramme de communications

- Diagramme de séquence et diagramme de communications





- Etude de cas guidée :
  - Guichet Automatique de Banque
- Exercice libre :
  - Centre de Documentation et d'Information

Étude de cas guidée : guichet automatique de banque

- Un système (simplifié) de guichet automatique de banque permet :
  - La distribution d'argent à tout porteur de carte de crédit (carte Visa ou carte de la banque)
  - La consultation de solde de compte, dépôt en numéraire et dépôt de chèques pour les clients de la banque porteurs d'une carte de crédit de la banque

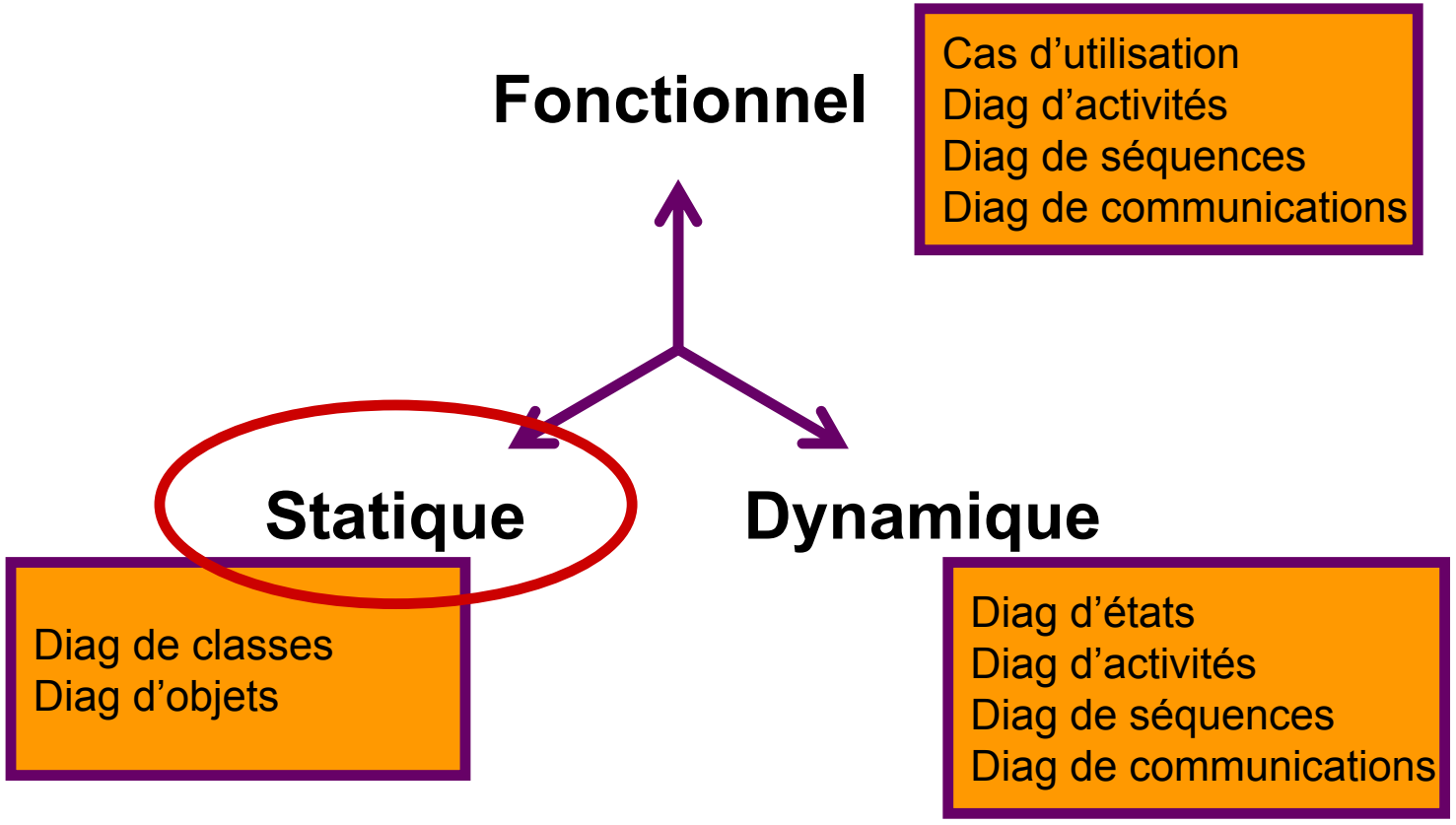
1. Identifier les acteurs
2. Identifier les cas d'utilisation
3. Construire un diagramme de cas d'utilisation
4. Décrire textuellement les cas d'utilisation
5. Compléter les descriptions par un diagramme dynamique (diagramme de séquences)
6. Organiser et structurer les cas d'utilisation

Exercice libre : CDI

- On souhaite modéliser avec UML le système « Centre de Documentation et d'Information » de Centrale Lille.

1. Identifier les acteurs
2. Identifier les cas d'utilisation
3. Construire un diagramme de cas d'utilisation
4. Décrire textuellement le cas d'utilisation « emprunter un ouvrage »
5. Compléter la description par un diagramme dynamique

…et les diagrammes étudiés



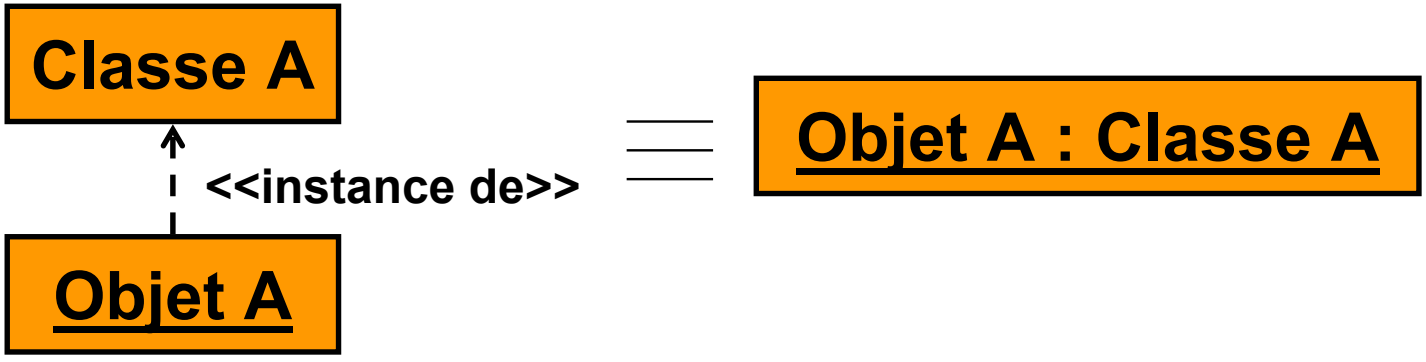
Rôle du diagramme de classes

- Identifier et commencer à organiser les données
- Construit à partir des spécifications et du diagramme des cas d'utilisation
- Prépare la conception en vue d'une implémentation informatique

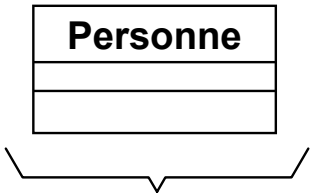
- La modélisation des données est basée sur le modèle entité-association
  - Entité
    - Informations intéressantes pour le domaine
    - Et pour lesquelles vous voulez mémoriser des données
    - Décrite avec un nom au singulier
    - Contient des attributs
  - Association
    - Pour établir un lien sémantique entre entités
    - Décrite par un groupe verbal
    - Peut être orientée

Représentation d'une classe

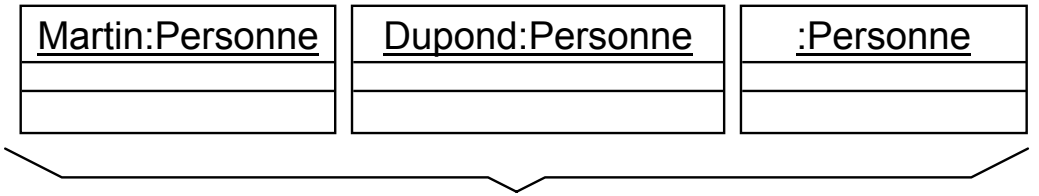
- Démarche :
  - Identification des caractéristiques communes à un ensemble d'éléments
  - Description condensée de ces caractéristiques
- Dépend du point de vue
- Représentation :



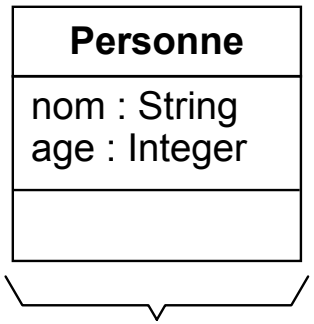
- Un objet est une instance d'une classe
- Une classe est un classificateur d'un objet



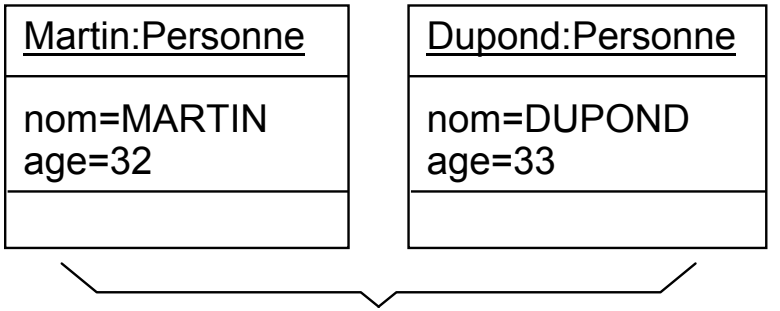
Classe



Objets

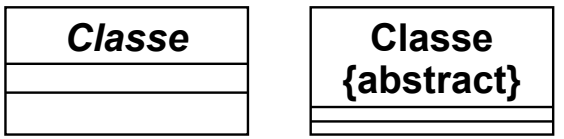


Classe avec attributs



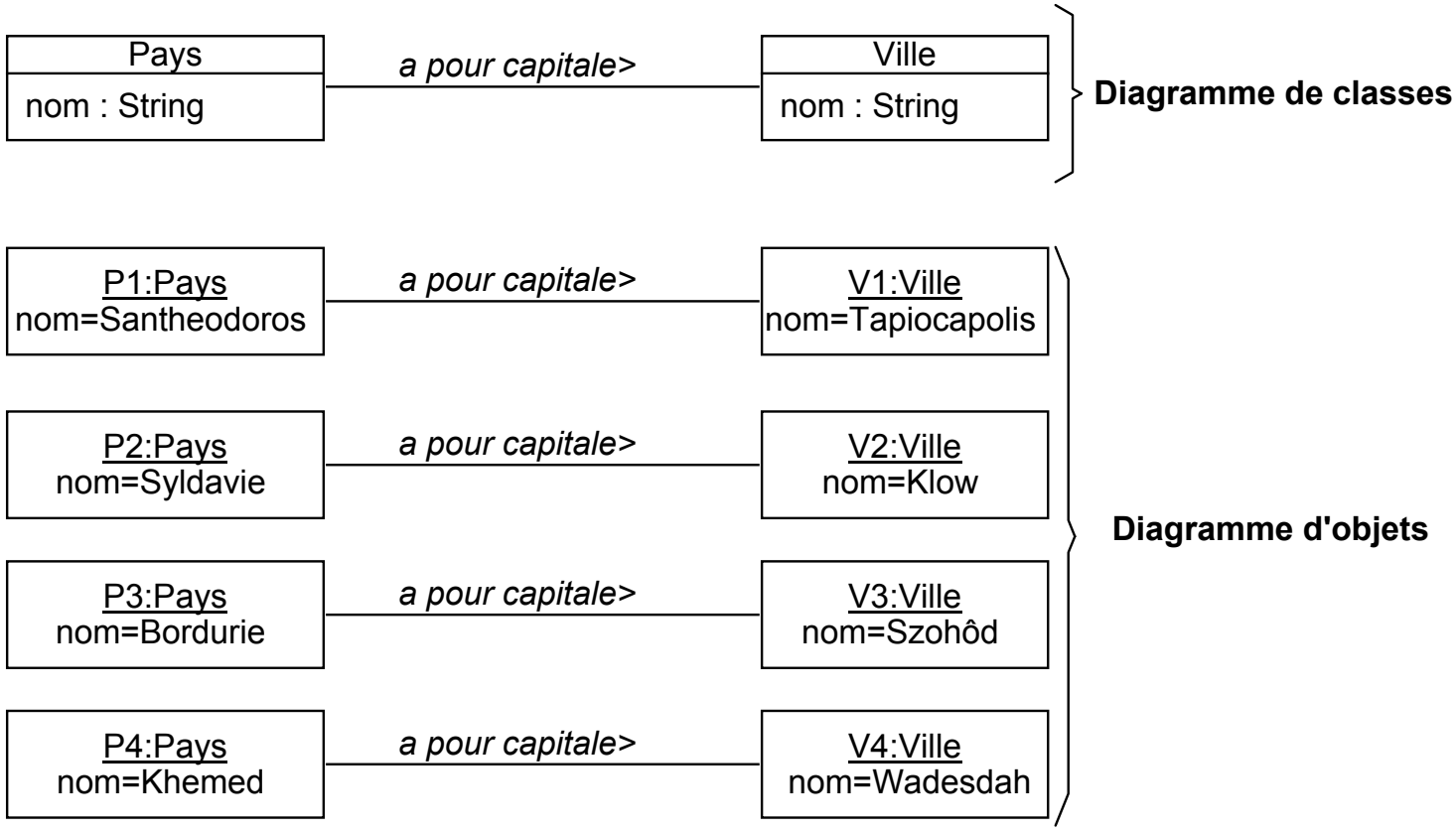
Objets avec valeurs

Classe abstraite



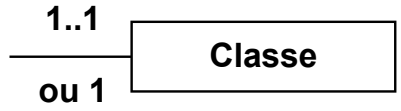
Liens et associations

- Liens : entre objets
- Associations : entre Classes

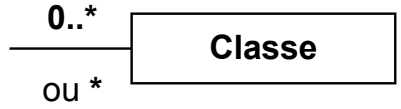


Multiplicités et contraintes

- Multiplicité : Contraintes valables durant toute l'existence des objets
  - une et une seule



  - plusieurs (0 ou plus)



  - optionnelle (0 ou 1)

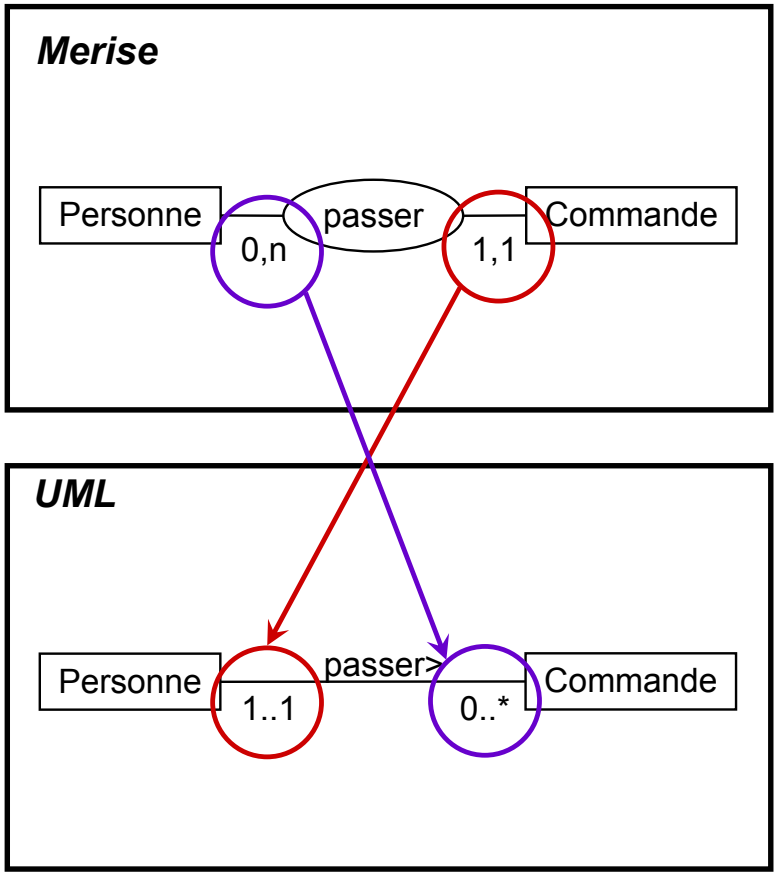
  - 1 ou plus

  - de m à n

- Autres contraintes sur association
  - {ordered},{subset},{xor},

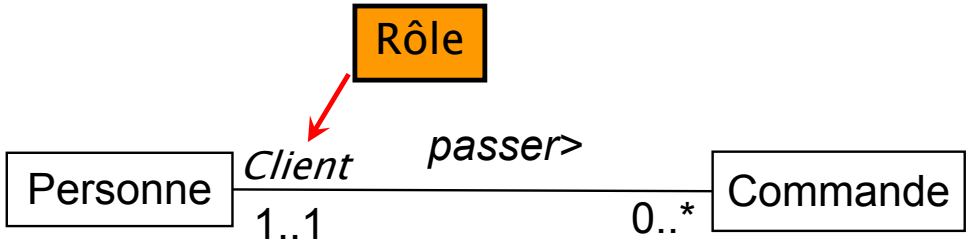
Comparaison des formalismes UML et Merise

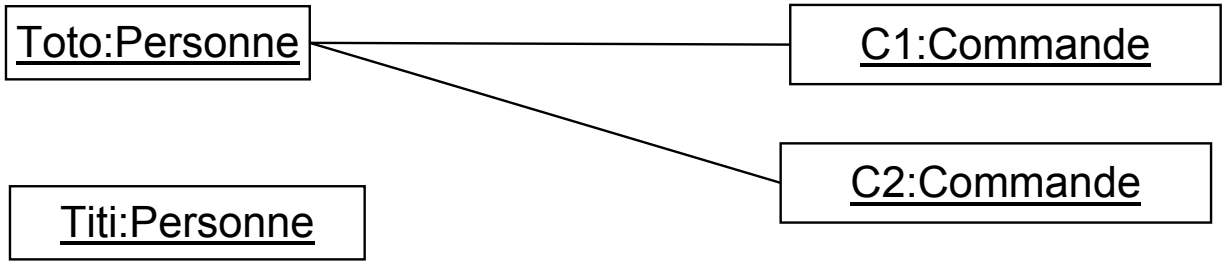
- Cardinalités (Merise), multiplicités (UML)



Exemple : commande

- Une personne peut passer plusieurs commandes
- Une commande est toujours passée par une et une seule personne





Exercice : gestion d'un magasin

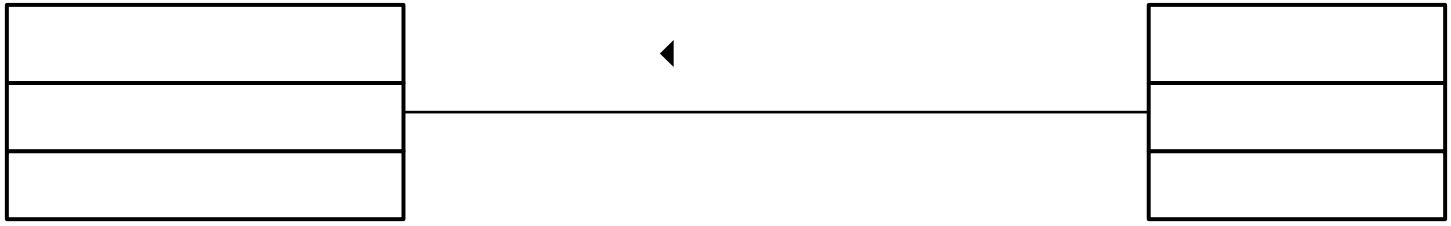
- Spécification :
  - Notre magasin stocke des lignes de produits
  - Chaque ligne de produits contient des articles
  - Chaque article correspond à une ligne de produits

- Réaliser le diagramme de classes correspondant

Exercice : gestion d'un magasin

- Spécification :
  - Notre magasin stocke des lignes de produits
  - Chaque ligne de produits contient des articles
  - Chaque article correspond à une ligne de produits

- Réaliser le diagramme de classes correspondant



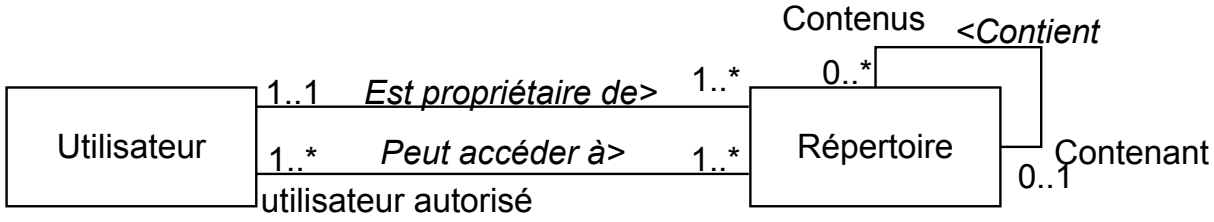
Il n'y a pas d'entité Magasin
car on ne gère pas des magasins
(il n'y en a qu'un)

Exercice : système de fichiers

- Spécification :
  - Un utilisateur possède au moins un répertoire
  - Un répertoire appartient à un et un seul utilisateur
  - Un répertoire peut contenir d 'autres répertoires
  - Un utilisateur peut accéder à au moins un répertoire
  - Un répertoire peut être accédé par au moins un utilisateur
- Réaliser le diagramme de classes correspondant

Exercice : système de fichiers

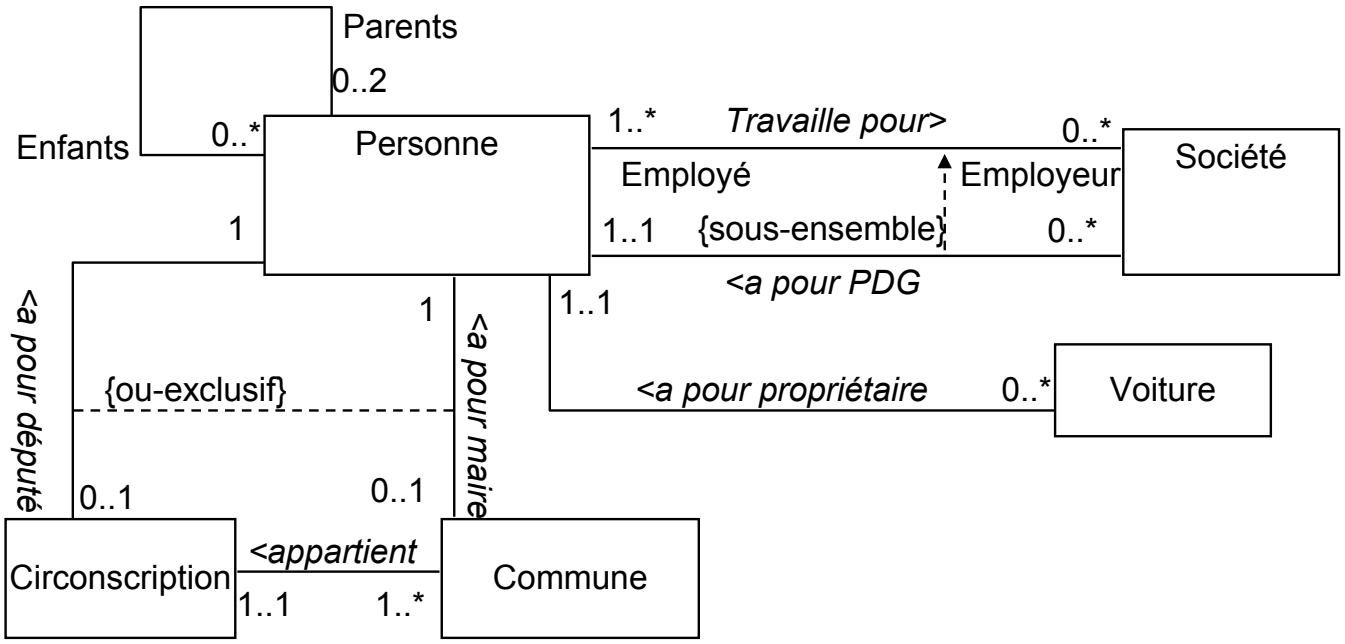
- Spécification :
  - Un utilisateur possède au moins un répertoire
  - Un répertoire appartient à un et un seul utilisateur
  - Un répertoire peut contenir d 'autres répertoires
  - Un utilisateur peut accéder à au moins un répertoire
  - Un répertoire peut être accédé par au moins un utilisateur
- Réaliser le diagramme de classes correspondant



Exercice : des personnes, des communes…

- Spécification :
  - Une personne peut travailler pour plusieurs sociétés
  - Une société a un et un seul PDG
  - Une personne peut être PDG de plusieurs sociétés
  - Une personne peut posséder plusieurs voitures
  - Une voiture a un seul propriétaire
  - Une personne peut avoir des enfants
  - Une personne peut être le maire d 'une commune
  - Une personne peut être le député d 'une circonscription
  - Une commune appartient à une circonscription

Exercice : des personnes, des communes…



Exercice : des pays et des langues

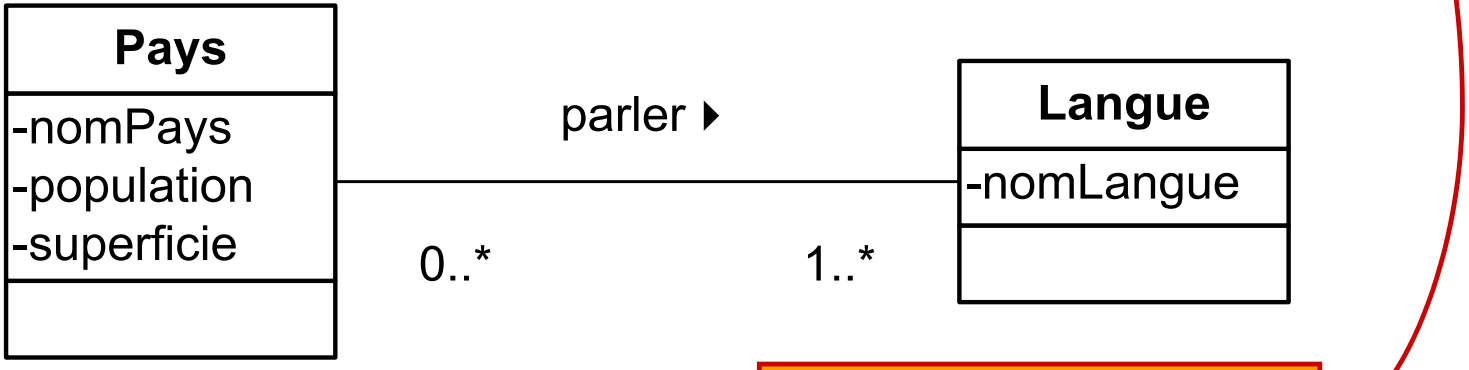
- Spécifications :
  - Un pays a un nom, un nombre d'habitants et une superficie
  - Dans ce pays, certaines langues sont parlées
  - Un pourcentage de la population parle une langue ; cette langue peut être officielle
- Réaliser le diagramme de classes correspondant

Exercice : des pays et des langues

- Spécifications :
  - Un pays a un nom, un nombre d'habitants et une superficie
  - Dans ce pays, certaines langues sont parlées
  - Un pourcentage de la population parle une langue ; cette langue peut être officielle
- Réaliser le diagramme de classes correspondant

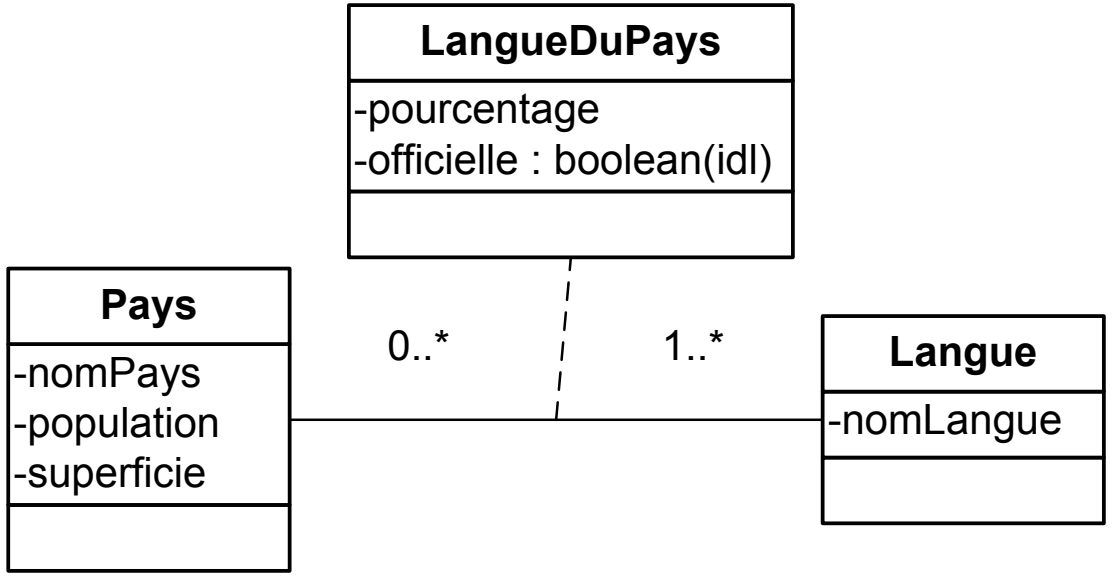
Exercice : des pays et des langues

- Spécifications :
  - Un pays a un nom, un nombre d'habitants et une superficie
  - Dans ce pays, certaines langues sont parlées
  - Un pourcentage de la population parle une langue ; cette langue peut être officielle
- Réaliser le diagramme de classes correspondant



Exercice : des pays et des langues

- Spécifications :
  - Un pays a un nom, un nombre d'habitants et une superficie
  - Dans ce pays, certaines langues sont parlées
  - Un pourcentage de la population parle une langue ; cette langue peut être officielle
- Réaliser le diagramme de classes correspondant



Comment construire un diagramme de classes

- Lister les noms
  - Un substantif (mot du domaine) est un candidat potentiel pour
    - Une entité
    - Un attribut
  - Supprimer :
    - Synonymes
    - Polysèmes
    - Noms superflus (inutiles ou hors domaine)
  - Procéder par itérations
- Identifier les classes
  - Regroupent des attributs

Comment construire un diagramme de classes

- Identifier les associations :
  - Verbes
- Identifier les classes interdépendantes
- Répartir les attributs dans les classes
- Indiquer les multiplicités
- Trouver les opérations

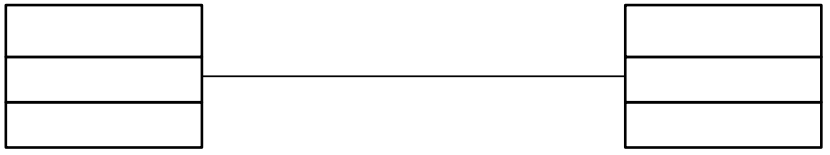
Exemple : magasin

- Le magasin doit être entièrement automatisé et les opérations journalières ne doivent pas nécessiter d'assistance humaine
- La quantité en stock sera de 500 articles maxi, chaque article étant identifié par une référence

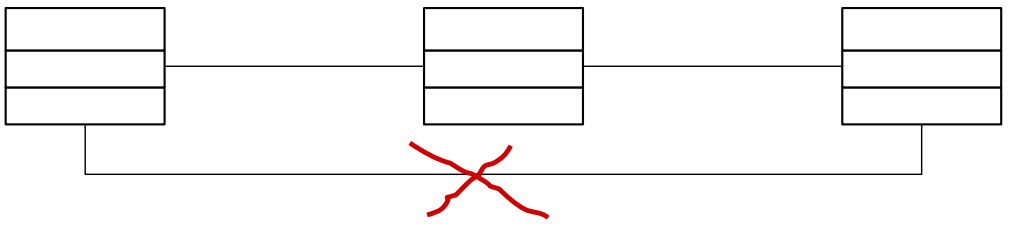
Magasin	*entité inutile pour le problème*
Opérations	*traitements*
Assistance humaine	*sans intérêt*
Quantité en stock	*attribut*
Article	*entité*
Référence	*attribut*

Quelques questions

- Faut-il fusionner les classes ?



- Y a-t-il des boucles ?



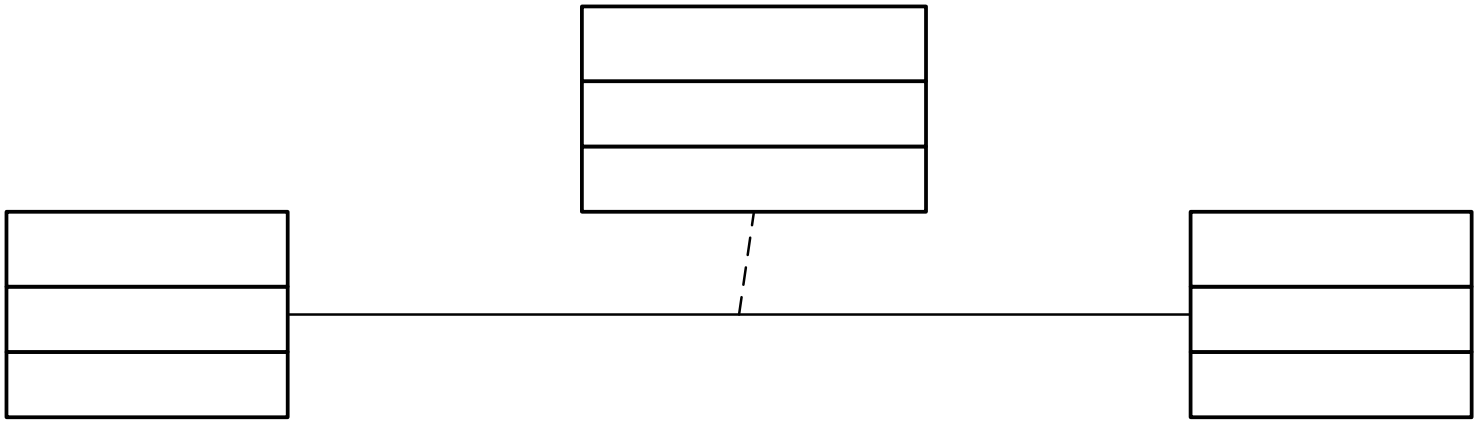
1..1

Attention : erreur classique !

- Par défaut, 2 instances ne peuvent être reliées au maximum qu'une seule fois par une association
  - Quand on parcourt une association à partir d'une instance, on obtient par défaut un ensemble au sens mathématique (donc sans doublon)
  - Formulation base de données : l'identifiant d'une association est constitué de la concaténation des identifiants des entités associées

Exemple : réservation restaurant

- Quel est le défaut du modèle suivant ?



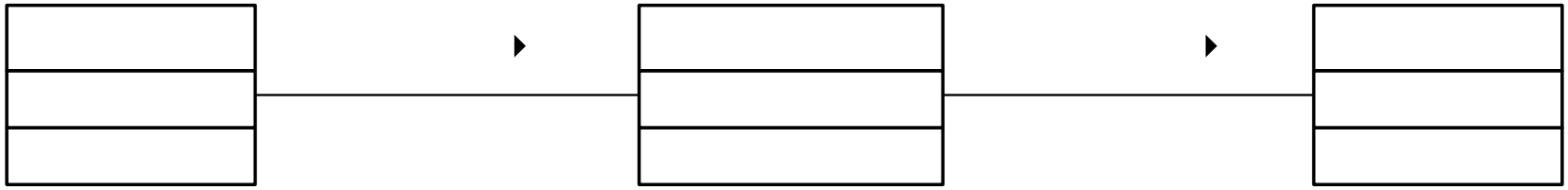
- Comment y remédier ?

Personne

0..*

Exemple : réservation restaurant

- Une solution :



- Remarque : il existe une autre solution en UML 2, mais elle est imbuvable

Personne

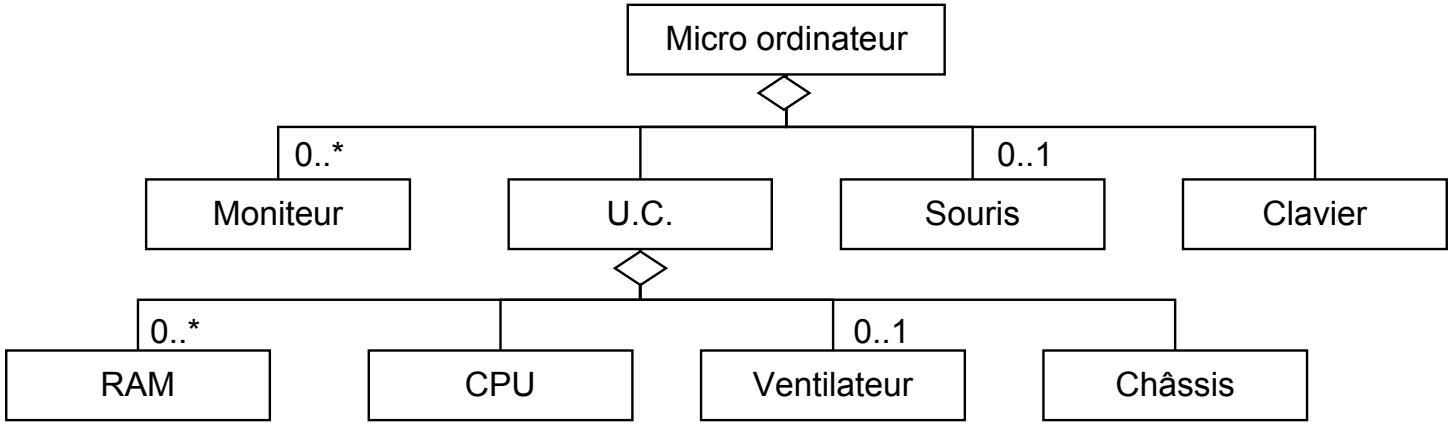
effectuer

-d

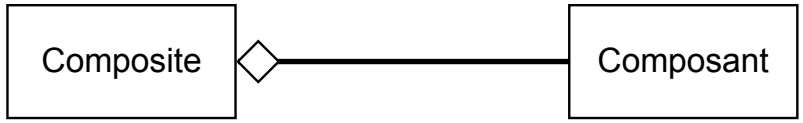
1..1

0..*

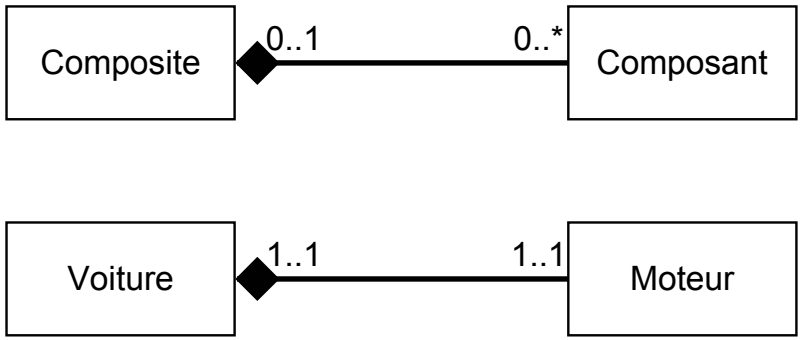
- Association avec une sémantique spécifique
  - Relation "made of" "part of"
  - Transitive
  - Antisymétrique
  - Propagation des propriétés avec modifications locales éventuelles



- Agrégation
  - Relation binaire de type tout-parties
  - Relation d'appartenance faible : le composant peut être partagé entre plusieurs composites : préciser la multiplicité



- Composition
  - Agrégation avec relation d'appartenance forte et coïncidence des durées de vie
  - Les composants peuvent être créés après le composite, mais après création ils ont la même durée de vie
  - Les composants peuvent être enlevés avant la mort du composite
  - Chaque composant appartient à, au plus, un composite
  - Propagation composite vers composant

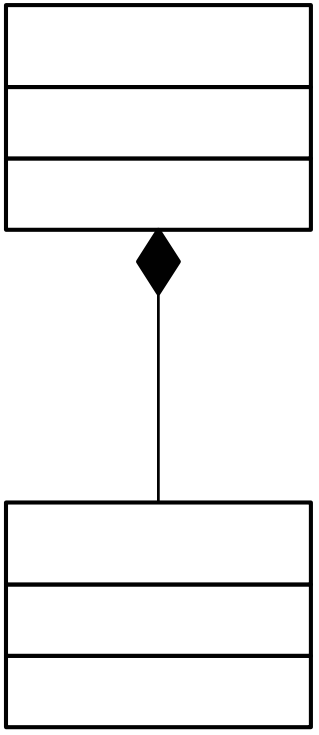


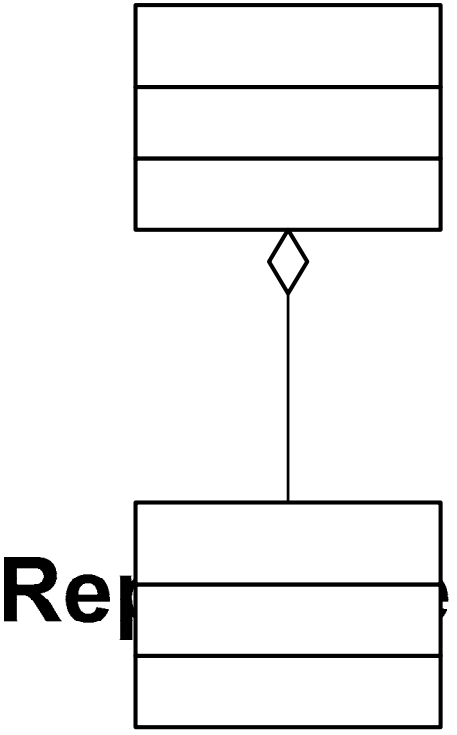
Nota : UML 2 introduit d'autres notations de la composition

Exemples

- Un répertoire contient des fichiers
- Une pièce contient des murs

- Un répertoire contient des fichiers
- Une pièce contient des murs

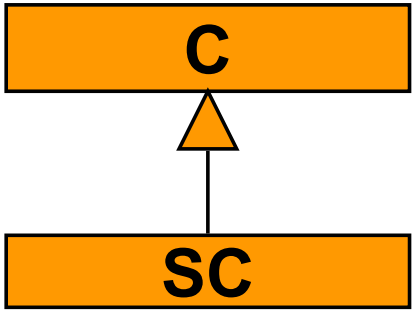




- Généralisation
  - Au niveau des concepts
  - Relation "is a" ou "a kind of" (Classification)
  - Généralisation/Spécialisation
  - Relation entre une classe (super-classe) et une ou plusieurs spécialisations (sous-classes)
  - Une instance d'une sous-classe est également une instance de la super-classe
  - Classe concrète et classe abstraite

- Héritage
  - Au niveau de la mise en œuvre (programmation OO)
  - N'est pas la seule technique d'implantation d'une généralisation
  - Héritage : Relation transitive sur un nombre arbitraire de niveaux
  - Relation statique à couplage fort

Hiérarchie de classes et héritage

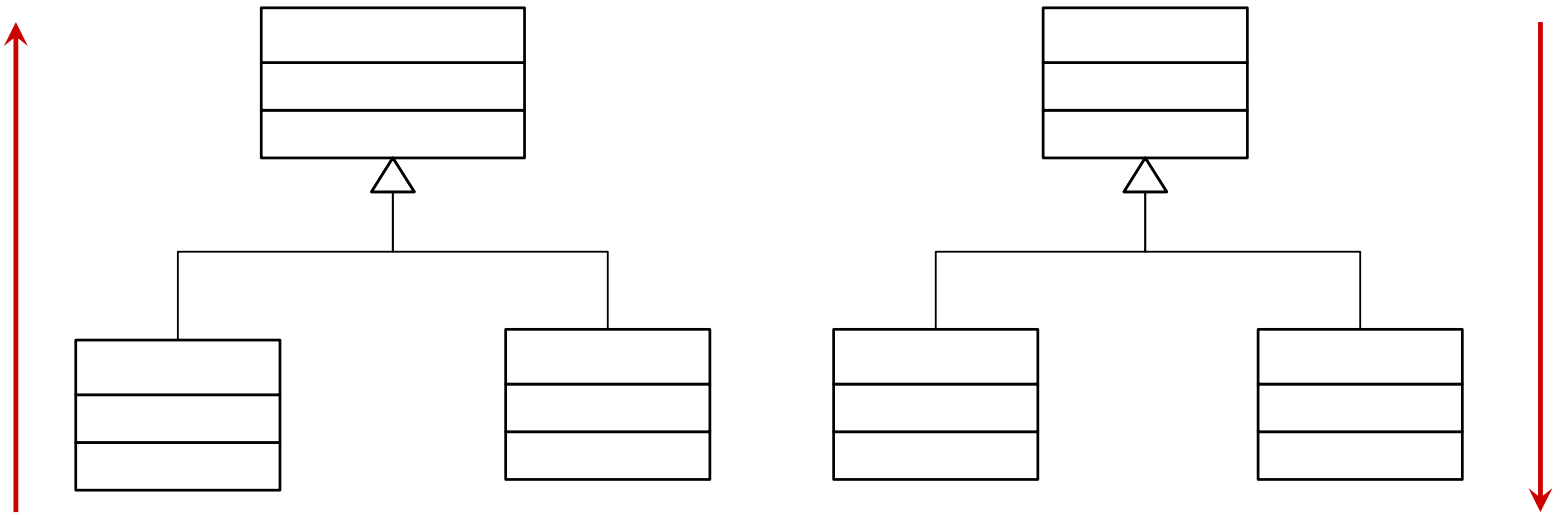


- Les objets de SC héritent des attributs et méthodes de C
- On peut redéfinir dans SC une méthode de C (surcharge)

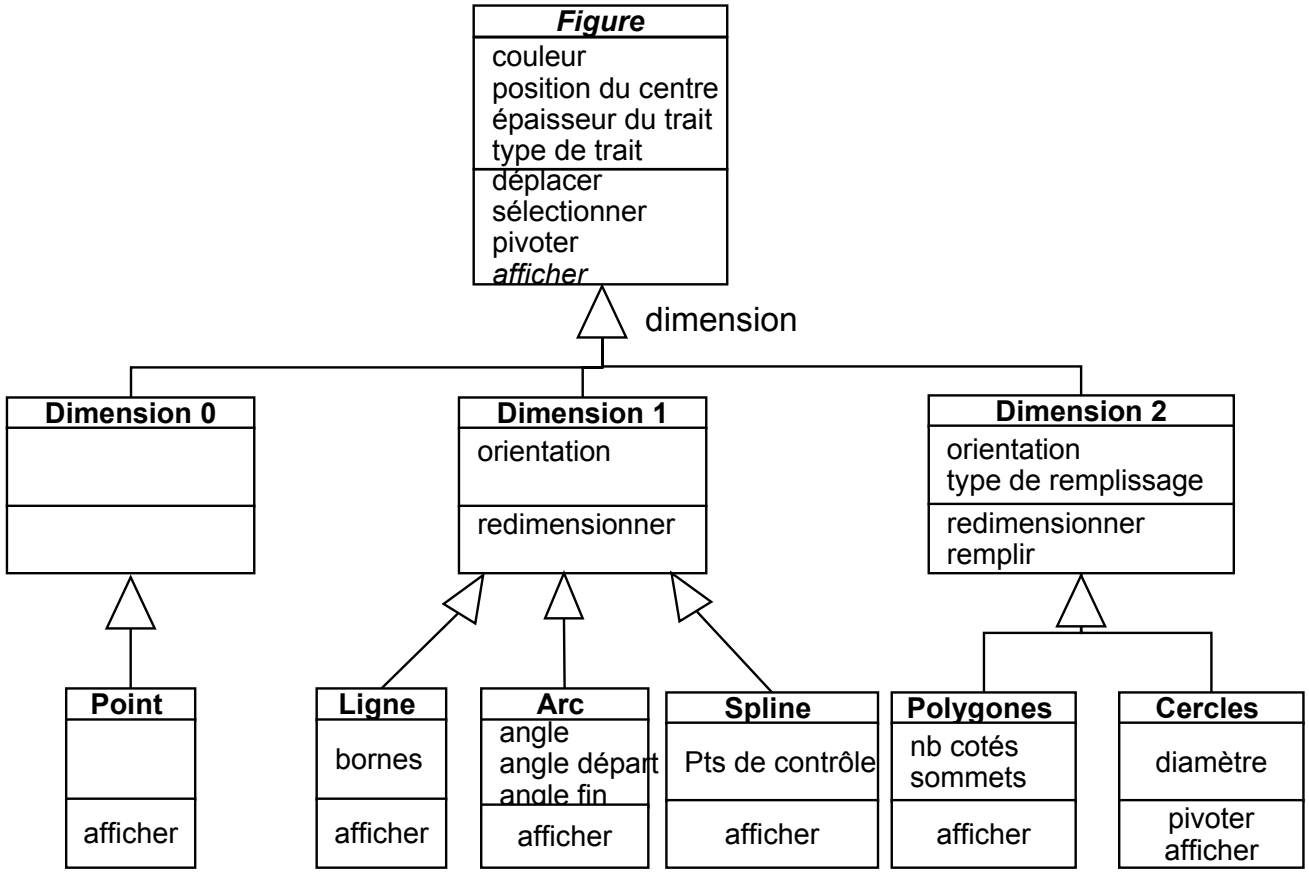
Exemples

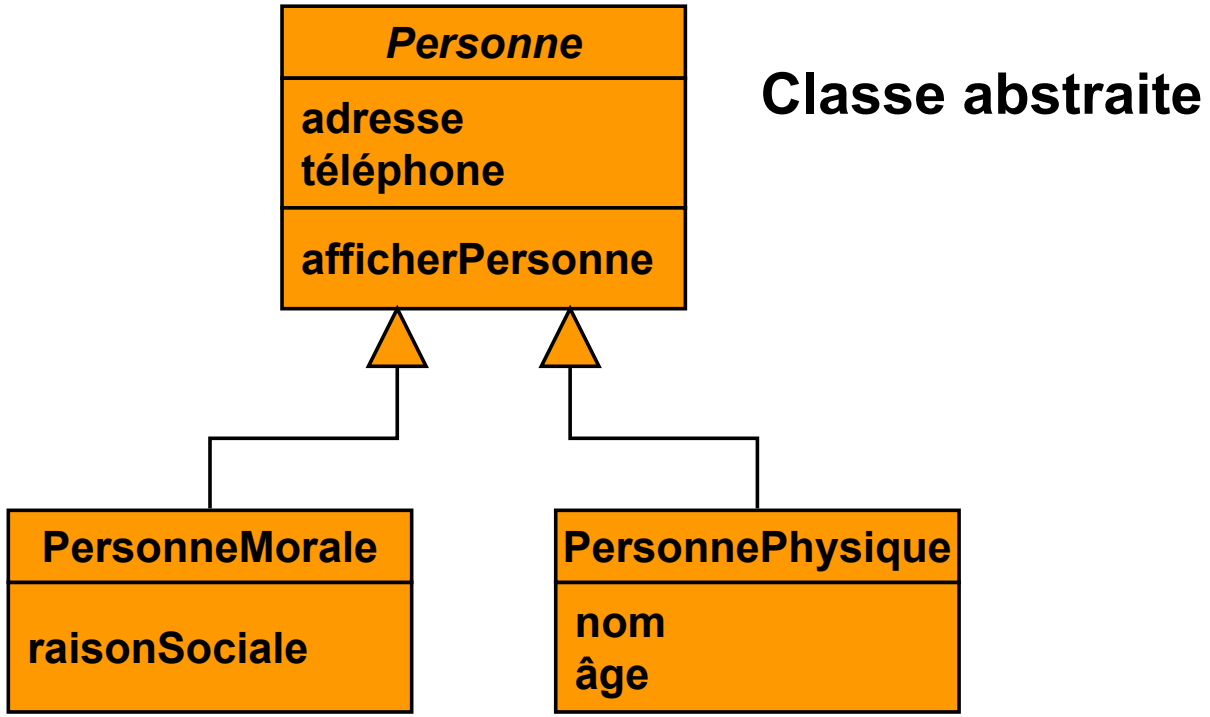
- Les modems et les claviers sont des périphériques d'entrée-sortie
- Une transaction boursière est un achat ou une vente

- Les modems et les claviers sont des périphériques d'entrée-sortie
- Une transaction boursière est un achat ou une vente

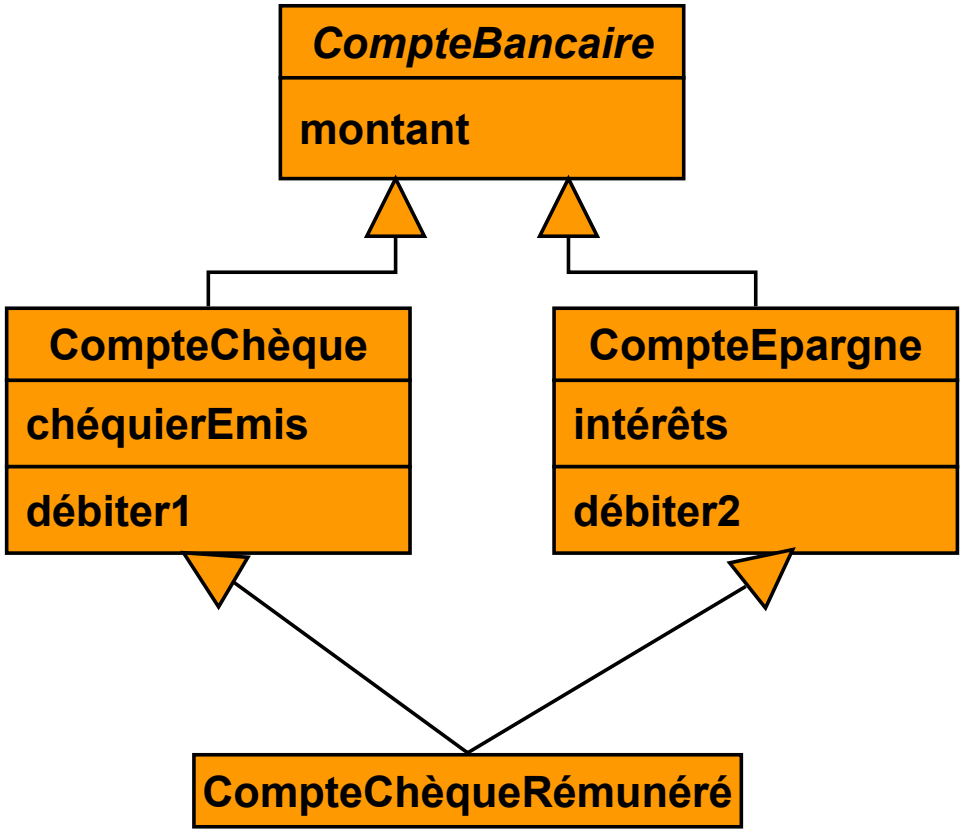


Généralisation et Héritage




Personne
adresse
téléphone
afficherPersonne
Classe abstraite
PersonneMorale
raisonSociale
PersonnePhysique
nom
âge

Pourquoi bannir l'héritage multiple



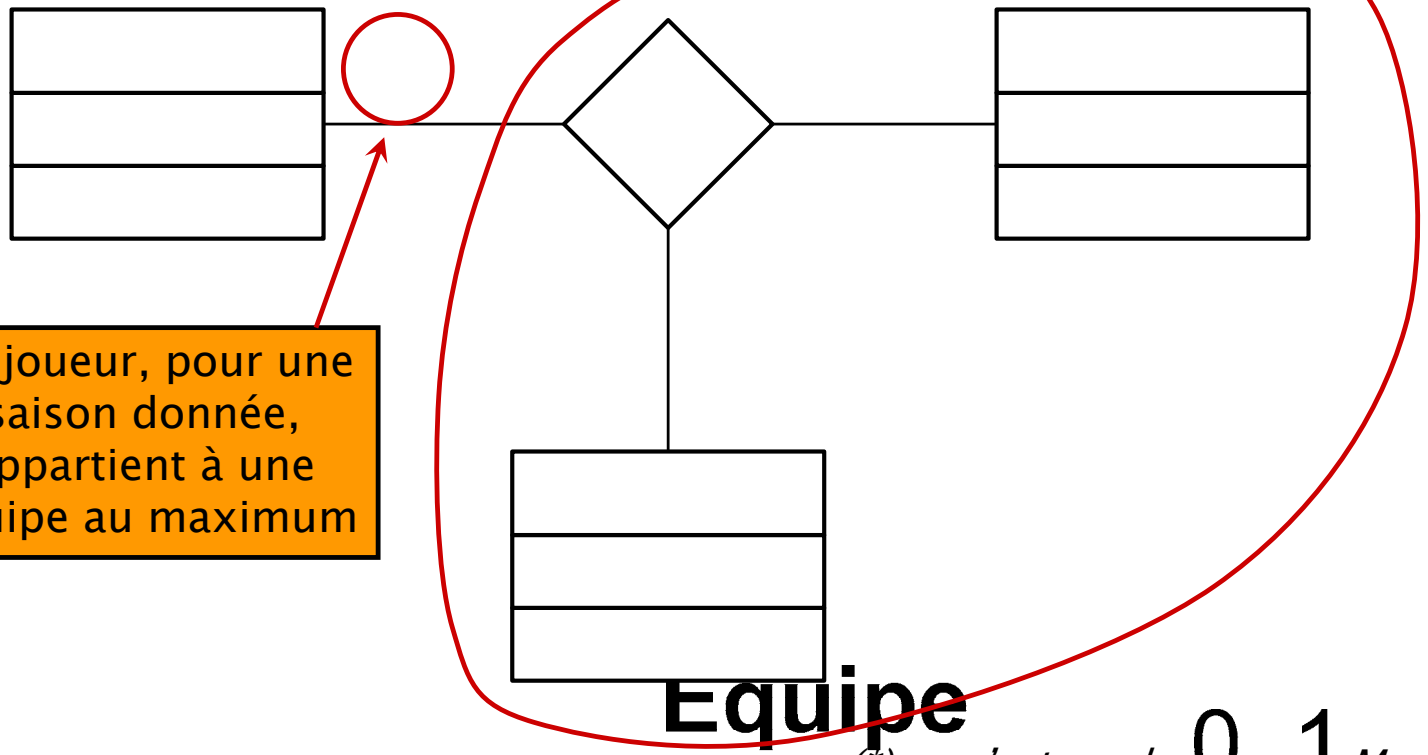
Une instance de CompteChèqueRémunéré doit-elle hériter 2 fois du montant ?
De laquelle des 2 méthodes doit hériter la classe CompteChèqueRémunéré ?

Quelques exemples

- Un compte bancaire peut appartenir à une personne physique ou morale
- Deux personnes peuvent être mariées
- Modélisez des feutres et des stylos
- Un pays a une capitale

Multiplicités des associations n-aires

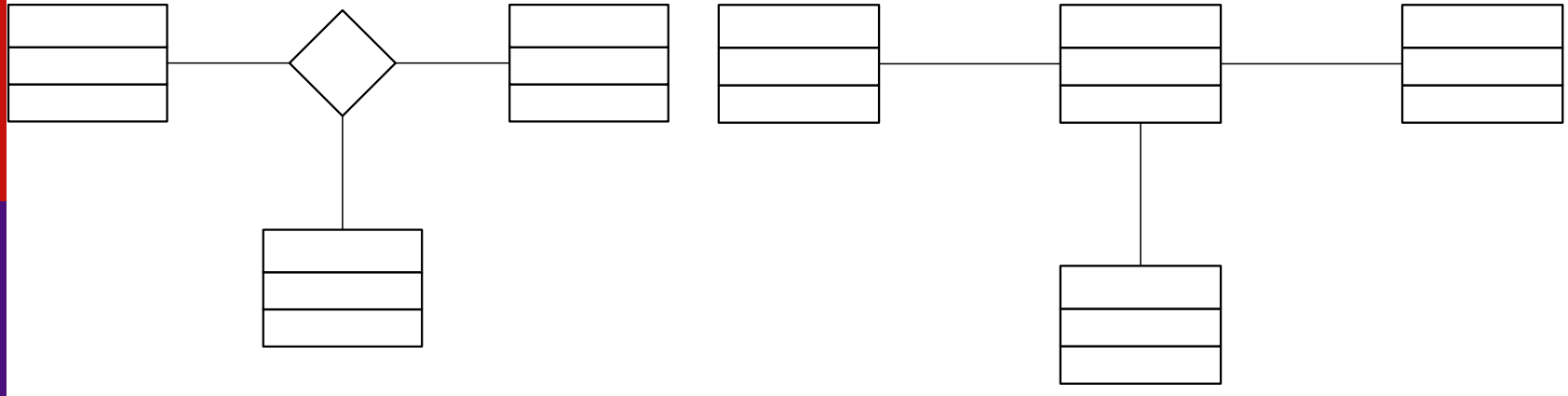
- La multiplicité d'un rôle représente le nombre potentiel de n-uplets d'instances dans l'association quand les n-1 autres valeurs sont fixées (*)



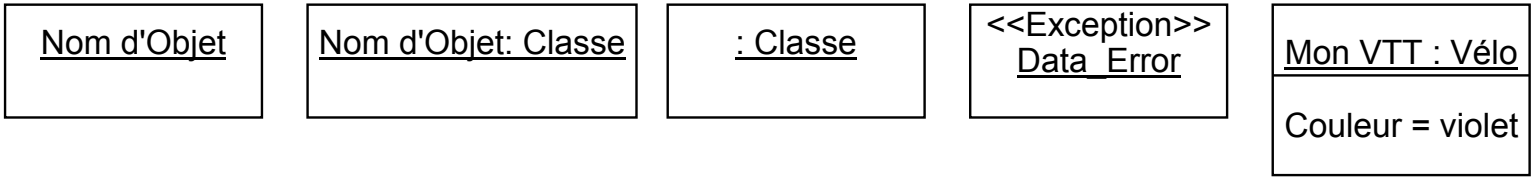
(*) ce n'est pas le cas avec Merise

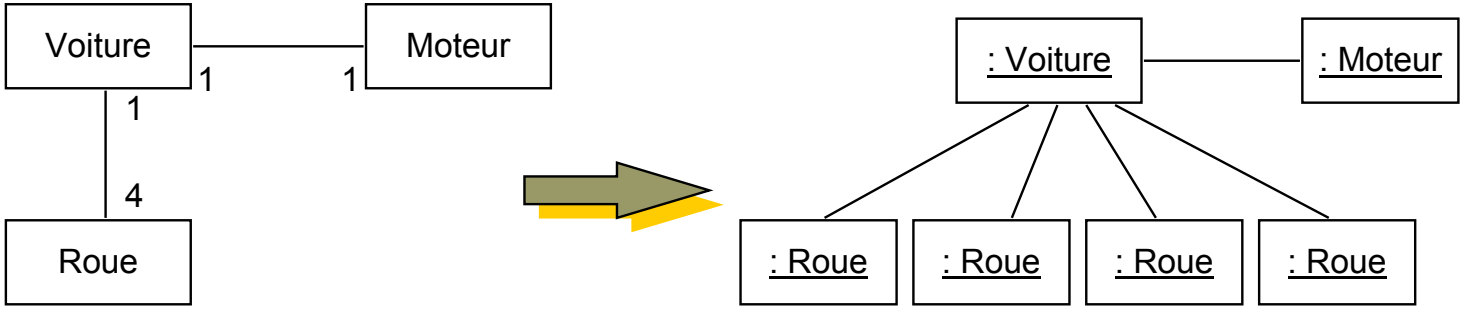
Alternative aux associations n-aires

- Association n-aire : difficile à écrire, à comprendre, à modéliser avec les outils, à coder…
- Souvent remplacée par une classe + n associations binaires, mais
  - Les multiplicités sont modifiées
  - L'unicité du n-uplet n'est pas conservée

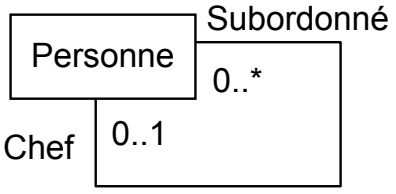


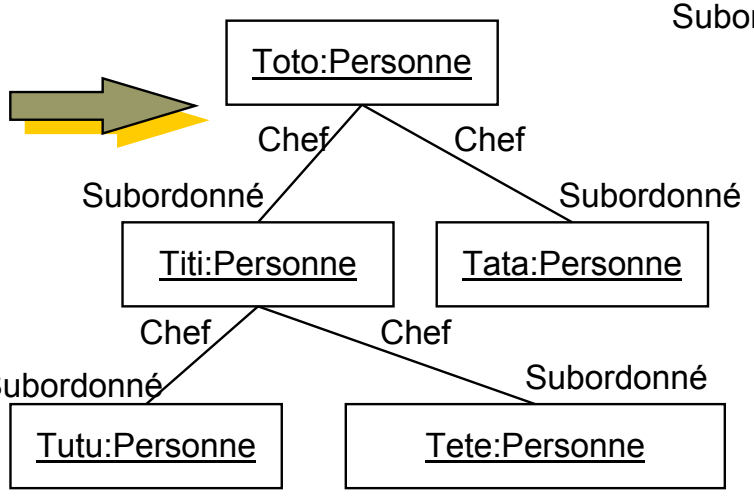
- Ou diagrammes d'instances
- Pour :
  - Montrer un contexte avant ou après une interaction
  - Faciliter la compréhension des structures complexes

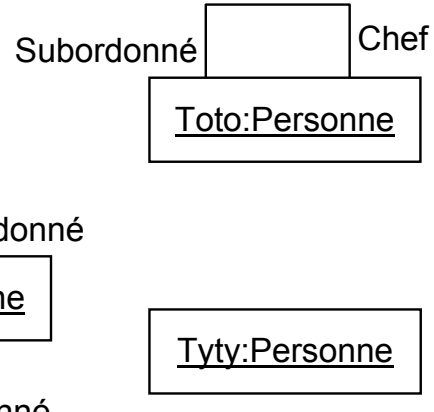




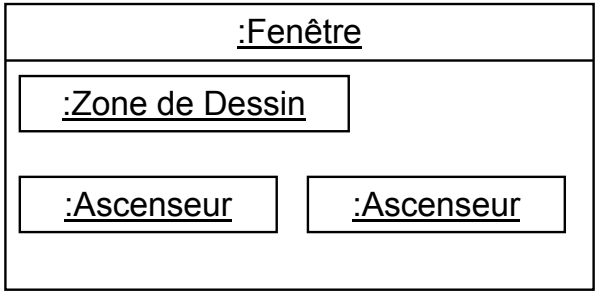
Relations réflexives





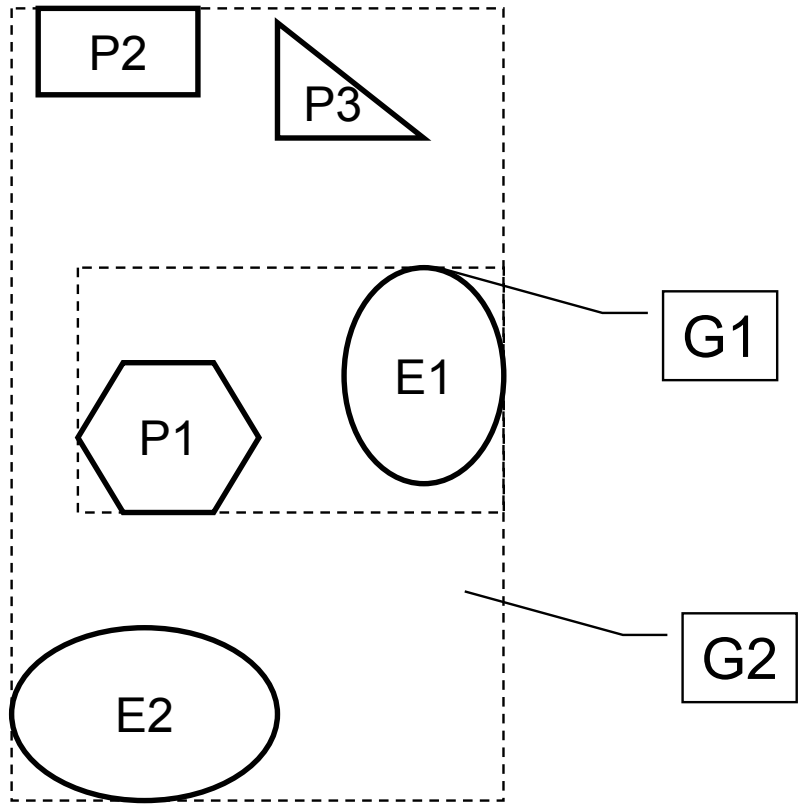


Objets composites

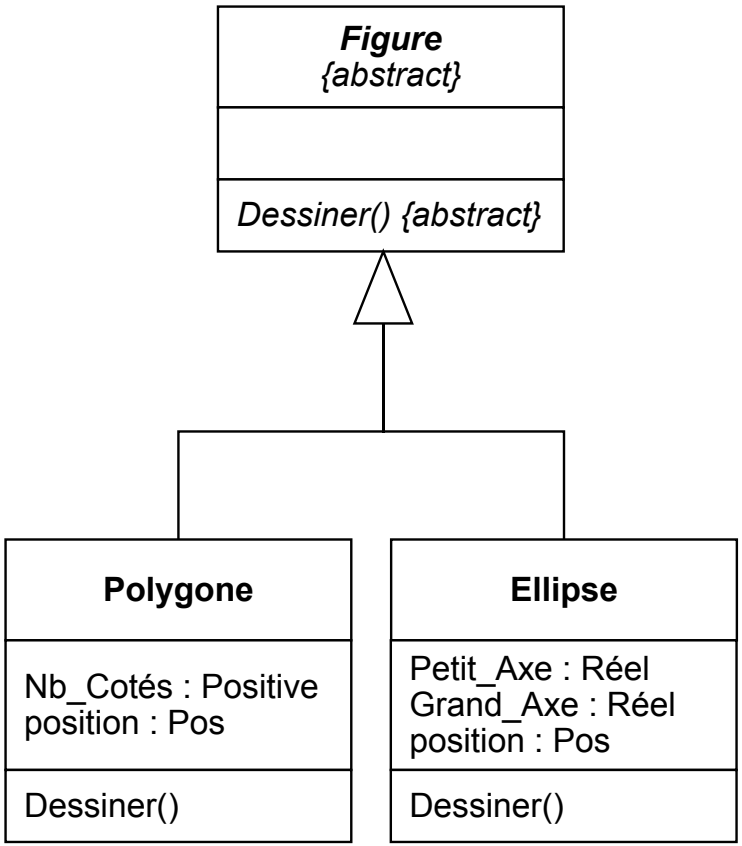


Exemple récapitulatif classes et objets

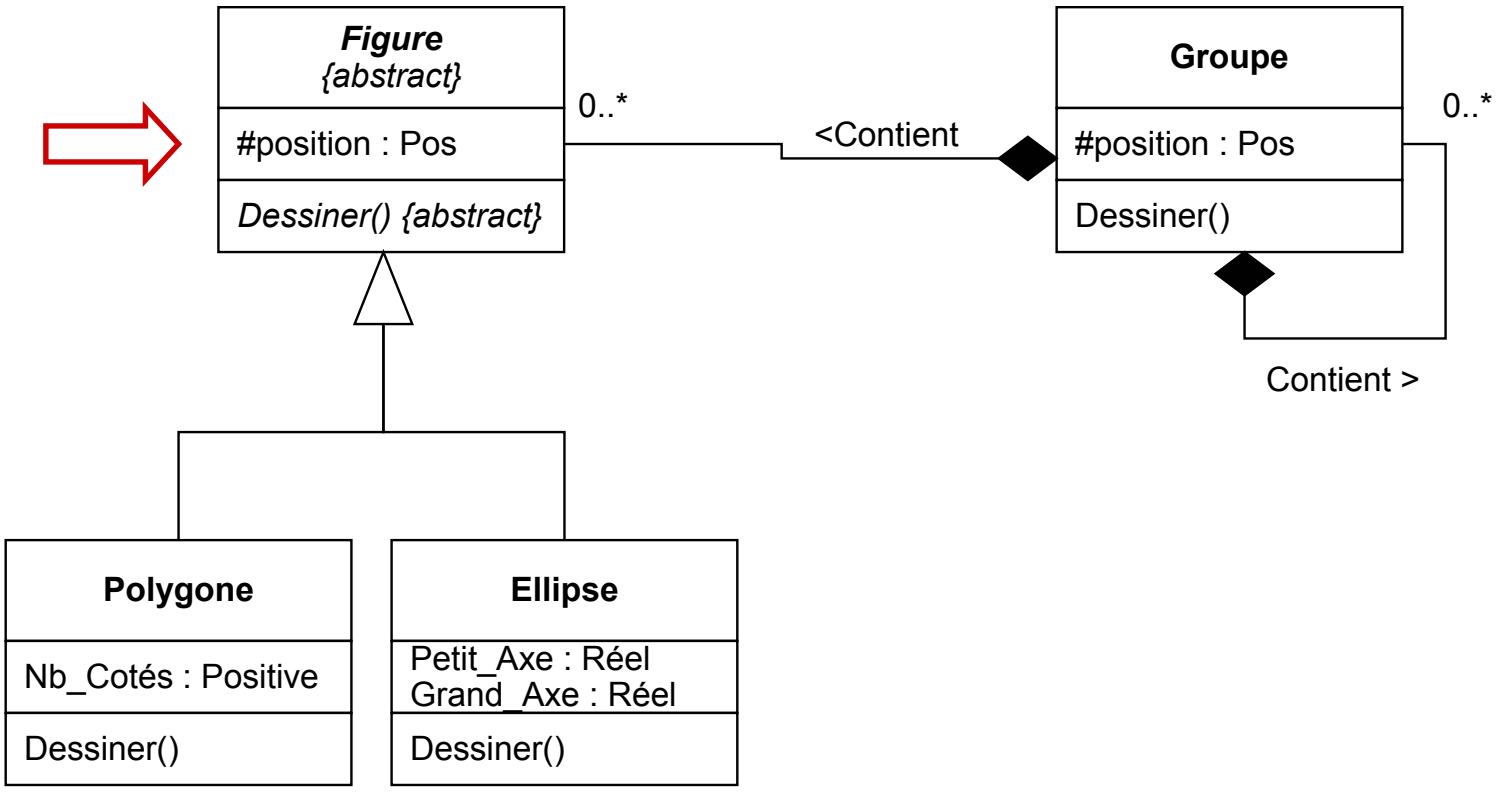
- Création de figures
  - Ellipses
  - Polygones
- Possibilités de
  - Les dessiner
  - Créer des groupes



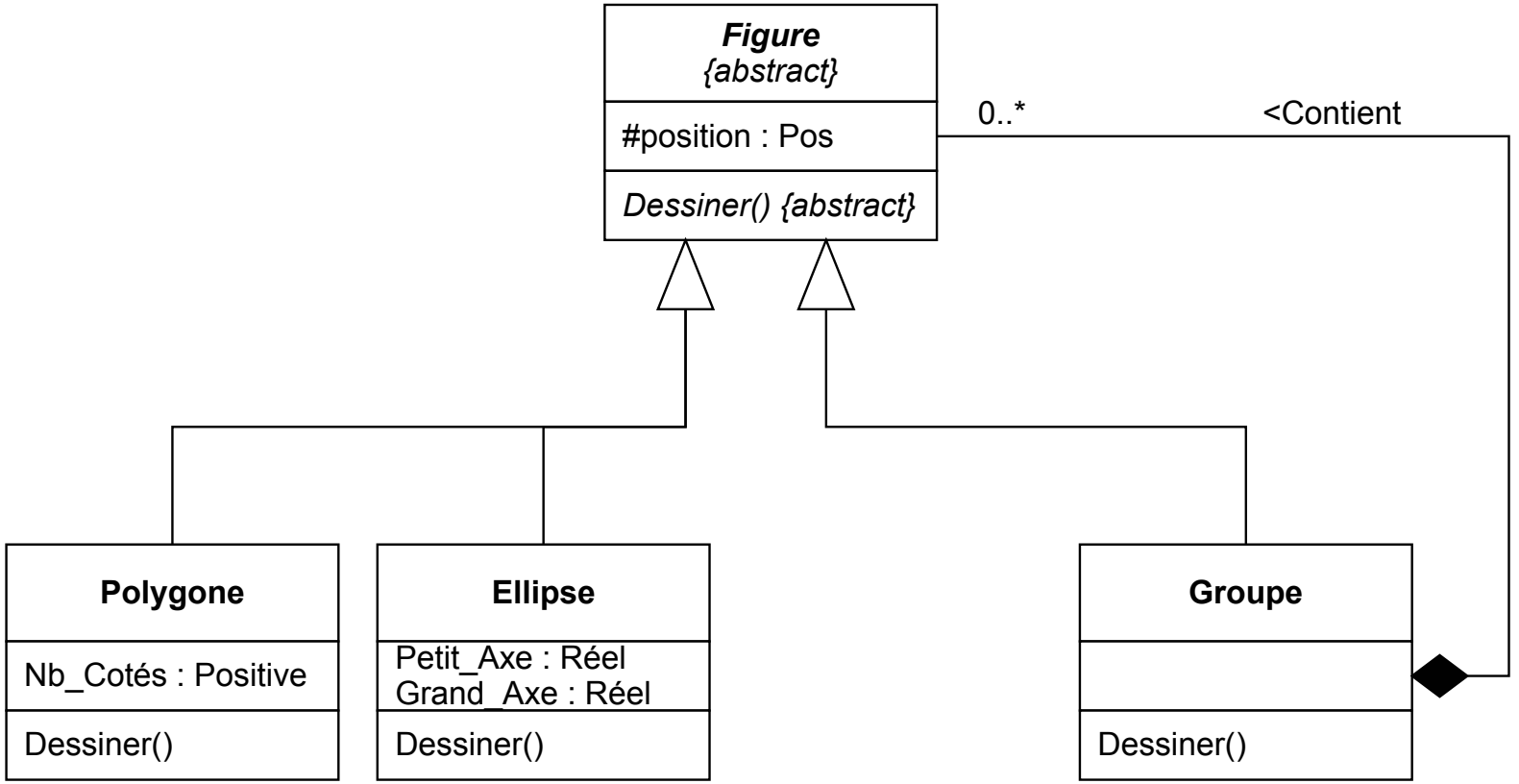
Exemple : premier diagramme de classes



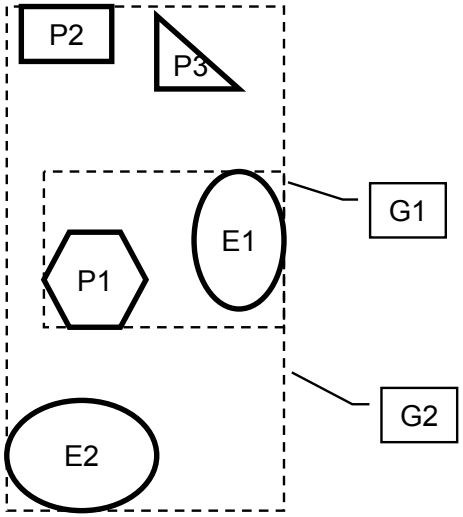
Exemple : premier diagramme de classes



Solution (nettement) améliorée



Exemple : diagramme d'instances



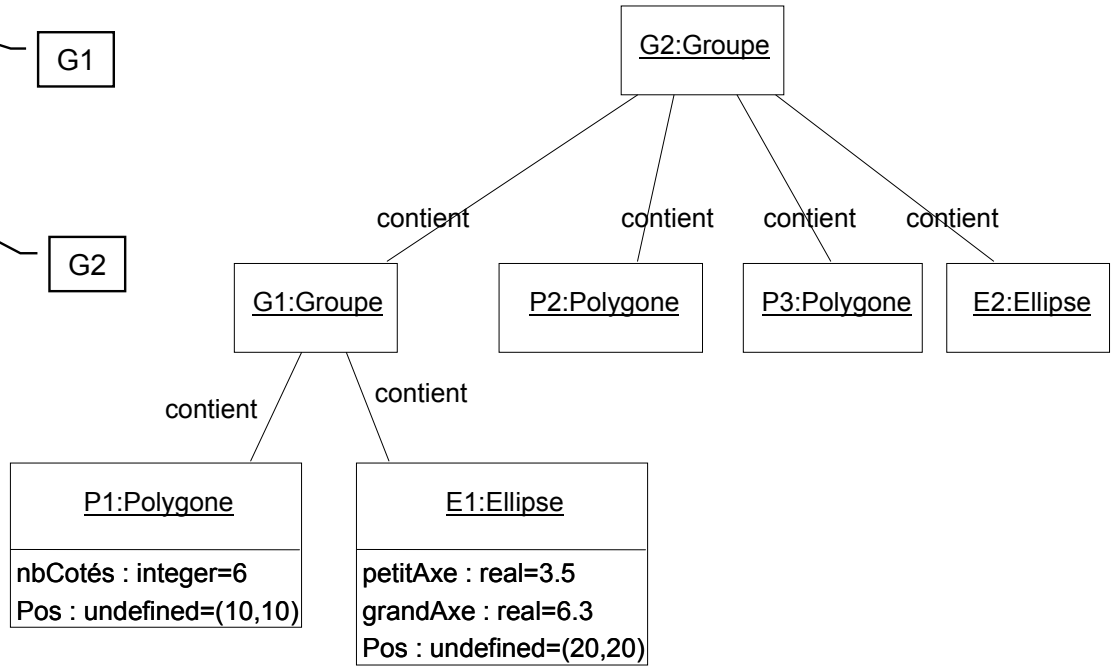
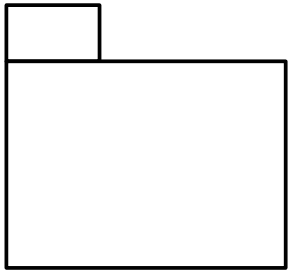


Diagramme de modules

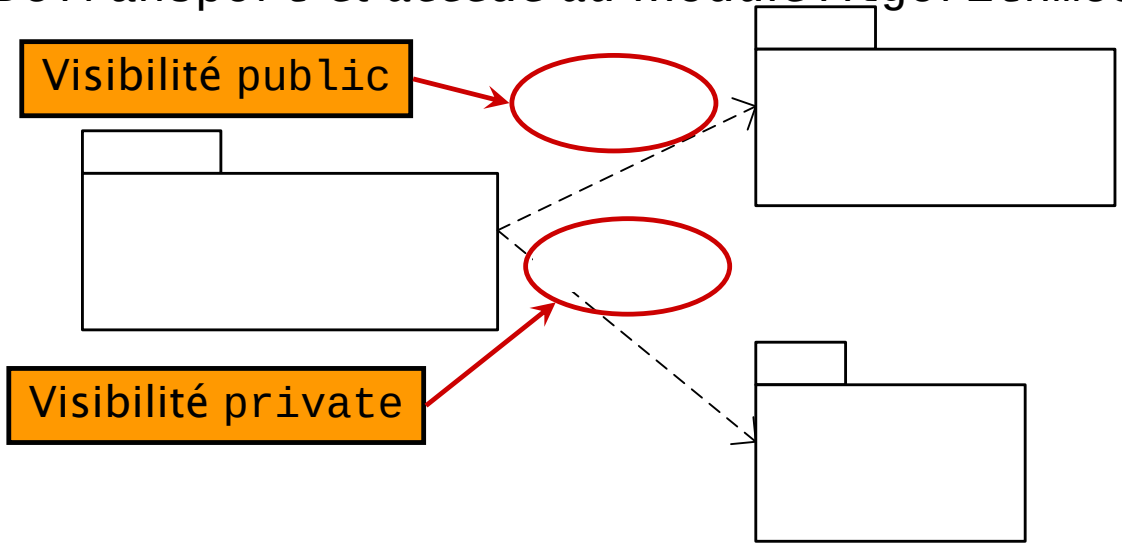
- Un module (ou paquetage) permet de regrouper des éléments UML interdépendants dans une même entité
- Représentation sur un exemple :



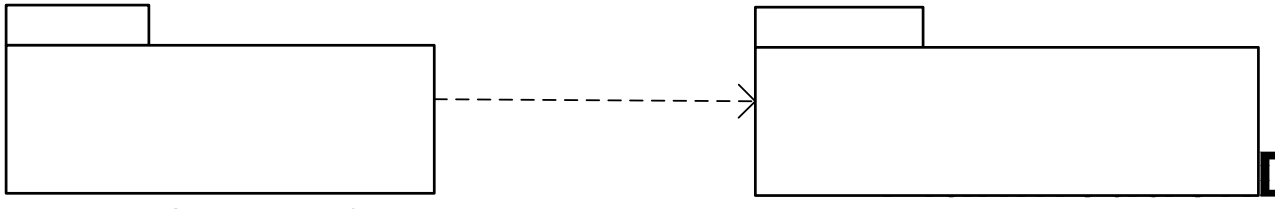
- Tous les éléments UML peuvent être regroupés dans des modules
- Chaque élément du module est nommé par :
  `nom_du_module::nom_de_l_element`
  exemple : `Utilitaires::Horloge`

Dépendances entre modules (exemples)

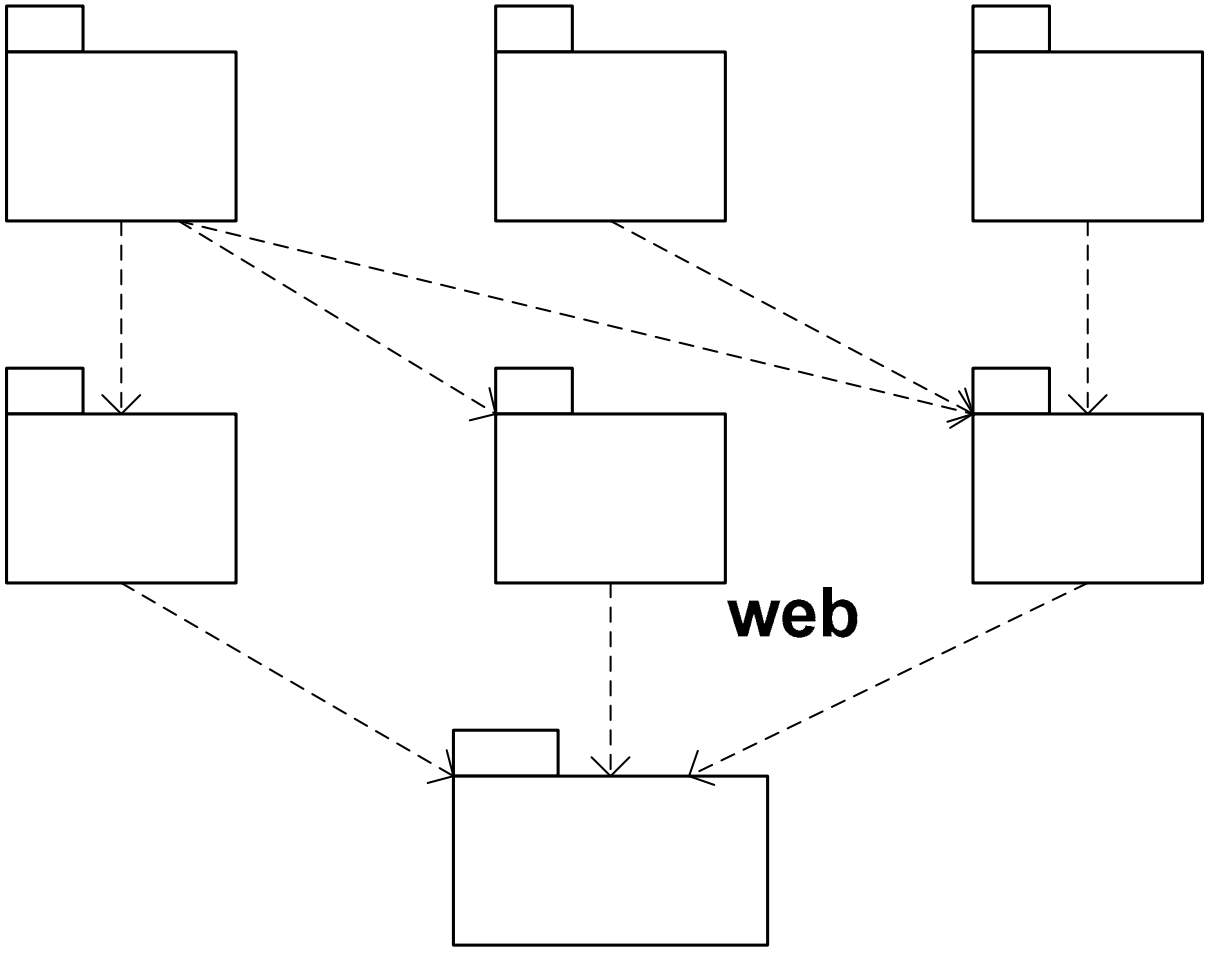
- `PlanificationDItineraire` importe le module `MoyensDeTransport` et accède au module `Algorithmes`



- `MachineDEtatParProtocole` fusionne les éléments de `MachineDEtatComportementale` pour intégrer les classes contenues



Structuration d'un projet à l'aide de modules

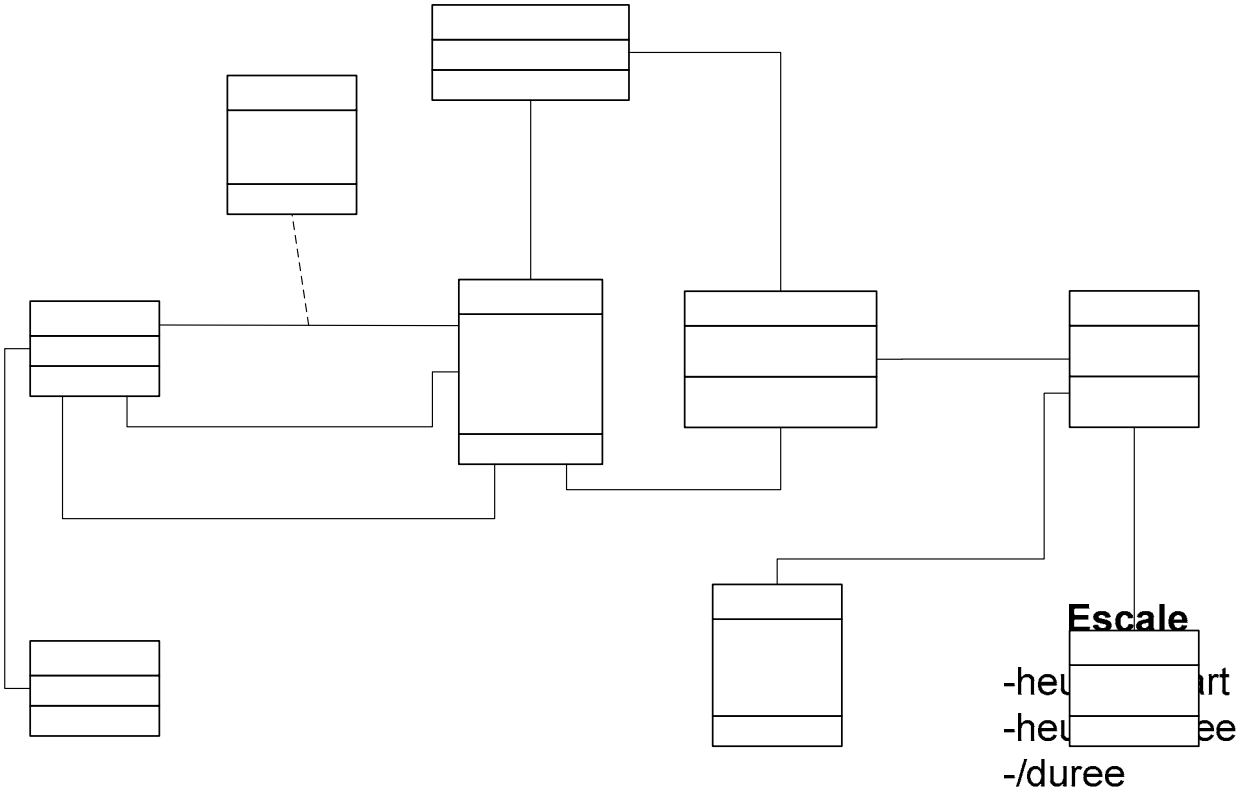


- Etude de cas guidée :
  - Système de réservation de vol
- Exercice libre :
  - Configurateur de voitures

Etude de cas guidée : réservation de vols

- Cette étude porte sur un système simplifié de réservation de vols pour une agence de voyages.
- Les entrevues avec des experts du métier ont permis d'extraire les connaissances suivantes du domaine :
  1. Des compagnies aériennes proposent différents vols
  2. Un vol est ouvert à la réservation et refermé sur ordre de la compagnie
  3. Un client peut réserver un ou plusieurs vols, pour des passagers différents
  4. Une réservation concerne un seul vol et un seul passager
  5. Une réservation peut être annulée ou confirmée
  6. Un vol a un aéroport de départ et un aéroport d'arrivée
  7. Un vol a un jour et une heure de départ et d'arrivée
  8. Un vol peut comporter des escales dans des aéroports
  9. Une escale a une heure d'arrivée et une heure de départ
  10. Chaque aéroport dessert une ou plusieurs villes
- Faire le diagramme de classes UML
- Proposer une structuration en paquetages

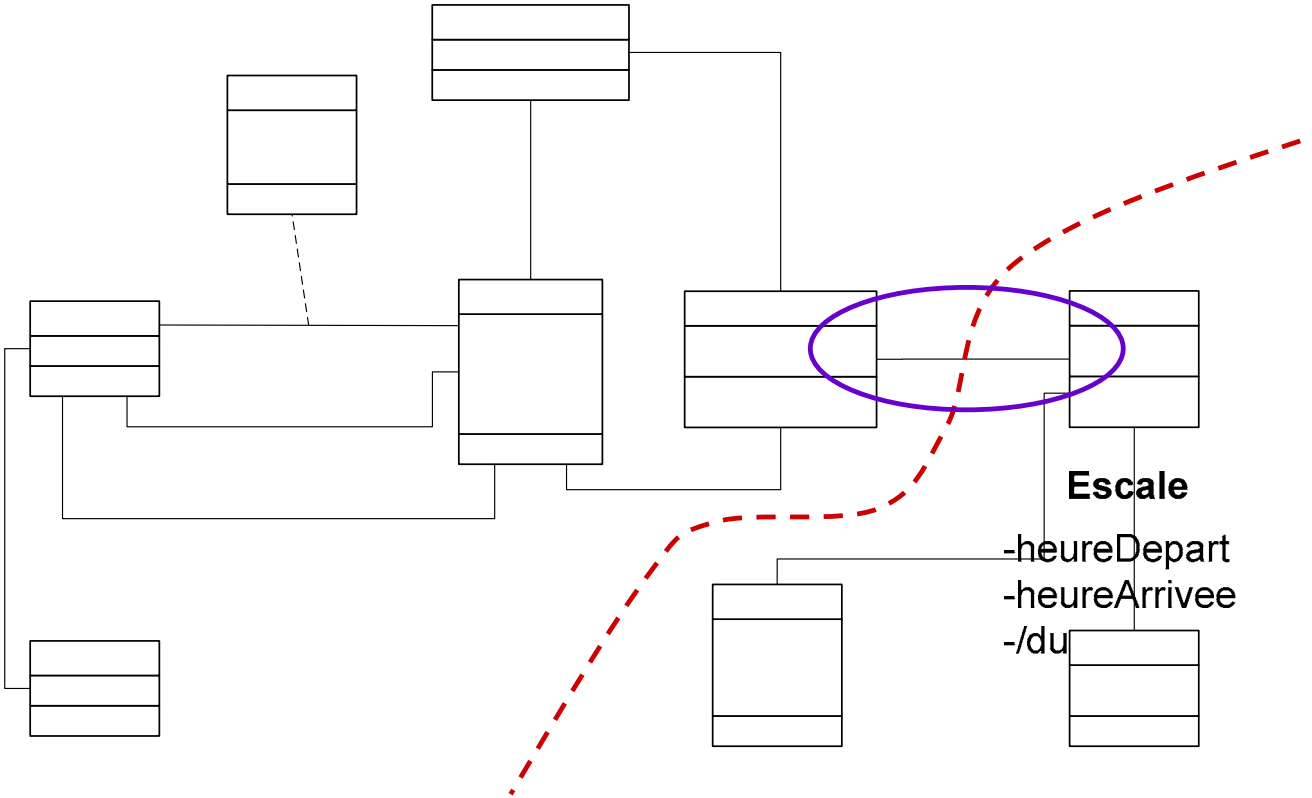
Réservation de vols



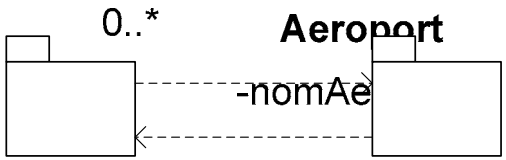
Principes de structuration en paquetages

- Regrouper les classes sémantiquement proches
  - Finalité : les classes rendent des services de même nature aux utilisateurs
  - Évolution : isoler les classes stables de celles susceptibles d'évoluer (classes métier et classes applicatives…)
  - Cycle de vie des objets : distinguer les classes dont les objets ont des durées de vie très différentes
- Minimiser les dépendances entre paquetages

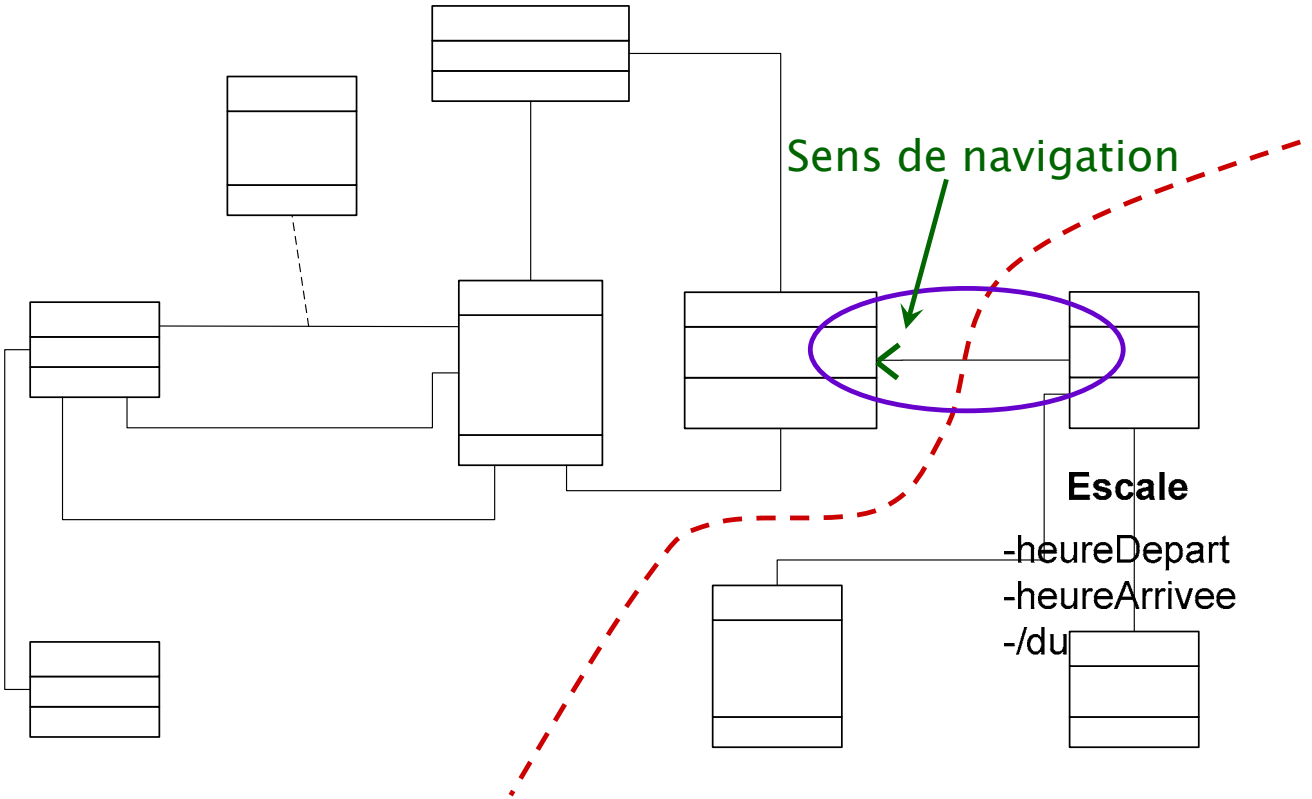
Découpage en 2 paquetages (solution 1)



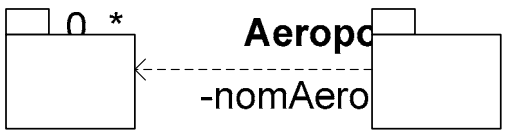
Dépendance mutuelle



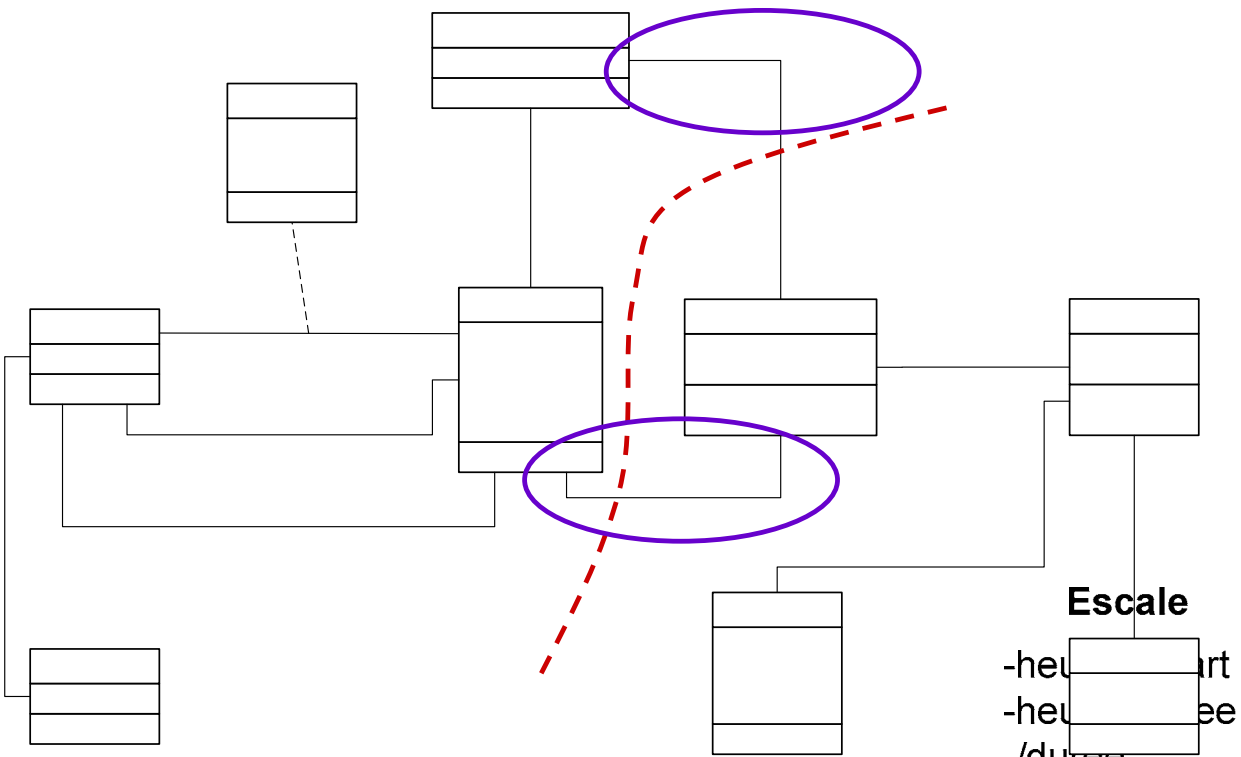
Découpage en 2 paquetages (solution 1 corrigée)



Couplage minimisé
entre paquetages



Découpage en 2 paquetages (solution 2)



La dépendance mutuelle entre paquetages est inévitable

Exercice libre : configurateur de voiture

- Cette étude porte sur l'étude des informations utiles à la réalisation d'un configurateur de voiture
- Le configurateur choisi est celui du constructeur automobile Renault (hors financement)

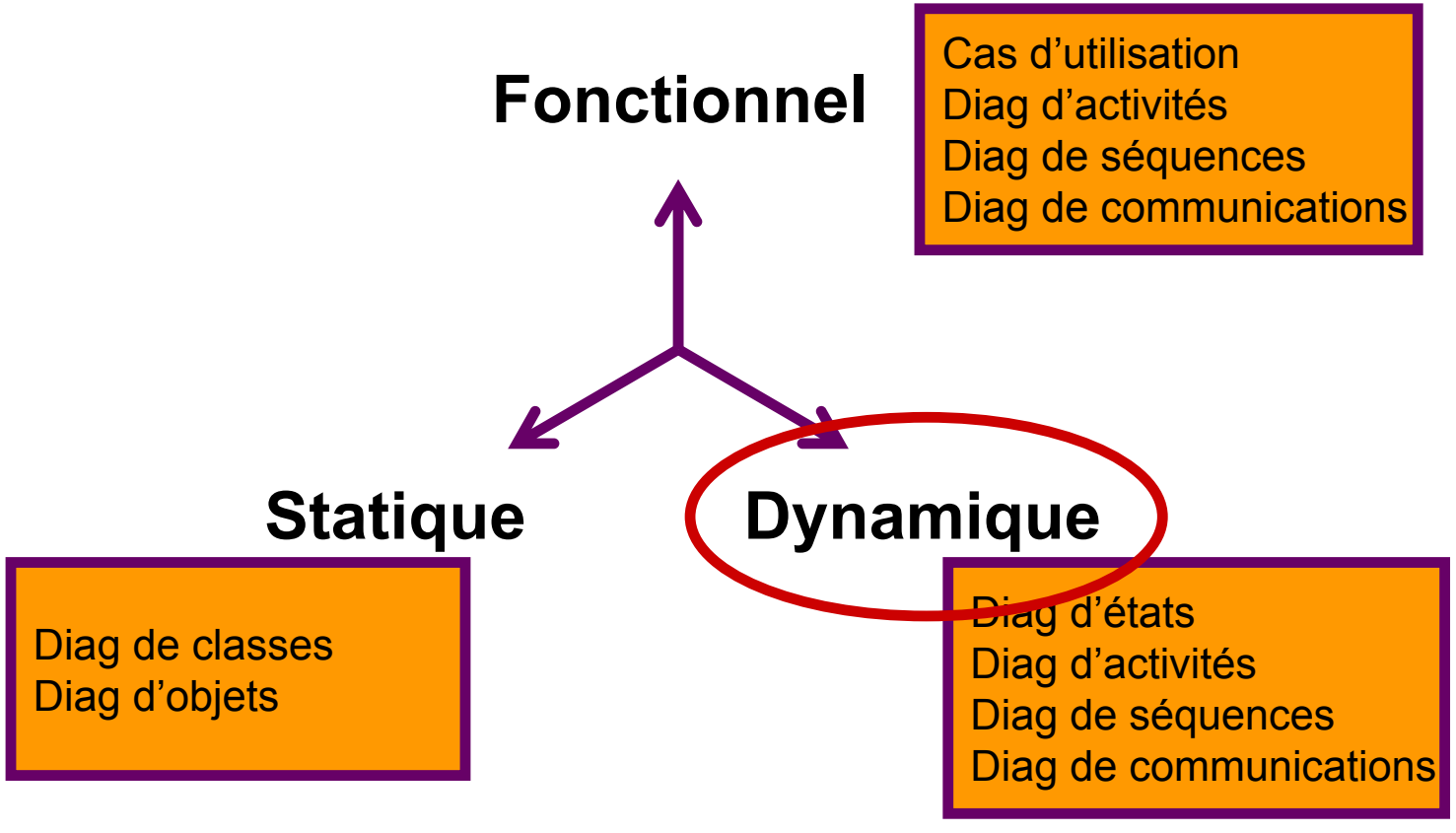
1. Aller sur le site du constructeur et configurer plusieurs véhicules en essayant de diversifier les choix
2. Réaliser le diagramme de classes UML correspondant
3. Réaliser le diagramme d'objets UML relatif au texte donné sur la diapo suivante
4. Réaliser le diagramme de classes UML permettant de représenter la vente entre 2 acteurs

Voiture : texte complémentaire

- Monsieur Duchemin possède une Renault Clio immatriculée 360 CMD 59 ayant pour numéro de série ABC123456789. C'est une version Dynamique 1.5dCi 105 (diésel) 5 portes d'une cylindrée de 1461cm3 et d'une puissance administrative de 6cv. Le prix TTC conseillé est 18550€.

- Mademoiselle Dulcinée possède la même voiture en version 3 portes, immatriculée 314 API 62 dont le numéro de série est ZYX987654321. Elle a choisi les options suivantes : peinture métallisée (390€) fournie par Renault, régulateur de vitesse (230€) fourni par Valéo et pré-équipement téléphone GSM (130€) fourni par Siemens.

…et les diagrammes étudiés



Rôle des modèles dynamiques

- Constituent un plus par rapport aux méthodes systémiques (Merise)
  - Relations temporelles et événementielles entre objets
  - États des objets au cours de déroulement de l'application
  - Actions effectuées par les objets dans un contexte donné
  - Actions et réactions des objets vis à vis des sollicitations extérieures

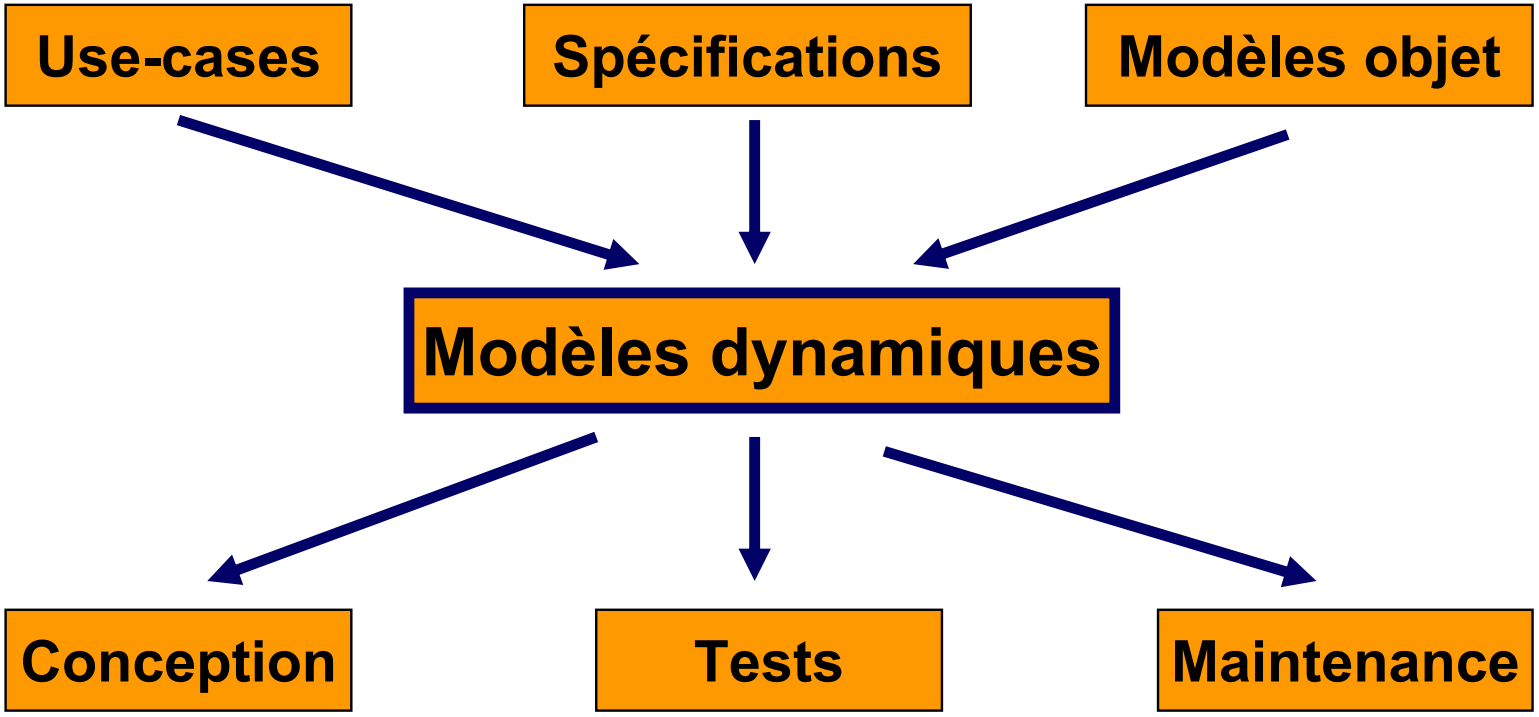
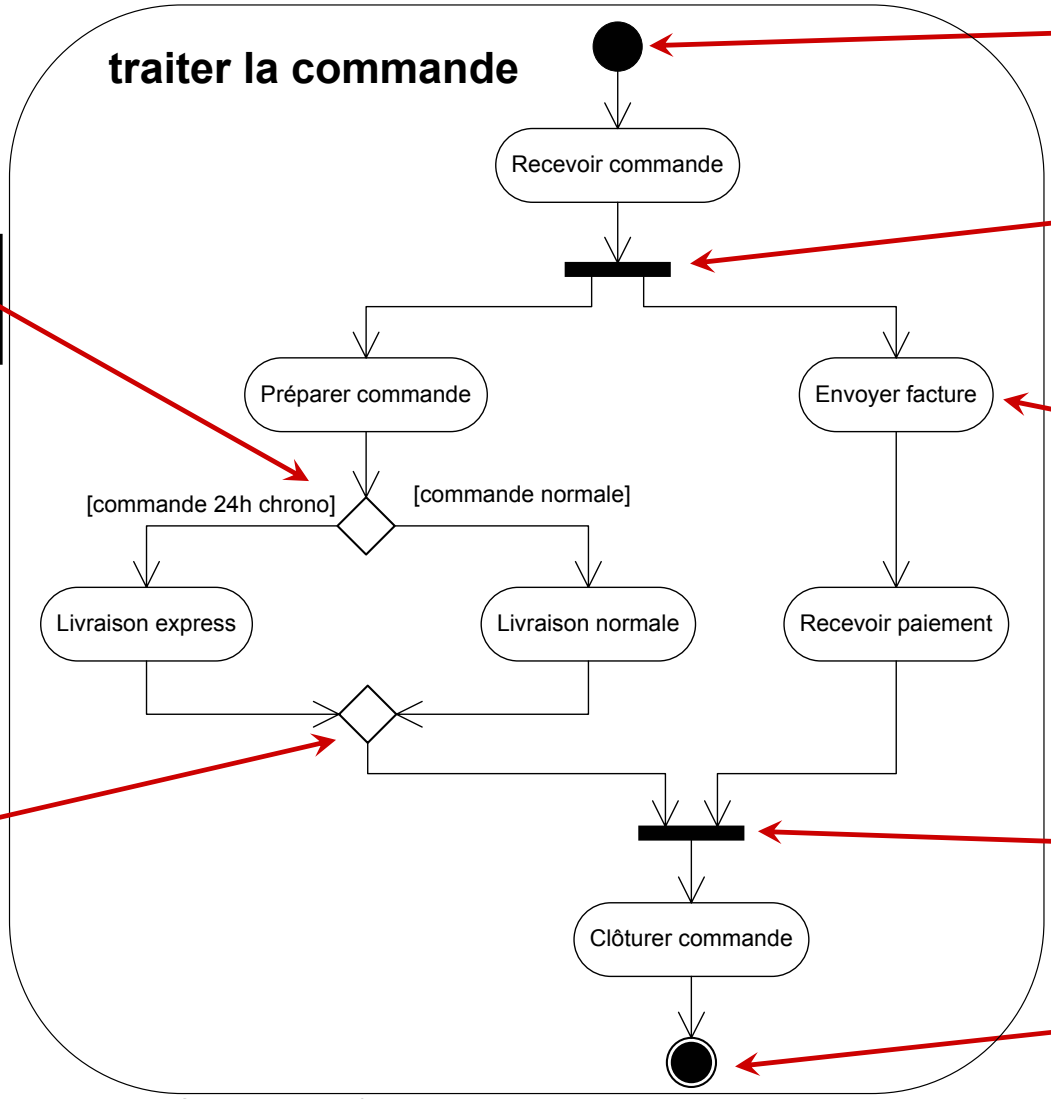

Use-cases
Spécifications
Modèles objet
Modèles dynamiques
Conception
Tests
Maintenance

Diagramme d'activités

- Décrit la logique procédurale, les processus métier et les enchaînements d'activités
- Semblables aux organigrammes, mais permettent de plus de représenter des comportements parallèles

Diagramme d'activités



Nœud initial

Nœud fourche

Nœud décisionnel

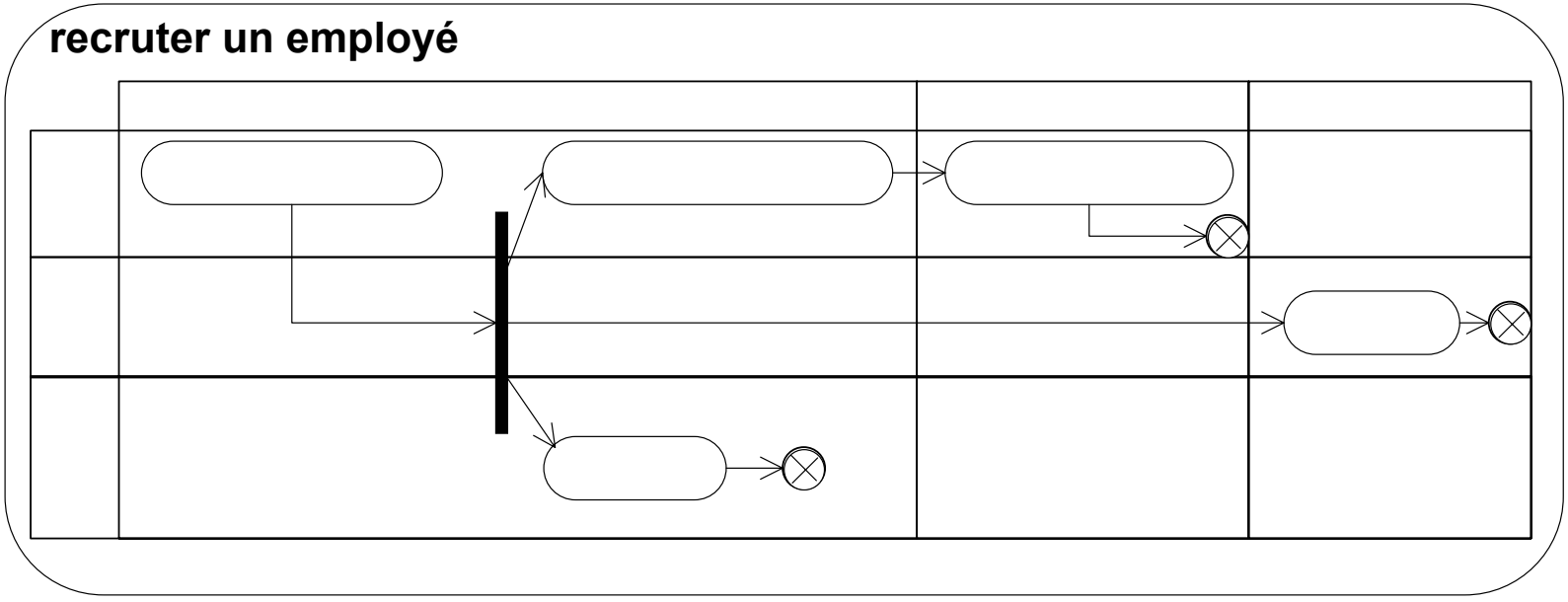
Activité

Nœud de confluence

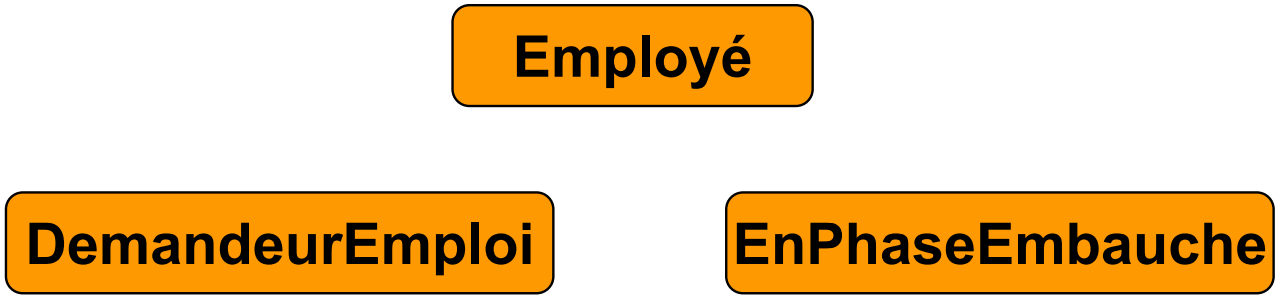
Nœud de jonction

Nœud final

Diagramme d'activités avec partitions multidimensionnelles



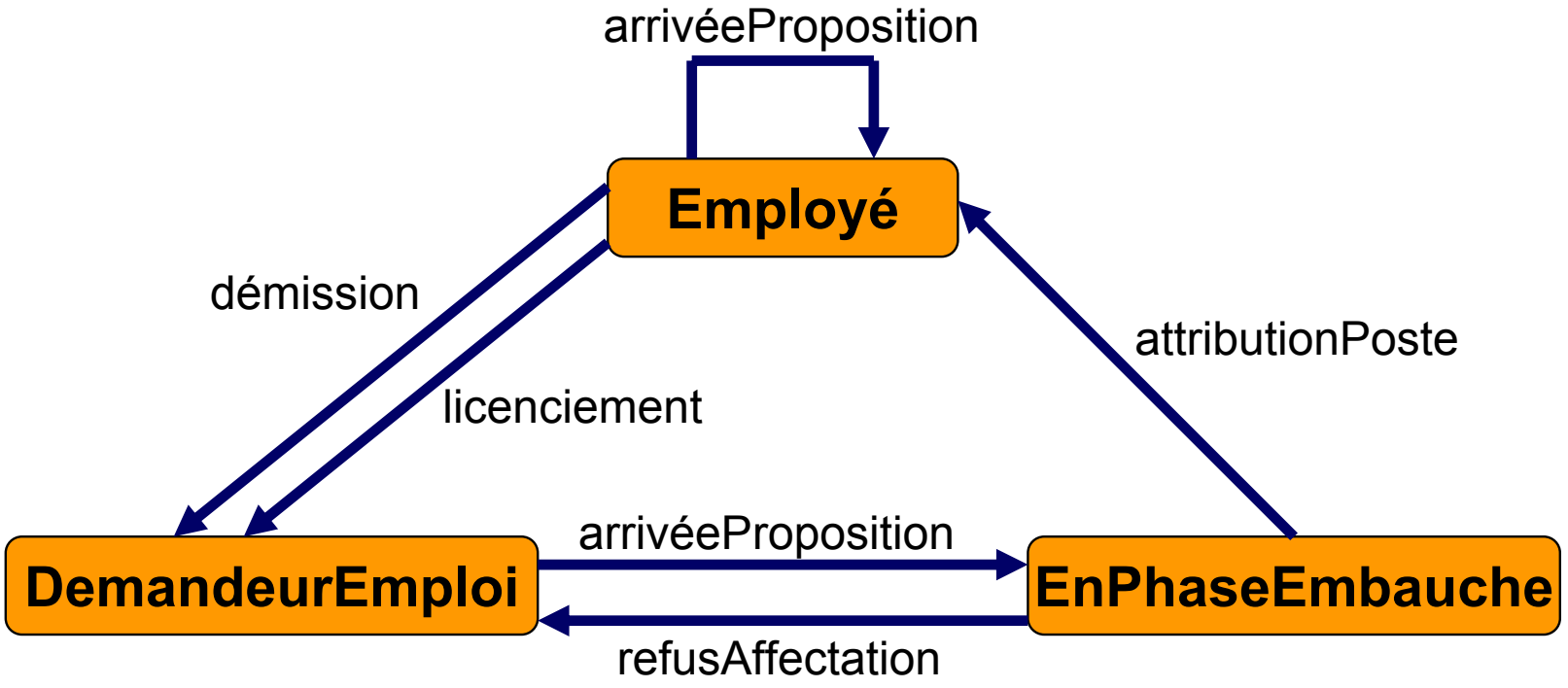
- Notion d'état d'un objet :
  - Valeur de ses variables d'instance et de ses liens avec les autres objets pour un temps séparé par 2 événements



- Notion d'événement et de message :
  - Événement : stimulus pouvant transporter des informations (exemple : clic souris)
  - Message : événement particulier impliqué dans l'interaction entre 2 objets

- Notion de transition :
  - Changement d'état d'un objet causé par un événement

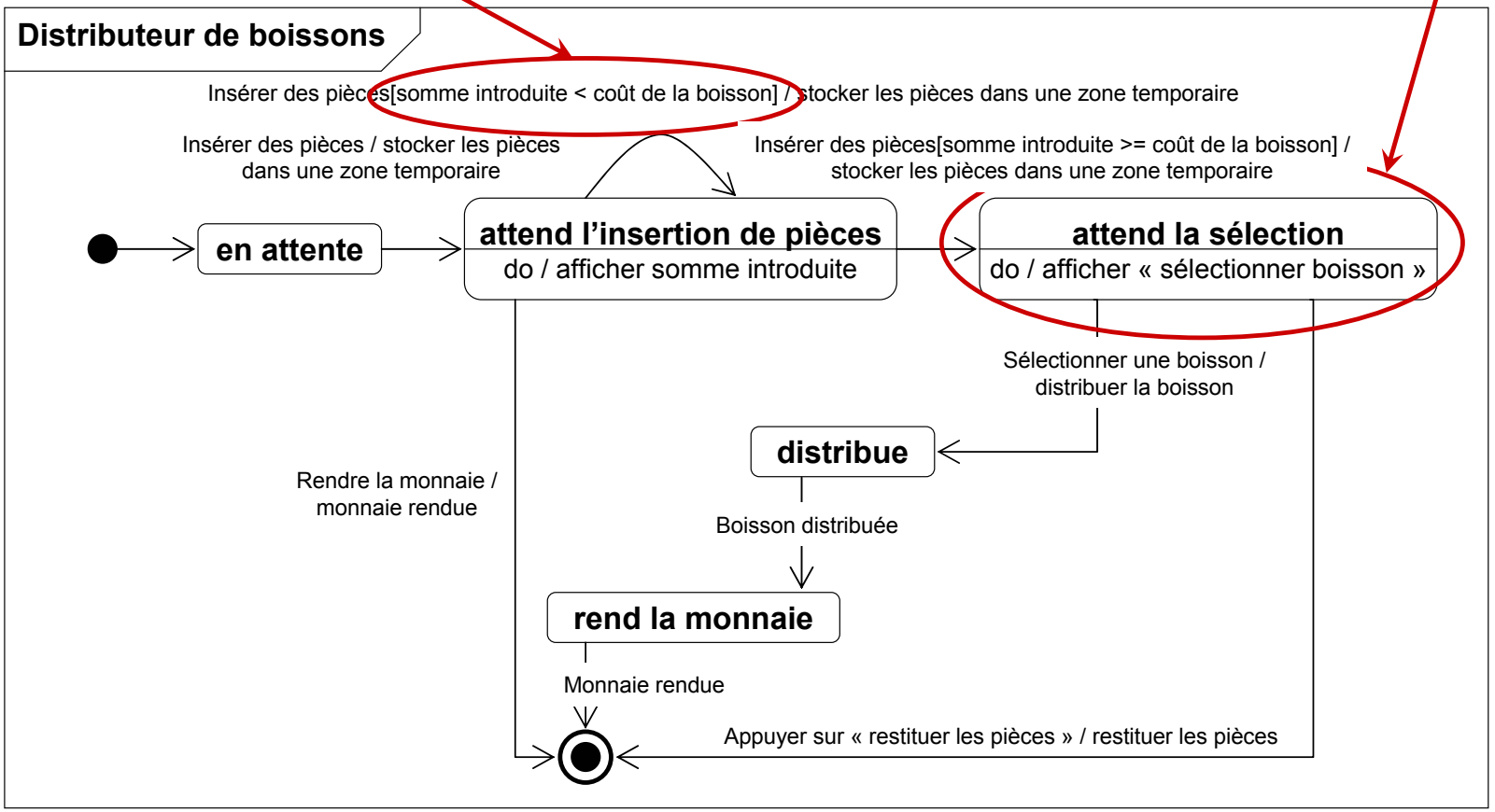
Diagramme d'états



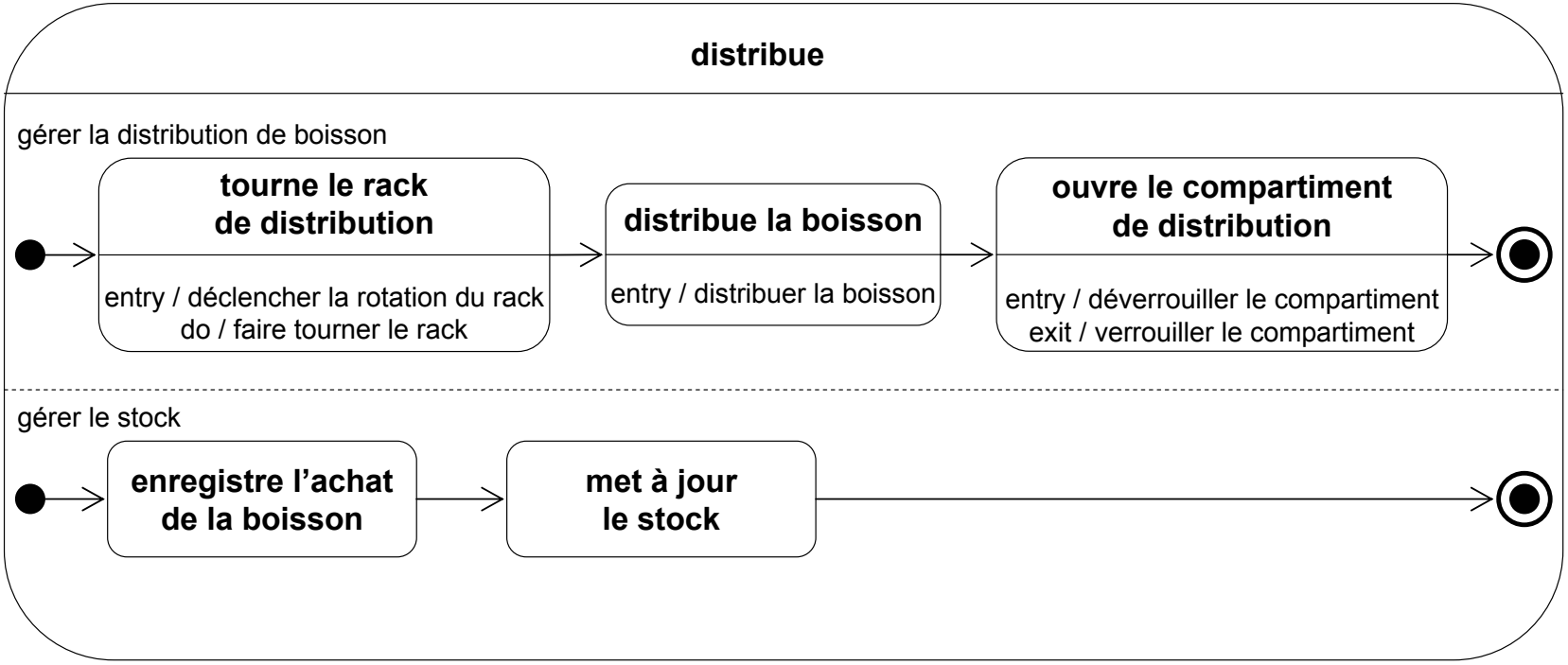
Exemple : distributeur de boissons

Condition de garde empêchant un changement d'état tant que le client n'a pas versé toute la somme à payer

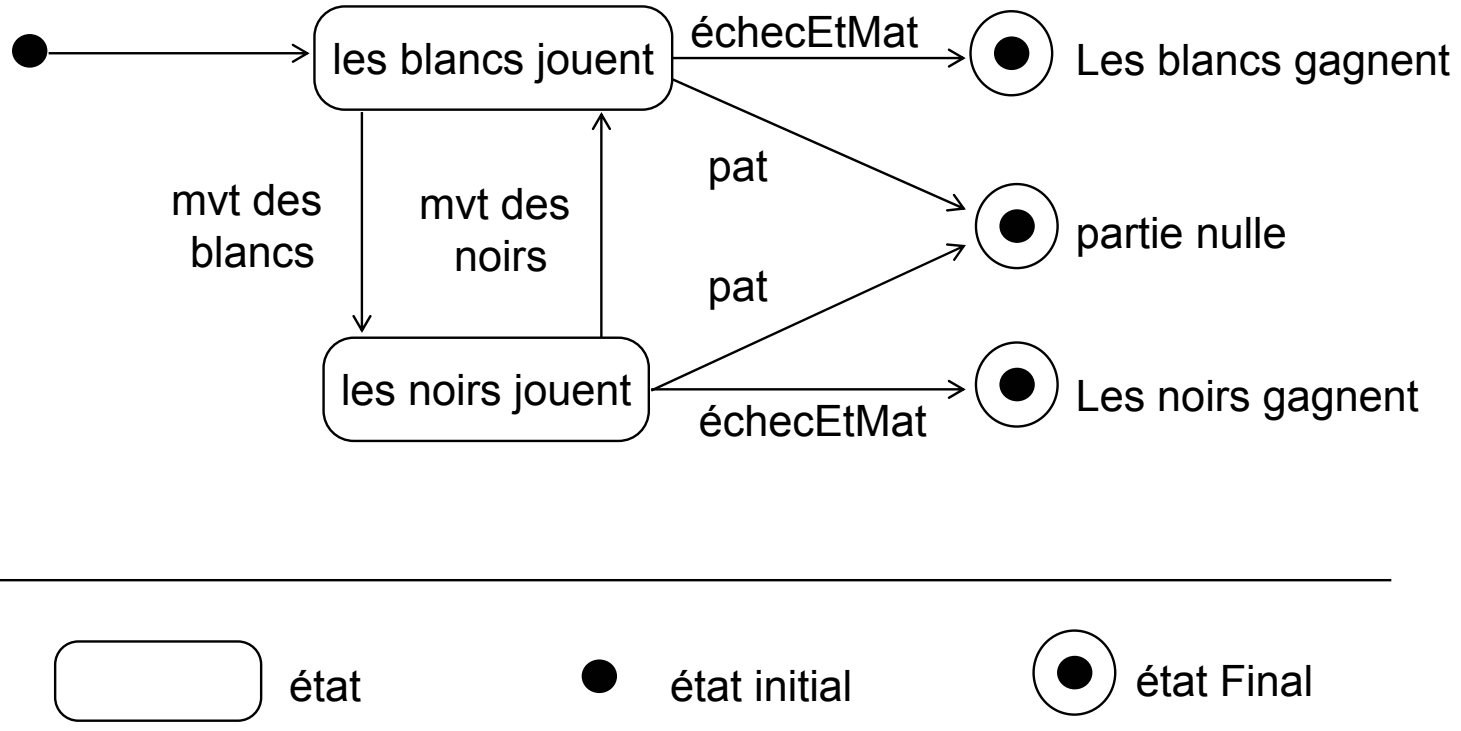
État comportant une activité de type do



- Un état composite possédant 2 régions



Exemple de diagramme états-transitions



Autres diagrammes dynamiques

- Déjà vus précédemment :
  - Diagramme de séquences
  - Diagramme de communications

Applications

- Etudes de cas guidée :
  - Publiphone
  - Montre
- Exercices libres :
  - Diagramme d'activité du Guichet Automatique de banque
  - Contrat
  - Projets

Etude de cas guidée : Publiphone

- Cette étude porte sur un système simplifié de publiphone à pièces.
  1. Le prix minimal d'une communication interurbaine est de 0,50€.
  2. Après l'introduction de la monnaie, l'utilisateur a 2mn pour composer son numéro (ce délai est décompté par le standard).
  3. La ligne peut être libre ou occupée.
  4. Le correspondant peut raccrocher le premier.
  5. Le publiphone consomme de l'argent dès que l'appelé décroche et à chaque unité de temps (UT) générée par le standard.
  6. On peut ajouter des pièces à tout moment.
  7. Lors du raccrochage, le solde de monnaie est rendu.
- Identifier les acteurs
- Identifier les cas d'utilisation
- Construire le diagramme de communication
- Élaborer le diagramme d'états du publiphone

Etude de cas guidée : Montre

- Le mode courant est « affichage »
- Quand on appuie une fois sur le bouton mode, la montre passe en « modification heure » ; chaque pression sur le bouton avance incrémente l'heure d'une unité
- Quand on appuie une nouvelle fois sur le bouton mode, la montre passe en « modification minute » ; chaque pression sur le bouton avance incrémente les minutes d'une unité
- Quand on appuie une nouvelle fois sur le bouton mode, la montre repasse en mode « affichage »
- De plus, quand on appuie sur le bouton avance plus de 2'', les heures ou les minutes s'incrémentent rapidement jusqu'à ce que le bouton soit relâché
- Faire le diagramme d'états UML correspondant
- Compléter le diagramme d'états en prenant en compte la gestion de l'éclairage, l'affichage de l'alarme, la mise sous alarme

Exercice libre : Contrat

- Plusieurs sociétés décident d'établir un contrat.
- Pour ce faire elles rédigent un projet par itérations successives.
- Le contrat est ensuite informellement accepté par les parties, et devient ce que l'on appelle un pré-accord. A ce stade il peut toujours être l'objet de modification et revenir à l'état de projet.
- Une fois le pré-accord définitivement établi, le contrat est signé par les parties. Dès ce moment les partenaires sont liés.
- Une fois signé, le contrat peut être rendu exécutoire par une décision d'une des parties. Un contrat en exécution peut faire l'objet de discussions qui sont réglées par un arbitre désigné à cet effet.
- Le contrat une fois exécuté prend fin.
- Faire le diagramme d'états UML correspondant.

Exercice libre : BD projets

- Dans le but de faciliter la communication interne, on souhaite réaliser une base de données reprenant systématiquement quelques informations relatives aux projets G1-G2 :
  - Code projet, titre, élèves, directeur scientifique, pilote, consultants, partenaire, résumé, photos, poster, présentations, rapports
- On souhaite préciser pour le budget :
  - La part fournie par l'école
  - La part fournie par un organisme, et lequel
  - La part fournie par chaque entreprise partenaire, quelle est cette entreprise, et qui sont ses contacts (fonction, téléphone, mél…)
- Pour l'équipe d'encadrement, on veut définir pour chacun des intervenants ses domaines de compétences, de façon à permettre de répondre à une question comme « Qui est compétent en capteurs ? »
- L'administration fournira une table des élèves
- Modéliser ce système d'information en utilisant UML