Introduction à UML

Jean-Yves Didier

Détails pratiques

- En cas de questions :

  didier@iup.univ-evry.fr

- Où trouver ce cours :

  http://lsc.univ-evry.fr/~didier/pedagogie/uml1.pdf

Organisation

- 2 intervenants :
  - Jean-Yves Didier (partie académique),
  - Hamid Nait-Bouda (partie professionnelle),

- 4 séances de 4 h chacune :
  - Semaines 50, 51, 2 et 3,
  - Partie académique : 50, 51 et 2
    - Semaines 50 et 51 : cours,
    - Semaine 2 : exemple détaillé,
  - Partie professionnelle : semaine 3

Pré-requis

- Programmation orientée objets,
  - Concepts de POO : héritage, polymorphisme, etc ...
- Programmation système,
  - Processus concurrents,
- Méthode MERISE,
  - Analyse des besoins de l'utilisateur,
  - Modèles conceptuels et logiques.
- Grafcets et automates.

Plan

- Introduction générale,
- Meta-modèle UML,
- Notion de vues d'un système,
- Diagrammes UML,

Plan

- **Introduction générale,**
- Meta-modèle UML,
- Notion de vues d'un système,
- Diagrammes UML,

Qu'est ce que UML ?

- **UML** signifie **U**nified **M**odelling **L**anguage,
- *UML est un langage standard, pour spécifier, visualiser, concevoir et documenter tous les aspects d'un système d'information*,
- **UML** fournit un support de communication : un **langage graphique** comportant **13 diagrammes** standards (pour UML 2.0) représentant des *'vues'* d'un système d'information,
- **UML** permet d'exprimer et d'élaborer des **modèles objet**, indépendamment de tout langage de programmation.

Pourquoi UML ?

- Les 'lacunes' de l'approche objet :
  - Moins intuitive que l'approche fonctionnelle,
    - Comment conduire une analyse ?
    - Comment concevoir de manière pertinente ?
    - Comment comparer les solutions éventuelles ?
  - L'approche objet nécessite une grande rigueur,
    - Vocabulaire précis : les erreurs créent des ambiguïtés,
    - Comment décrire la structure d'un objet de manière pertinente ?
- Nécessité d'un outil à dimension méthodologique pour l'approche objet !

Historique (1/2)

- UML est la fusion de 3 langages de modélisation objet des années 90 :
  - OMT (James Rumbaugh) : vues statiques dynamiques et fonctionnelles d'un système,
  - Booch (Grady Booch) : utilisation de 7 diagrammes pour représenter un système orienté objets,
  - OOSE (Ivar Jacobson) : analyse fondée sur la description des besoins des utilisateurs (use case),
- 1995 : fusion OMT, Booch (Unified method 0.8),
- 1996 : OOSE est inclus (UML 0.9), nouveaux acteurs : DEC, HP, Microsoft, Oracle, etc ...

Historique (2/2)

- 1997 :
  - (Janvier) Soumission de UML 1.0 à l'OMG,
  - (novembre) UML 1.1 standardisé par l'OMG,
- 1998 : UML 1.2
  - 1999 : UML 1.3
  - 2000 : UML 1.4
  - 2003 : UML 1.5
- Maintenant :
  - UML 1.4.2 : standard ISO/IEC 19501:2005,
  - L'OMG prépare la spécification de UML 2.0.

Pourquoi utiliser UML ?

- UML est un langage formel normalisé,
  - Gain de précision,
  - Gage de stabilité,
  - Encourage l'utilisation d'outils.
- UML est un support de communication performant,
  - Il cadre l'analyse,
  - Il facilite la compréhension de représentations abstraites complexes,
  - Il est souple et polyvalent.

UML et l'entreprise

- Sondage en 2004
  - 35 représentants de la MOA (Maîtrise d'OuvrAge),
  - Avril à Juin 2004,
  - Une partie concerne UML !
  - Résultats :
    - UML est utilisé dans 20% des projets,
    - UML est préconisé dans 35% des entreprises,
    - Son utilisation est laissée à l'appréciation des intervenants dans 20% d'entre elles.
- Taux de pénétration global en 2005 :
  - Estimé à 10% dans l'industrie informatique.

Plan

- Introduction générale,
- **Meta-modèle UML,**
- Notion de vues d'un système,
- Diagrammes UML,

Le méta-modèle UML

- UML : langage permettant de créer des modèles,
  - ➔ UML : modélisation des modèles, un **méta-modèle.**
- Le méta-modèle UML est en 4 couches:
  - (M3) métamétamodèle : (concept de métaclasse)

    Définit le langage pour la spécification des metamodèles,
  - (M2) métamodèle : (concept de classe)

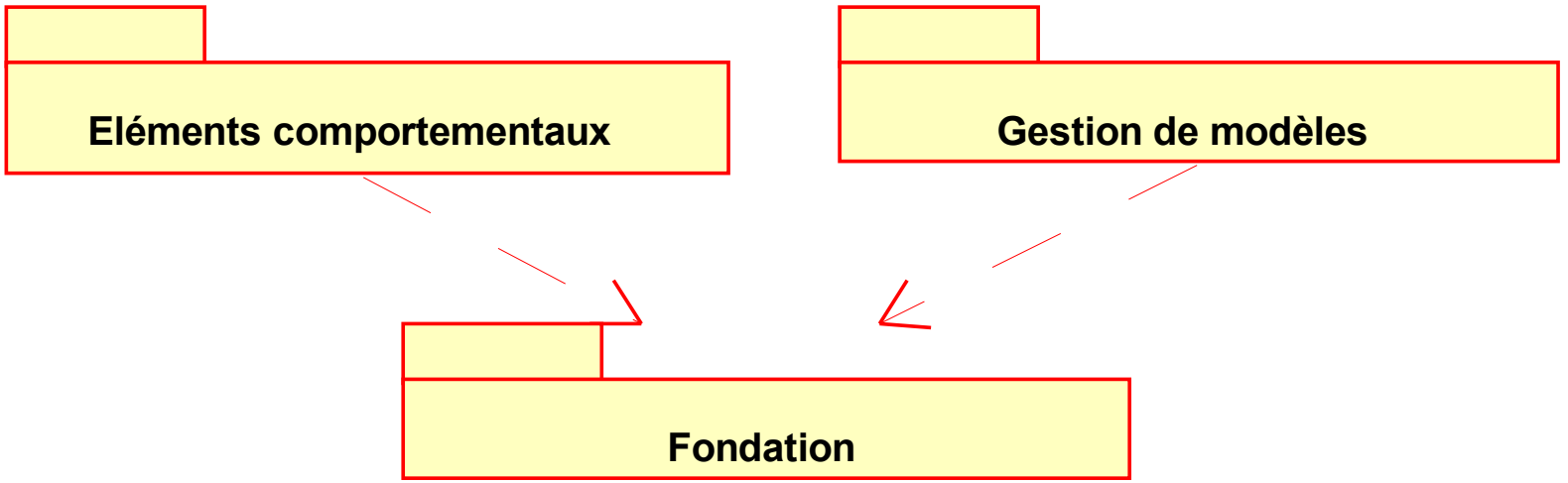
    Définit le langage pour la spécification des modèles,
  - (M1) modèle : (classe)

    Définit le langage pour les éléments d' un domaine,
  - (M0) objets utilisateur : (objet)

    Définit les données spécifiques du domaine.

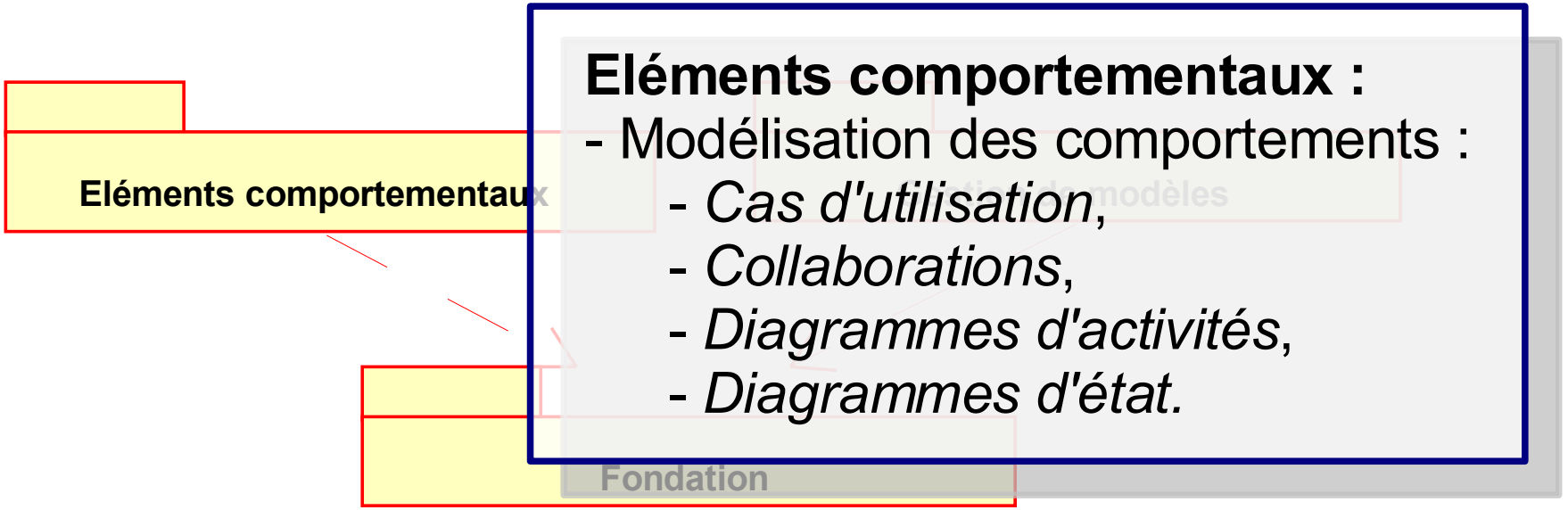
Organisation du métamodèle

- Métamodèle organisé en paquetages :



Organisation du métamodèle

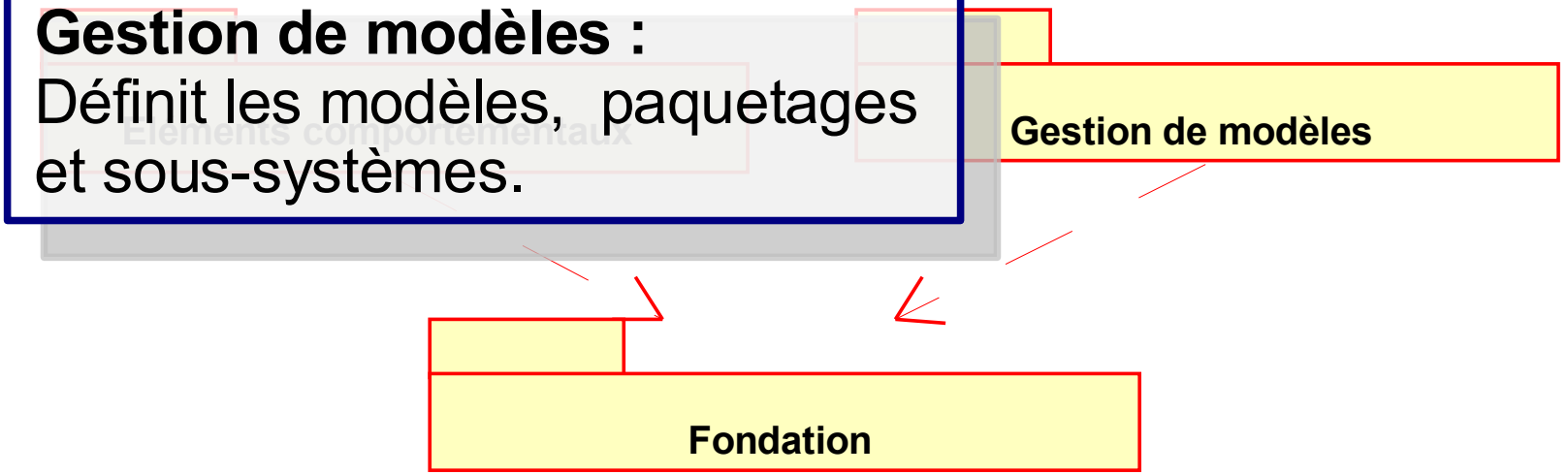
- Métamodèle organisé en paquetages :



Organisation du métamodèle

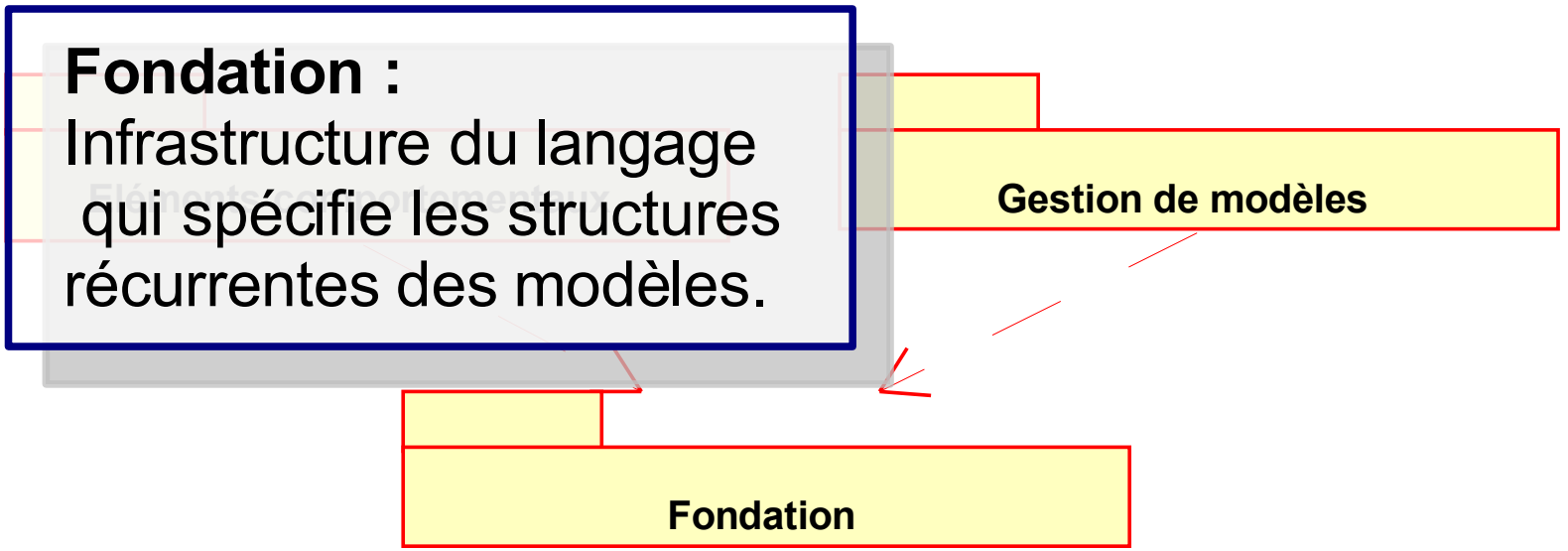
- Métamodèle organisé en paquetages :

Gestion de modèles :
Définit les modèles, paquetages et sous-systèmes.



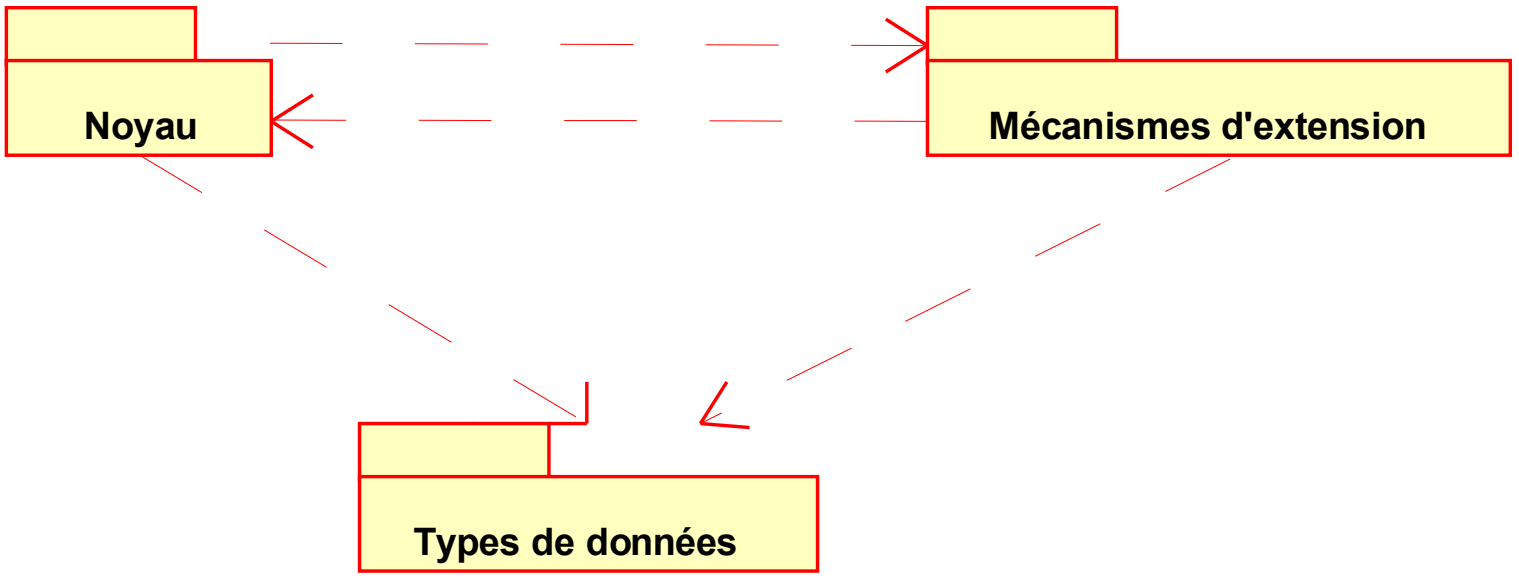
Organisation du métamodèle

- Métamodèle organisé en paquetages :



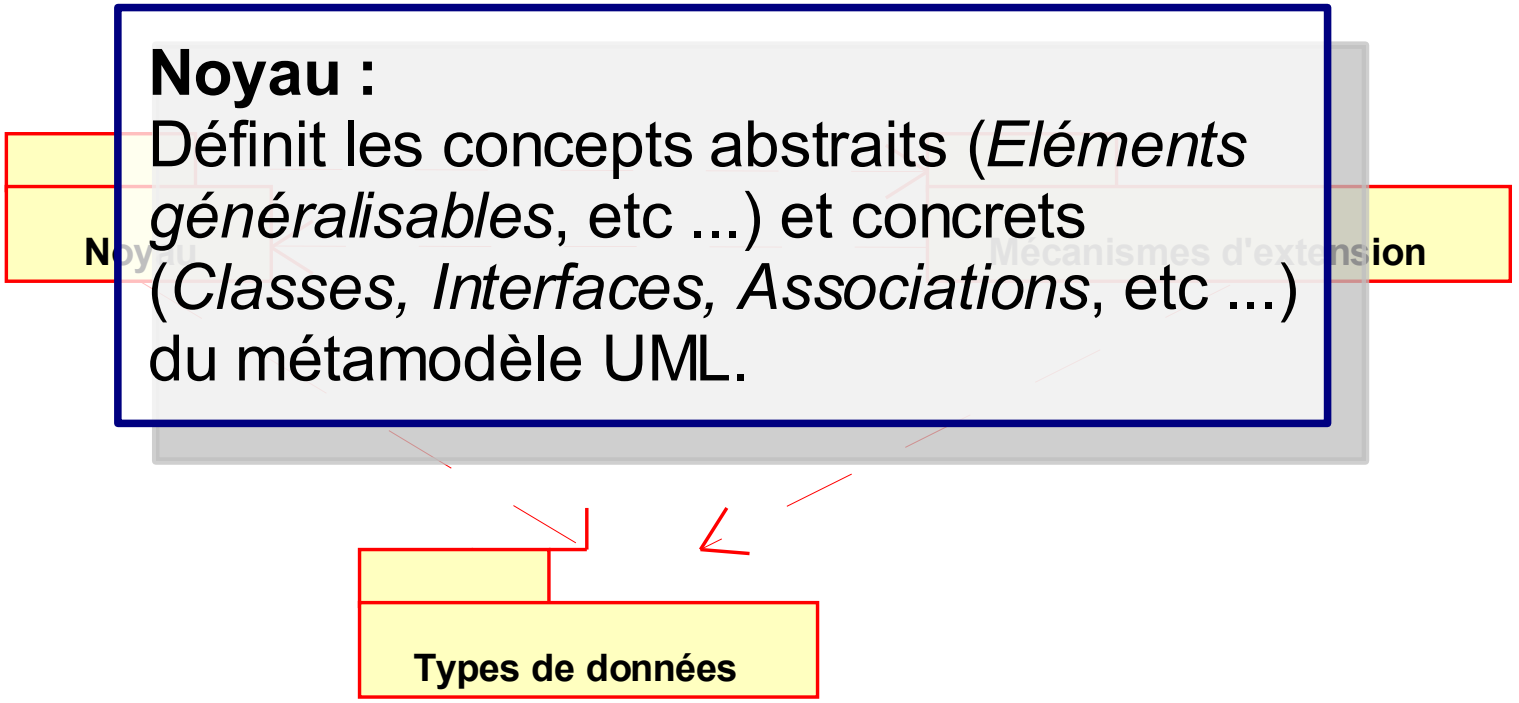
Le paquetage fondation

- Infrastructure du language qui spécifie les structures récurrentes des modèles.



Le paquetage fondation

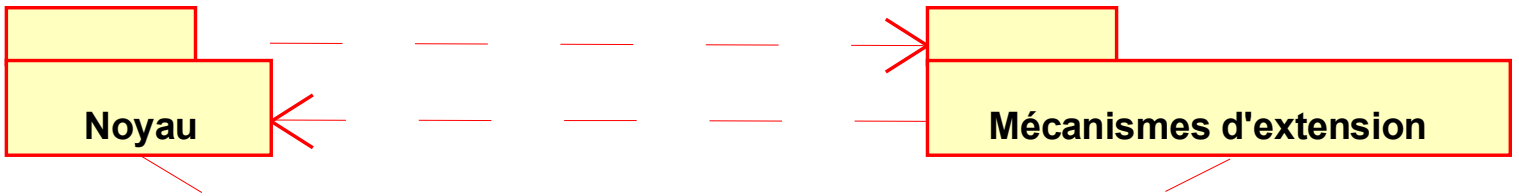
- Infrastructure du langage qui spécifie les structures récurrentes des modèles.



Noyau :
Définit les concepts abstraits (*Eléments généralisables*, etc ...) et concrets (*Classes, Interfaces, Associations*, etc ...) du métamodèle UML.

Le paquetage fondation

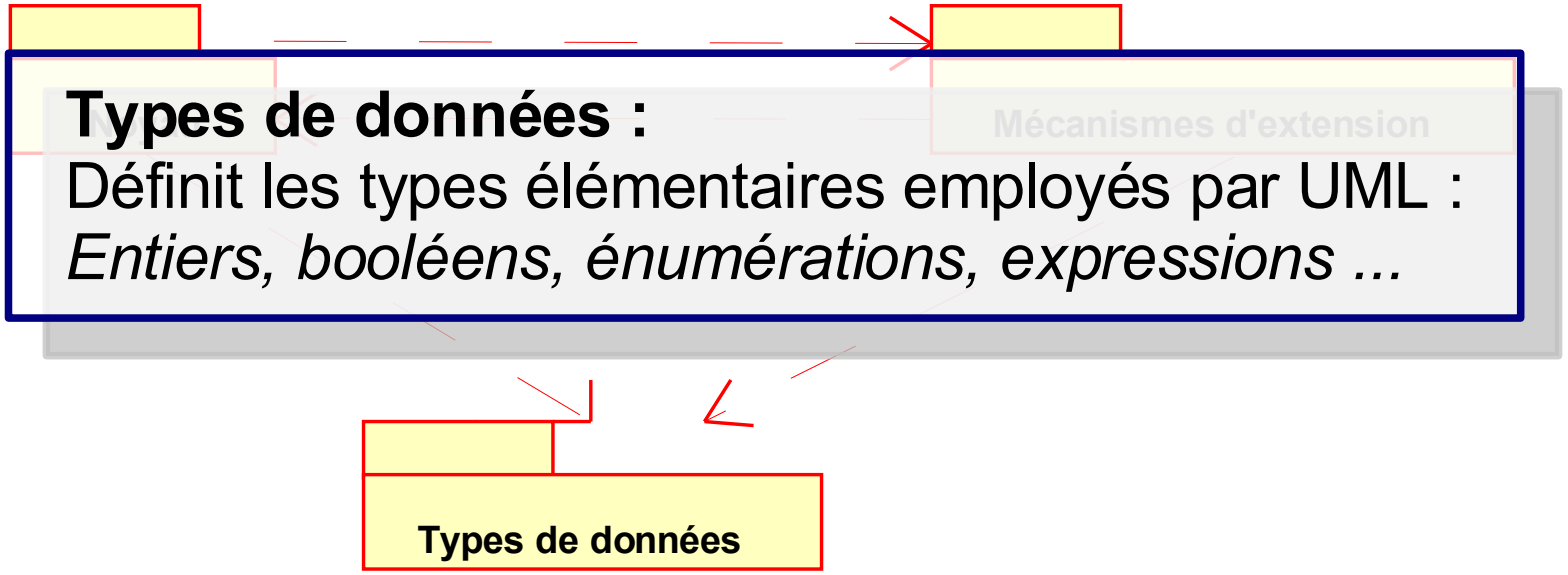
- Infrastructure du langage qui spécifie les structures récurrentes des modèles.



Mécanismes d'extension :
Définit la manière de modifier et d'adapter certains éléments spécifiques du modèle UML par le biais des *stéréotypes, contraintes, valeurs* et *définitions accolées*

Le paquetage fondation

- Infrastructure du langage qui spécifie les structures récurrentes des modèles.



Plan

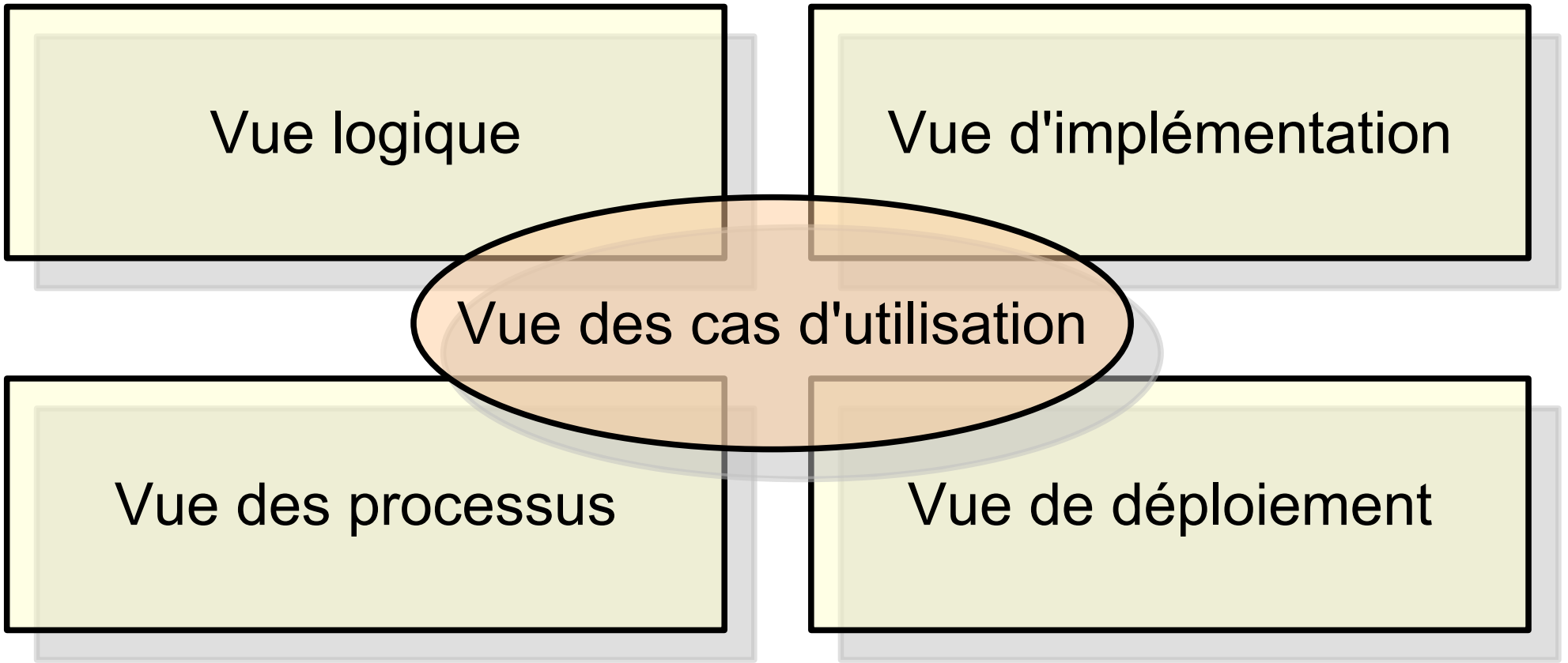
- Introduction générale,
- Meta-modèle UML,
- **Notion de vues d'un système,**
- Diagrammes UML,

La notion de *'vues'*

- Un système d'informations (SI) peut être envisagé suivant plusieurs points de vues :
  - Utilisateur final, développeur, ingénieur système, directeurs de projets, etc ...

- UML s'appuie sur le modèle 4+1 vues :
  - [Kruchten – IEEE Software 95]
  - L'idée est d'organiser la description d'une architecture logicielle en utilisant plusieurs vues concurrentes, chacune s'intéressant à un aspect particulier du problème.

Le modèle 4+1 vue



Le modèle 4+1 vue

Vue logique

Vue d'implémentation

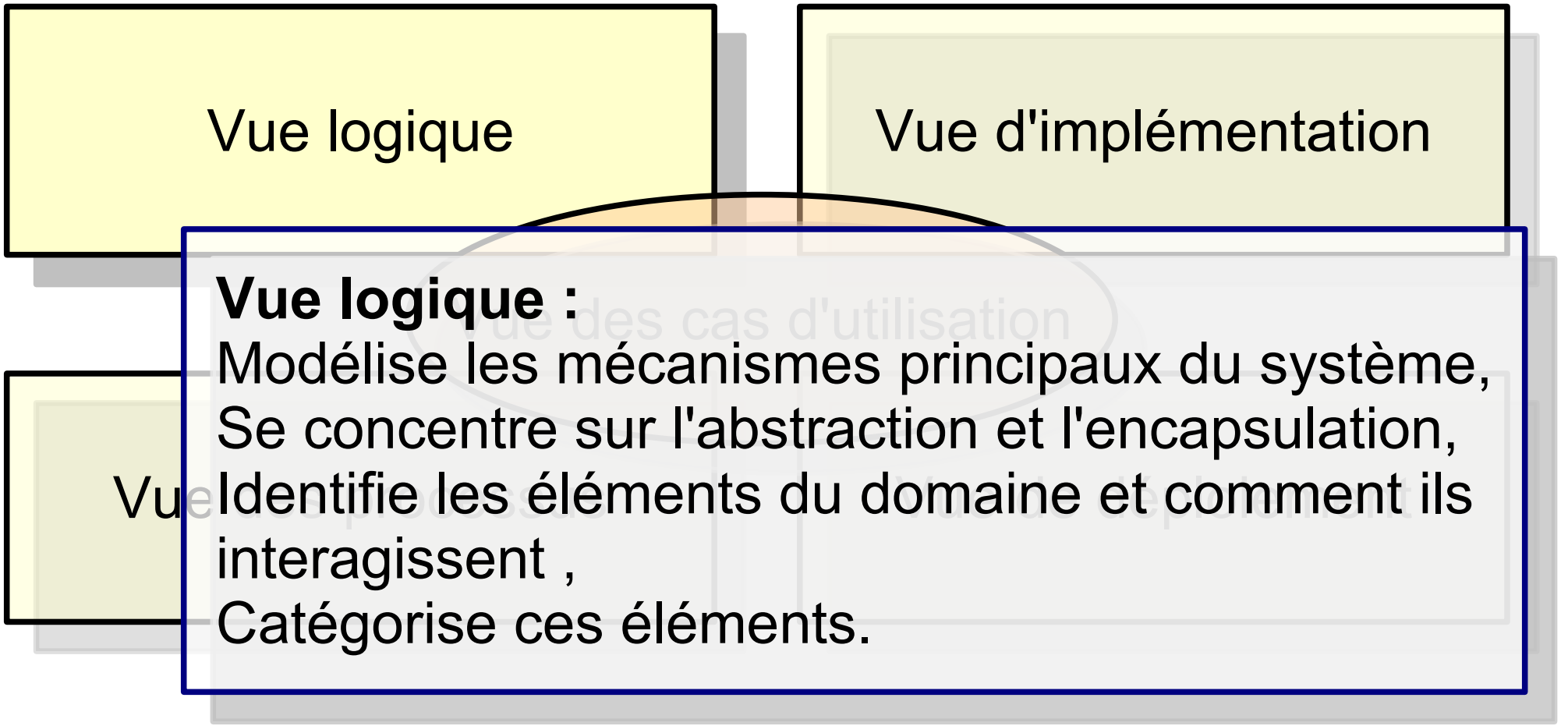
Vue des cas d'utilisation

Vue des pro

de déploiement

Vue des cas d'utilisation :
Description du système 'vue' par les acteurs du système,
Définit les besoins des **utilisateurs**,
Unifie les 4 autres vues pour répondre aux besoins des utilisateurs.

Le modèle 4+1 vue



Le modèle 4+1 vue

Vue des processus :
Importante dans les systèmes multi-tâches ou distribués (ex architecture client/serveur)
- décompose le système en processus,
- décrit les interactions entre processus,
- résout les problèmes de synchronisation.

Vue des processus

Vue de déploiement

Le modèle 4+1 vue

Vue de déploiement :
Décrit les ressources matérielles et la répartition du logiciel dans ces ressources :
- disposition et nature des matériels,
- répartition du logiciel sur les noeuds,
- exigences en termes de performances.

Vue logique

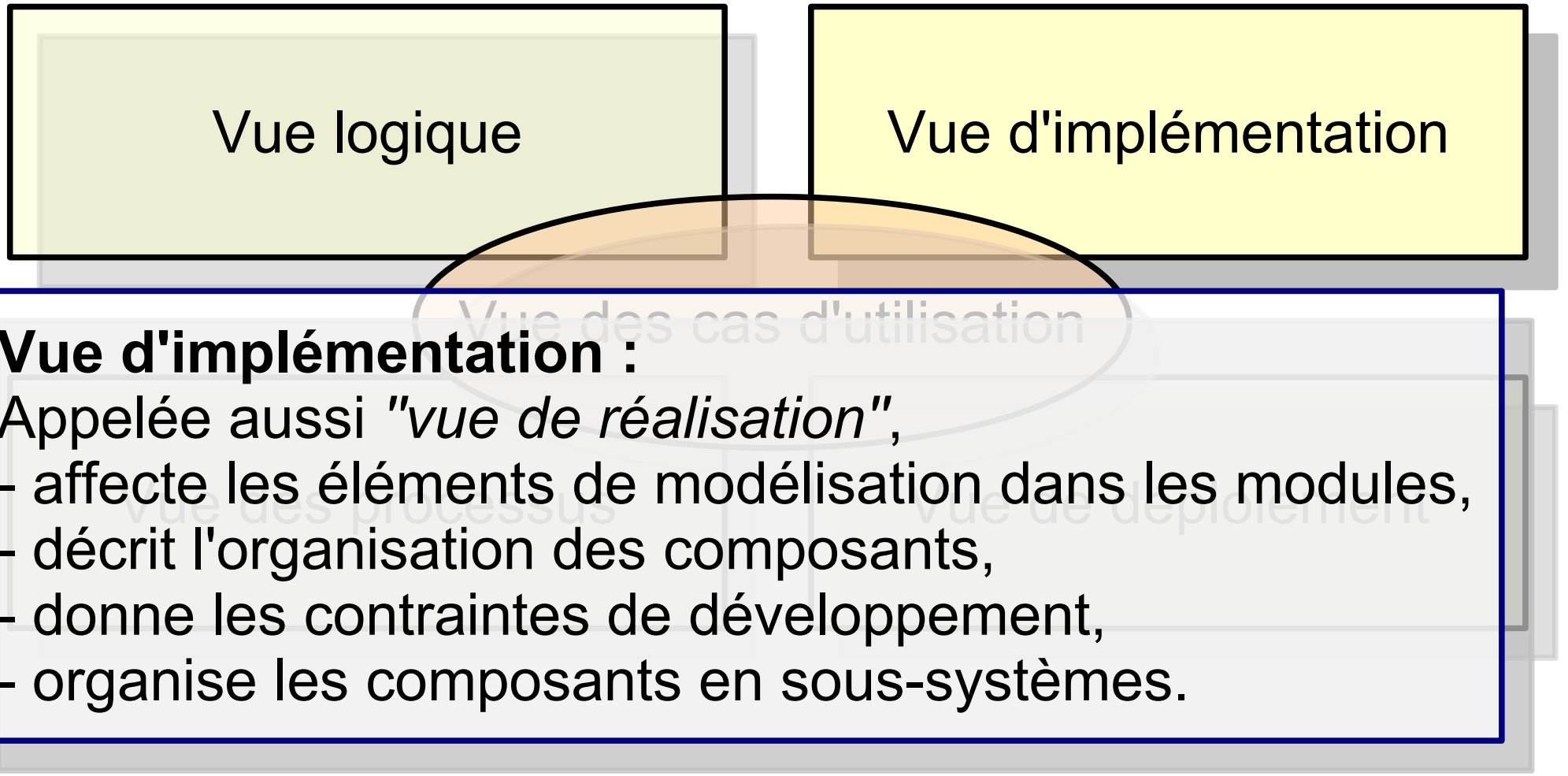
Vue d'implémentation

Vue des cas d'utilisation

Vue des processus

Vue de déploiement

Le modèle 4+1 vue



Vue d'implémentation :
Appelée aussi *"vue de réalisation"*,
- affecte les éléments de modélisation dans les modules,
- décrit l'organisation des composants,
- donne les contraintes de développement,
- organise les composants en sous-systèmes.

Plan

- Introduction générale,
- Meta-modèle UML,
- Notion de vues d'un système,
- **Diagrammes UML,**

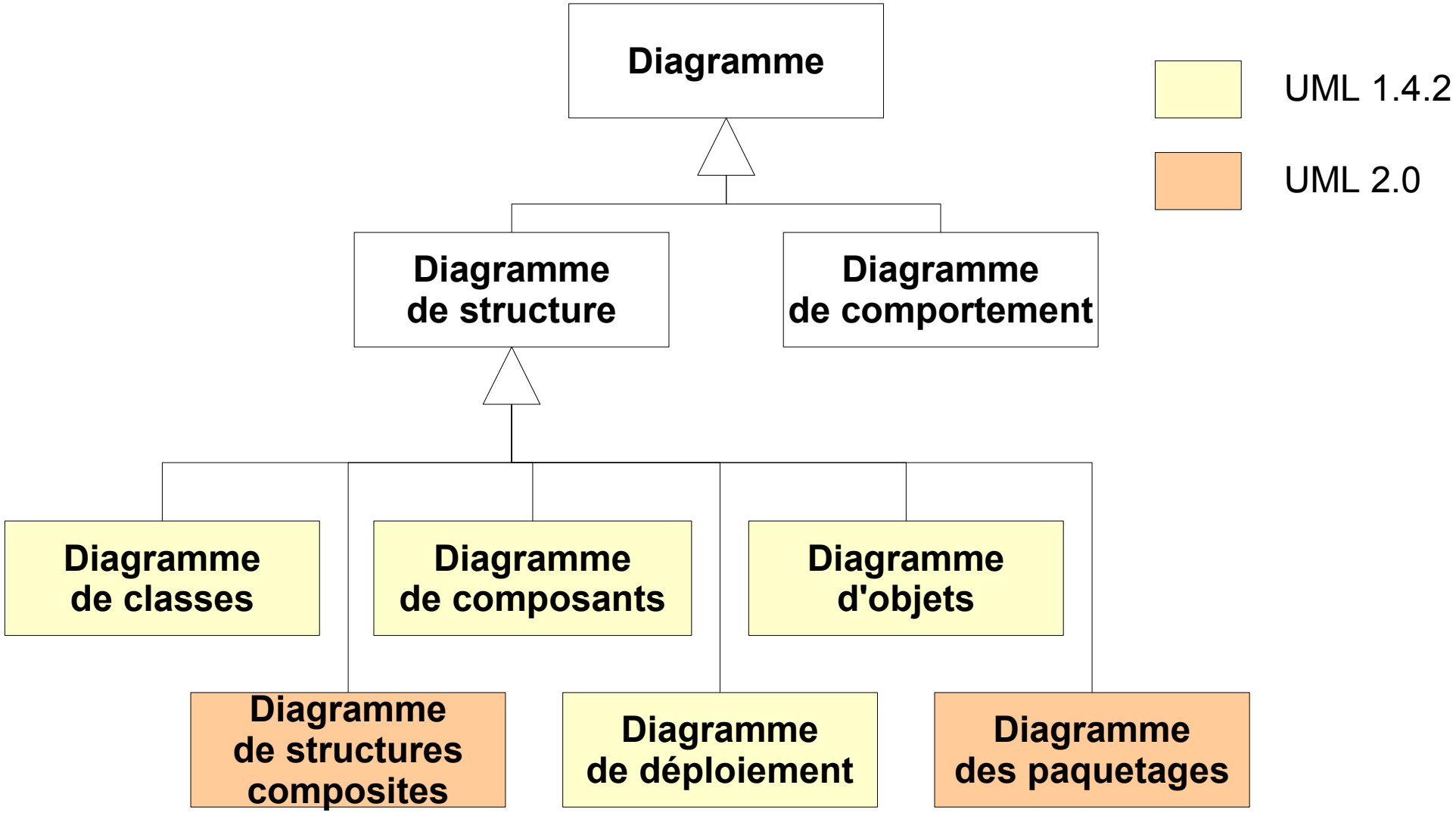
Modéliser avec UML

- UML définit un modèle à l'aide de *diagrammes*,
- Un diagramme = une représentation graphique d'un aspect du SI,
- Un diagramme = une structure précise,
- Un diagramme = une sémantique précise,
- Combinés, ces diagrammes donnent une vue globale du système selon le modèle 4+1 vues et selon les aspects statiques et dynamiques du modèle.

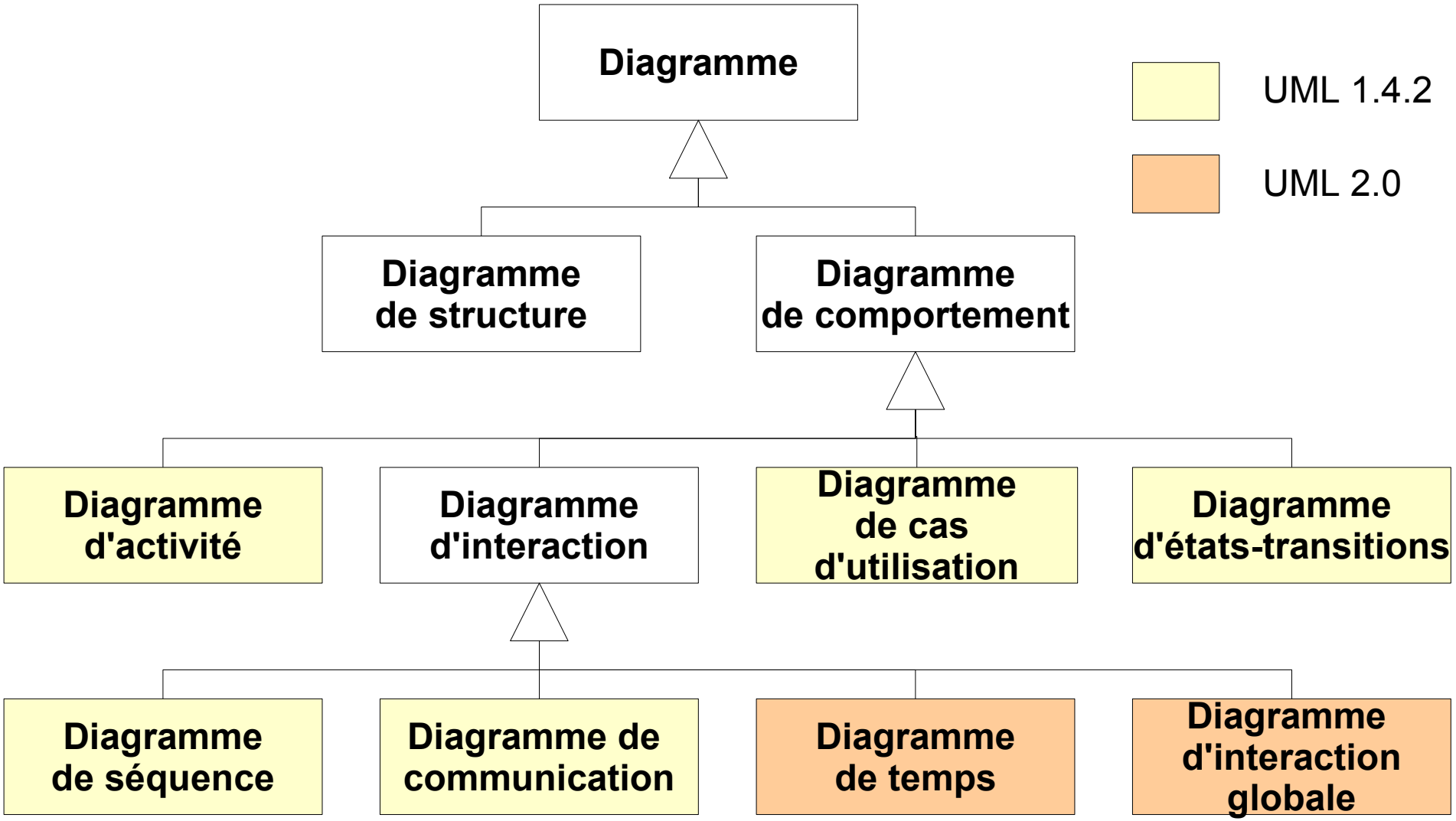
Caractéristiques des diagrammes

- Supportent l'abstraction :
  - Niveau de détail => niveau d'abstraction,
- Structure et notation **normalisée,**
- Diagrammes = graphiques !
  - Il existe des outils pour les dessiner et les gérer,
  - Ces outils permettent également, dans une certaine mesure, de générer du code à partir des diagrammes

Diagrammes UML



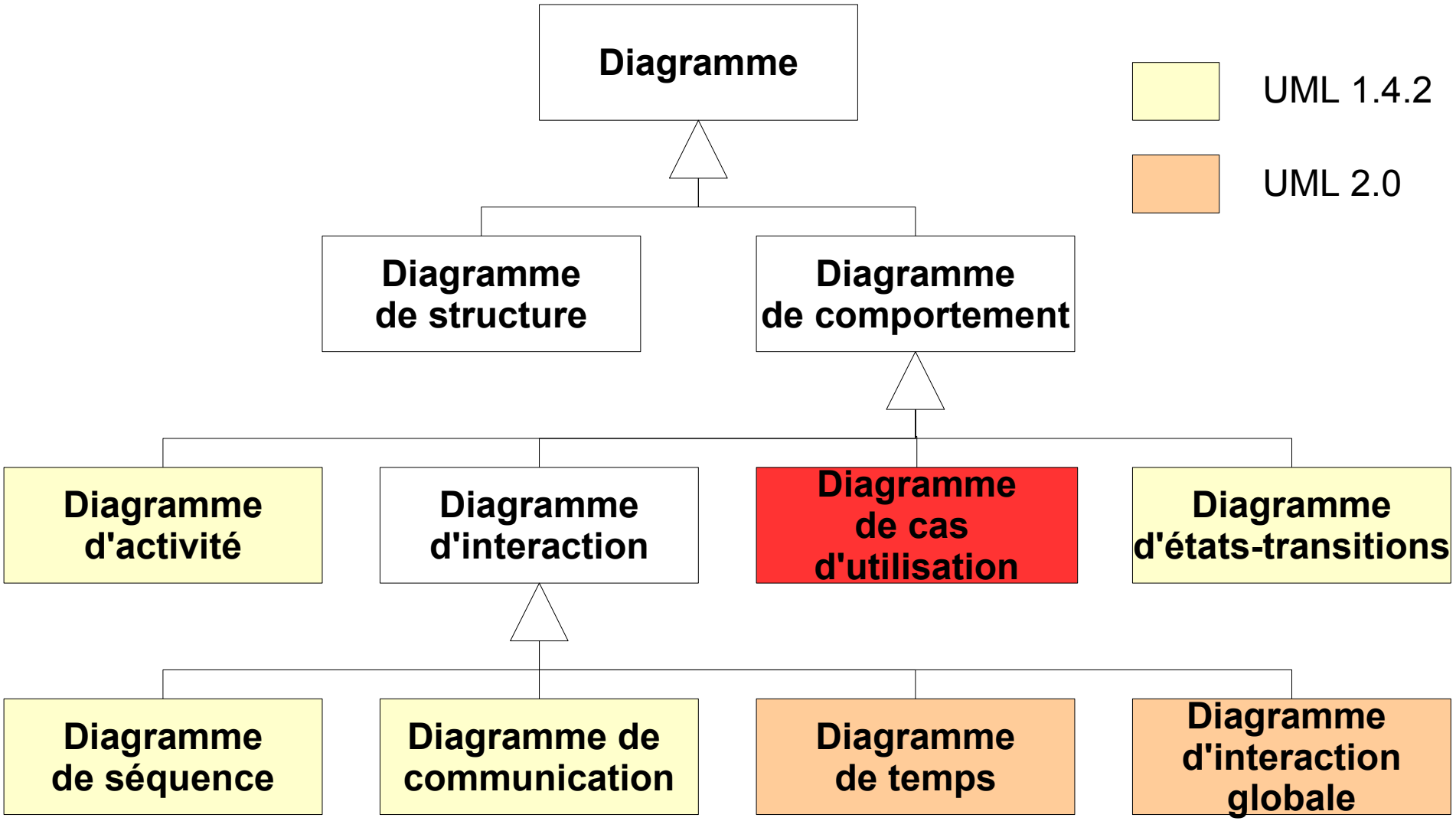
Diagrammes UML



Plan

- Introduction générale,
- Meta-modèle UML,
- Notion de vues d'un système,
- **Diagrammes UML,**
  - Diagramme des cas d'utilisation,
  - Diagramme des paquetages,
  - Diagramme de classes,
  - Diagramme d'objets,
  - Diagramme d'activités,
  - Diagramme d'états-transitions,

Diagrammes UML



Le modèle des cas d'utilisation

Solution représentant le modèle conceptuel :

- Spécifie les concepts utilisés pour définir les fonctionnalités d'une entité telle qu'un système,
- Définit le comportement d'une entité comme un système ou un sous-système sans spécifier sa structure interne,
- Focalisé sur l'objectif des fonctionnalités et limité aux préoccupations *"réelles"* des utilisateurs,
- Structure les besoins des utilisateurs et les objectifs correspondant du système,

Le modèle des cas d'utilisation

Conséquences :

- Identifie les utilisateurs (*les acteurs*) du système et leur interaction avec ce dernier,
- Classe les acteurs et structurent les objectifs du système,
- Sert de base à la traçabilité des exigences d'un système dans un processus de développement intégrant UML.

Les besoins de l'utilisateur et les objectifs du système sont la préoccupation majeure !!

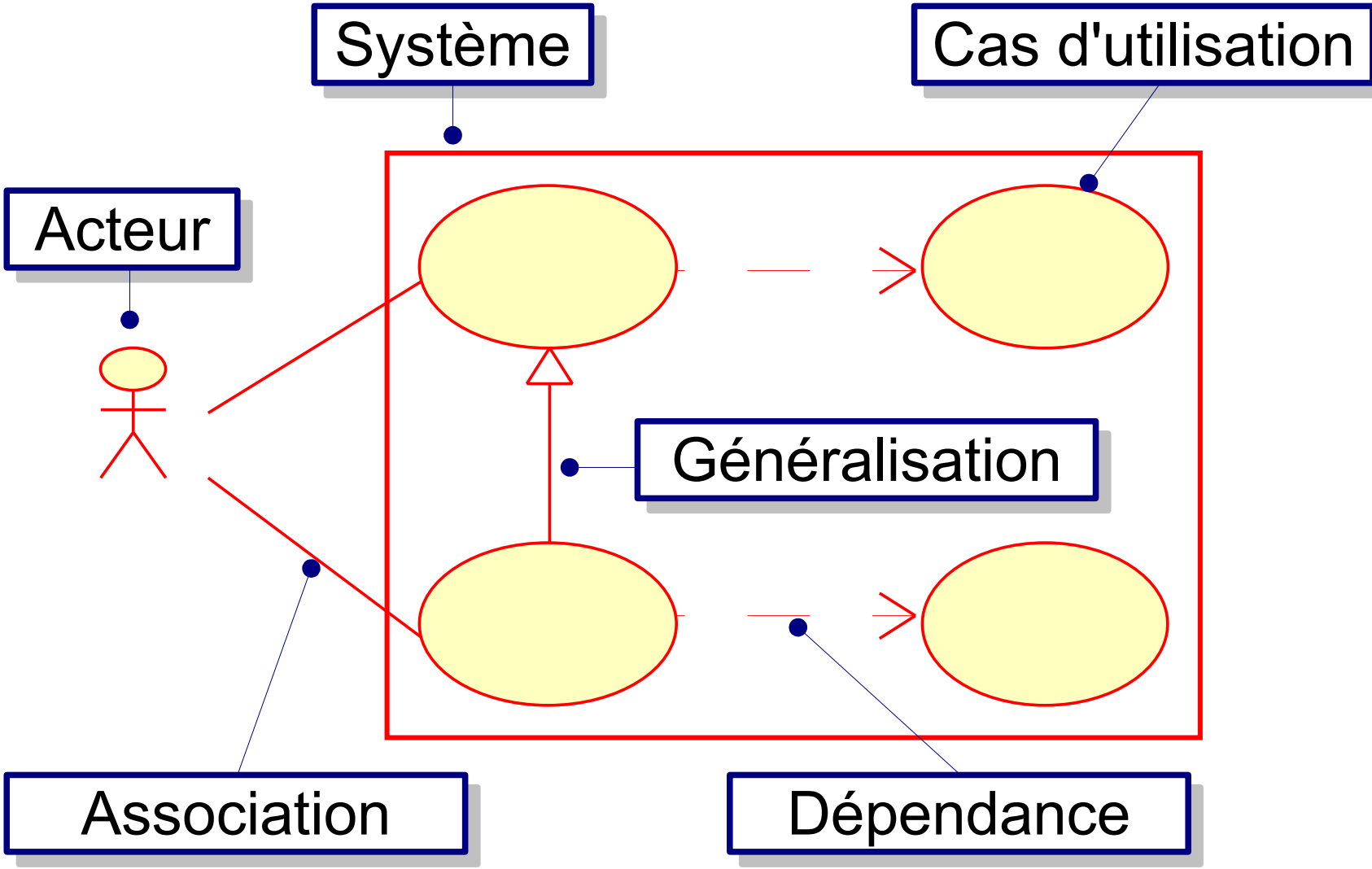
Éléments du diagramme de cas d'utilisation

- Diagramme composé de six éléments :
  - **Système** : fixe les limites du système entre les acteurs (externes) et les fonctions (internes),
  - **Acteur :** rôle joué par quelque chose qui intervient dans le fonctionnement du système. Il y a 4 catégories d'acteurs :
    - Les acteurs principaux (ex: usager, client, etc),
    - Les acteurs secondaires (ex: opérateur de maintenance, administrateur, etc),
    - Le matériel externe (ex: capteur, imprimante, etc),
    - Les autres systèmes (ex: serveur, etc),

Éléments du diagramme de cas d'utilisation

- **Cas d'utilisation :** identifie une fonction clé du système. Ce dernier doit pouvoir l'accomplir,
- **Dépendance :** identifie une relation de communication entre deux cas d'utilisation,
- **Généralisation :** définit une relation entre 2 acteurs ou 2 cas d'utilisations lorsque l'un d'entre eux hérite de l'autre,
- **Association :** identifie une interaction entre les acteurs et les cas d'utilisation. Elle doit être expliquée dans une description narrative qui fournit un ensemble de scénarios. Ces derniers jouent le rôle de test lors de l'évaluation de l'analyse, la conception et l'évaluation du cas d'utilisation.

Notations du diagramme de cas d'utilisation

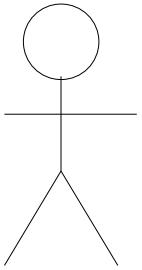


Le système de cas d'utilisation

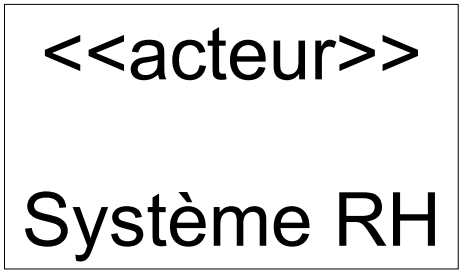
- Le système fournit un contexte dans lequel et autour duquel les éléments qui interviennent dans la construction du système sont placés,
- L'interface du système est ici importante, non la manière dont ce dernier est ensuite implémenté,
- Les interfaces sont les canaux de communication entre les acteurs du système et les éléments du système lui-même : les cas d'utilisation,
- Parfois, le système n'est pas représenté dans les diagrammes de cas d'utilisation.

Les acteurs du cas d'utilisation

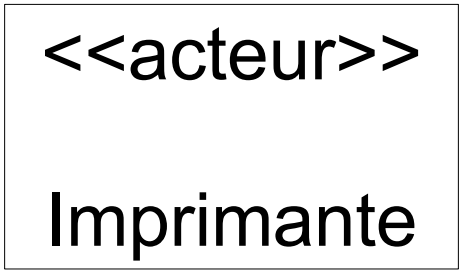
- Ce sont des personnes ou de systèmes,
- Plus exactement, il s'agit de rôles joués par une entité externe en relation avec le système.
- Les icônes employées varient :



Personne



Système



Matériel

Les cas d'utilisation

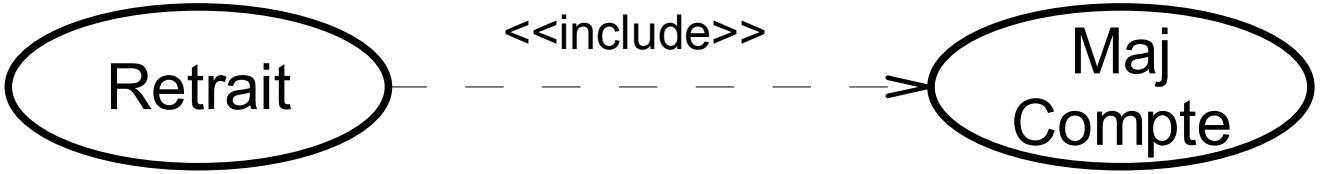
- Fonctions que le système doit accomplir,
- Syntagme verbal exprimant le but à accomplir,
  - Par exemple :
    - *"Déposer de l'argent",*
    - *"Retirer de l'argent",*
    - *"Mettre à jour le compte",*
- Cas d'utilisation : seulement les fonctions visibles du système (cad perçues par les acteurs).

Les associations

- Relie un acteur à un cas d'utilisation,
- Indique que l'acteur communique avec le cas d'utilisation,
- C'est une association bidirectionnelle,
  - L'acteur accède au cas d'utilisation,
  - Le cas d'utilisation fournit des fonctionnalités à l'acteur.

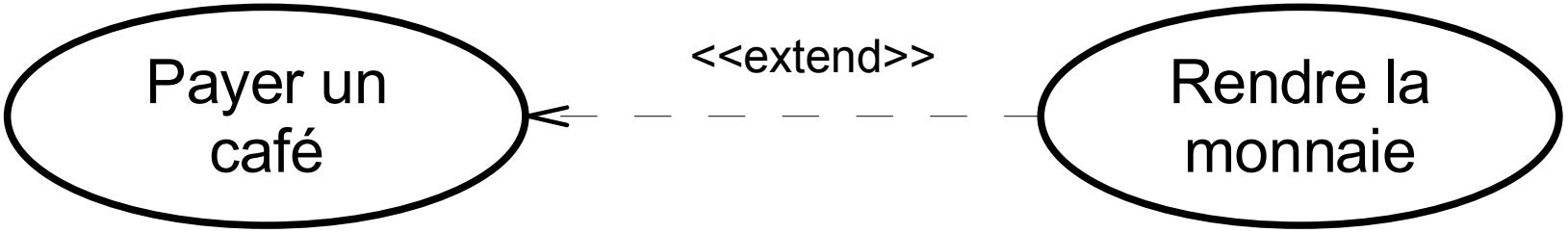
Les stéréotypes et les dépendances

- Stéréotypes : notés <<*stéréotype*>>,
  - Etendent UML sans le modifier,
  - Qualificateurs des éléments d'un modèle,
  - Donnent plus d'information,
  - N'agissent pas sur l'implémentation des éléments.
- Stéréotype <<*include*>> :
  - Indique qu'un cas d'utilisation délègue une tâche à un autre : dépendance forte !



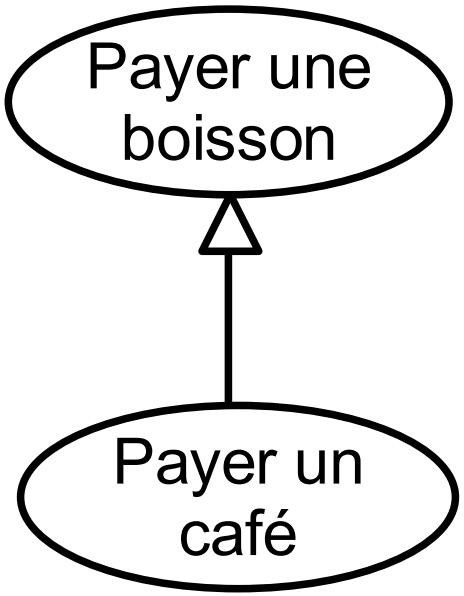
Les stéréotypes et les dépendances

- Stéréotype <<*extend*>>,
  - Indique qu'un cas pourrait avoir besoin d'un autre,
  - Dépendance faible car conditionnelle,
  - Le sens de la flèche est inversé par rapport au stéréotype <<include>>,
  - Indique que le cas source étends les objectifs du cas destination,



La généralisation

- Notion d'*héritage* appliquée aux cas d'utilisation,
- Appelé aussi relation *"est un"*,
- Ligne pleine terminée par un triangle vide,
- Le triangle est toujours du côté de l'origine de l'héritage.



Construction du diagramme des cas d'utilisation

1. Définition du contexte,
2. Identification des acteurs,
3. Identification des cas d'utilisation,
4. Définition des associations entre cas et acteurs,
5. Trouver les améliorations possibles,
6. Evaluation des dépendances <<include>>,
7. Evaluation des dépendances <<extend>>,
8. Trouver les opportunités de généralisation.

La description narrative

- Elle accompagne le modèle des cas d'utilisation.
- Plusieurs méthodologies existent pour créer la description narrative,
- Les éléments communs sont les suivants :
  - Présupposés,
  - Préconditions,
  - Démarrage,
  - Processus ou dialogue,
  - Arrêt,
  - Postconditions

Eléments de la description narrative

- Présupposés :
  - État du système requis par le cas d'utilisation pour permettre son application,
- Préconditions :
  - Tests préliminaires effectués par le cas d'utilisation avant son démarrage,
- Démarrage :
  - Recenser les actions lançant le cas d'utilisation,
  - Tous les acteurs ne démarrent pas nécessairement le cas d'utilisation de la même manière !

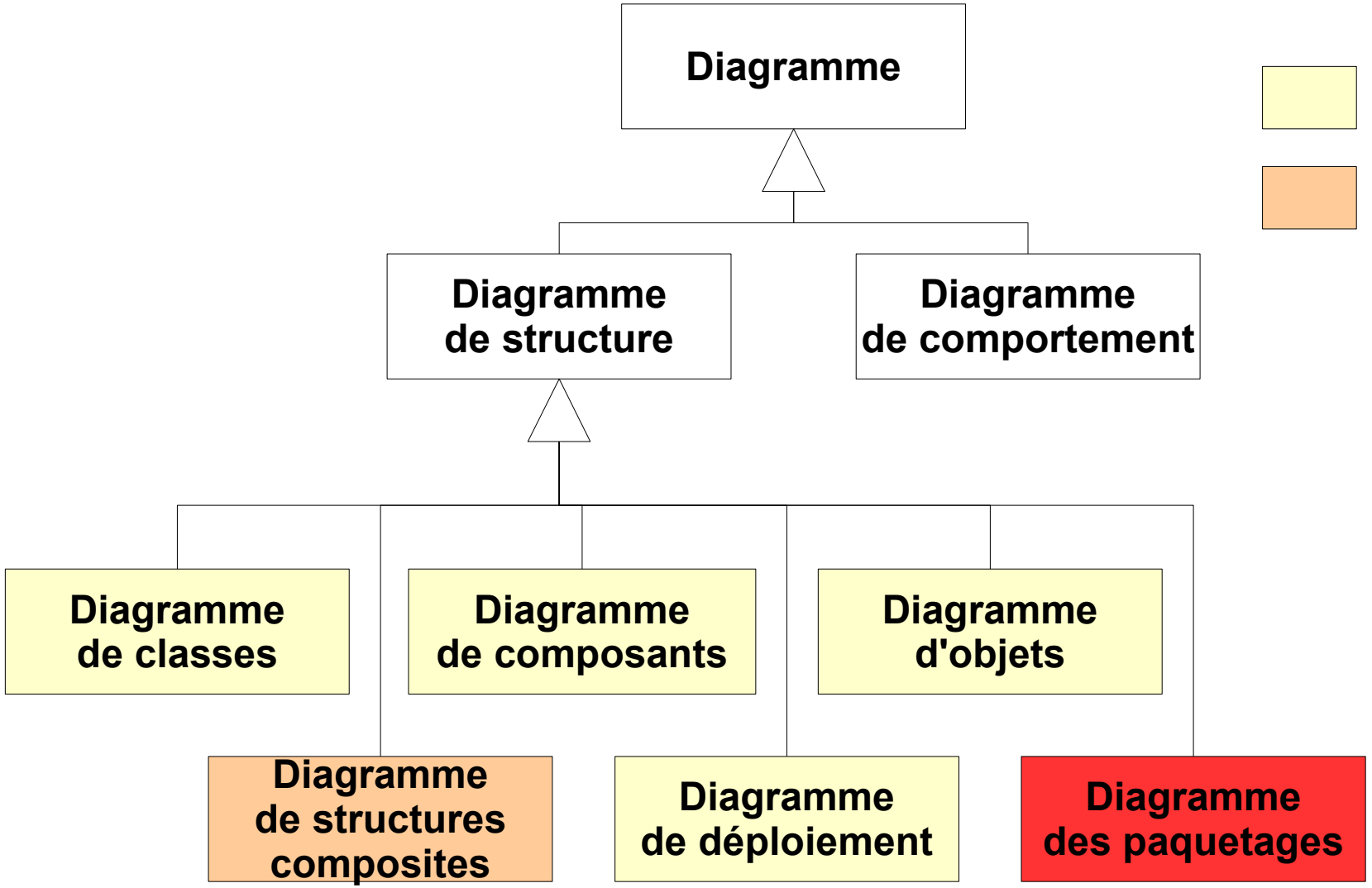
Eléments de la description narrative

- Dialogue :
  - Description pas à pas de la conversation entre le cas d'utilisation et l'acteur,
  - Cette séquence est souvent modélisée à l'aide d'un diagramme d'activité,
  - Permet de découvrir les étapes nécessaires ainsi que les variantes de dialogue.
- Arrêt :
  - Conditions d'arrêt (arrêt normal, erreurs, etc ...)
- Postconditions :
  - Etat dans lequel le système se trouve à l'arrêt du cas.

Les scénarios de cas d'utilisation

- Complètent l'étude des cas d'utilisation,
- Un scénario : un déroulement possible du cas,
- Plusieurs scénarios pour un cas d'utilisation,
- Identifiés à partir d'un description narrative ou à partir d'un diagramme d'activité décrivant le cas,
- Les scénarios aident à construire les plans de test des cas d'utilisation,
- Lors de leur identification, éviter les redondances
  - Décomposer en chemin indépendants.

Diagrammes UML



Les paquetages (packages)

- Éléments d'organisation du modèle,
- Regroupent les éléments de modélisation suivant des critères logiques,
- Encapsulent les éléments de modélisation,
- Structurent un système en catégories et sous-systèmes,
- Constituant de base d'une architecture,
- Bon niveau de granularité pour la réutilisation,
- Sont aussi des espaces de noms.

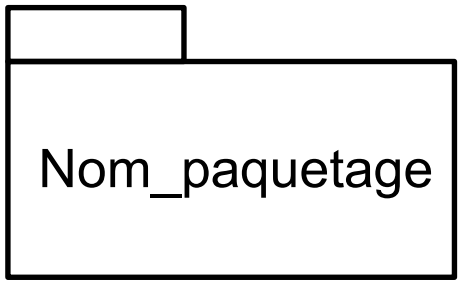
Le diagramme de paquetage

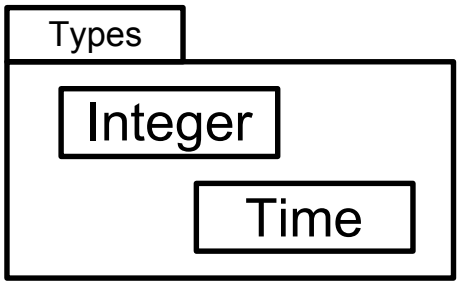
- N'apparaît pas dans UML 1.4.2,
  - Pourtant les paquetages sont définis et utilisés !
  - Déjà représentés en pratique,
  - UML 2.0 les introduit de manière cadrée et structurée.

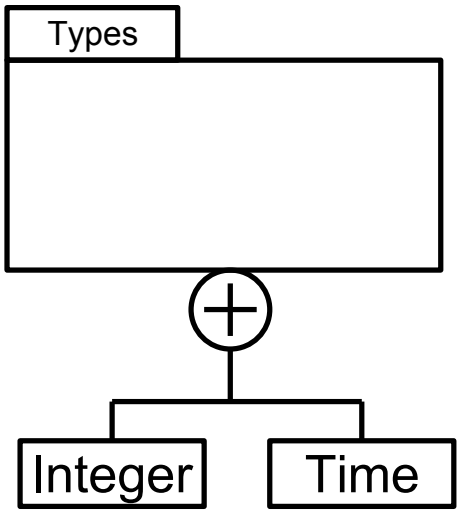
- Éléments constitutifs :
  - Paquetages,
  - Dépendances, et dépendances stéréotypées :
    - Fusions (PackageMerge),
    - Imports (PackageImport).

Relations entre paquetages

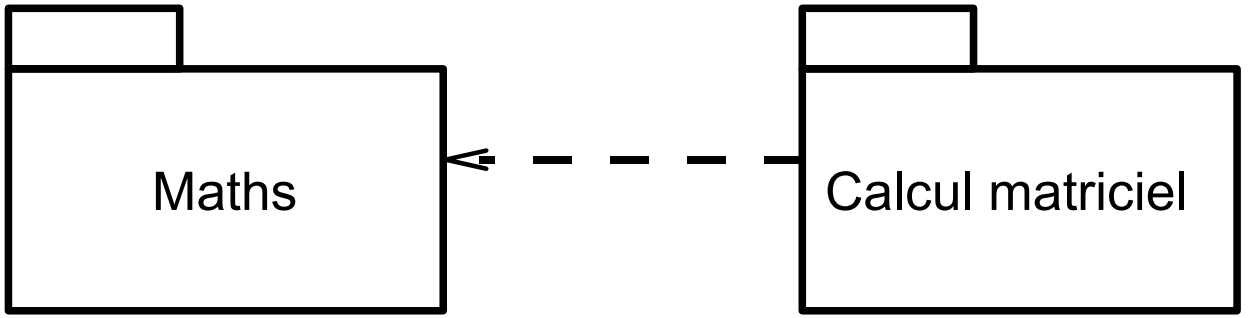
- Représentation d'un paquetage :





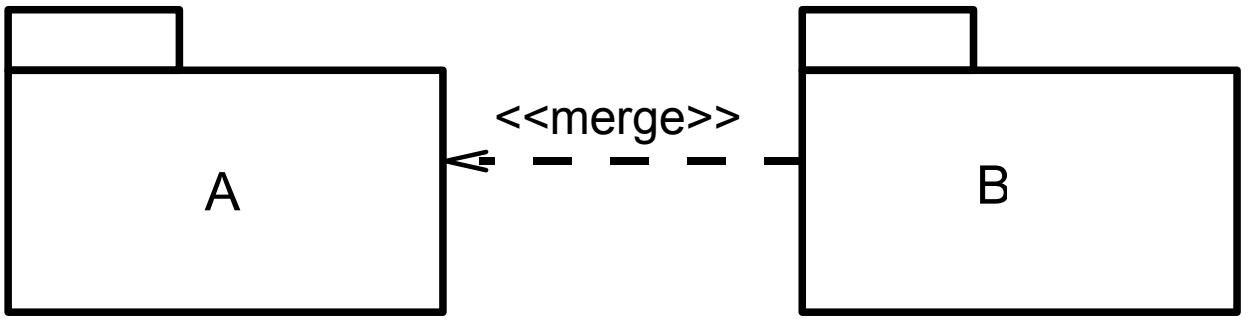


- Dépendance :



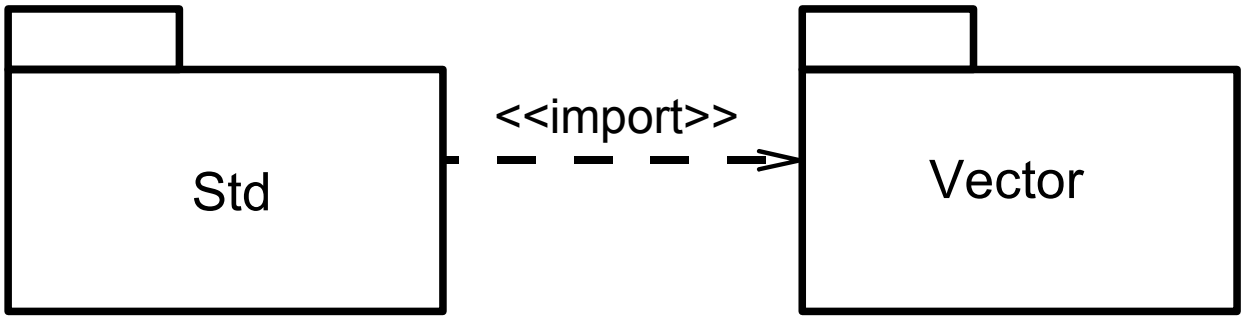
Relations entre paquetages

- Fusion de paquetages :
  - Le contenu des deux paquetages est combiné,
  - Deux définitions différentes d'un même concept représentant deux aspects différents,
  - Il s'agit d'un mécanisme d'extension,



Relations entre paquetages

- Import de paquetages :
  - Import des éléments du paquetage dans l'espace de nommage du paquetage de destination,
  - Import public ou privé :
    - Stéréotype <<*import*>> pour import public,
    - Stéréotype <<*access*>> pour import privé,



Diagrammes UML

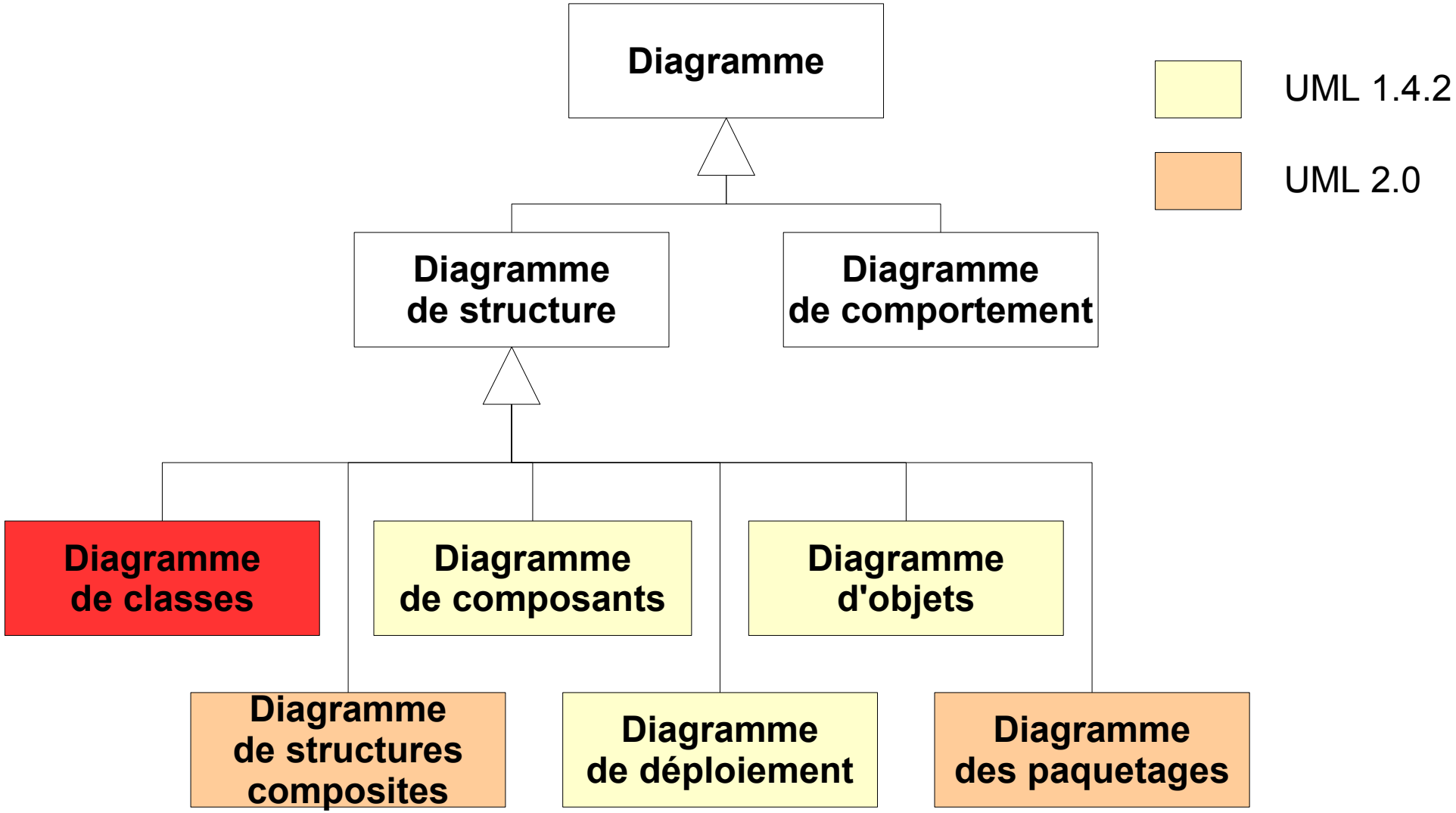


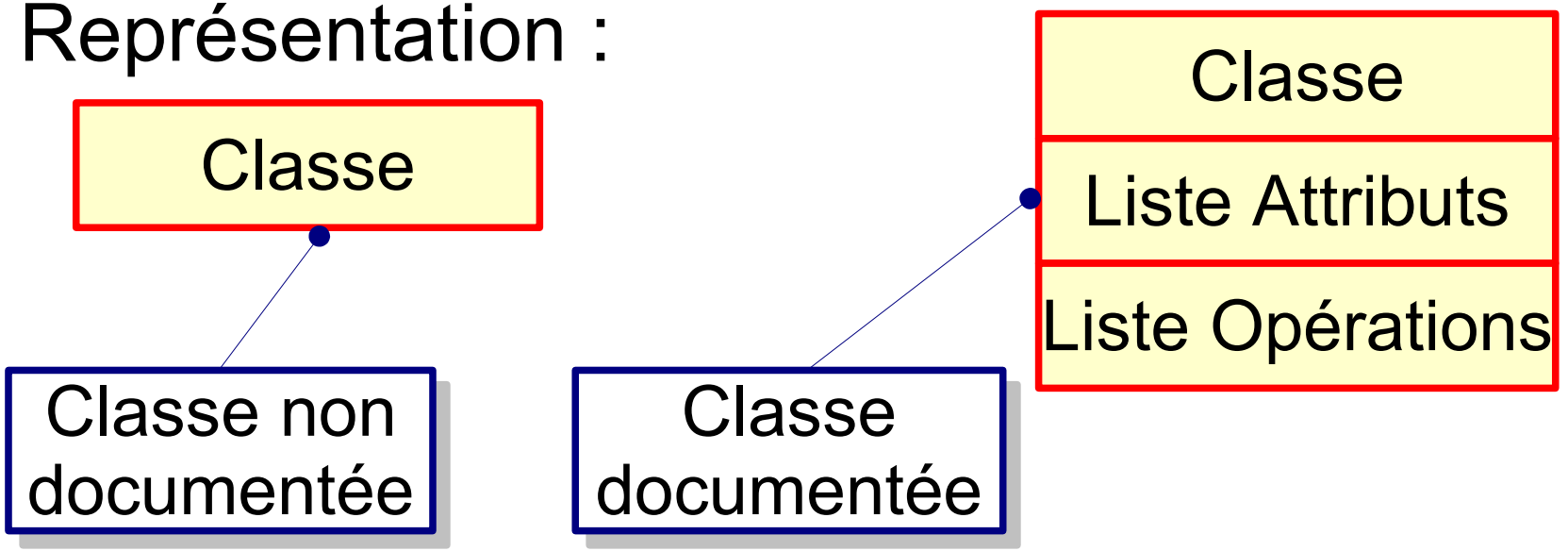
Diagramme de classes

- Le plus connus des diagrammes de conception orientée objets,
  - Utilisé ultérieurement pour la génération automatique de code,
  - Important pour le développement,

- Représente les classes, leur composants et la façon dont ces classes sont reliées,
- Une classe : définition d'un type d'objet

Les classes

- Classe : type abstrait caractérisé par des propriétés (*membres*) :
  - *Attributs* : données,
  - *Opérations* : (méthodes) traitements appliqués sur les données.
- Représentation :



Attributs d'une classe

- Attribut = nom + type,
- Visibilité :
  - Public (+) :
    - accès possible par n'importe qui,
  - Private (-) :
    - accès de l'intérieur de l'objet,
  - Protected (#) :
    - accès possible pour un objet d'une sous-classe,
  - Package (~) :
    - accès pour les objets appartenant au même package

Attributs d'une classe

- Attribut dérivé (/) :
  - Valeur calculée à partir d'autres attributs,
- Contraintes ({...}):
  - Régles nécessaires pour garantir l'intégrité de l'attribut,
- Attribut de classe (souligné),
- Un attribut a un nom unique dans une classe,
- Représentation :

  visibilité/nom: type = valeur par défaut {contraintes}

Création d'une spécification d'attribut

1. Créer l'attribut,

   matière

2. Ajouter le type de données,

   matière: caractères

3. Ajouter la valeur par défaut éventuelle,

   matière: caractères = IN91

4. Indiquer les contraintes,

   matière: caractères = IN91 {4 à 5 caractères}

5. Ajouter la visibilité,

   -matière: caractères = IN91 {4 à 5 caractères}

Opérations d'un classe

- Opérations : Nom + arguments + valeur retour,
- Notations communes avec les attributs,
  - Visibilité (+ - # ~),
  - Contraintes ({...}),
  - Opération de classe (soulignée),

- Représentation :

  visibilité nomOperation(nomArgument: type {contraintes},...): type valeur retour {contraintes}

Création d'une spécification d'opération

1. Nommer l'opération,

   div_entiere

2. Définir les arguments,

   div_entiere (num: entier, den: entier)

3. Définir le type de la valeur de retour,

   div_entiere (num: entier, den: entier): entier

4. Identifier et écrire les contraintes,

   div_entiere (num: entier, den: entier {≠ 0}): entier

5. Indiquer la visibilité de l'opération,

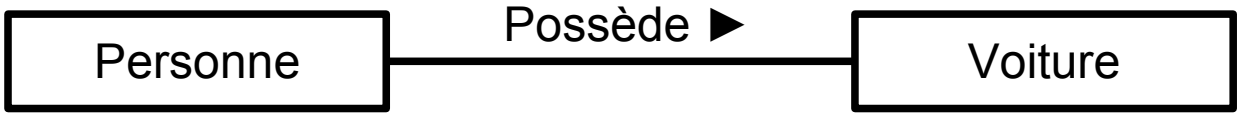
   +div_entiere (num: entier, den: entier {≠ 0}): entier

Représentation complète d'une classe

Fraction
- numerateur : int = 0 - dénominateur : int = 1{!=0} - /quotientEntier : int = 0
+ getNumerateur() : int + setNumerateur(numerateur : int = 0) : void + getDenominateur() : int + setDenominateur(denominateur : int = 1 {!=0}) : void + getQuotientEntier() : int

Associations de classes

- Décrite par un verbe ou un syntagme verbal,
- Nécessite 4 éléments :
  - Les classes participantes,
  - L'association représentée par une ligne,
  - Le nom de l'association,
  - La direction de lecture.

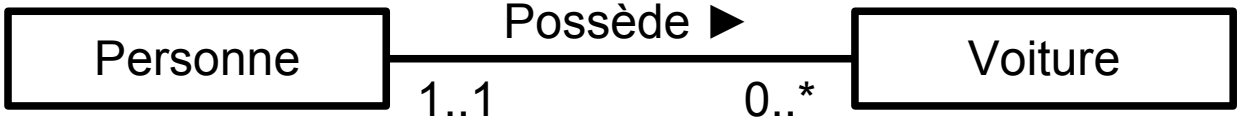


Associations de classes

- Les cardinalités (*multiplicities*):
  - Affectées à chacune des classes participant à l'association,
  - Exprimées de différentes manières :

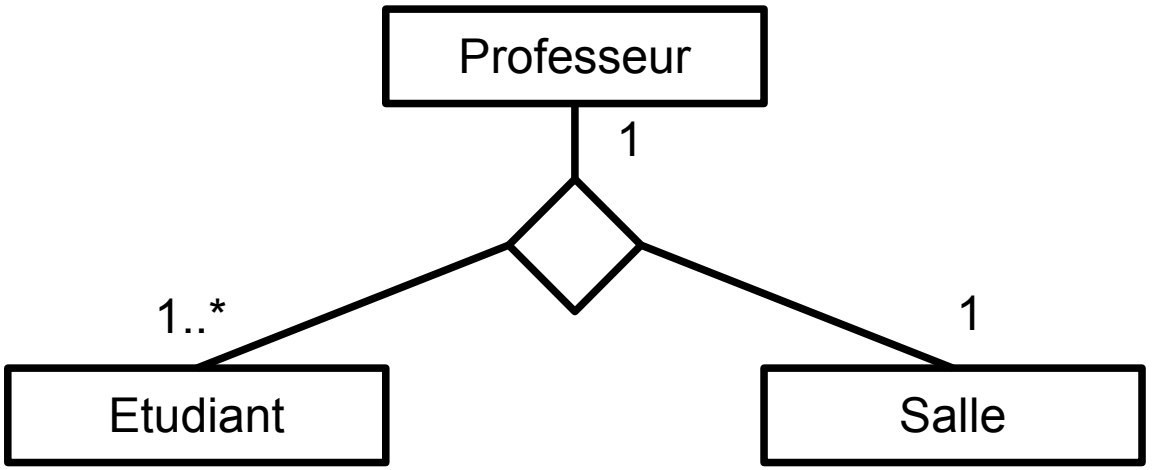
    Minimum..Maximum

    val1,val2,val3
  - Indications inversées par rapport à la notation Merise !



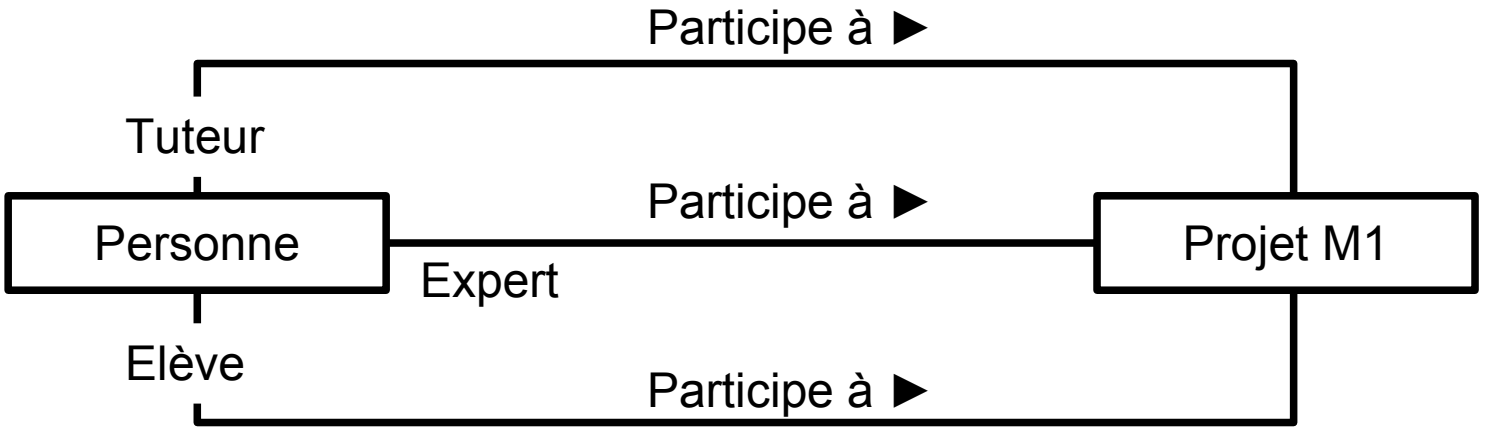
Associations de classes

- Associations n-aires :
  - Relient plus de 2 classes,
  - Sont peu utilisées en raison de leur complexité,
  - Souvent promues au rang de classe d'association,



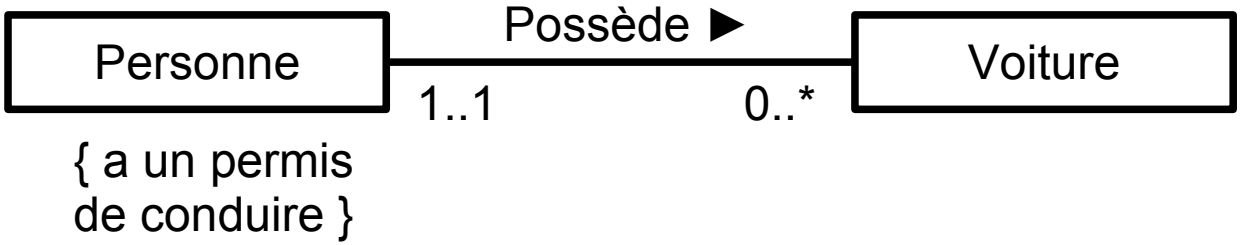
Associations de classes

- Utilisation des rôles :
  - Permet de qualifier au mieux la relation,
  - Les rôles engendrent du code pour permettre la différenciation

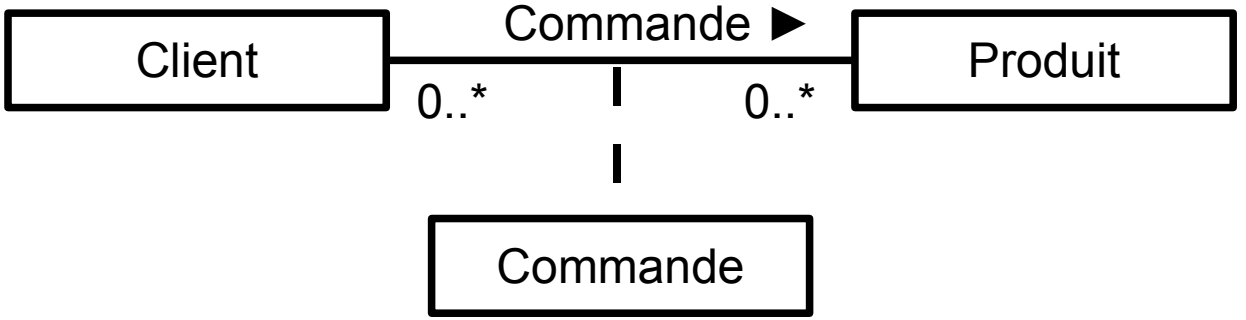


Associations de classes

- Contraintes d'association :
  - Notées entre accolades :

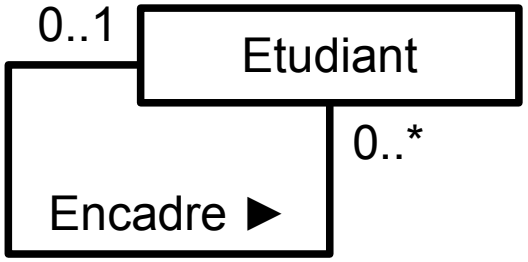


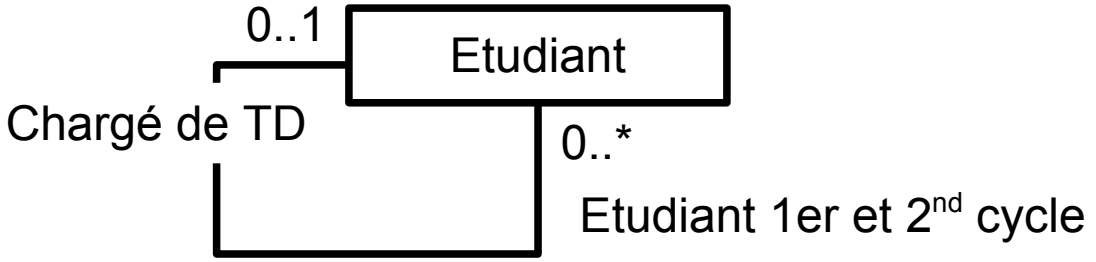
- Les classes d'associations :



Associations de classes

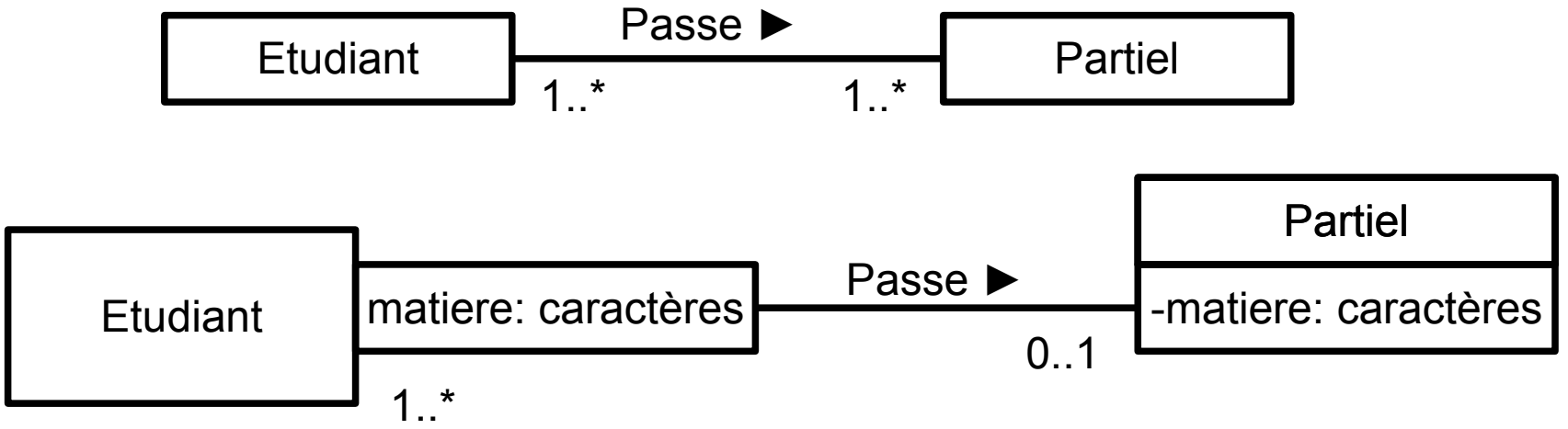
- Associations réflexives :
  - S'applique à des objets d'une même classe,
  - Peut s'exprimer à l'aide de rôles ou d'un syntagme verbal,





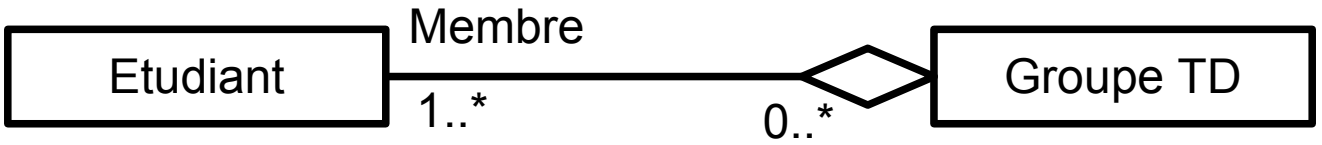
Association de classes

- Associations qualifiées :
  - Fonctions similaires à celles des index,
  - Un objet utilise le qualificateur pour accéder à l'autre type d'objet qualifié,
  - Le qualificateur est généralement un nom d'attribut de l'autre classe.



Agrégations de classes

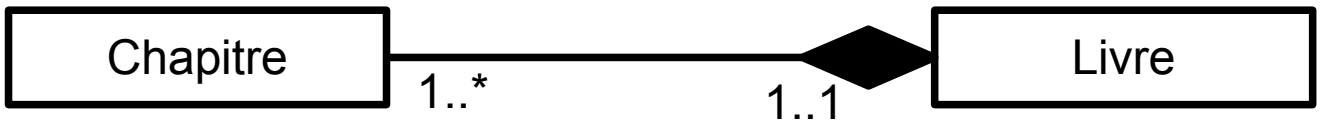
- L'Agrégation est un type d'association,
- Les objets impliqués sont assemblés ou configurés pour créer de nouveaux objets,
  - Il s'agit d'une entité unique,
  - Le contrôle est assuré par un des objets,
  - L'intégrité de la configuration est protégée
- Représentation : un losange vide
  - Les rôles et les contraintes sont conservées



Compositions de classes

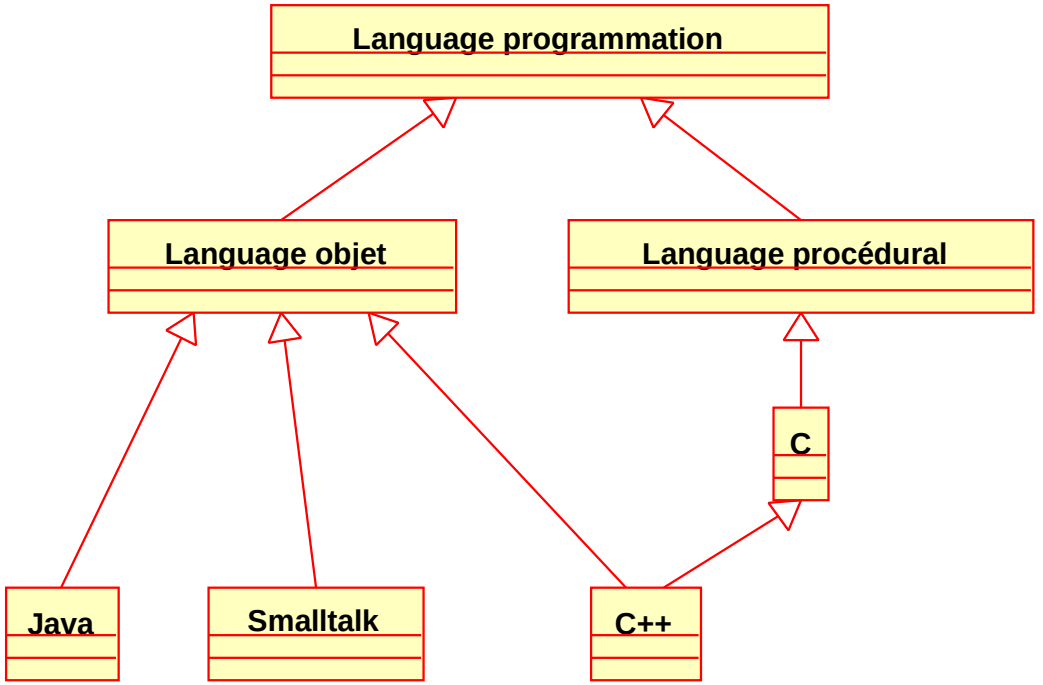
- Forme d'agrégation :
  - Chaque partie n'est membre que d'un agrégat,
  - La suppression de l'objet maître entraîne la suppression des autres parties de l'agrégat

- Représentation : un losange rempli



Généralisation

- La *généralisation* est aussi appelée *héritage*,
- La généralisation n'est pas une association,
- Elle lie les classes lorsqu'une classe contient un sous-ensemble d'élément nécessaire à une autre.



Généralisation : définitions

- Superclasse : classe dont on hérite,
- Sous-classe : classe qui hérite d'une superclasse,
- Classe abstraite : classe contenant une interface mais ne pouvant s'instancier,
- Classe concrète : classe contenant une implémentation,
- Discriminant : attribut ou règle décrivant la façon dont est identifié l'ensemble des sous-classes d'une superclasse.

Généralisation et discriminant

- Notation des discriminants :
  - Placé sur la ligne reliant les sous-classes à leur superclasse

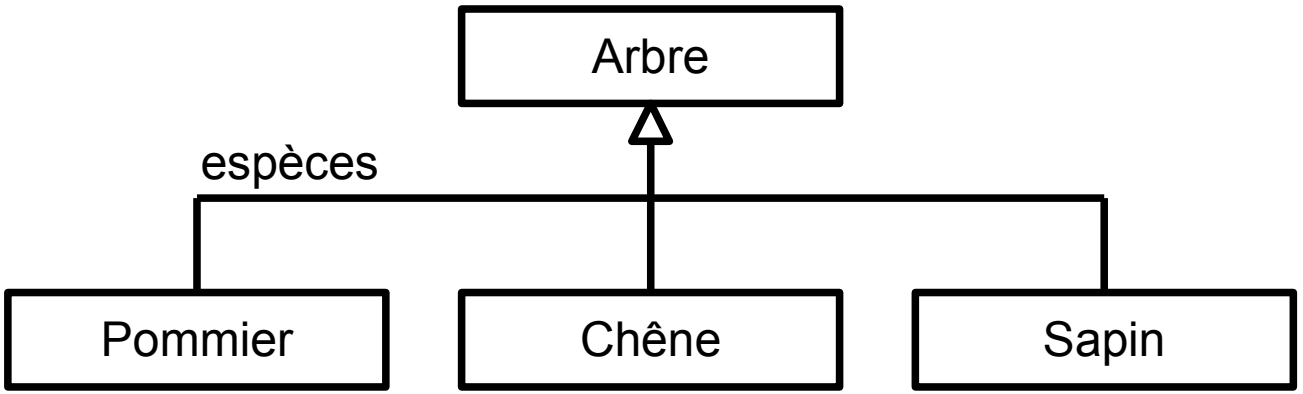


Diagramme de classes

- Le diagramme de classes : un rôle central au niveau du développement. Il est épaulé par les autres diagrammes :
  - Les cas d'utilisation aident à déterminer les classes,
  - Les diagrammes de séquence et de collaborations aident à déterminer les interfaces des objets,
  - Le diagramme d'activité décrit le comportement interne des objets et l'enchaînement des opérations.
- Ce diagramme est testé à l'aide du diagramme des objets.

Diagrammes UML

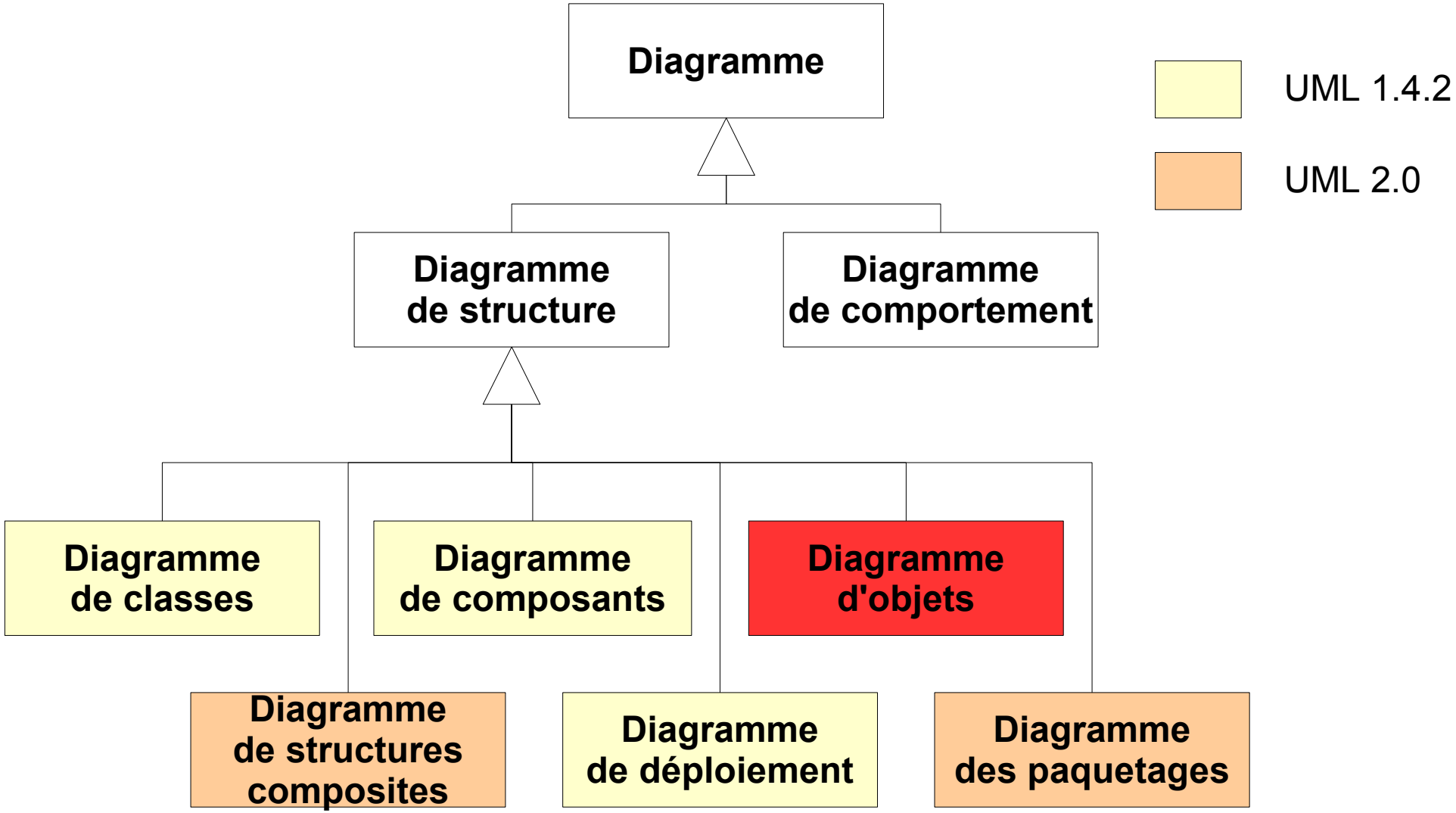


Diagramme d'objets

- Outil de recherche et de tests :
  - Aide à la conceptualisation par le biais d'exemples,
  - Aide à tester le modèle donné par le diagramme des classes.
- Diagramme de classes : règles définissant les relations entre les classes,
- Diagramme des objets : faits tangibles et réels.
  - Objets : instances de classes,
  - Sont utilisés pour monter un contexte particulier,

Diagramme d'objets

- Deux éléments :
  - Les objets (instances de classes),
  - Les liens (relations sémantiques) entre ces objets.
- Représentations d'une instance :
  - Nom : *identifiant: classe,*
  - 2 compartiments seulement : nom et attributs

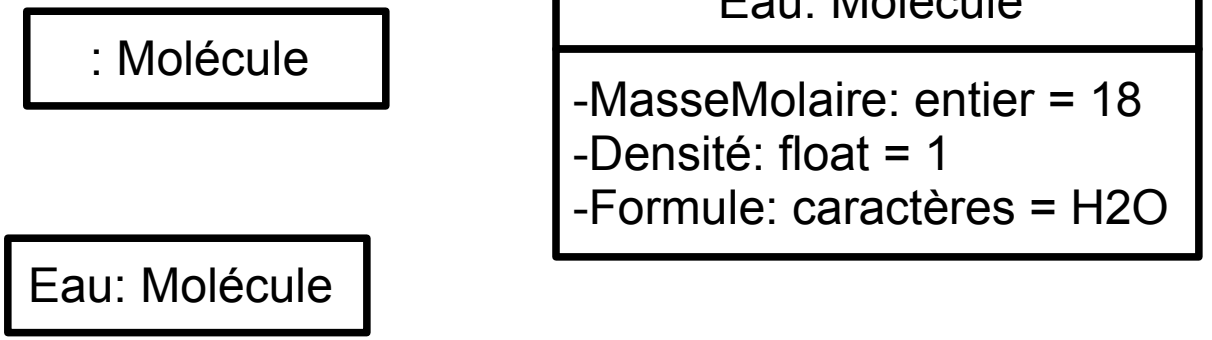
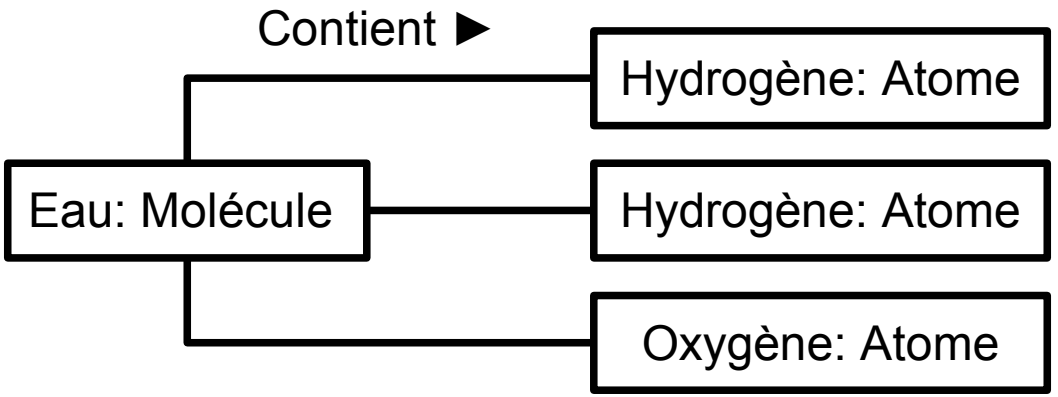


Diagramme d'objets

- Représentation des liens :
  - Les cardinalités "éclatent",
  - Les liens sont de un à un,
  - Les rôles peuvent être utilisés,
  - Les liens peuvent être nommés



Diagrammes UML

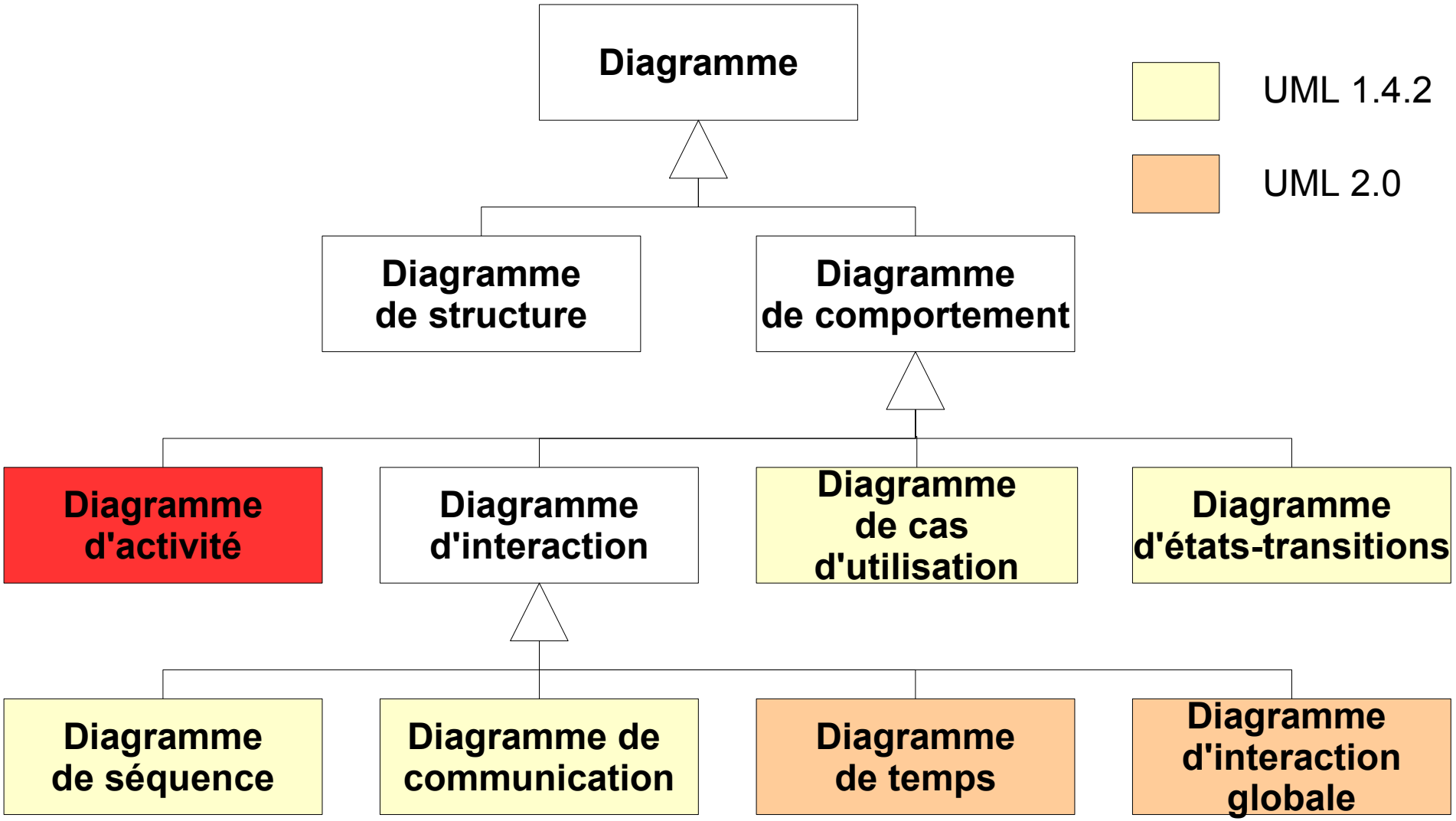
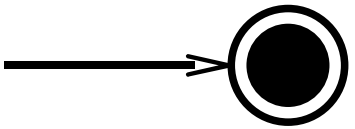


Diagramme d'activité

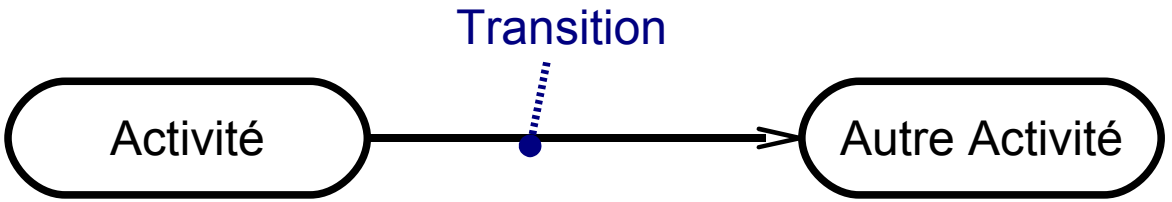
- Représentation graphique du comportement d'une méthode ou du déroulement d'un cas d'utilisation :
  - Le diagramme d'activités est une variante du diagramme état-transitions,
- Une activité :
  - représentation d'exécution d'un mécanisme, une liste d'étapes se déroulant de manière séquentielle,
- Passage d'une activité à une autre : transition,
  - Transitions déclenchées par la fin d'une activité et provoquent le début d'une autre,

Activités et transitions

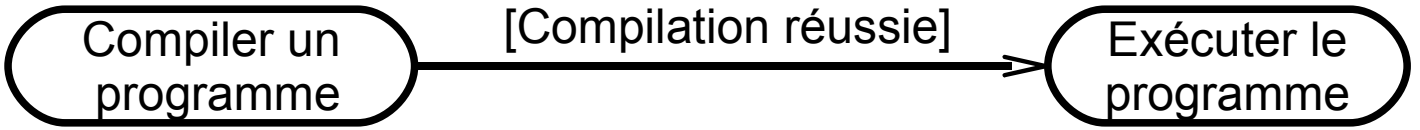
- Début et fin :
  - Point de départ : 
  - Point d'arrivée (peuvent être multiples) :



- Notation simple :

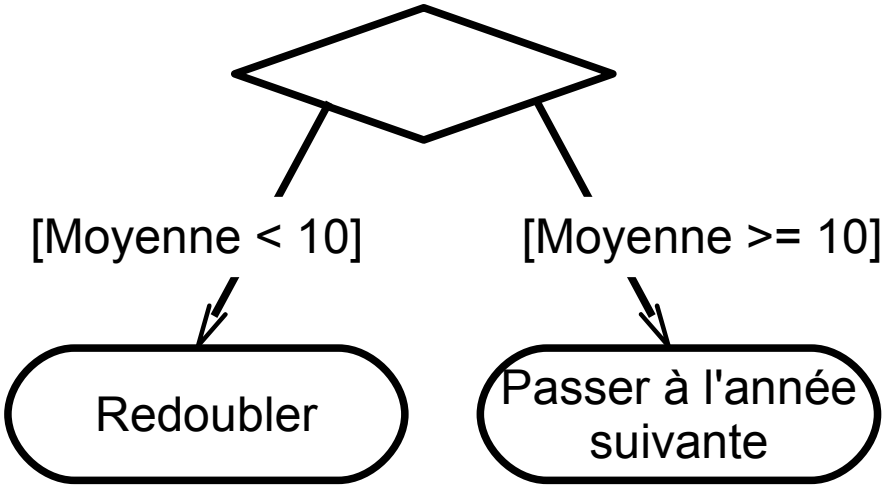


- Condition surveillée :

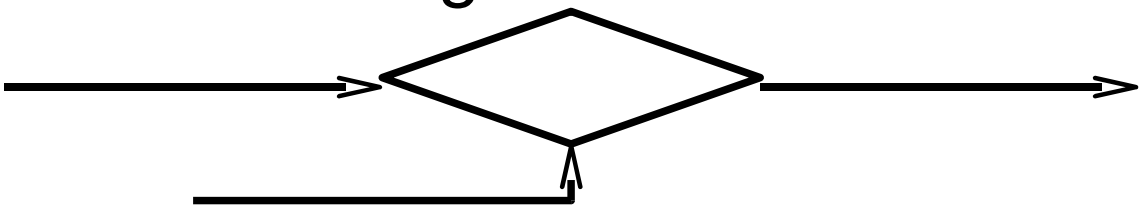


Structures de décision

- Représentent des alternatives :
  - La décision est représentée par un losange,
  - Les transitions issues du losange sont conditionnelles et exclusives

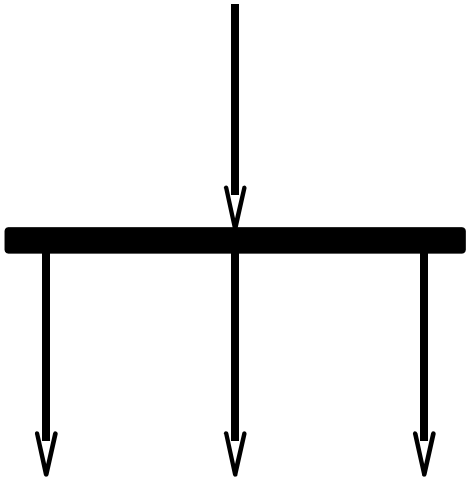


  - Points de convergence :

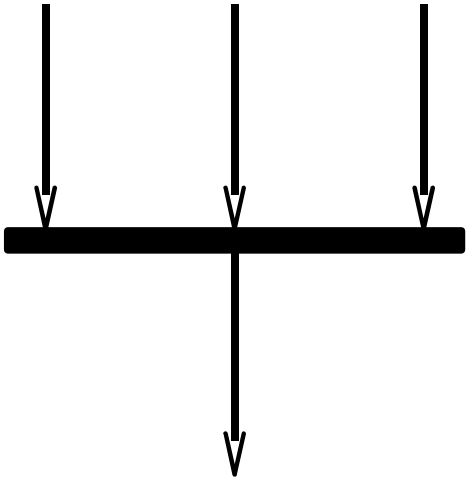


Parallélisme

- Possibilité de modéliser plusieurs processus fonctionnant en parallèle (processus concurrents),



Fourches (fork)



Synchronisation

Parallélisme

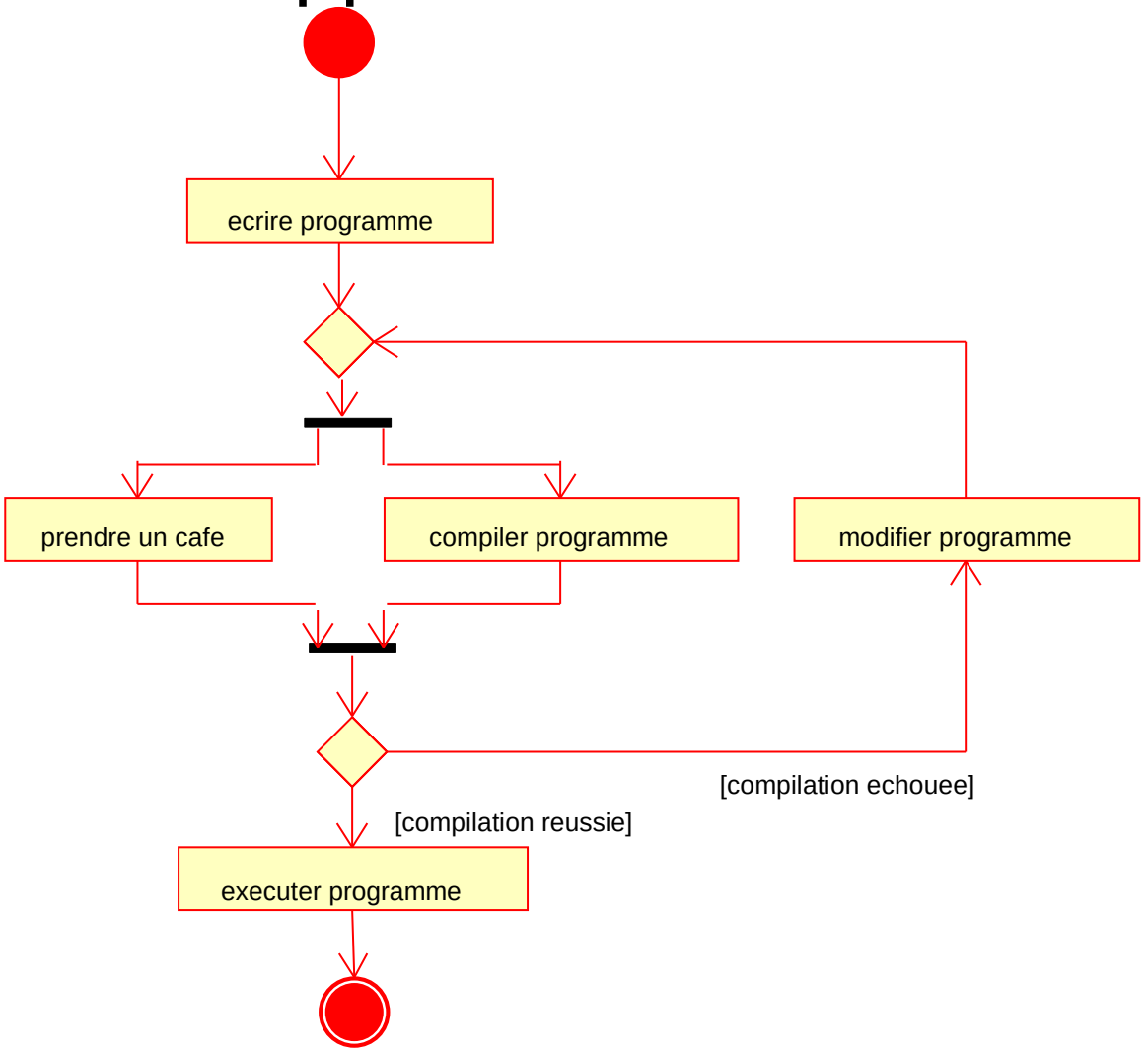
- Les *fourches* et *synchronisations* sont aussi appelée "*barres de synchronisation*",

- Une fourche déclenche l'exécution en parallèle,
  - Les transitions sortantes ont lieu en même temps,

- La synchronisation attend que toutes les branches parallèles se soient exécutées,
  - La transition sortante n'est franchie que si les transitions entrantes sont réalisées.

Diagramme d'activités

- Cycle de développement



Diagrammes UML

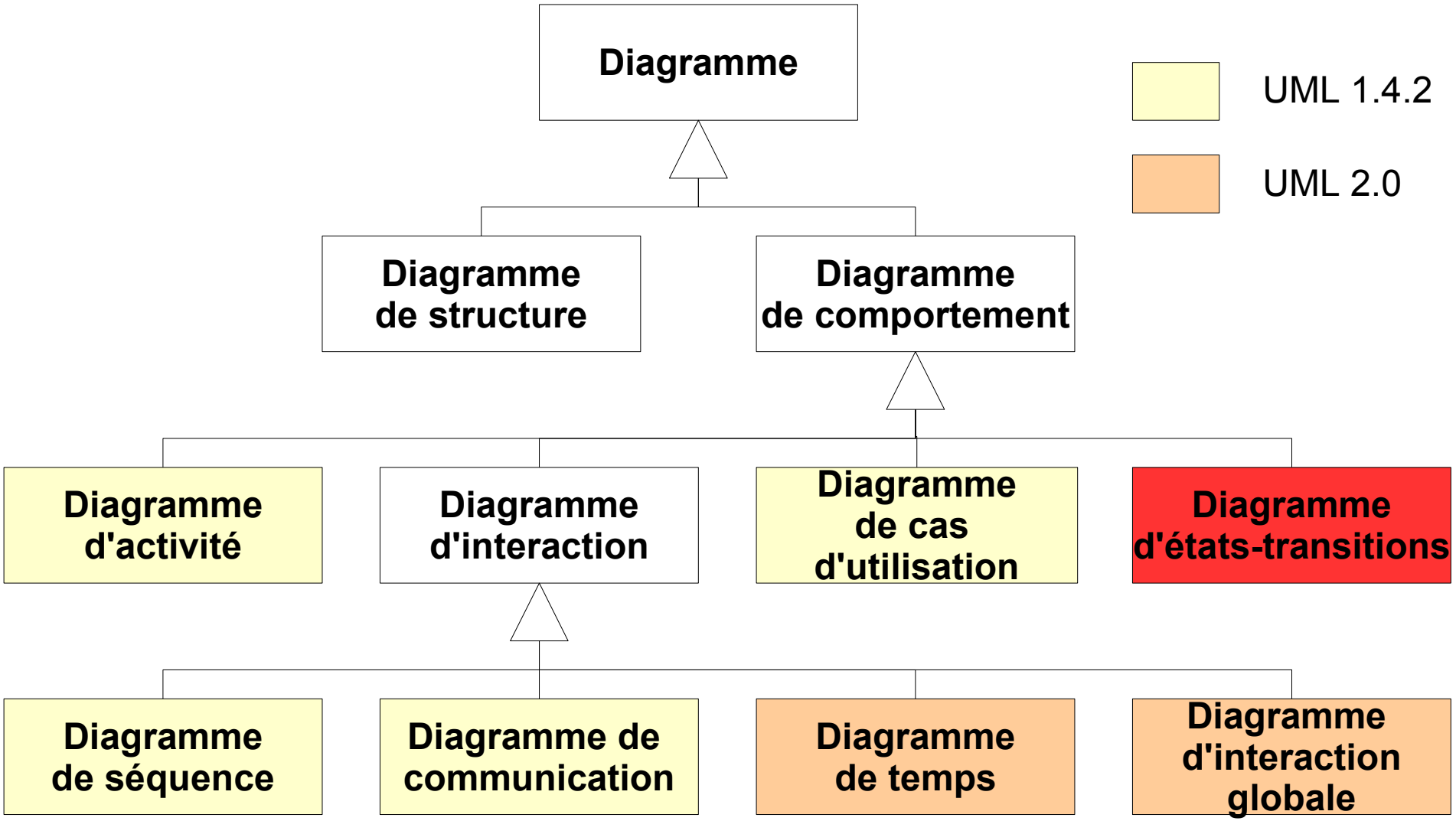


Diagramme d'état-transitions

- Vu précédemment : le diagramme d'activités est une variante du diagramme d'état-transition,
- Décrit la vie d'un objet, son état interne,
- Identifie les évènements internes et externes qui modifient l'état de l'objet.

Diagramme d'états-transitions

- Représente des automates d'états finis,
  - Représentation états/transitions,
- Un état se caractérise par sa durée et sa stabilité,
- Transition :
  - passage instantané d'un état à un autre,
  - déclenchée par un événement,
  - peut être conditionnelle,

Diagramme d'états-transitions

- Notations communes avec le diagramme d'activités :
  - Début et fin,
  - Notation équivalente entre état et activité,
  - Transition conditionnelle,
  - Les barres de synchronisation.

Diagramme d'états-transitions

- Super-état :
  - Élément de structuration,
  - Englobe d'autres états et transitions,

- Historique :
  - Mémorise le dernier sous-état actif d'un super-état pour y revenir ultérieurement,
  - Notation :

Diagramme d'états-transitions

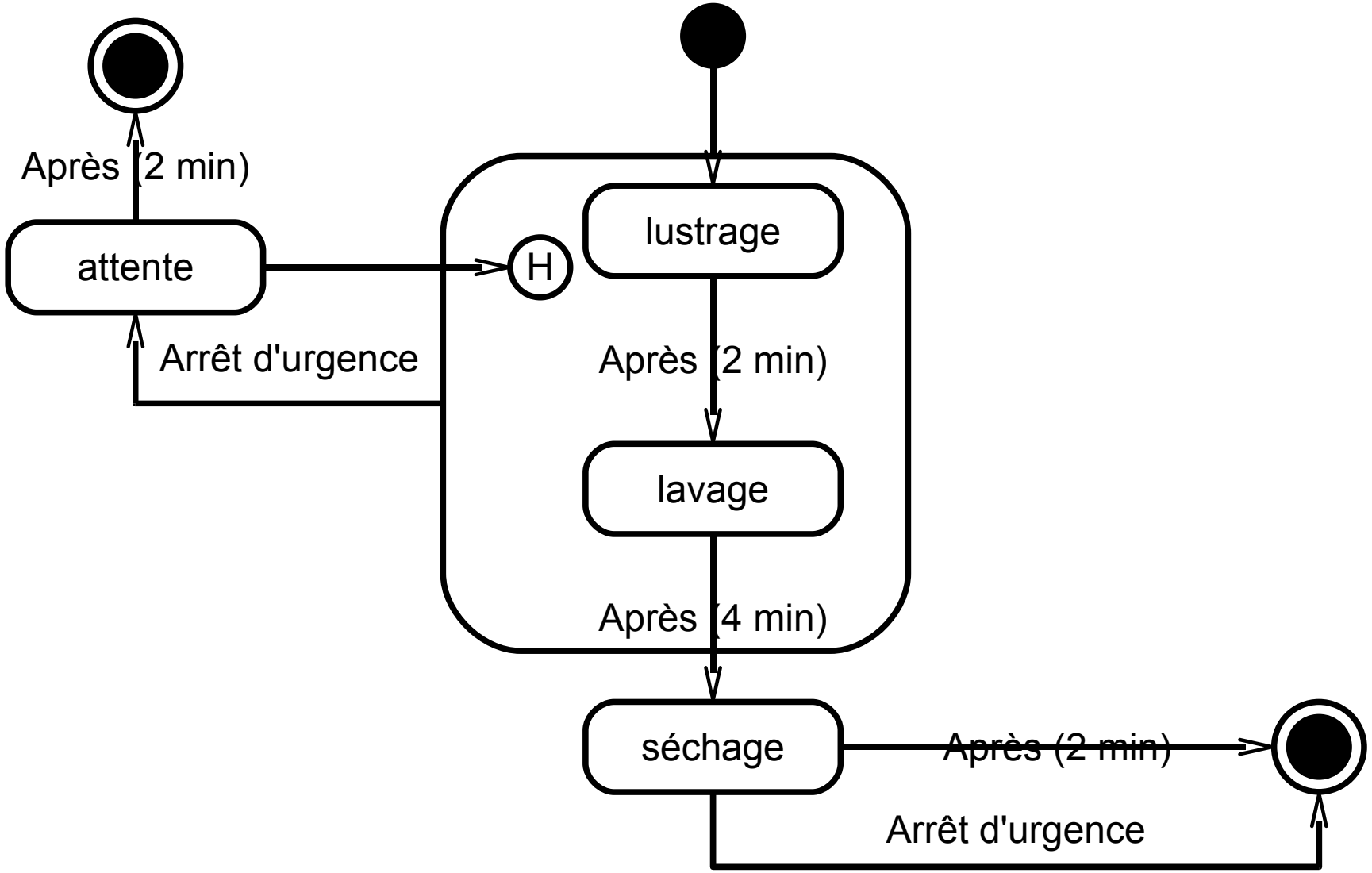
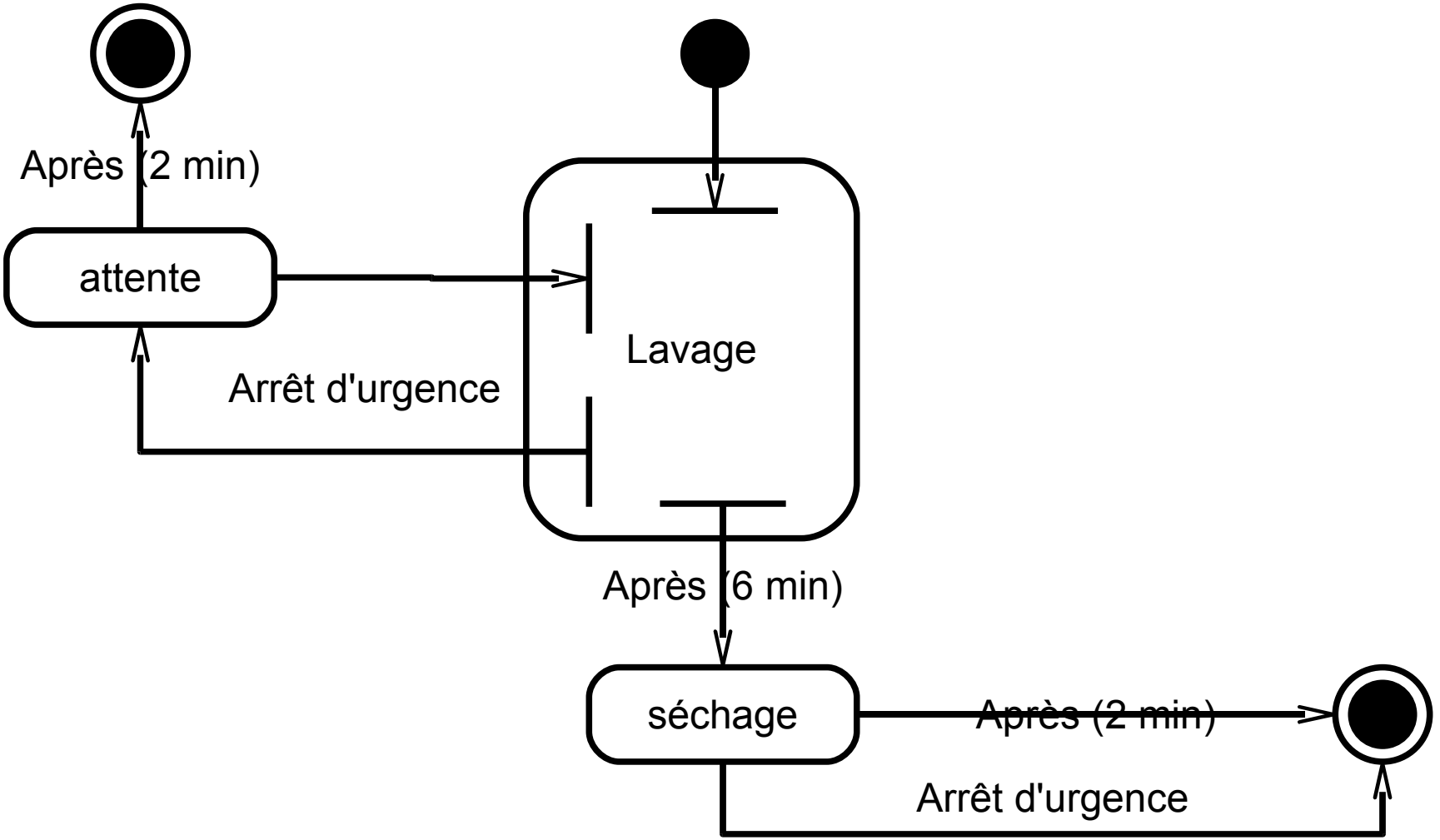


Diagramme d'états-transitions

- Les souches simplifient le diagramme :

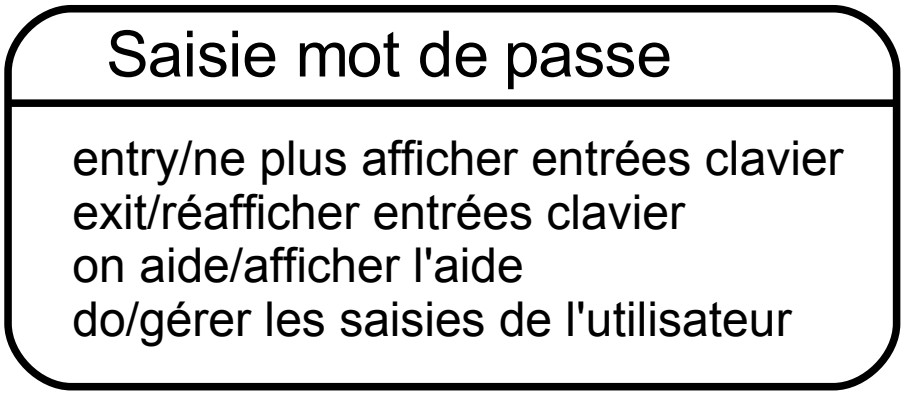


Actions attachées

- Une action peut être associée à l'évènement qui déclenche une transition,
- La transition entraîne l'exécution de l'action spécifiée sur l'objet,
- Une action correspond à une opération disponible dans l'objet dont on représente les états,
- Les actions peuvent être documentées directement à l'intérieur d'un état,

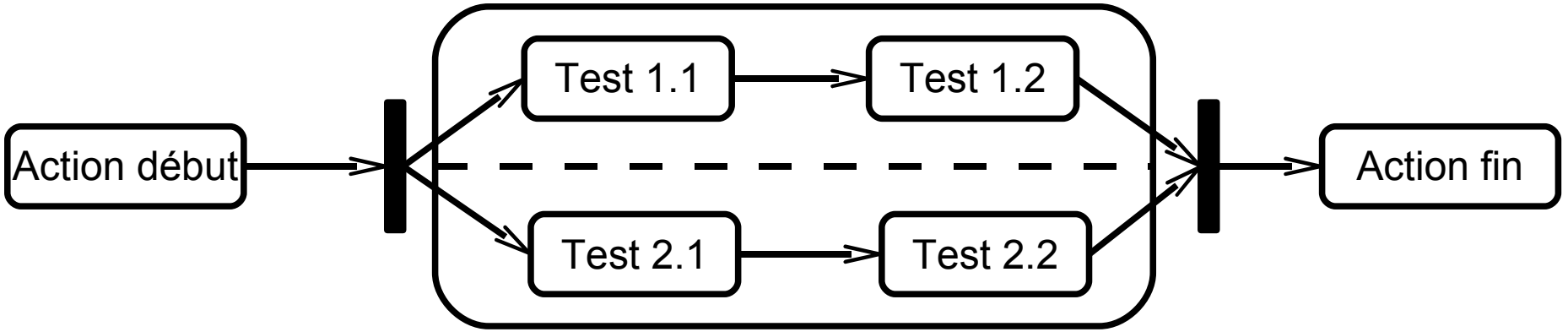
Actions dans un état

- Des champs sont définis pour décrire les actions dans un état :
  - entry/action : à l'entrée
  - exit/action : à la sortie
  - on événement/action : sur un évènement,
  - do/action : exécutée dans l'état



Etats concurrents

- Notation simple : barre de synchronisation,



- Notation complexe :
  - S'effectue sur les transitions à l'aide du mot clé *"in"*,
  - Spécifie l'état dans lequel se doit trouver l'autre automate pour activer la transition.

Ressources sur UML

- Site officiel d'UML :
  - http://www.uml.org/
- Site français sur UML (plus vieux) :
  - http://uml.free.fr/
- Site d'IBM sur UML :
  - http://www.ibm.com/software/rational/uml/
- Site officiel de l'OMG :
  - http://www.omg.org/
- Sondage :
  - http://www.volle.com/travaux/gtmodelisation5.htm

Logiciels de modélisation UML

- Logiciels libres :
  - Umbrello : http://uml.sourceforge.net/
  - ArgoUML : http://argouml.tigris.org/
  - BoUML : http://bouml.free.fr/

- Logiciels propriétaires :
  - Poseidon : http://www.gentleware.com/
  - Rational rose : http://www.rational.com/
  - Together : http://www.borland.com/