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/uml2.pdf

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)

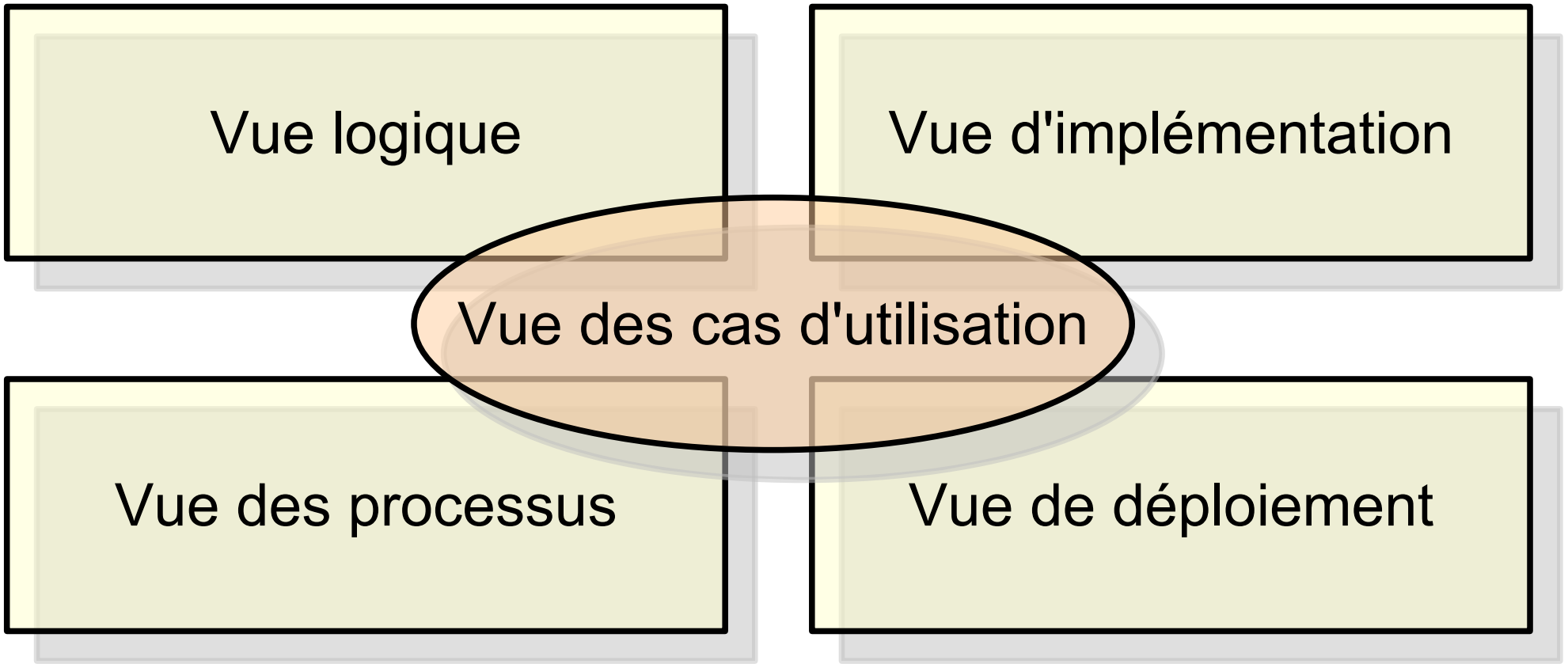
    Définit le langage pour la spécification des metamodèles,
  - (M2) métamodèle : (concept de classe)

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

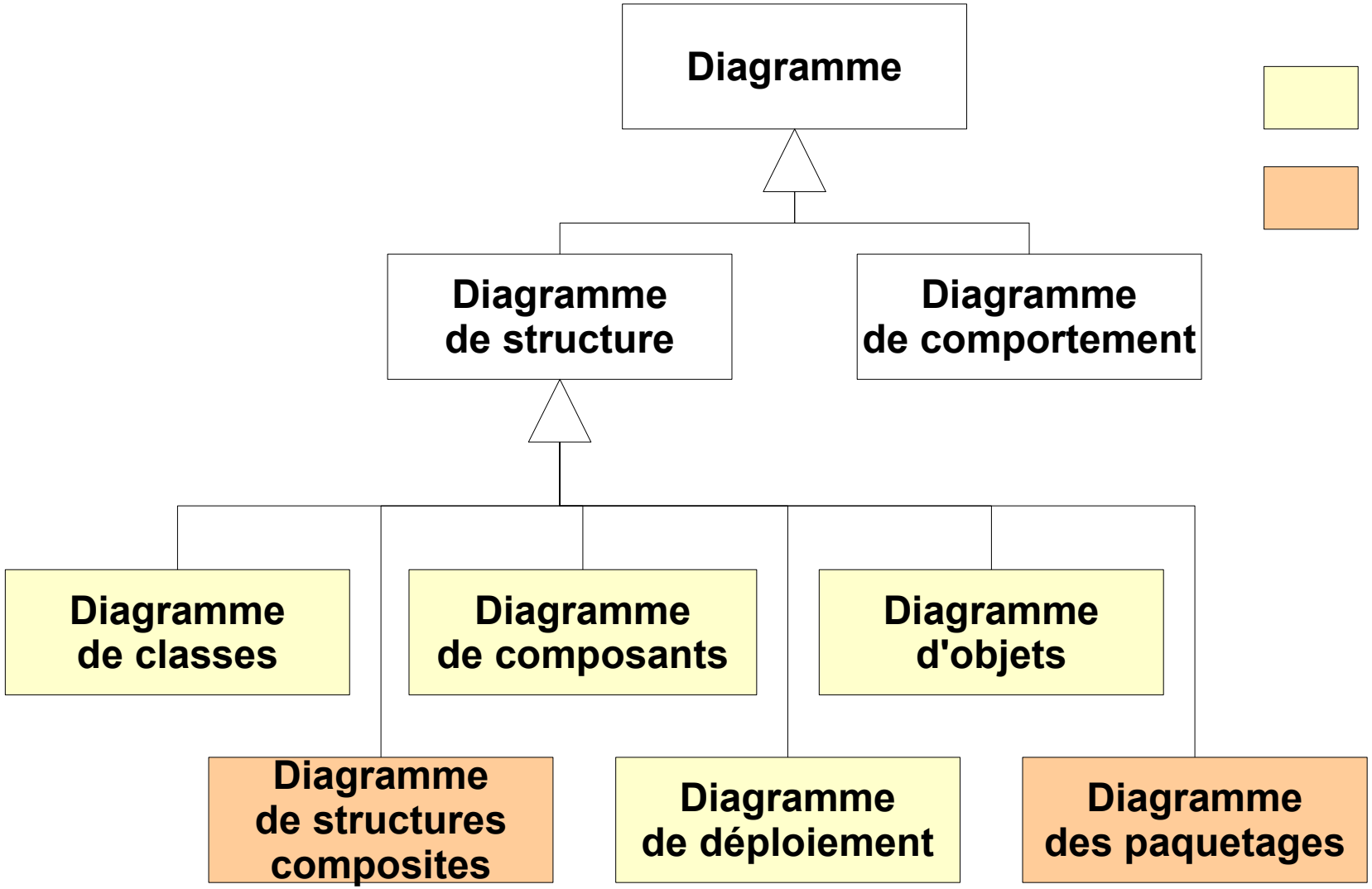
    Définit le langage pour les éléments d' un domaine,
  - (M0) objets utilisateur : (objet)

    Définit les données spécifiques du domaine.

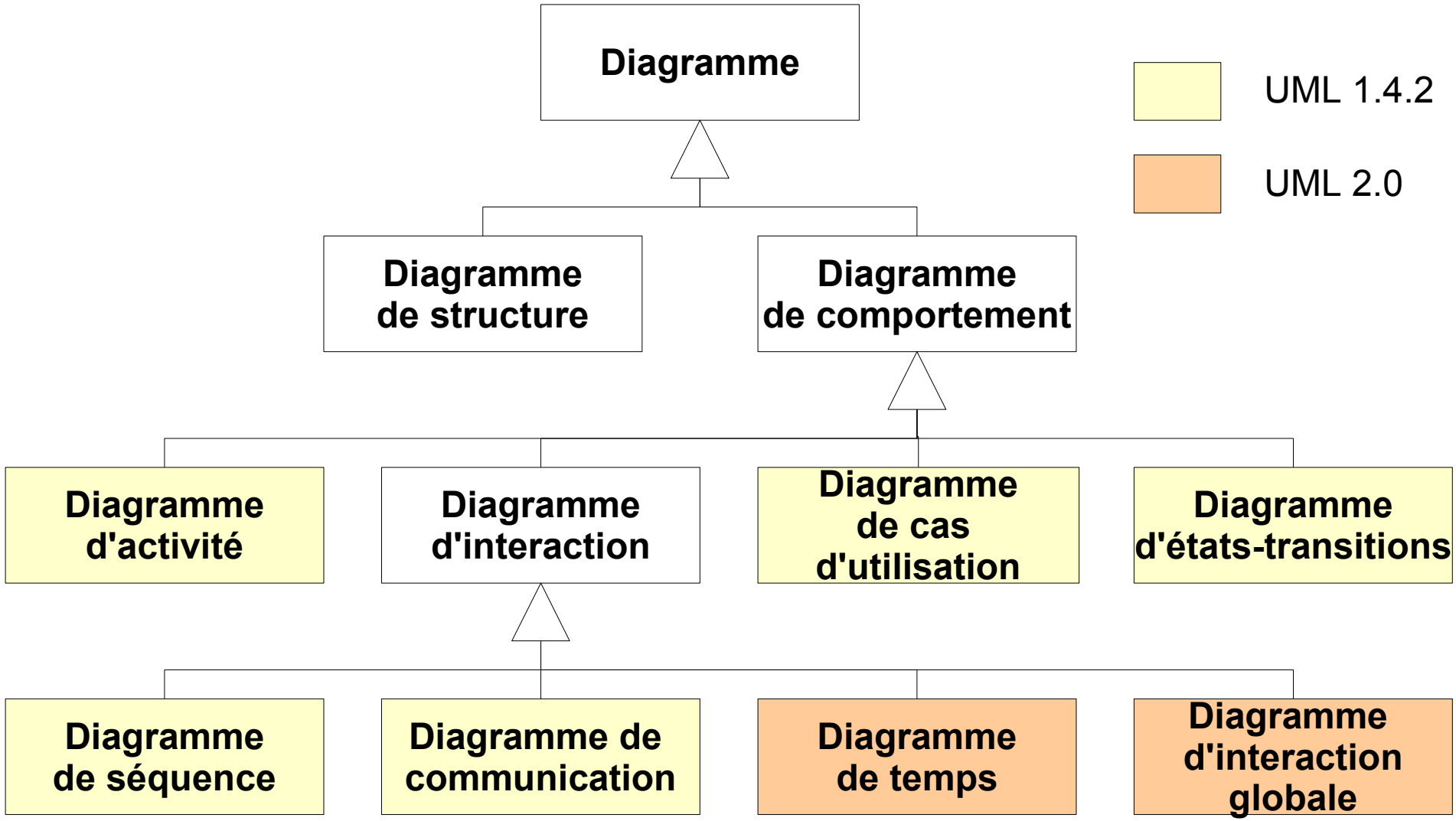
Le modèle 4+1 vue



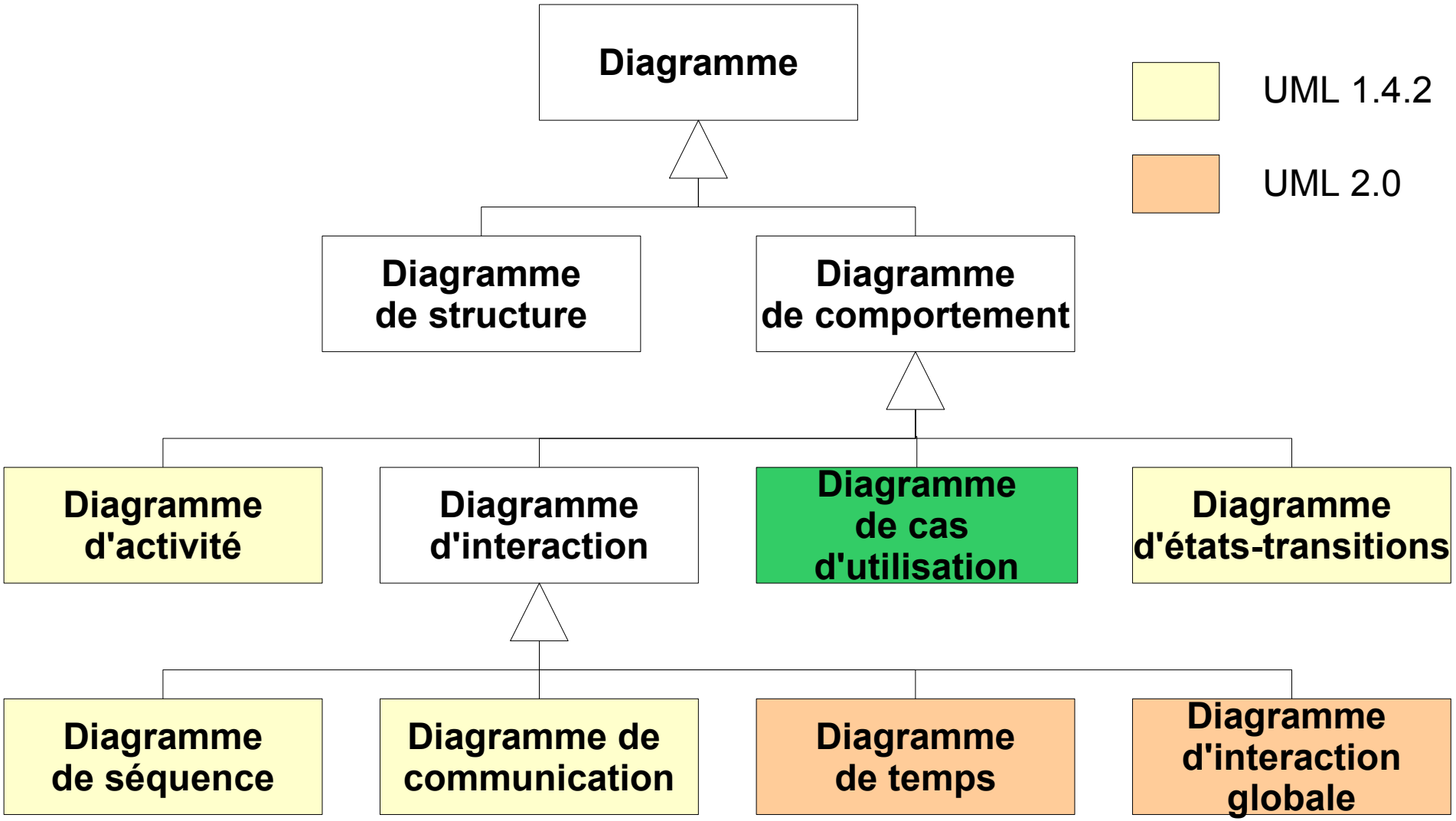
Diagrammes UML



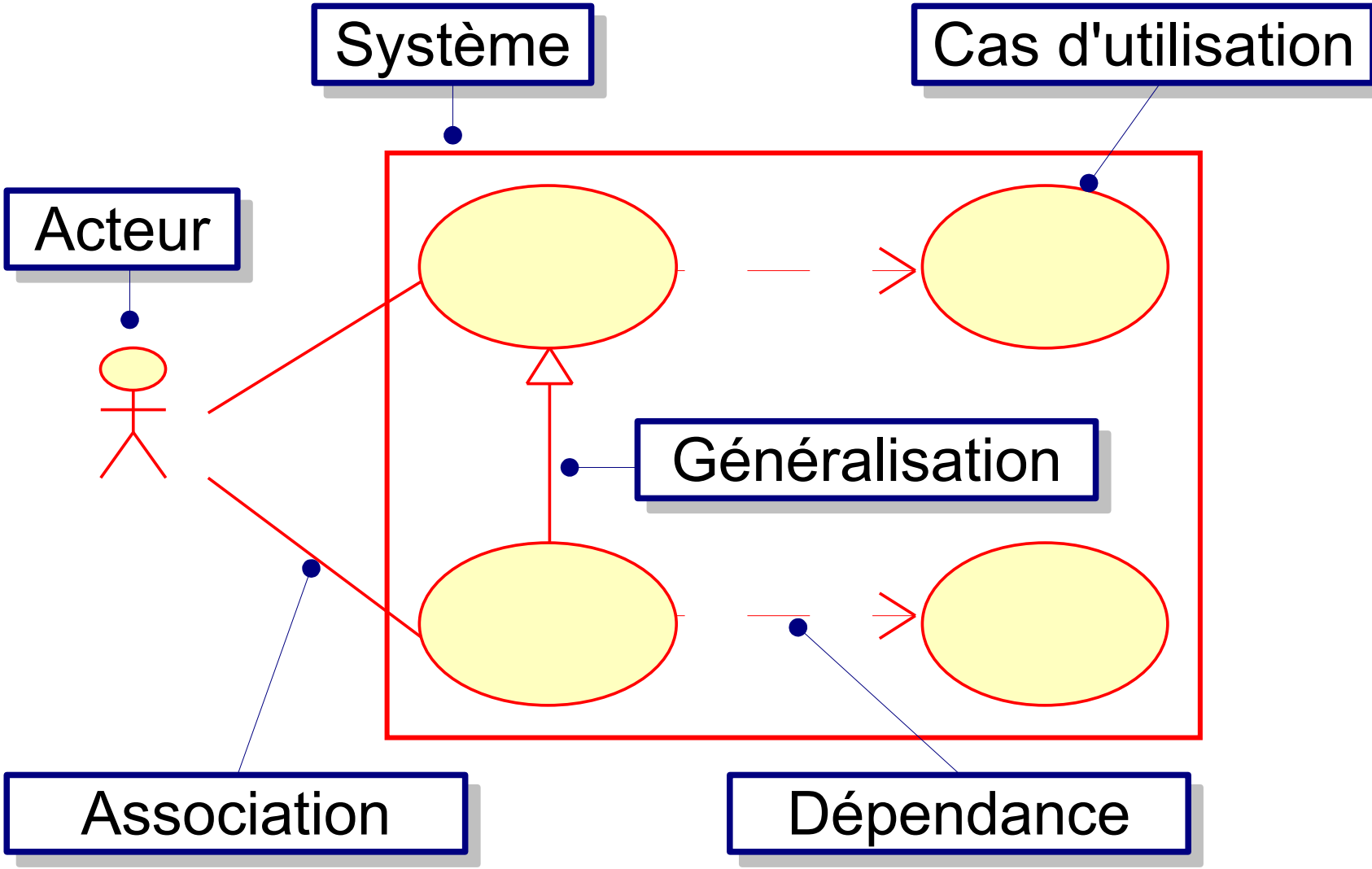
Diagrammes UML



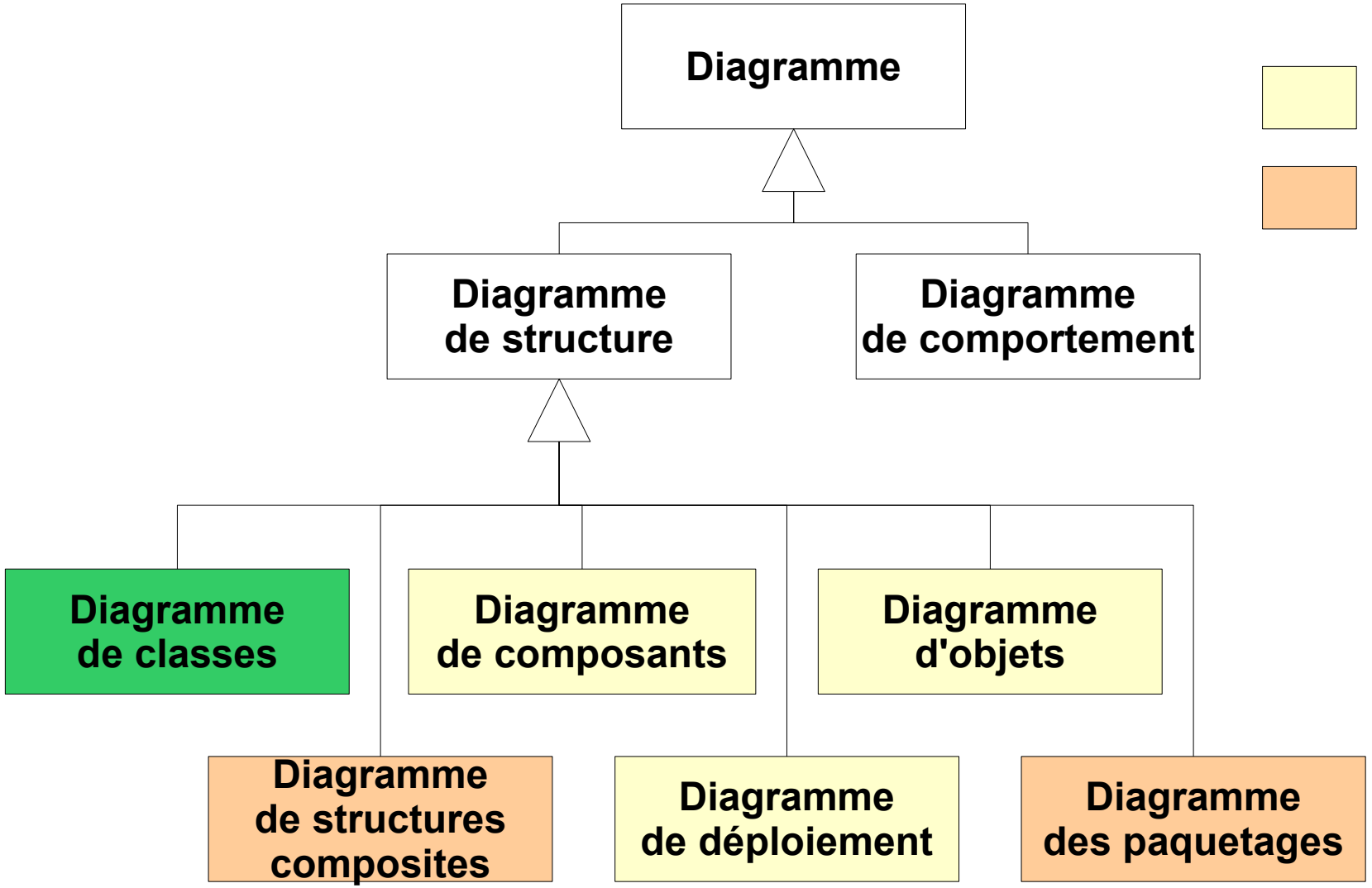
Diagrammes UML



Notations du diagramme de cas d'utilisation



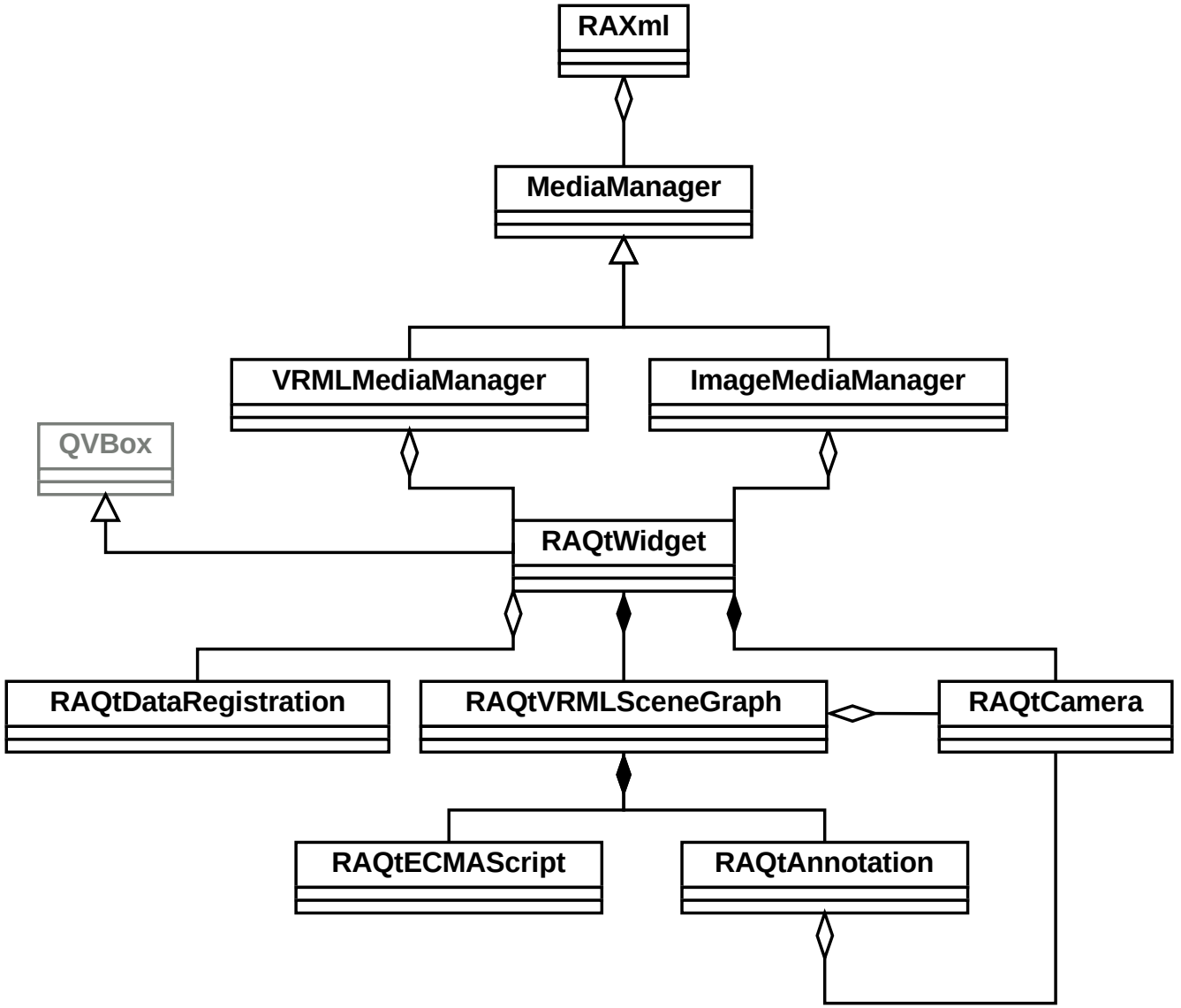
Diagrammes UML



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

Diagramme de classes



Diagrammes UML

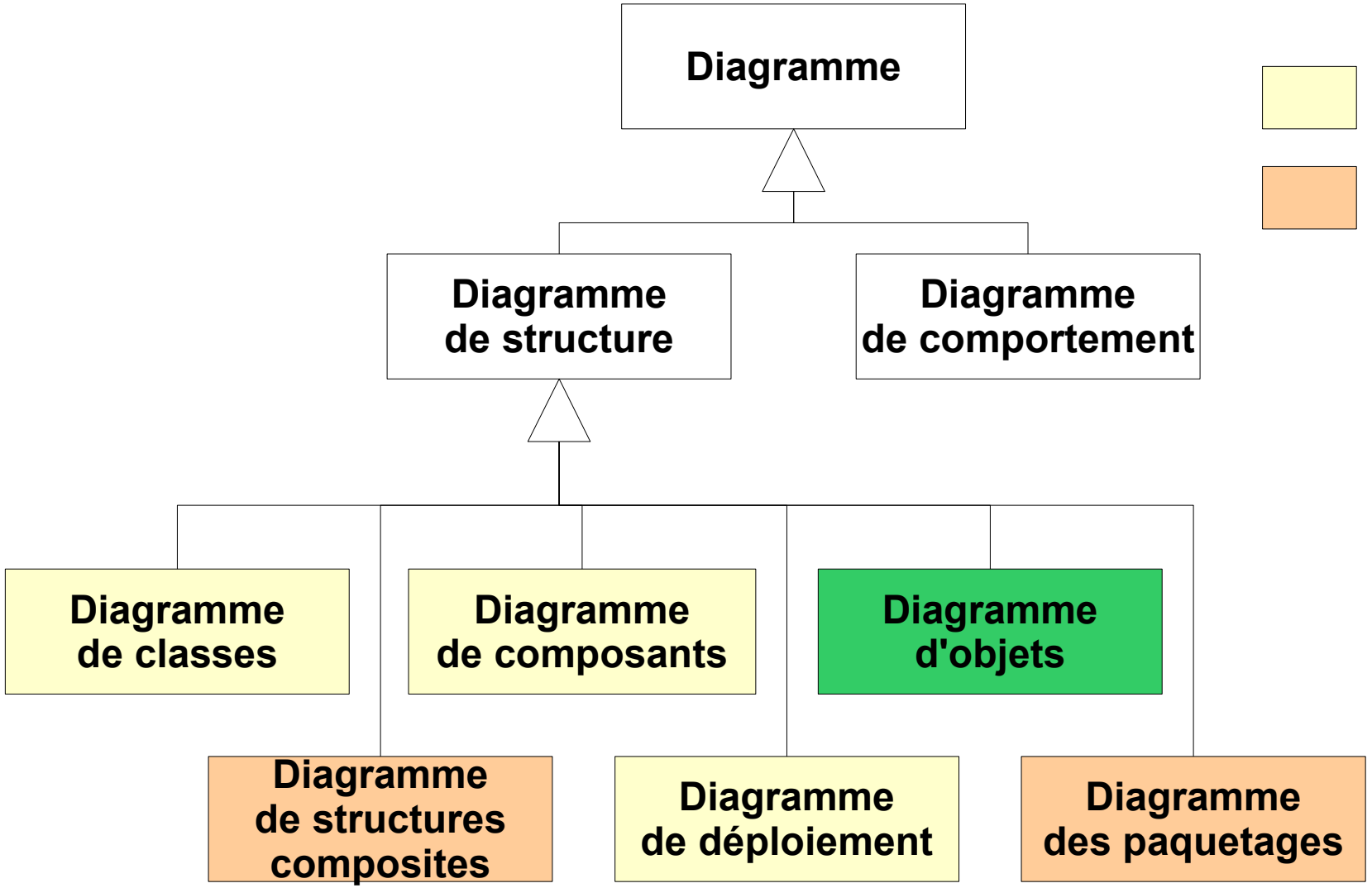
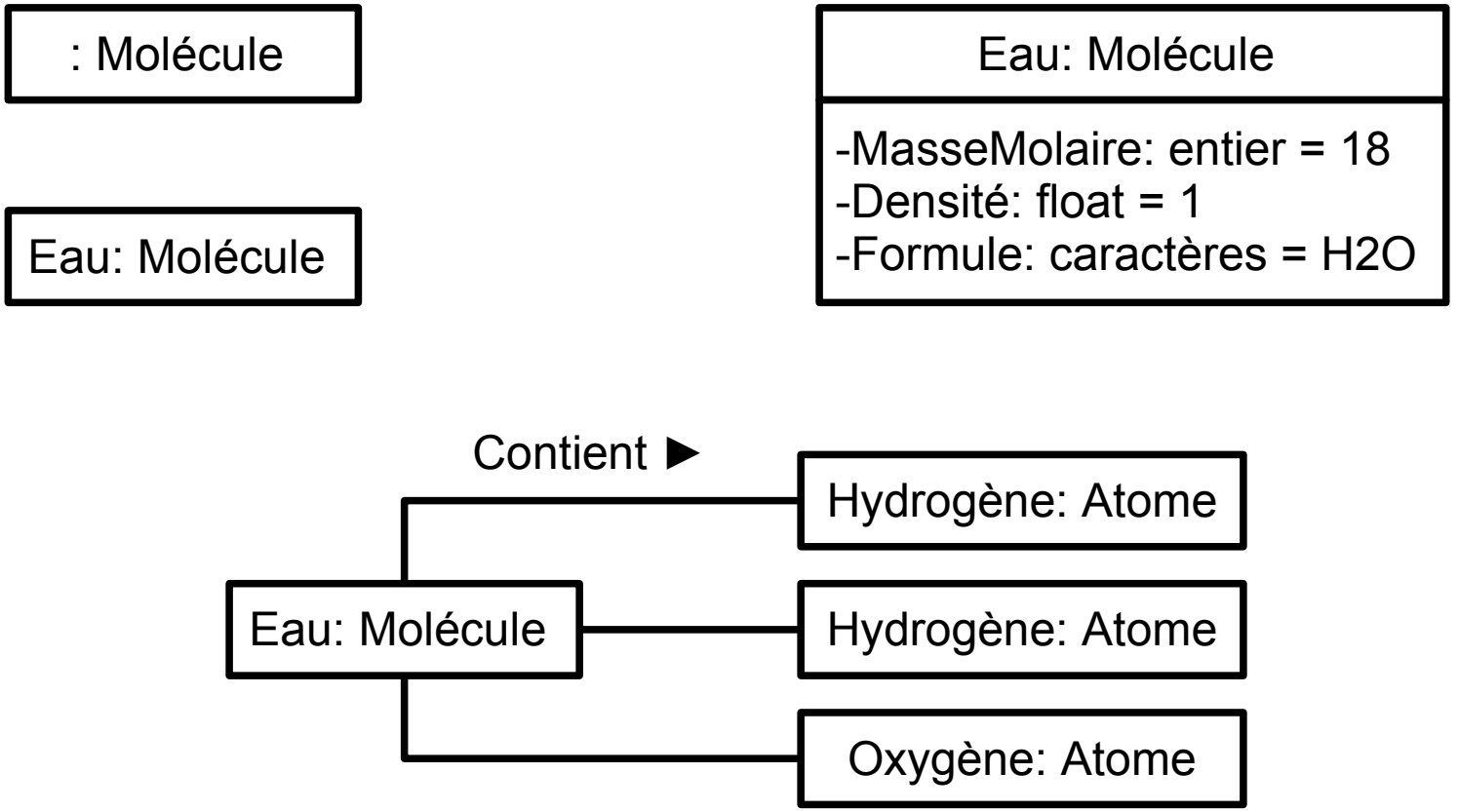


Diagramme des objets



Diagrammes UML

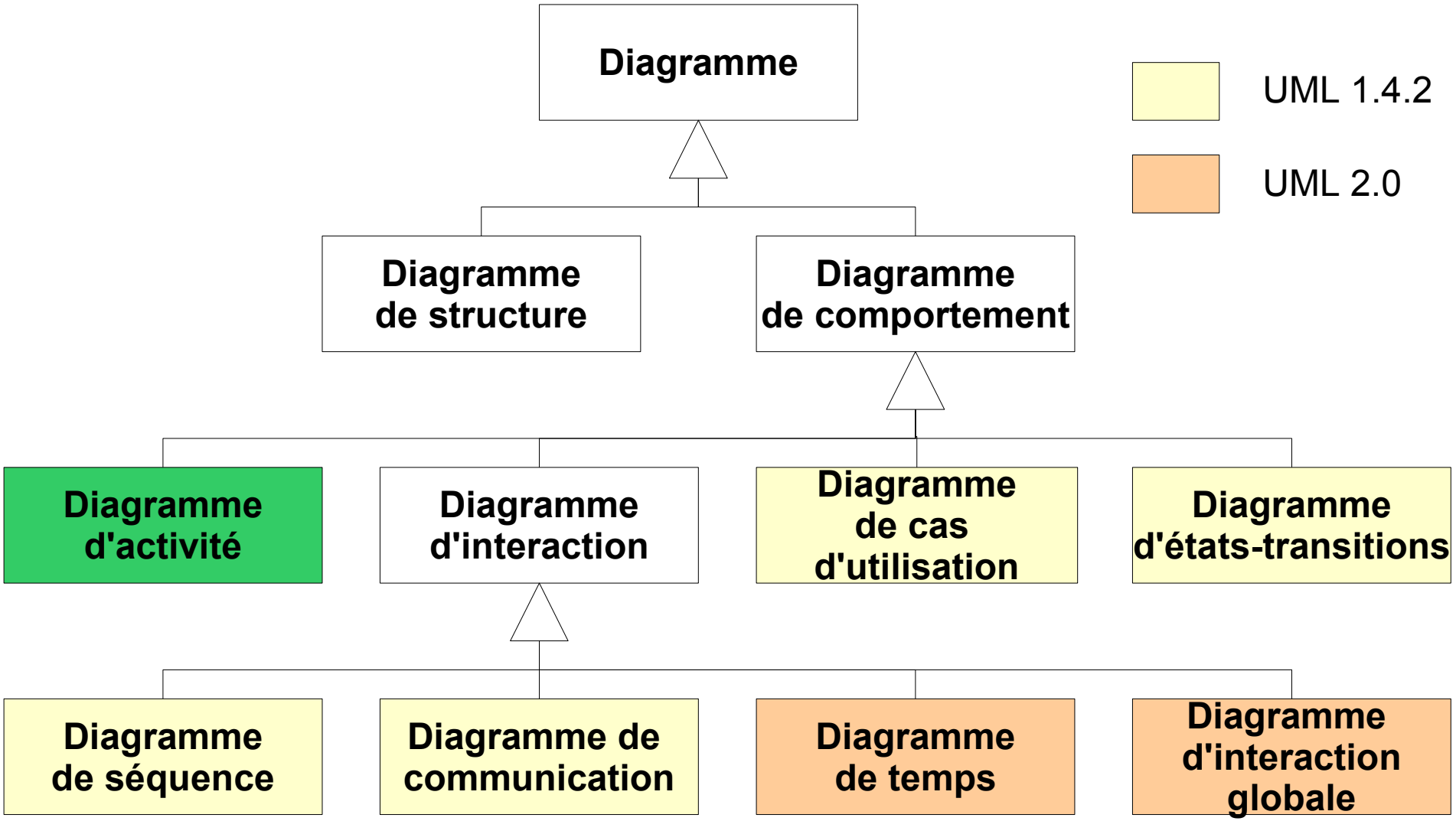
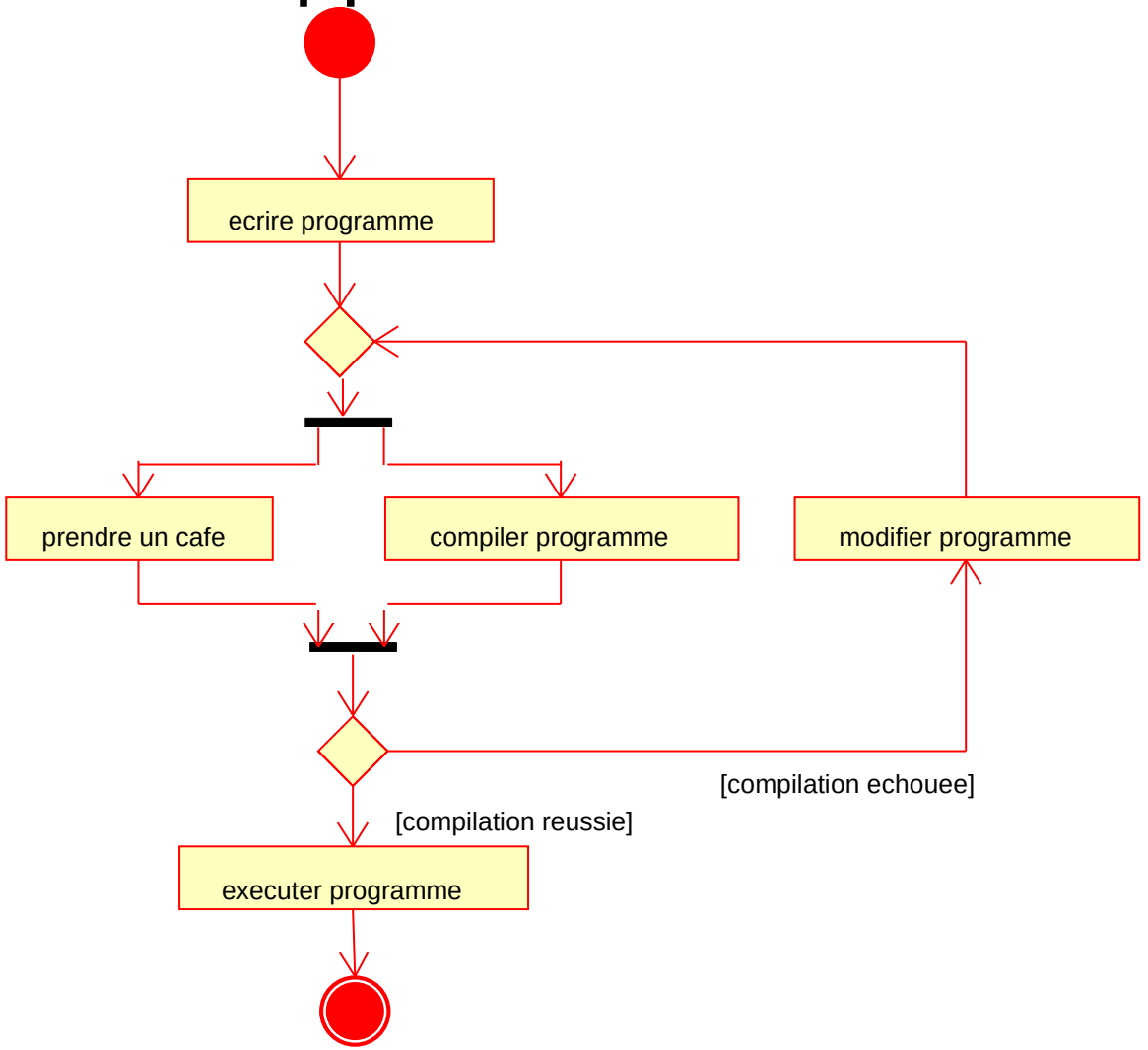


Diagramme d'activités

- Cycle de développement



Diagrammes UML

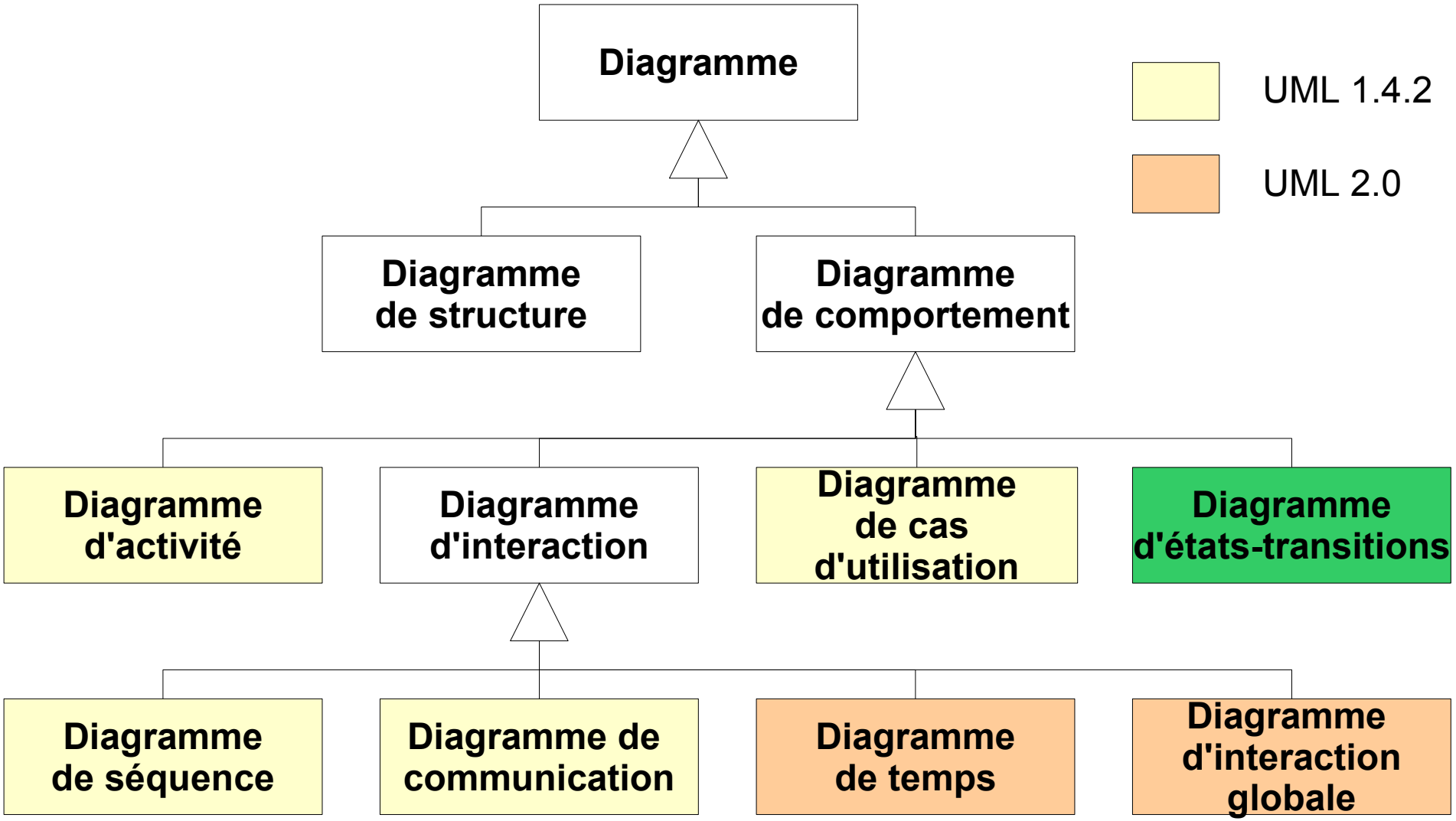
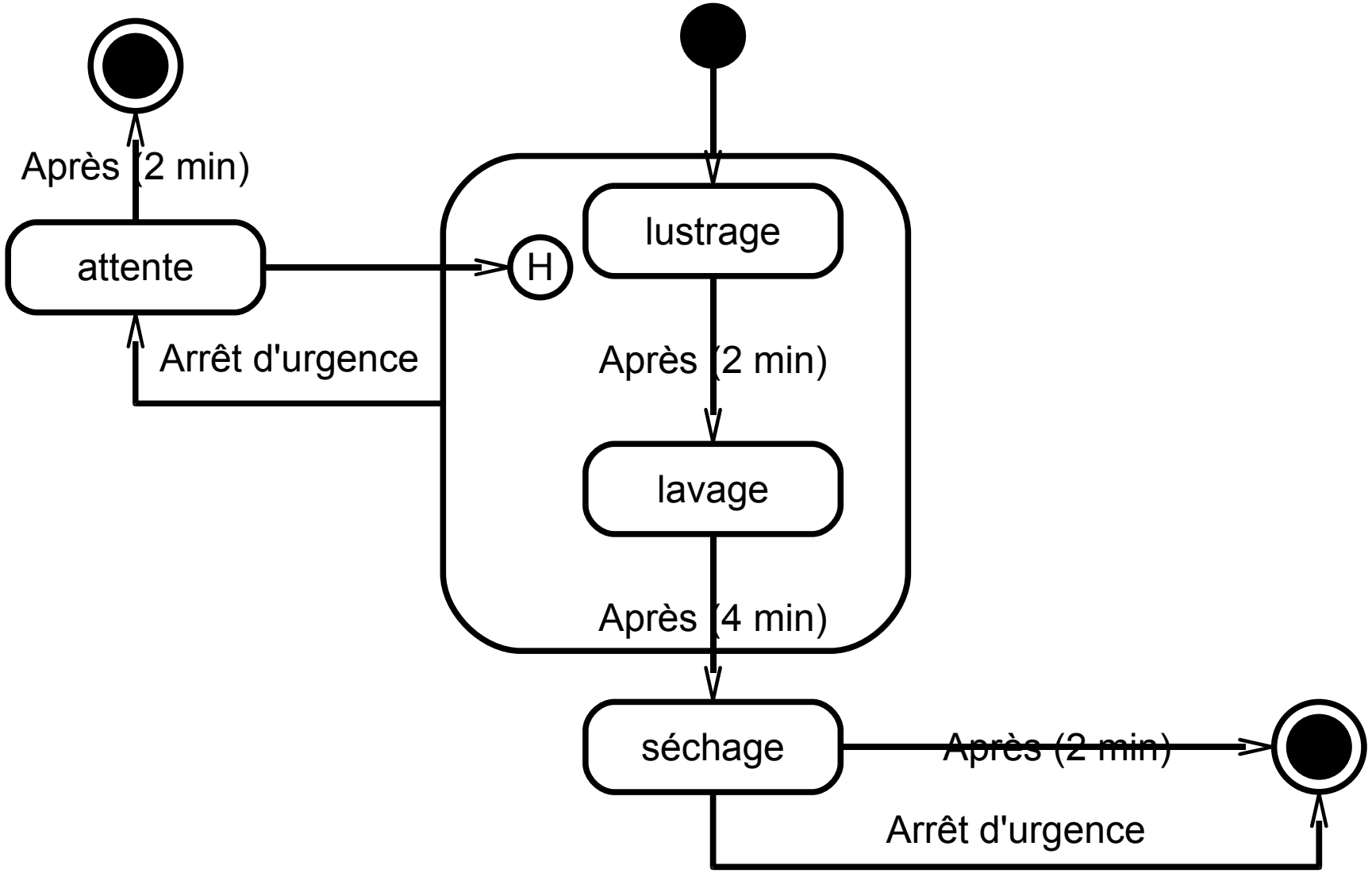


Diagramme d'états-transitions



Plan

- Introduction générale,
- Meta-modèle UML,
- Notion de vues d'un système,
- Diagrammes UML,
- A quelle étape de la création du logiciel utiliser les diagrammes UML ?
- Synthèse et conclusion.

Plan

- Introduction générale,
- Meta-modèle UML,
- Notion de vues d'un système,
- **Diagrammes UML,**
- A quelle étape de la création du SI utiliser les diagrammes UML ?
- Synthèse et conclusion.

Diagrammes UML

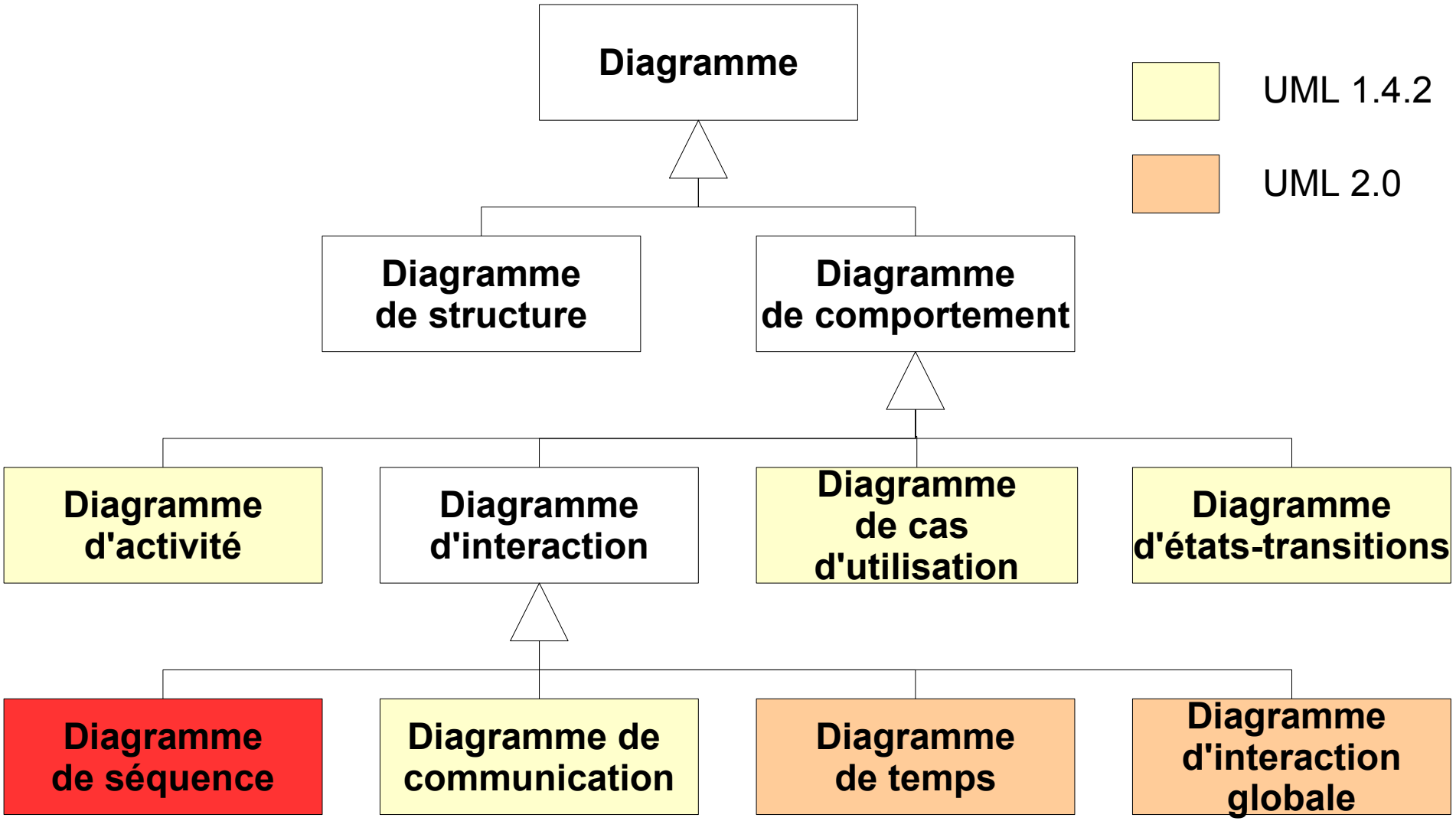
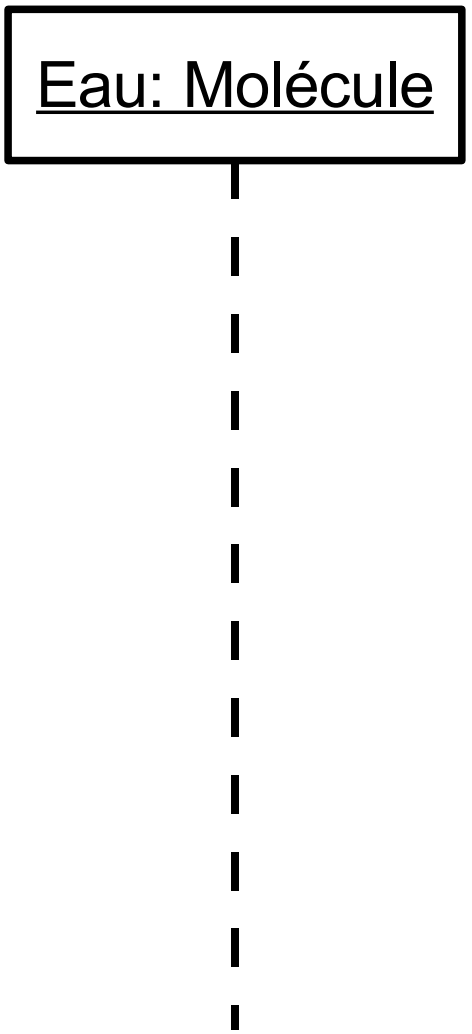


Diagramme de séquence

- Représente des collaborations entre objets selon un point de vue temporel,
- Se concentre sur l'expression des interactions,
- Peut servir à décrire un cas d'utilisation,
- Modélisation au niveau des objets,
- 3 éléments importants :
  - Les objets,
  - Ligne de vie des objets,
  - Message/stimulus,

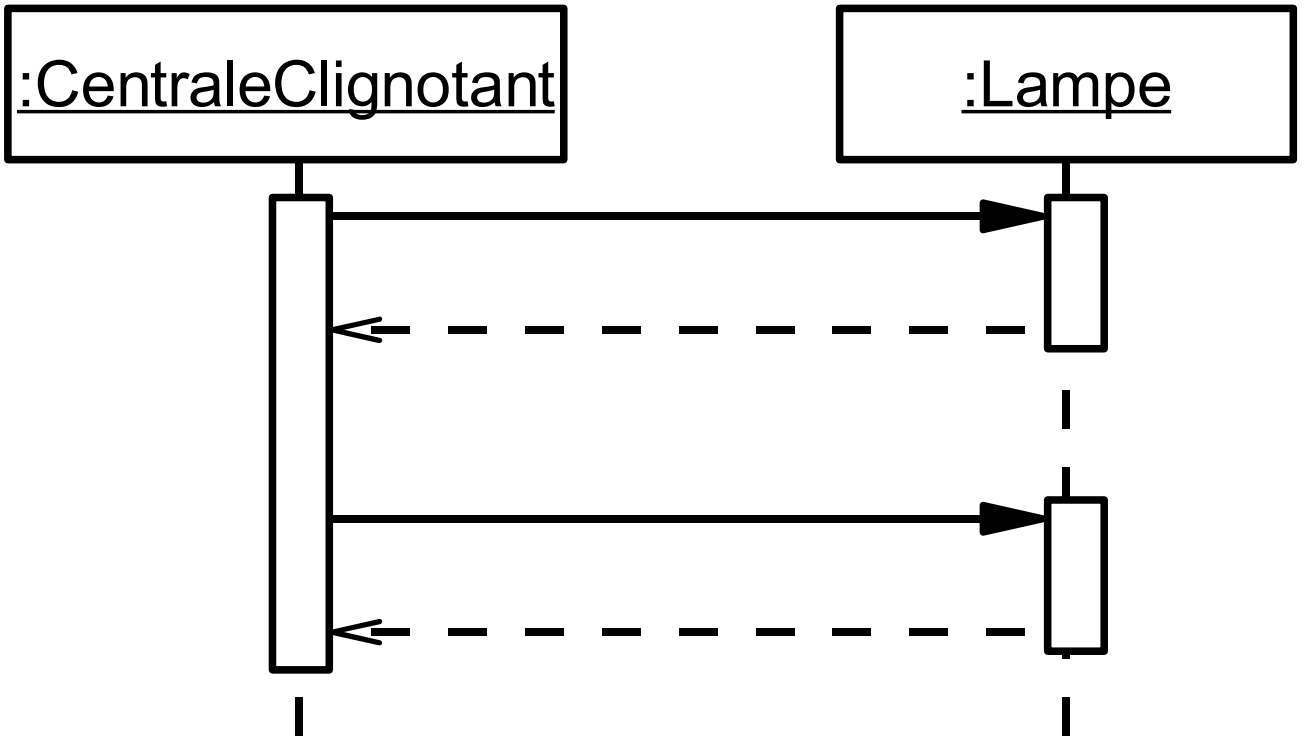
Objets et Lignes de vie

- Les objets sont représentés avec la même notation que dans le diagramme des objets (les objets peuvent aussi être des acteurs),
- Ligne de vie : ligne verticale pointillée dirigée vers le bas à partir de chaque objet,
- La ligne de vie symbolise une durée qui dépend du scénario et du comportement modélisé,



Objets et Lignes de vie

- La ligne de vie peut être remplacée par un rectangle : *la barre d'activation,*
- Elle traduit l'activation de l'objet.



Message/stimulus

- Généralement un appel, un signal ou une réponse. Représenté par une flèche,
- Le type de flèche correspond au type de message,

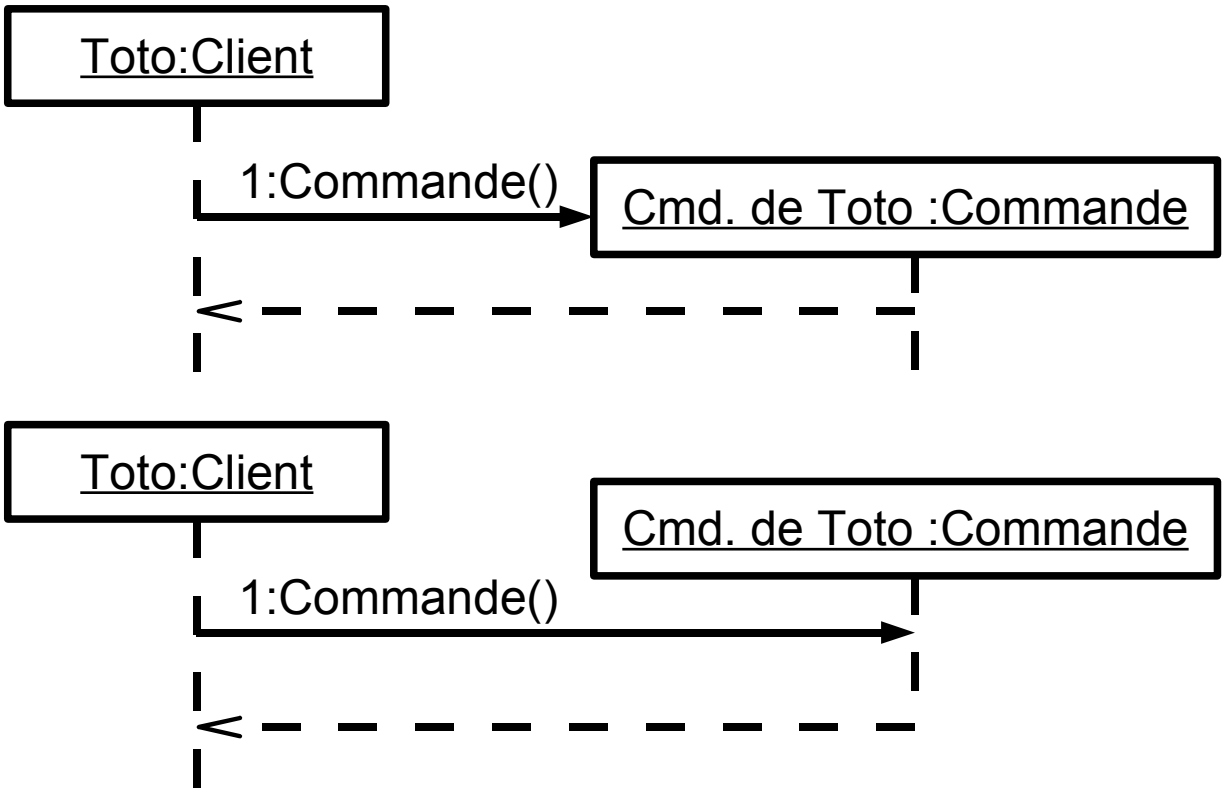
Type	Représentation
Message asynchrone	
Appel de méthode	
Réponse, création d'objets	
Message perdu	
Message trouvé	
Message minuté (timeout)	

Message/stimulus

- Un libellé accompagne les messages qui décrit une interface/opération à effectuer,
- Le message contient la signature de l'opération :
  - Son nom, ses arguments, le type retourné :
    - quotientEntier(Fraction):int
- Une opération peut être effectuée de manière répétée ou de manière conditionnelle :
  - [pour chaque élève]
  - [élève suit matière=oui]
- La flèche inclinée si délai de propagation.

Création/destruction d'objets

- 2 façons de modéliser la création d'objets :



- Le destruction est marqué par un '**X**' en fin de ligne de vie de l'objet détruit.

Diagrammes UML

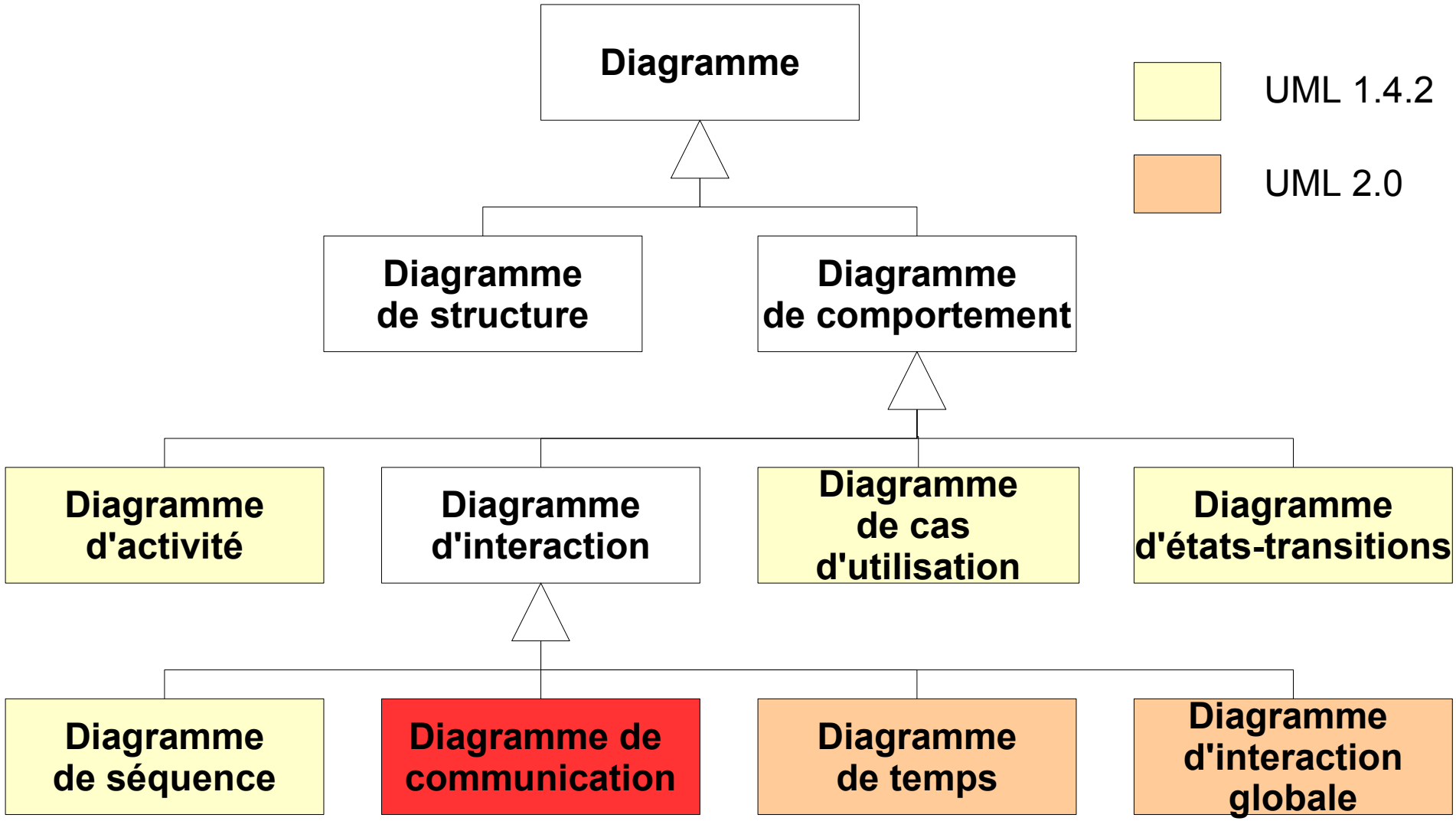


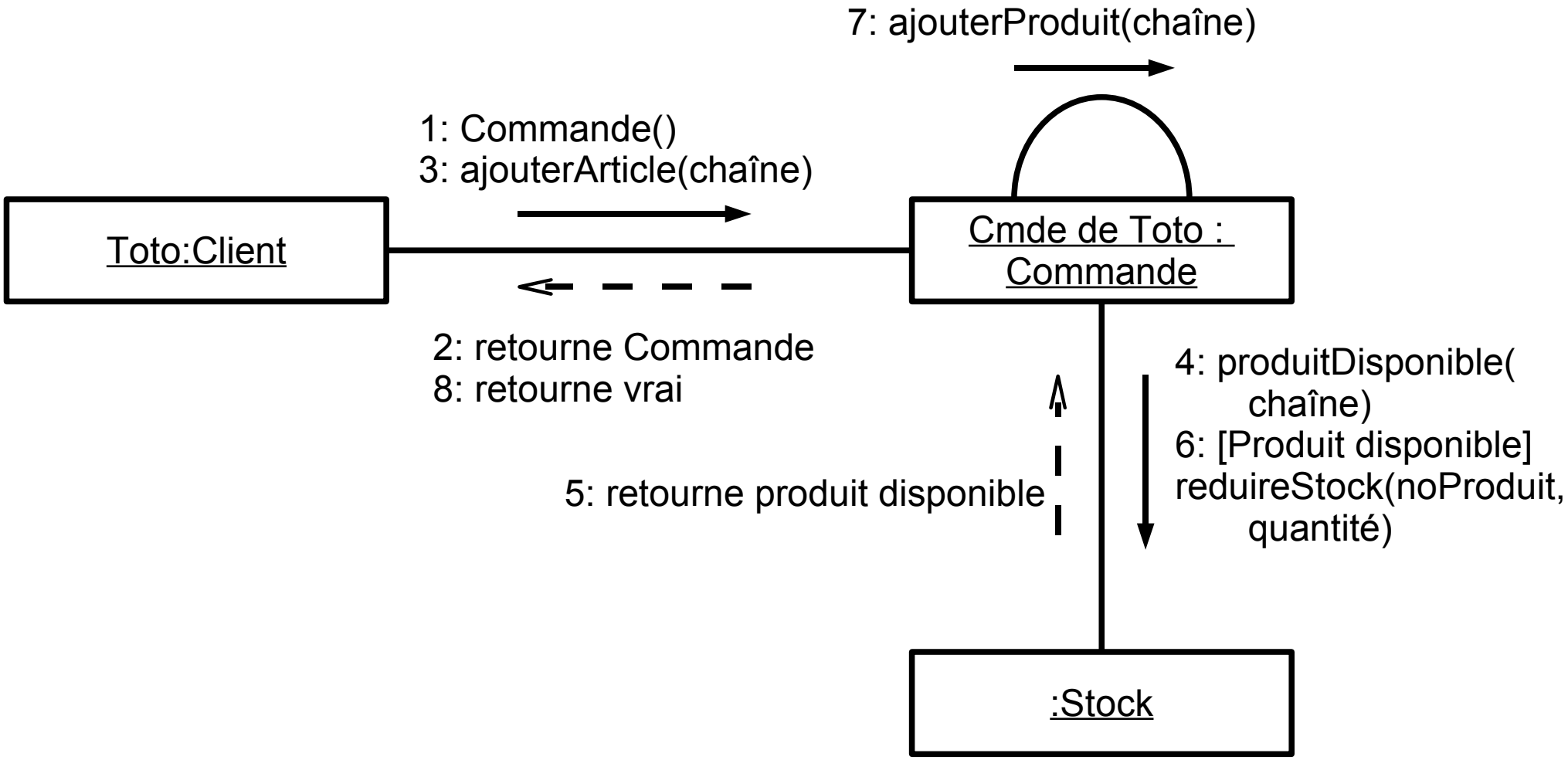
Diagramme de communication

- Appelé aussi diagramme de collaboration (UML1.4),
- Montre des interactions entre objets (instances de classe et acteurs),
- Représente le contexte d'une interaction,
- Précise les états des objets qui interagissent,
- Fournit des outils pour déterminer les interfaces des objets :
  - La description du message donne la signature de l'opération pour l'objet récepteur.

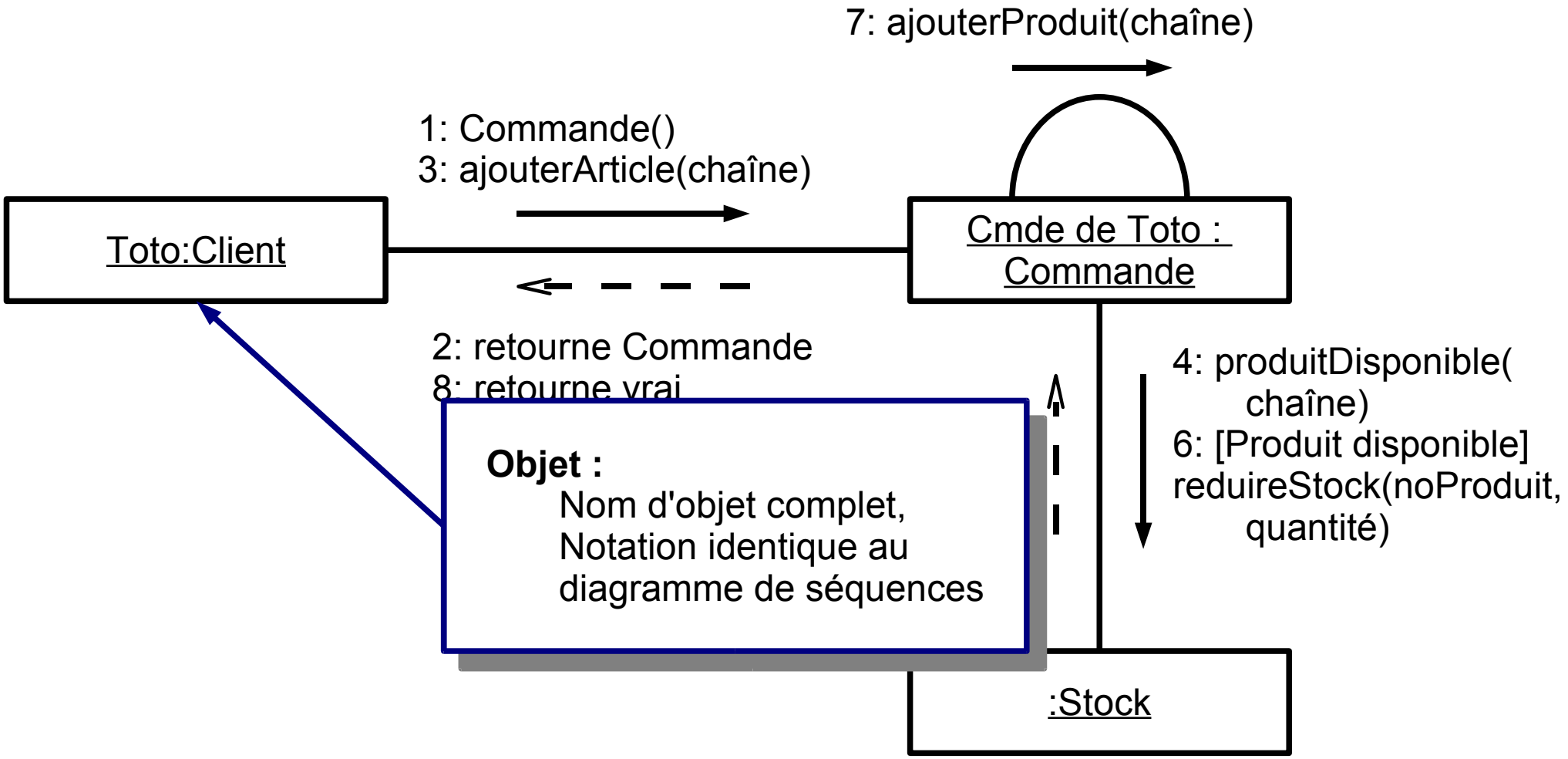
Diagramme de communication et de séquences

- Deux vues du même phénomène :
  - Il existe des outils de passage de l'un à l'autre,
- Diagramme de communications :
  - Donne la priorité à la figuration des interactions,
  - Aide à valider le diagramme de classes en mettant en évidence la nécessité de chaque association comme moyen de transmission des messages,
- Diagramme de séquences :
  - Modélise la création/destruction d'objets,
  - Montre l'activation des objets,

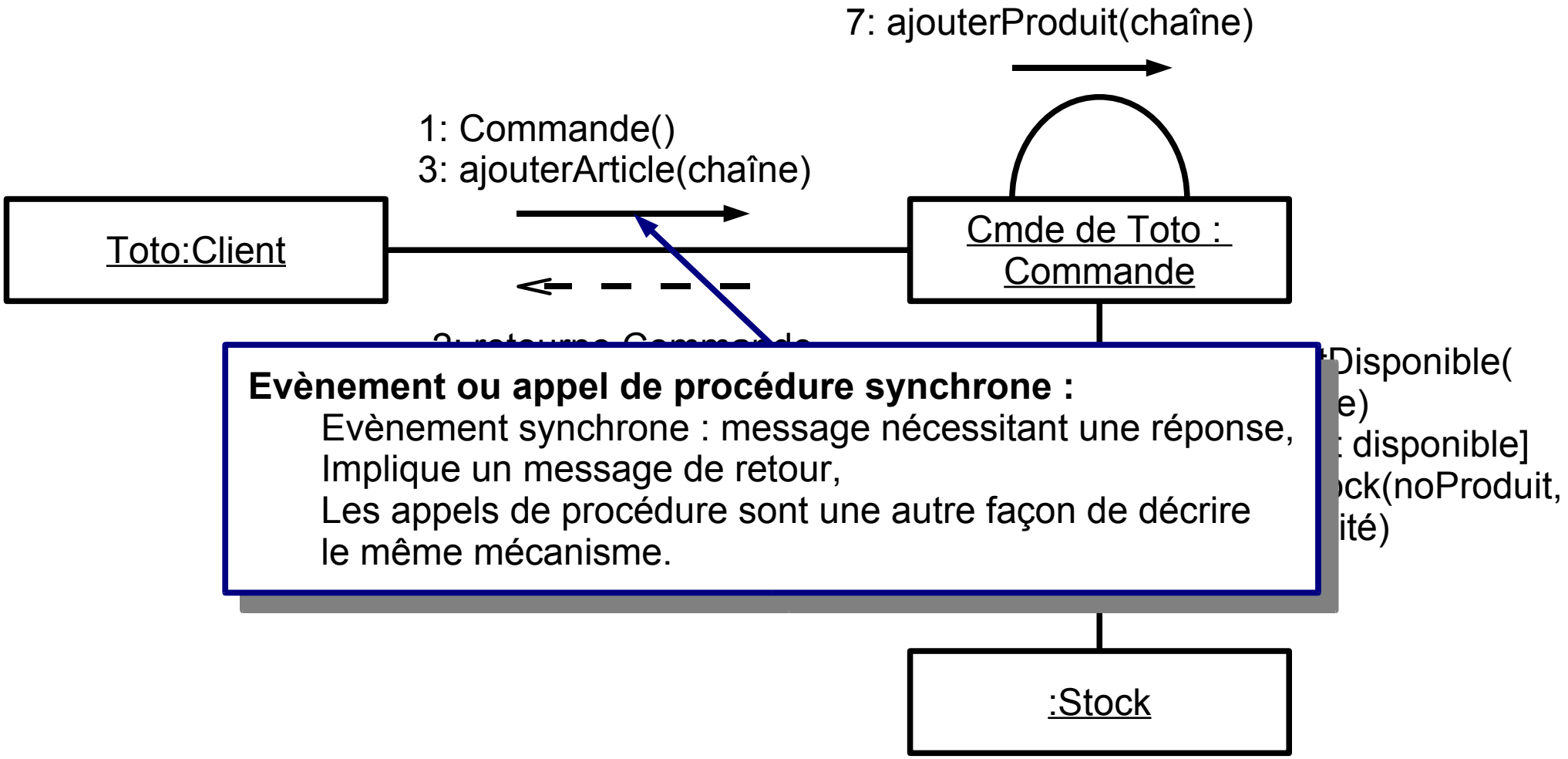
Exemple de diagramme



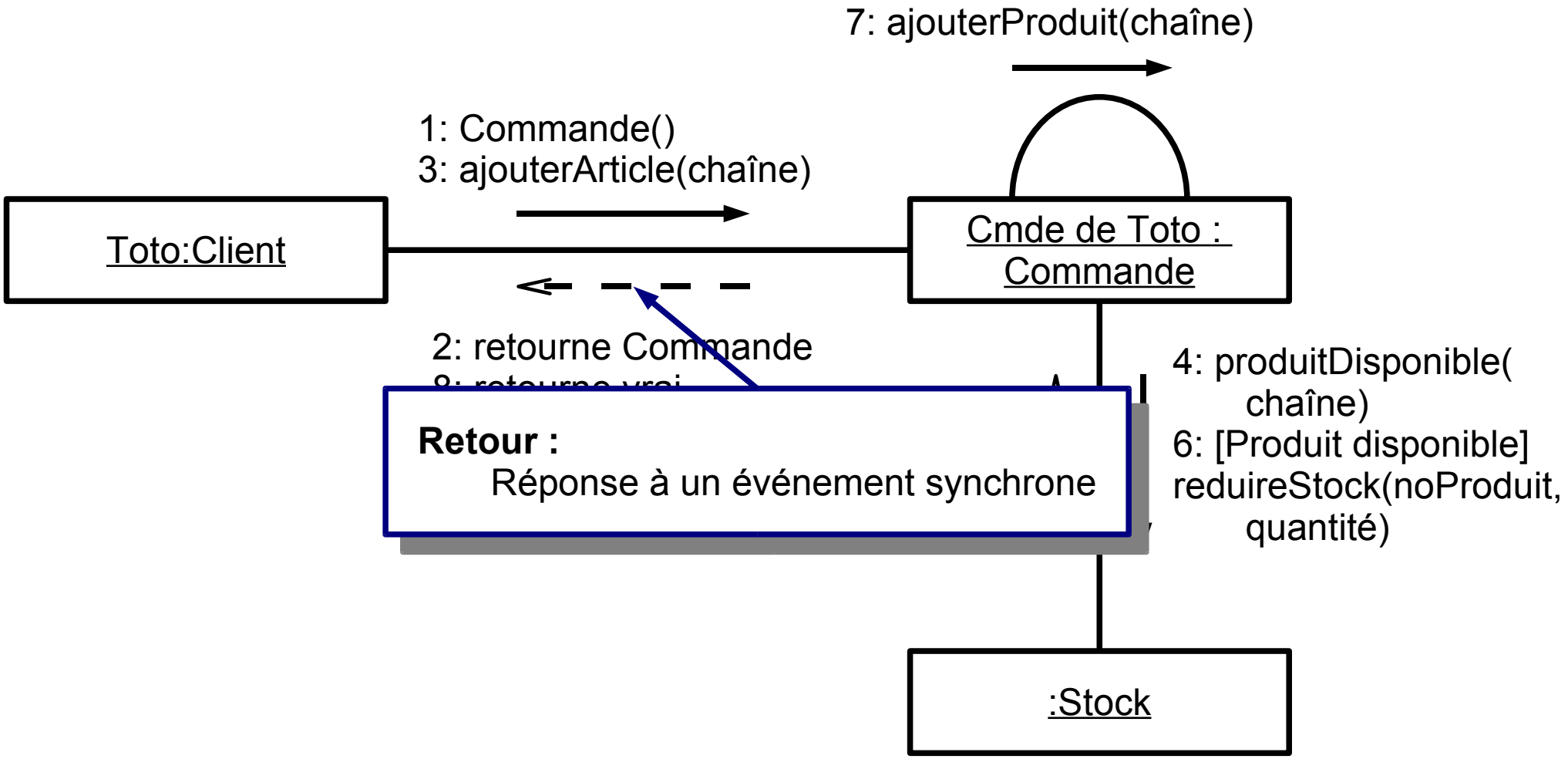
Exemple de diagramme



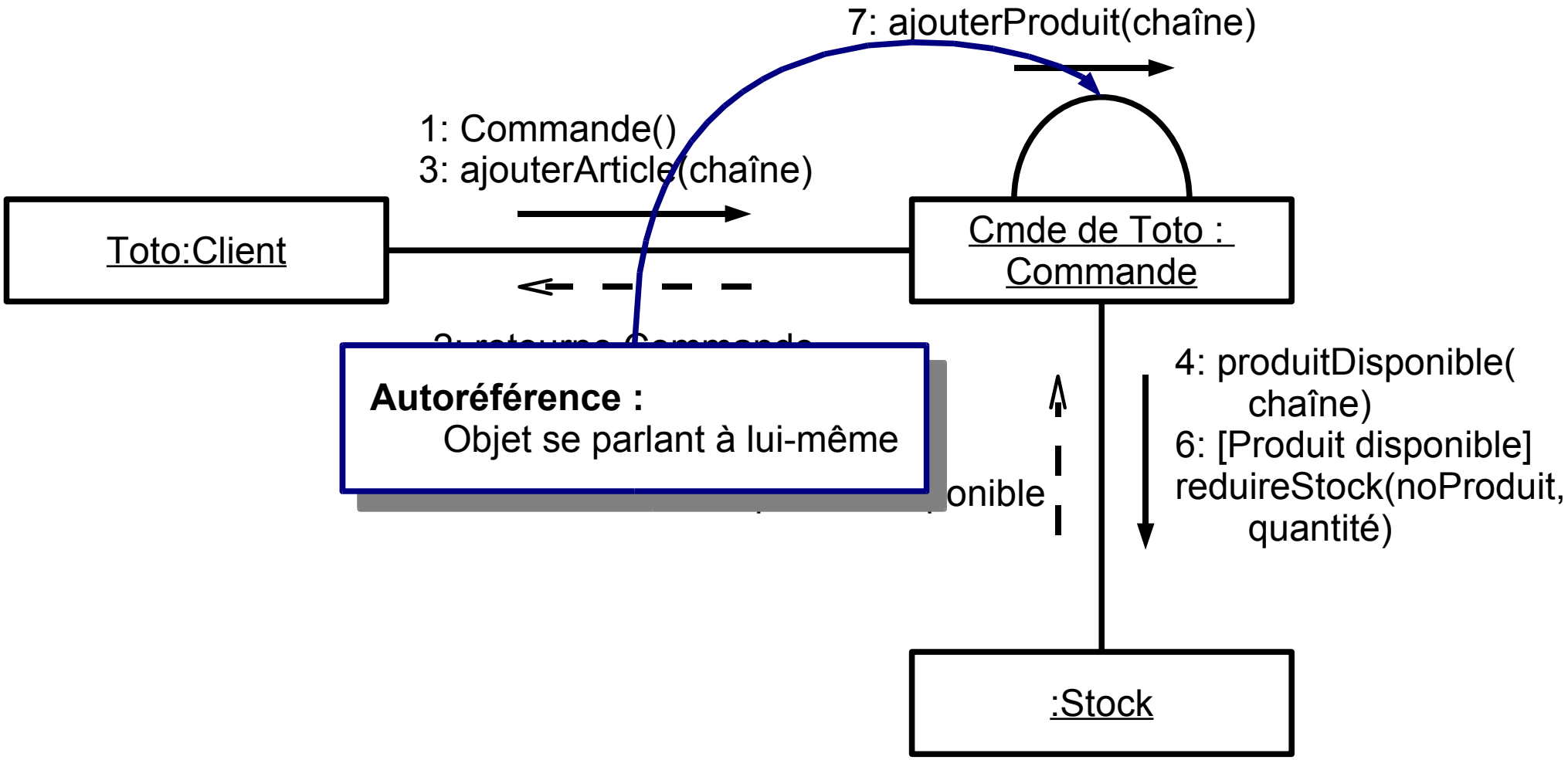
Exemple de diagramme



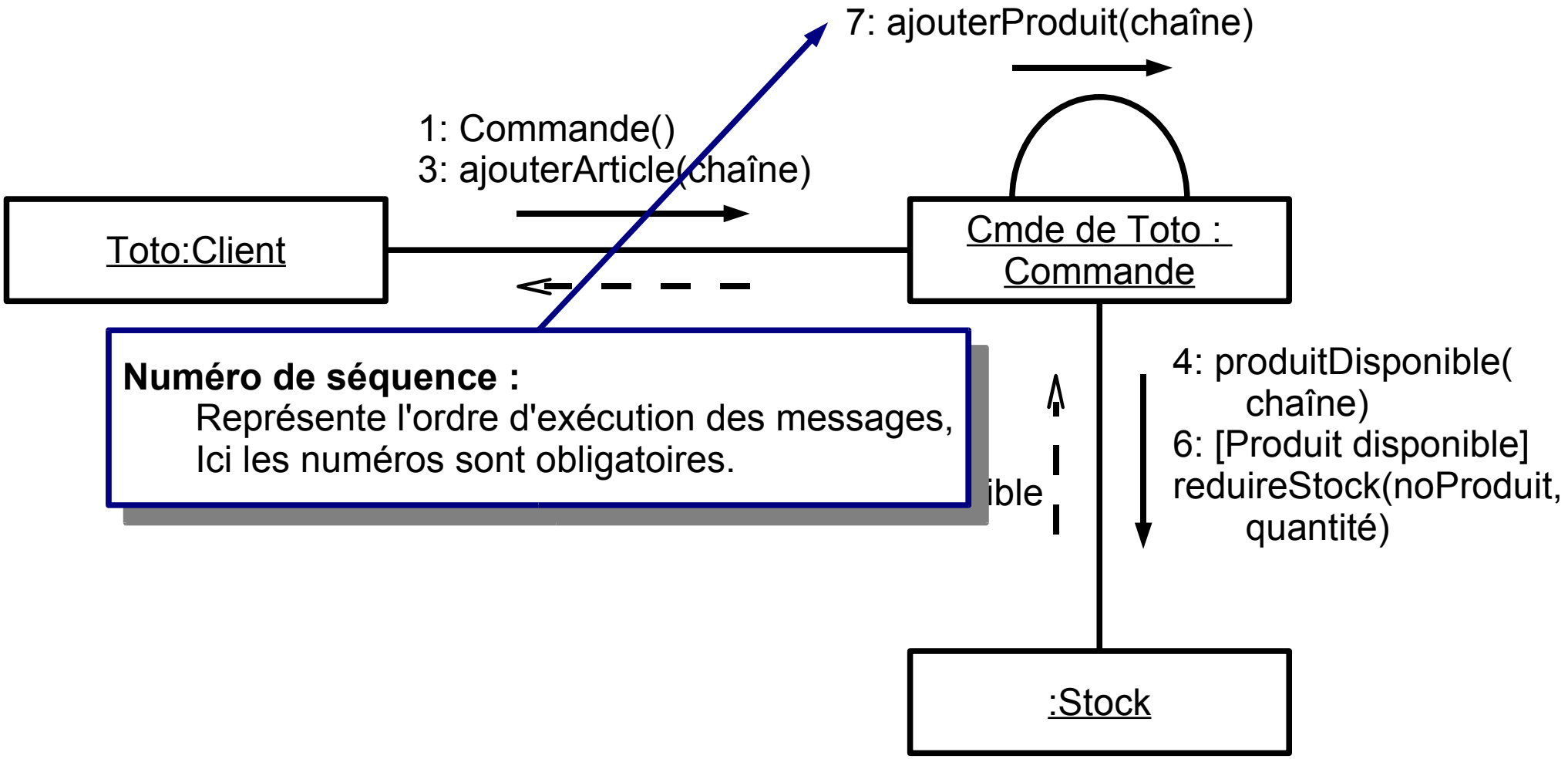
Exemple de diagramme



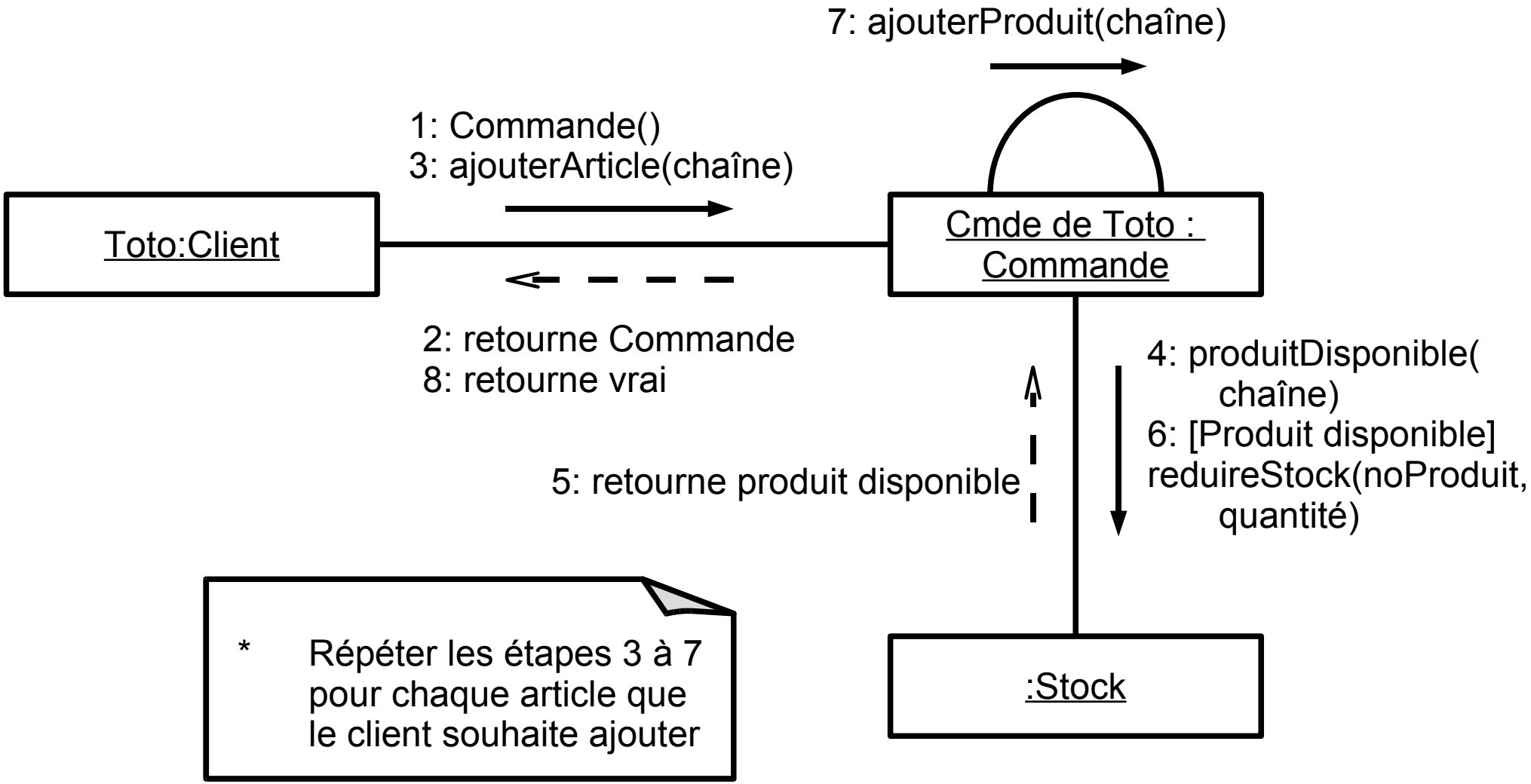
Exemple de diagramme



Exemple de diagramme



Exemple de diagramme



Diagrammes UML

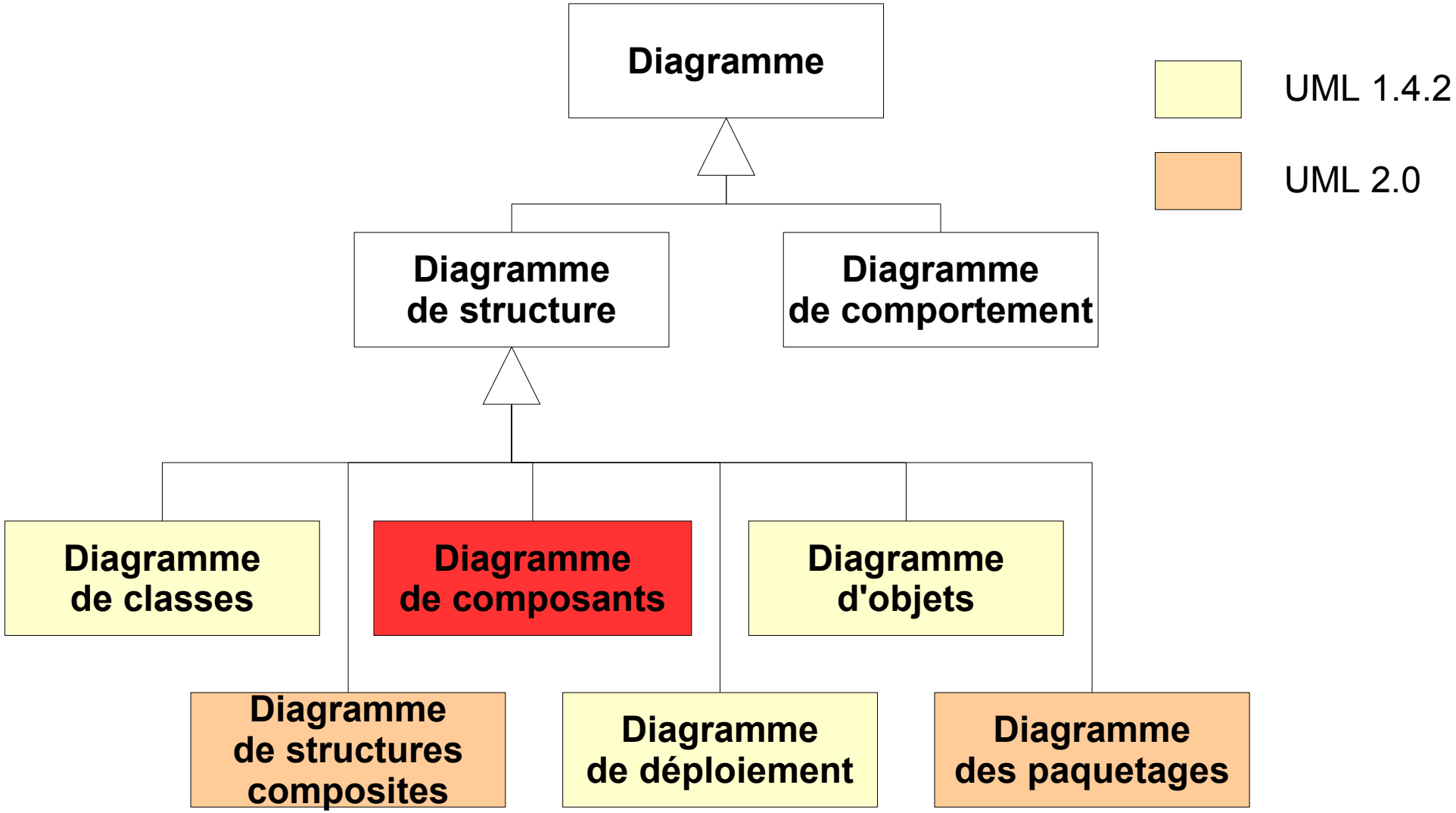


Diagramme de composants

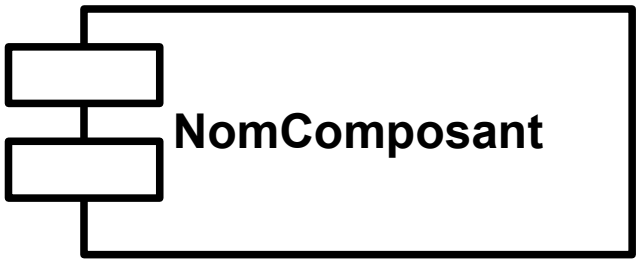
- Décrit l'architecture physique et statique d'une application en termes de modules : fichiers sources, librairies, exécutables, ...
- Montre la mise en oeuvre physique des modèles de la vue logique avec l'environnement de développement,
- Identifie les contraintes de compilation grâce à la mise en évidence des dépendances,
- Les composants peuvent être organisés en paquetage,

Diagramme de composants

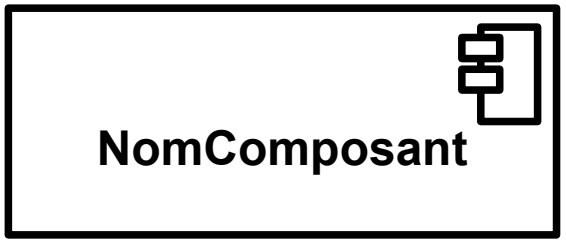
- 3 catégories principales de composants :
  - Les composants de déploiement :
    - Nécessaires pour faire fonctionner le système,
  - Les composants de travail :
    - Englobent les modèles, le code source et les fichiers de données utilisés pour créer les composants de déploiement,
  - Les composants d'exécution :
    - Composants créés pendant le fonctionnement de l'application.
- Il existe des dépendances entre composants

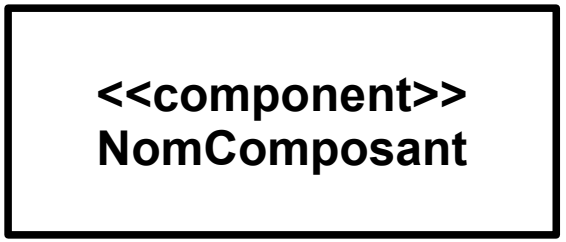
Composant : notation

- Rectangle avec 2 rectangles plus petits du côté gauche et le nom au centre (+stéréotype) :



- **Notation changée avec UML 2.0 !**





Stéréotypes de composants

- Décrivent le rôle joué par le composant :
  - <<executable>> : composant exécutable,
  - <<library>> : bibliothèque de fonction,
  - <<table>> : composant de base de données,
  - <<file>> : fichier de données,
    - <<source>> : code destiné à être compilé,
    - <<document>> : document, par exemple texte ou image,
    - <<script>> : script qui peut être interprété par le SI,
- Ce sont ce que l'on appelle des artefacts d'implémentation.

Interfaces de composants

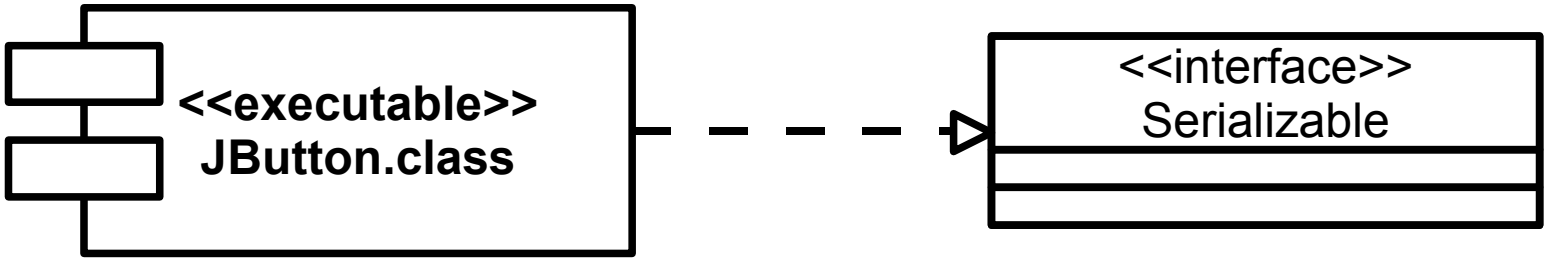
- Une interface représente la déclaration d'un ensemble cohérent de fonctionnalités et d'obligations.
- Une interface spécifie un contrat : chaque instance qui réalise cette interface doit respecter ce contrat.
- Les interfaces sont des déclarations, elles ne sont pas instanciables.
- Une spécification d'interface est implémentée par l'instance d'une classe concrète.

Interfaces de composant

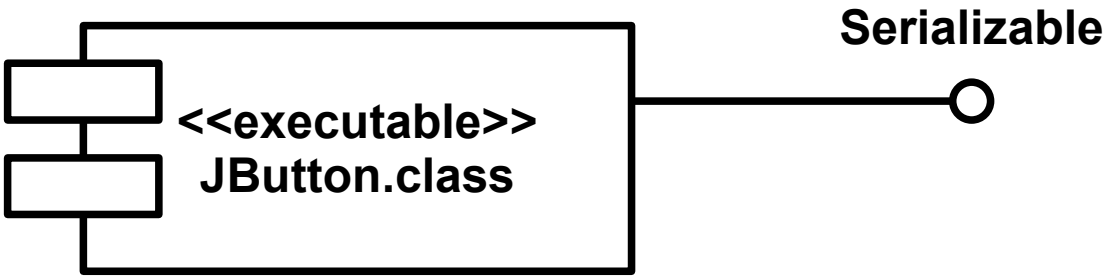
- Un composant *réalise* une interface, c'est à dire l'applique,
- L'interface implémentée par un composant est en fait implémentée par les classes de celui-ci,
  - Cette interface doit se retrouver dans le diagramme des classes,
- Un composant peut implémenter plusieurs interfaces,
  - Leur nombre est dicté par les classes implémentant le composant.

Interfaces de composants

- 2 manières de modéliser la réalisation :
  - A l'aide d'une classe stéréotypée :

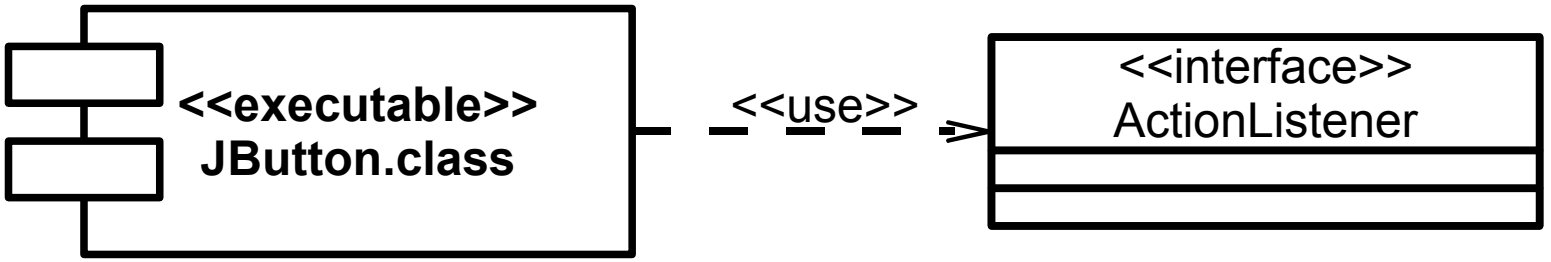


- A l'aide d'une "sucette" :

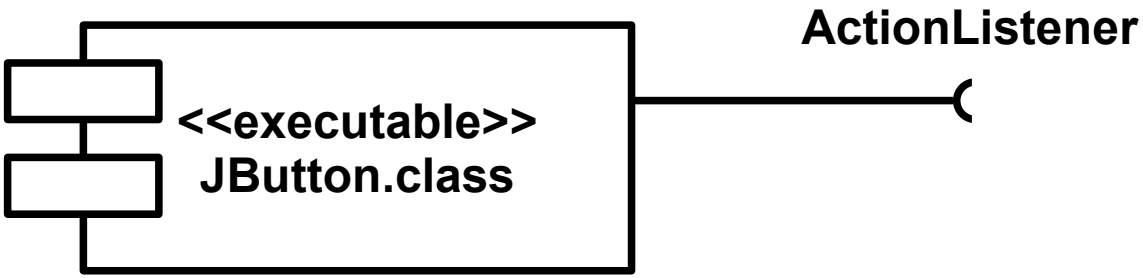


Interfaces de composants

- 2 manière de modéliser la connaisance et l'utilisation d'une interface :
  - A l'aide d'une classe stéréotypée :



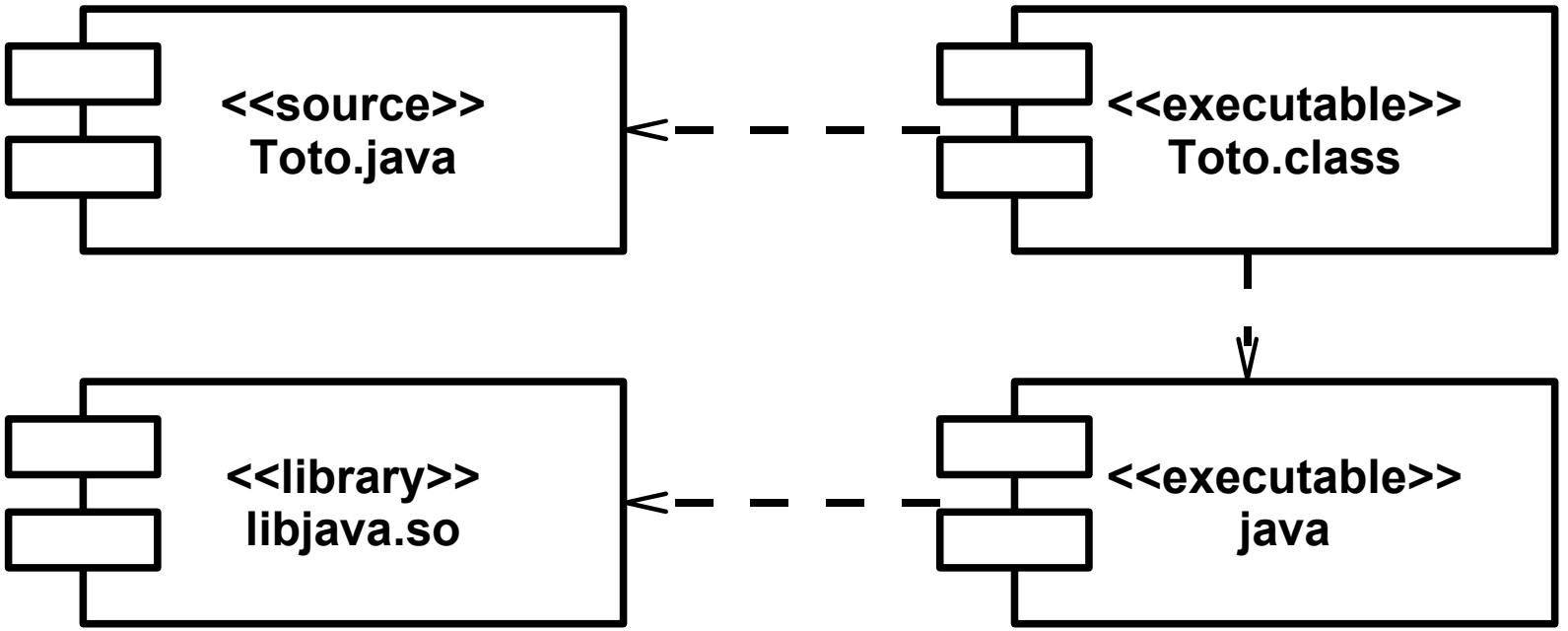
  - A l'aide d'une "coupe" :



Dépendances de composants

- Flèche pointillée du composant dépendant à celui qui fournit l'aide,
  - Exemple :

    Composants pour une classe exécutable Java :



Diagrammes UML

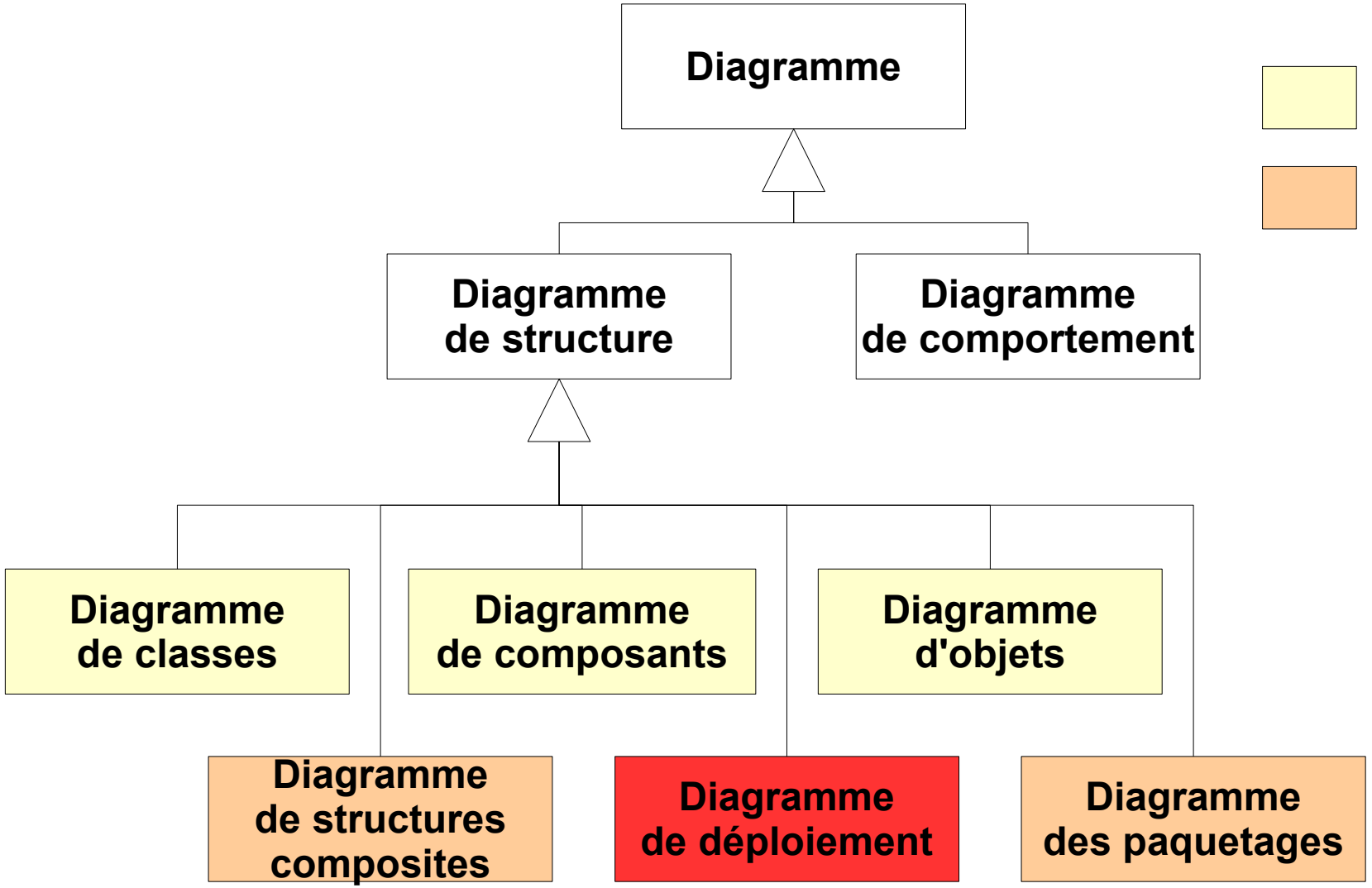


Diagramme de déploiement

- L'implantation physique doit tenir compte de trois problèmes :
  - Le logiciel,
    - Diagramme de composants,
  - Le matériel,
    - Diagramme de déploiement,
  - L'intégration des deux élements :
    - Ensemble des deux diagrammes.

Diagramme de déploiement

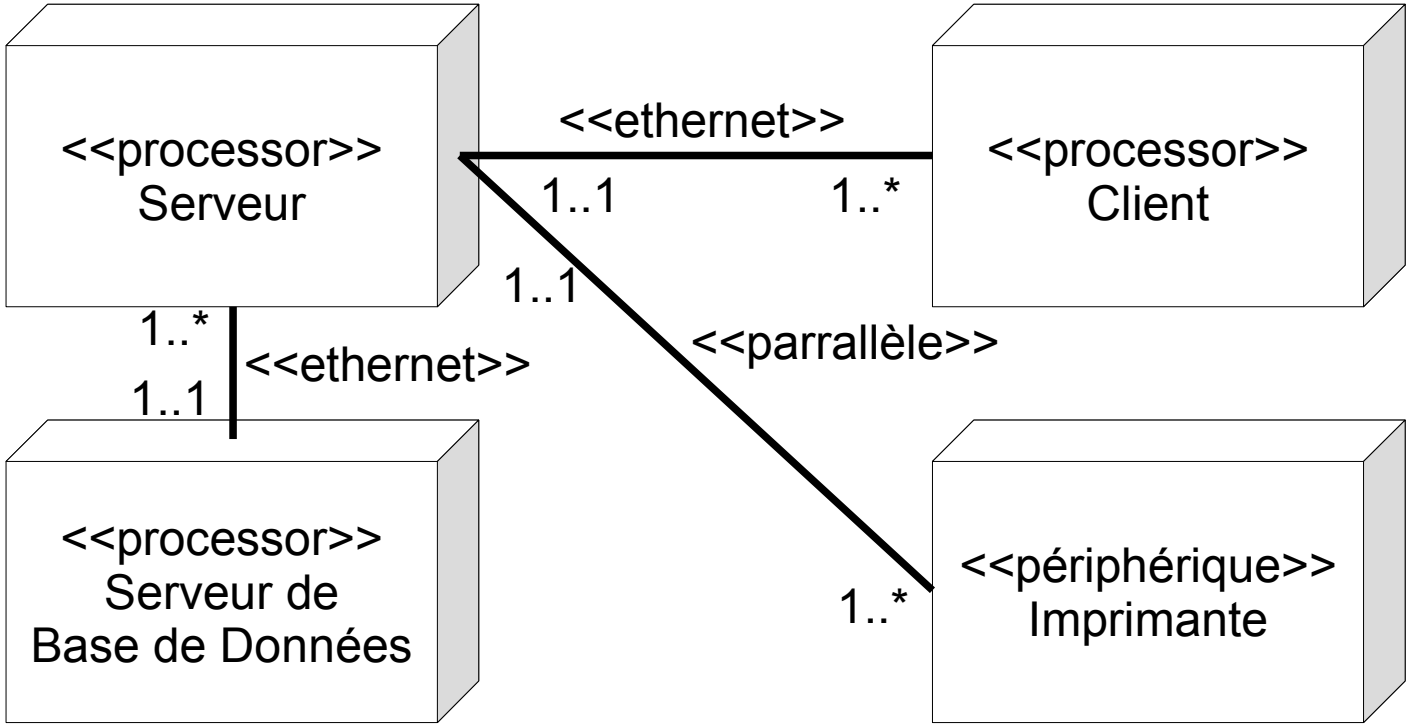
- Montre la disposition physique des matériels,
- Montre la répartition des composants,
- Ressources matérielles : sous forme de noeuds,
- Connections sous forme de supports de communication,
- Montre des instances de noeuds ou des classes de noeuds,
- Correspond à la vue de déploiement.

Diagramme de déploiement

- Composé :
  - Des artefacts : les pièces physiques d'information utilisées ou produites par le SI,
    - Les artefacts sont équivalents aux composants,
    - **Les notations de UML1.4 et UML 2.0 diffèrent !**
  - Les Chemins de communication : des associations entre 2 cibles de déploiement au travers desquelles les informations sont échangées,
  - Les noeuds, les ressources de calcul sur lesquelles les artefacts peuvent être déployés,

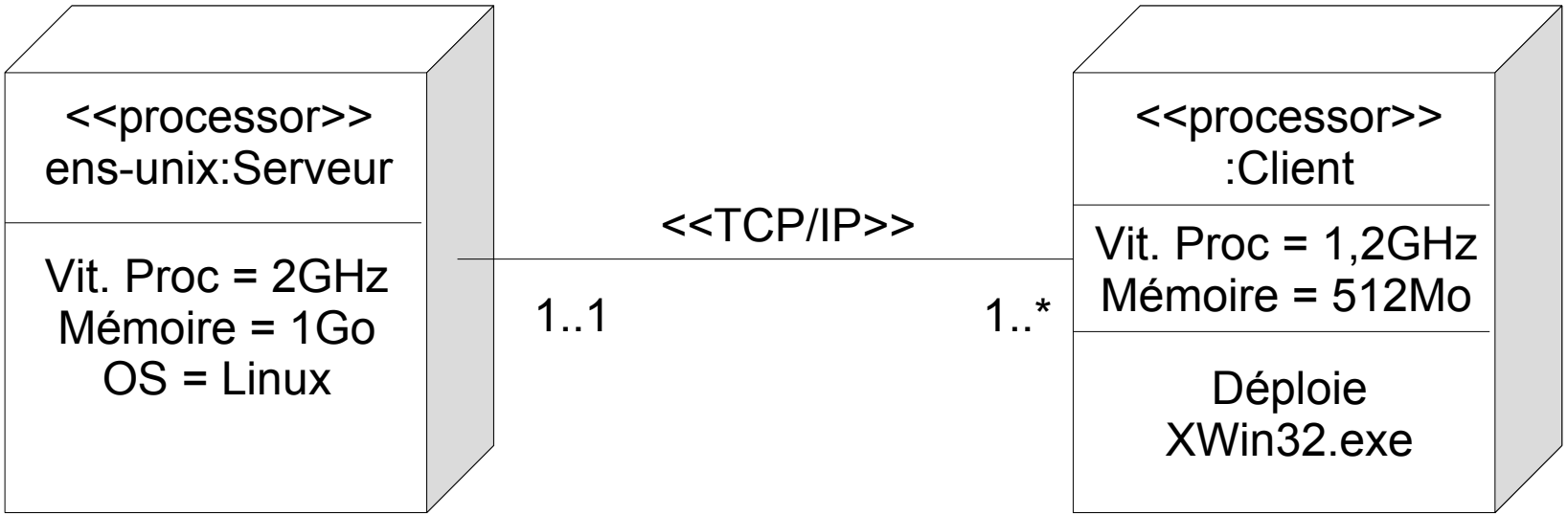
Premières notations

- Noeud : boîte 3D,
- Associations : traits pleins,
- Les 2 entités peuvent être stéréotypées !



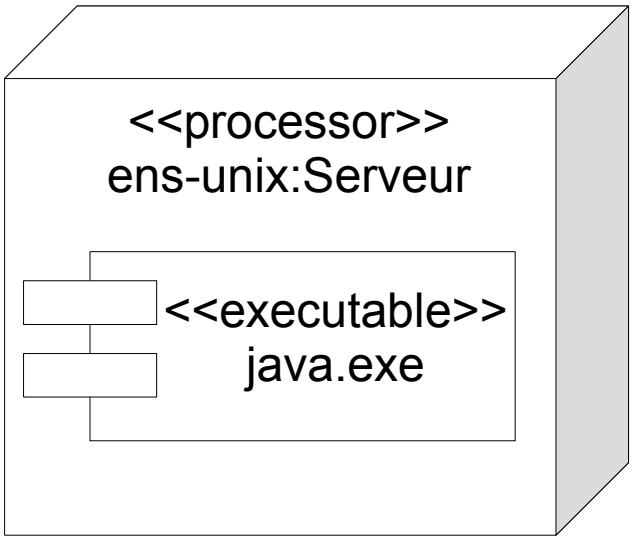
Contraintes sur les noeuds

- Un noeud peut posséder des compartiments avec attributs et opérations :

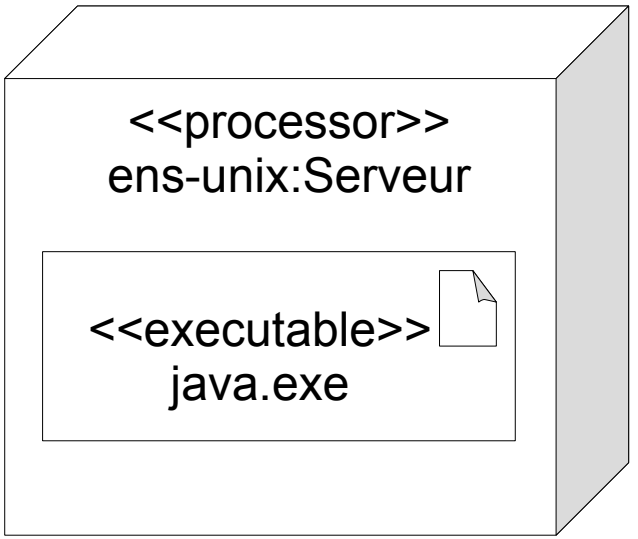


Artéfacts déployés

- Les artéfacts sont mis à l'intérieur des noeuds :



Notation UML 1.4



Notation UML 2.0

- Il est possible de combiner diagramme de composants et diagramme de déploiement dans un seul diagramme.

Diagrammes UML

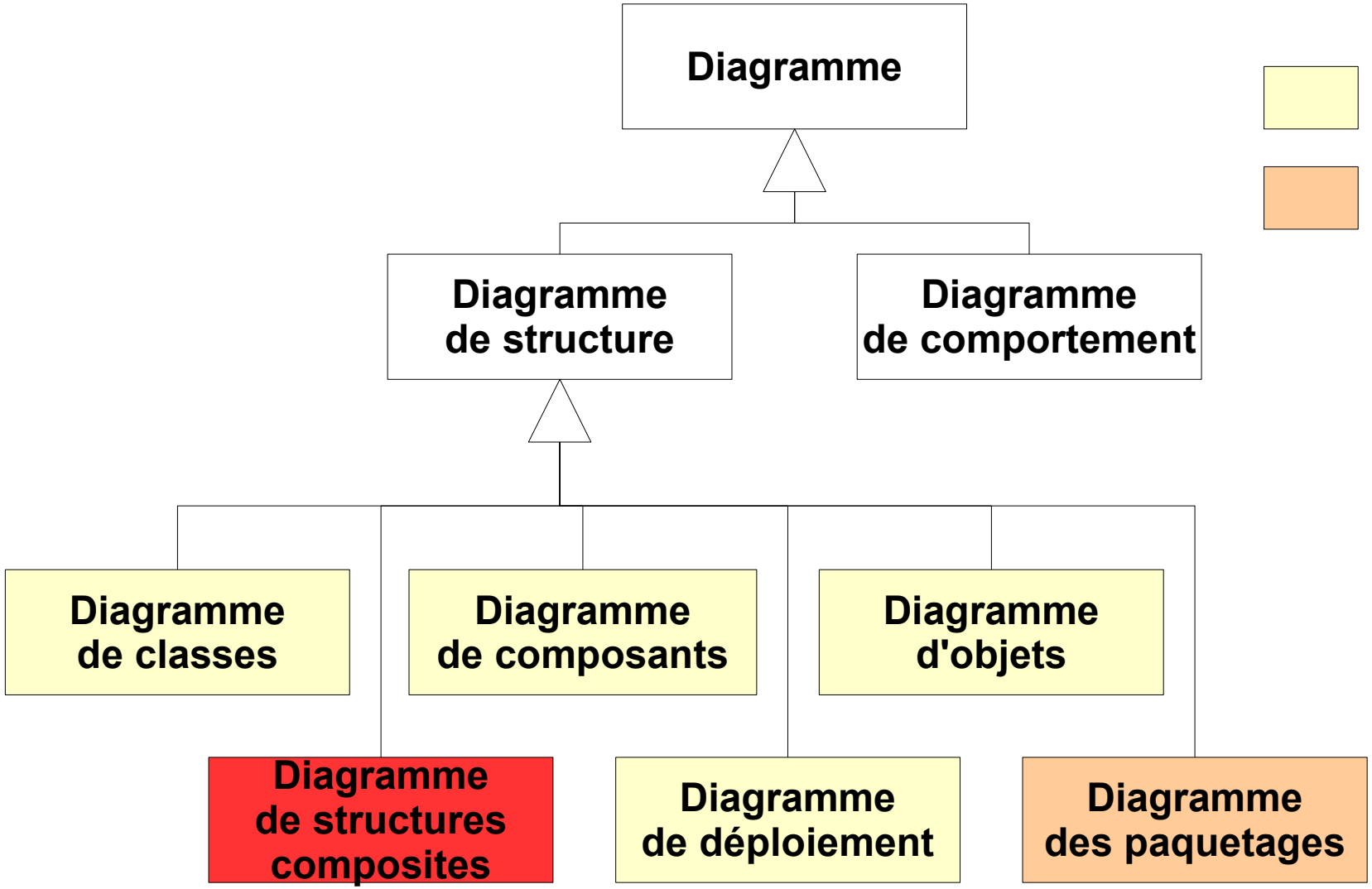


Diagramme de structures composites

- Structure composite :
  - Composition d'éléments interconnectés représentant des instances collaborant par le biais de liens de communication afin de parachever des objectifs communs.
- 5 éléments :
  - Les Classificateur structuré,
  - Les parties,
  - Les ports,
  - Les connecteurs,
  - Les collaborations,

Diagramme de structures composites

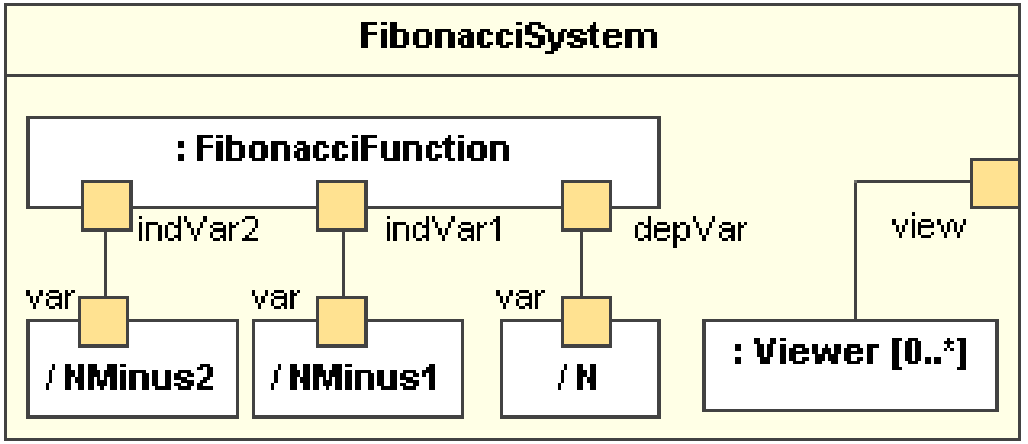
- Classificateur structuré :
  - Classe, souvent abstraite, dont le comportement peut-être déterminé suivant ses interactions avec les différentes parties,
- Partie :
  - Rôle joué à l'exécution par une instance ou une collection d'instances de classes,
- Port :
  - Point d'interaction utilisé pour connecter les classificateurs structurés avec leurs parties et leur environnement.

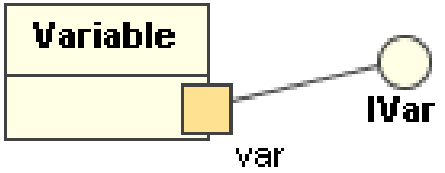
Diagramme de structures composites

- Connecteur :
  - Lien entre au minimum 2 entités,
- Collaboration :
  - Structure purement abstraite,
  - Représentée comme un ovale pointillé contenant le rôle que l'instance peut jouer dans la collaboration.

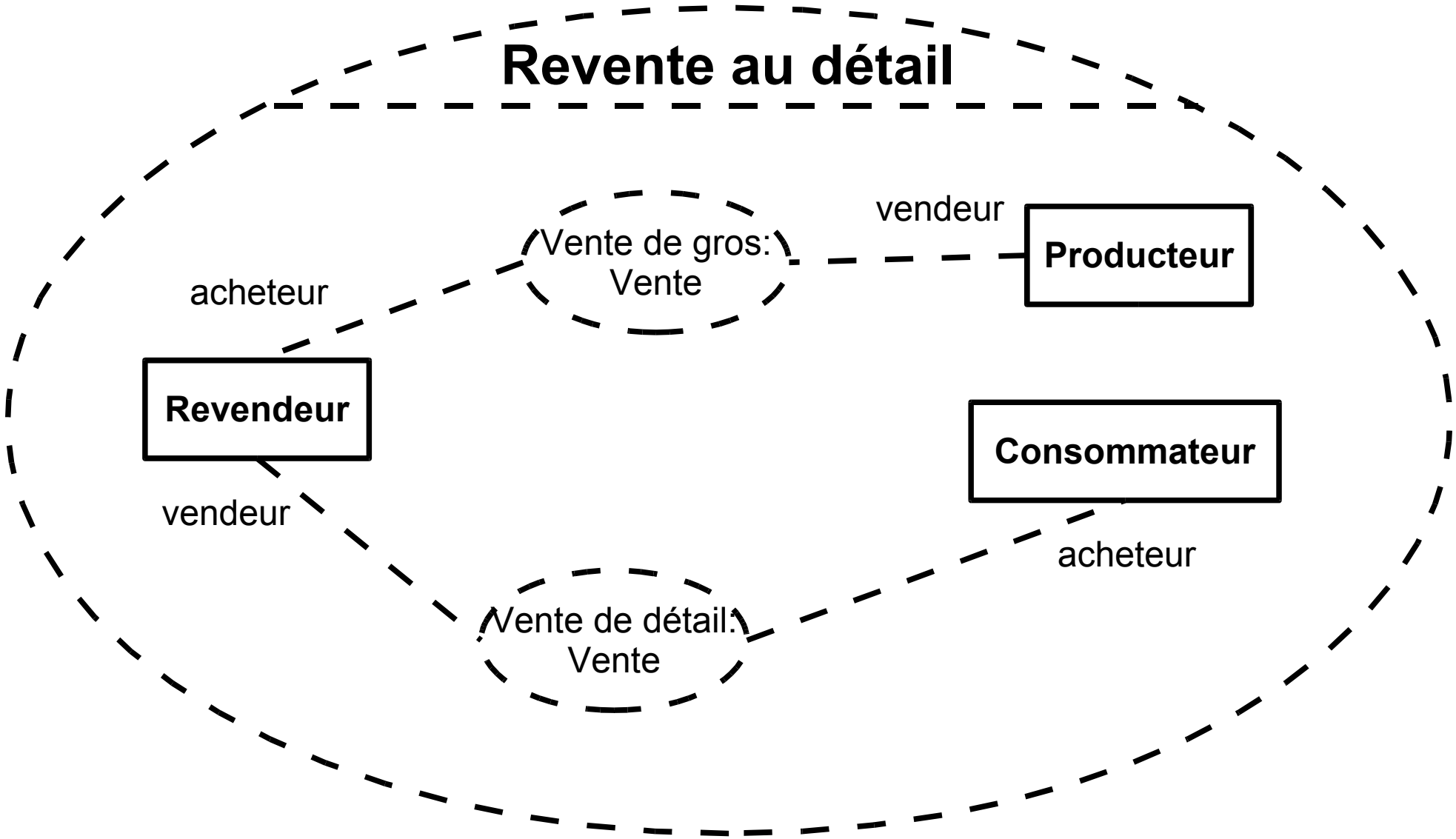
Exemple de diagramme de structures composite

- La suite de Fibonacci :
  - F(n) = F(n-1) + F(n-2), F(0) = 0, F(1) = 1,
  - Les parties sont *NMinus2, NMinus1 et N,*





Exemple de diagramme de structures composite



Diagrammes UML

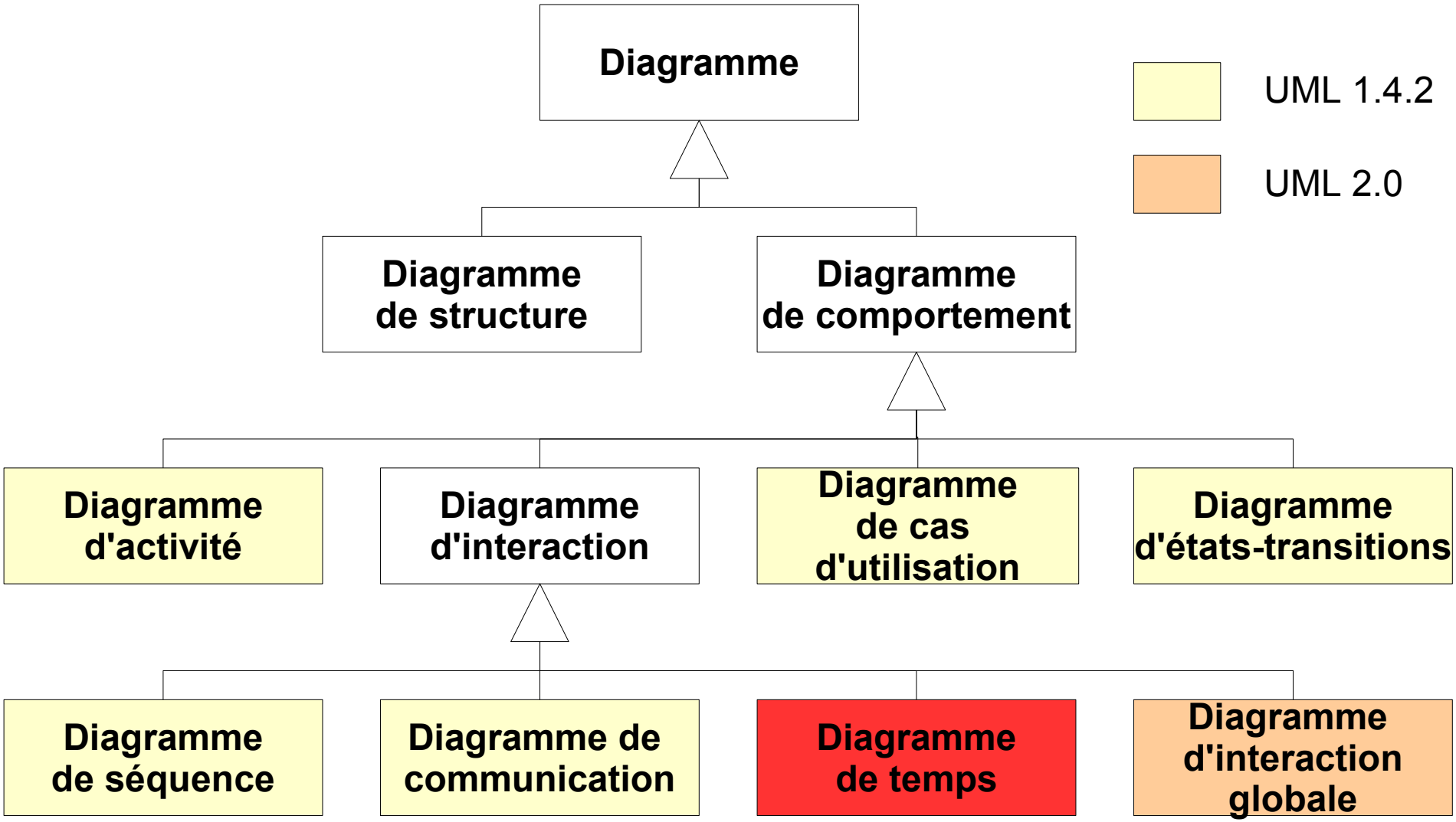
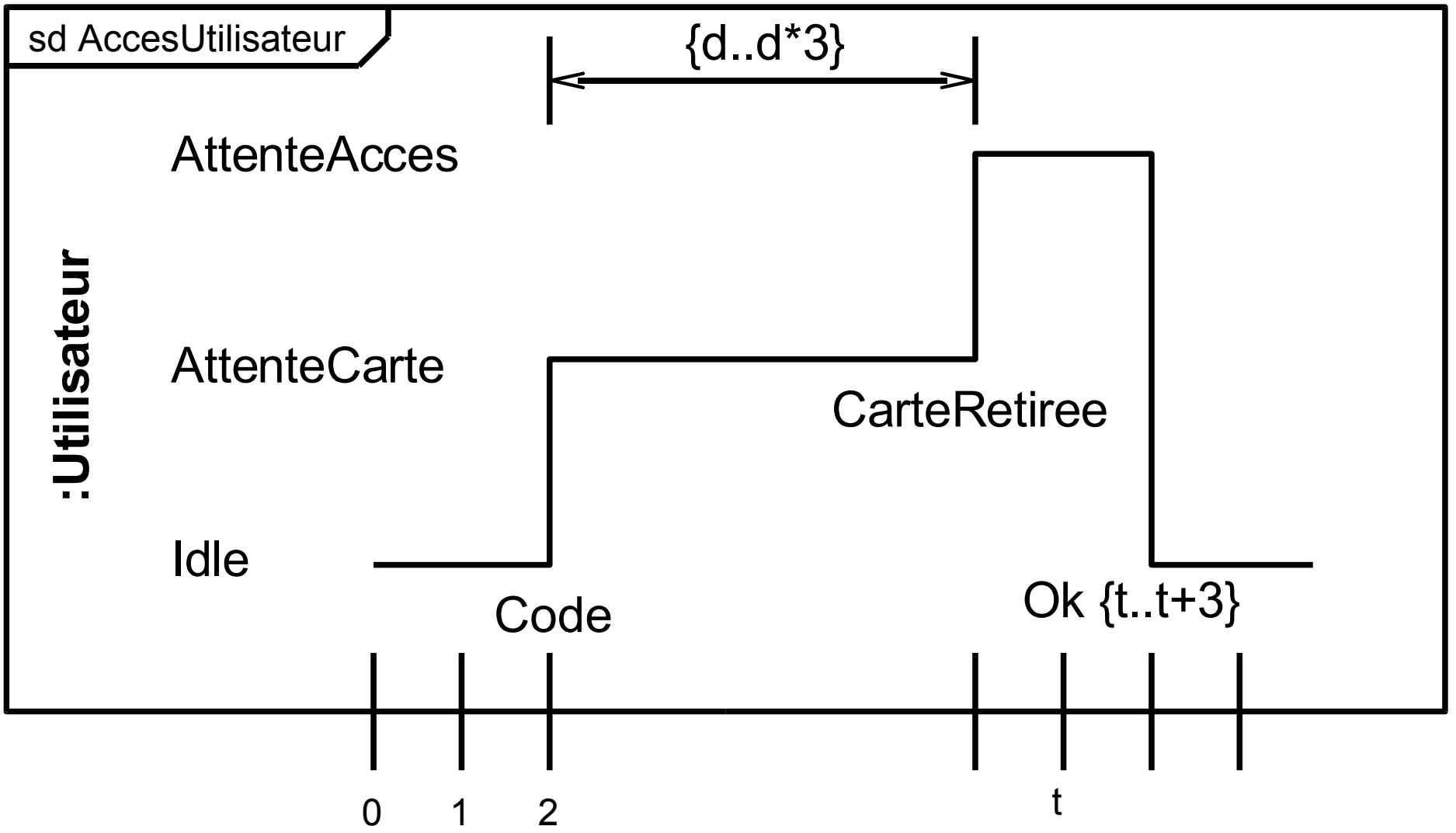


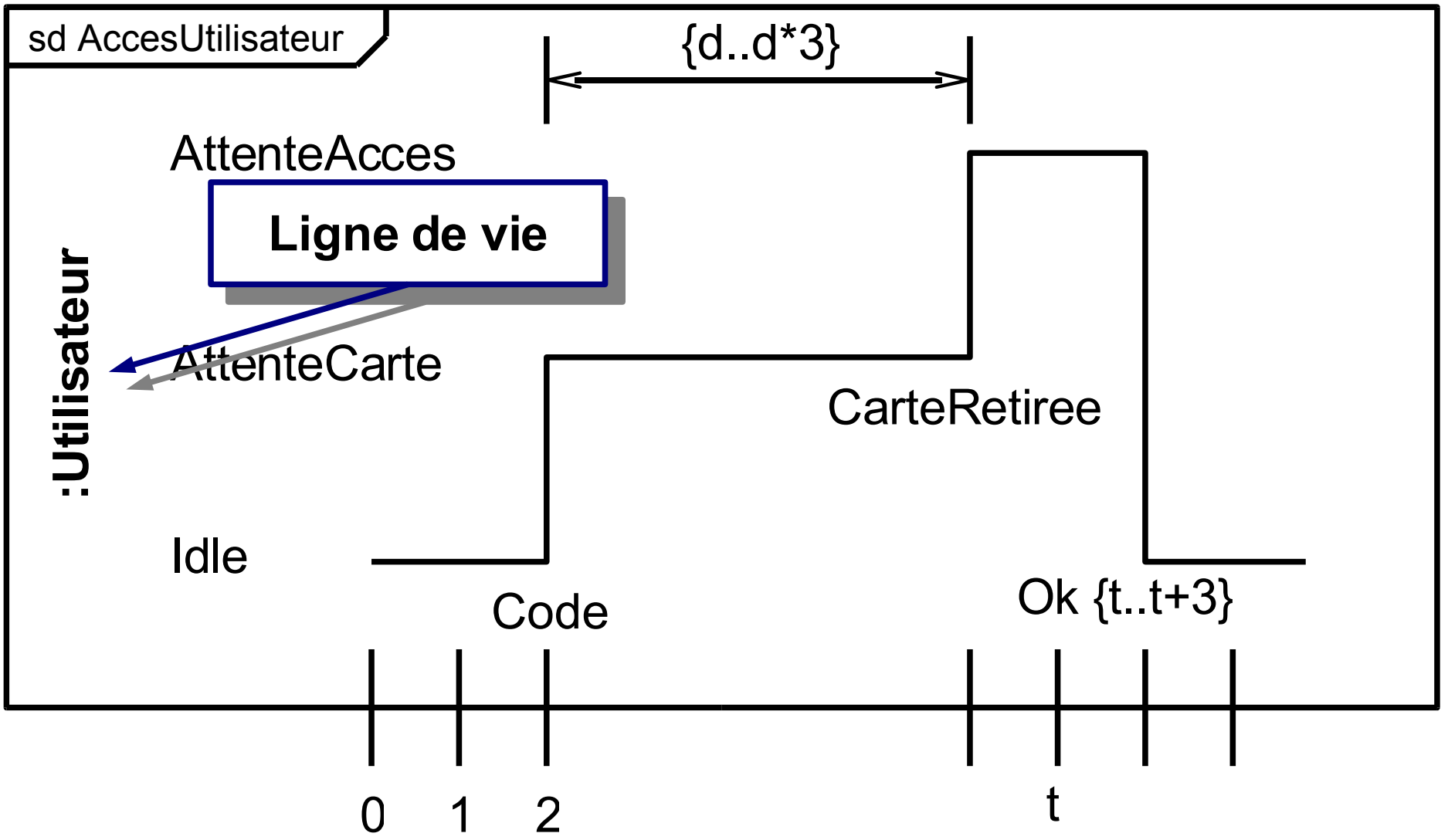
Diagramme de temps

- Se concentre sur les conditions de changement à l'intérieur et parmi les lignes de vies suivant un axe linéaire. Complémentaire au diagramme de séquences,
- Décrit le comportement du classificateur et des interactions avec le classificateur, se concentrant sur l'instant d'occurrence des évènements modifiant l'état du classificateur,
- Comparable à un chronogramme,
- Notation des messages et des destructions similaires au diagramme de séquences,

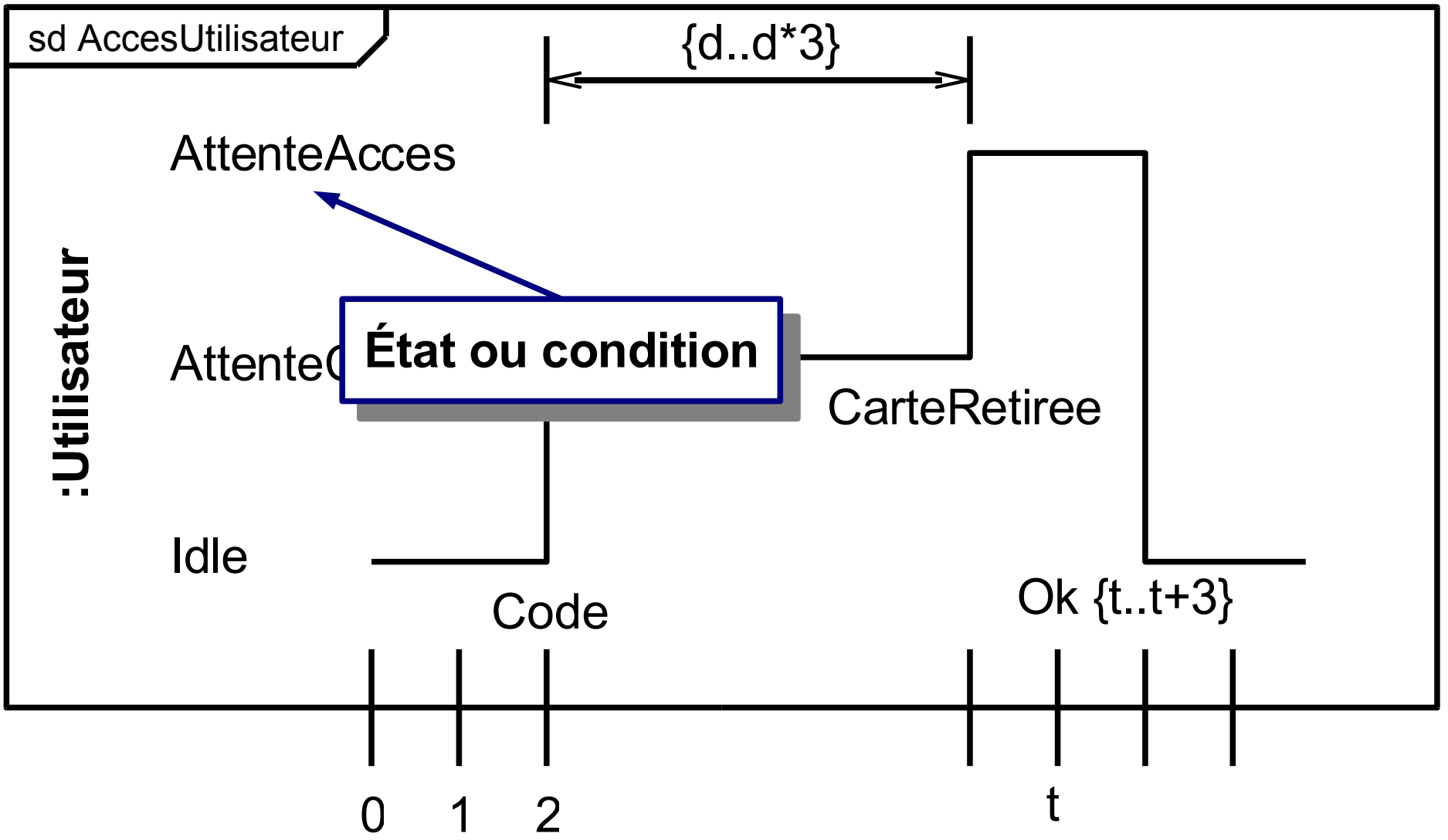
Exemple de diagramme de temps



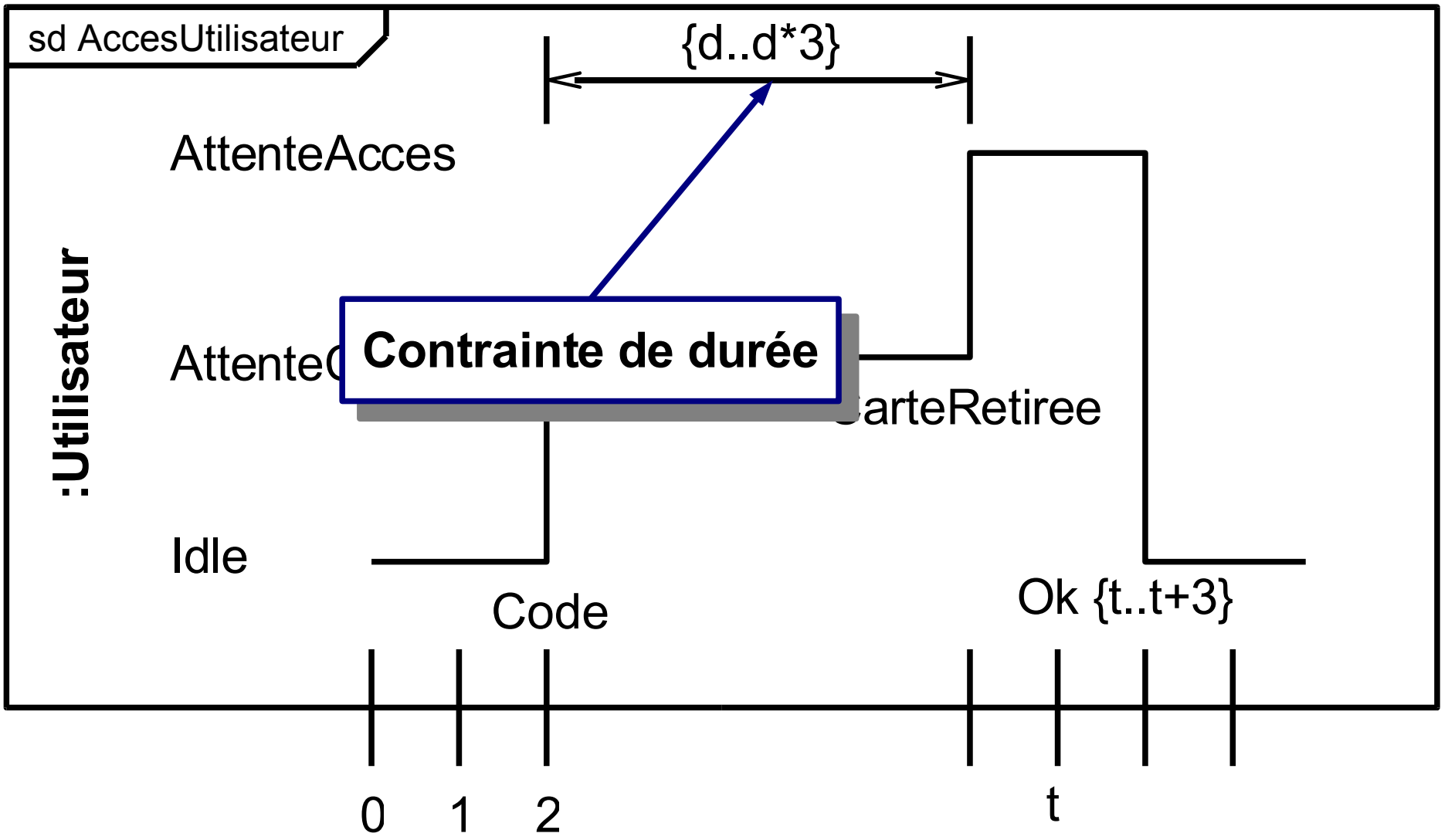
Exemple de diagramme de temps



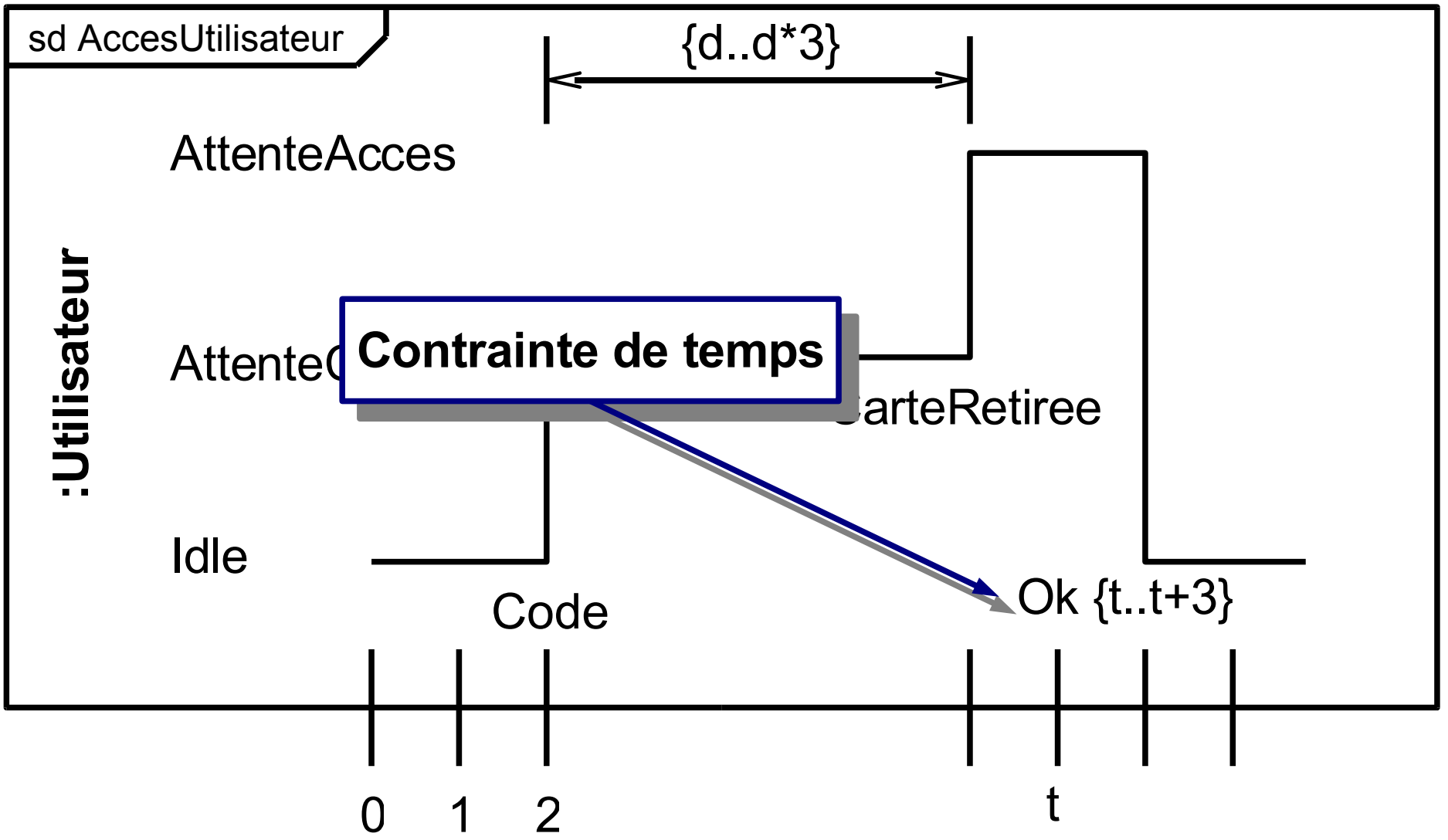
Exemple de diagramme de temps



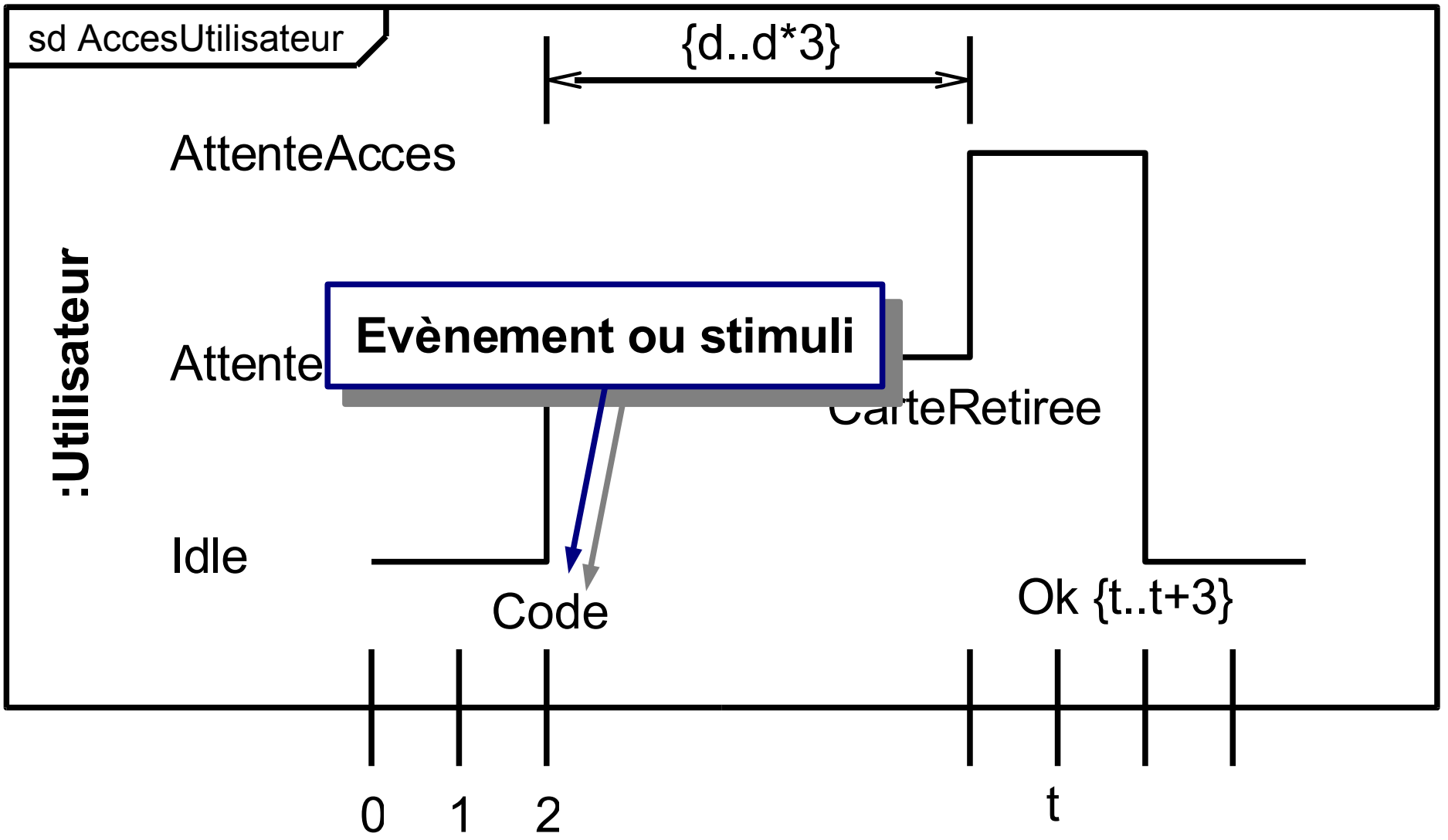
Exemple de diagramme de temps



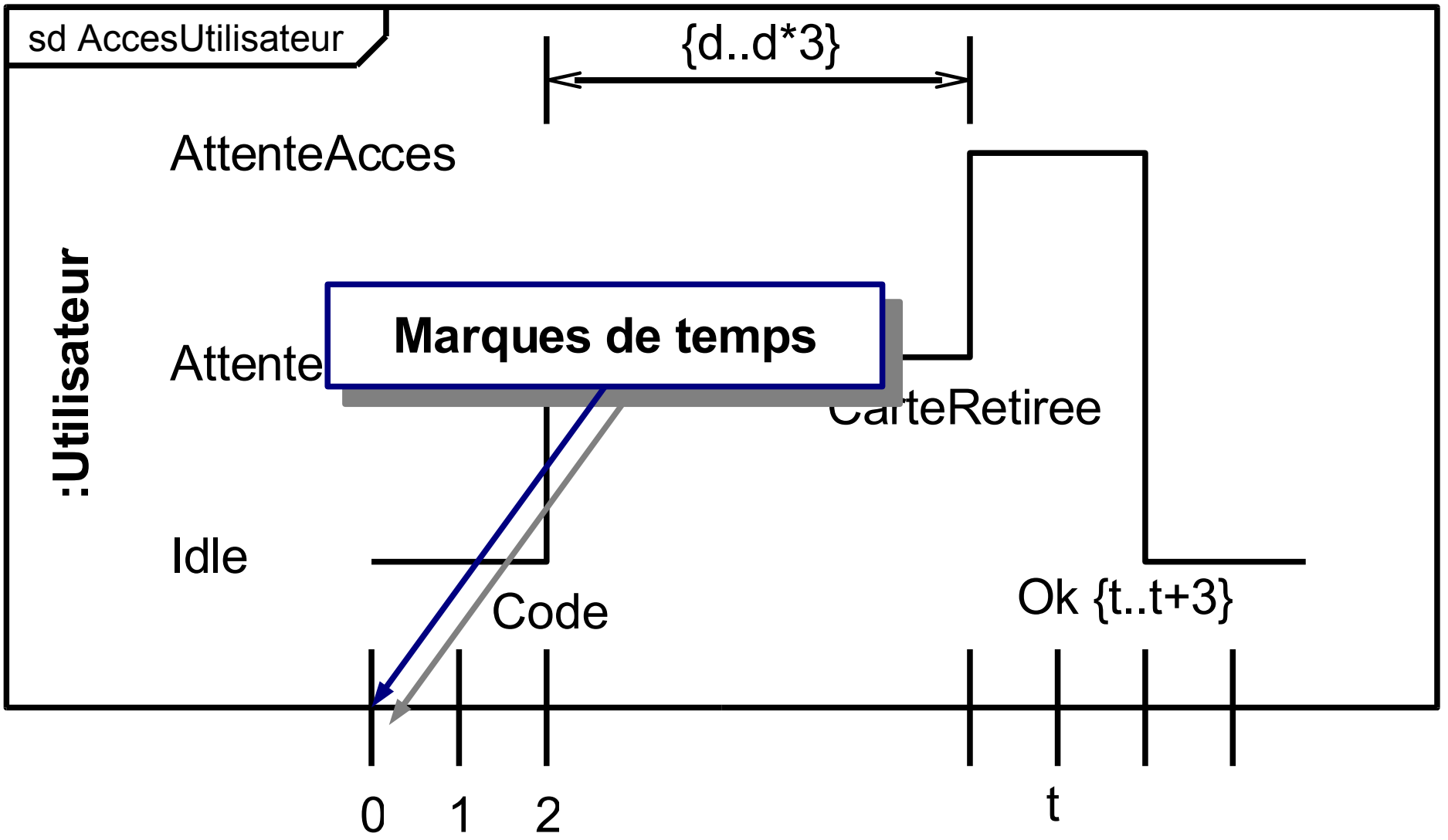
Exemple de diagramme de temps



Exemple de diagramme de temps

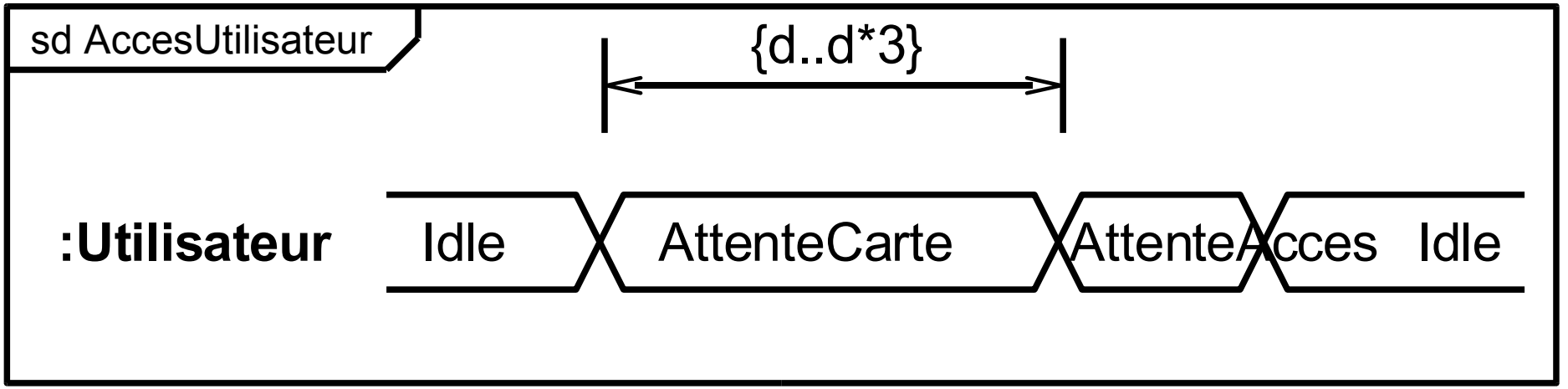


Exemple de diagramme de temps



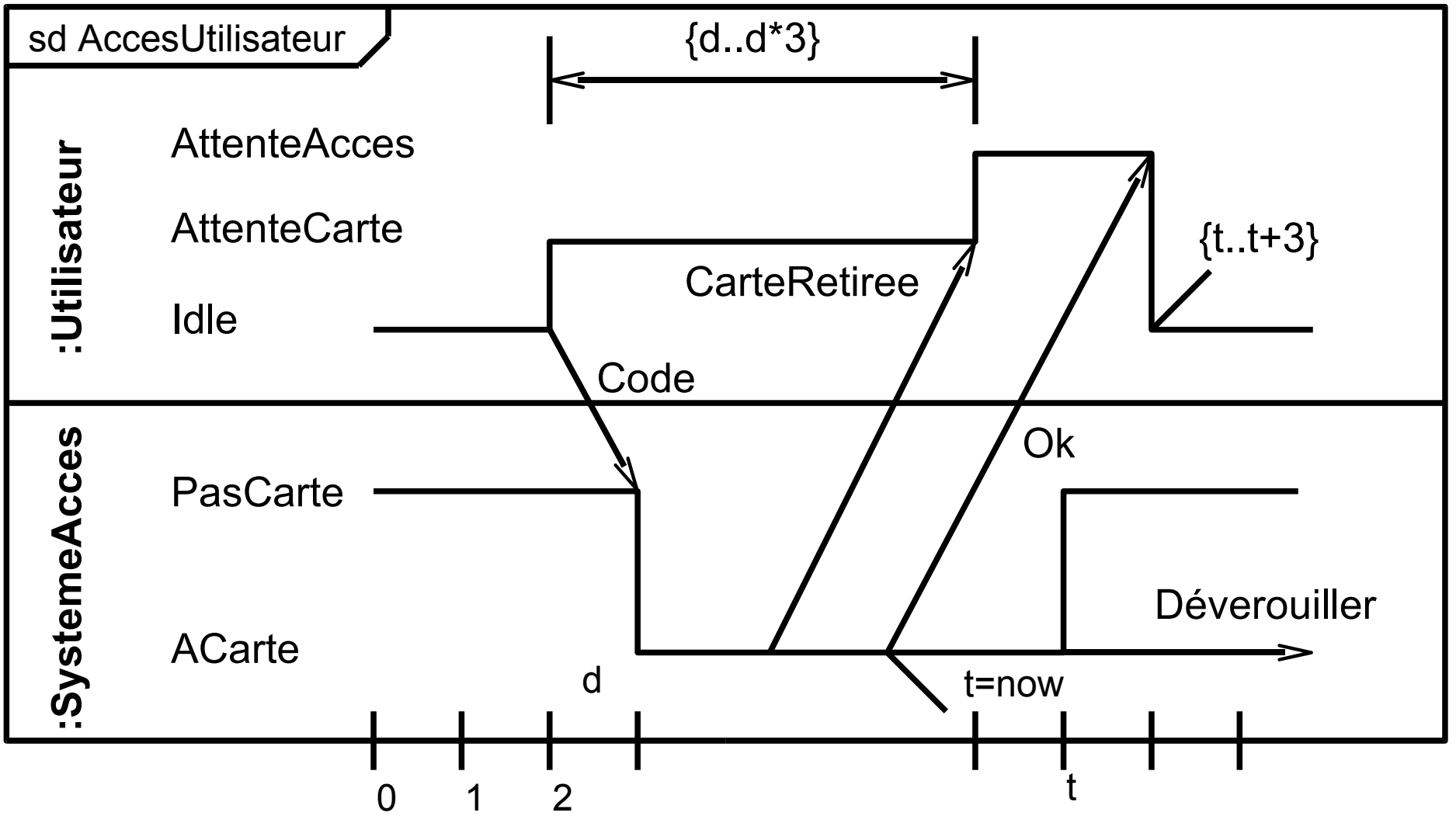
Exemple de diagramme de temps

- Notation compacte



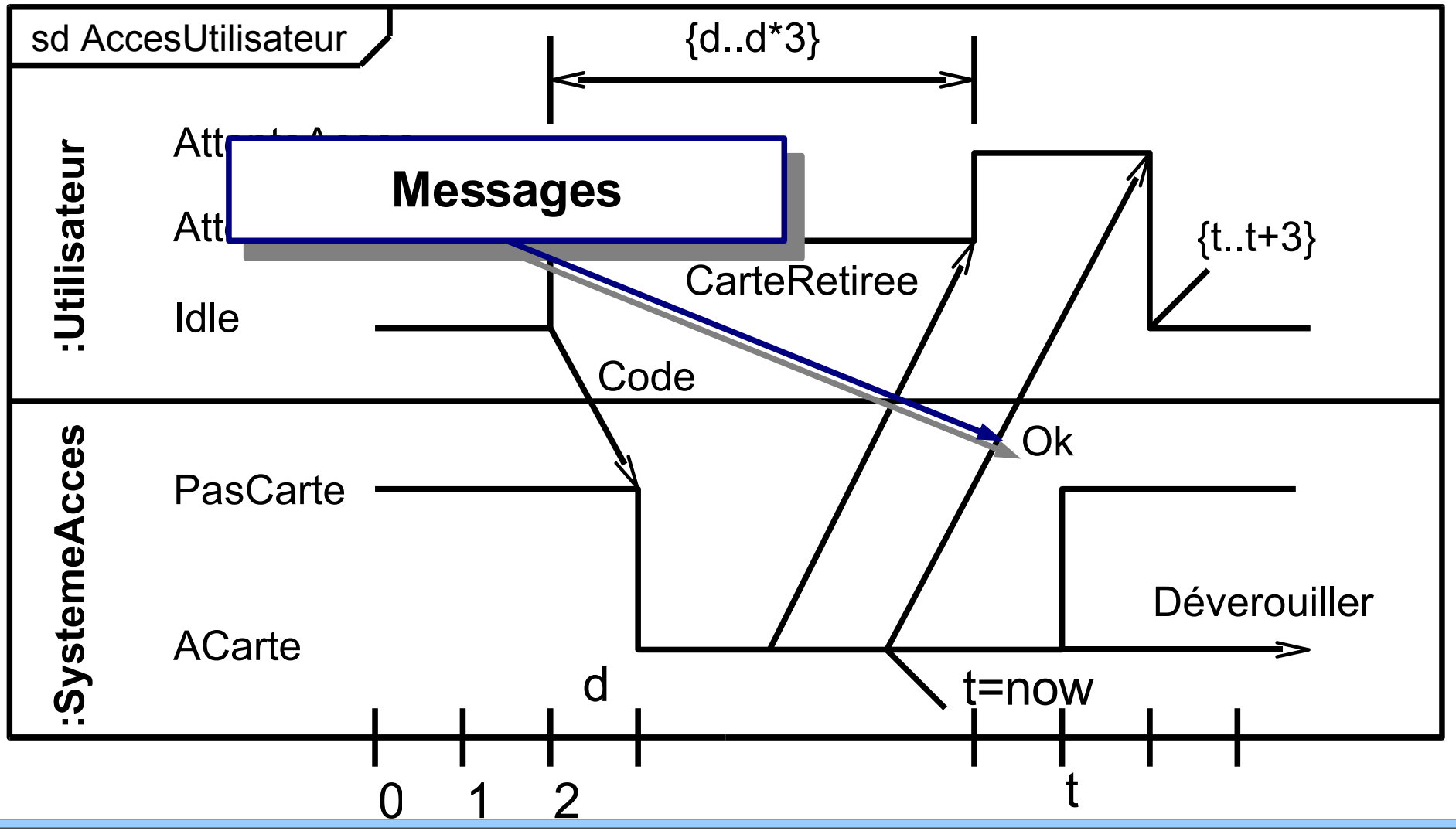
Exemple de diagramme de temps

- Avec plusieurs lignes de vie :



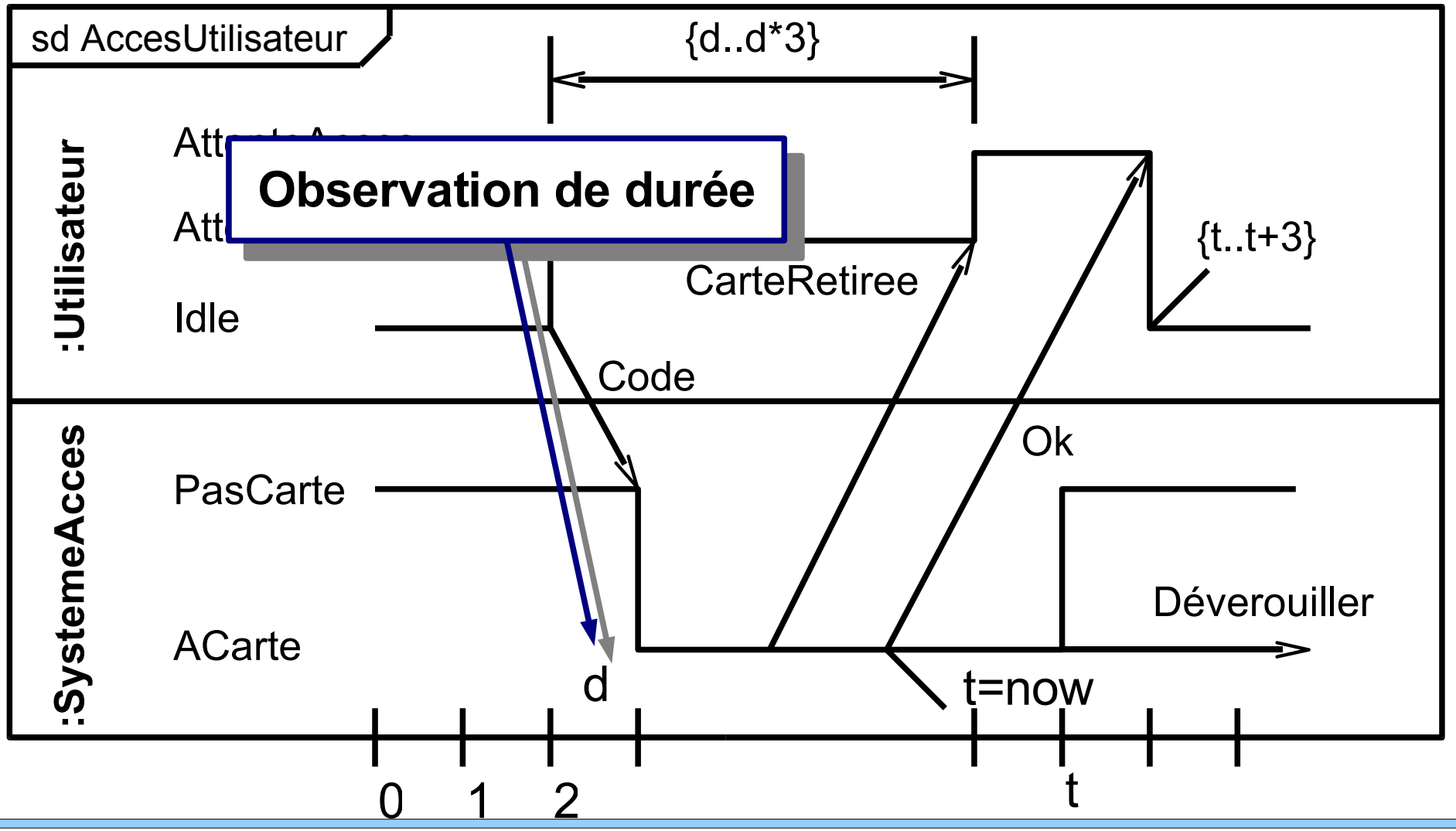
Exemple de diagramme de temps

- Avec plusieurs lignes de vie :



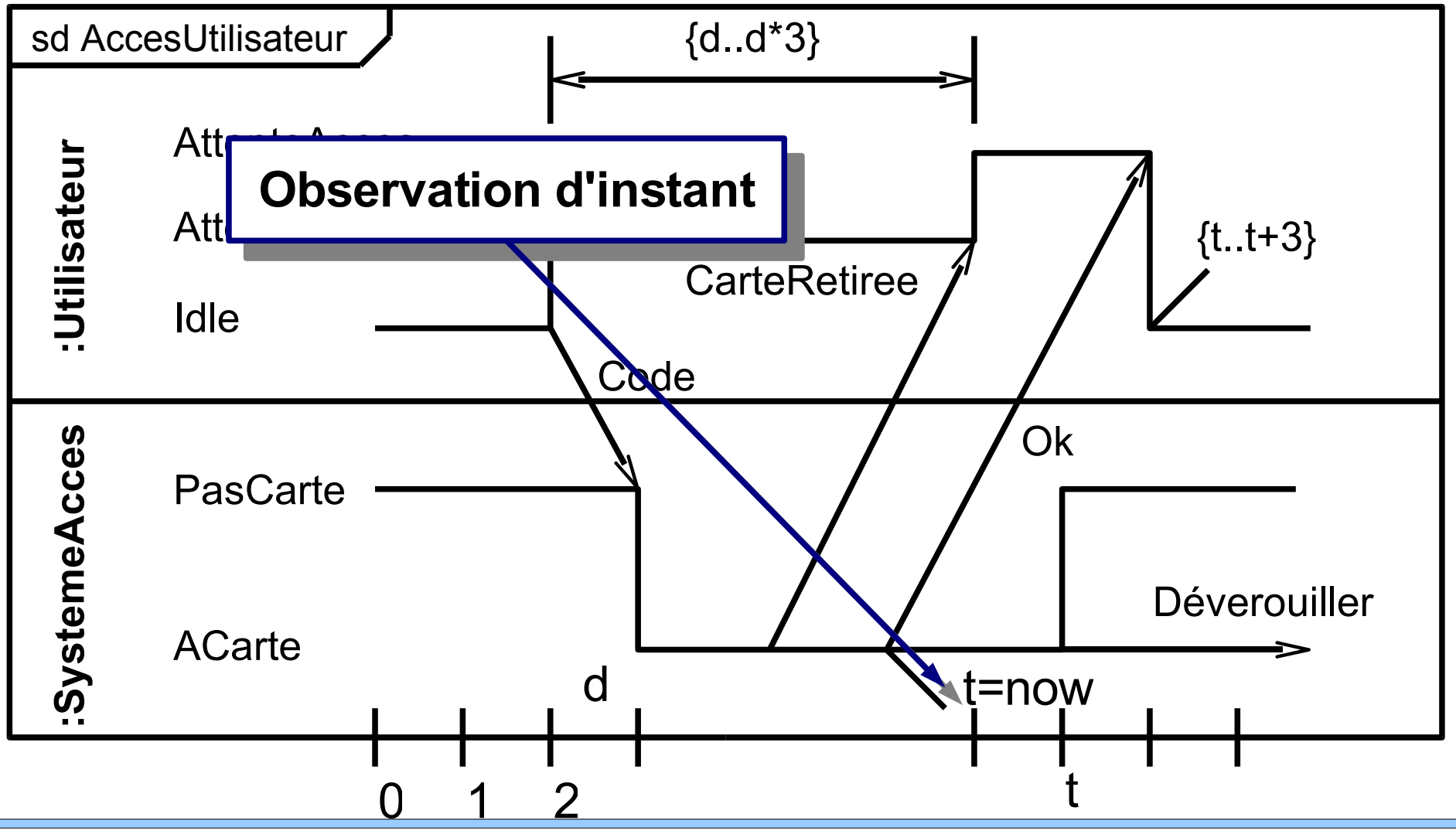
Exemple de diagramme de temps

- Avec plusieurs lignes de vie :



Exemple de diagramme de temps

- Avec plusieurs lignes de vie :



Diagrammes UML

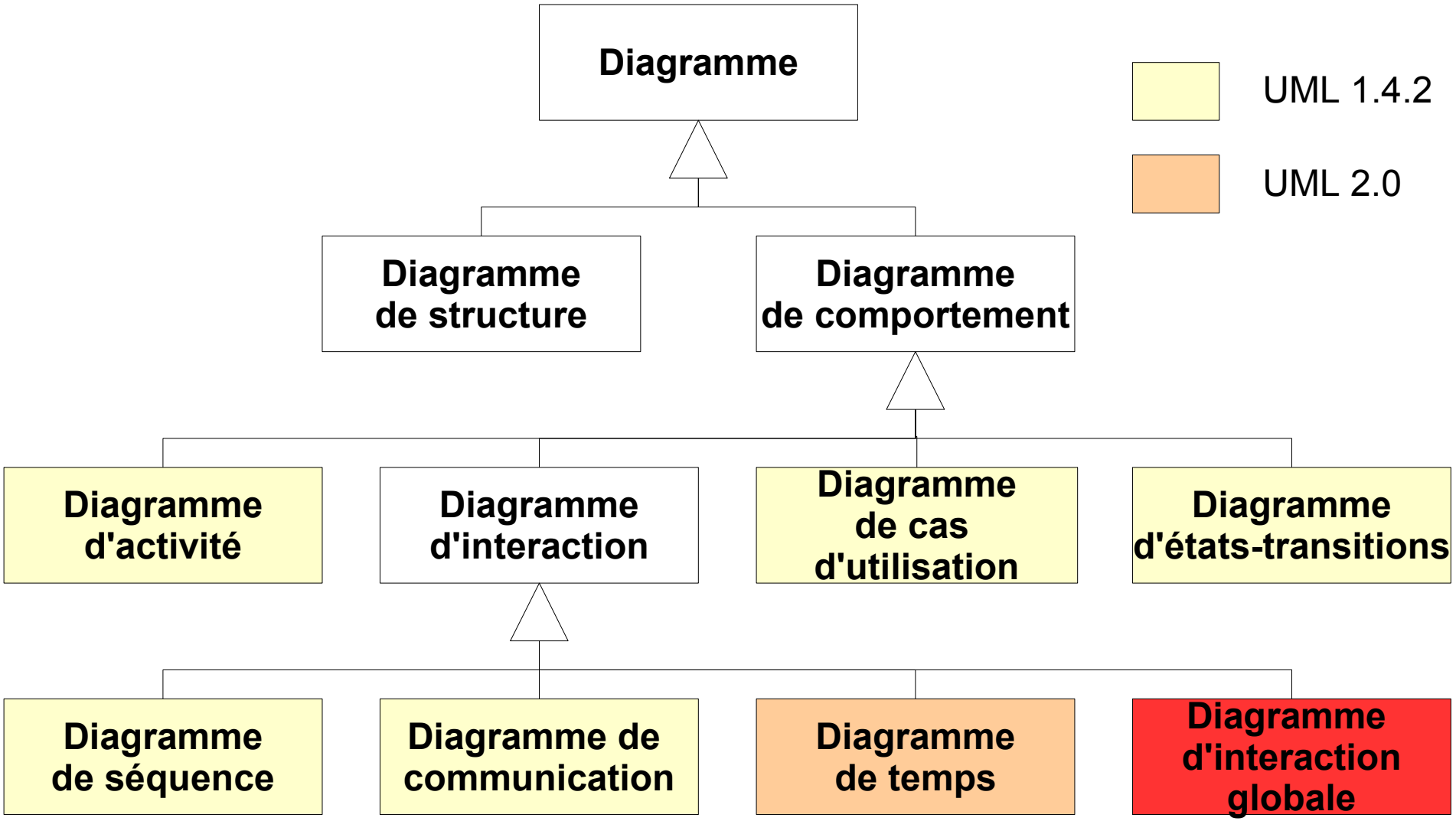
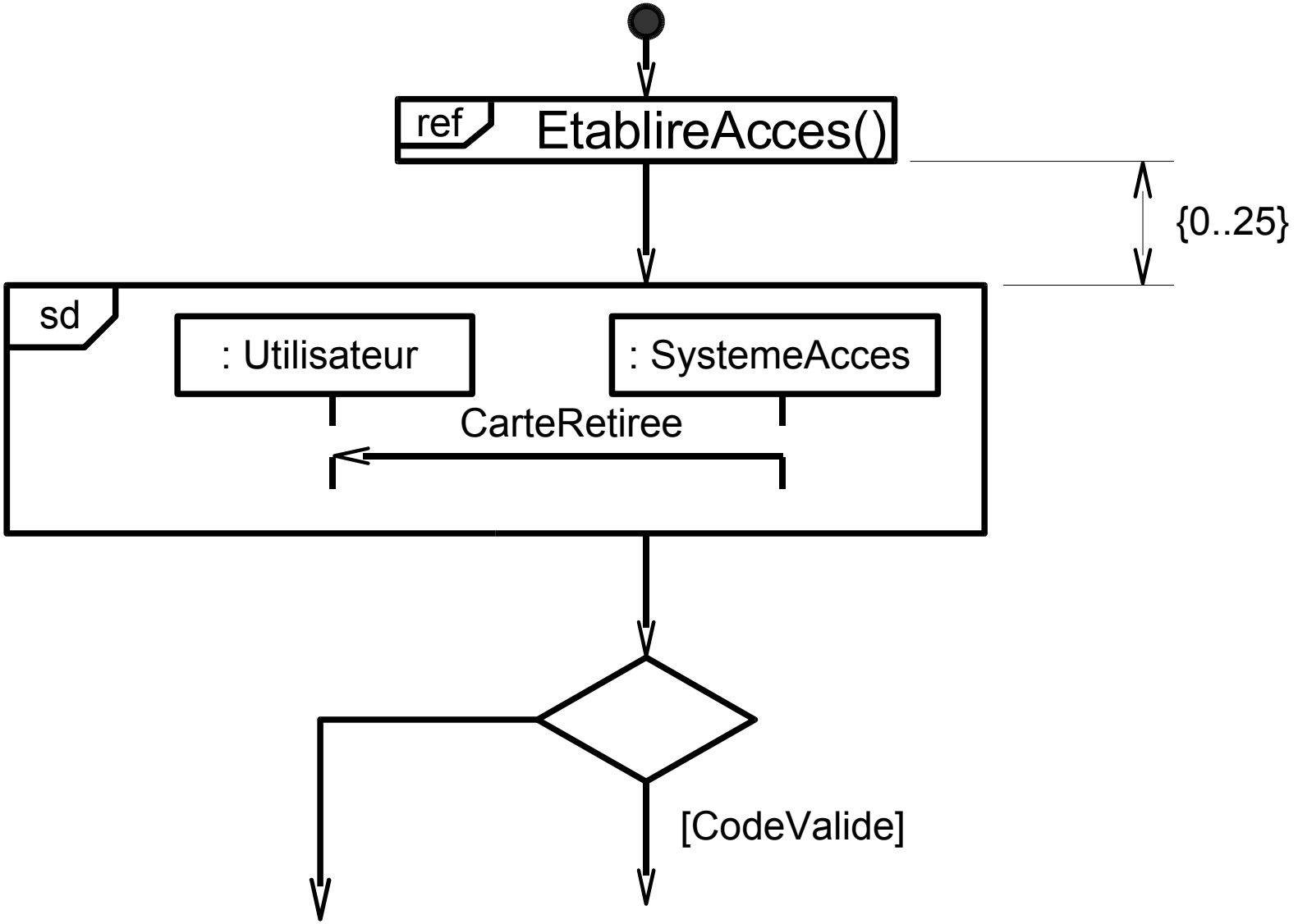


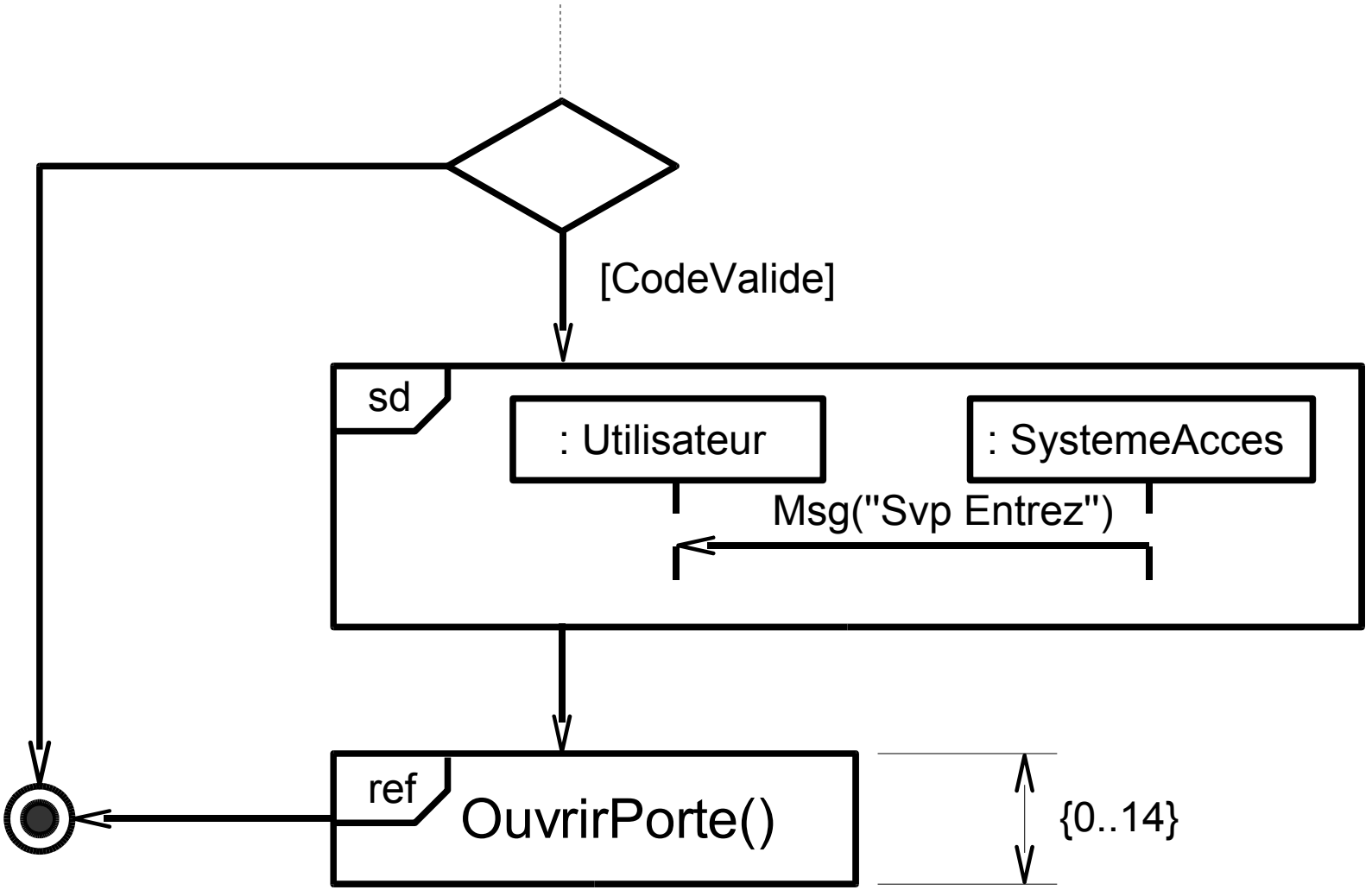
Diagramme d'interaction globale

- Définit les interactions au travers d'une variante du diagramme d'activités,
- Cherche à mettre en évidence une vue globale du flux de contrôle,
- Les noeuds sont des *interactions*,
- Les lignes de vie et les messages n'apparaissent pas à ce niveau,
- Mélange diagramme de séquences et diagramme d'activités.

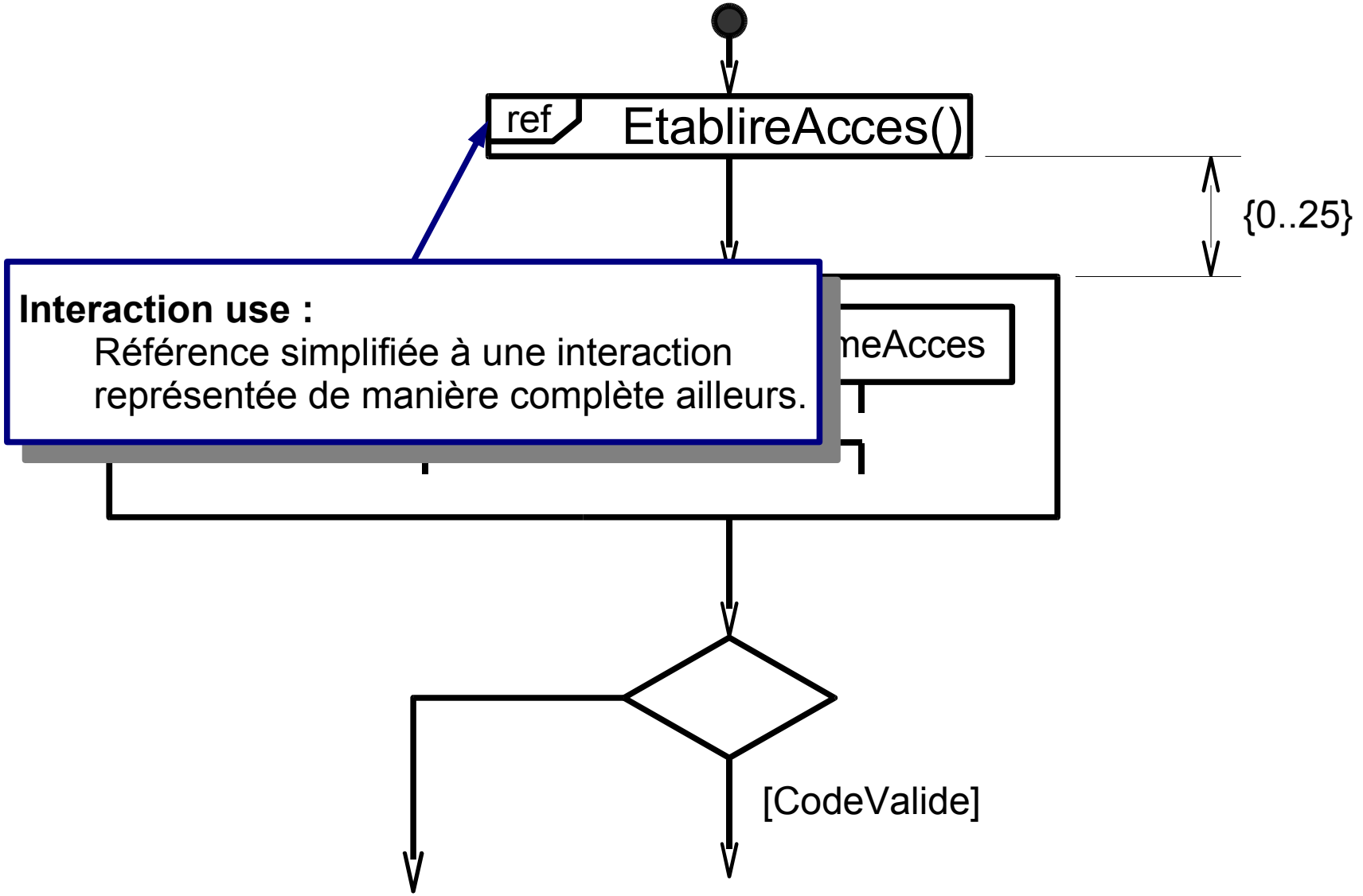
Exemple de diagramme d'interaction globale



Exemple de diagramme d'interaction globale

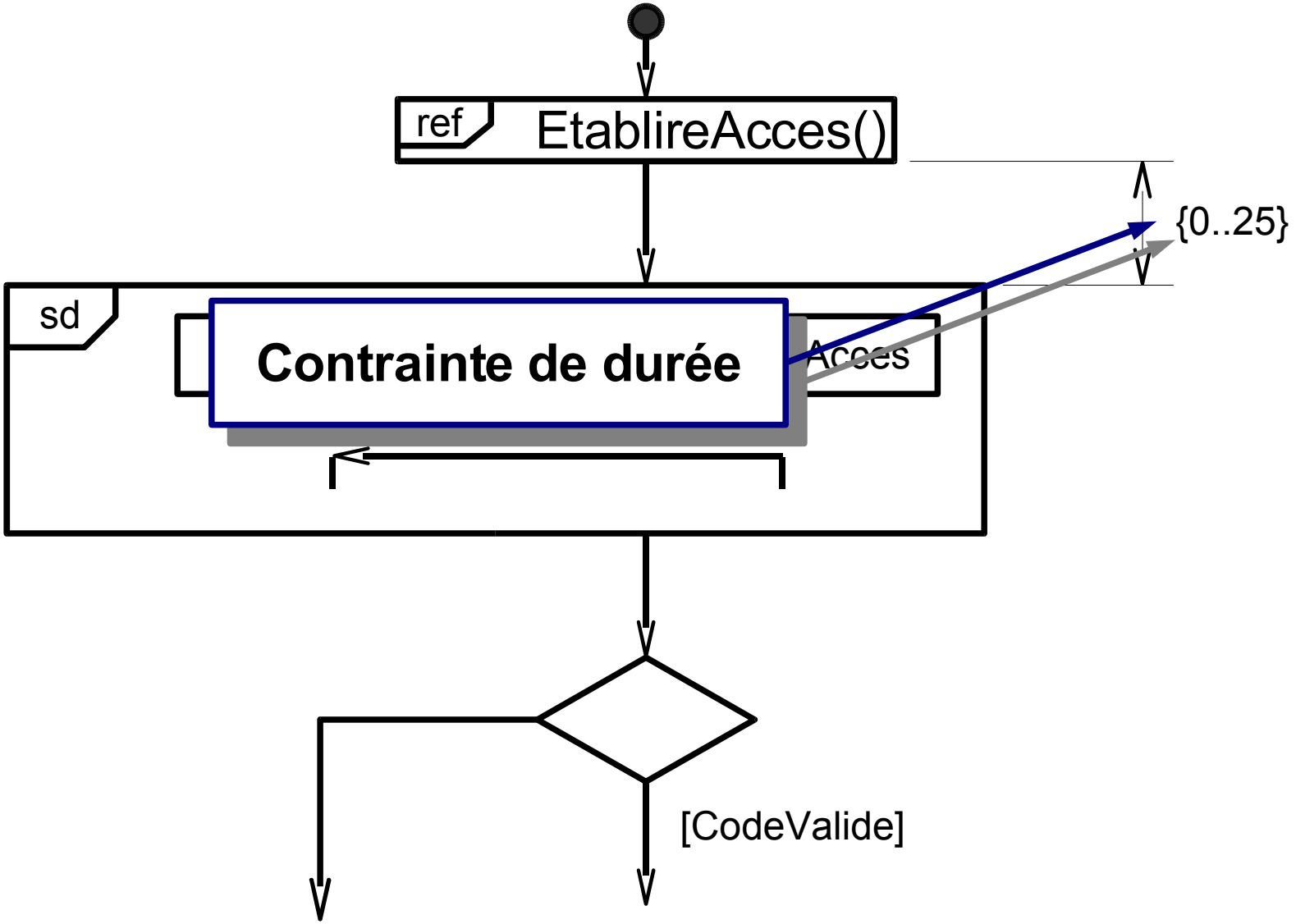


Exemple de diagramme d'interaction globale

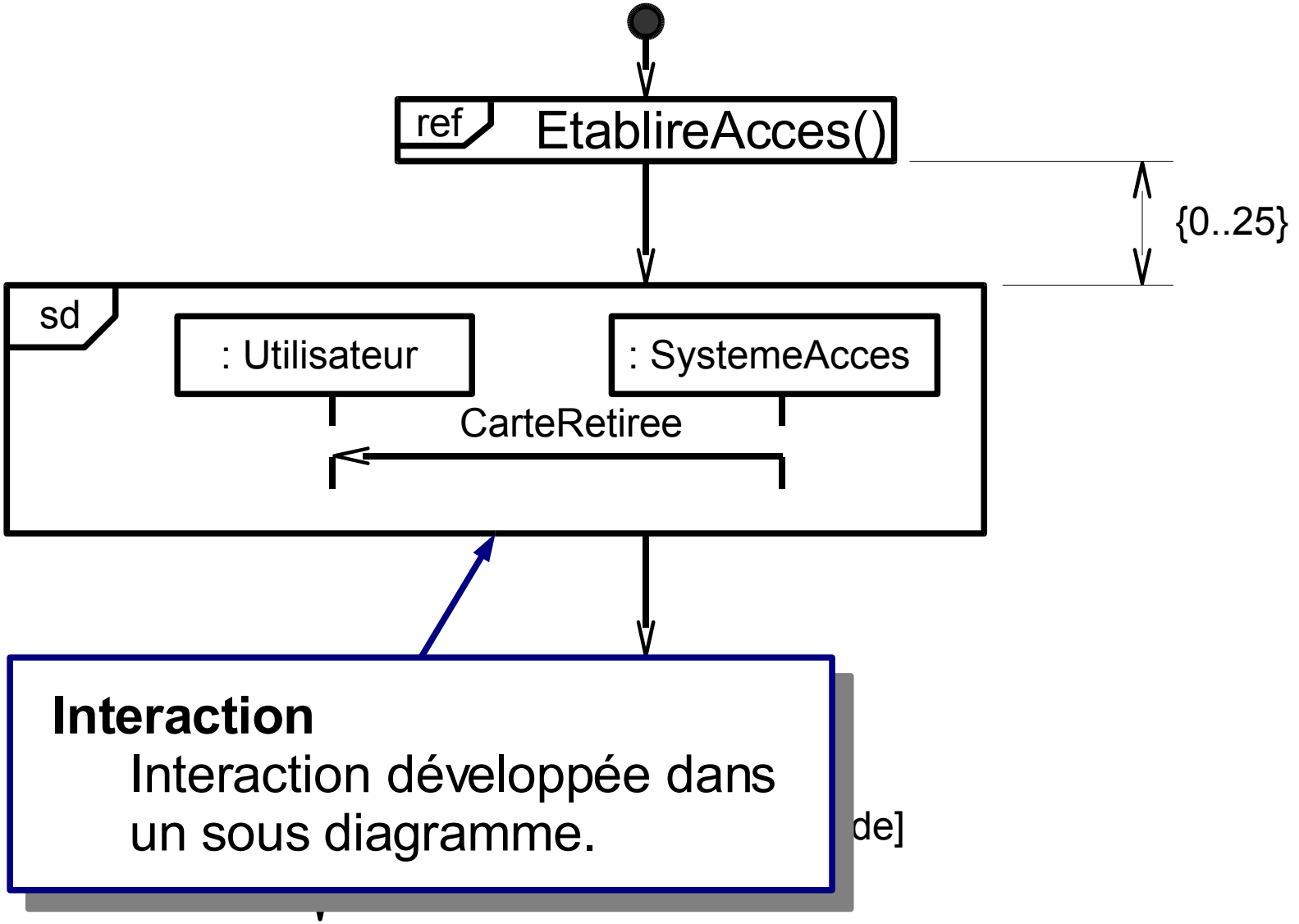


Interaction use :
Référence simplifiée à une interaction représentée de manière complète ailleurs.

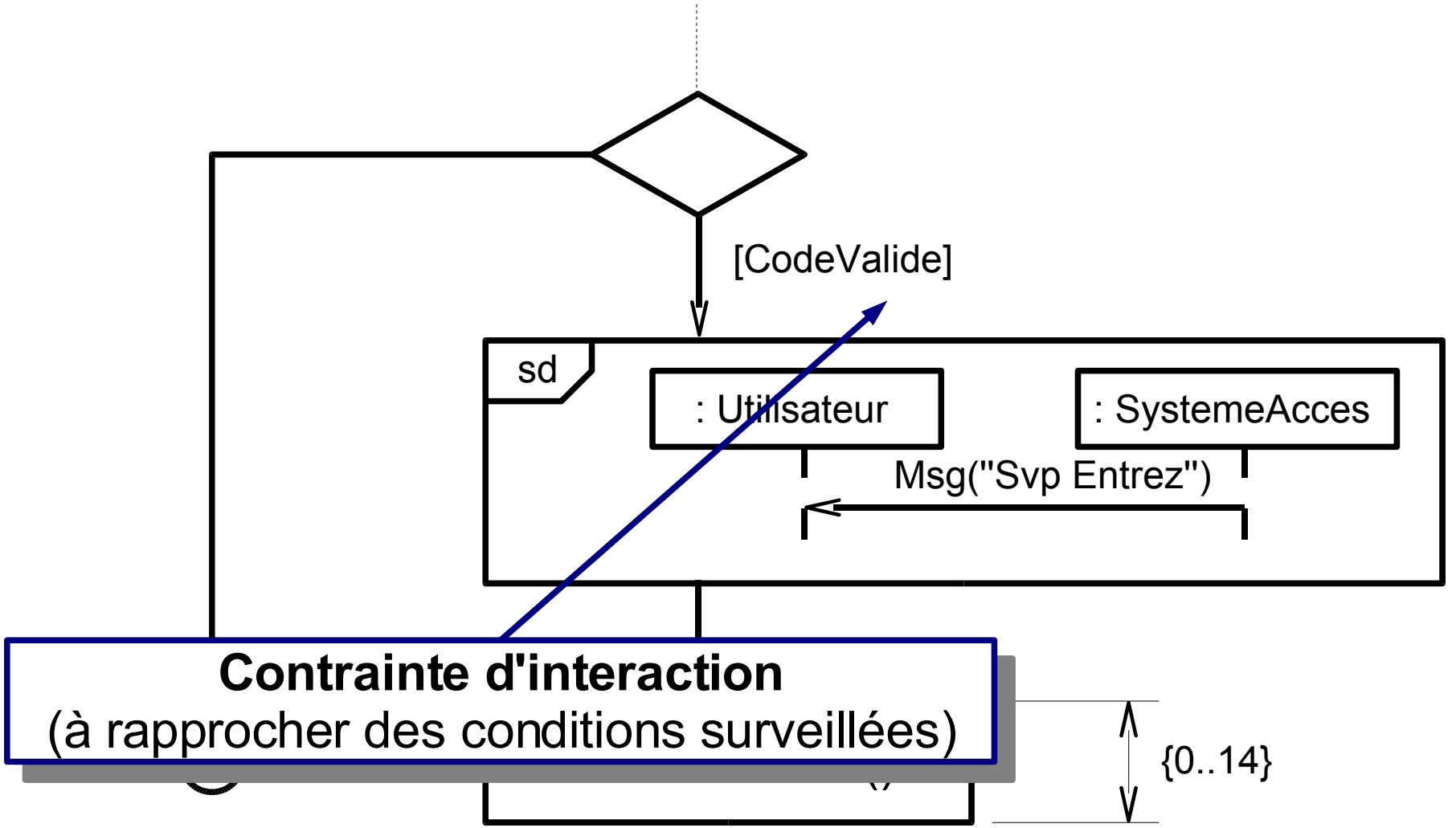
Exemple de diagramme d'interaction globale



Exemple de diagramme d'interaction globale



Exemple de diagramme d'interaction globale



Plan

- Introduction générale,
- Meta-modèle UML,
- Notion de vues d'un système,
- Diagrammes UML,
- **A quelle étape de la création du SI utiliser les diagrammes UML ?**
- Synthèse et conclusion.

Conception et UML

- Étude des diagrammes :
  - Une étude exhaustive des notations,
  - N'explique pas quand les appliquer,
  - Ni quels sont les diagrammes les plus importants,

- UML est simplement un outil pour aider à la conception :
  - Seul, il ne sert à rien,
  - Est englobé dans une méthodologie de conception,

Méthodologie de conception objet

- Plusieurs méthodologies de conception exploitent l'outil UML :
  - Le Rational Unified Process (RUP),
  - La méthode Shlaer-Mellor,
  - CRC (Class, Responsabilities and Collaborators),
  - Extreme Programming (XP),

Méthodologies

- RUP :
  - Fournit une planification exhaustive des activités et artéfacts nécessaires au processus de développement,
  - Insiste sur l'aspect incrémental et itératif du développement,
  - Adaptée aux gros projets,
- Shlaer-Mellor
  - Insiste sur l'intégration et la simulation pour valider les produits pendant le processus de développement,
  - Adaptée aux projets temps réel,

Méthodologies

- CRC :
  - Basée sur la simplicité et sur la détermination des rôles pour mettre en évidence les classes nécessaires à la conception du système,
  - Utile pour aider les programmeurs à apprendre la technologie orientée objet,
- XP :
  - Se concentre sur un environnement de développement idéal,
  - Mets l'accent sur le test continu et la communication,
  - Adapté pour les petits projets évolutifs.

Méthodologies

- Chaque méthodologie a ses avantages et ses inconvénients :
  - Choisir la méthodologie adaptée au projet mené.
- En savoir plus sur les méthodologies de développement orientées objets :
  - http://www.cetus-links.org
    - Lien OOAD methodologies
    - Liste assez exhaustive de ressources sur le sujet,
    - Environ une cinquantaine de méthodologies répertoriées.

Étapes importantes de conception

- Itérations comportant 5 étapes :
  - Étape 1 : Analyse des besoins
    - Diagramme des cas d'utilisation,
  - Étape 2 : Détermination des interfaces,
    - Diagramme de collaborations/communications,
  - Étape 3 : Élaboration de la vue statique,
    - Diagramme de classes,
  - Étape 4 : Définition des interactions,
    - Diagramme de séquences,
  - Étape 5 : Regroupement des fonctionnalités,
    - Diagramme de paquetages et de composants,

Étape 1 : Analyse des besoins

- Phase où techniciens et non techniciens se réunissent pour définir les fonctionnalités et les objectifs du système.
- En général : création de 10 à 12 cas d'utilisation de haut niveau,
- Puis subdivision :
  - L'équipe de test prépare les tests à l'aide des scénarios de cas d'utilisation,
  - L'équipe de développement re-subdivise les cas d'utilisation en cas d'utilisation bas-niveau.

Étape 2 : détermination des interfaces

- A partir des cas d'utilisation, le diagramme des collaborations peut être tracé,
- Il permet de trouver le comportement des divers objets au travers des cas d'utilisation,
- Clarification du fonctionnement :
  - Transition d'un modèle haut-niveau aux objets,
  - Définition du cycle de vie des objets rattachés à chaque cas d'utilisation,
  - Permet de déterminer quelles invocations de méthodes créent des collaborations,

Étape 3: Élaboration de la vue statique

- Le diagramme de collaboration donne les interfaces des objets et donc des classes,
- Les relations statiques entre les objets doivent être déterminées :
  - Types d'objets employés,
  - Interfaces spéciales à implémenter,
  - Détermination des dépendances,
  - Détermination des compositions et des agrégations,
  - Attributs et opérations à créer pour chaque classe,

Etape 4 : Définition des interactions

- Les diagrammes de séquence vont aider à structurer les appels de méthodes,
- Permet d'approfondir les séquences de collaboration,
- Permet d'analyser la conception des classes,
- Fournit un fil directeur clair pour l'implémentation des classes.

Étape 5 : Regroupement des fonctionnalités

- Les diagrammes de paquetages et de composants vont aider à définir les relations de dépendances entre objets à un plus haut niveau que celui des classes,
- Permet de faire comprendre comment l'itération de développement va s'intégrer aux itérations précédentes.

Plan

- Introduction générale,
- Meta-modèle UML,
- Notion de vues d'un système,
- Diagrammes UML,
- A quelle étape de la création du SI utiliser les diagrammes UML ?
- **Synthèse et conclusion.**

Synthèse et conclusion

- Programmation objet :
  - Programmer en langage objet n'est pas concevoir objet :
    - Seule une analyse objet conduit à une solution objet (qui respecte les concepts de l'approche objet),
    - Le langage de programmation est un moyen d'implémentation, il ne garantit pas le respect des concepts objets,
  - UML ne reste qu'un support de communication :
    - Son utilisation seule ne garantit pas le respect des concepts objets : il faut s'en servir à bon escient.
- **UML et le langage ne restent que des outils !**

Synthèse et conclusion

- Utilisation d'UML :
  - Multiplier les vues sur les modèles :
    - Un diagramme n'est qu'une vue partielle mais néanmoins précise d'un modèle,
    - Utiliser les vues complémentaires (statiques/dynamiques),
  - Rester simple :
    - Utiliser plusieurs niveaux d'abstraction,
    - Ne pas trop surcharger les diagrammes,
    - Commenter les diagrammes (notes, textes, ...)
  - Utiliser des outils appropriés pour réaliser les modèles !

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 :
  - EclipseUML : http://www.eclipseuml.com
  - Poseidon : http://www.gentleware.com/
  - Rational rose : http://www.rational.com/
  - Together : http://www.borland.com/