- UML:
  - Diagrammes d'objets

UML 2.0

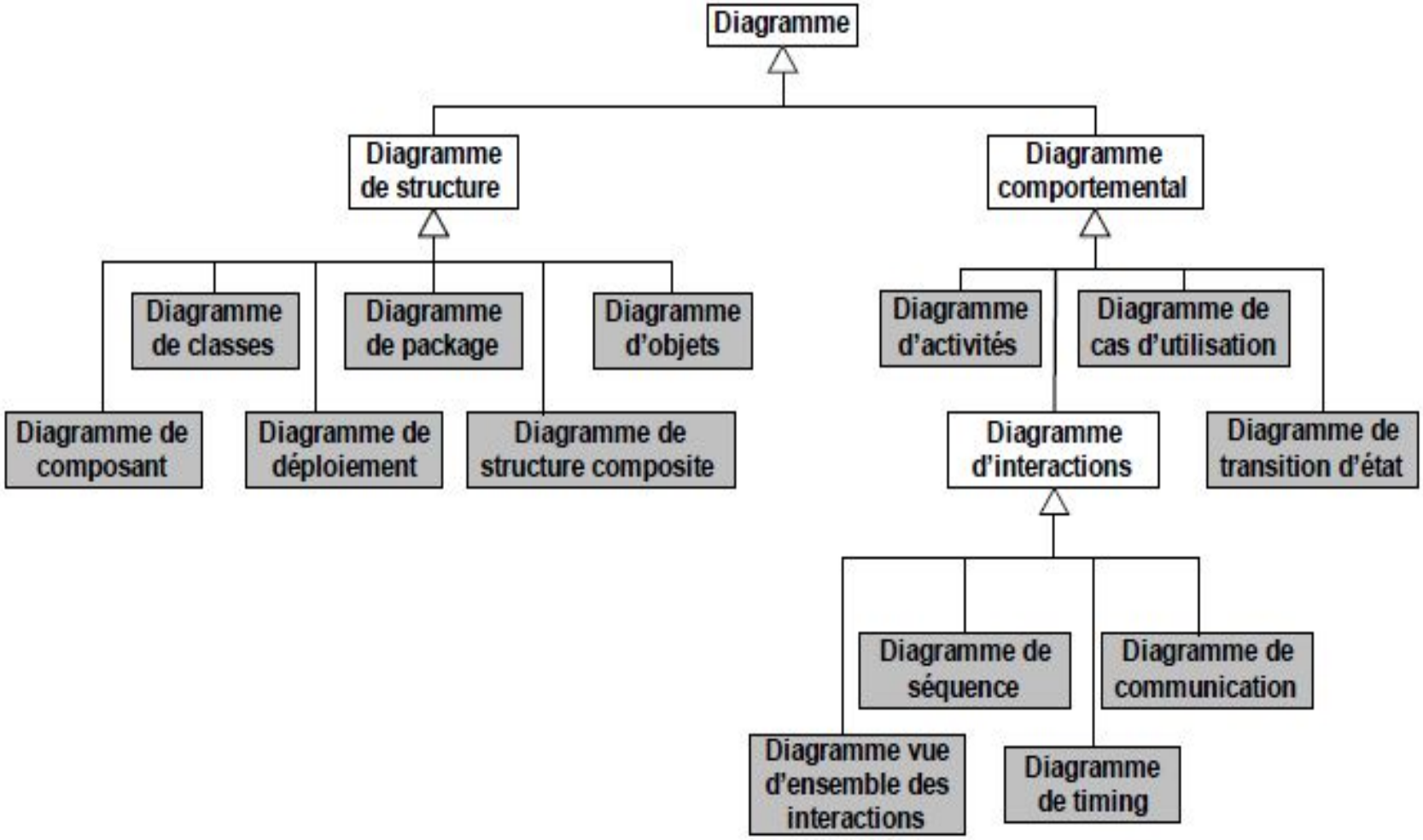


Diagramme
Diagramme de structure
Diagramme comportemental
Diagramme de classes
Diagramme de package
Diagramme d'objets
Diagramme de composant
Diagramme de déploiement
Diagramme de structure composite
Diagramme d'activités
Diagramme de cas d'utilisation
Diagramme d'interactions
Diagramme de transition d'état
Diagramme de séquence
Diagramme de communication
Diagramme vue d'ensemble des interactions
Diagramme de timing

RAPPEL DIAGRAMMES

Diagramme de structures :

- Diagramme de classes
  Class diagram
- **Diagramme d'objets**
  Object diagram
- Diagramme de composants
  Component diagram
- Diagramme de déploiement
  Deployment diagram
- Diagramme de paquetages
  Package diagram
- Diagramme de structure composite
  Composite Structure diagram

Diagrammes Comportementaux :

- Diagramme de cas d'utilisation
  Use case diagram
- Diagramme de communication (collaboration UML1)
  Communication diagram
- Diagramme de séquence
  Sequence diagram
- Diagramme d'états-transition
  State Machine diagram
- Diagramme d'activités
  Activity diagram
- Diagramme global d'interaction
  Interaction Overview diagram
- Diagramme de temps
  Timing diagram

Version 2.X

Diagramme d'objets

- Représente les objets (instances de classes) et les liens (instances de relations) à un instant donné.

Diagramme d'objets

Objectifs:

- Illustrer par un exemple concret un diagramme de classes.

- Faciliter la validation d'un diagramme de classes complexe en présentant une ou plusieurs instanciation de celui-ci.

- Visualiser un instantané de l'état d'un système.

- Exprimer une exception en modélisant des cas particuliers.

Exemple: Gestion des commandes client diagramme de classes et d'objets

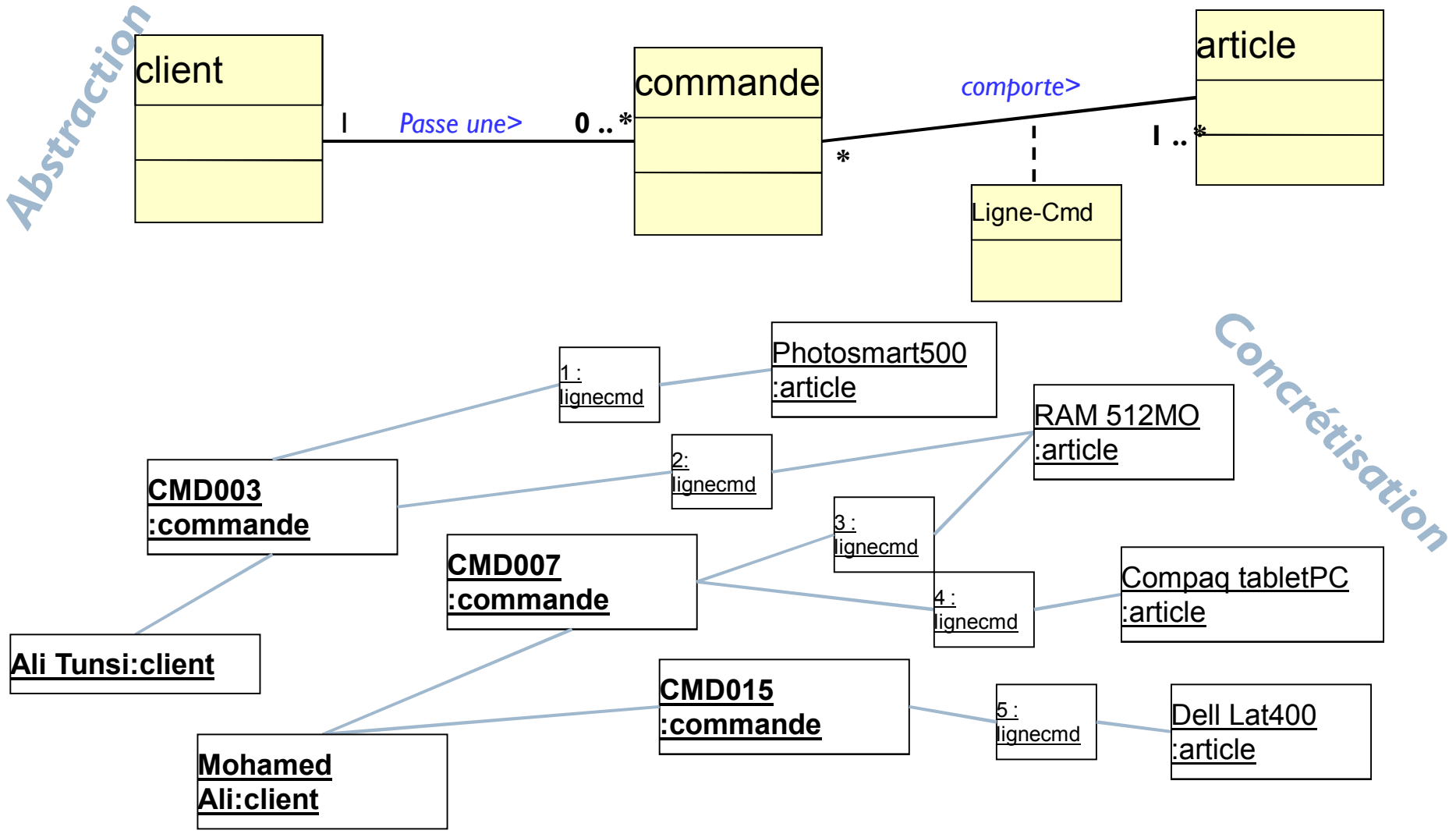
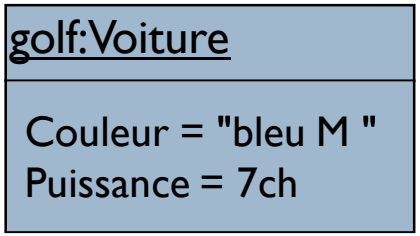


Diagramme d'objets

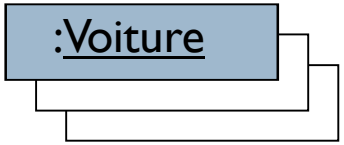
Instance anonyme de la classe voiture

Instance nommée de la classe voiture

Instance nommée d'une classe anonyme



Spécification des attributs



Collection d'instance (tableau)

Diagramme d'objets

- Le nom d'un objet est souligné. Plusieurs représentations sont possibles:
  - Nom de l'objet: Classe
  - Nom de l'objet
  - :Classe

- La représentation des objets peut contenir les attributs significatifs.

Les caractéristiques fondamentales des objets

- L'identité
- L'état
- Le comportement

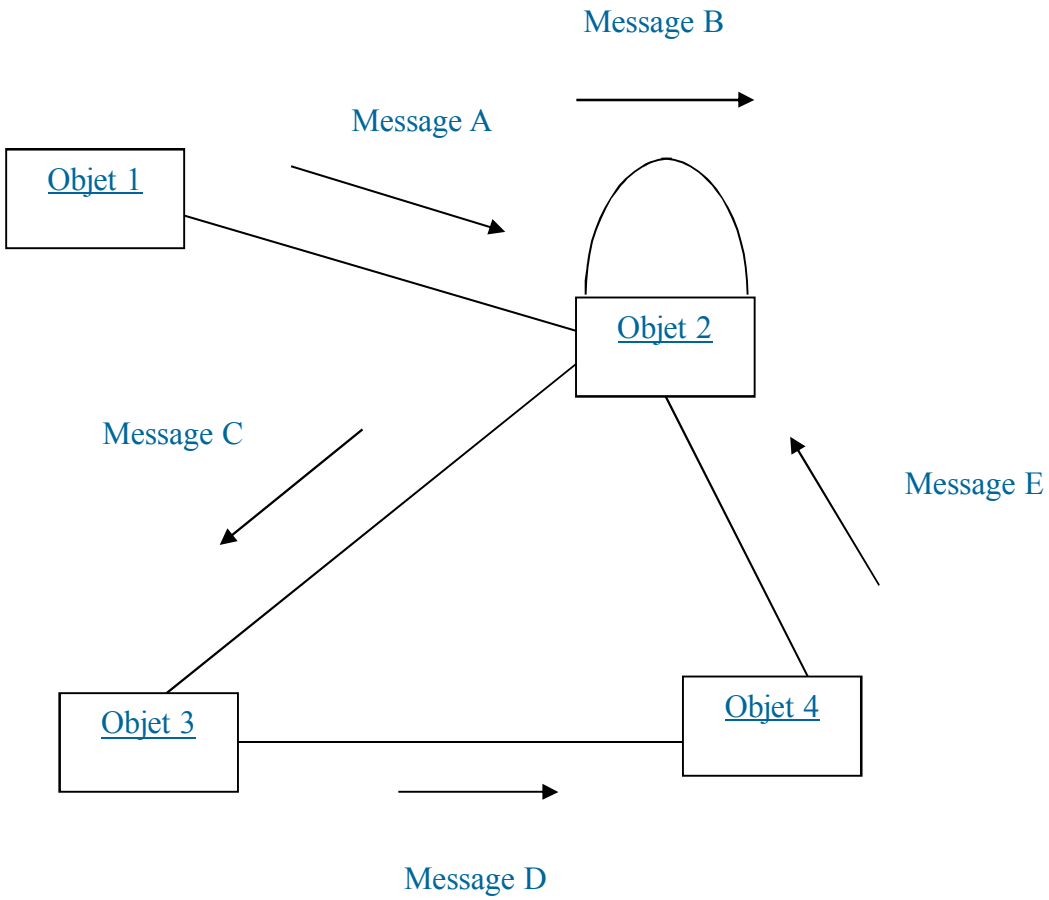
L'identité d'un objet

- Chaque objet a sa propre **identité** ;
- Deux objets sont distincts même si toutes les valeurs de leurs attributs sont identiques.
- Dans le modèle objet, l'identité est implicite. Chaque objet est une instance d'une classe qui reçoit , lors de sa création une identité propre et unique.
- Dans le modèle relationnel, l'identité est donnée par une clé primaire qui est unique pour chaque ligne de la table.

État & Comportement

- L'état d'un objet correspond aux valeurs de tous ses attributs à un instant donné.
- Les propriétés sont définies dans la classe d'appartenance de l'objet.
- Le comportement d'un objet est caractérisé par l'ensemble des opérations qu'il peut exécuter en réaction aux messages provenant des autres objets.
- Les opérations sont définies dans la classe d'appartenance de l'objet.

Communication entre objets



Une application, en orienté objet, est une société d'objets collaborant.

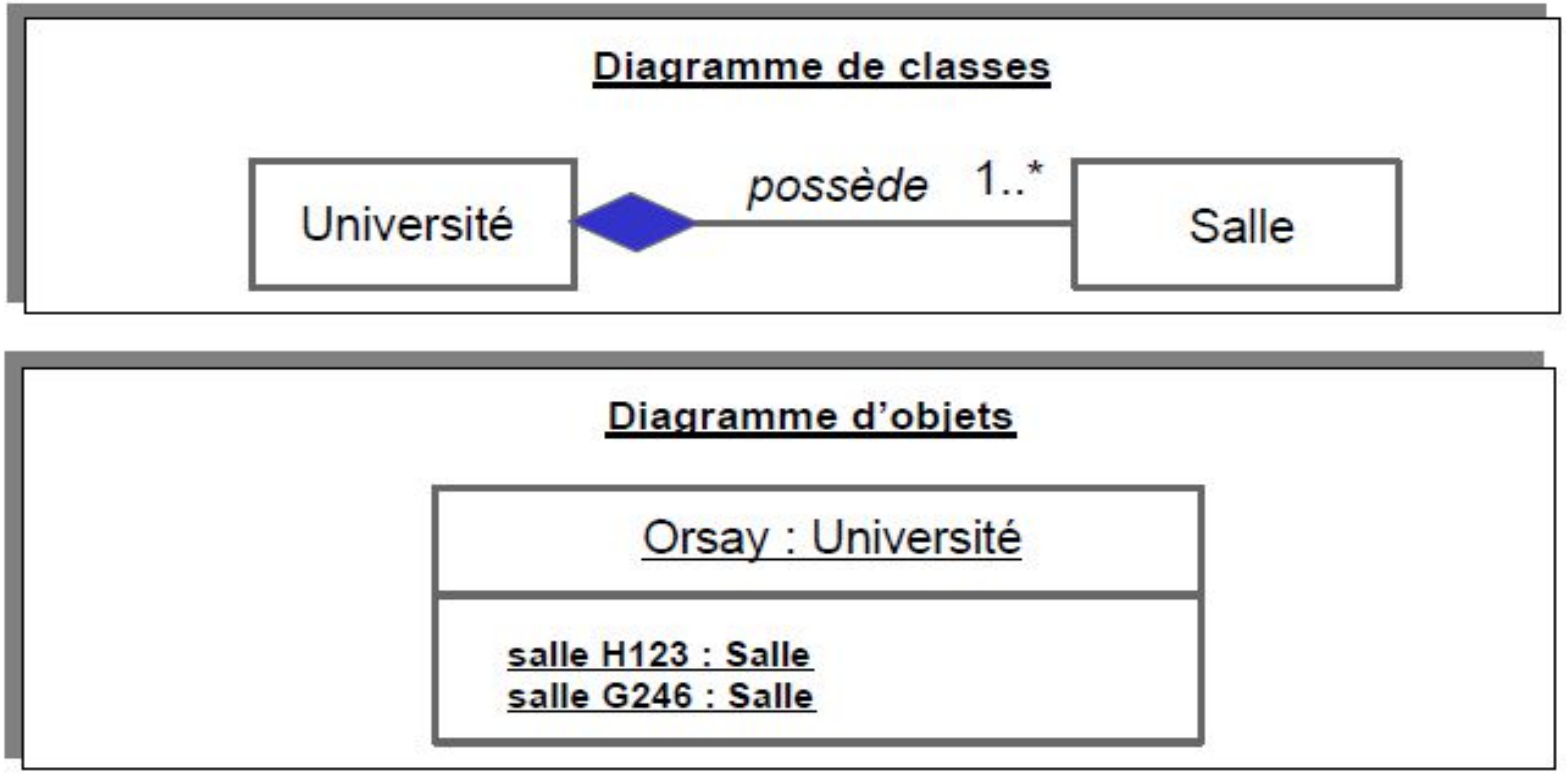
L'unité de communication entre les objets est le message

Donc, l'achèvement d'une tâche par une application repose sur la communication entre les objets qui la composent.

État & Comportement

- Cet objet est caractérisé par la liste de ses attributs et son état est représenté par les valeurs de ses attributs :
  - n° employé : 1245,
  - nom : Ali,
  - qualification : ingénieur,
  - lieu de travail : site N.
- Son comportement est caractérisé par les opérations qu'il peut exécuter.
  - entrer dans l'organisme,
  - changer de qualification,
  - changer de lieu de travail,
  - sortir de l'organisme.

Objet composite



Chapitre 4: Diagrammes dynamiques

UML 2.0

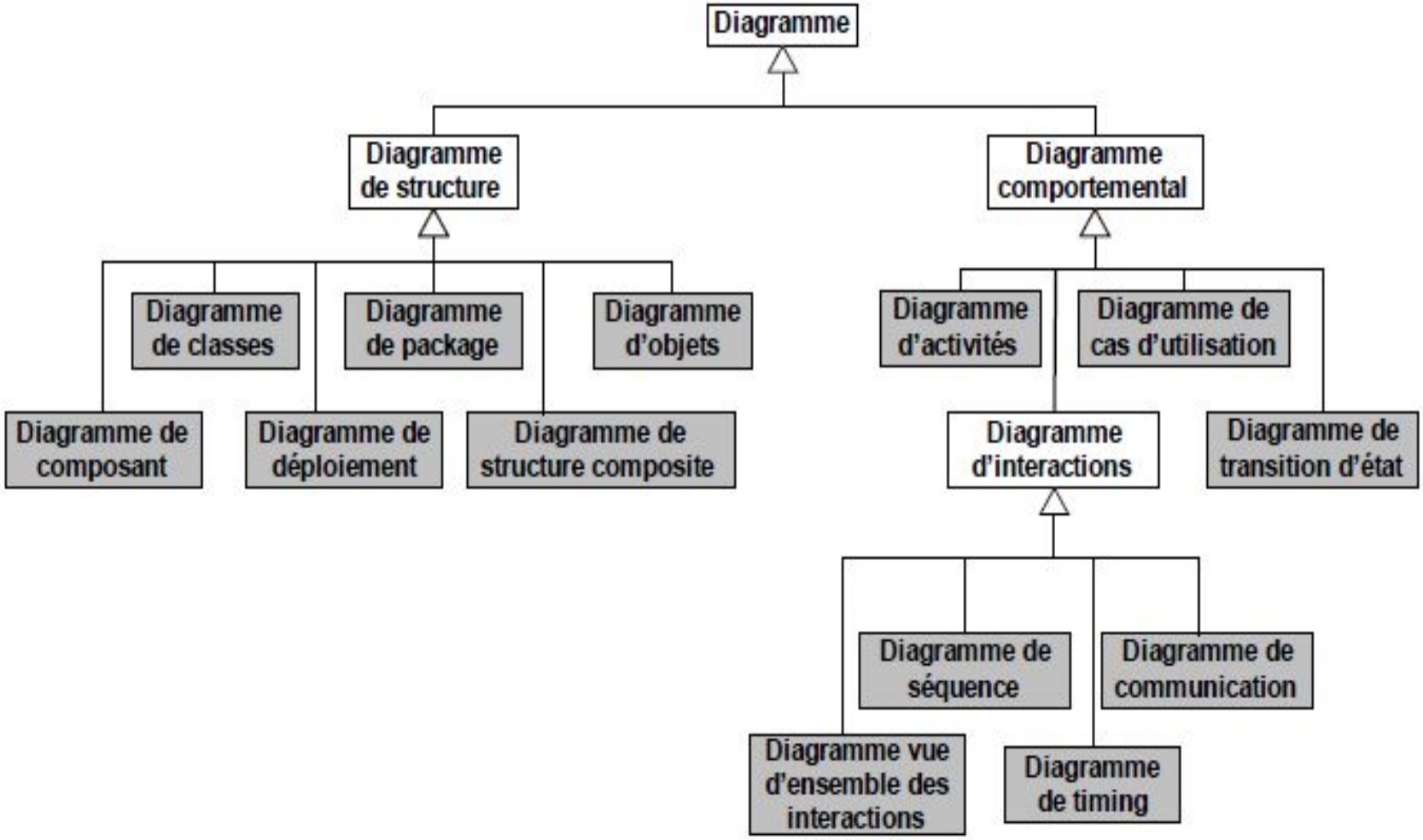


Diagramme
Diagramme de structure
Diagramme comportemental
Diagramme de classes
Diagramme de package
Diagramme d'objets
Diagramme de composant
Diagramme de déploiement
Diagramme de structure composite
Diagramme d'activités
Diagramme de cas d'utilisation
Diagramme d'interactions
Diagramme de transition d'état
Diagramme de séquence
Diagramme de communication
Diagramme vue d'ensemble des interactions
Diagramme de timing

Remarque

- *Il n'y a pas un diagramme de classes mais plusieurs, chacun étant une évolution du précédent et une ébauche du suivant.*
- *Ce diagramme de classes est cohérent avec les autres diagrammes (de séquences, de collaborations...). Il est en permanence mis à jour diagramme après diagramme.*

Diagrammes d'interaction

Ces diagrammes se concentrent sur une collection d'objets qui coopèrent :

- Diagramme de communication : organisation structurelle des objets qui échangent des messages
- Diagramme de séquence : chronologie de l'envoi des messages.

Ces diagrammes font le lien entre les diagrammes des cas d'utilisations et le diagramme des classes.

Se concentrer sur des sous-ensembles d'objets

Modélisation UML: Diagramme de séquence

Diagramme de séquence

- En général, un diagramme de séquence capture le comportement d'un cas d'utilisation.

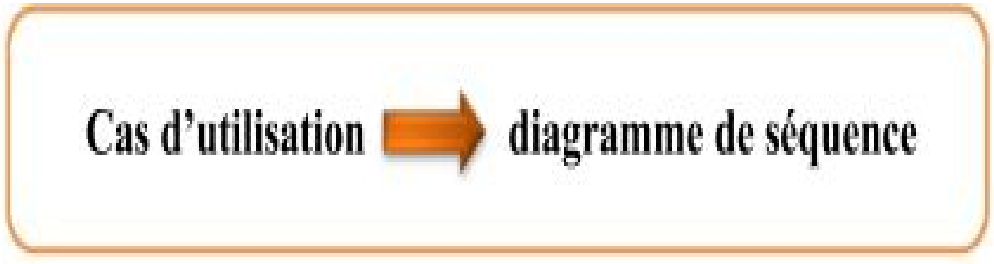


Diagramme de séquence

- Un diagramme de séquences est un moyen de capturer le comportement des objets et acteurs du système selon un point de vue temporel.

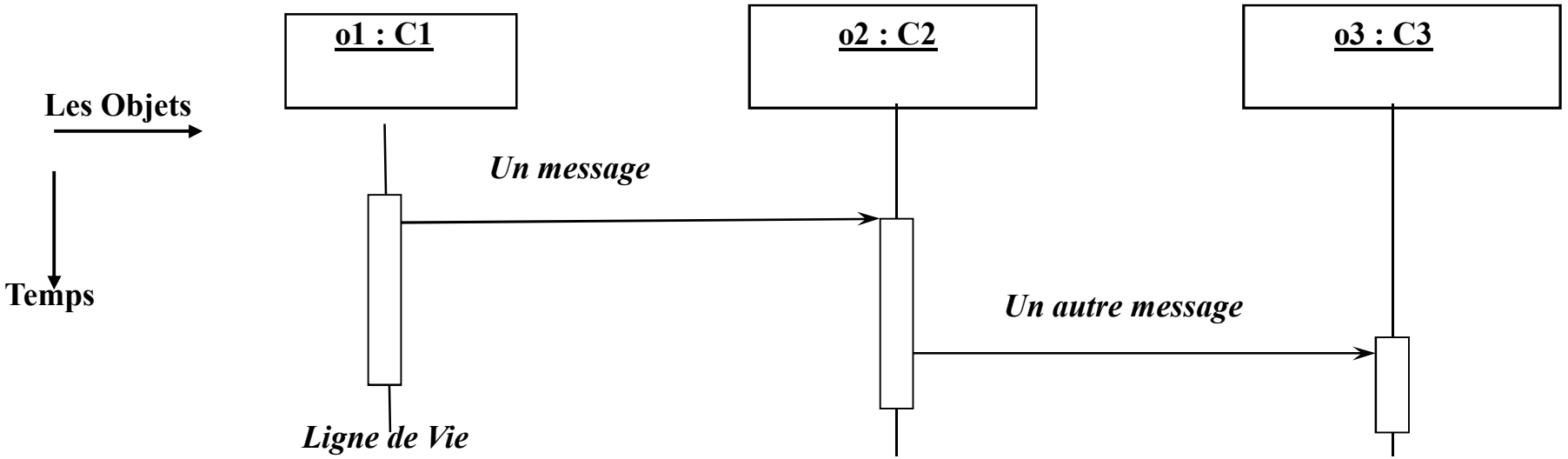
- Deux types:

  - Diagramme de séquence système (phase de capture des besoins)
    - Le système est une boîte noire
    - Analyse d'un cas d'utilisation

  - Diagramme de séquence objet (analyse et de conception)
    - Le système est une boîte blanche
    - Analyse d'un diagramme de classe

Diagrammes de Séquences Objet

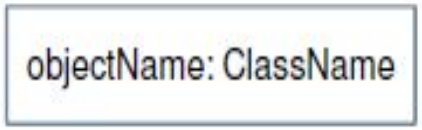
- Définissent les interactions entre Objets (et acteurs) selon un point de vue temporel,
- L'unité de communication entre objets s'appelle message.



Représentation des objets

- Deux façons de désigner l'objet interagissant

Instance nommée de la classe



Instance anonyme de la classe

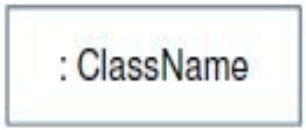


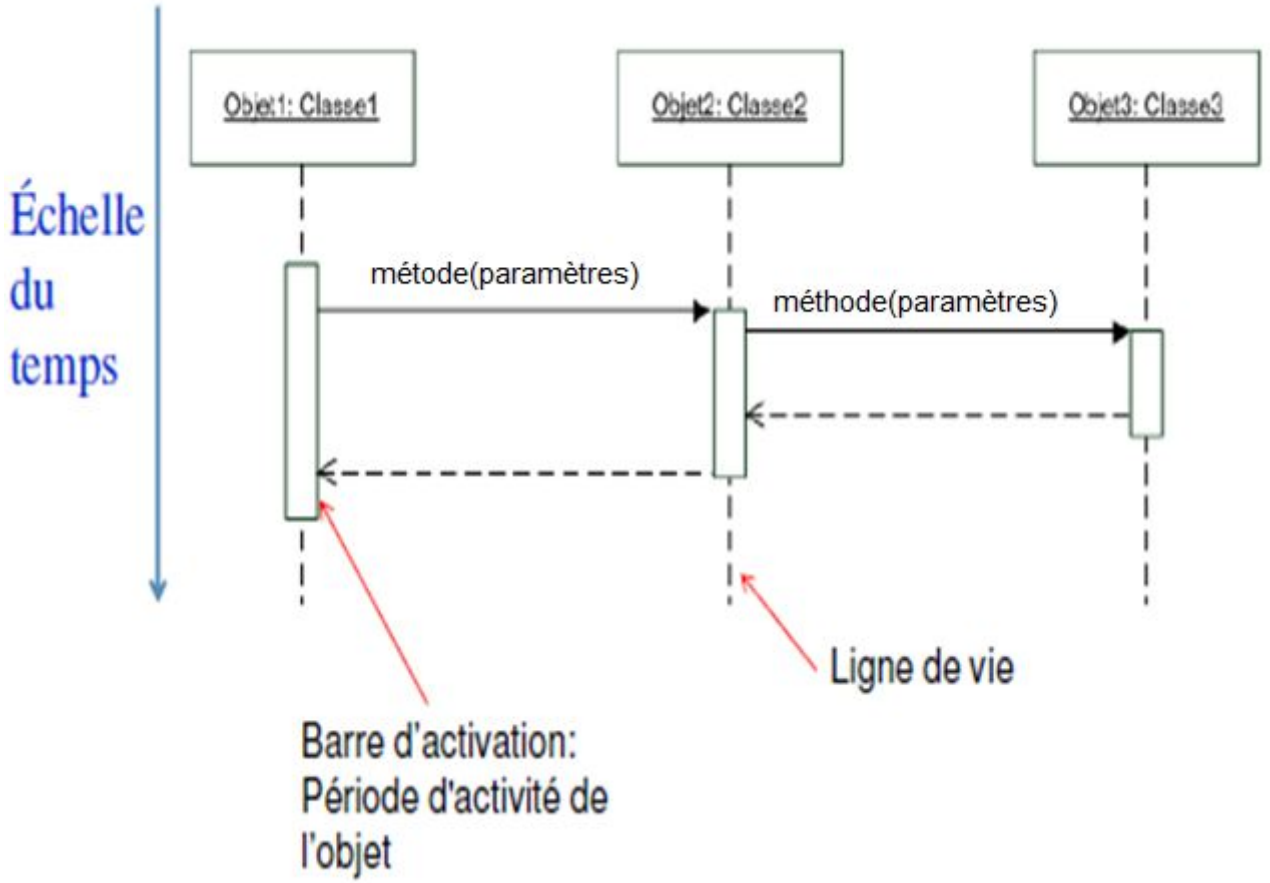
Diagramme de Séquence

- L'ordre d'envoi d'un message est déterminé par sa position sur l'axe vertical du diagramme.
  - le temps s'écoule "de haut en bas" de cet axe
- La disposition des objets sur l'axe horizontal n'a pas de conséquence pour la sémantique du diagramme.
- Les diagrammes de séquences et les diagrammes d'état-transitions sont les vues dynamiques les plus importantes d'UML.

MESSAGE

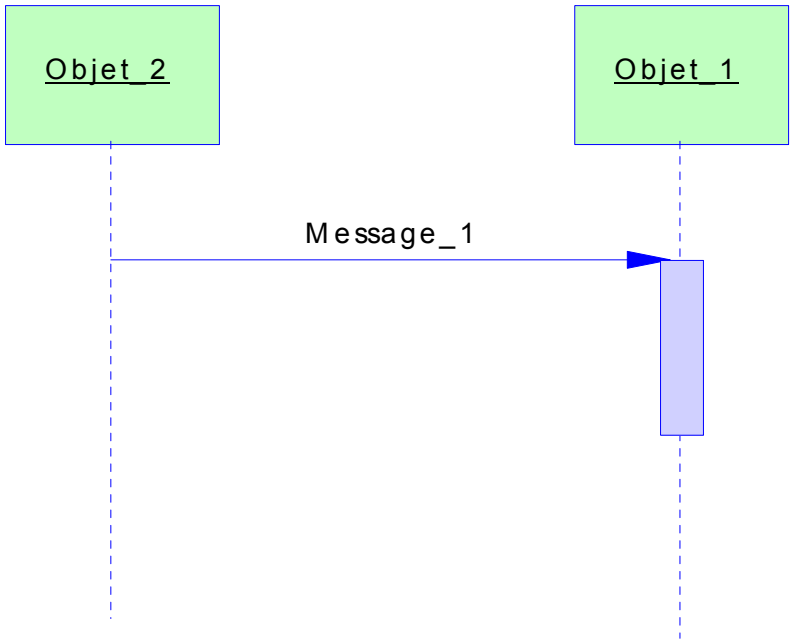
- Les messages sont représentés par des flèches placées le long des liens qui unissent les objets.
- Lors de la réception d'un message, un objet devient actif, marquant le traitement du message.
- Les acteurs peuvent également communiquer avec des objets, ainsi ils sont énumérés en colonne.
- Une ligne de vie représente un rôle (objet ou acteur) dans l'interaction.
- La ligne de vie représente l'existence de l'objet à un instant particulier. Commence avec la création de l'objet et se termine avec la destruction de l'objet.

Eléments de base

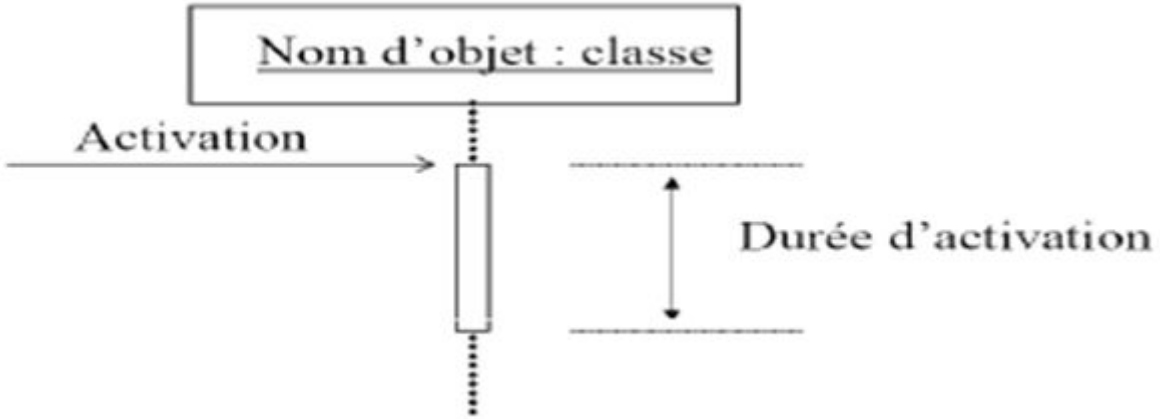


Période d'activité

- Représentée par une bande rectangulaire superposée à la ligne de vie de l'objet.



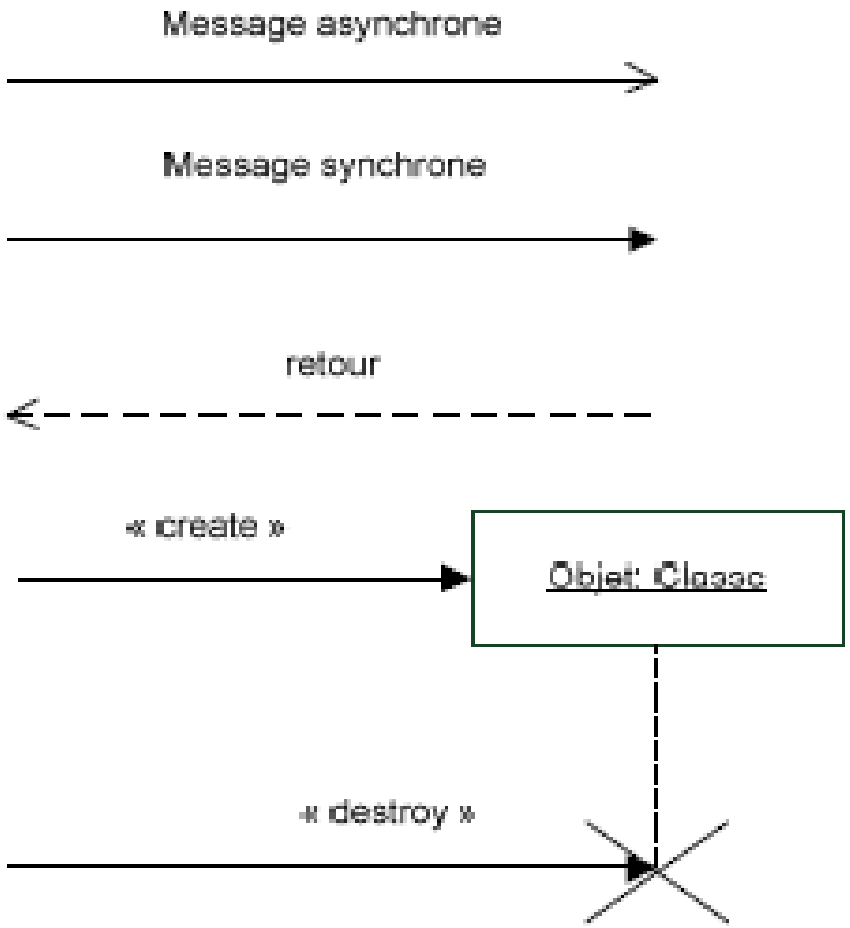
Activation d'un objet



- La période d'activation correspond simplement à la durée pendant laquelle l'objet « travaille » : pendant laquelle le code de cet objet est exécuté.

Synchronisation des messages

Notation



Types de message

- Un message synchrone bloque l'expéditeur jusqu'au retour du destinataire. Le flôt de contrôle passe de l'émetteur au récepteur.
  - Typiquement : appel de méthode
    - Si un objet A invoque une opération d'un objet B, A reste bloqué tant que B n'a pas terminé.

- On peut associer aux messages d'appel de la méthode un message de retour (en pointillés) marquant la reprise du contrôle par l'objet émetteur du message synchrone.

Types de message

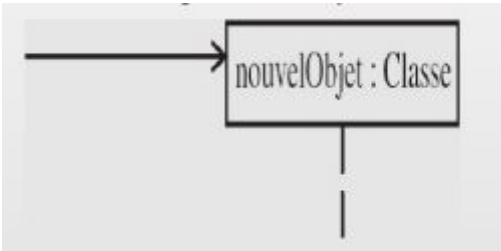
- Un message asynchrone n'interrompt pas l'exécution de l'expéditeur.
- Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.
  - Typiquement : envoi de signal (message porteur d'information)

« Signal »

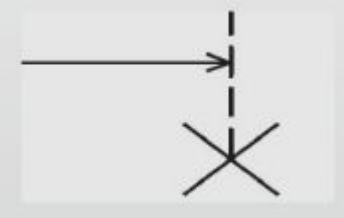
- Un évènement = La réception d'un signal.
- UML propose de décrire les signaux par des classes avec le stéréotype « signal ».

Types de message

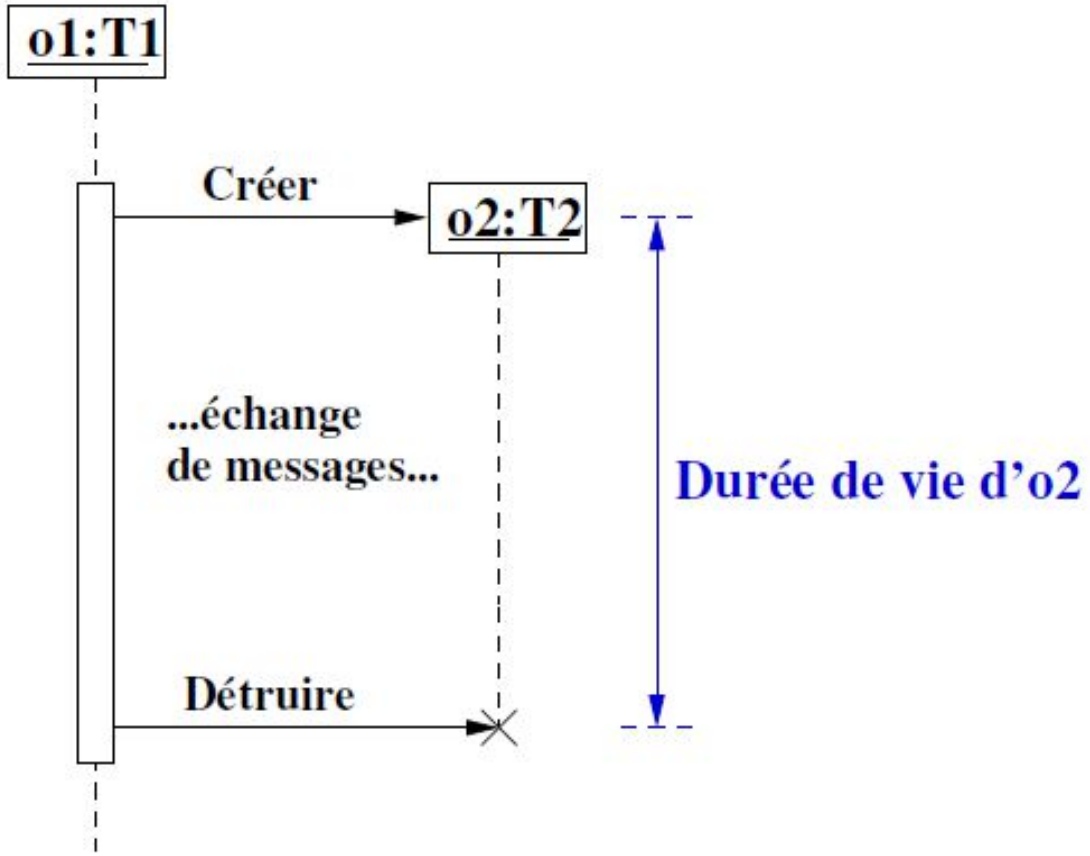
- La création d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie.



- La destruction d'un objet est matérialisée par une croix qui marque la fin de la ligne de vie de l'objet.



Création et destruction d'objets



Exemple

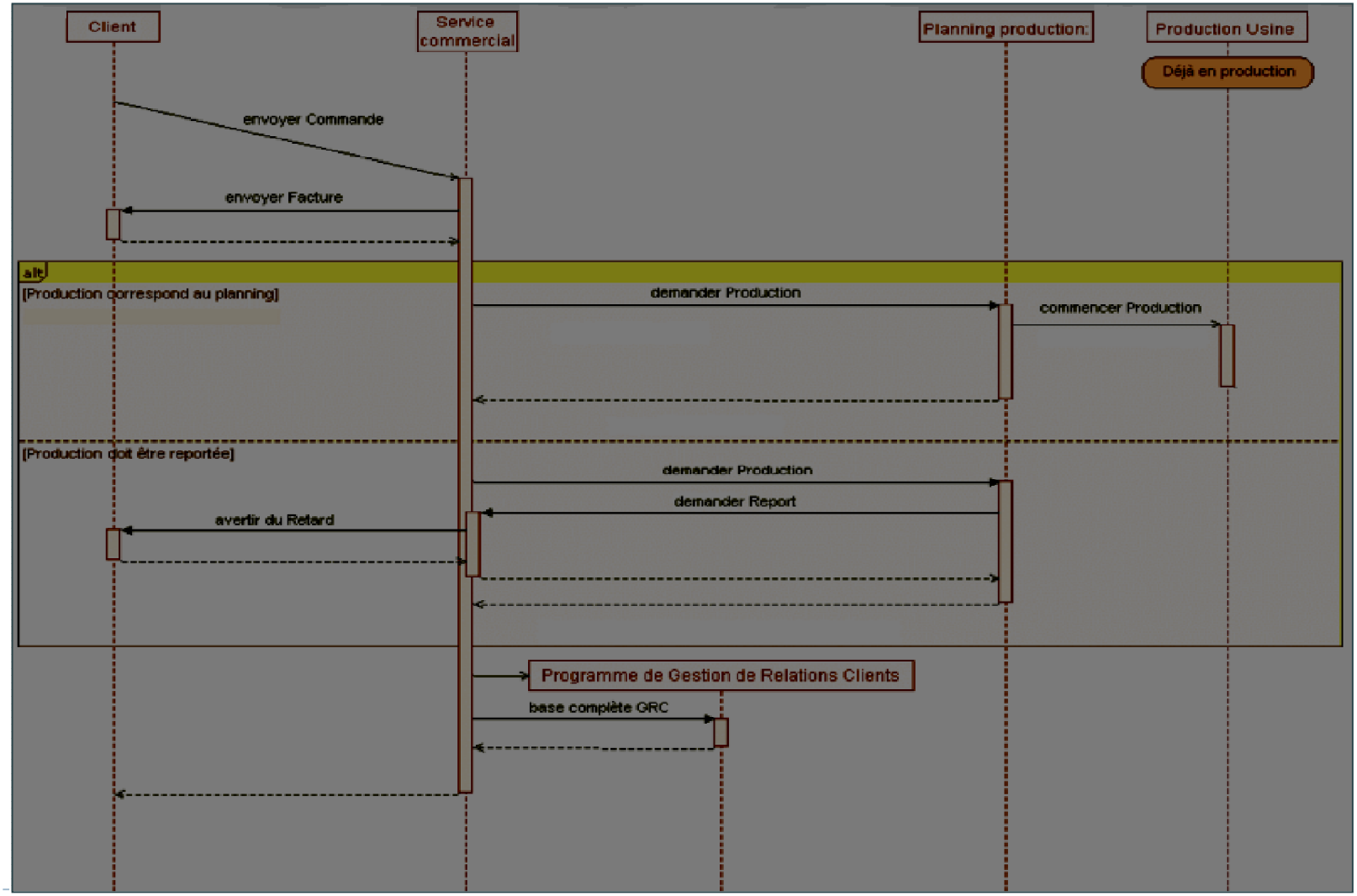
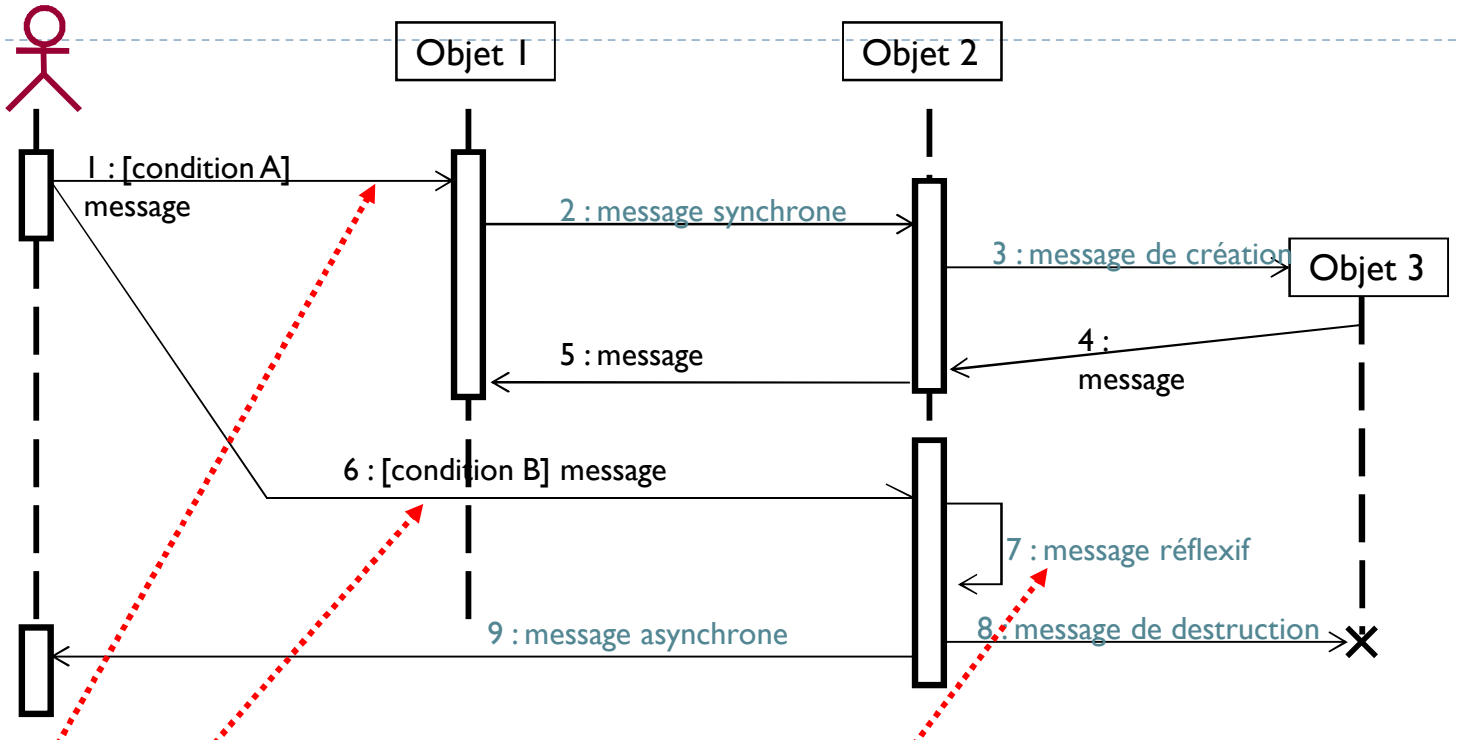

Client
Service commercial
Planning production:
Production Usine
Déjà en production
envoyer Commande
envoyer Facture
alt
[Production correspond au planning]
demander Production
commencer Production
[Production doit être reportée]
demander Production
demander Report
avertir du Retard
Programme de Gestion de Relations Clients
base complète GRC

Diagramme de Séquences (UML1)



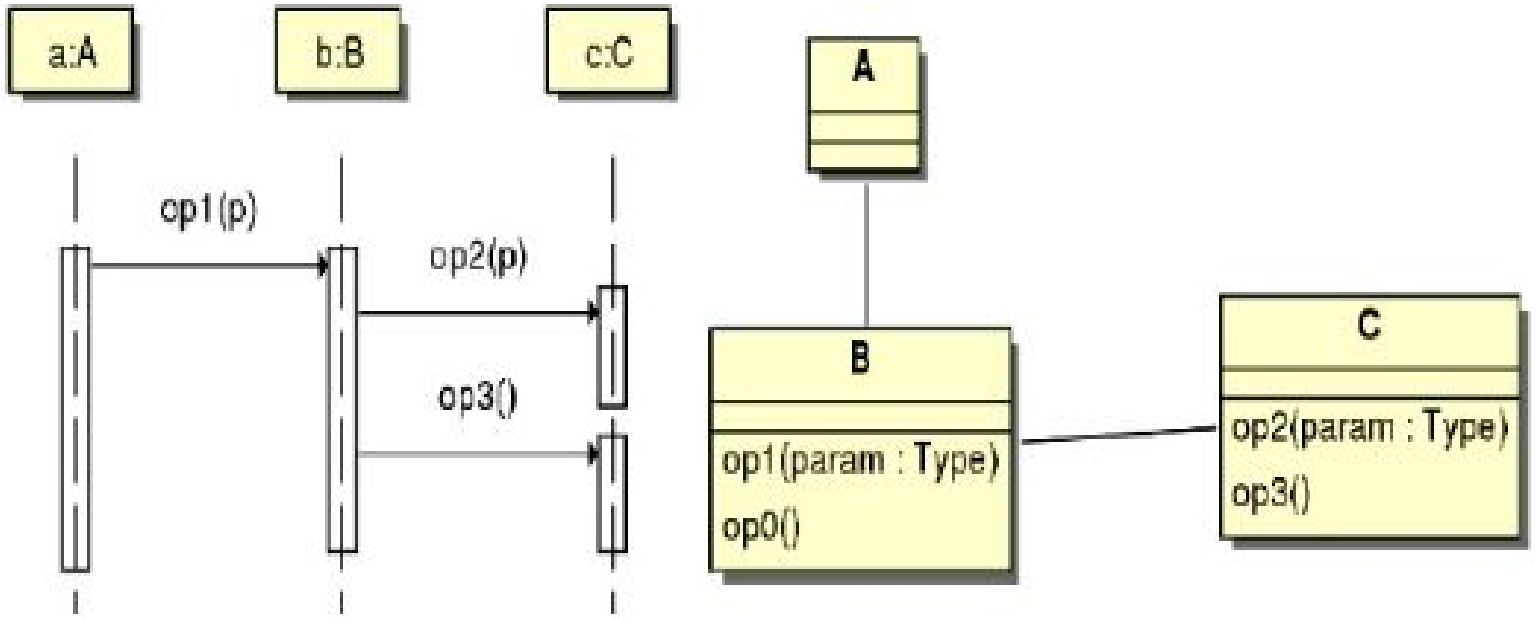
message synchrone: l'émetteur est bloqué et attend que l'appelé ait fini de traiter le message *(message 1)*

message asynchrone: l'émetteur n'est pas bloqué et peut continuer son exécution *(message 6)*

Un message il représente une activité interne à l'objet (qui peut être détaillée dans un diagramme d'activités) *(message 7)*

Messages

Les **messages synchrones** correspondent à des opérations dans le diagramme de classes



Syntaxe d'un message

- [attribut=] <nomSignalouOpération>[(listeArgument,)][: valeur-retour]

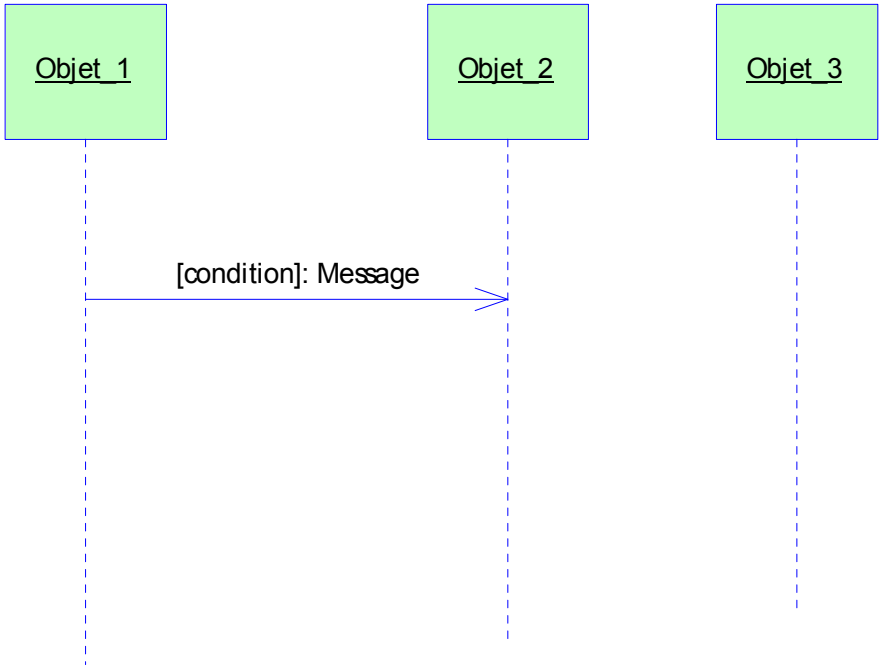
- Attribut: le nom d'un attribut pour stocker la valeur du retour.

- La syntaxe des arguments est la suivante :
  - <nomParametre >=< valeurArgument >
  - Valeur-argument
  - nomParamètre

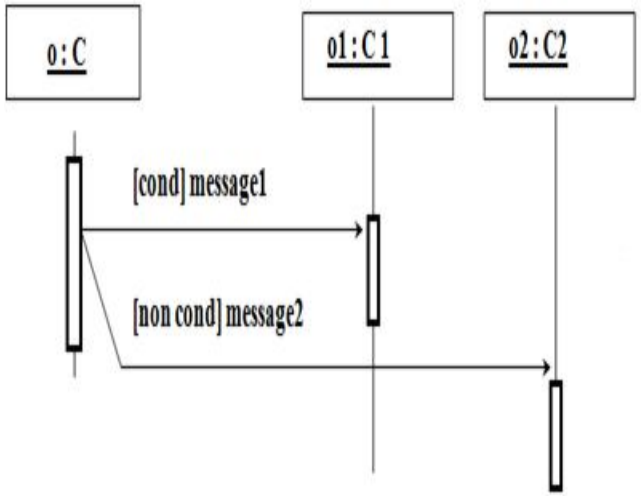
- Exemples :
- appeler("Capitaine Hadock", 54214110)
- afficher(x,y)
- initialiser(x=100)
- Dans le cas de signaux, les arguments correspondent aux attributs du signal.

Traitement alternatif (UML 1)

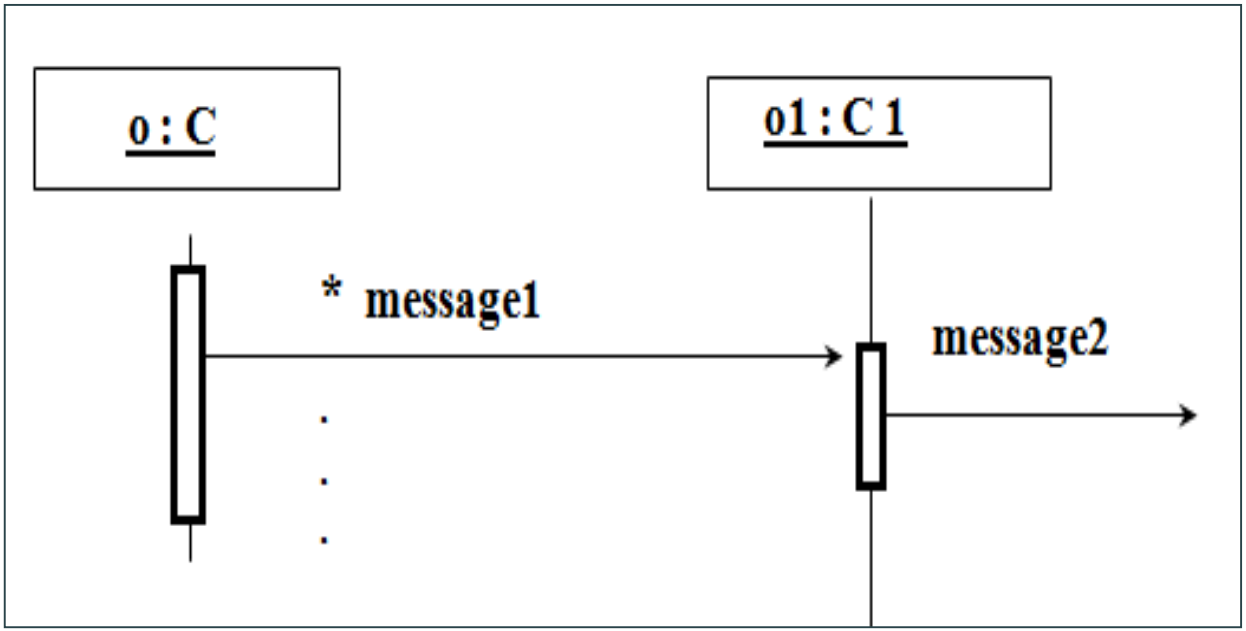
- La condition précédée le message et elle est délimitée par des crochets (Si ..Alors)



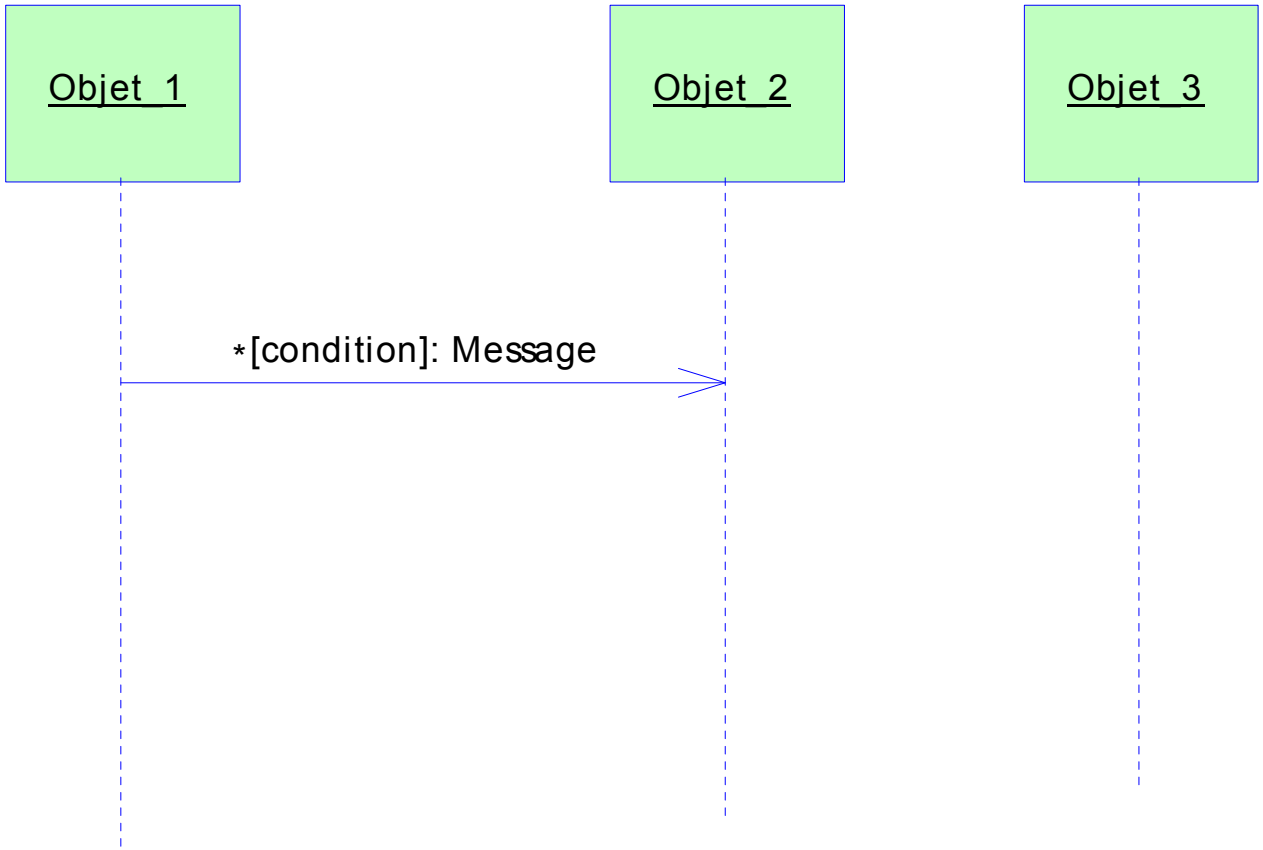
- Si..Alors..Sinon



Itération (UML 1)



Itération (UML 1)

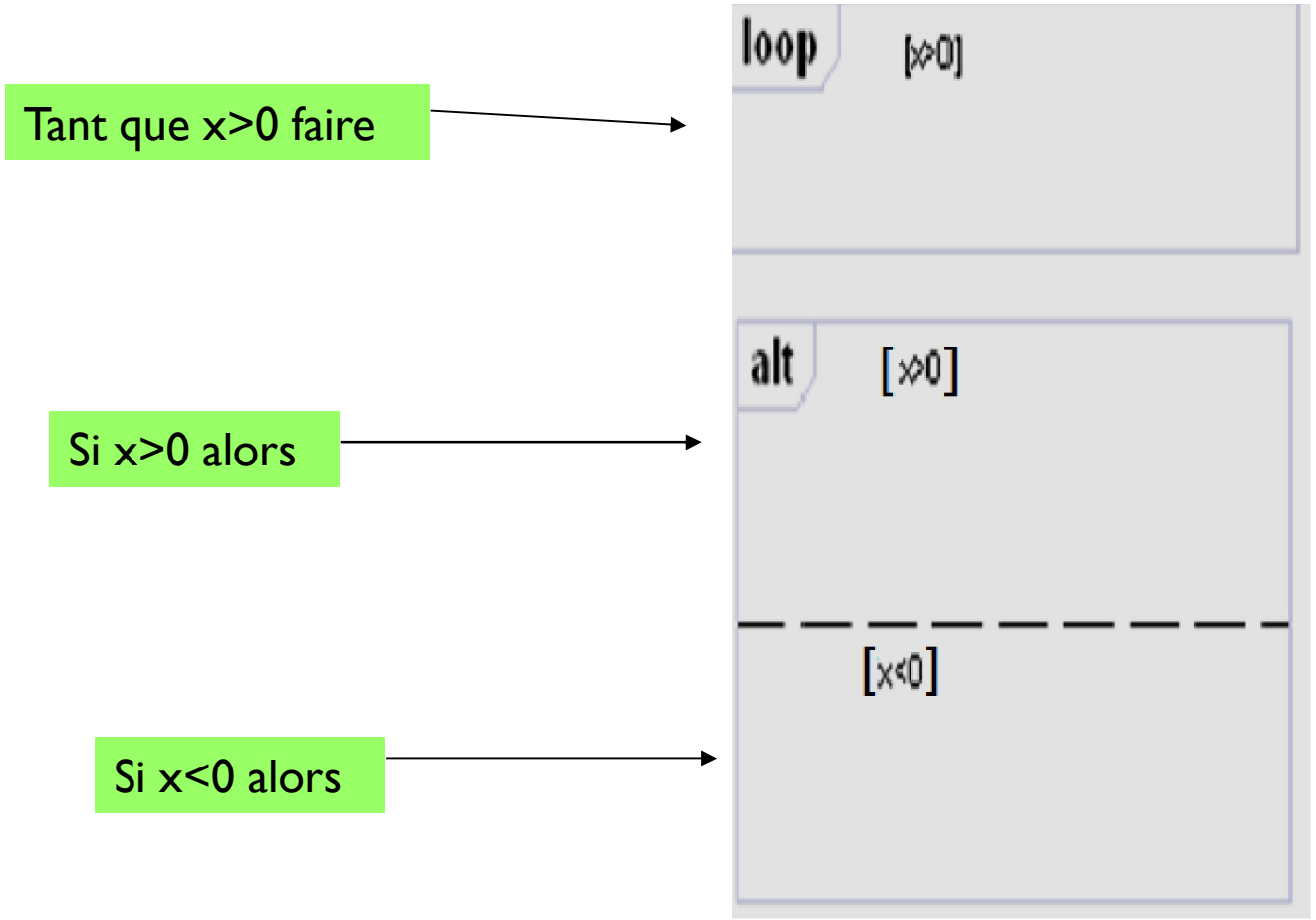


UML 2: Fragments combinés

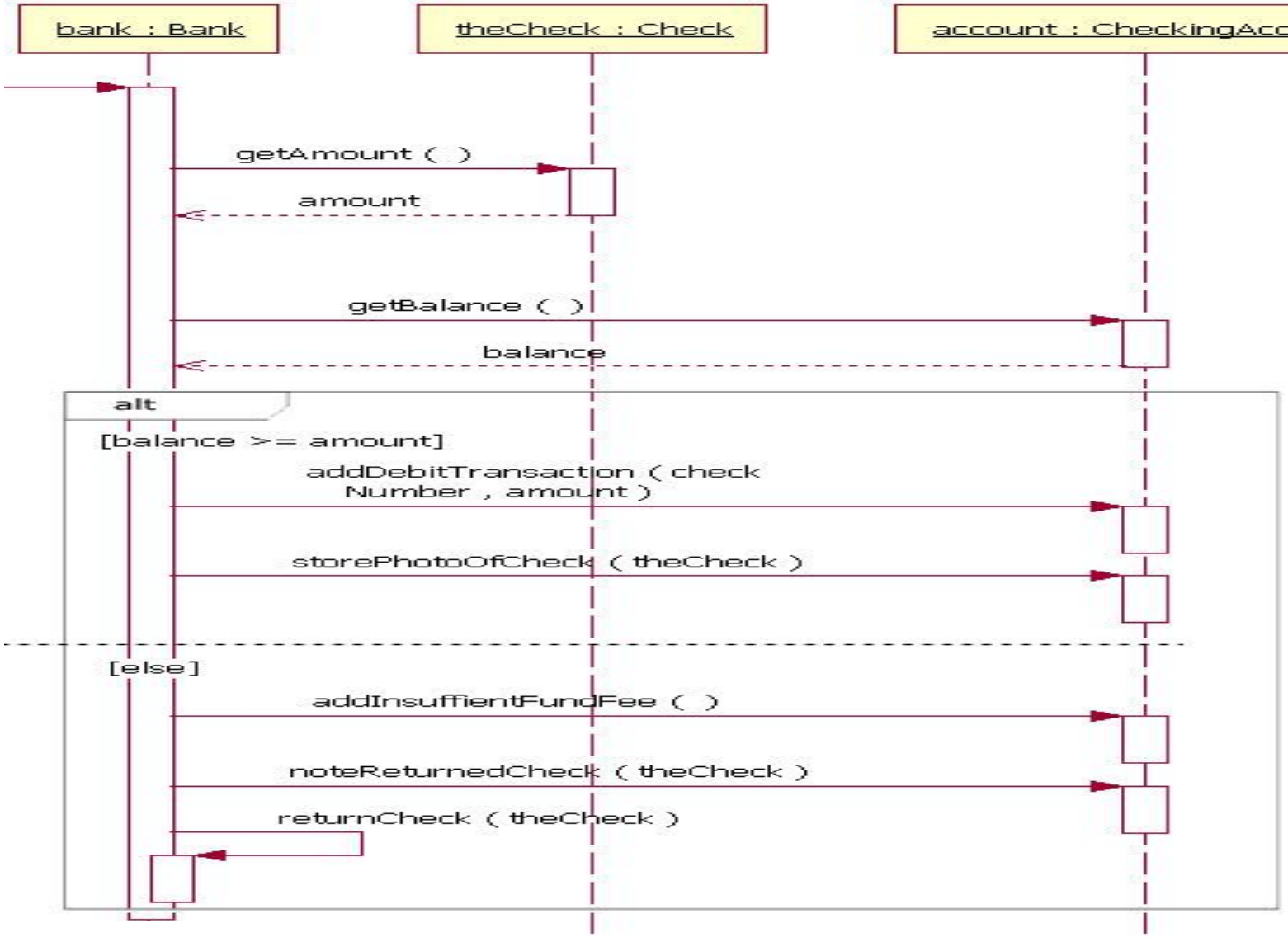
- Représenté par un rectangle dont le coin supérieur gauche contient un pentagone.

- Nous utilisons des fragments combinés pour représenter :
  - Des alternatives,
  - Des options,
  - Des boucles
  - Des références

Fragments: loop et alt



Exemple: Diagramme de séquence du cas d'utilisation « Retrait chèque »



Balance= solde du compte

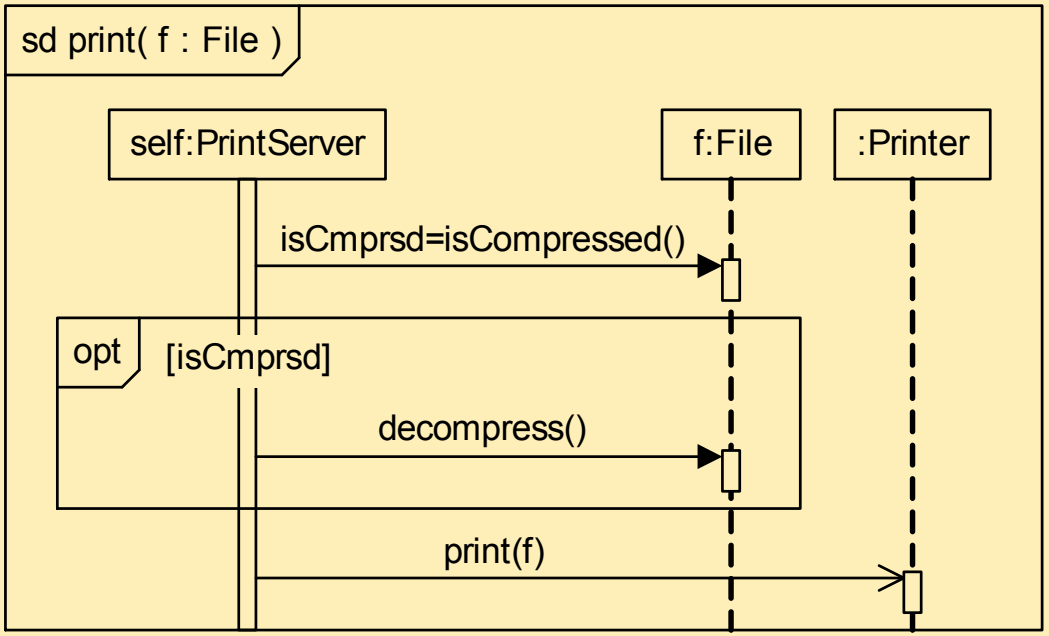
OPÉRATEUR "OPTION"

L'opérateur **"opt"** désigne un fragment combiné optionnel comme son nom l'indique : c'est à dire qu'il représente un comportement qui peut se produire... ou pas. Un fragment optionnel est équivalent à un fragment "alt" qui ne posséderait pas d'opérande else (qui n'aurait qu'une seule branche). Un fragment optionnel est donc une sorte de **SI...ALORS**.

Opérateur « opt »

L'opérateur **"opt"** désigne un fragment combiné optionnel comme son nom l'indique : c'est à dire qu'il représente un comportement qui peut se produire... ou pas. Un fragment optionnel est équivalent à un fragment "alt" qui ne posséderait pas d'opérande else (qui n'aurait qu'une seule branche). Un fragment optionnel est donc une sorte de **SI...ALORS**.

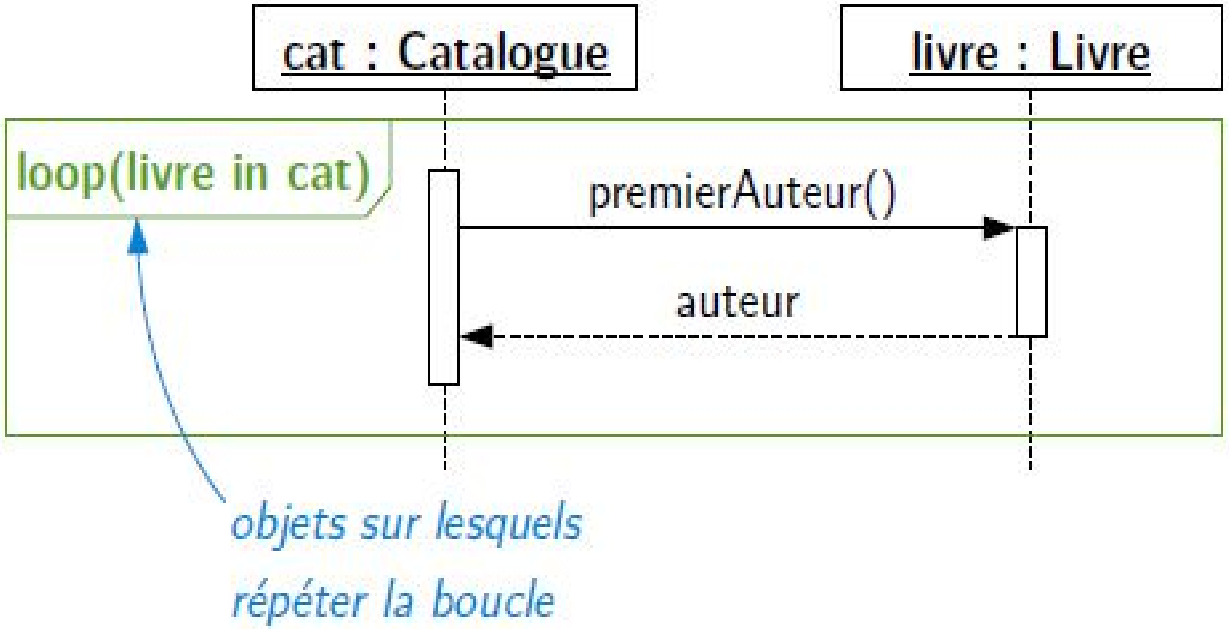
Opérateur opt



Opérateur « Loop »

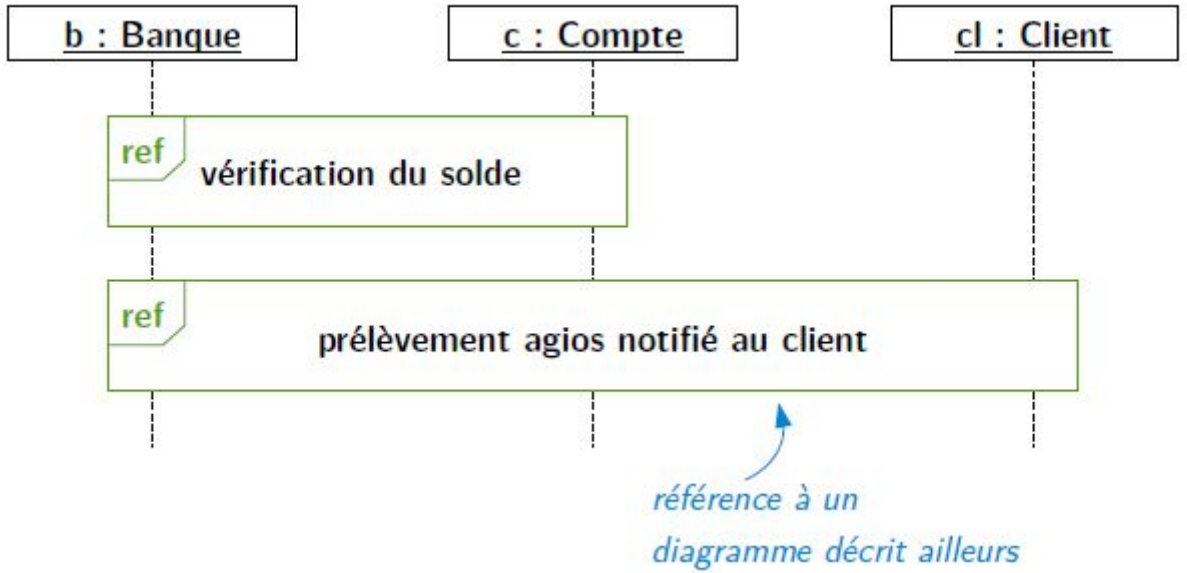
"Loop" (boucle) est noté **"loop"**. Cet opérateur est utilisé pour décrire un ensemble d‘ intéraction qui s'exécutent **en boucle**. En général, une contrainte appelée **garde** indique le nombre de répétitions (minimum et maximum) ou bien une condition booléenne à respecter.

Itération

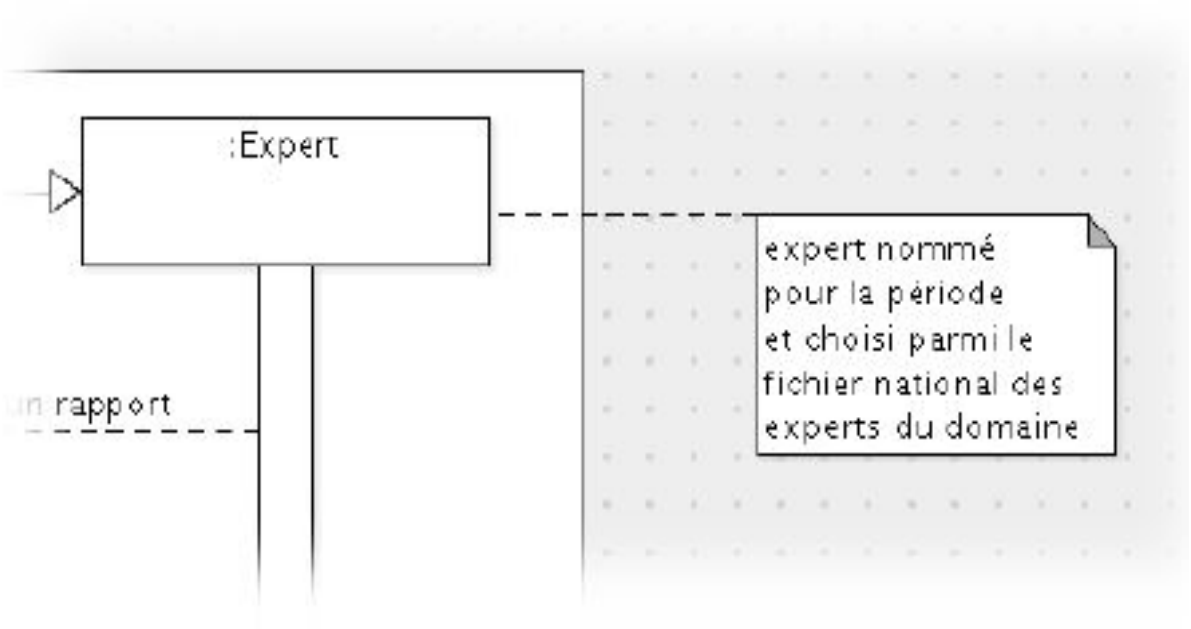

cat : Catalogue
livre : Livre
loop(livre in cat)
premierAuteur()
auteur
objets sur lesquels
répéter la boucle

LE LABEL« REF »

Réutiliser une interaction consiste à placer un fragment portant la référence « **ref** » là où l'interaction est utile. On spécifie le nom de l'interaction dans le fragment.



Note



Notes pour **préciser** ce qui n'est pas modélisable

Remarque

- *L'utilisation de note est le meilleur moyen de noter quelque chose d'important et qui n'est pas défini par UML.*

- *Attention néanmoins à ne pas* ***surcharger inutilement*** *le diagramme de notes inutiles au risque de le rendre illisible.*

UML 2.0

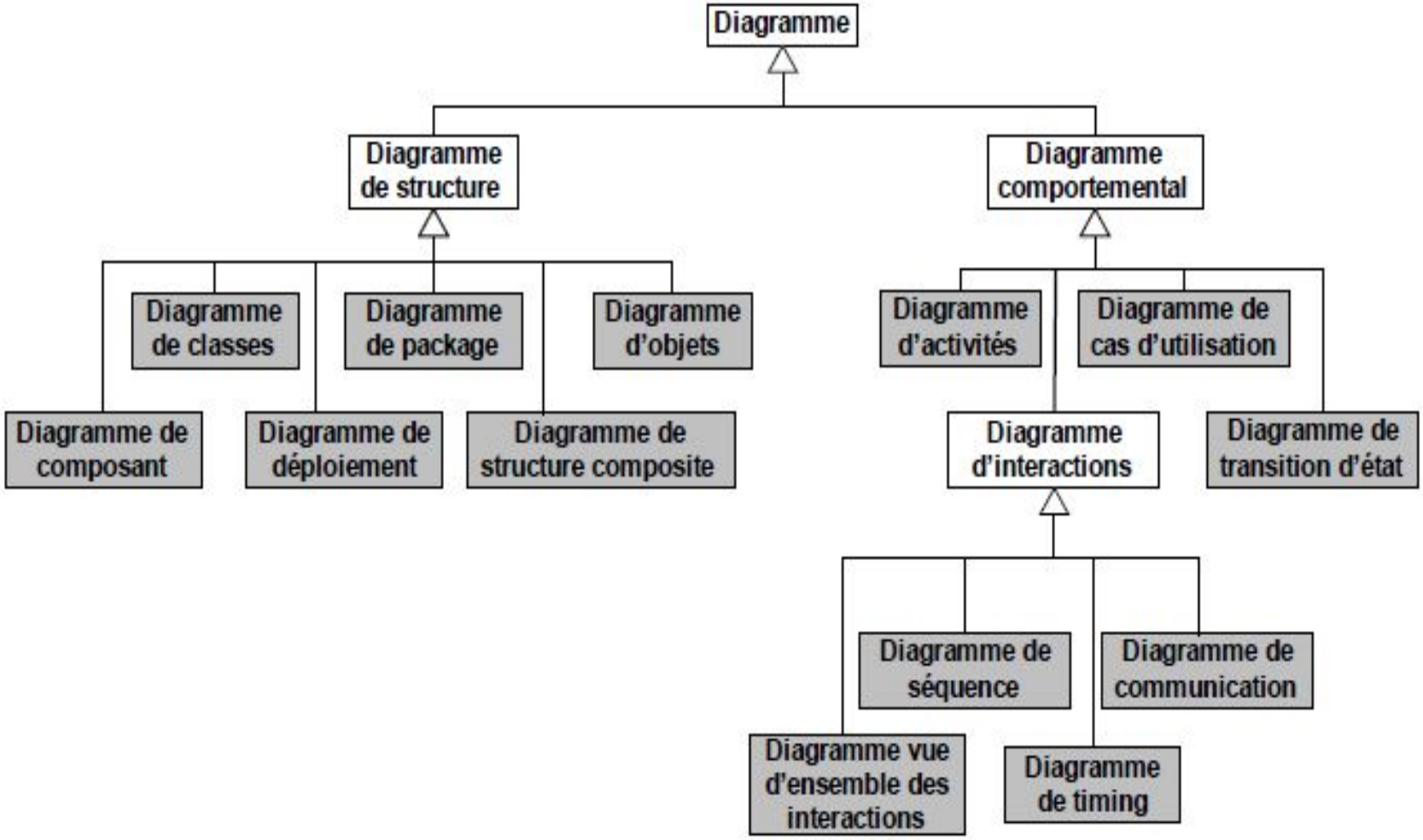
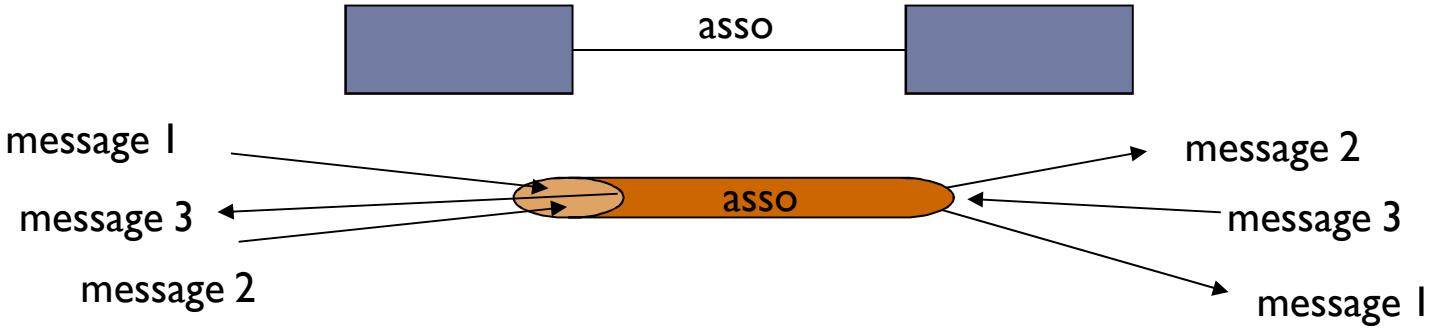


Diagramme
Diagramme de structure
Diagramme comportemental
Diagramme de classes
Diagramme de package
Diagramme d'objets
Diagramme de composant
Diagramme de déploiement
Diagramme de structure composite
Diagramme d'activités
Diagramme de cas d'utilisation
Diagramme d'interactions
Diagramme de transition d'état
Diagramme de séquence
Diagramme de communication
Diagramme vue d'ensemble des interactions
Diagramme de timing

Une association entre des classes correspond à une structure statique. C'est également le support de la collaboration entre les objets des classes.



Par une seule association, plusieurs messages passeront.