



# **Cours 3 de UML:**

## **L'aspect fonctionnel du logiciel**

M.WANE

Ingénieur Informaticien/Formateur

Analyste, Concepteur et Développeur d'Application

Email:[douvewane85@gmail.com](mailto:douvewane85@gmail.com)

**L'aspect fonctionnel du logiciel** permet de définir qui utilisera le logiciel et pour quoi faire, comment les fonctionnalités vont se dérouler, etc.

### **I) Vue des processus (3)**

Démontre :

- la décomposition du système en processus et actions ;
- les interactions entre les processus ;
- la synchronisation et la communication des activités parallèles.


La vue des processus s'appuie sur plusieurs diagrammes:

- **Le Diagramme de séquence**
- **Le diagramme d'activité**
- **Le diagramme de collaboration**
- **Le diagramme d'état-transition**
- **Le diagramme global d'interaction**

## I.I) Le diagramme de séquence

Les diagrammes de cas d'utilisation modélisent à **QUOI** sert le système, en organisant les interactions possibles avec les acteurs, alors que les diagrammes de séquences permettent de décrire **COMMENT** les éléments du système interagissent entre eux et avec les acteurs :

- Les objets au cœur d'un système interagissent en s'échangeant des messages.
- Les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

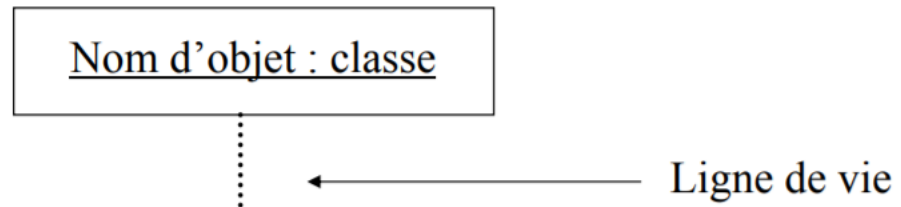


Le Diagramme de Séquence est description d'une simulation du fonctionnement d'un cas d'utilisation. Il met en jeu :

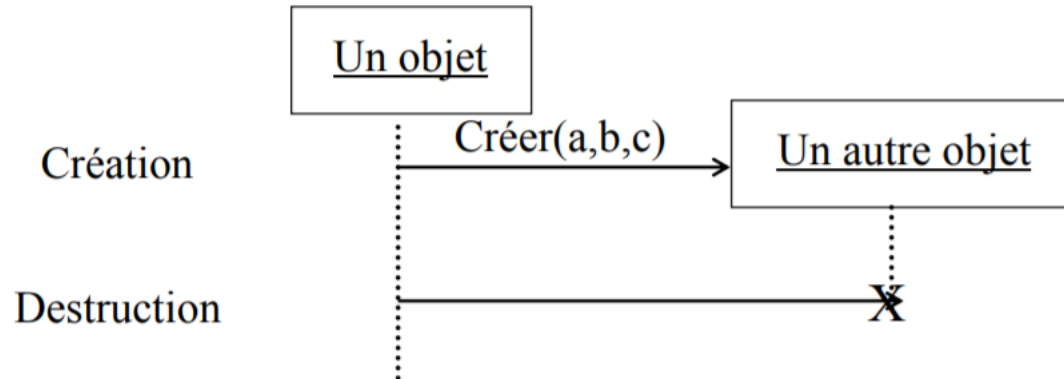
- un acteur
- un ensemble d'objets
  - la chronologie des échanges entre les objets (messages avec leurs paramètres et leur valeur de retour)
- les contraintes de temps.

## I.II) Représentant les interactions entre objets

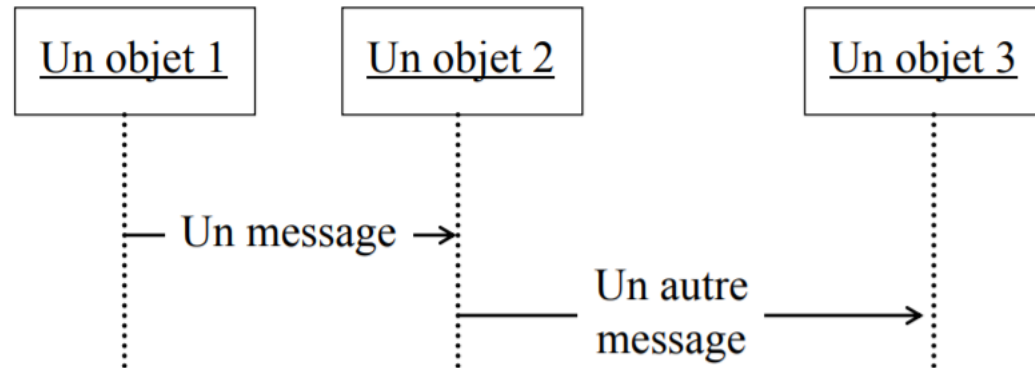
- La représentation des objets



- La création et la destruction des objets



- La représentation des interactions

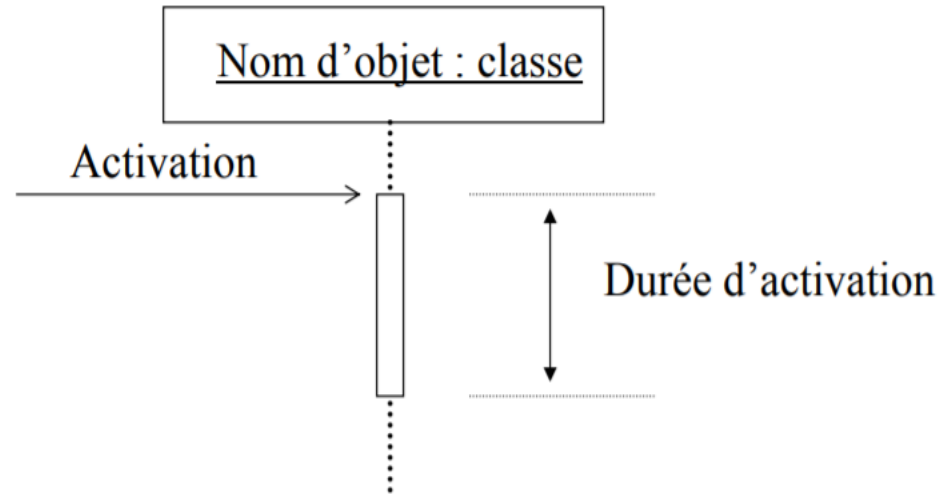


La dimension verticale représente l'écoulement dans le temps (de haut en bas par défaut).

L'ordre d'envoi est ainsi donné par la position des messages sur les lignes de vie des objets. Il est possible de graduer la dimension verticale pour exprimer précisément les contraintes temporelles.

La dimension horizontale n'a pas de signification particulière.

- La représentation des périodes d'activité des objets



Une période d'activité est le temps durant lequel un objet effectue une action, soit directement, soit par l'intermédiaire d'un autre objet qui lui sert de sous-traitant.

## ● Messages

Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie :

- Ils sont représentés par des flèches
- Ils sont présentés du haut vers le bas le long des lignes de vie, dans un ordre chronologique

Un message définit une communication particulière entre des lignes de vie (objets ou acteurs). Plusieurs types de messages existent, dont les plus courants :

- l'envoi d'un signal ;
- l'invocation d'une opération (appel de méthode) ;
- la création ou la destruction d'un objet.

La réception des messages provoque une période d'activité (rectangle vertical sur la ligne de vie) marquant le traitement du message (spécification d'exécution dans le cas d'un appel de méthode).

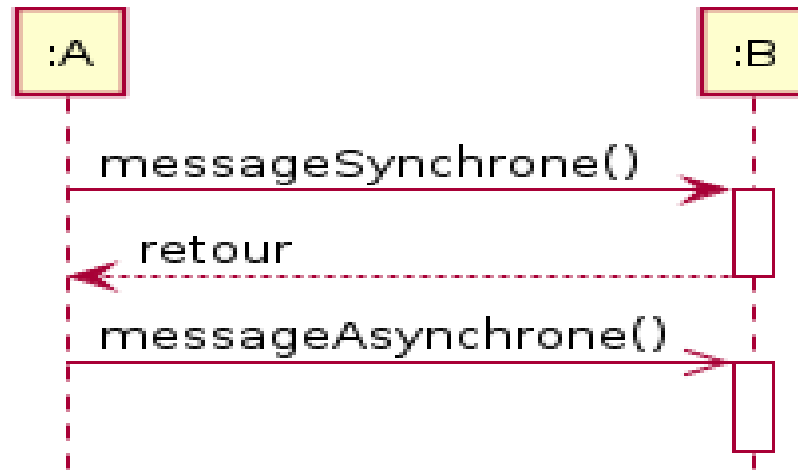
## Messages synchrones et asynchrones

Un **message synchrone** bloque l'expéditeur jusqu'à la réponse du destinataire. Le flot de contrôle passe de l'émetteur au récepteur.

- Si un objet A envoie un message synchrone à un objet B, A reste bloqué tant que B n'a pas terminé.
- On peut associer aux messages d'appel de méthode un message de retour (en pointillés) marquant la reprise du contrôle par l'objet émetteur du message synchrone.

Un **message asynchrone** n'est pas bloquant pour l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.





- **Création et destruction d'objets (et de lignes de vie)**

**Création** : message asynchrone stéréotypé

**<< create >>** pointant vers le rectangle en tête de la ligne de vie

**Destruction** : message asynchrone stéréotypé **<< destroy >>** précédant une croix sur la ligne de vie

- **Fragment combiné**

Un fragment combiné permet de décomposer une interaction complexe en fragments suffisamment simples pour être compris.

- Recombiner les fragments restitue la complexité.
- Syntaxe complète avec UML 2 : représentation complète de processus avec un langage simple (ex : processus parallèles).

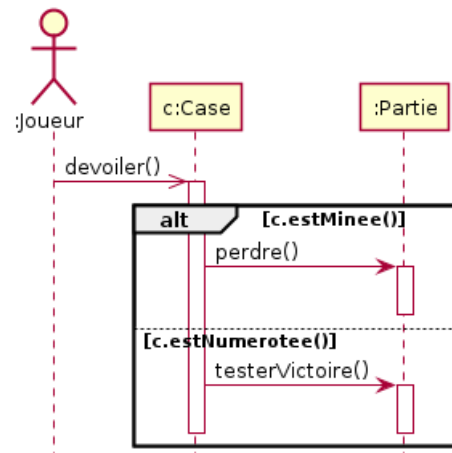
Un fragment combiné se représente de la même façon qu'une interaction. Il est représenté par un rectangle dont le coin supérieur gauche contient un pentagone.

Dans le pentagone figure le type de la combinaison (appelé *opérateur d'interaction*).

- ❖ **Fragment alt** : opérateur conditionnel

Les différentes alternatives sont spécifiées dans des zones délimitées par des pointillés.

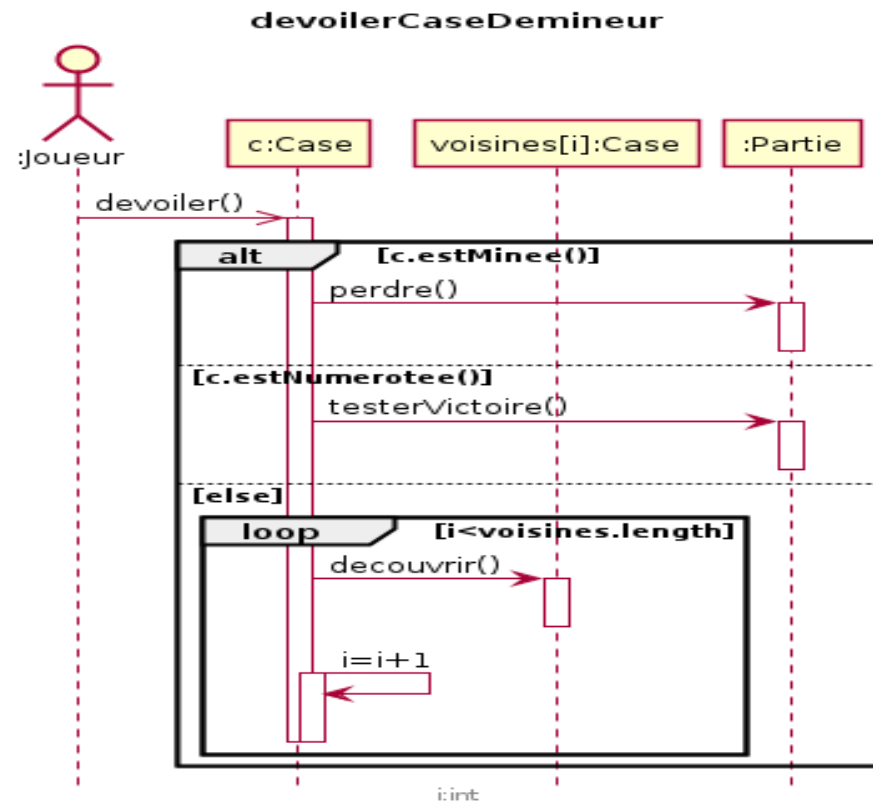
- Les conditions sont spécifiées entre crochets dans chaque zones.
- On peut utiliser une clause [else]



## ❖ Fragment loop : opérateur d'itération

Le fragment loop permet de répéter ce qui se trouve en son sein.

On peut spécifier entre crochets à quelle condition continuer.



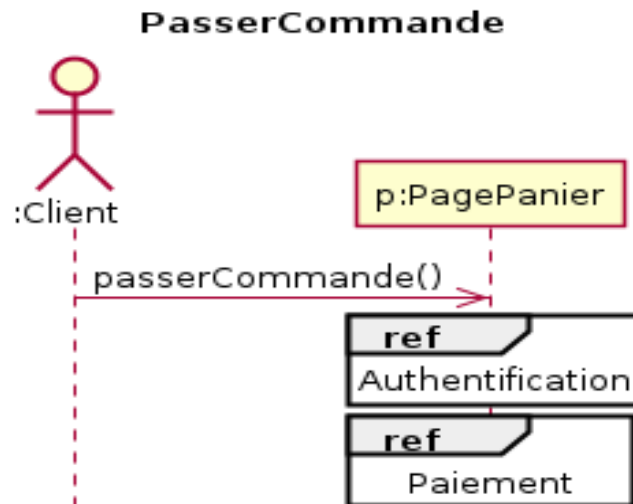
## Remarques

- Les fragments peuvent s’imbriquer les uns dans les autres
- On peut émettre des messages réflexifs et dans ce cas, on définit une activité “dans” l’activité
- Tous les éléments d’un diagramme doivent être définis. Typiquement, les attributs doivent correspondre :
  - ❖ soit à des attributs définis dans un diagramme de classes au niveau de la ligne de vie contrôlant le flux d’exécution
  - ❖ soit à des attributs définis localement au diagramme de séquence (ici, i)
- **Opérateurs de flux de contrôle**
- **opt** (*facultatif*<sup>\*</sup>) : Contient une séquence qui peut ou non se produire. Dans la protection, vous pouvez spécifier la condition sous laquelle elle se produit.
- **alt** : Contient une liste des fragments dans lesquels se trouvent d’autres séquences de messages. Une seule séquence peut se produire à la fois.
- **loop** : Le fragment est répété un certain nombre de fois. Dans la protection, on indique la condition sous laquelle il doit être répété.
- **break** : Si ce fragment est exécuté, le reste de la séquence est abandonné. Vous pouvez utiliser la protection pour indiquer la condition dans laquelle la rupture se produira.

## • Réutilisation de séquences

Un fragment **ref** permet d'indiquer la réutilisation d'un diagramme de séquences défini par ailleurs.

- En supposant qu'il existe un diagramme intitulé Authentification et un autre Paiement, on peut établir le diagramme suivant :



## **II) Vue logique (2)**

**La vue logique** a pour but d'identifier les éléments du domaine, les relations et interactions entre ces éléments. Cette vue organise les éléments du domaine en « catégories ». Deux diagrammes peuvent être utilisés pour cette vue:

- **Le diagramme de classes**
- **Le diagramme d'objets**

**II,1) Le diagramme de classes**