UML

Diagramme d’interaction

Diagramme d'interaction : Introduction (1)

- Une interaction décrit le comportement d'un classeur en se focalisant sur l'échange d'informations entre les éléments du classeur. Les participants à une interaction sont appelés «lignes de vie».
- Une ligne de vie représente un participant unique à une interaction.

Exemple:
Le diagramme de classes suivante montre la structure interne d'un système de pilotage automatique composé des trois classes : PiloteAutomatique, Voiture et Moteur. Ce diagramme n'indique pas comment le pilotage est réalisé.
Pour ajouter un aspect dynamique à la modélisation, il faut « descendre » au niveau des instances et montrer comment elles interagissent.

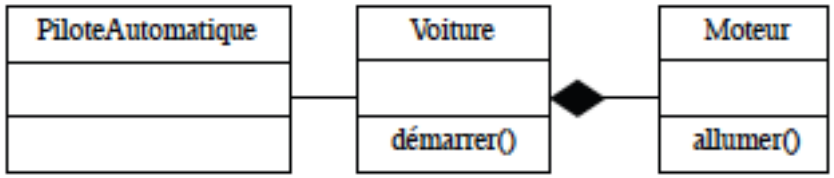


Diagramme d'interaction : Introduction (2)

L'illustration d'une interaction :

le diagramme de séquence;

le diagramme de communication;

le diagramme de timing;

- Les diagrammes de communication et de séquence représentent des interactions entre des lignes de vie.
- Un diagramme de séquence montre des interactions sous un angle temporel, et plus particulièrement le séquencement temporel de messages échangés entre des lignes de vie, tandis qu'un diagramme de communication montre une représentation spatiale des lignes de vie.

Diagramme d'interaction : Introduction (3)

- Un objet interagit pour implémenter un comportement.

On peut décrire cette interaction de deux manières complémentaires :

- l'une est centrée sur des objets individuels (diagramme d'états-transitions)
- l'autre sur une collection d'objets qui coopèrent (diagrammes d'interaction).

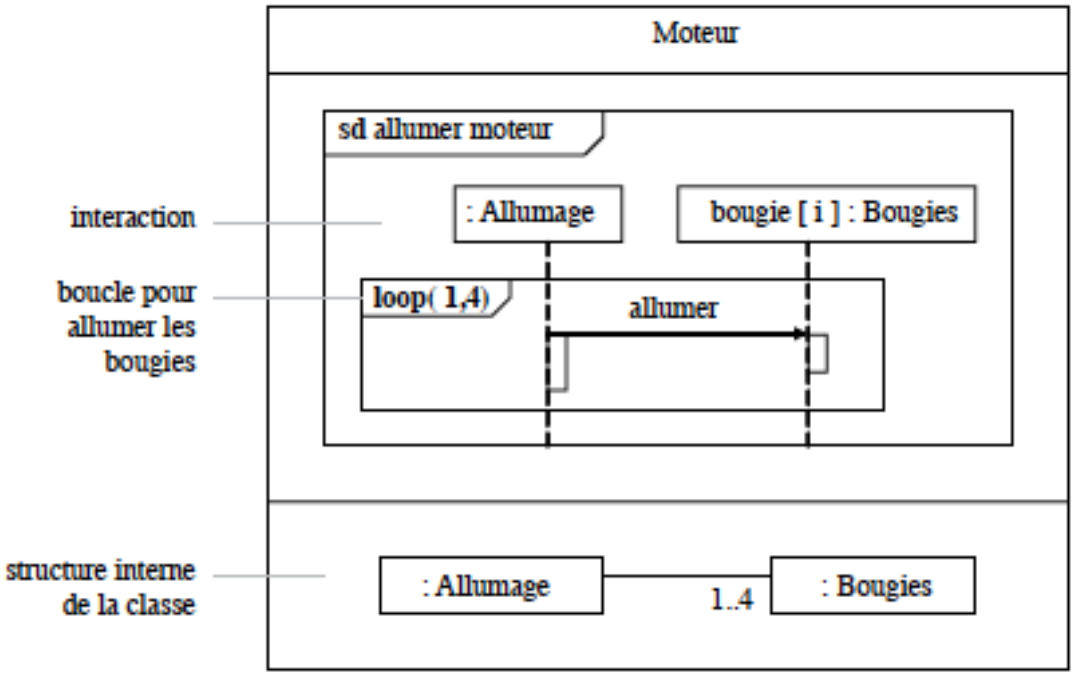
- **Le diagramme de communication** est un diagramme d'interaction mettant l'accent sur l'organisation structurelle des objets qui envoient et reçoivent des messages.
- **Le diagramme de séquence** est un diagramme d'interaction mettant l'accent sur la chronologie de l'envoi des messages.

Les diagrammes d'interaction permettent d'établir un lien entre les diagrammes de cas d'utilisation et les diagrammes de classes : ils montrent comment des objets (*i.e.* des instances de classes) communiquent pour réaliser une certaine fonctionnalité. Ils apportent ainsi un aspect dynamique à la modélisation du système.

Interactions dans les classeurs structurés

Considérons une classe qui est décomposée pour montrer sa structure interne :

- une classe Moteur est composée des classes Allumage et Bougie.
- Un diagramme de séquence montre comment les bougies et l'allumage interagissent pour démarrer le moteur.



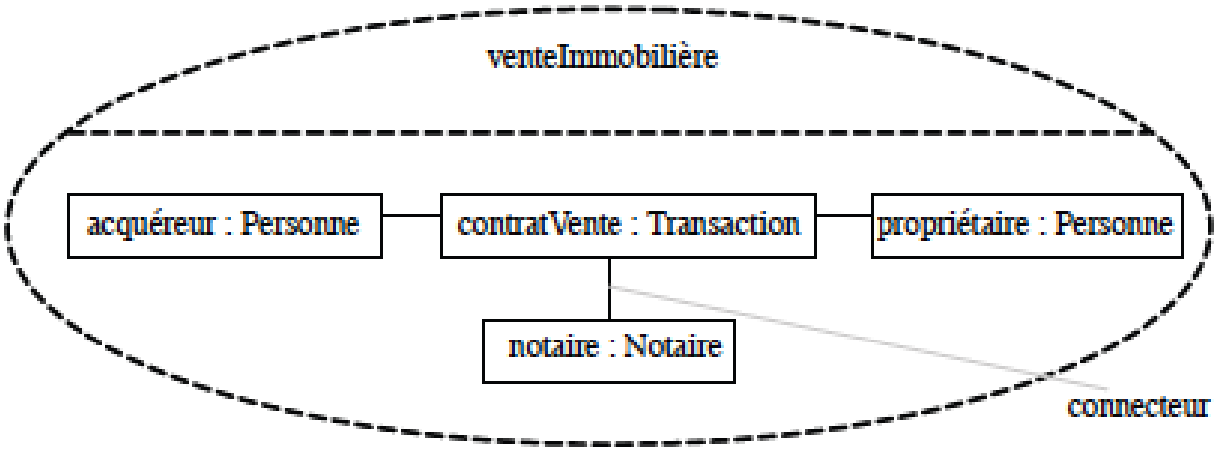
Collaboration (1)

- Une collaboration permet de décrire la mise en œuvre d'une fonctionnalité par un jeu de participants. Un rôle est la description d'un participant. Contrairement aux paquetages et aux classeurs structurés, une collaboration ne détient pas les instances liées à ses rôles.
- Une collaboration montre des instances qui collaborent dans un contexte donné pour mettre en oeuvre une fonctionnalité d'un système.

- Une collaboration se représente par une ellipse en trait pointillé comprenant deux compartiments.
- Le compartiment supérieur contient le nom de la collaboration ayant pour syntaxe :
- **<nomDuRôle>[':' <NomDuType>][multiplicité]**
- Le compartiment inférieur montre les participants à la collaboration.

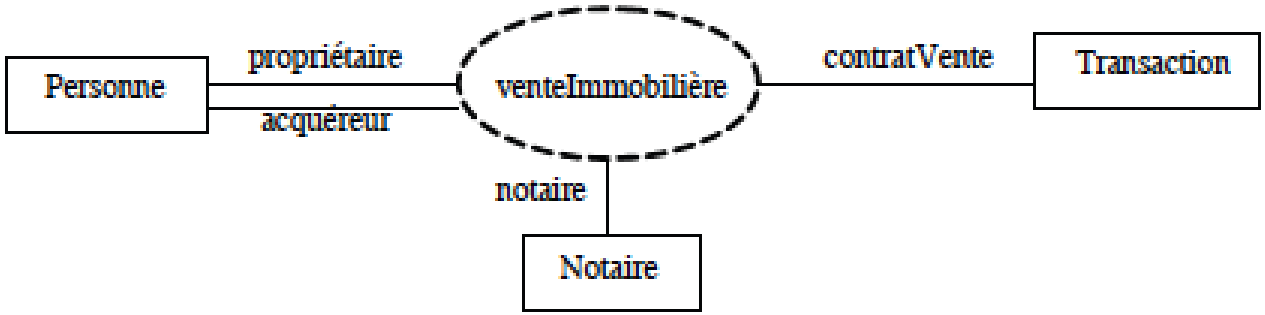
Collaboration (2)

Exemple: Transaction immobilière:

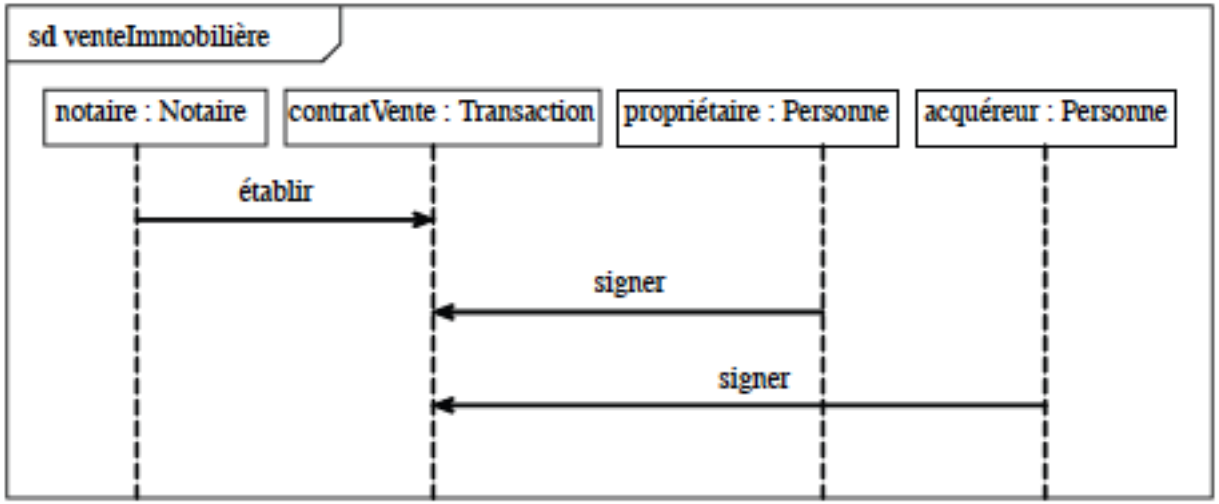


Une collaboration peut aussi se représenter par une ellipse sans compartiment, portant le nom de la collaboration en son sein.
Les instances sont reliées à l'ellipse par des lignes qui portent le nom du rôle de chaque instance.

Collaboration (3)



Les interactions au sein d'une collaboration peuvent être représentées par un diagramme d'interaction :



Notations

- Un diagramme d'interaction se représente par un rectangle contenant, dans le coin supérieur gauche, un pentagone accompagné du mot-clé **sd** lorsqu'il s'agit d'un diagramme de séquence, et de **com** lorsqu'il s'agit d'un diagramme de communication.
- La liste des lignes de vie qui figurent dans le diagramme peut suivre le nom de l'interaction.

- Une ligne de vie se représente par un rectangle auquel est accrochée une ligne verticale pointillée.

Le rectangle contient un identifiant dont la syntaxe est la suivante :

[<nomDeLaLigneDeVie> ['['sélecteur']']] : <NomDeClasseur> [décomposition]

Les diagrammes d'interaction sont utilisés tout au long du cycle de vie d'un projet

Exemple

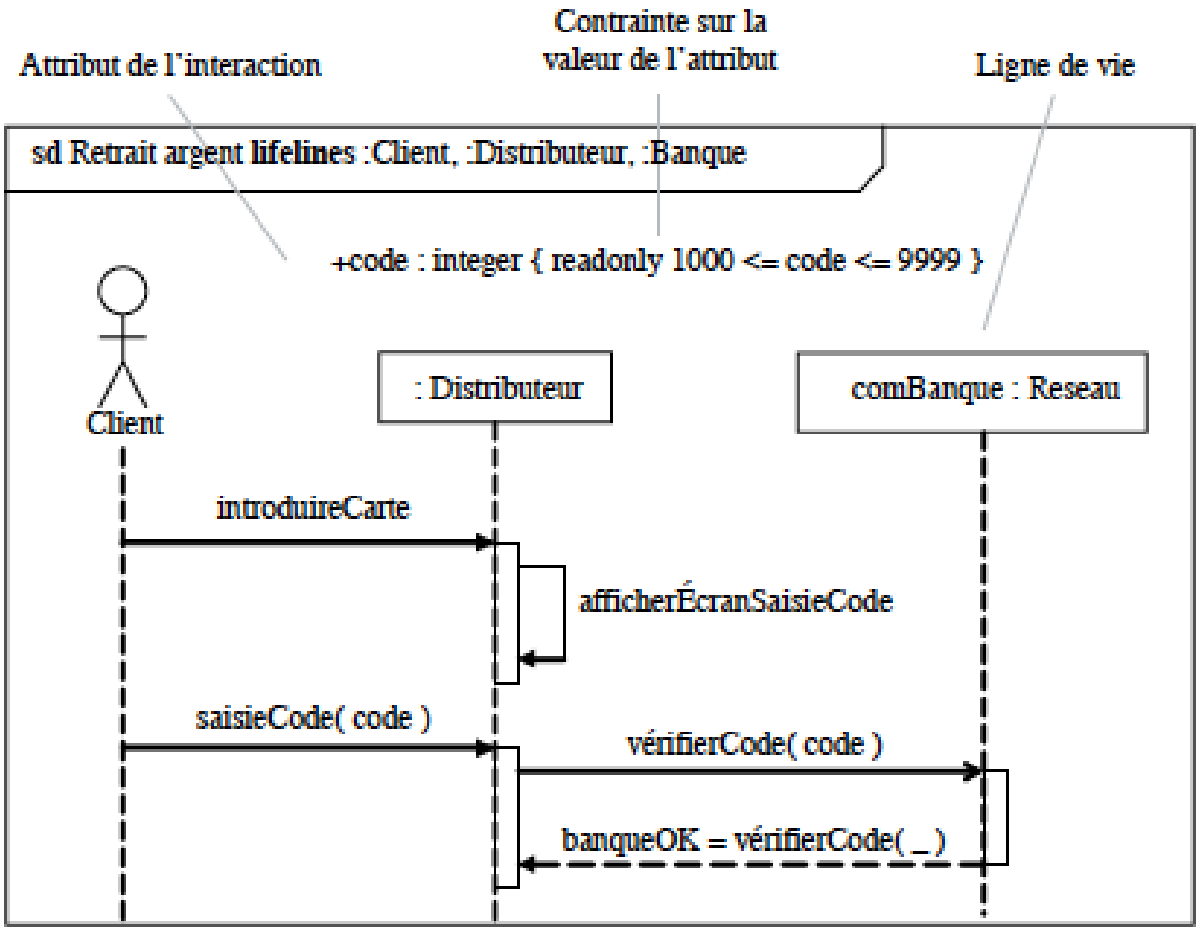

Attribut de l'interaction
Contrainte sur la valeur de l'attribut
Ligne de vie
sd Retrait argent lifelines :Client, :Distributeur, :Banque
+code : integer { readonly 1000 <= code <= 9999 }
Client
: Distributeur
comBanque : Reseau
introduireCarte
afficherÉcranSaisieCode
saisieCode(code)
vérifierCode(code)
banqueOK = vérifierCode(_)

Diagramme de séquence

Les diagrammes de cas d'utilisation modélisent à QUOI sert le système, en organisant les interactions possibles avec les acteurs.

Les diagrammes de classes permettent de spécier la structure et les liens entre les objets dont le système est composé : ils spécie QUI sera à l'oeuvre dans le système pour réaliser les fonctionnalités décrites par les diagrammes de cas d'utilisation.

Les diagrammes de séquences permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs.

Les objets au coeur d'un système interagissent en s'échangent des messages.

Les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

Les objectifs

- Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie. Un message définit une communication particulière entre des lignes de vie.

Plusieurs types de messages existent:

- l'envoi d'un signal ;
- l'invocation d'une opération ;
- la création ou la destruction d'une instance.

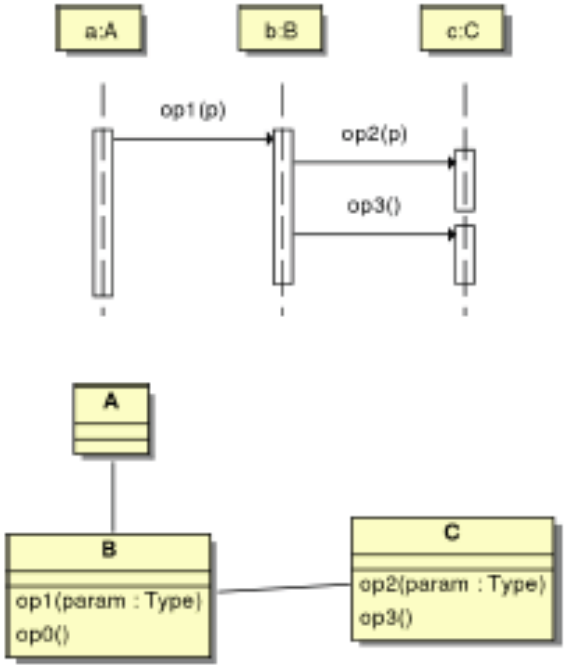
Les messages (1)

- Un message synchrone se représente par une flèche à l'extrémité pleine qui pointe sur le destinataire du message. Ce message peut être suivi d'une réponse qui se représente par une flèche en pointillé.
- Un message asynchrone se représente par une flèche à l'extrémité ouverte.

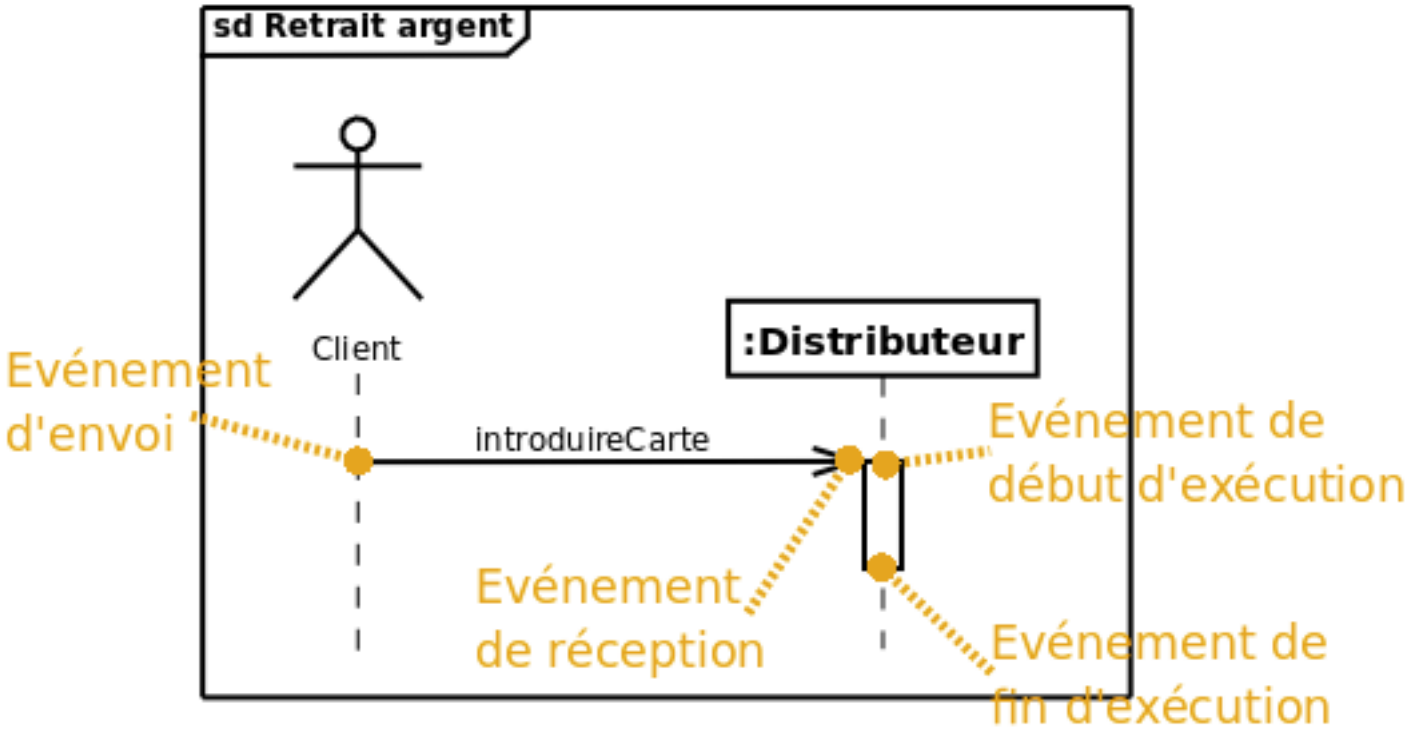
- La création d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie.
- La destruction d'un objet est matérialisée par une croix qui marque la fin de la ligne de vie de l'objet.

Les messages (2)

- Les messages synchrones correspondent à des opérations dans le diagramme de classes.
- Envoyer un message et attendre la réponse pour poursuivre son activité revient à invoquer une méthode et attendre le retour pour poursuivre ses traitements.



Evénements et messages (1)

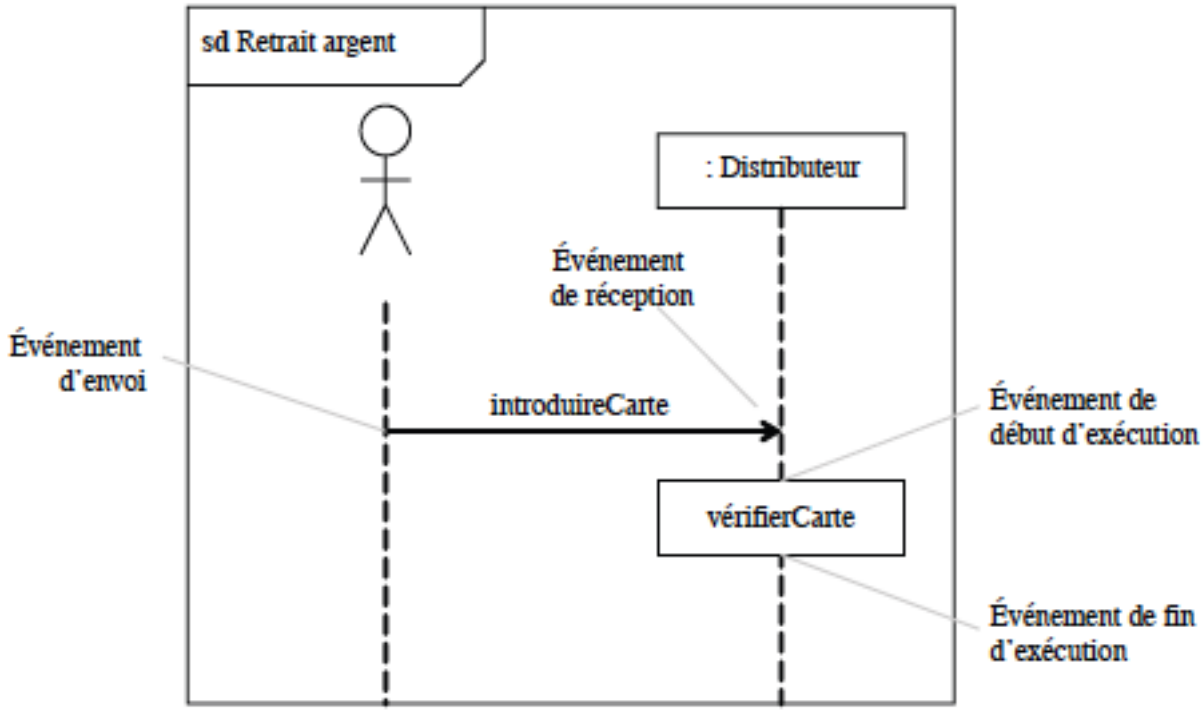


L'invocation d'une opération ou l'envoi d'un signal peut déclencher une réaction chez le Récepteur.

Evénements et messages (2)

La spécification de l'exécution d'une réaction se fait par un rectangle blanc ou gris placé sur la ligne de vie.

Autre notation : un rectangle portant un label.



Evénements et messages (3)

Un message complet est tel que les événements d'envoi et de réception sont connus.Un message complet est représenté par une flèche partant d'une ligne de vie et arrivant à une autre ligne de vie.

Un message perdu est tel que l'événement d'envoi est connu, mais pas l'événement de réception. La flèche part d'une ligne de vie mais arrive sur un cercle indépendant marquant la méconnaissance du destinataire.

Un message trouvé est tel que l'événement de réception est connu, mais pas l'événement d'émission (une flèche qui part d’une boule noire).

Syntaxe de messages

<nomDuSignalOuDeOpération> [’(’ [<argument>’,’ … ’)’]

où la syntaxe des arguments est la suivante :

[<nomDeParamètre> ’=’] <valeur de l’argument>

Par défaut, les paramètres transmis ne peuvent pas être modifiés par le récepteur (les arguments sont « en entrée »). Pour indiquer que des paramètres peuvent être modifiés (argument « en sortie » ou « en entrée/sortie »), il faut écrire :

<nomDuParamètreEnSortie> [’:’ <valeur de l’argument>]

Pour transmettre uniquement la valeur des arguments, le **caractère -** peut être utilisé en lieu et place de n’importe quel argument pour signifier : valeur indéfinie.
Le **caractère *** peut être utilisé en lieu et place du message complet (tout type de message est accepté).

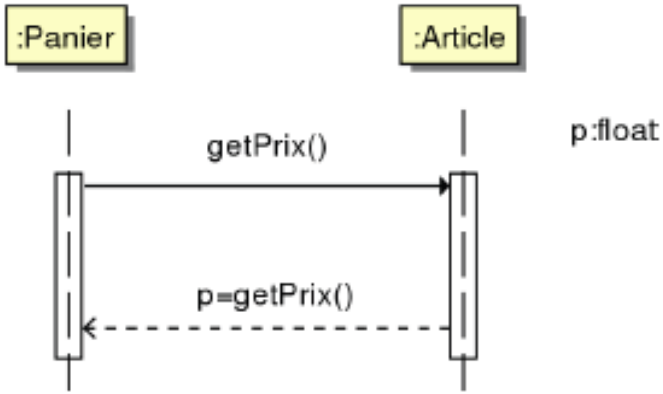
Syntaxe de messages : exemples

```
Appeler( "Capitaine Hadock", 54214110 )
afficher( x, y )
initialiser( x = 100 ) est un message dont l'argument en entrée reçoit la
valeur 100.
f( x :12 ) est un message avec un argument en entrée/sortie x qui prend
initialement la valeur 12.
appeler( "Castafiore", - )
```

Messages de retour

- Le récepteur d'un message synchrone rend la main à l'émetteur du message en lui envoyant un message de retour
- Les messages de retour sont optionnels : la finn de la période d'activité marque également la fin de l'exécution d'une méthode.

Ils sont utilisés pour spécifier le résultat de la méthode invoquée.

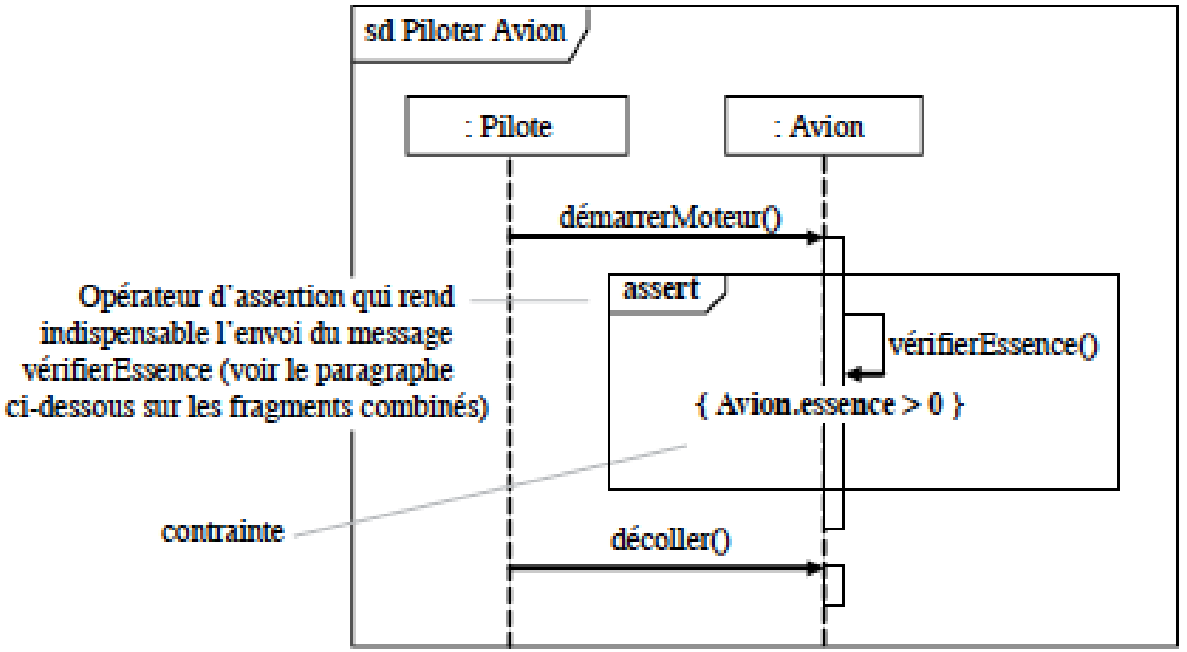


La syntaxe de réponse à un message est la suivante :
[<attribut> '='] message [':' <valeur de retour>]
où message représente le message d'envoi.

Exemple : Montrer un message de réponse signalant que quarante-trois occurrences de la référence « Tintin » figurent dans une médiathèque.

Lignes de vie : contraintes

- Une contrainte est indiquée sur une ligne de vie par un texte entre accolades. Une contrainte peut aussi être contenue dans une note attachée à l'occurrence de l'événement sur lequel elle agit.



Fragments d'interactions combinés

Un fragment combiné représente des articulations d'interactions. Il est défini par un opérateur et des opérandes. L'opérateur conditionne la signification du fragment combiné.

- Un fragment combiné se représente de la même façon qu'une interaction : par un rectangle dont le coin supérieur gauche contient un pentagone. Dans le pentagone figure le type de la combinaison (appelé « opérateur d'interaction »).

Les opérandes d'un opérateur d'interaction sont séparés par une ligne pointillée. Les conditions de choix des opérandes sont données par des expressions booléennes entre crochets.

Les opérateurs d'interaction

La liste suivante regroupe les opérateurs d'interaction par fonctions :

- les opérateurs de choix et de boucle : **alternative**, **option**, **break** et **loop** ;
- les opérateurs contrôlant l'envoi en parallèle de messages : **parallel** et **critical region** ;
- les opérateurs contrôlant l'envoi de messages : **ignore**, **consider**, **assertion** et **negative** ;
- les opérateurs fixant l'ordre d'envoi des messages : **weak sequencing**, **strict sequencing**.

Les opérateurs d'interaction : les opérateurs de choix et de boucle

La syntaxe d'une boucle est la suivante :

loop ['(' <minint> [',' <maxint>] ')']

où la boucle est répétée au moins **minint** fois avant qu'une éventuelle condition booléenne ne soit testée (la condition est placée entre crochets sur la ligne de vie) ;

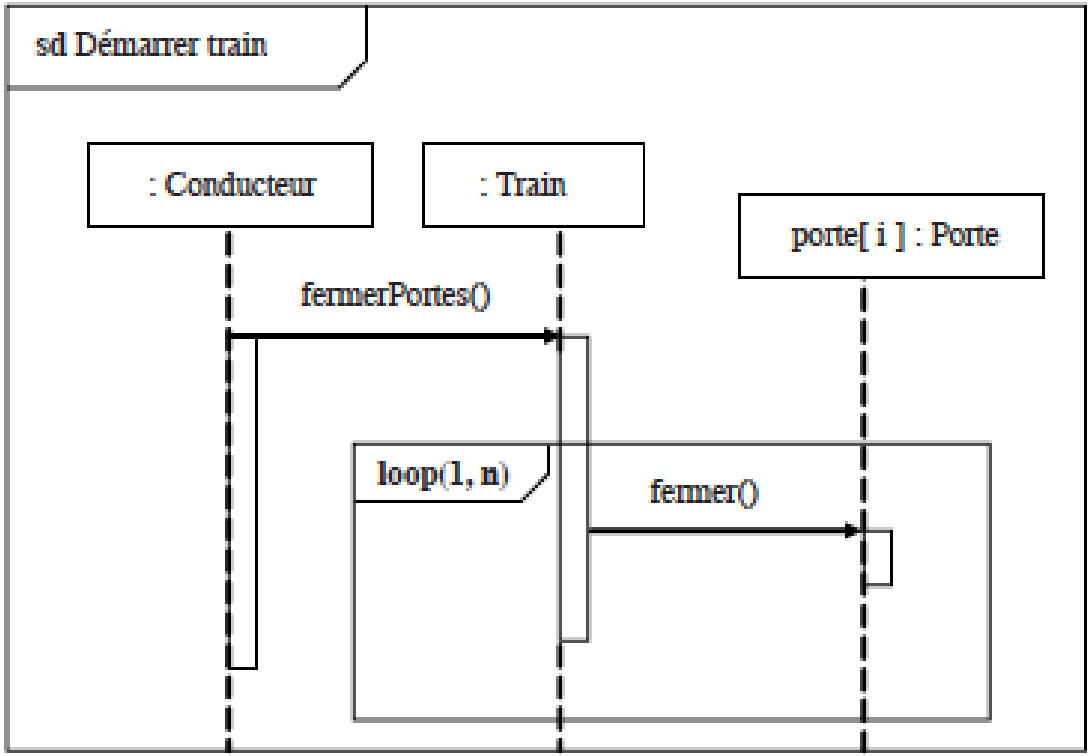
tant que la condition est vraie, la boucle continue, au plus **maxint** fois (minint est un entier supérieur ou égalà 0, maxint est un entier supérieur ou égal à minit).

loop(valeur) est équivalent à **loop(valeur, valeur)**.

loop est équivalent à **loop(0, *)**, où * signifie « illimité ».

Diagramme de séquence montrant une boucle : exemple

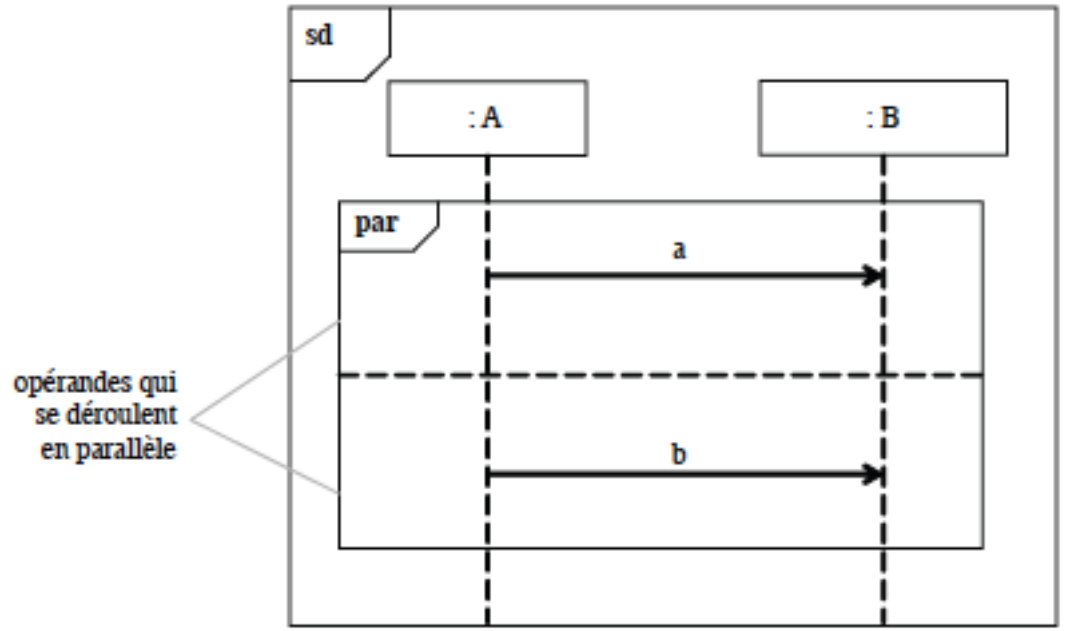
la fermeture en boucle de toutes les portes d'un train :



L'opérateur ***par*** (1)

- L'opérateur ***par*** permet d'envoyer des messages en parallèle.

Ce qui se passe de part et d'autre de la ligne pointillée est indépendant.



Exemple : Microwave - objet effectuant deux tâches en parallèle.

Assert, ignore et consider

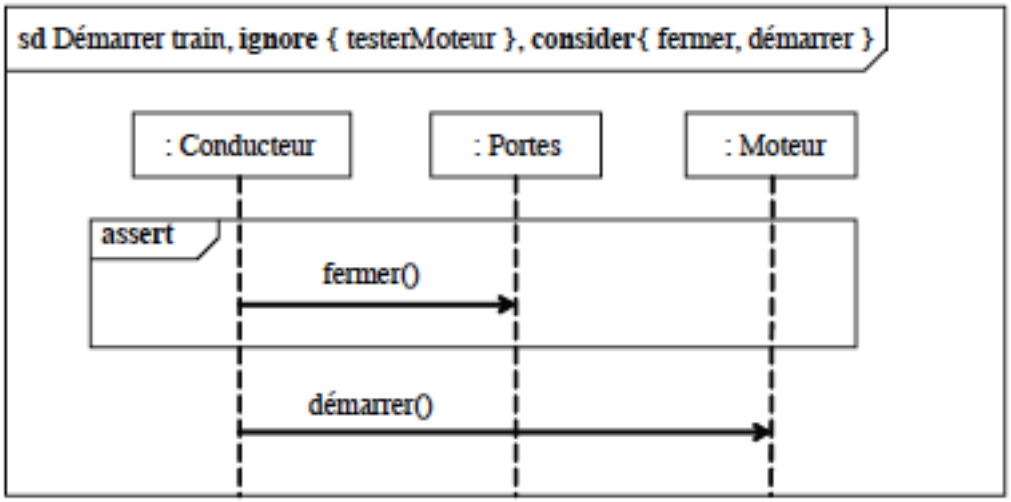
- La syntaxe de l'opérateur ignore est :

ignore { listeDesNomsDesMessagesÀIgnorer }

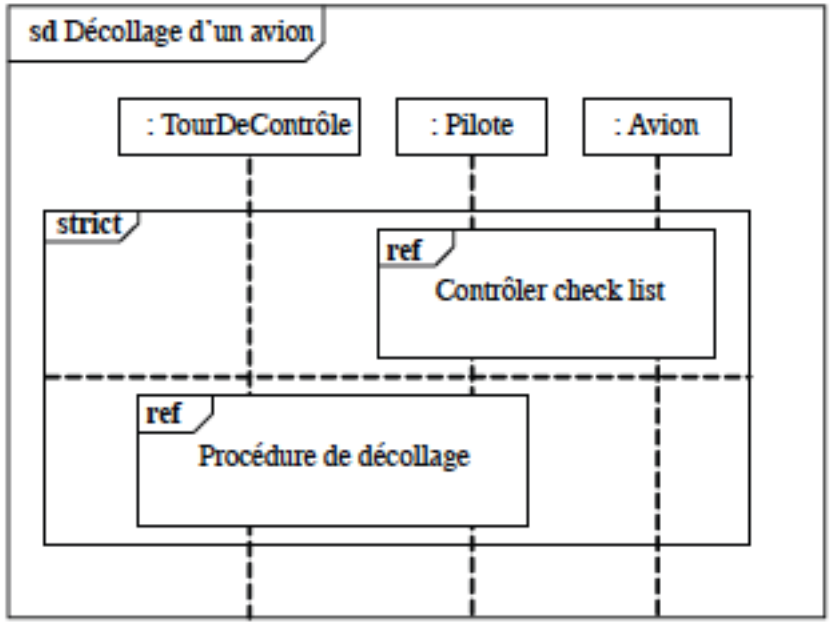
- La syntaxe de l'opérateur consider est :

consider { listeDesNomsDesMessagesÀConsidérer }

Dans les listes, les messages sont séparés par des virgules.



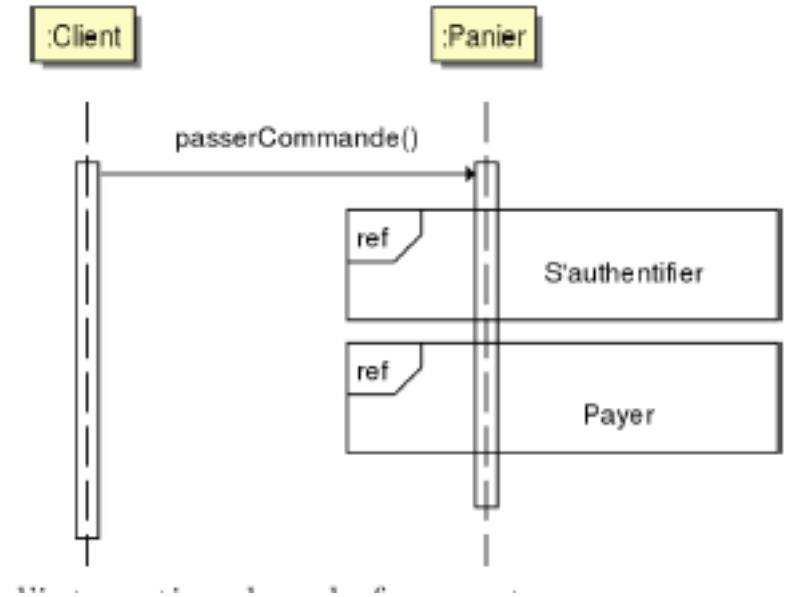
- L'opérateur ***option***, ou ***opt***, comporte une opérande et une condition de garde associée. Le sous-fragment s'exécute si la condition de garde est vraie et ne s'exécute pas dans le cas contraire.
- L'opérateur ***strict sequencing*** impose que ses opérandes se déroulent dans l'ordre strict de leur présentation, c'est-à-dire de haut en bas.



Avant de faire décoller un avion, il faut contrôler sa ckeck-list. Le détail des interactions pour contrôler la check-list et pour faire décoller l'avion n'est pas donné.
D'autres diagrammes référencés sous le label ref s'en chargent.

Décomposition d'une ligne de vie

- Une décomposition est référencée dans le rectangle en tête de ligne de vie, sous le label **ref**.



Réutilisation d'une interaction

Chaque cas d'utilisation donne lieu à un diagramme de séquences

Diagramme de communication