

COURS 10 : DIAGRAMME DE SEQUENCE

Samia BOULKRINAT

(Basé sur le cours de Ilhem BOUSSAID)

Plan

But du diagramme de séquence

Concepts principaux :

1. Les participants
2. Les messages
3. Des Fragments combinés





Diagrammes de séquence

- Les diagrammes de séquences permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs.
 - Montrent les interactions entre objets selon un point de vue temporel
 - Description de scénarios types et des exceptions
- Deux utilisations principales :
 1. Documentation des CU (point de vue Fonctionnel)
 2. Représentation précise des interactions (point de vue Dynamique)
 - identification des messages, des envois, réceptions, etc.

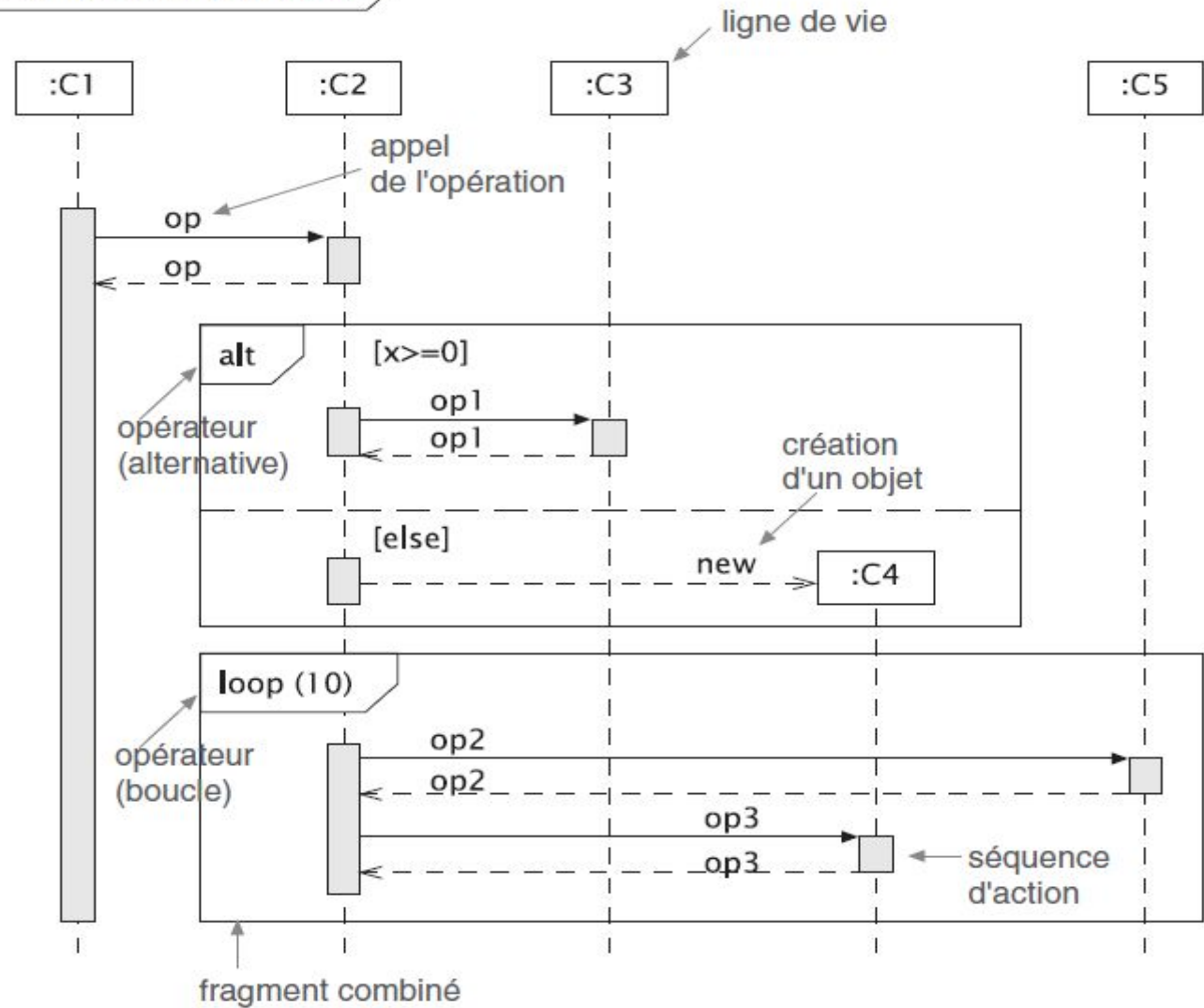
Diagrammes de séquence

Concepts principaux :

1. Les participants (le plus souvent des objets)
 - Une ligne de vie : représente un participant à une interaction (objet ou acteur).
 - Des zones d'activation
2. Les messages :
 - L'opération et éventuellement ses paramètres
 - Éventuellement son résultat
3. Des Fragments combinés
 - Alt : conditionnelle
 - Loop : boucle
 - Ref : reference a un autre diagramme de sequence (=appel de fonction)
 - Etc.

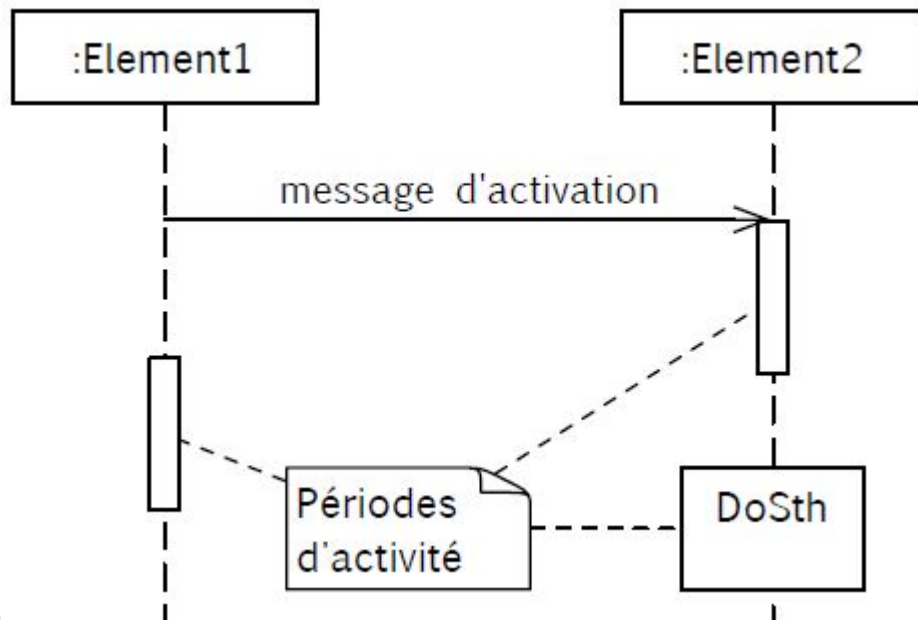
Représentation

sd Diagramme de séquence

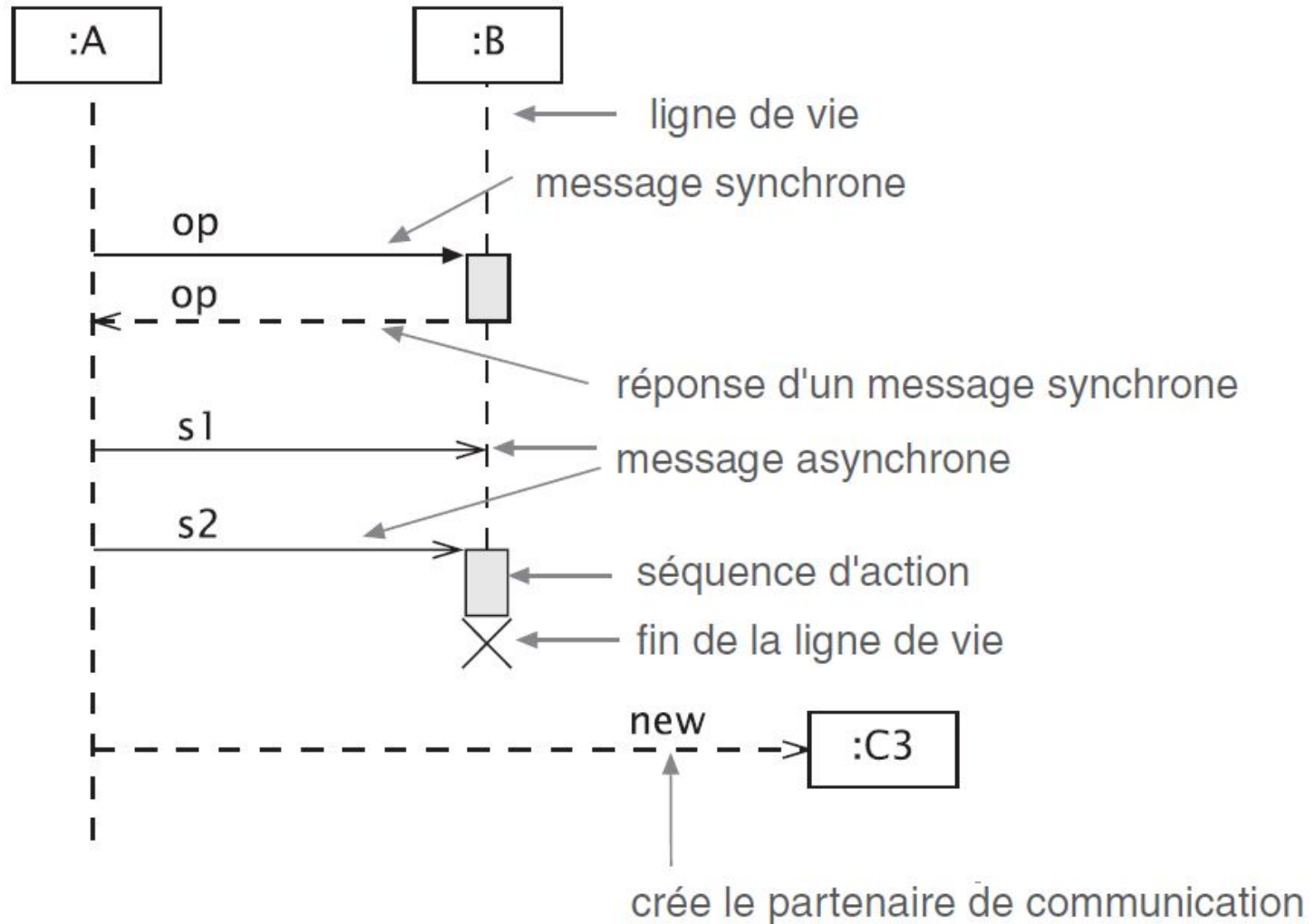


Période d'activité

- ❑ La réception des messages provoque une période d'activité (rectangle vertical sur la ligne de vie) marquant le traitement du message.
- ❑ Période durant laquelle un objet effectue une action
- ❑ Etat **actif** (= durée de vie)
- ❑ un objet peut être actif plusieurs fois



Catégories de messages



Catégories de messages

Principales catégories de messages :

- ❑ **Message synchrone** : émetteur bloqué pendant le traitement du message par le récepteur (appel)
 - Typiquement : appel de méthode (Si un objet A invoque une méthode d'un objet B, A reste bloqué tant que B n'a pas terminé.



- ❑ **Message asynchrone** : non bloquant.
 - Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré.

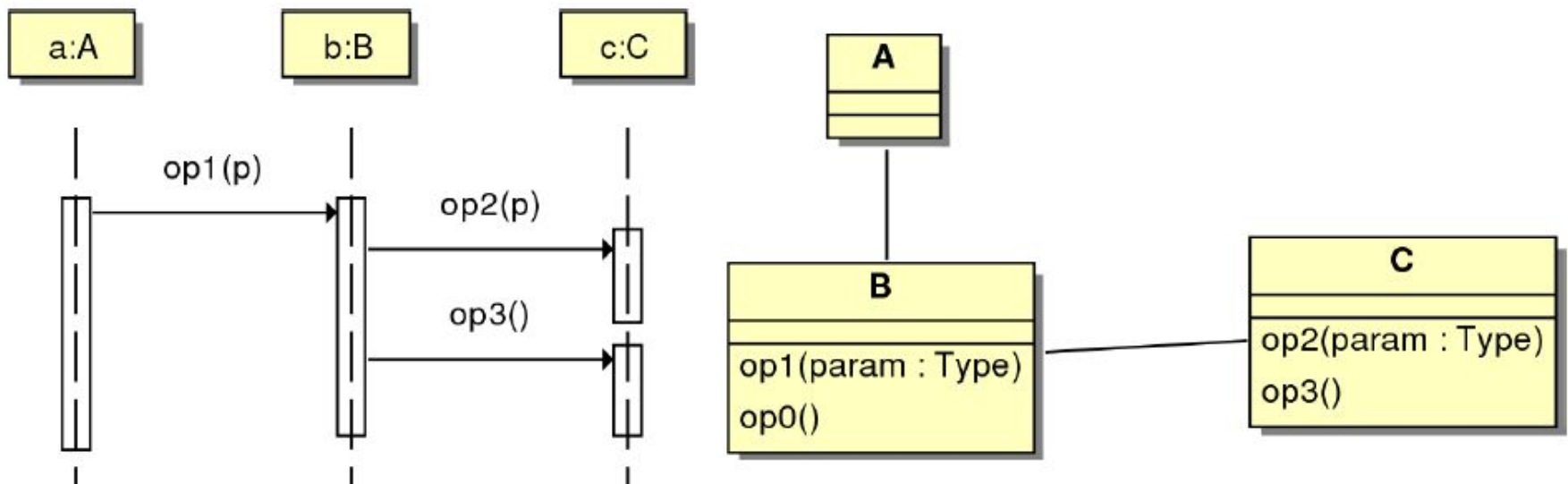


- ❑ **Message de retour** : On peut associer aux messages d'appel de méthode un message de retour (en pointillés) marquant la reprise du contrôle par l'objet émetteur du message synchrone.



Correspondance messages / opérations

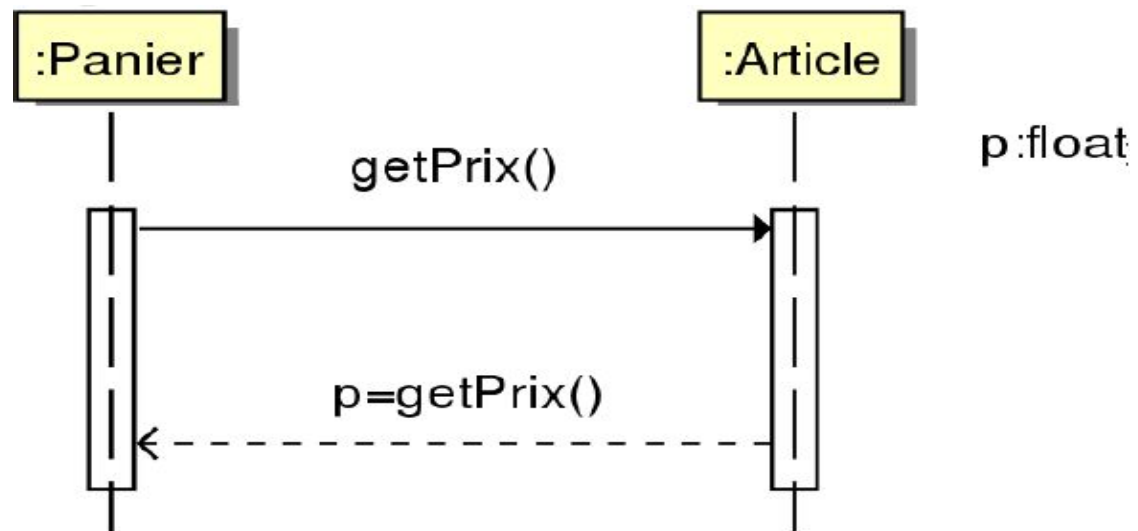
- ❑ Les messages synchrones correspondent à des opérations dans le diagramme de classes.
- ❑ Envoyer un message et attendre la réponse pour poursuivre son activité revient à invoquer une méthode et attendre le retour pour poursuivre ses traitements.





Messages de retour

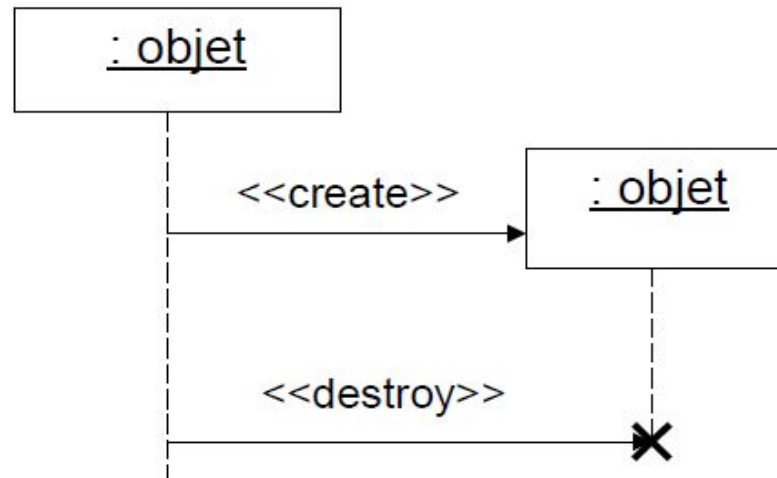
- ❑ Le récepteur d'un message synchrone rend la main à l'émetteur du message en lui envoyant un message de retour
- ❑ Les messages de retour sont optionnels : la fin de la période d'activité marque également la fin de l'exécution d'une méthode.
- ❑ Ils sont utilisés pour spécifier le résultat de la méthode invoquée.



Messages de création/destruction

Un message peut entraîner la création ou la destruction d'objets

- La **création** d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie (On peut aussi utiliser un message asynchrone ordinaire portant le nom «create»).
- La **destruction** d'un objet est matérialisée par une croix qui marque la fin de la ligne de vie de l'objet.



Catégories de messages

- message **trouvé** (Found message) :
 - message dont on ignore la provenance
 - en dehors du cadre décrit par le Diagramme de Séquence
- message **perdu** (Lost message) :
 - message envoyé, mais jamais reçu

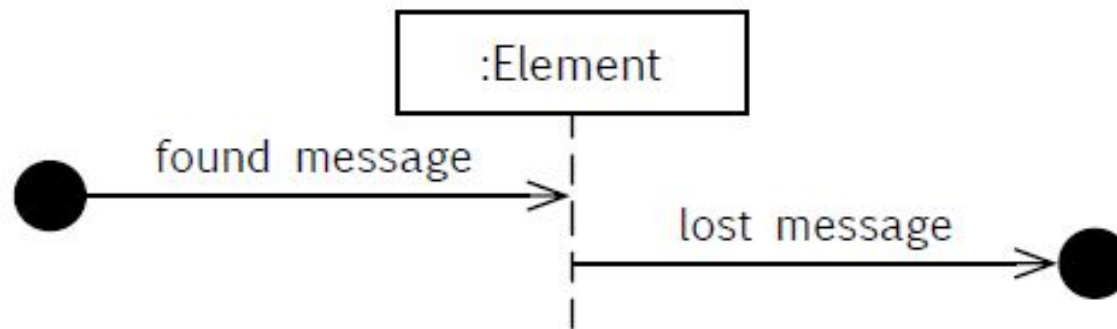
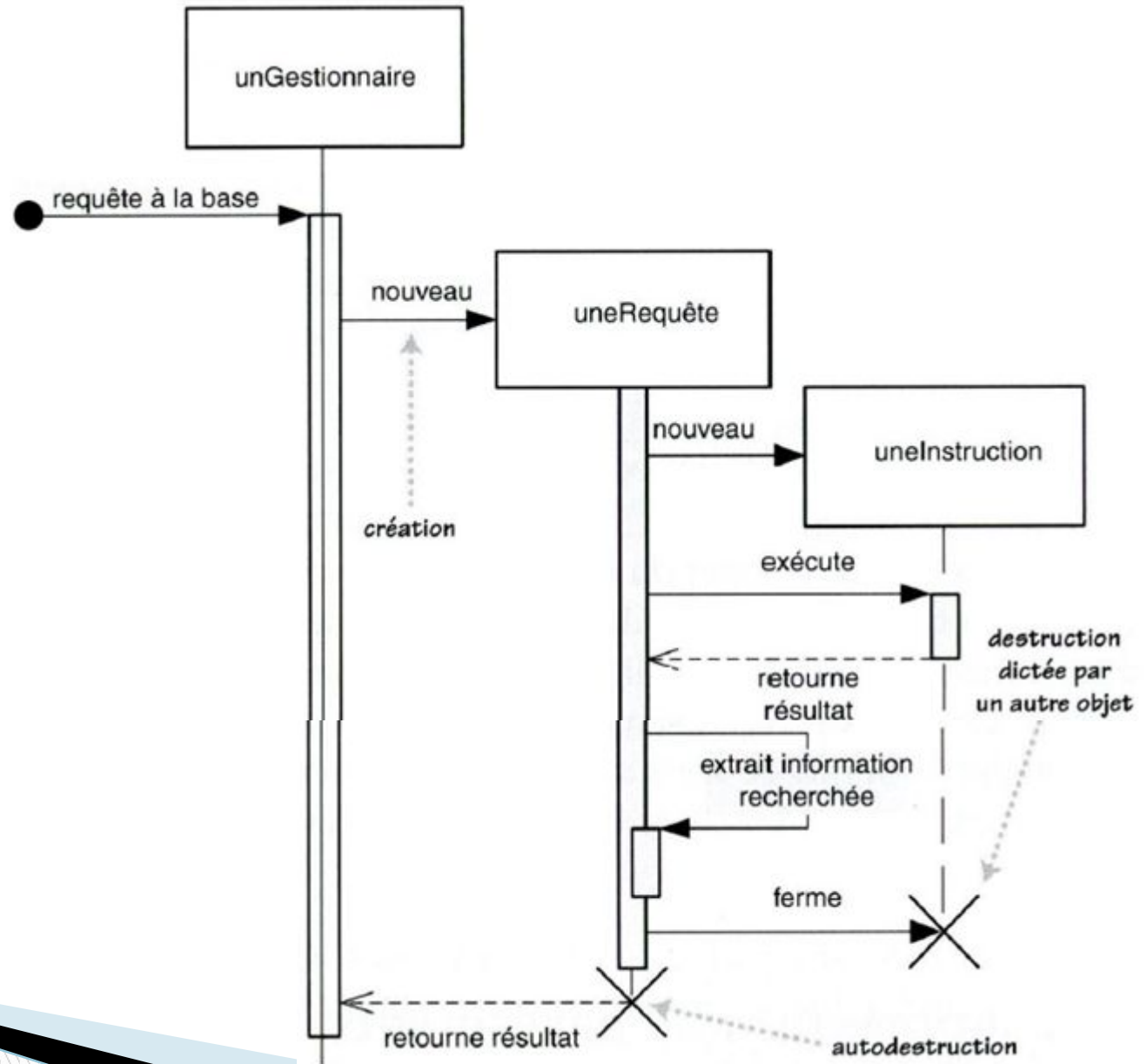


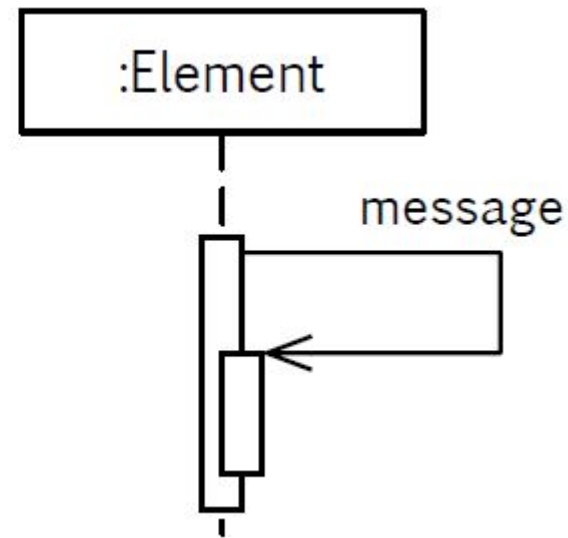
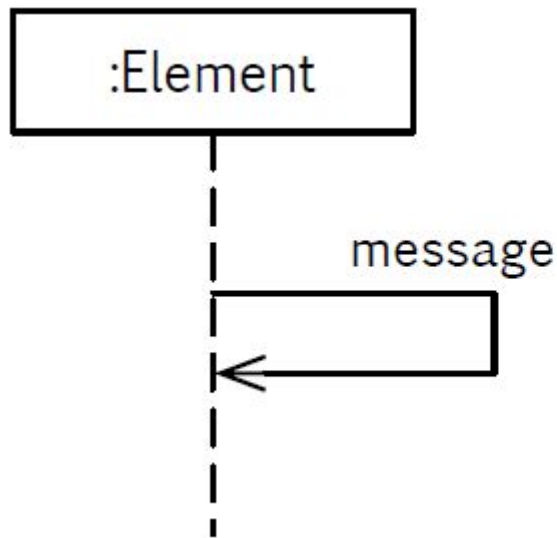
Diagramme de séquence – Exemple



Message réflexif

Un objet peut s'envoyer des messages

- appel à une autre méthode de l'objet
- appel récursif



Durées et contraintes temporelles

- Représentation des délais de transmission :

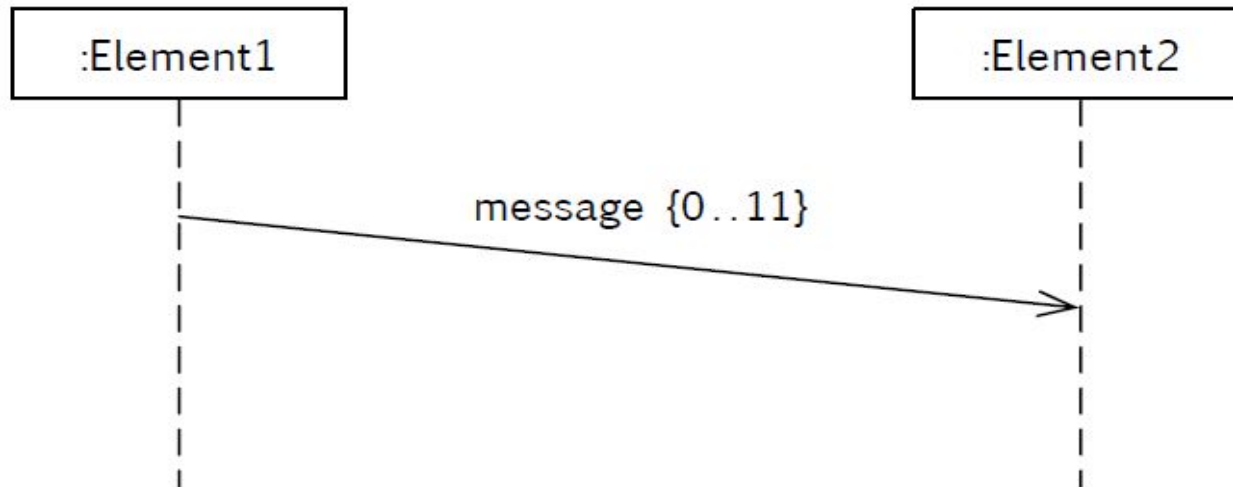
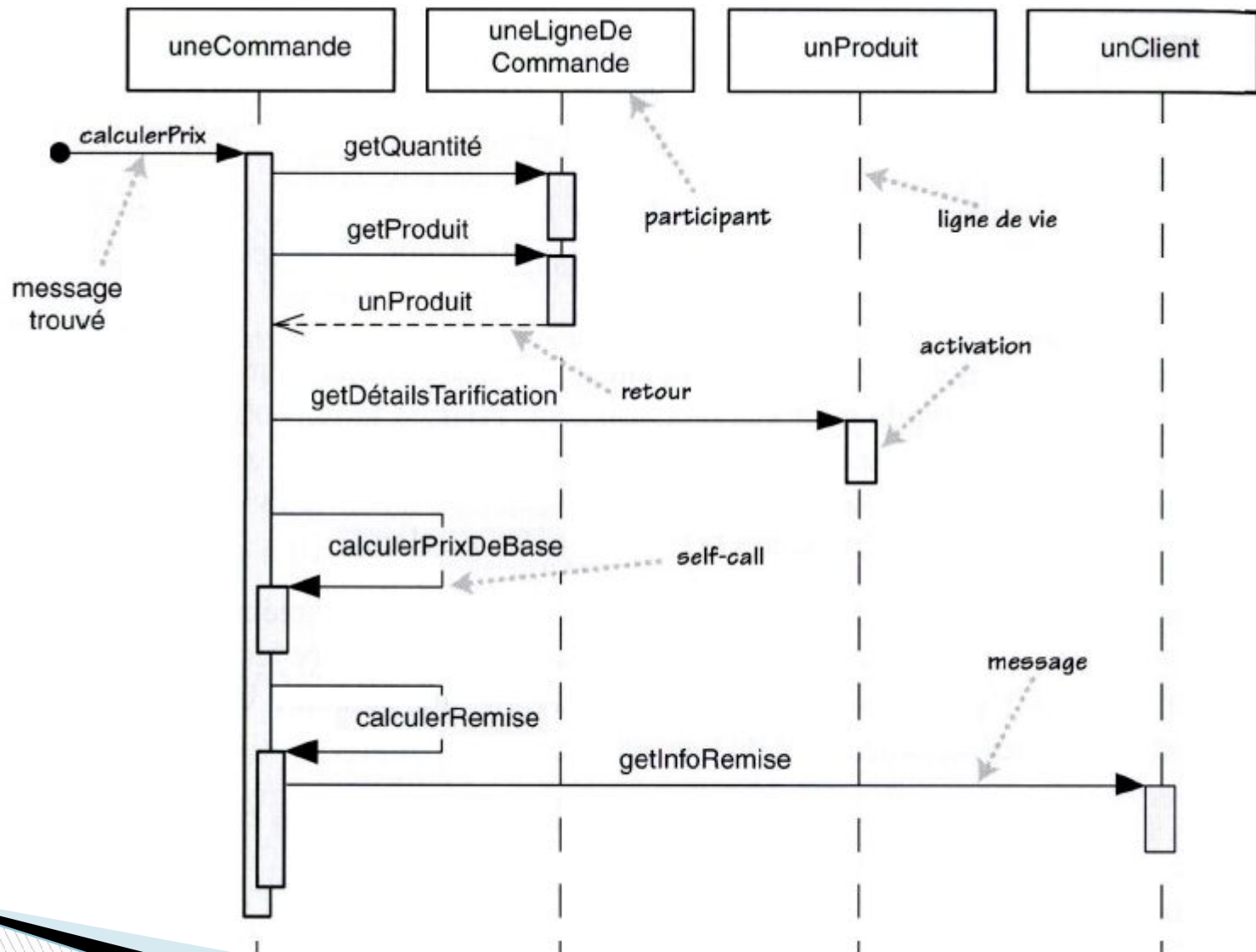


Diagramme de séquence – Exemple

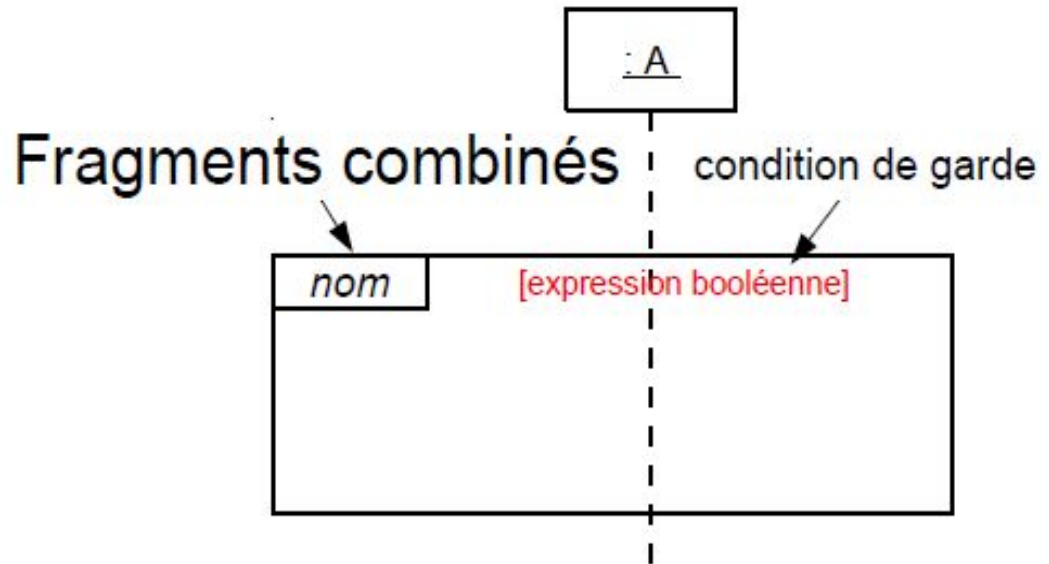


Fragments combinés

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

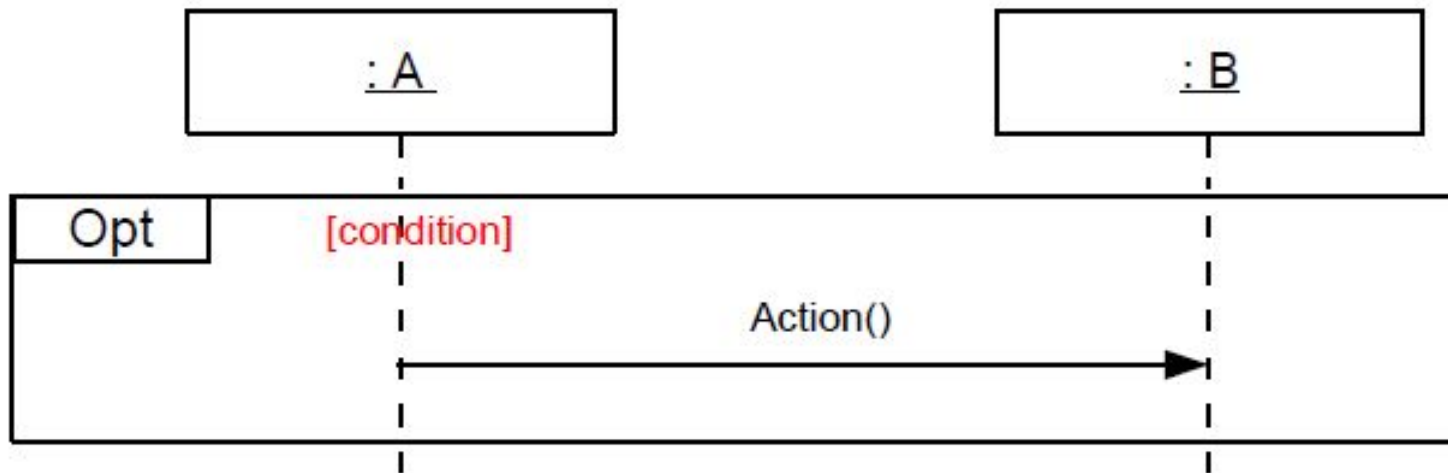
▣ *Types*

1. Opt
2. Loop
3. Alt
4. Break
5. Critique
6. Ref
7. ...



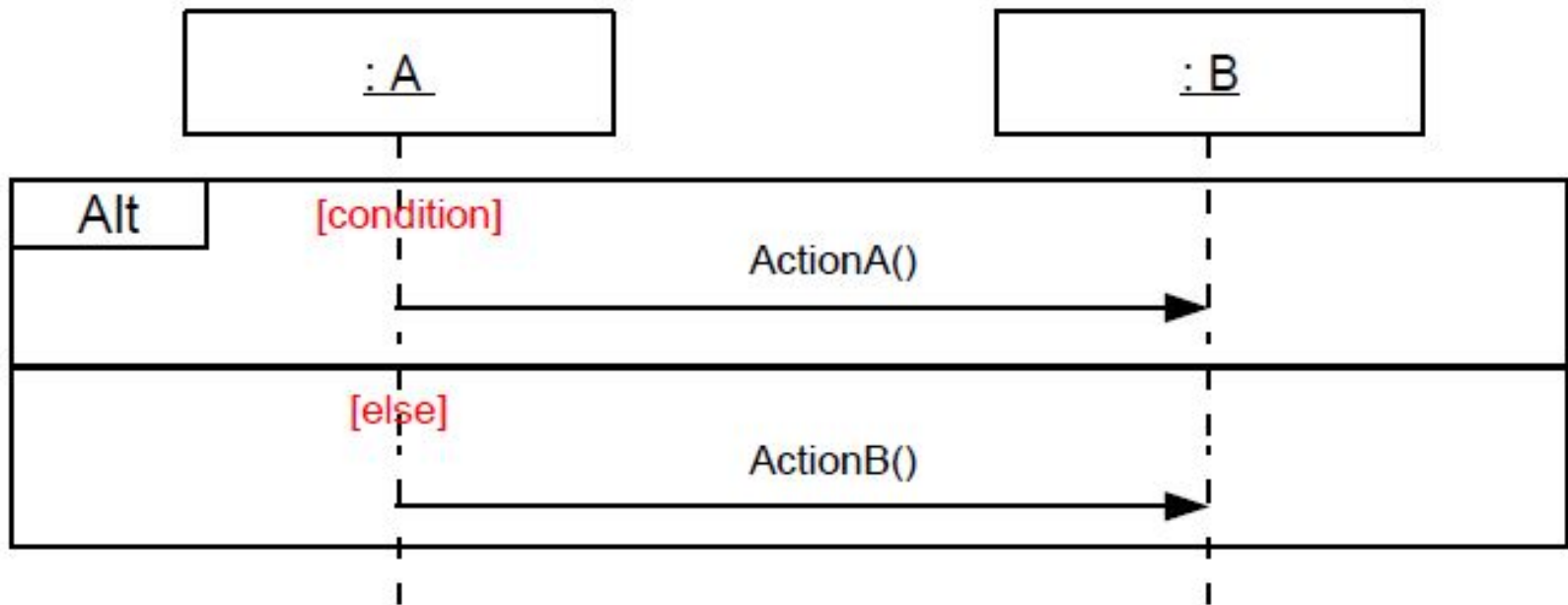
Fragments combinés

- ❑ **Opt** : Fragment parcouru si une condition est vérifiée

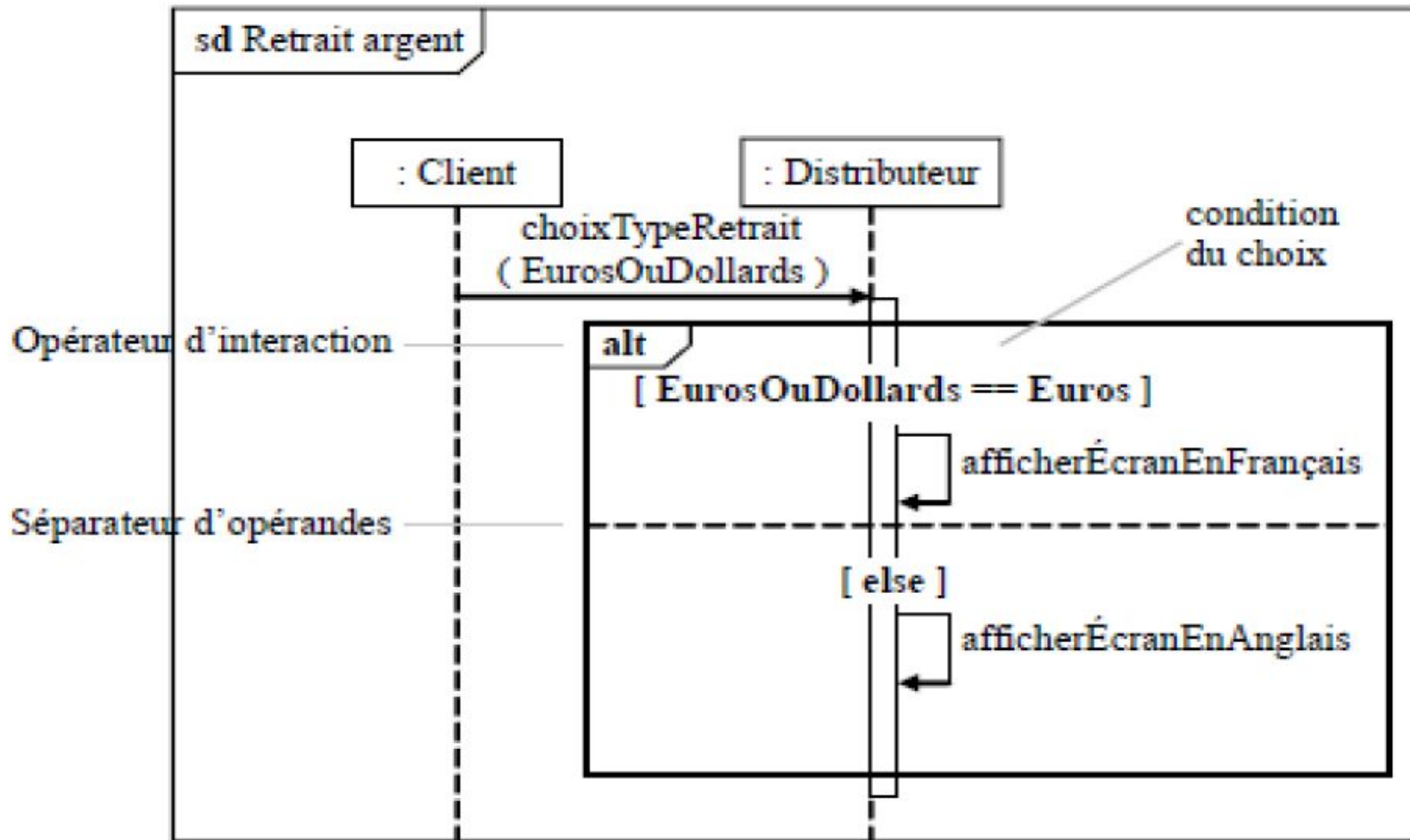


Fragments combinés

- Alt : Equivalent à la structure de contrôle "si .. alors .. sinon".

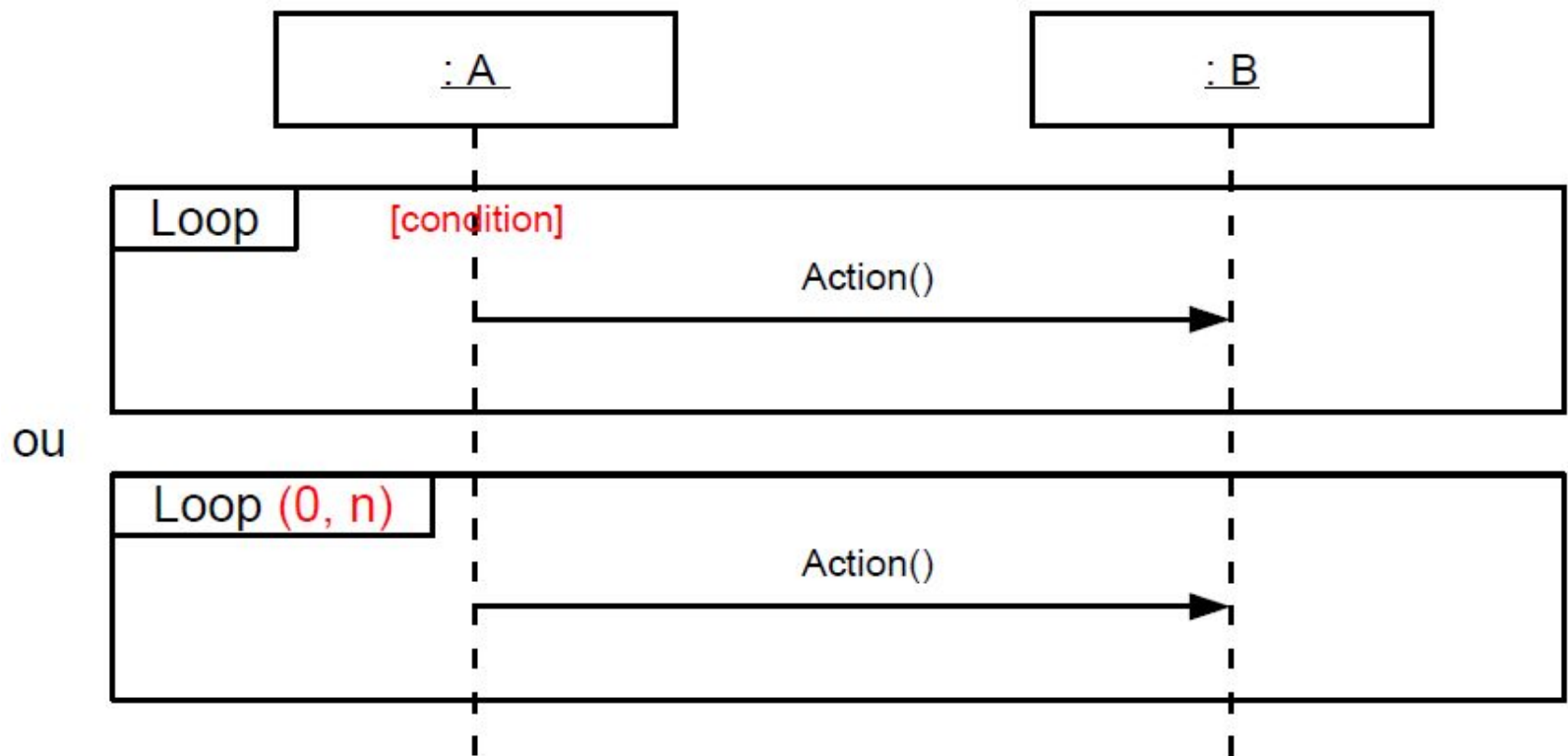


Fragments combinés - Exemple

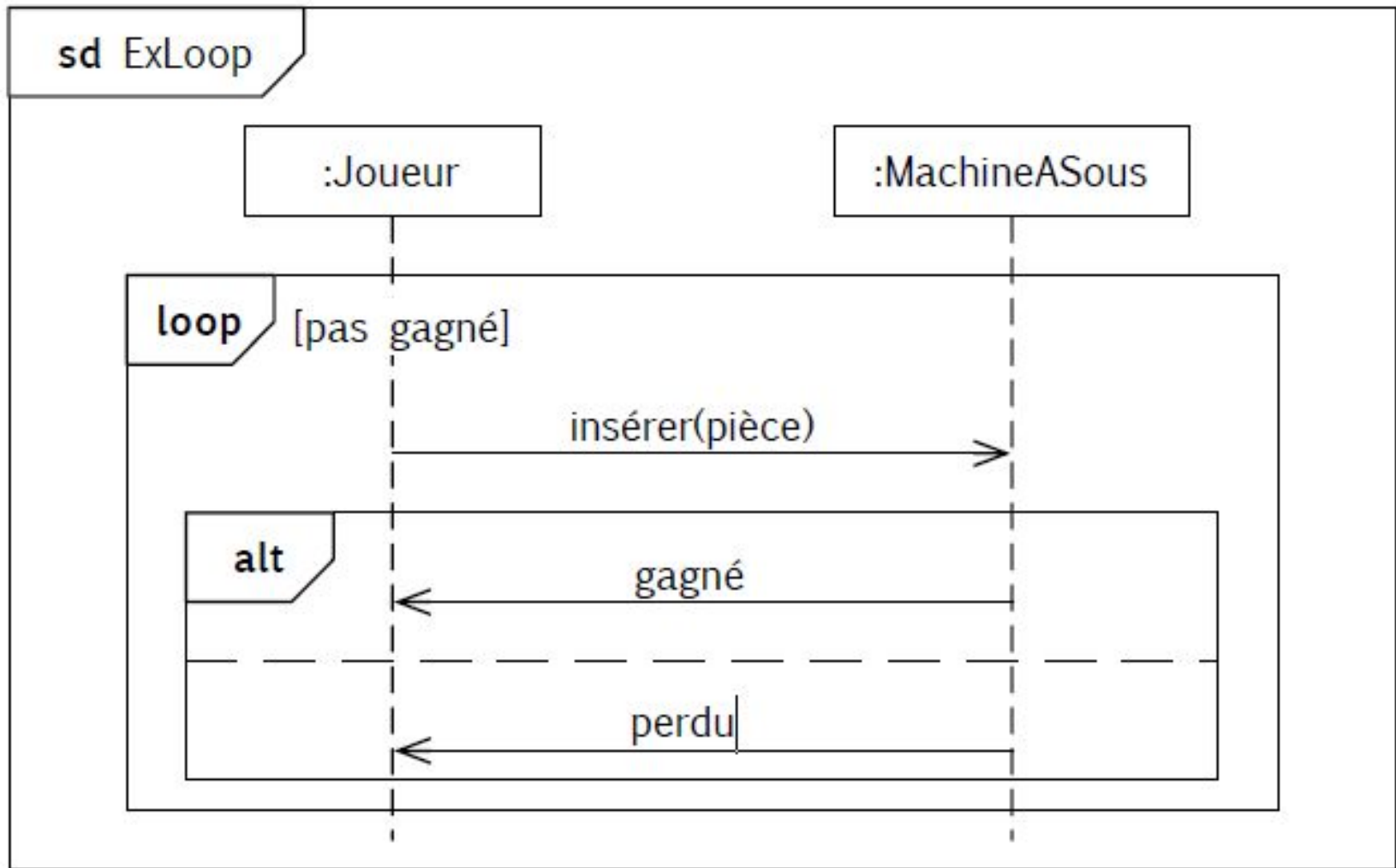


Fragments combinés

- **Loop** : Répétition du fragment tant que la condition est vérifiée

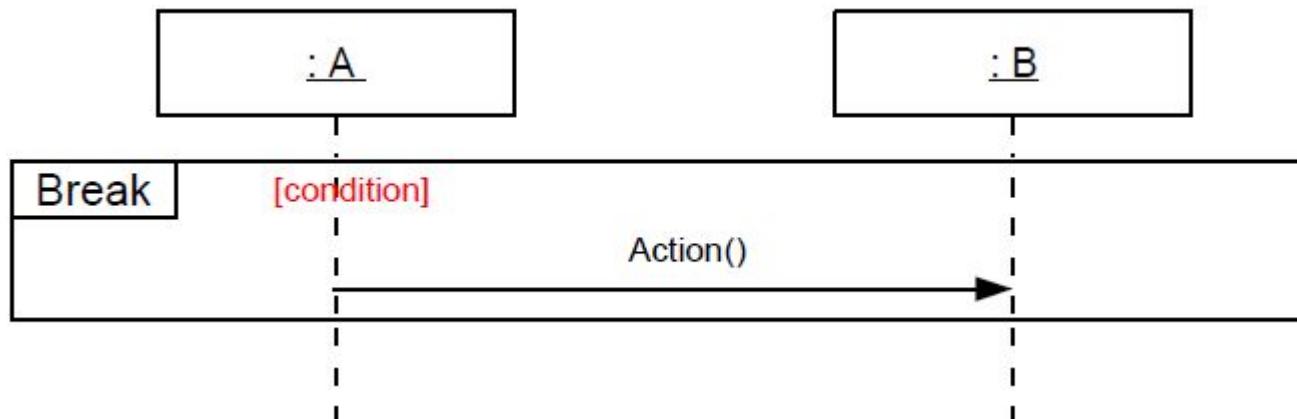


Fragments combinés - Exemple



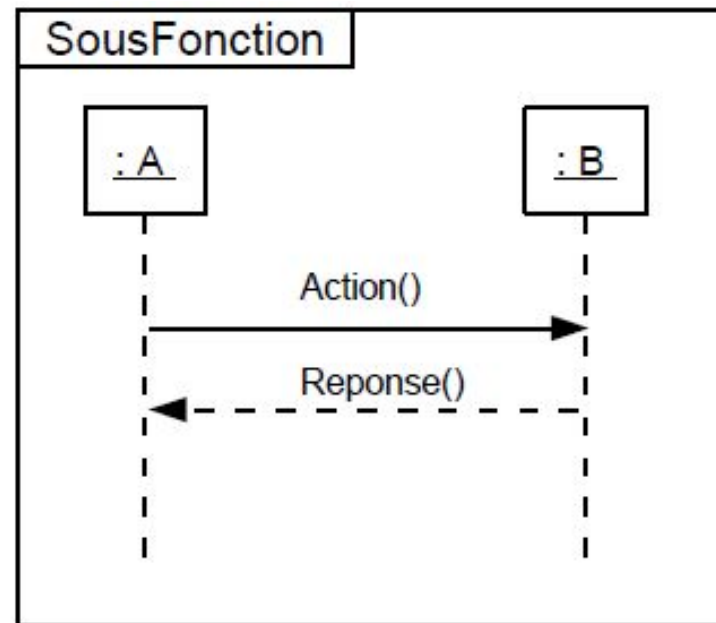
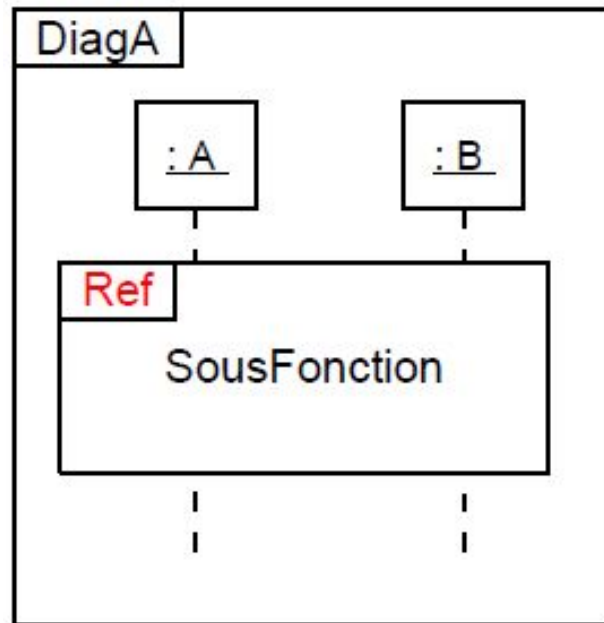
Fragments combinés

- **Break** : Fragment exécuté et met fin au fragment englobant



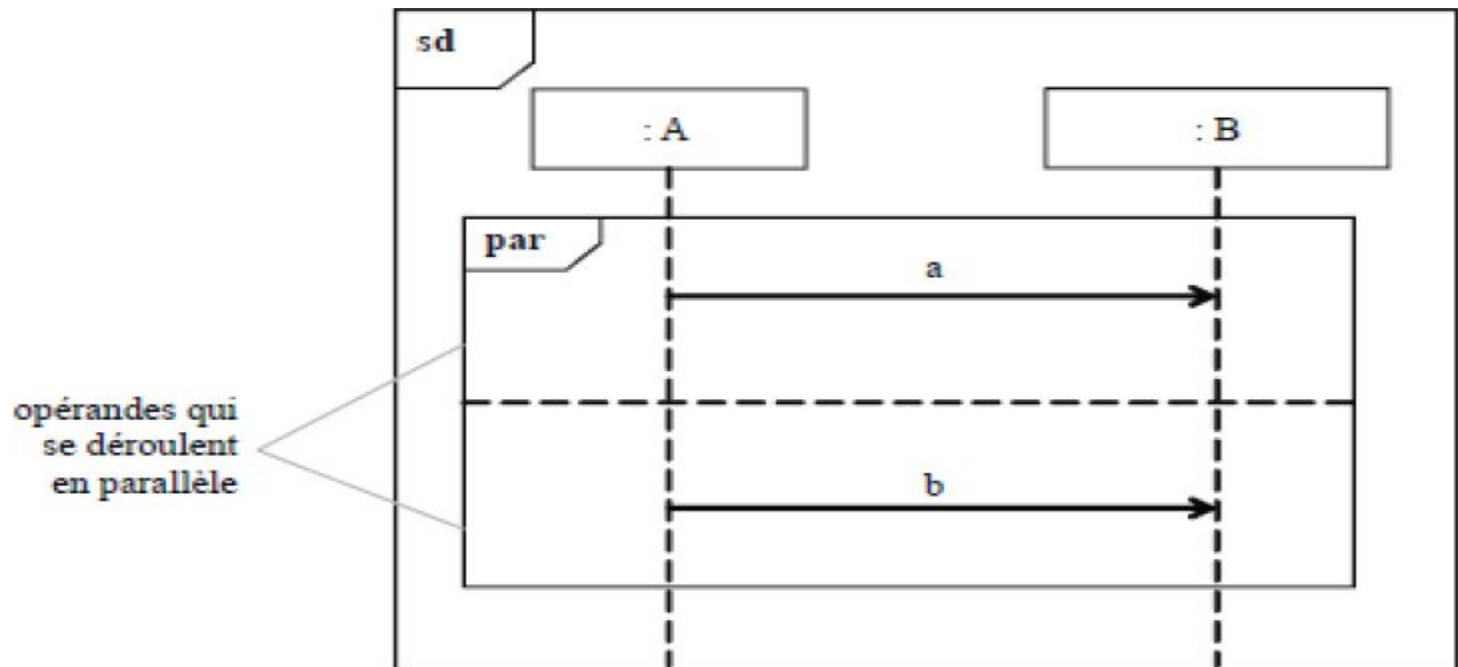
Fragments combinés

- ❑ **Décomposition** : mot clef "ref"



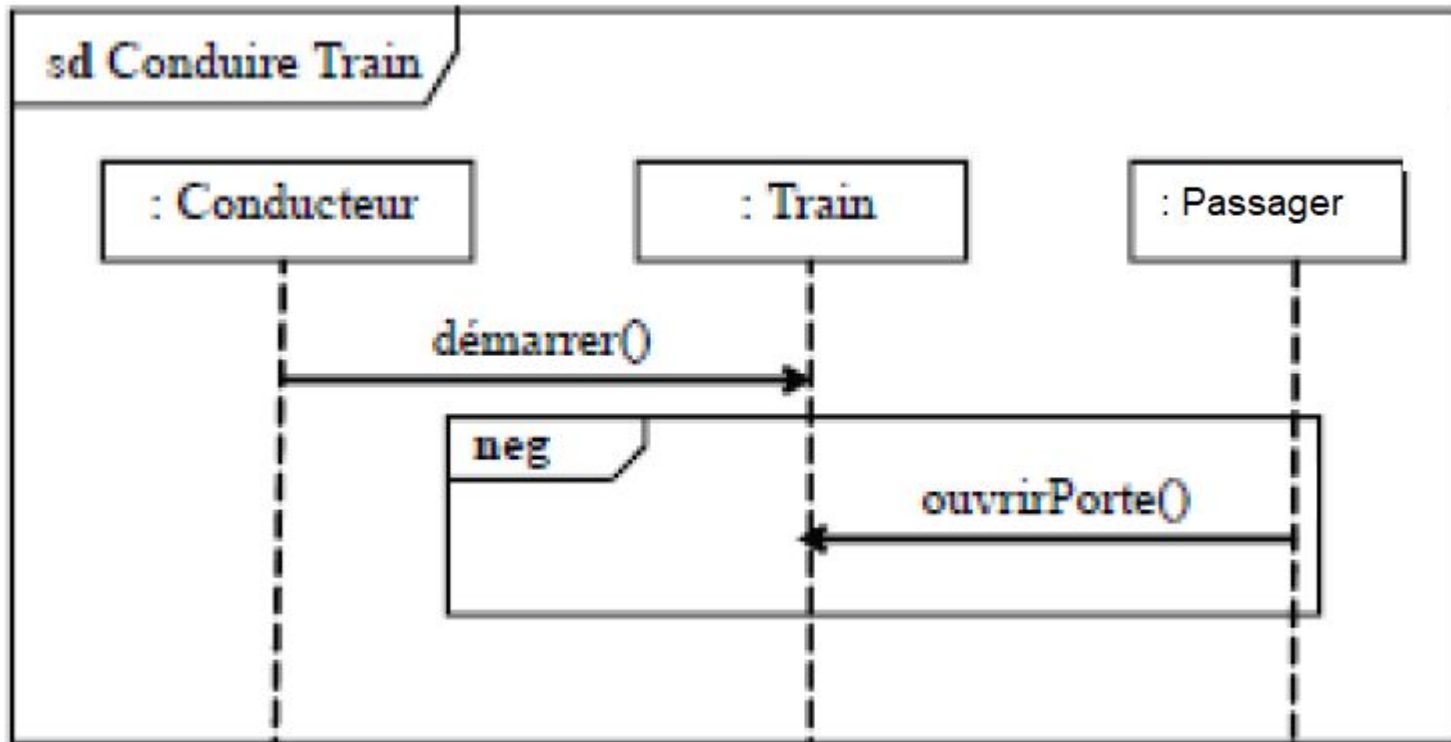
Fragments combinés

- ❑ **Par** : Fragment pour modéliser des opérations qui se déroulent en parallèle.

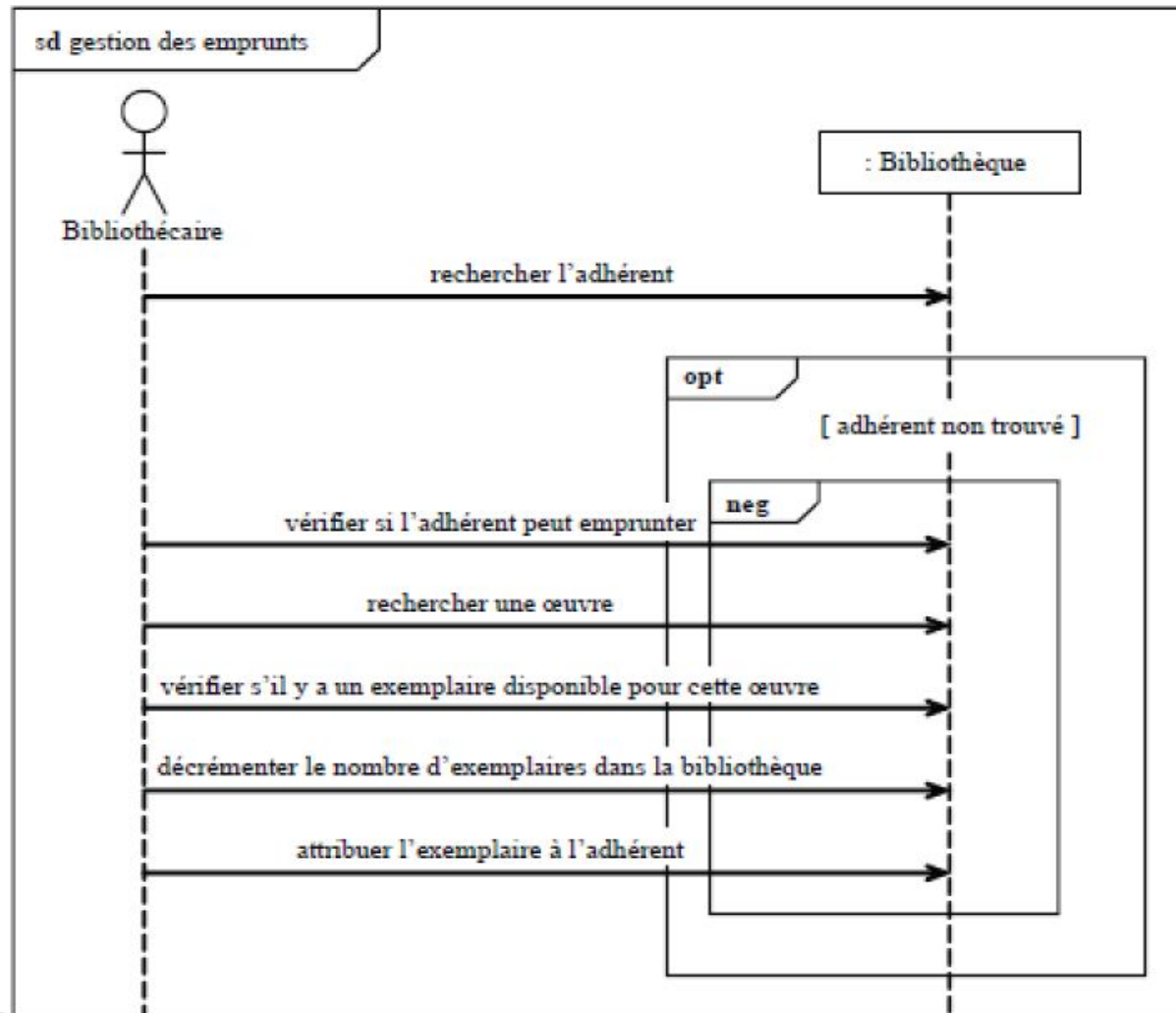


Fragments combinés

- ❑ **Neg** : pour interdire l'exécution de certaines opérations.

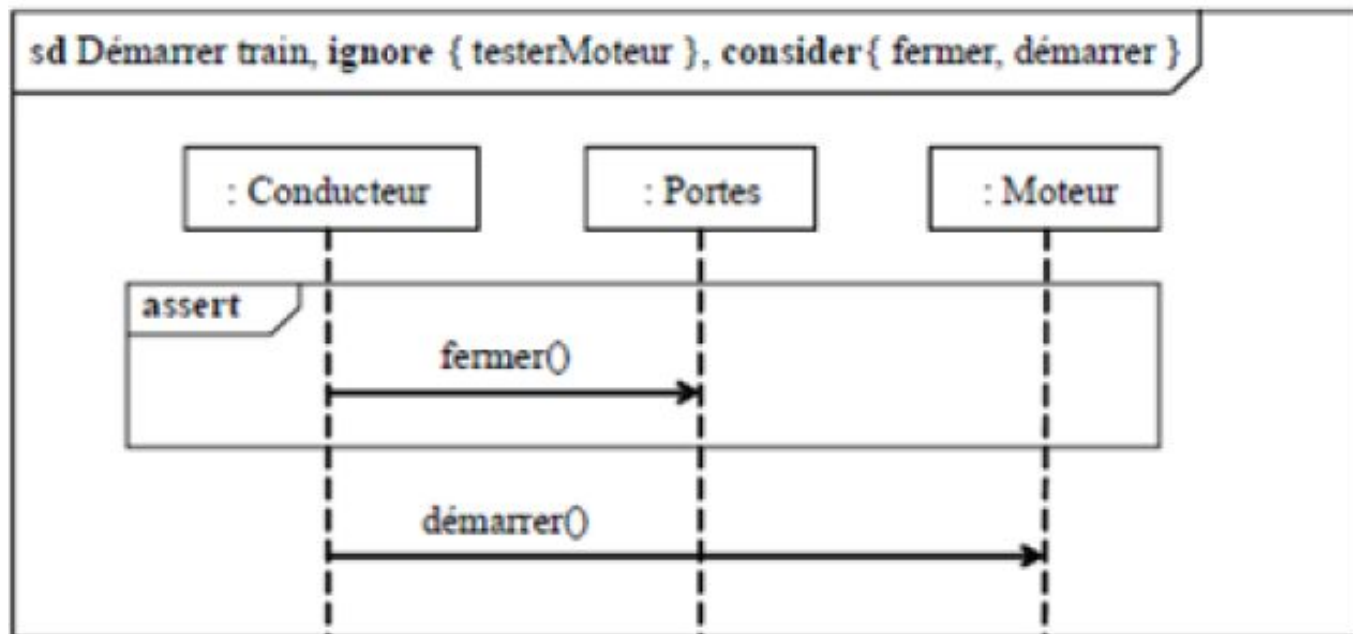


Fragments combinés – Exemple « Alt » et « Neg »

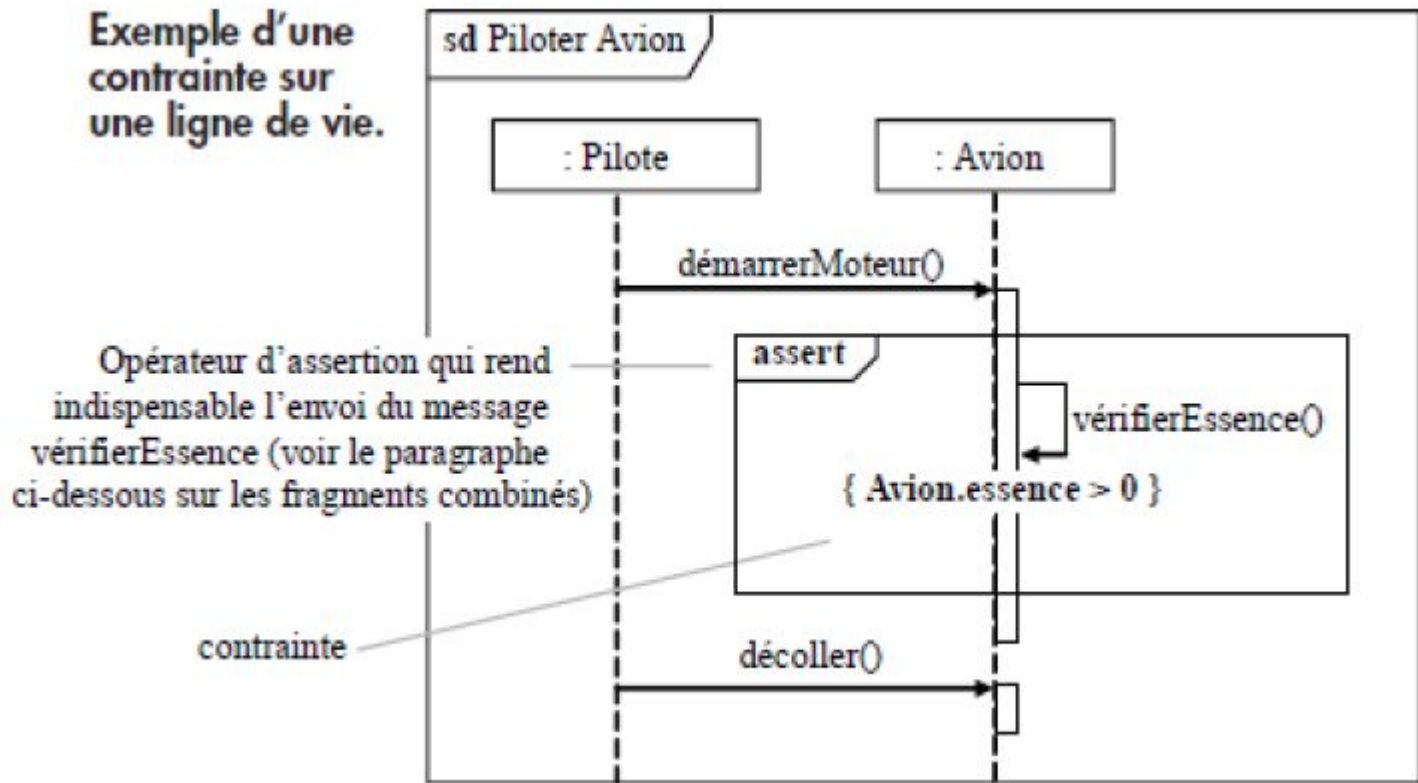


Fragments combinés

- ❑ **Assert, Ignore, Consider** : focalisent l'attention sur une partie des messages envoyés car certains messages sont importants, d'autres moins.



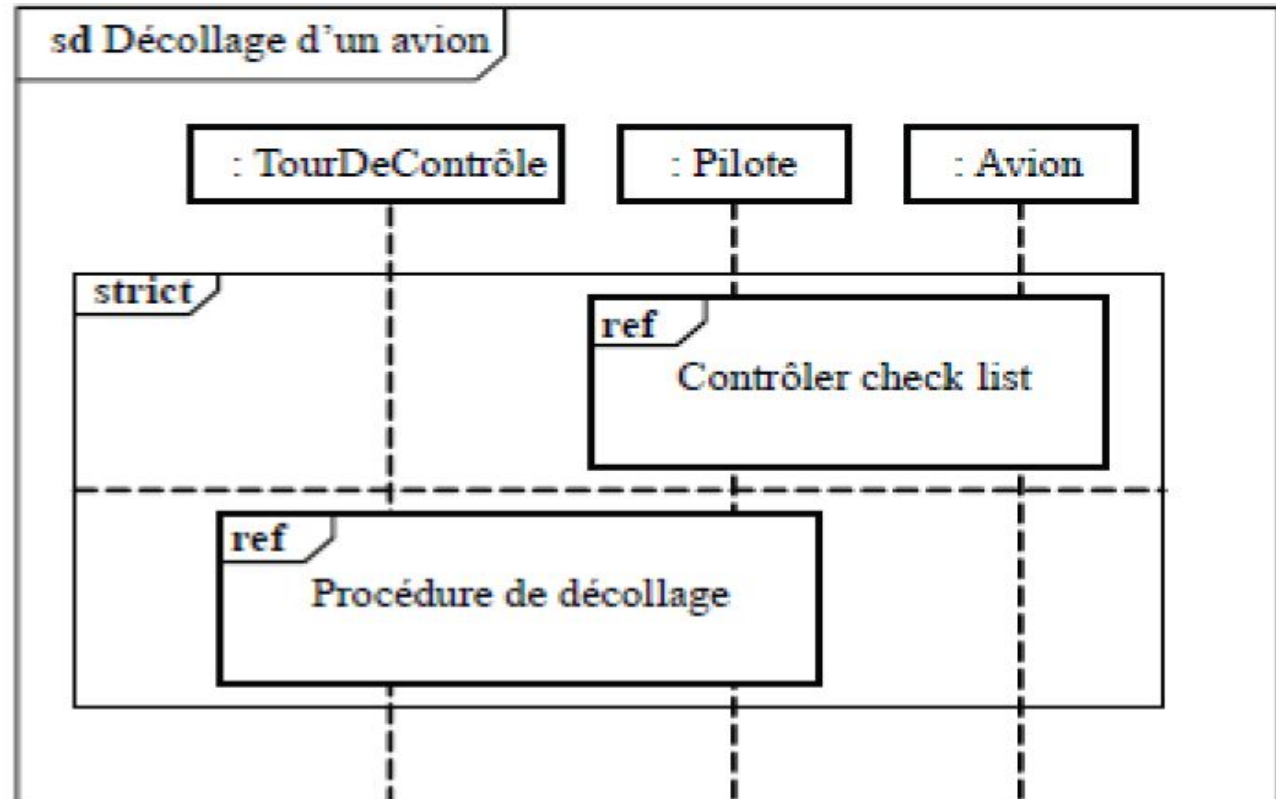
Fragments combinés – Exemple : Contrainte sur une ligne de vie



Fragments combinés

□ **Strict** : sequencing

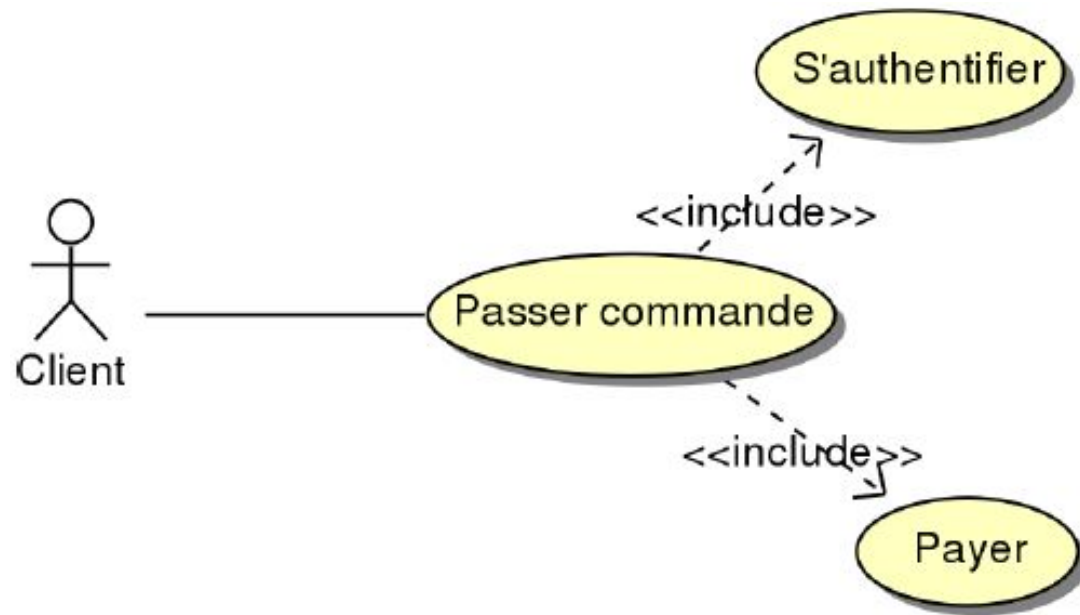
Déroulement des
opérandes d'un
opérateur dans
un ordre strict.



□ **Weak**: sequencing

Utiliser d'un DS pour modéliser un cas d'utilisation

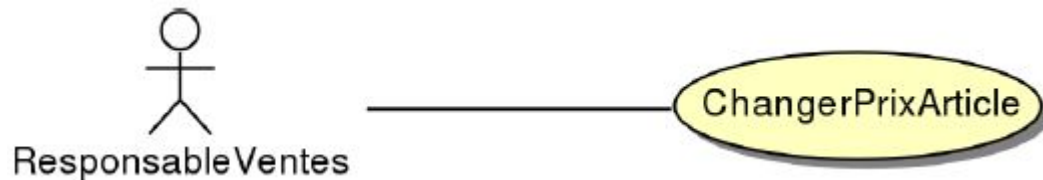
- Chaque cas d'utilisation donne lieu à un diagramme de séquences



- Les inclusions et les extensions sont des cas typiques d'utilisation de la réutilisation par référencement

Diagramme de séquence – Exemple d'interaction

- Cas d'utilisation :



- Diagramme de séquences correspondant :

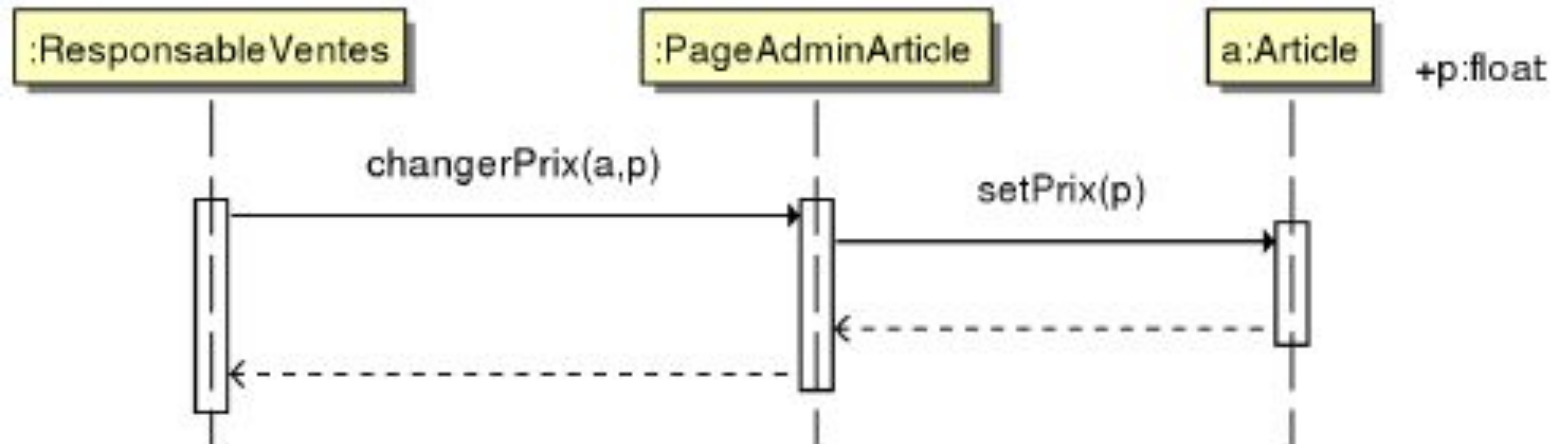
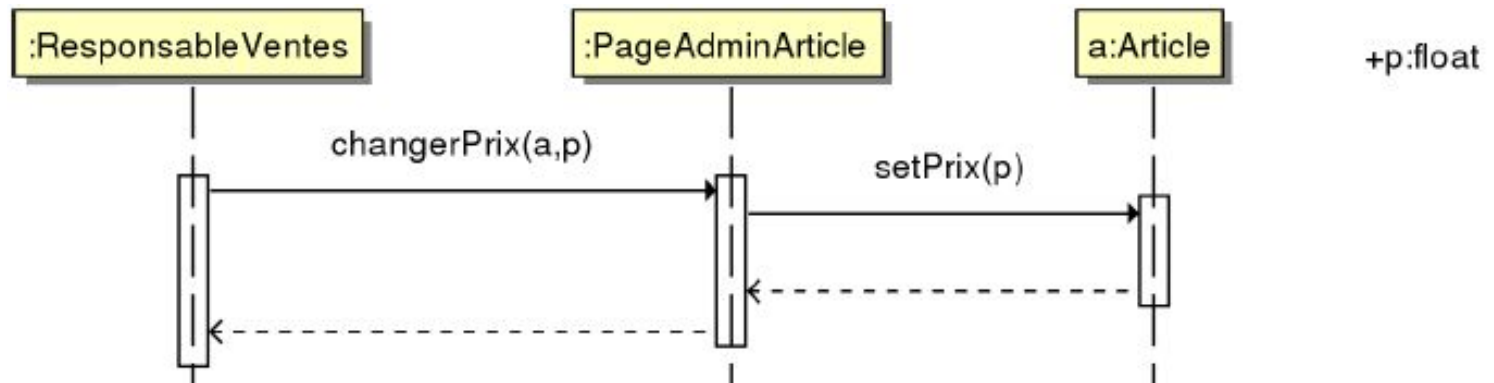


Diagramme de séquence – Exemple d'interaction

- Diagramme de séquences correspondant :



- Opérations nécessaires dans le diagramme de classe:

