Diagramme de séquence

Dame Samb
UCAD/FST/DMI

Introduction

- Le DCL/DOB montre la structure statique du système en terme de classes/objets et de relations entre elles.
- Cependant, cette description ne montre pas comment ces classes/objets interagissent ensemble pour réaliser des tâches et fournir les fonctionnalités du système.
- Les objets communiquent entre eux en envoyant ou en recevant des messages.
- La communication effectuée entre un ensemble d'objets dans l'objectif de réaliser une tâche est **une interaction** qui peut être décrite dans un **diagramme de séquence**.

Diagramme de séquence

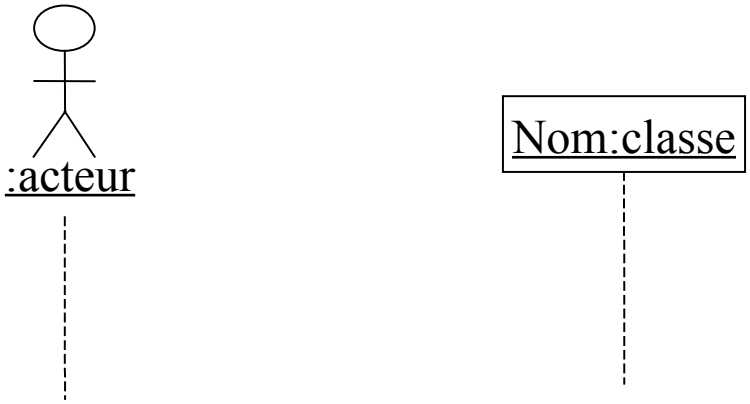
- **Diagramme de séquence:**
  - représente les interactions entre les objets qui composent le système.
  - se concentre sur la séquence (ordre) des messages échangés entre les objets.
- Le diagramme de séquence permet de faire apparaître :
  - les objets intervenant dans l'interaction (acteurs ou objets appartenant au système)
  - la description des interactions (messages) entre les intervenants

Diagramme de séquence

- Le diagramme de séquence est utilisé de deux manières différentes:
  - Documentation des cas d'utilisation :
    - description des interactions en des termes proches de l'usager,
      - L'indication portée sur une flèche correspond à un événement (une action) qui survient dans le domaine de l'application
      - Les flèches ne traduisent pas à ce niveau des envois de messages au sens des langages de programmation. On est encore dans le modèle logique
  - Représentation précise des interactions "informatiques"
    - Interactions entre objets
    - Le séquencement des flots de contrôle :
    - Exemple: Représentation des structures de contrôle (IF THEN ELSE) et de la structure de la boucle WHILE

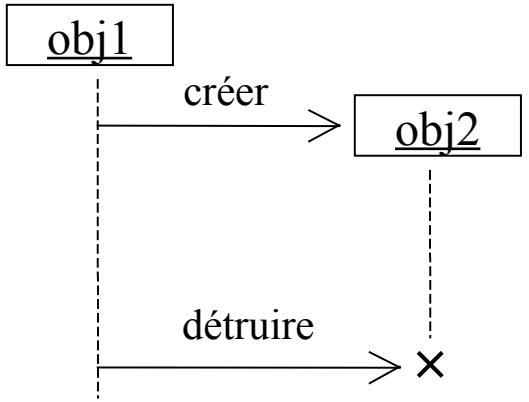
Interactions

- Une interaction se traduit par un envoi de messages entre objets.
- Les objets interagissant sont des instances jouant des rôles
  - instances d'acteurs pour la documentation de cas d'utilisation
  - instances de classes pour la représentation des interactions "informatiques".
- Dans UML, les objets sont représentés comme suit:



Interactions

- Dans un DS, les objets sont associés à une ligne verticale, appelée **ligne de vie**.
- La ligne de vie représente la période de temps durant laquelle l'objet "existe".
  - **Création d'un objet**: un message pointe sur le symbole de l'objet.
  - **Destruction d'un objet**: sa ligne de vie se termine par une croix en trait épais (×).

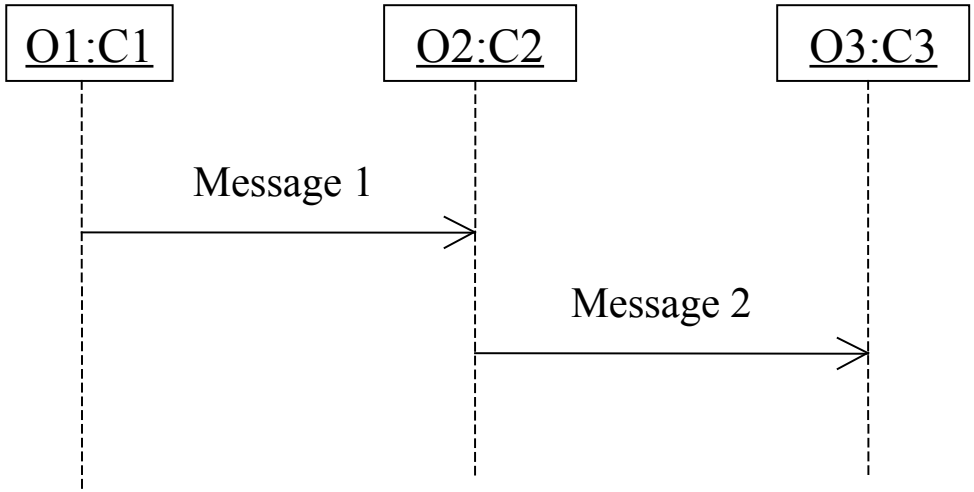


Interactions

- Les objets communiquent en échangeant des messages représentés au moyen de flèches horizontales, orientées de l'émetteur du message vers le destinataire.
- La dimension verticale représente l'écoulement du temps.
- Les messages sont étiquetés par le nom de l'opération ou du signal invoqué.
- L'ordre d'envoi des messages est donné par la position de ces messages sur les lignes de vie des objets.

Interactions

- Exemple:



- Message 1 précède dans le temps Message 2

Les messages

- Un *message* est la spécification d'une communication entre objets avec les informations nécessaires pour qu'une activité s'ensuive
- Il est représenté par une flèche entre l'objet qui envoie le message et l'objet qui le reçoit
- Exemple
- dépiler()
- PresserTouche(2)
- Types de messages : **Appel, Retour, Envoi, Destruction, Création,**

Les messages: Appel

- Un message d'appel invoque une opération sur un objet
- Un objet peut envoyer un message à lui-même pour invoquer une opération locale
- **Exemple**

Message: Retour d'appel

- Un message de retour retourne une valeur à l'appelant
- **Exemple**

Message: Envoi

- Un message d'envoi envoie un signal à un objet
- Un message d'envoi permet d'invoquer une opération d'une manière asynchrone
- Représentation graphique

Message: Création et destruction

- Un message «créer» invoque l'opération de création d'un objet
- **Exemple**

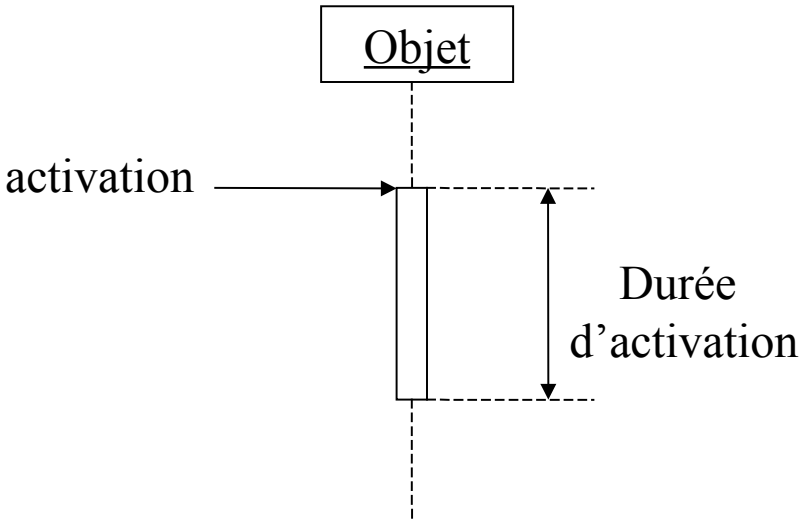
- Un message «détruire» invoque l'opération de destruction d'un objet
- Un objet peut se suicider en s'envoyant un message détruire
- **Exemple**

Les messages

- Une étiquette décrit les messages auxquels elles sont attachées.
  - Syntaxe: ['['garde']'] [itération] [resultat :=]nom message ['('arguments')']
    - **Itération séquentielle** : envoi séquentiel de *n* instances du même message.
      - **Syntaxe** : *[clause d'itération]
    - **Itération parallèle** : envoi parallèle de *n* instances du même message.
      - **Syntaxe** : *|||[clause d'itération]

Activation

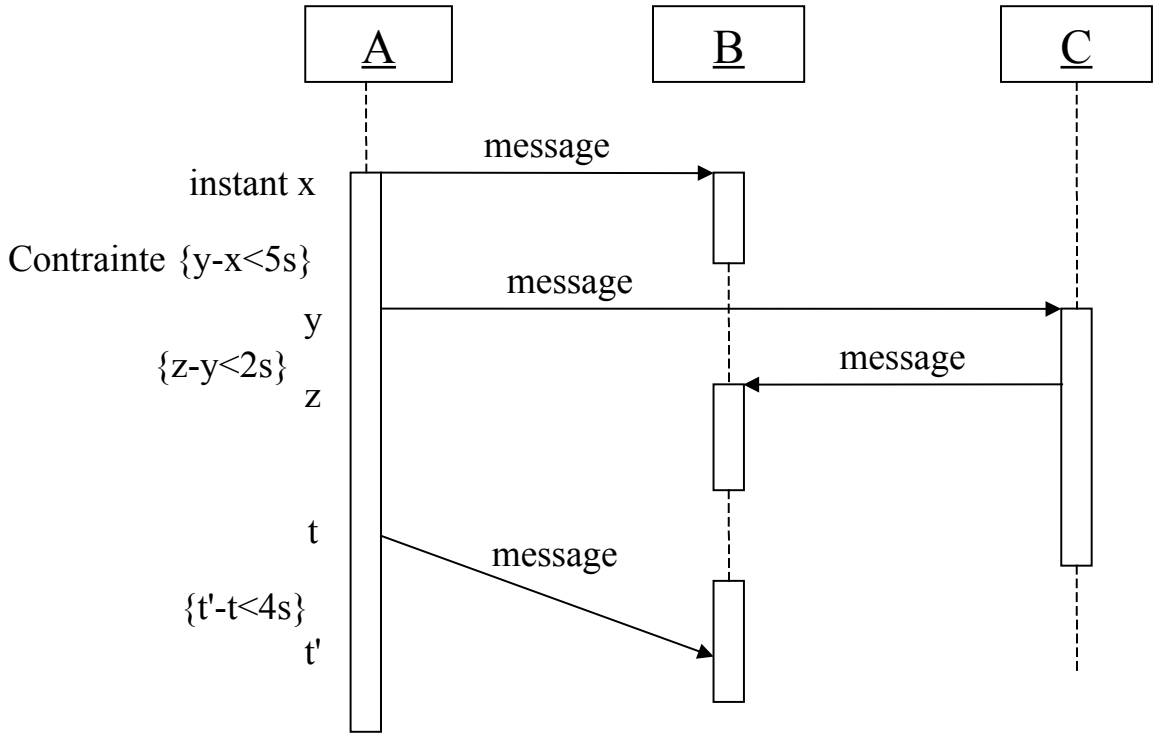
- Le DS permet représenter, en plus des interactions entre les objets, les **périodes d'activité** des objets.
- Une période d'activité correspond au temps pendant lequel un objet effectue une action directe ou indirecte.
- Les périodes d'activité sont représentées par des bandes rectangulaires placées sur les lignes de vie.



Contraintes temporelles

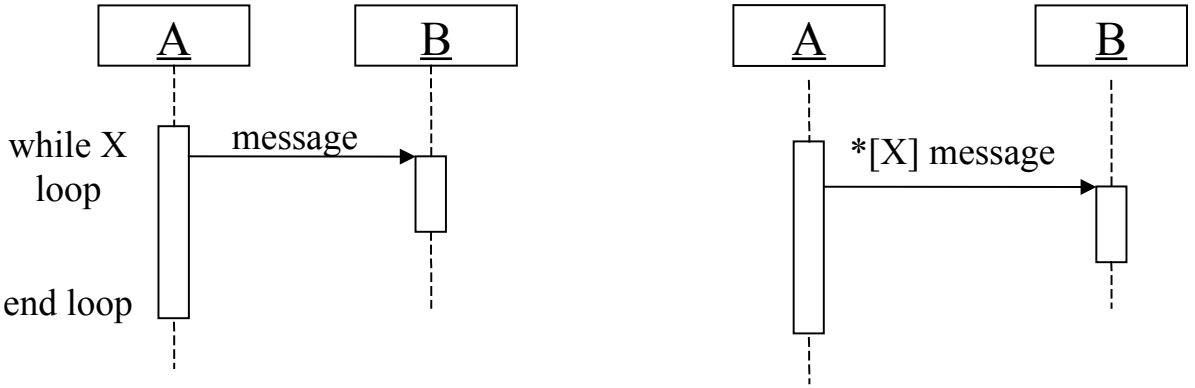
- Pour modéliser les délais de transmission non négligeables par rapport à la dynamique de l'application, on utilise:
  - une flèche oblique,
  - ou des notations temporelles dans la marge.
  - L'instant d'émission d'un message peut être indiqué sur le diagramme à proximité de l'origine de la flèche qui symbolise le message.
  - Cette indication sert de référence pour construire les contraintes temporelles
  - Les instants d'émission et de réception d'un message sont représentés par le couple (*symbole, symbole'*).

Contraintes temporelles



Structures de contrôle

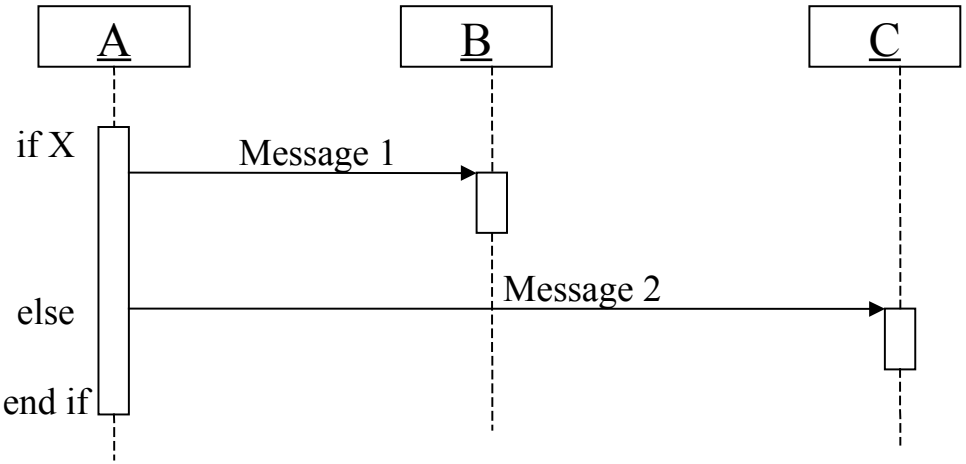
- il est possible de représenter des structures de contrôles itératives par
  - pseudo code (while X loop end loop)
  - condition d'itération (*[X]) sur le message lui-même



L'objet A envoie un message à B tant que la condition X est vraie

Structures de contrôle

- il est possible de représenter des structures de contrôles conditionnelles
  - chez l'expéditeur d'un message :
    - par pseudo code (if X else end if)
    - par garde ([X])
  - chez le destinataire d'un message :
    - par duplication de la ligne de vie



Structures de contrôle

- représentation des structures de contrôles conditionnelles