# Analyse, Conception Objet

# Diagrammes de Séquences

Une partie du matériau de ce cours est issue du cours de S.Galland (Stephane.Galland@emse.fr)

Octobre 2002







## Sommaire







## Définition

- Notation dérivée des “Object Message Sequence Charts” du Siemens Pattern Group.
- Description de l’ordre des interactions entre les objets qui composent le système.
- Représentation se concentrant sur la séquence des interactions d’un point de vue temporel.
- Adaptés à la modélisation des aspects dynamiques des systèmes temps réels et des scénarios complexes mettant en œuvre peu d’objets.
- Les diagrammes de séquences sont des diagrammes d’interaction comme les diagrammes de collaboration.





## Interactions dans le système

- Une interaction se traduit par un envoi de message entre objets.
- Les diagrammes de séquences permettent de faire apparaître :
  - les objets intervenant dans l’interaction (acteurs ou objets appartenant au système)
  - la description de l’interaction (messages)
  - les interactions entre les intervenants (diagramme de séquences)
- Les diagrammes de séquences servent à communiquer autant pour les usagers que pour les développeurs

## Utilisation des diagrammes de séquences

- Documentation des cas d'utilisation :
  - description des interactions en des termes proches de l'usager,
    ⇝ les étiquettes des messages correspondent à des évènements se produisant dans le système.
- Représentation des interactions "informatiques" et répartition des flots de contrôle :
  - le concept de message unifie les formes de communication entre objets (appel de procédure, évènement discret, signal, ...)





## Objets

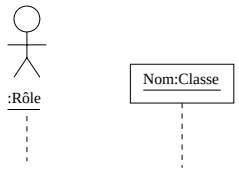
- Les objets sont des entités appartenant au système (instance d'une classe) ou se trouvant à ses limites (acteurs)
- Ils représentent :
  - soit des concepts abstraits, soit des acteurs (documentation de cas d'utilisation)
  - soit des objets d'implantation (diagrammes de séquences pour les interactions "informatiques")
- Ils sont identifiés par l'intermédiaire des cas d'utilisation ou des diagrammes de classe.





## Objets : représentation

- Dans UML, les objets sont représentés comme suit :




- Le nom de l'objet est composé de son rôle (`rôle` ou `nom`) et/ou du nom de la classe instanciée (`classe`).
- Le nom est souligné pour indiquer qu'il s'agit d'une instance.

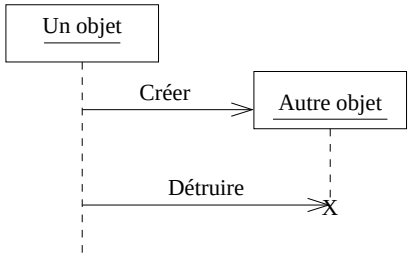




## Ligne de vie des objets

- Elle est représentée par une ligne verticale en dessous des objets.
- Elle 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 (×).

## Ligne de vie des objets (suite)



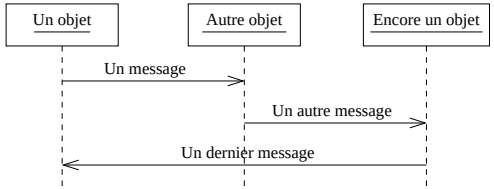







## Messages

- Les objets communiquent en échangeant des messages représentés sous forme de flèches.
- L'ordonnancement horizontal des messages n'a aucune signification.
- La dimension verticale représente l'écoulement du temps.
- Les messages sont étiquettés par le nom de l'opération ou du signal invoqué.

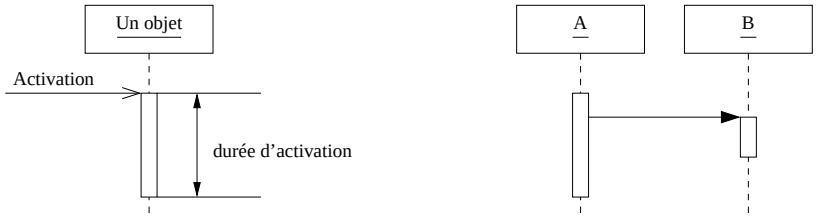







## Messages : activation des objets

- Une période d'activité correspond au temps pendant lequel un objet effectue une action directe ou indirecte.
- Représentation : bande verticale le long de la ligne de vie de l'objet.





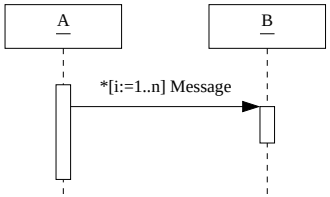



## Messages : étiquettes

- Les étiquettes décrivent les messages auxquels elles sont attachées.
- Syntaxe générale:
  `['['garde']'] [itération] [résultat :=] nom_message ['('arguments')']`
- `nom_message` : nom de l'opération ou du signal invoqué par l'intermédiaire de ce signal,
- `garde` : condition booléenne et optionnelle (représentée entre crochets) autorisant ou non l'envoi d'un message (utilisation d'OCL).

## Étiquettes des messages : itération

- 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 ]








## Étiquettes des messages : arguments

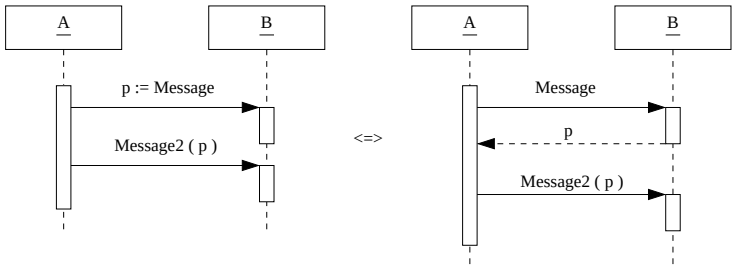
- Liste des paramètres du message, séparés par des virgules.
- Les arguments et le nom de l'action déterminent sans ambiguïté l'action à réaliser.
- Les arguments peuvent contenir des valeurs retournées par des messages envoyés précédemment.
- Exemples :
  `Afficher ( x, y )` – affiche les valeurs x et y
  `Soustraire( Aujourd'hui, DateDeNaissance )` – calculer le nombre de jours entre deux dates





## Étiquettes des messages : résultat

- Le résultat est constitué d'une liste de valeurs retournées par le message.
- Ces valeurs peuvent être utilisées comme paramètres des autres messages.

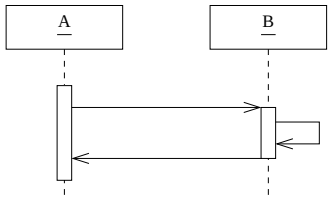






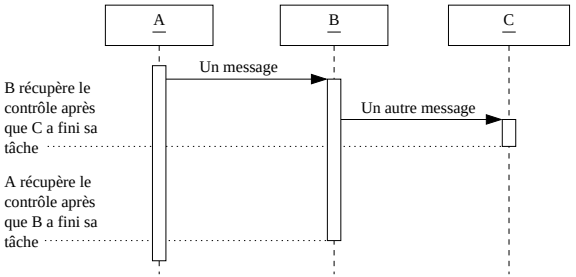

## Messages : flot de contrôle à plat

- Catégorie de messages utilisée pour indiquer la progression vers une prochaine étape d'une séquence.
- Tous les messages de cette catégorie sont asynchrones.
- Messages représentés par une seule flèche.
- Cas particulier : dans le cas de systèmes concurrents, une demi-flèche indique l'envoi d'un message, et une flèche un message avec attente de prise en compte.



## Messages : appel de procédure

- Dans un appel de procédure (flot de contrôle emboîté), la séquence emboîtée doit se terminer pour que la séquence englobante reprenne le contrôle.
- Les appels de procédure sont représentés par des flèches à pointe triangulaire.

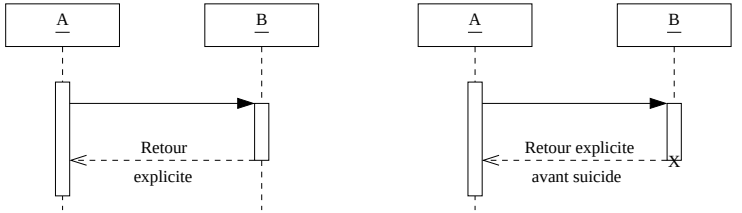







## Messages : retour explicite

- Dans le cas d'un système coucurrent, il est utile d'expliciter la fin de l'exécution de sous-procédures.
- On utilise une flèche pointillé (déjà utilisée dans le cadre des valeurs retournées).

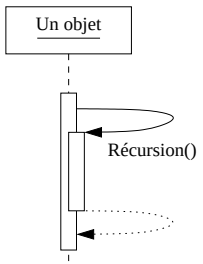







## Messages : appel récursif

- L'envoi de messages récursifs se représente par un dédoublement de la bande d'activation.
- L'objet apparaît alors comme s'il était actif plusieurs fois.

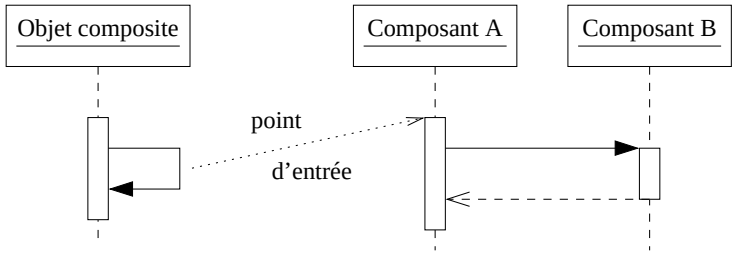






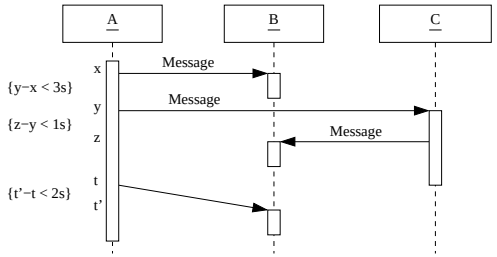

## Messages : réflexivité

- Un objet peut s'envoyer un message.
- Cette construction peut indiquer un point d'entrée dans une activité de plus bas niveau.



## Contraintes temporelles

- Pour modéliser les délais de transmission non négligeables, on utilise les deux notations suivantes :
  - une flèche oblique,
  - ou des notations temporelles dans la marge.
- Les instants d'émission et de réception d'un message sont représentés par le couple $\langle$`symbole`,`symbole`$'\rangle$.

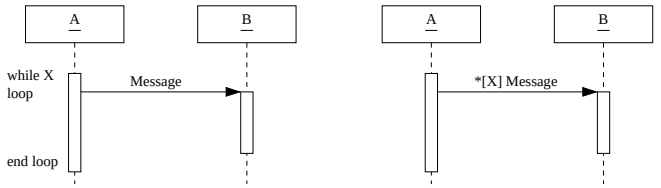







## Structures de contrôle : boucles

- Modélisation des structures de contrôles itératives :
  - par pseudo-code (`while X loop end loop`)
  - par condition d'itération (`*[X]`) sur le message lui même.







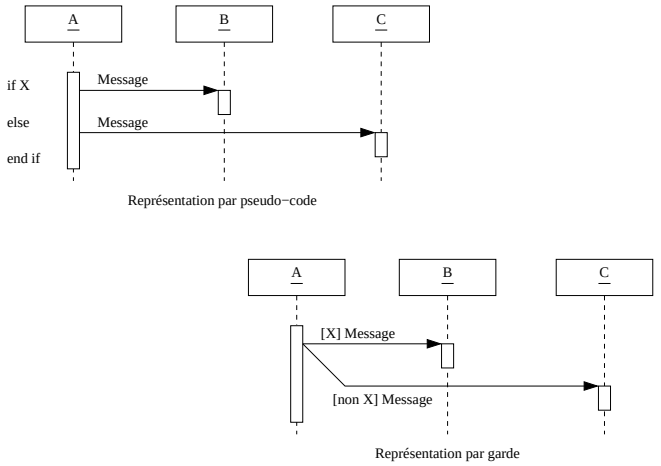

## Structures de contrôle : conditions

- Modélisation 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

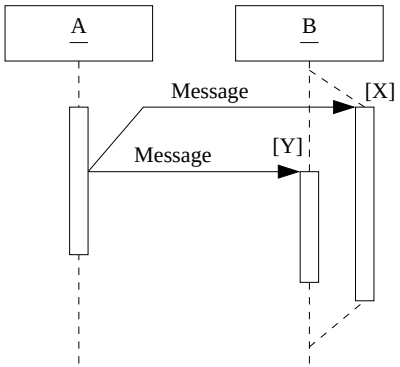




## Structures de contrôle : conditions (suite)



## Structures de contrôle : conditions (suite)




Représentation par dédoublement de la ligne de vie