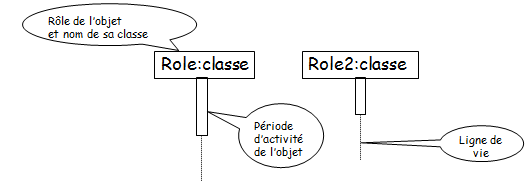
**FICHE 1 : DIAGRAMME DE SEQUENCE**

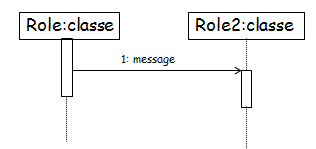
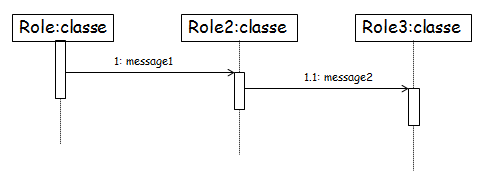
**Introduction**

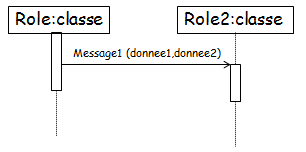
* Le diagramme de séquence se focalise sur les aspects temporels. Il décrit la dynamique du système (c’est-à-dire l’évolution dans le temps). L’évolution globale sera représentée par un ensemble de diagramme de séquence. Pour interagir entre eux les objets s’envoient des messages. Lors de la réception d’un message, un objet devient actif et exécute la méthode de même nom. Un envoi de message est donc un appel de méthode. Il décrit donc les interactions entre un groupe d’objets en montrant, de façon séquentielle, les envois de message qui interviennent entre objets.

**La ligne de vie :**

* Le diagramme de séquence fait intervenir les instances de classes intervenant dans la réalisation de la sous-fonction qui lui est liée.
* A chaque instance est associée une ligne de vie qui montre ses actions et réactions, ainsi que les périodes pendant lesquelles elle est active, c’est-à-dire où elle exécute l’une de ses méthodes.
* Un diagramme de séquence contient plusieurs lignes de vie car il traite des interactions entre plusieurs objets.

**L’envoi des messages :**

* Les envois de message sont représentés par des flèches horizontales reliant la ligne de vie de l’objet émetteur à la ligne de vie de l’objet destinataire.
* L’objet de gauche envoie un message à l’objet de droite. Ce message donne lieu à l’exécution de la méthode message de l’objet de droite, ce qui provoque son activation.
* Les messages sont numérotés séquentiellement, à partir de un. Si un message est envoyé alors que le traitement du précédent n’est pas terminé, il est possible d’utiliser une numérotation composée où l’envoi du message numéro 2 intervient pendant l’exécution du message 1.



* La transmission d’information est aussi possible. Elle est représentée par des paramètres transmis avec le message.

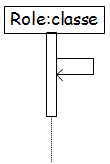
**Les différents types de messages.**

* Le message asynchrone est le plus souvent utilisé. Son emploi signifie que l’expéditeur du message attend que l’activation de la méthode invoquée chez le destinataire soit terminée avant de continuer son activité.
* Dans le cas du message asynchrone, l’expéditeur n’attend pas la fin de

l’activation de la méthode invoquée chez le destinataire. Ceci se produit lors de la modélisation d’un système où les objets peuvent fonctionner en parallèle. En UML 1 dans la représentation d’un message asynchrone une demi flèche

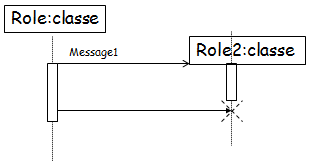
supérieure était utilisée.

* Le message de retour de l’invocation d’une méthode n’est pas systématique. En effet toutes les méthodes ne retournent pas un résultat.

**Remarque :**

* Un objet peut s’envoyer un message à lui-même (cas réflexif).

**La création et la destruction d’objets**

* Puisque le diagramme de séquence décrivant la dynamique d’un

système, cela demande de fréquentes créations et destruction d’objet.

* La création d’un objet est représentée par un message spécifique qui

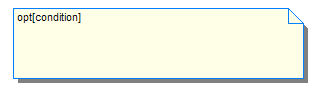
donne lieu au début de la ligne de vie du nouvel objet.

* La destruction d’objet est un message envoyé à un objet existant et qui

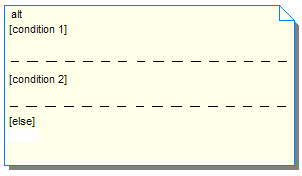
donne lieu à la fin de sa ligne de vie. Il est représenté par une croix.Les

**Les cadres d’interactions**

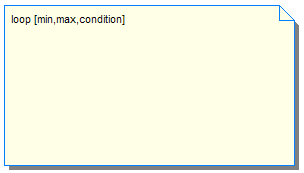
Jusqu’à présent nous avons uniquement visualisé la schématisation de scénario sans possibilité de traduire les alternatives et les boucles. Nous étions dans une représentation UML1. UML2 permet la représentation dynamique des cas d’utilisations grâce à l’introduction de la notion de cadre d’interactions.

Un cadre d’interaction est une partie du diagramme de séquence associé à une étiquette.

* *L’alternative :* elle s’obtient en utilisant l’opérateur **opt** suivi d’une condition de test. Si la condition est vérifiée, le contenu du cadre est exécuté.



* Il existe un autre opérateur pour l’alternative. Il permet de faire plusieurs conditions de test. Il se nomme **alt** et est suivi de plusieurs conditions de test puis du mot clé **else**. Le cadre est alors scindé en plusieurs parties dont le contenu n’est exécuté que si la condition associée est remplie.



* La boucle : elle est réalisée par l’opérateur **loop** suivi des paramètres **min,max** et d’une **condition** de test. Le contenu du cadre est exécuter min fois, puis tant que la condition de test est vérifiée et tant que le nombre maximal d’exécutions de la boucle ne dépasse pas max.

Chaque paramètre est optionnel.