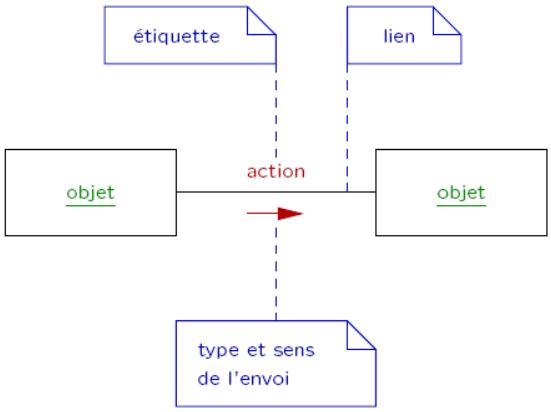
Chapitre 1 : Diagrammes UML

www.Exoco-LMD.com

1. Les diagrammes de collaboration

Tout comme un diagramme de séquence, un diagramme de collaboration décrit le déroulement d'une interaction entre plusieurs objets.
Sur un diagramme de collaboration, les objets constituent les nœuds d'un graphe dont les arcs correspondent aux actions (i.e., l'envoi de messages, l'instanciation et la destruction d'objets).
Le temps n'est pas associé à une dimension particulière du diagramme, mais la chronologie de l'interaction est exprimée par des numéros de séquence qui accompagnent les actions.

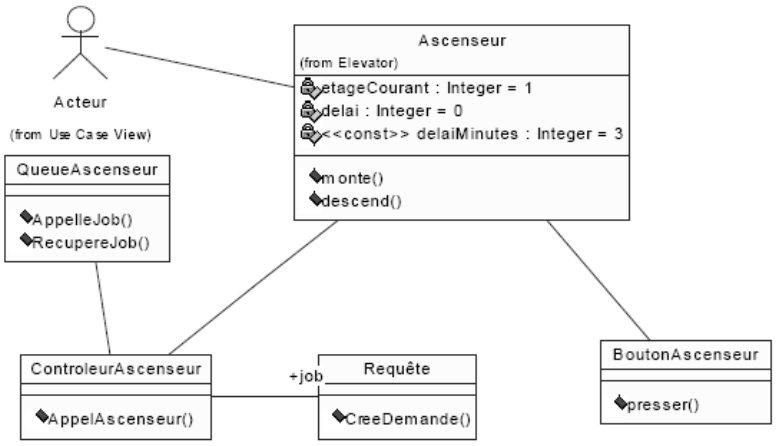
Les diagrammes de collaboration montrent les ***interactions*** et les ***liens*** entre un ensemble d'objets participant à un scénario.
Ils focalisent leur attention sur l'***aspect spatial*** des interactions.
Ils sont particulièrement utiles pour montrer la ***réalisation des use-cases*** (sans que les aspects temporels des interactions ne soient nécessairement rendus explicites).
Les ***objets*** sont dessinés comme des ***classes*** mais leurs ***noms*** sont ***soulignés***.
Les ***liens*** sont illustrés par des ***lignes*** (qui ressemblent à des lignes d'associations dans les diagrammes de classes mais sans les informations de multiplicité).
Un lien entre deux objets dénote l'envoi d'un message ou une action d'instanciation ou de destruction.
Un ***message***, accompagné d'une ***étiquette***, peut être associé à un lien pour montrer le ***sens*** et l'***ordonnancement*** de la transmission du message.



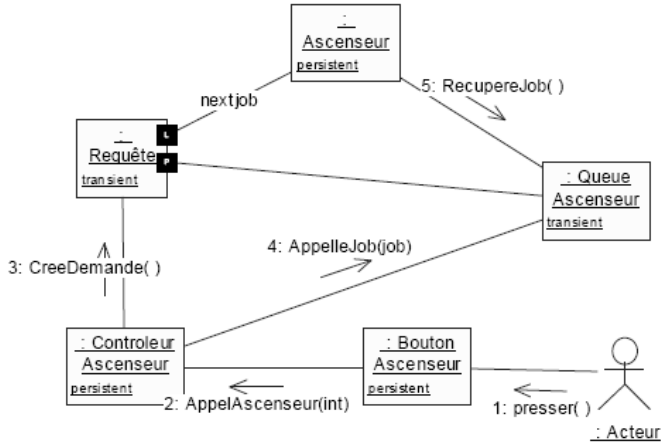
Les étiquettes des liens sont similaires à celles des transitions d'un diagramme de séquence. Elles sont cependant préfixées d'un numéro de séquence qui précise l'ordre dans lequel les actions sont effectuées. Ces numéros peuvent être imbriqués afin de mettre en évidence la structure de l'interaction.

Exemple 1

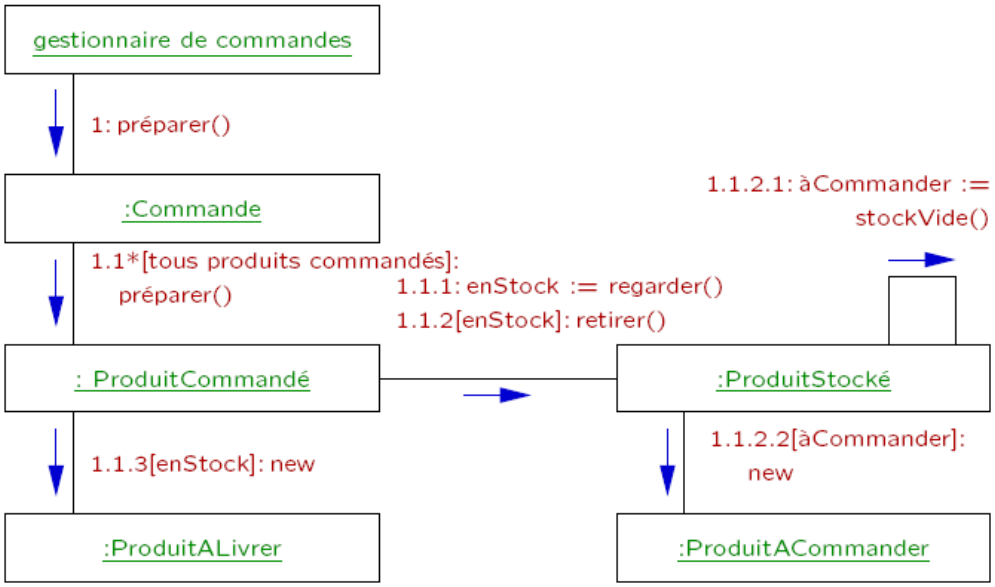
Exemple pour l'ascenseur : ***Classes***



Exemple pour l'ascenseur : ***Collaboration***



Exemple 2 : Exemple de gestion des commandes.



Les étiquettes de message associées aux liens adoptent la syntaxe suivante:

```
predecessor guard-condition sequence-expression return-value := signature
```

Les différents champs de l'étiquette ont la signification suivante:

1. `predecessor`: sert à la synchronisation entre processus légers signifiant que les messages reliés à des numéros de séquence spécifiques doivent être traités avant que le message courant soit transmis;
2. `guard-condition`: condition qui doit être vraie pour que le message soit transmis (syntaxe: `[condition booléenne]`);
3. `sequence-expression`: la syntaxe de cette partie de l'étiquette est la suivante: `[nombre-entier | nom] [récurrence]` où `nombre-entier` représente le numéro du message dans la séquence de messages qui sont envoyés dans la collaboration et où `nom` représente le nom du processus léger de contrôle. Le terme `récurrence` représente le nombre de fois que le message est envoyé. La syntaxe pour la récurrence est la suivante: `*[iteration-clause] [condition-clause]` où `iteration-clause` représente la façon dont l'itération doit s'effectuer (par exemple `i := 0..n`) et où `condition-clause` représente une condition de branchement1 (par exemple `x >= n`).
4. `return-value` est la variable dans laquelle est stockée la valeur de retour d'un message (par exemple `x := calc(n)`);
5. `signature`: syntaxe du message envoyé (nom, type de la valeur de retour, liste des noms et types des arguments).

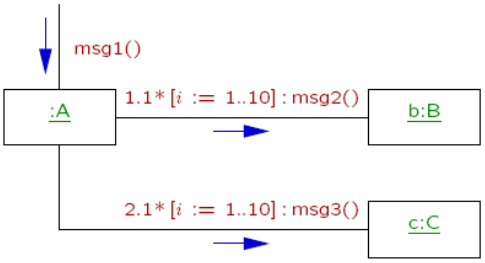
L'itération

Lorsque les numéros de séquence de deux marqueurs d'itération ne diffèrent que par leur dernier élément, on considère que ces deux marqueurs sont relatifs à la même itération (les descriptions de cette itération doivent alors coïncider).

Exemples :

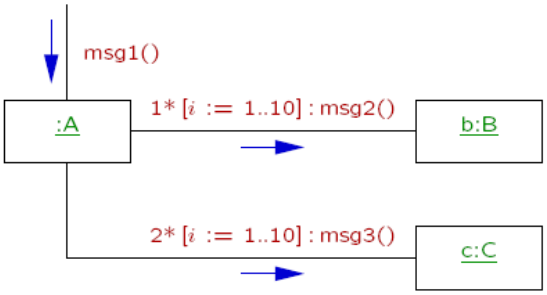
Cas 1 :

```
msg1()
{
  for (i = 1; i <= 10; i++)
    {
      b.msg2();
      c.msg3();
    }
}
```



Cas 2 :

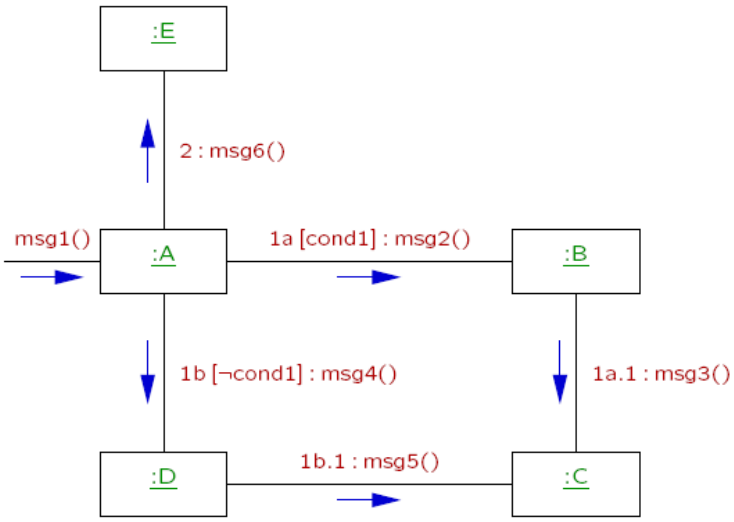
```
msg1()
{
  for (i = 1; i <= 10; i++)
    b.msg2();
  for (i = 1; i <= 10; i++)
    c.msg3();
}
```



Le branchement

Les différentes branches d'un choix conditionnel sont repérées par le même numéro de séquence auquel on ajoute un suffixe propre à la branche (e.g., 2a, 2b, 2c, . . .).

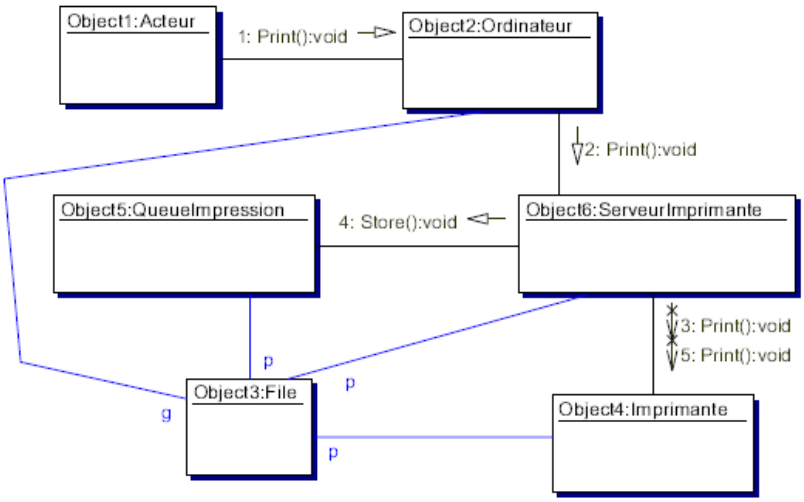
Exemple :



*Les **liens** dans les diagrammes de collaboration*
Le ***rôle*** de chaque objet peut être inclus aux extrémités de la connexion avec les qualificatifs du lien (rappel: les rôles et qualificatifs sont aussi inclus dans le diagramme de classe)
Le ***stéréotype*** du lien peut aussi être inclus dans la connexion.
On compte les ***stéréotypes*** suivants: ***global***, ***local***, ***parameter***, ***self***, ***vote***, ***broadcast***

*Les **stéréotypes** de **liens***
Global: associé à un rôle du lien pour montrer que l'instanciation en présence est ***globale*** (visible dans tout le système). Le récepteur est un objet global. Ceci se produit lorsque la référence à l'objet peut être obtenue à l'aide d'une méthode statique. Le stéréotype a utiliser est «global», ou [G].
Local: associé à un rôle du lien pour montrer que l'instanciation est une variable locale dans une opération. Le récepteur est contenu dans une variable locale de la méthode émettrice. Ceci se produit fréquemment lorsque l'objet a été créé par la méthode émettrice ou lorsqu'un résultat précédent a retourné un objet. Le stéréotype à utiliser est «local» ou [L].
Parameter: montre que l'instance est visible parce que c'est un paramètre d'une opération. Une référence à l'objet récepteur a été reçue comme paramètre par la méthode émettrice. Le stéréotype à utiliser est «parameter» or [P].
Self: montre qu'un objet peut s'envoyer des messages
Vote: contrainte imposée à un message limitant l'éventail des valeurs de retour (spécifie que la valeur de retour est choisie au vote majoritaire entre différentes valeurs de retour)
Broadcast: contrainte appliquée à un ensemble de messages pour signifier qu'ils ne sont pas exécutés dans un ordre donné.

Exemple



On remarque que, contrairement au diagramme séquentiel, le diagramme de collaboration inclut un objet de la classe `File`. En effet, dans l'interaction des objets de ce point fonctionnel, l'objet de la classe `File` ne reçoit aucun message et il n'est donc pas pertinent de le montrer dans le diagramme séquentiel. Par contre, l'objet de la classe `File` intervient dans la collaboration soit en tant qu'objet global que comme paramètre dans des messages. Le diagramme de collaboration rend explicite la présence de l'objet de la classe `File` et montre comment il intervient dans la collaboration. On remarque également les étiquettes avec leur numéro accompagnant les liens entre les objets d'une collaboration1. Il convient de remarquer que la classe `File` possède des liens avec les autres classes dans le diagramme de classes, ce qui permet d'inclure les mêmes liens entre les objets du diagramme de collaboration pour le point fonctionnel illustrant l'impression d'un fichier.