Chapitre 4

Diagrammes UML
Vue Dynamiques

1. Diagrammes d'interaction

1.1 Introduction

Les diagrammes d'interaction permettent d'établir un lien entre les diagrammes de cas d'utilisation et les diagrammes de classes : ils montrent comment des objets (instances de classes) communiquent pour réaliser une certaine fonctionnalité. Ils modélisent des aspects dynamiques du système. Généralement, un diagramme d'interaction correspond à un cas d'utilisation.
Les éléments essentiels d'un diagramme d'interaction sont les ***objets,*** les ***acteurs*** et les ***messages***.
Les actions entre objets et acteurs sont principalement: ***envoi de messages***, ***création*** et ***destruction d'objets***.
Deux types de diagramme d'interaction :
– diagramme de séquence
– diagramme de collaboration
Une même interaction peut être présentée aussi bien par l'un que par l'autre.

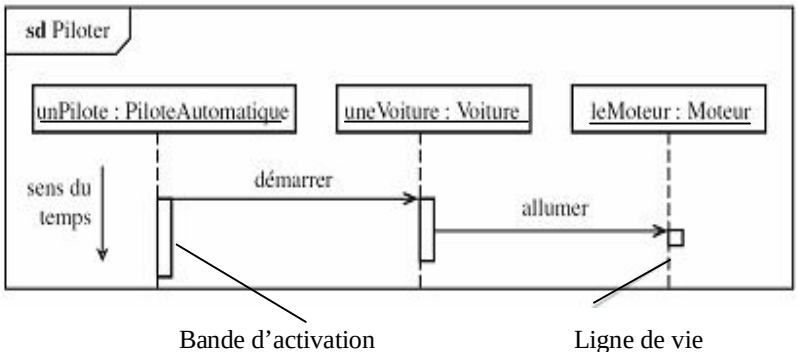
1.2 Diagramme de séquence

Un diagramme de séquence décrit la séquence temporelle des échanges de messages entre les objets et l'acteur pour réaliser une certaine tâche.
Un diagramme de séquence montre quels sont les objets qui participent à l'exécution d'un cas d'utilisation et quels sont les messages qu'ils échangent d'un point de vue temporel.
Les diagrammes de séquence privilégient ainsi la représentation temporelle à la représentation spatiale et sont plus aptes à modéliser les aspects dynamiques du système.

Notation :
Le diagramme de séquence permet de visualiser les messages par une lecture de haut en bas.
- L'axe vertical représente le temps,
- l'axe horizontal les instances qui collaborent.
- Une ligne de vie (ligne verticale en pointillé) est attachée à chaque instance et représente sa durée de vie.

- **Instances**
  - Un diagramme de séquence met en œuvre des instances (instance **de classe**, instance **d'acteur)** sur l'axe horizontal
  - l'activité d'une instance se voit sur sa ligne de vie grâce à une bande rectangulaire verticale appelée **bande d'activation** de l'instance.

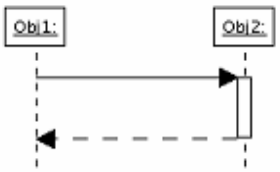
Exp : Cas d'un système de pilotage



- **Messages**

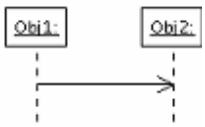
Un message, représenté par une flèche étiquetée par le nom de l'opération ou du signal invoqué (une interruption ou un évènement sont de bons exemples de signaux), définit une communication particulière entre des lignes de vie. Il peut être synchrone ou asynchrone :

- Synchrone : l'émetteur reste bloqué le temps que le récepteur traite le message envoyé Ce type de message est représenté par :



Un message de retour sera représenté avec des traits en pointillés

- Asynchrone : l'émetteur n'est pas bloqué lorsque le récepteur traite le message envoyé. Un tel message est représenté par :

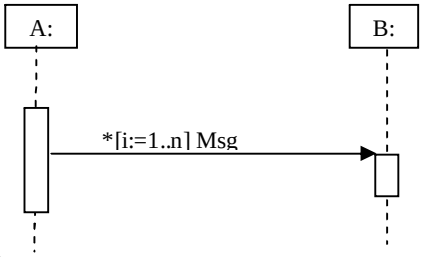


Syntaxe générale d'un message :
['['condition']' [*[||] '['itération']'] :] [résultat :=] message([paramètres])

La syntaxe de réponse à un message est la suivante :
[<attribut> =] message [: <valeur_de_retour>]

condition est une condition optionnelle sous forme d'expression booléenne.
itération spécifie (en langage naturel, entre crochets) l'envoi séquentiel (ou en parallèle, avec ||). On peut omettre cette spécification et ne garder que le caractère "*" (ou "*||") pour désigner un message récurrent envoyé un certain nombre de fois.
Résultat est la valeur de retour du message, qui sera par exemple transmise en paramètre à un autre message.
messsage est le nom du message.
paramètres désigne les paramètres (optionnels) du message.

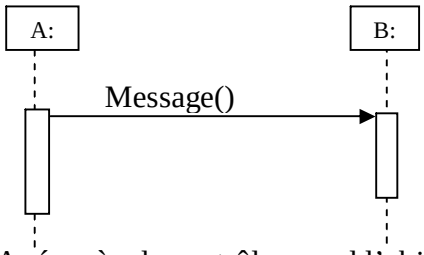
Exp :



Catégories de message :

- **Message d'appel (ou flot de contrôle emboîté)**

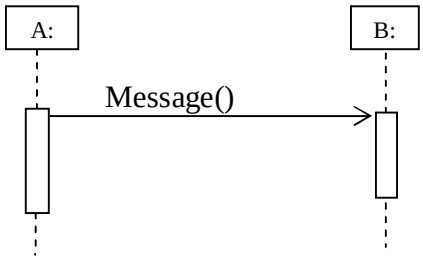
L'invocation d'une opération est le type de message le plus utilisé en programmation objet. L'invocation peut être asynchrone ou synchrone. Dans la pratique, la plupart des invocations sont synchrones.



L'objet A récupère le contrôle quand l'objet B a fini sa tâche.

- **Message d'envoie (flot de contrôle plat)**

Un message d'envoie envoie un signal à un objet : message asynchrone. Il indique la progression vers la prochaine étape d'une séquence.

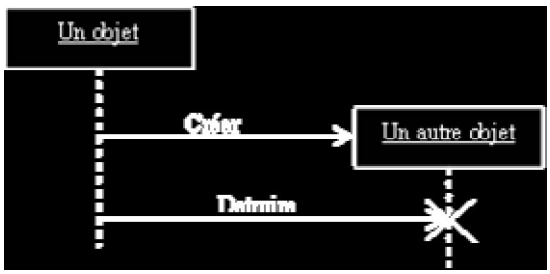


- **Message de création**

Une instance peut créer une autre instance grâce à un message de création. La création d'un objet est matérialisée par une flèche qui pointe sur le sommet d'une ligne de vie.

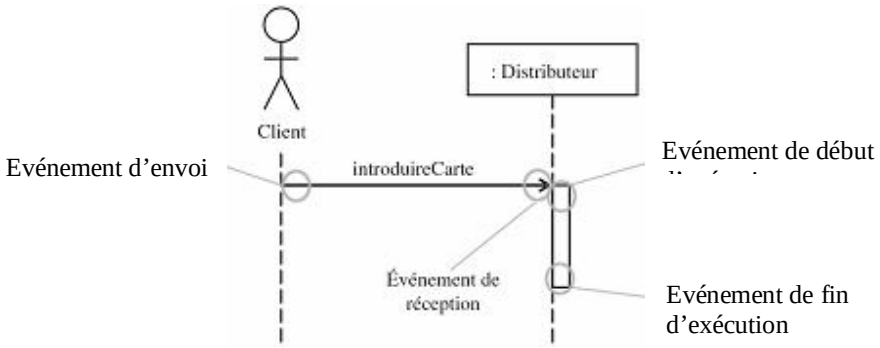
- **Message de destruction**

Une instance peut détruire une autre instance grâce à un message de destruction.

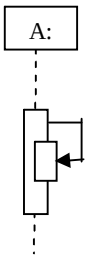


Remarques :

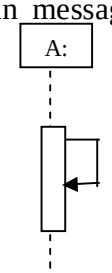
- UML permet de séparer clairement l'envoi du message, sa réception, ainsi que le début de l'exécution de la réaction et sa fin.
- Dans la plupart des cas, la réception d'un message est suivie de l'exécution d'une méthode d'une classe.



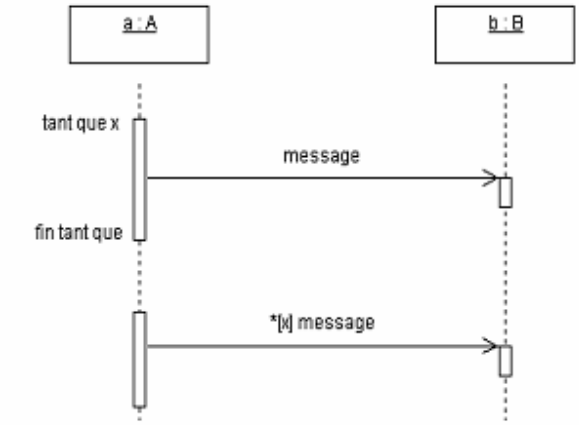
- Un message peut être récursif. L'envoi de **messages récursifs** se représente par un dédoublement de la bande d'activation.



- Un objet peut envoyer un message à lui-même (cas d'un objet composite) : **appel reflexif**.

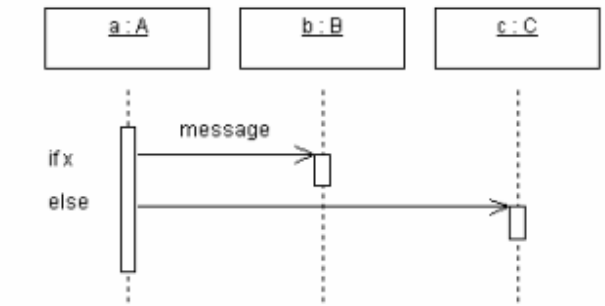


Représentation des itérations et des alternatives :

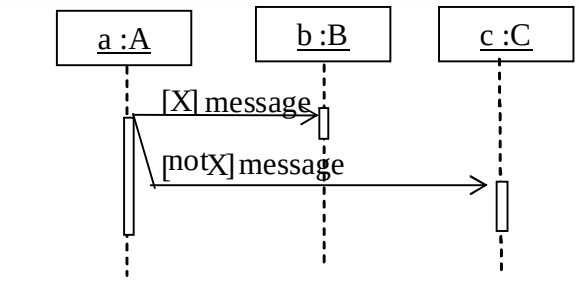


: Itération (le pseudo-code peut être while x loop end loop

: Itération

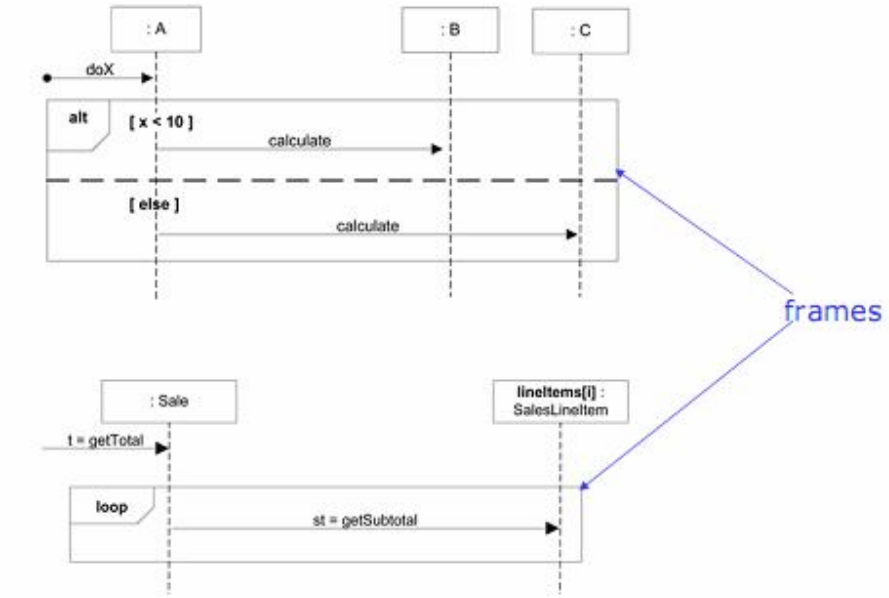


: Alternative



: Alternative

- Avec la notation de UML 2.x, les alternatives ou les itérations peuvent être représentées en utilisant des **frames.**



Limites des diagrammes de séquence : comment faire apparaître des opérations non séquentielles ?

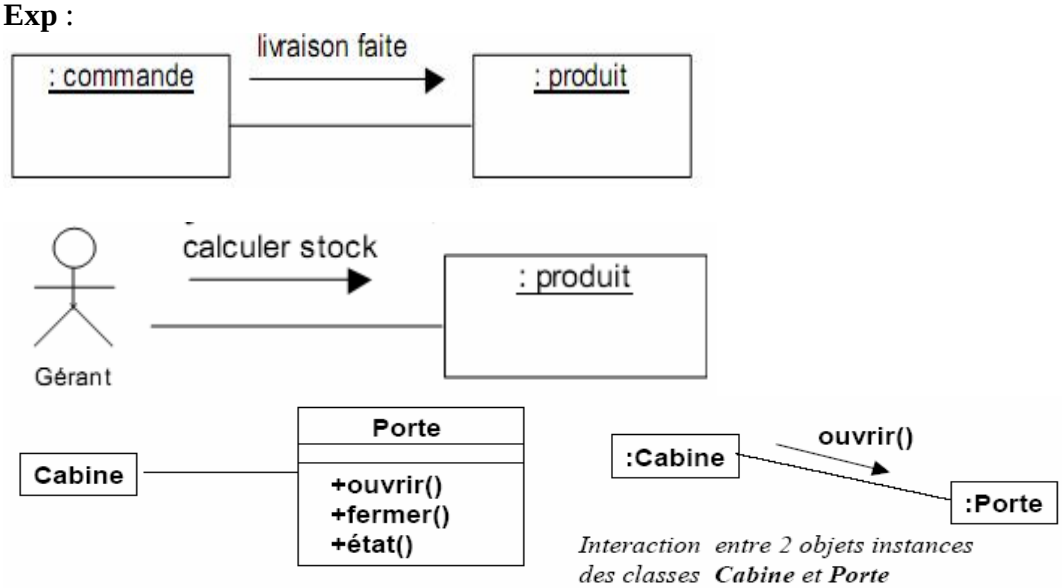
1.3 Diagramme de collaboration

Le diagramme de collaboration permet de mettre en évidence les interactions entre les différents objets du système. Une interaction montre le comportement d'une collaboration d'un groupe d'instances en se focalisant sur l'échange d'informations entre les éléments de la collaboration. Contrairement à un diagramme de séquence, un diagramme de collaboration rend compte de l'organisation spatiale des participants à l'interaction.
Un diagramme de collaboration est une extension d'un diagramme d'objets.
Il peut être utilisé dans le cadre de l'analyse,
- pour préciser le contexte dans lequel chaque objet évolue
- pour mettre en évidence les dépendances entre les différents objets impliqués dans l'exécution d'une opération ou d'un cas d'utilisation.

Notations :
Un diagramme de collaboration fait apparaître les interactions entre des objets et les messages qu'ils échangent.

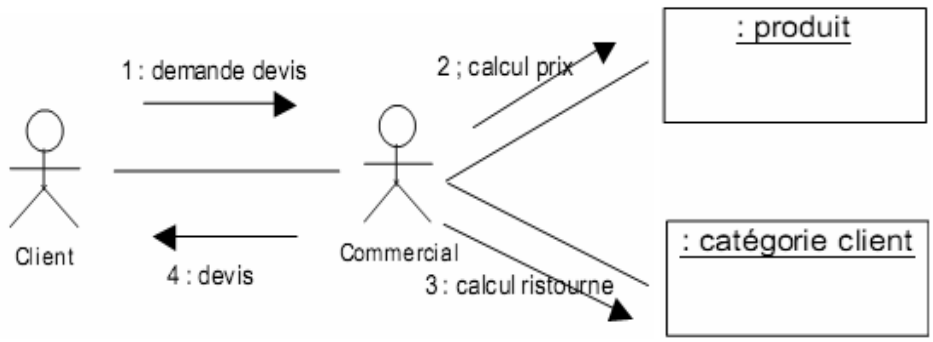
- **Instances et liens de communication:** Un lien montre l'envoie d'un message d'une instance à une autre. Il représente une instance d'une association. Les relations entre les instances (acteurs ou objets) sont appelées connecteurs et se représentent par un trait plein reliant les instances.
- **Interactions :** Une interaction définit la communication entre les objets sous la forme d'un ensemble partiellement ordonné de messages.

Exp :



- **Messages :** les messages sont généralement ordonnés selon un numéro de séquence croissant. La syntaxe des messages est la même que pour un diagramme de séquence excepté deux points :
  – la direction du message est spécifiée par une flèche supplémentaire au dessus du connecteur reliant les objets.
  – On numérote les messages par envoi et sous envoi désignés par des chiffres séparés par des points : ainsi l'envoi du message 1.4.3 est antérieur à celui du message 1.4.4 mais postérieur à celui du message 1.3.5. La simultanéité d'un envoi est désignée par une lettre les messages 1.6.a et 1.6.b sont envoyés en même temps.
  La séquence permet de préciser l'ordre d'émission des messages.

Exp :

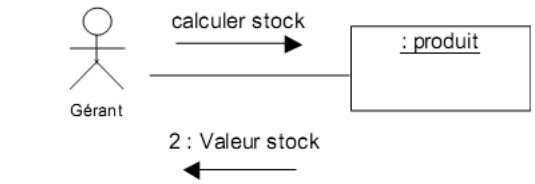


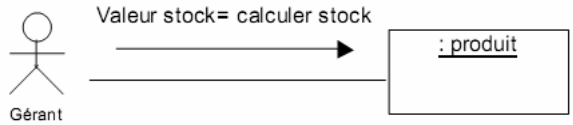
Syntaxe générale :

['['condition']' [séquence] [*[||] '['itération']'] :] [résultat :=] message([paramètres])

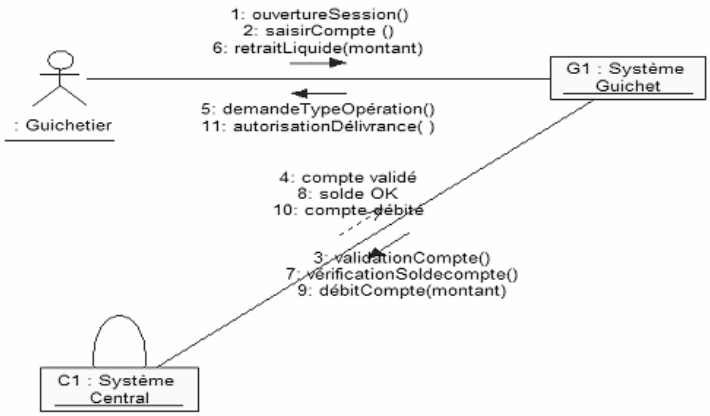
Certains messages peuvent solliciter un résultat. Ce cas peut être modélisé de 2 façons :
- un message de demande et un message de réponse
- indiquer sur le premier message le résultat attendu (lorsque le message en retour est attendu

immédiatement)





Le message de réponse peut être représenté par une flèche en trait discontinu.



2. Diagrammes d'activités

2.1 Présentation et intérêt

Les diagrammes d'activités permettent de modéliser un comportement tout en mettant l'accent sur les traitements. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle (séquence, choix, itération, parallélisme) et de flots de données entre activités alors que les diagrammes d'interaction modélisent le flot de contrôle entre objets.

Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation. Les diagrammes d'activités sont relativement proches des diagrammes d'états-transitions dans leur présentation, mais leur interprétation est différente. Ils permettent de spécifier des traitements a priori séquentiels et offrent une vision très proche de celle des langages de programmation impératifs comme C++ ou Java.

Les diagrammes d'activités sont très utilisés dans la phase de conception. Ils sont particulièrement adaptés à la description des cas d'utilisation. Plus précisément, ils viennent illustrer et consolider la description textuelle des cas d'utilisation. De plus, leur représentation sous forme d'organigrammes les rend facilement intelligibles et beaucoup plus accessibles que les diagrammes d'états-transitions.

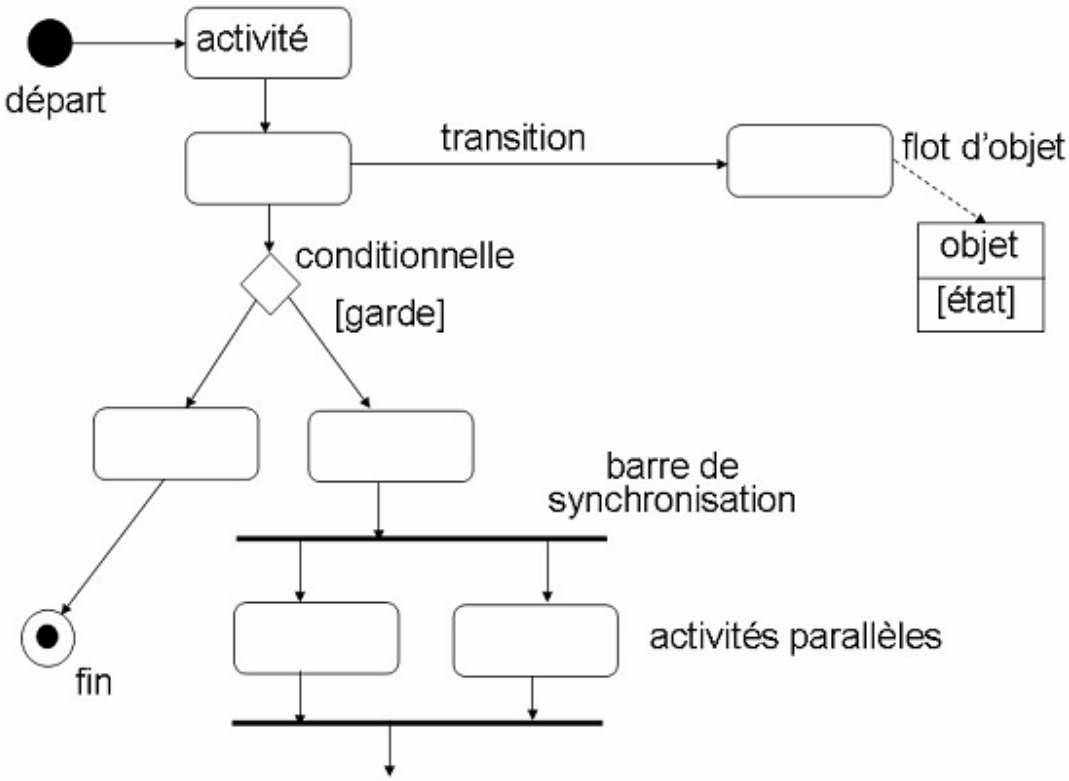
Les diagrammes d'activités sont également utiles dans la phase de réalisation car ils permettent une description si précise des traitements qu'elle autorise la génération automatique du code.

Un diagramme d'activités comprend :

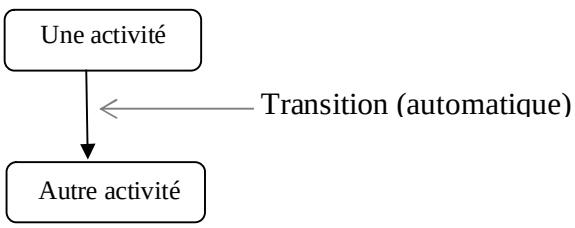
- Les activités effectuées par le système et par les acteurs,
- L'ordre dans lequel ces activités sont effectuées et
- Les dépendances éventuelles entre les activités.

Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles
Le passage d'une activité vers une autre est matérialisé par une transition.
Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre (elles sont automatiques).
Il n'y a pas d'autres événements déclencheurs, à l'exception des **signaux** qui sont échangés par les objets. Outre les signaux, on peut trouver des conditions, des synchronisations ainsi que des points de jonctions sur ces transitions « automatiques ».

2.2 Notation

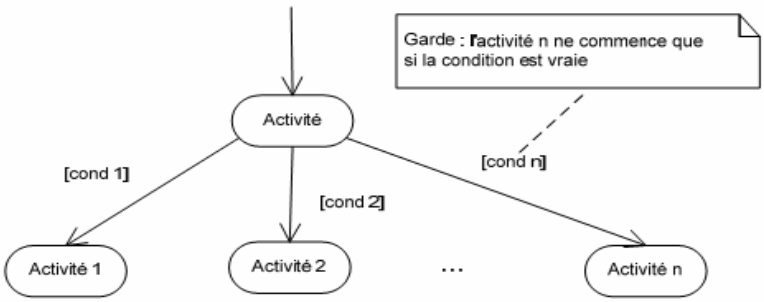


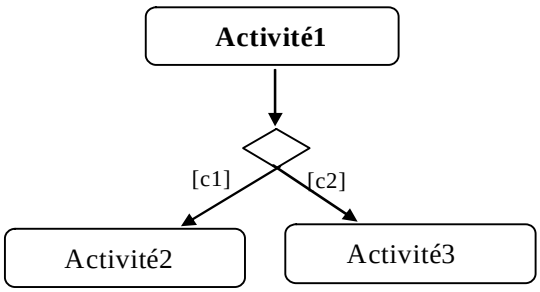
2.2.1 Activités et transition



Une activité peut définir une action simple (affectation, calcul arithmétique, émission, réception,

création…) ou une séquence organisée d'unités de comportement.
Les transitions spécifient l'enchaînement des traitements et définissent le flot de contrôle.
Les transitions peuvent être conditionnelles :





2.2.2 Nœuds d'activités

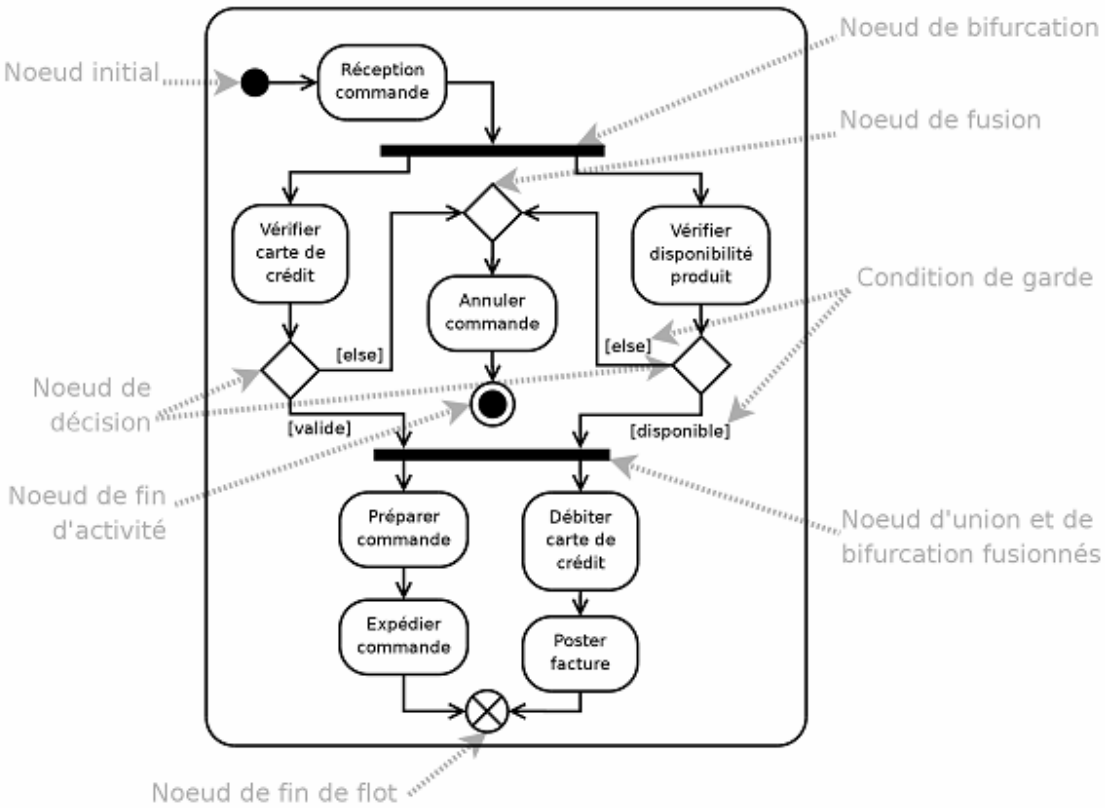
Il existe trois familles de nœuds d'activités :
– les nœuds exécutables ;
– les nœuds objets et
– les nœuds de contrôle

De la gauche vers la droite le nœud représente:
une action, qui est une variété de nœud exécutable,
un nœud objet,
un nœud de décision ou de fusion,
un nœud de bifurcation ou d'union,
un nœud initial, un nœud final et
un nœud final de flot.

Un **nœud exécutable** est un nœud d'activité qu'on peut exécuter (i.e. une activité).
Un **nœud de contrôle** est un nœud d'activité abstrait utilisé pour coordonner les flots entre les nœuds d'une activité.
Il existe plusieurs types de nœuds de contrôle :
– nœud **initial**;

– nœud **de fin d'activité** ;
– nœud **de fin de flot**;
– nœud **de décision**
– nœud **de fusion**;
– nœud **de bifurcation**;
– nœud **d'union**.

Exp :



Un nœud de décision est un nœud de contrôle qui permet de faire un choix entre plusieurs flots sortants. Il possède un arc entrant et plusieurs arcs sortants.

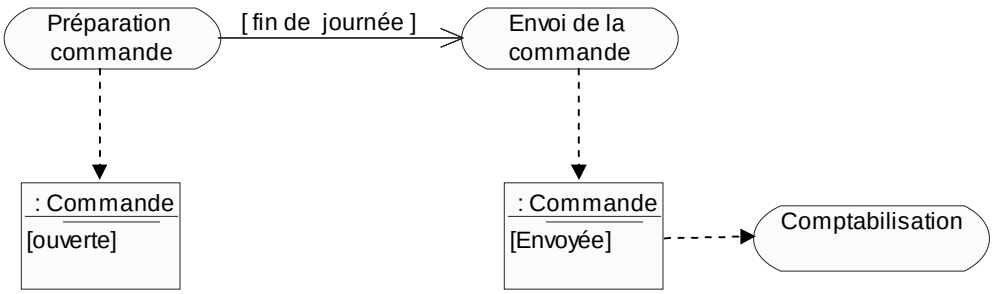
Un nœud de fusion rassemble plusieurs flots alternatifs entrants en un seul flot sortant.

Un nœud de bifurcation (ou de débranchement) sépare un flot en plusieurs flots concurrents. Un tel nœud possède donc un arc entrant et plusieurs arcs sortants. On apparie généralement un nœud de bifurcation avec un nœud d'union pour équilibrer la concurrence.

Un nœud d'union (ou de jointure) synchronise des flots multiples. Un tel nœud possède donc plusieurs arcs entrants et un seul arc sortant. Lorsque tous les arcs entrants sont activés, l'arc sortant l'est également.

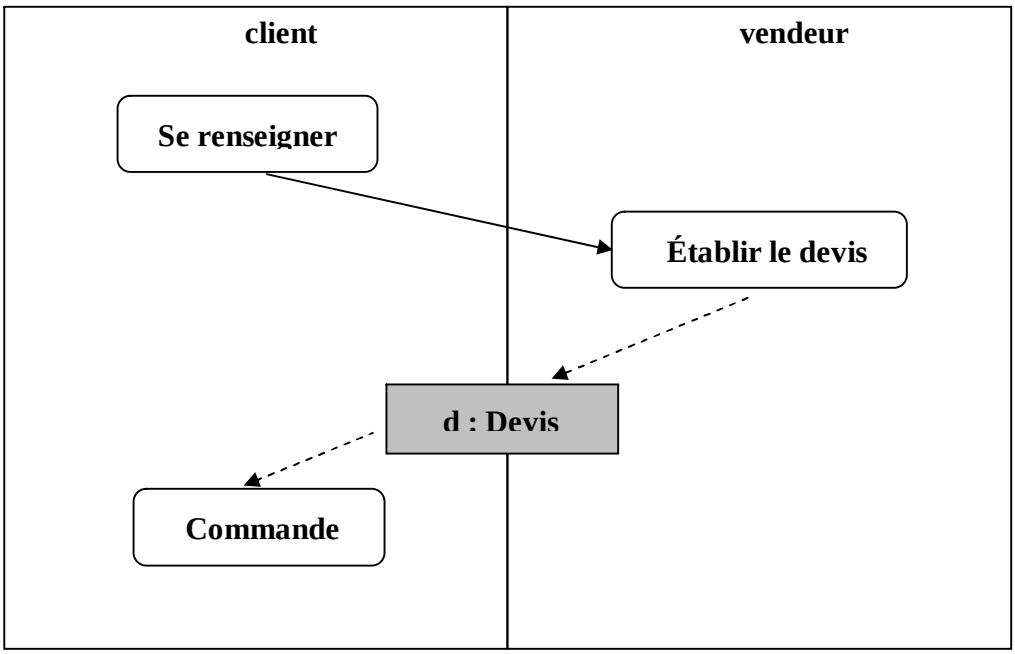
2.2.3 Flot d'objet

Un flot d'objets permet de passer des données d'une activité à une autre ou encore fait intervenir un nœud d'objet détaché d'une activité particulière et pourra ensuite être relié par des arcs à des activités sources et cibles. Le nom d'un état, ou d'une liste d'états, de l'objet peut être précisé entre crochets après ou sous le type de l'objet.



2.3 Partition

Les partitions, souvent appelées couloirs, permettent d'organiser les nœuds d'activités dans un diagramme d'activités en opérant des regroupements pour mieux mettre en évidence les différentes responsabilités au sein d'un service.



2.4 Construction d’un diagramme d’activités

1. **Identifiez la portée (« scope ») du diagramme d'activité :**

Commencez en identifiant ce que vous allez modéliser. Un seul UC? Une partie d'unUC? Un traitement qui inclut plusieurs UC? Une méthode de classe ?

2. **Ajouter l’état de *départ* et de *terminaison*** (noeuds initial et de fin de flot)
3. **Ajouter les activités:**

Si vous modélisez un use case, introduisez une activité pour chaque use case principal. Si vous modélisez un traitement « workflow », introduisez une activité pour chaque processus principal, souvent un UC. Enfin, si vous modélisez une méthode, il est souvent nécessaire d’avoir une activité pour chaque grande étape de la méthode.

4. **Ajouter des transitions (séquentielles), des transitions alternatives (conditionelles), des synchronisation entre des activités, des itérations.**
5. **Identifier des couloirs et répartir des activités identifiées dans ces couloirs.**