

Face à la complexité sans cesse croissante des systèmes que nous utilisons quotidiennement, l'industrie s'est dotée de moyens de plus en plus performants pour piloter (coût, délai, qualité, impact environnemental, etc.) les différentes phases de conception, d'intégration et de validation de systèmes apportant une solution économique et performante aux besoins d'un client.

Cet objectif est atteint grâce à :

- ✚ une prise en compte rationnelle et précise de la demande du client,
- ✚ une coordination efficiente des nombreux métiers sollicités lors du développement,
- ✚ une décomposition ordonnée de l'architecture du système étudié,
- ✚ la prise en compte, tout au long du projet, de son cycle de vie

1. Le cycle de vie d'un produit



Lors du cours précédent, nous avons vu que le cahier des charges peut être décrit grâce à la méthode APTE (diagramme « bête à cornes » et « pieuvre ». Il existe une autre modélisation qui se développe de plus en plus dans les entreprises qui est le langage SYSML

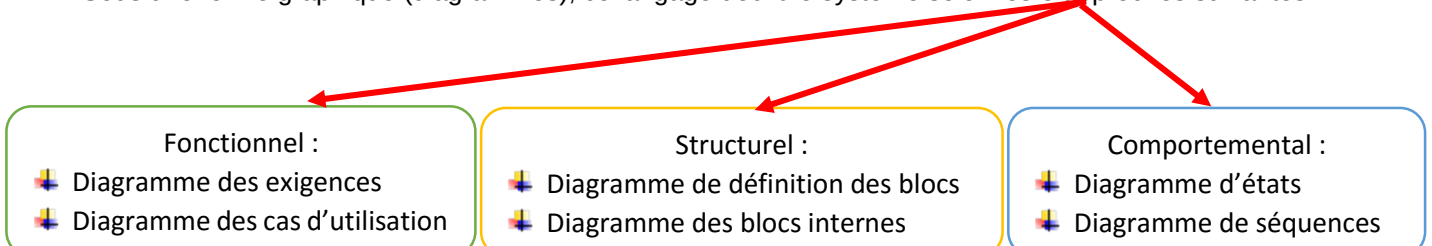
2. Le langage SYSML :

Afin de faciliter :

- ✚ La collaboration transdisciplinaire,
- ✚ L'interprétation, le stockage et le partage des données,
- ✚ Les modélisations communes,

dans les phases de conception, d'intégration et de validation du projet étudié, langage est utilisé : SysML. (Systems Modeling Language).

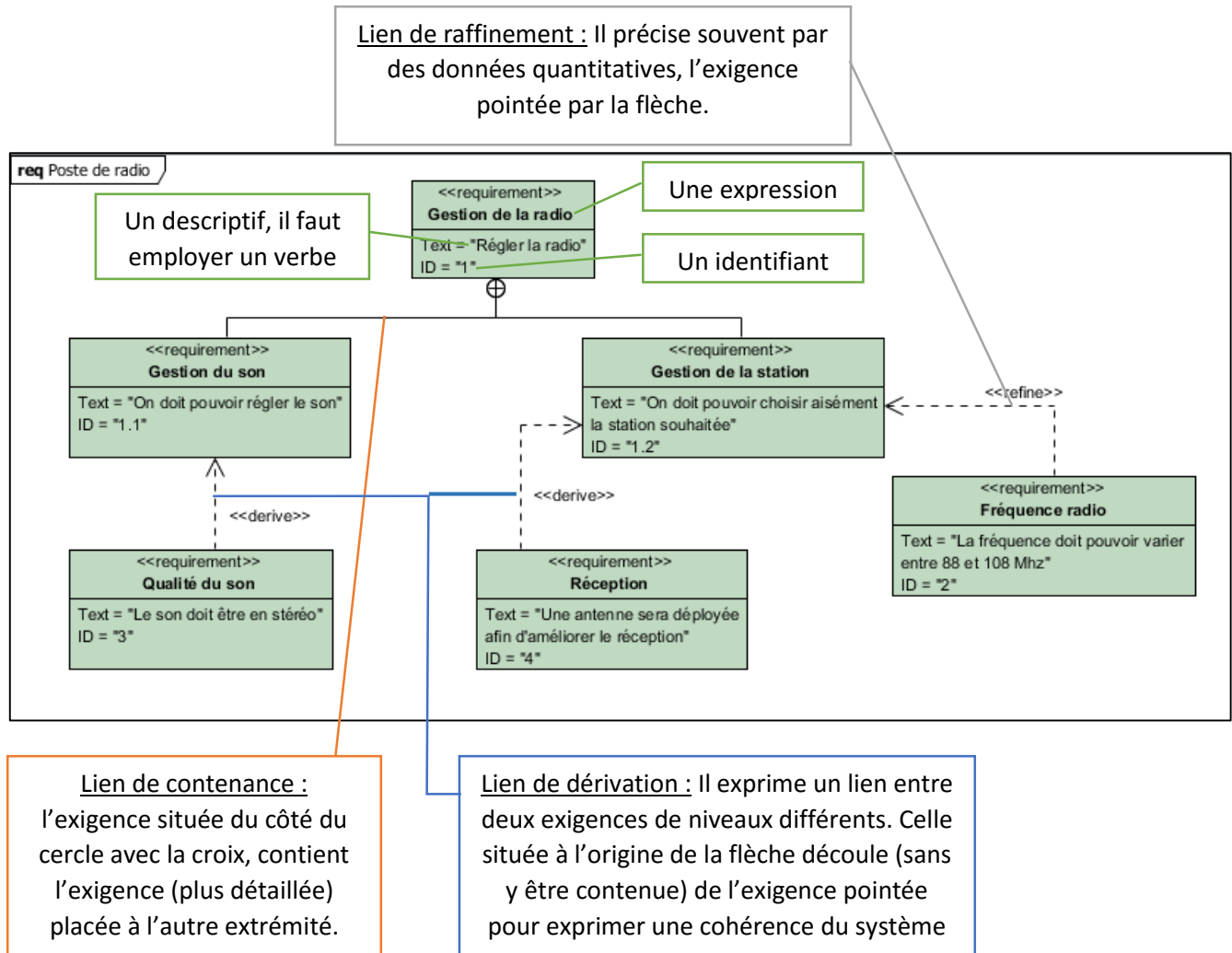
Sous une forme graphique (diagrammes), ce langage décrit le système selon les 3 approches suivantes :



a) Le diagramme des exigences (Requirements) :

Il représente, hiérarchise, spécifie et documente les **exigences** fonctionnelles, techniques, environnementales et économiques du système. Une **exigence** permet de spécifier une capacité, une contrainte, une condition de performance, de fiabilité ou de sécurité que doit satisfaire le système.

Exemple : un poste de radio



Remarque : Ne pas placer toutes les exigences sur un seul diagramme qui deviendrait alors illisible.

b) Le diagramme des cas d'utilisation (Use cases) :

Il représente les fonctionnalités ou les services attendus par le système du point de vue de l'acteur.

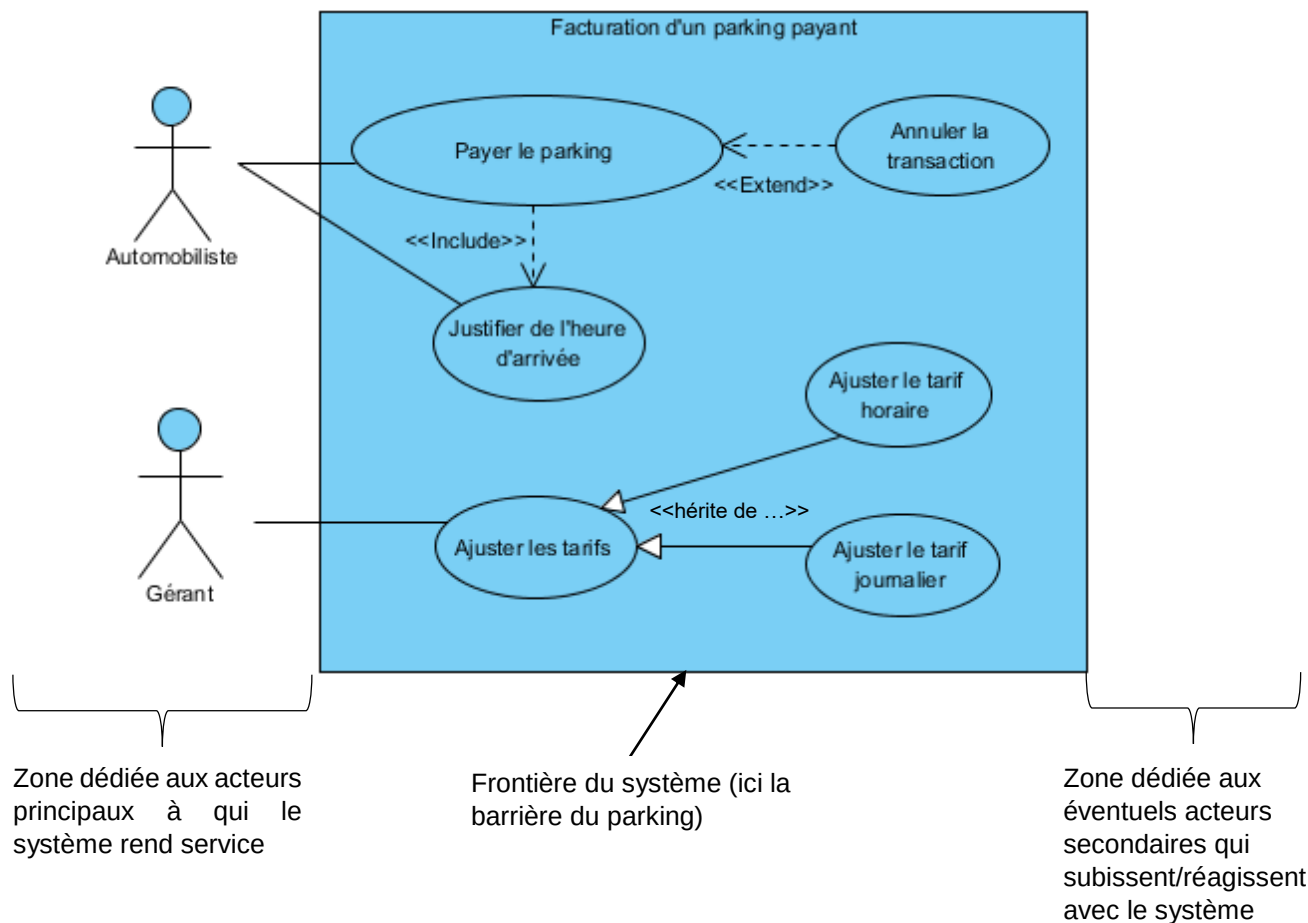
Les cas d'utilisation sont placés dans des ovales et exprimés par **un verbe à l'infinitif suivi de complément(s)**.

Par **acteur**, on entend toute entité externe au système étudié qui :

- interagit (réalise/accomplit ou subit) de façon autonome avec le système,
- dispose d'une 'intelligence' (il ne peut pas être de nature inerte).

Par exemple : un humain, un serveur, un autre système, un animal.

Représentation : un humain par :  un système par un cube : 



- ✚ « **extend** » **Lien d'extension** : Le cas d'utilisation de base peut (option) incorporer celui placé à l'origine de la flèche.
- ✚ « **include** » **Lien d'inclusion** : Le cas d'utilisation de base incorpore systématiquement celui placé à l'extrémité de la flèche.
- ✚ « **hérite de ...** » **Liens de spécialisation/généralisation** : Ils relient des cas d'utilisation descendants qui héritent de la description d'un cas d'utilisation supérieur (parent commun).

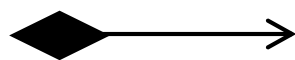
Remarque :

- ✚ Ce diagramme ne doit indiquer ni la manière dont il va assurer les services, ni les solutions technologiques envisagées.
- ✚ Plusieurs diagrammes d'utilisation peuvent être établis pour un système afin d'en améliorer la compréhension.

c) Le diagramme de définition des Blocs (BDD) :

Il définit le système dans son environnement (éléments humains et matériels qui interagissent avec lui)

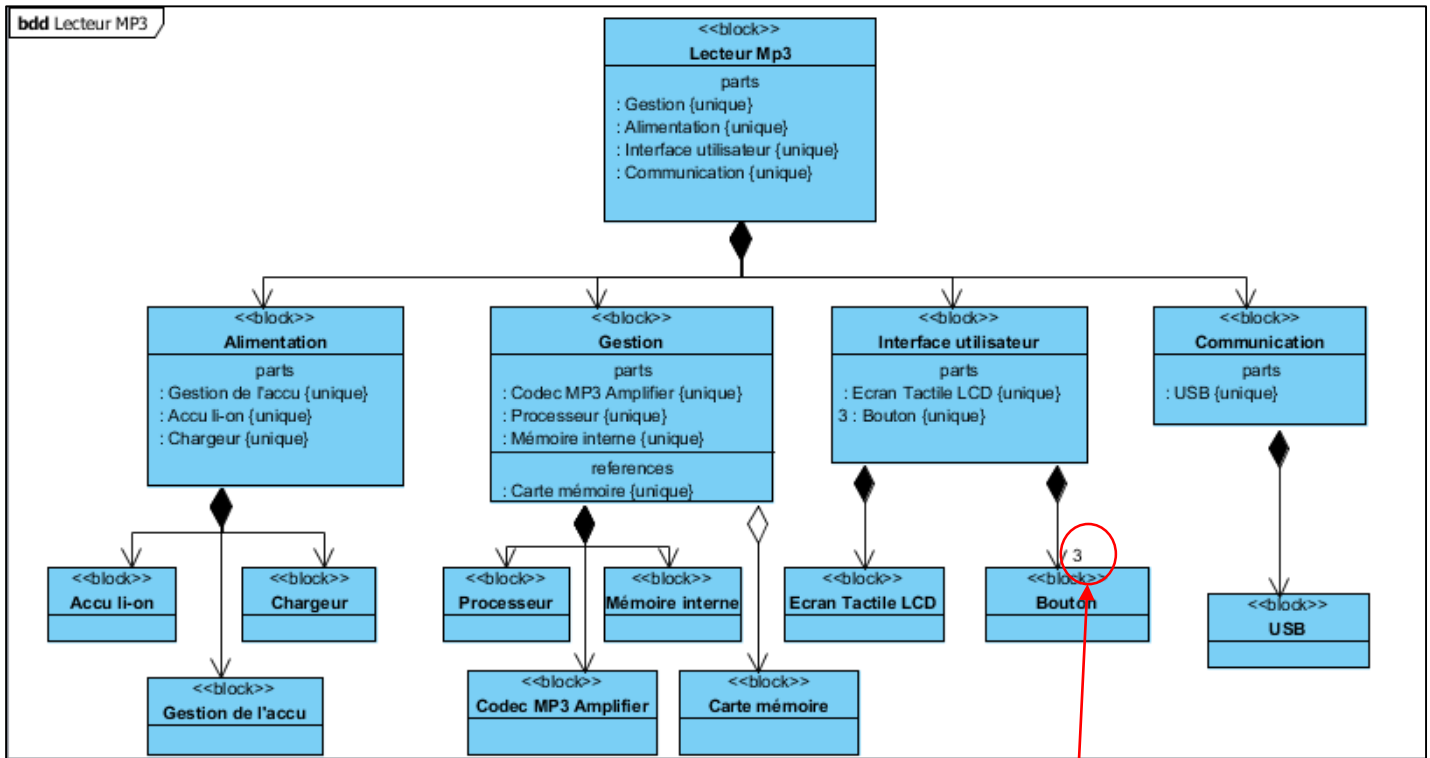
Il définit l'architecture matérielle et logicielle globale du système sous une représentation arborescente de blocs. Chacun d'eux se limite à la définition d'une famille (classe) de composants principaux.



Relation de composition : le bloc situé du côté du losange plein a besoin du sous-bloc coté flèche. Sa lecture : « possède un »

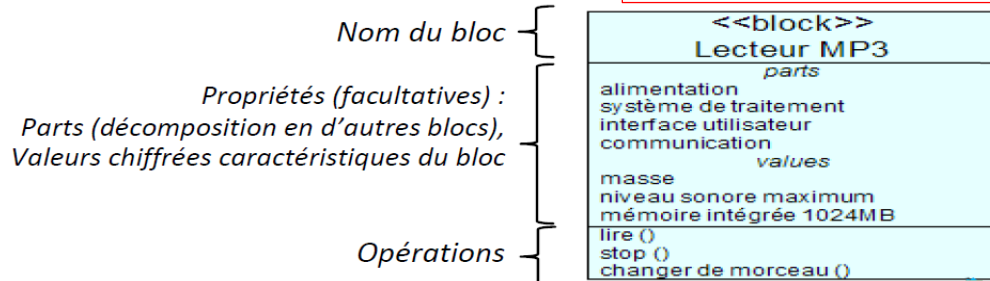


Relation d'agrégation : le bloc situé du côté du losange vide peut avoir besoin du sous-bloc coté flèche, c'est une relation optionnelle. Sa lecture : « peut avoir un »



Cardinalité ou multiplicité : Nombre de sous-blocs possédant les mêmes caractéristiques. Cela évite d'avoir à tous les représenter.

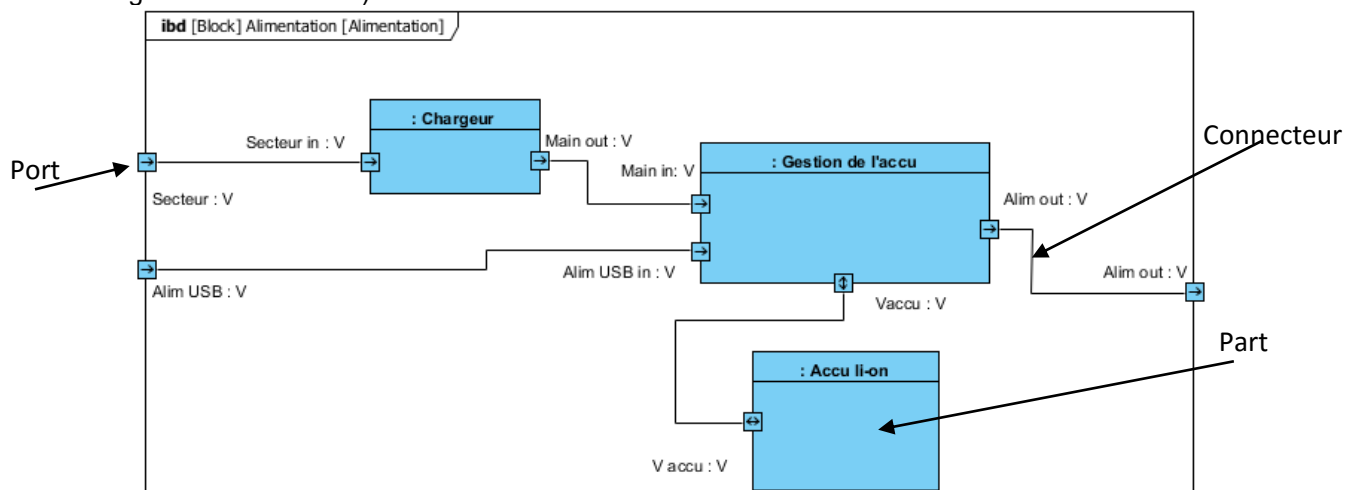
On peut détailler plus les blocks :



Remarque : Un BDD ne décrit ni les fonctions, ni le comportement du système

d) Le diagramme des Blocs internes (IBD) :

Il décrit la structure interne d'un bloc issu du BDD, c'est-à-dire ses composants et les échanges (flux de matière, d'énergie ou d'information) entre les blocs internes.



Un port symbolise ce qui peut entrer/sortir d'un block. On en distingue 2 types :

- ✚ Le port de flux (flow port)  qui correspond à l'entrée/sortie d'un **flux** de matière, d'énergie, de données, etc. Le sens de circulation peut être précisé par une flèche.
- ✚ Le port standard  qui représente un point de communication lié à un **service** :
 - une entrée/sortie véhiculant des informations (ou des ordres) logiques/numériques comme l'état d'un bouton poussoir ;
 - Une communication plus élaborée entre 2 parts via un réseau.

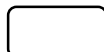
Les connecteurs représentent les liaisons entre les ports ou les parts, en précisant éventuellement la nature du lien ou ce qui est réellement véhiculé.

Remarque : les deux diagrammes doivent être cohérents (le bloc d'alimentation comprend 3 parties (blocks) que l'on retrouve dans le IBD de l'alimentation).

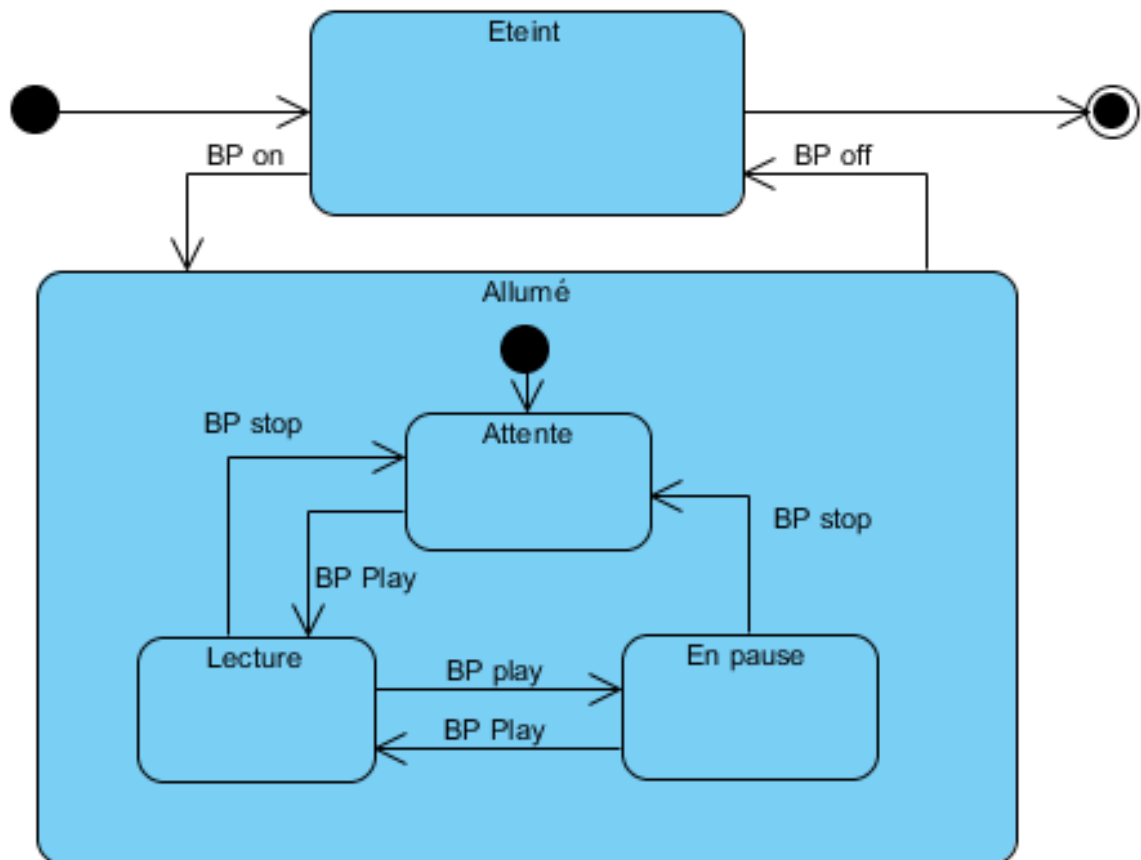
e) Le diagramme d'états (STM : State Machine) :

Il décrit les états que peut prendre le système et les transitions qui régissent les changements d'états.

Etat initial : ● **Etat final :** ⊙

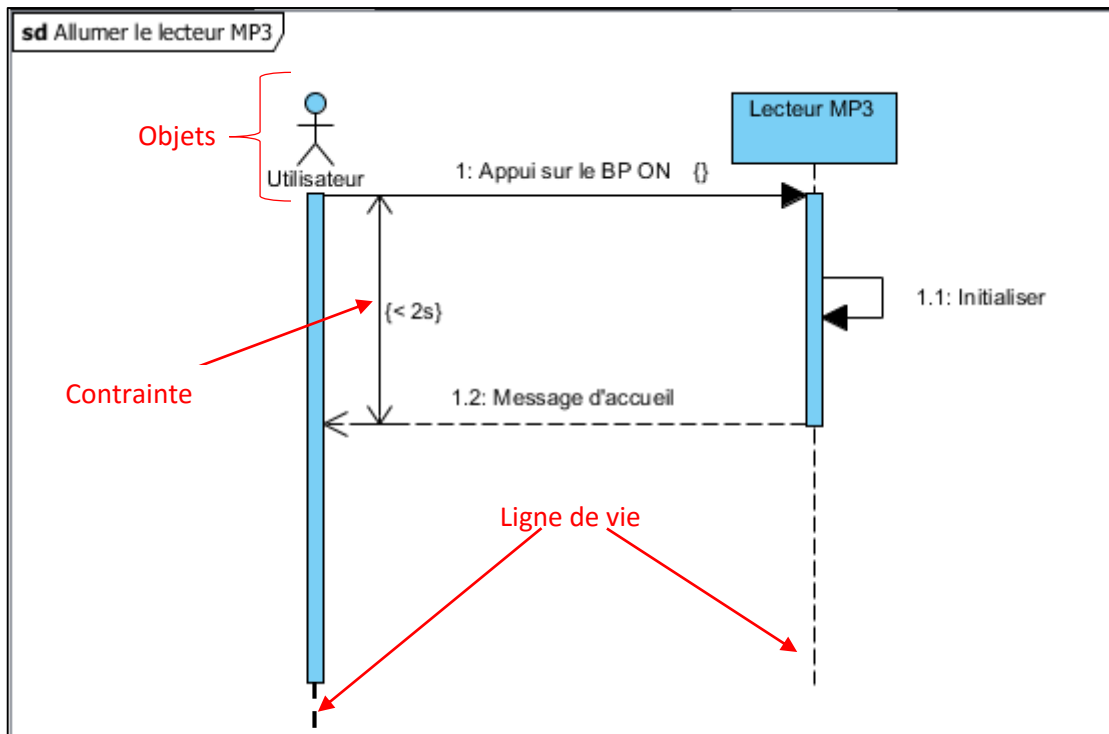
 Etat exclusif d'un bloc qui correspond à une séquence = entrée → phase active (ou attente) → sortie

→ La **transition** entre 2 états se formalise par une flèche et un **évènement** (simple expression). Pour préciser une **transition**, l'**évènement** peut nécessiter une **condition** (expression booléenne) et induire un **effet** (action).



f) Le diagramme de séquence (SD) :

Il représente l'échange de messages entre le système et des acteurs, ou entre des parties du système, de manière chronologique en précisant d'éventuelles contraintes de temps. La lecture d'un tel diagramme se fait de haut en bas.



Lorsque une séquence de message n'est pas linéaire (conditionnelle, répétitive, simultanée), les messages concernés sont encadrés par des fragments combinés :

