Séance 2

SYSML

Un langage pour décrire les systèmes

Table des matières

Introduction

Dans un monde de plus en plus technologique, la conception et la gestion de systèmes complexes nécessitent des outils capables de représenter et de structurer les informations de manière claire et rigoureuse. Le SysML (Systems Modeling Language) se positionne comme un langage de modélisation puissant, adapté aux besoins de nombreux secteurs industriels, notamment dans la filière STI2D.

Objectifs généraux

- Développer l’approche système
- Favoriser l’analyse et la résolution de problématiques techniques
- Renforcer les compétences numériques

Compétences Visées

- Analyser un besoin et un système technique
- Concevoir des solutions techniques
- Valider les choix techniques
- Communiquer et collaborer

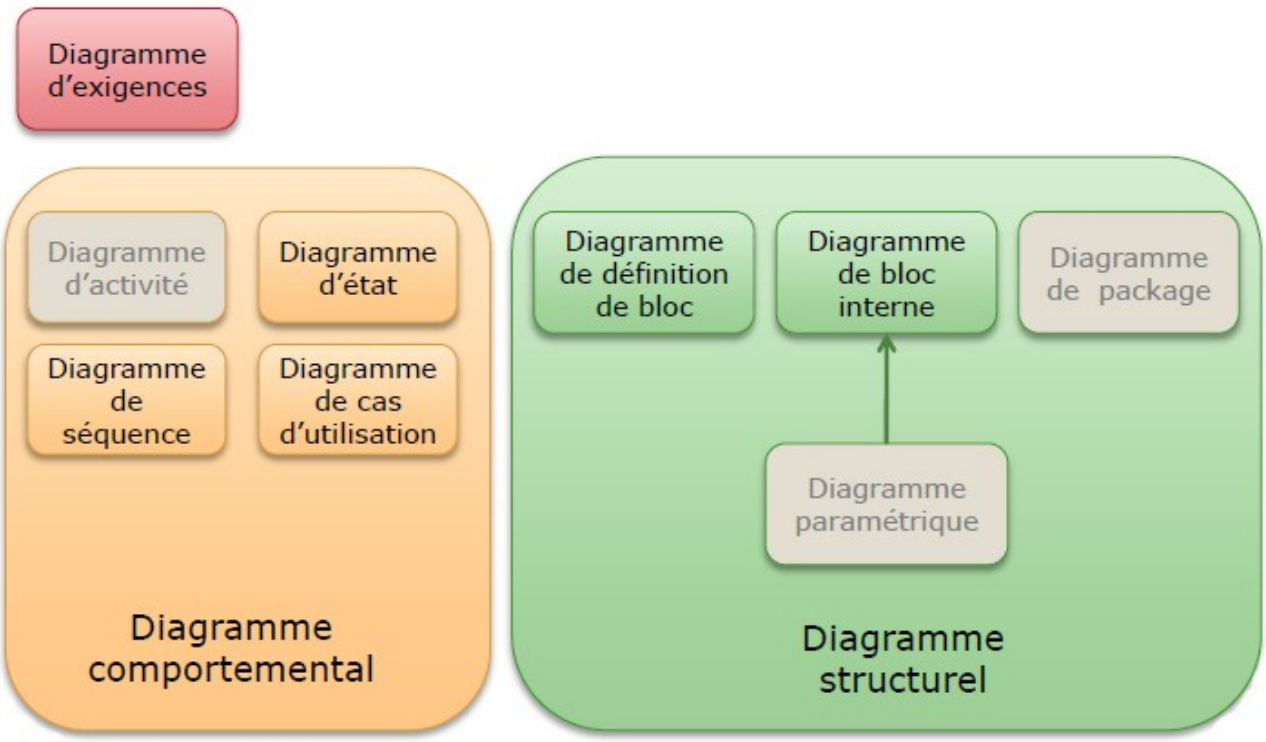
Langage de modélisation graphique de système

Les systèmes sont devenus plus complexes et pluri techniques, un besoin de langage transversal et unifié apparaît. Le SysML doit permettre ainsi à des acteurs de corps de métiers différents de collaborer autour d'un modèle commun pour définir un système.

A travers des outils graphiques, les diagrammes, le SysML permet de décrire un système suivant différents points de vue :

- **Fonctionnel** : *que doit faire le système ?*
- **Structurel** : *comment est construit le système ?*
- **Comportemental :** *comment doit se comporter le système dans le temps ?*

Le langage SysML qui provient de l'univers informatique n'est pas uniquement un outil descriptif puisqu'à partir de certains diagrammes, une passerelle permet de programmer directement le système.
En STI2D, nous nous limiterons à lire et à compléter des diagrammes déjà établis.

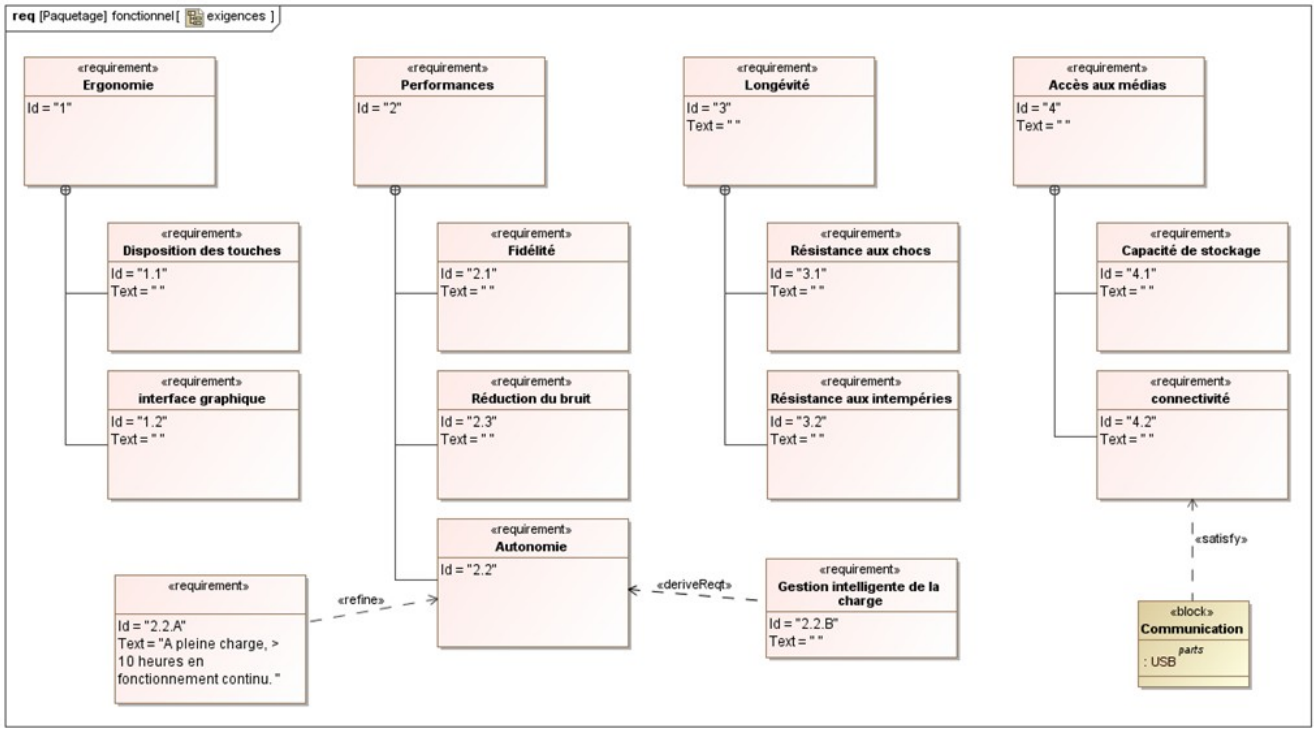


Les diagrammes grisés ne sont pas abordés en sti2d.

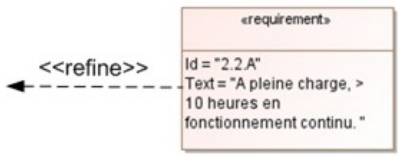
Pour illustrer les différents diagrammes, nous utiliserons comme support un iPod.

Diagramme des exigences (req – requirement diagram)

Une exigence permet de spécifier une fonction que le système devra réaliser ou une condition de performance, de fiabilité, de sécurité… Le diagramme permet de structurer les besoins. C'est un moyen de communication entre les concepteurs et les clients du système.
La représentation part de l'exigence principale puis se décompose en exigences nécessaires pour la réalisation de l'exigence principale. Il peut faire apparaître les blocs qui vont satisfaire ces exigences.



Indique qu'une exigence générale est constituée d'un ensemble d'exigences plus détaillées.



« refine » (Raffinement) : précise un élément, un paramètre de l'exigence pointée.

« deriveRqt » (Découle) : est déduit de l'exigence pointée.

« satisfy » (Satisfait) : répond à la demande formulée par l'exigence pointée.

Diagramme de cas d'utilisation (uc – use case diagram)

Ce diagramme est une représentation des fonctionnalités du système. Il indique dans quel cas ce système est utilisé et par qui. C'est à partir de ces « cas d'utilisation » que l'ensemble de la description comportemental se décline. Un diagramme des cas d'utilisation peut être complété au fur et à mesure que l'analyse du problème se précise. C'est notamment le rôle des « extensions » (*extend*) et des « inclusions » (*include*).

Lien possible entre les cas d'utilisation et les acteurs :

A ——— B		Association
A ———▷ B		Généralisation, héritage. A est une spécialisation de B. A est une sorte de B.
A - - - *Include* - - -> B		Relation d'inclusion : le cas A inclut obligatoirement le cas B. Liaison d'obligation.
A - - - *Extend* - - -> B		Relation d'extension : le cas A est une extension du cas B. Le cas A complète le comportement du cas B. L'extension est optionnelle.

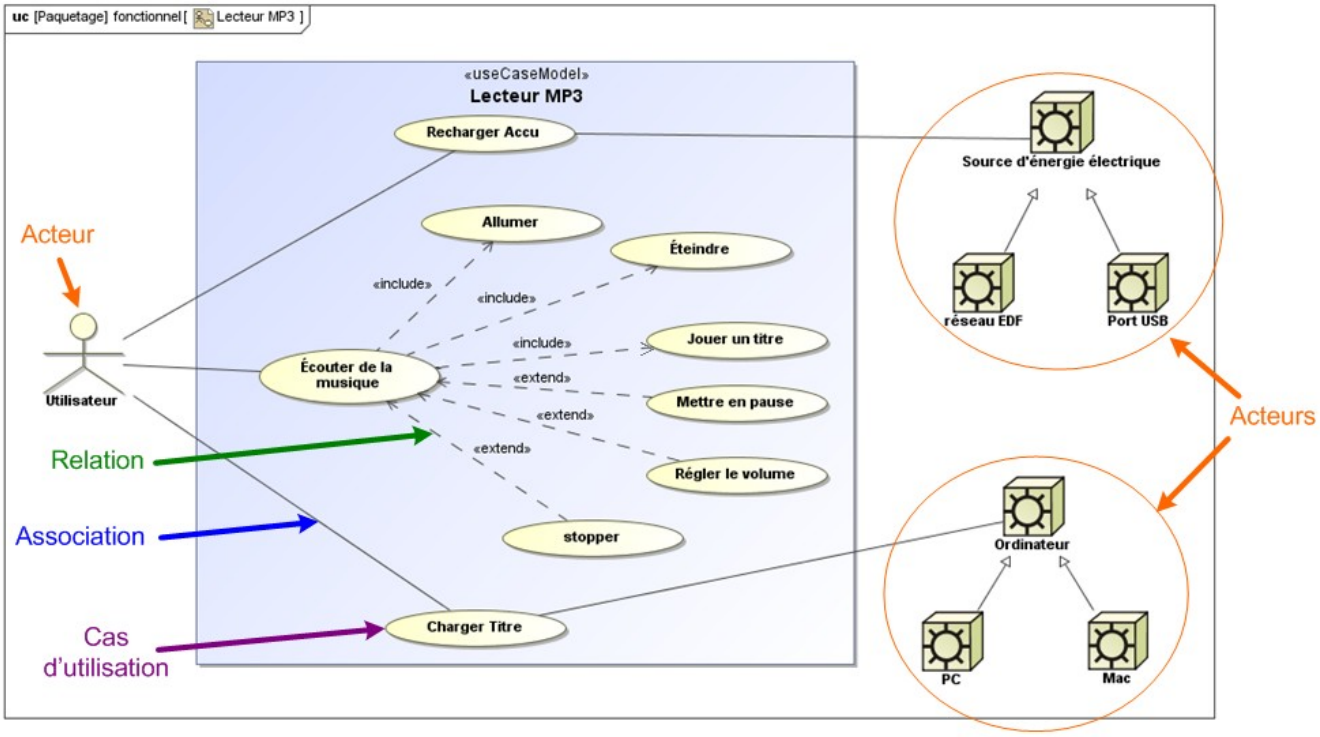
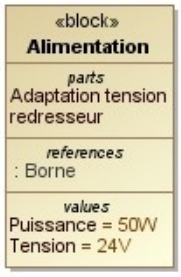


Diagramme de définition de blocs (bdd – block definition diagram)

Le diagramme de définition de blocs est similaire à la première page d'une notice de montage indiquant la liste des éléments et des pièces à assembler. Ainsi le bloc principal et la hiérarchie des blocs qui le composent sont spécifiés.

On peut distinguer différentes zones :



Définition du bloc : nom et type (« block » par défaut, « système » : niveau supérieur).

Part : dans certain cas, la zone *Part* permet de définir les parties qui composent le bloc.

Operations : comportements possibles, actions à réaliser.

Attributs ou *Values*: propriétés du bloc. Ça peut aussi être des valeurs (*values*) qui permettent de caractériser le bloc.

« **Bloc** » et son identification.

« **Composition** » : indique que les blocs du dessous font partie du bloc marqué par un losange plein.

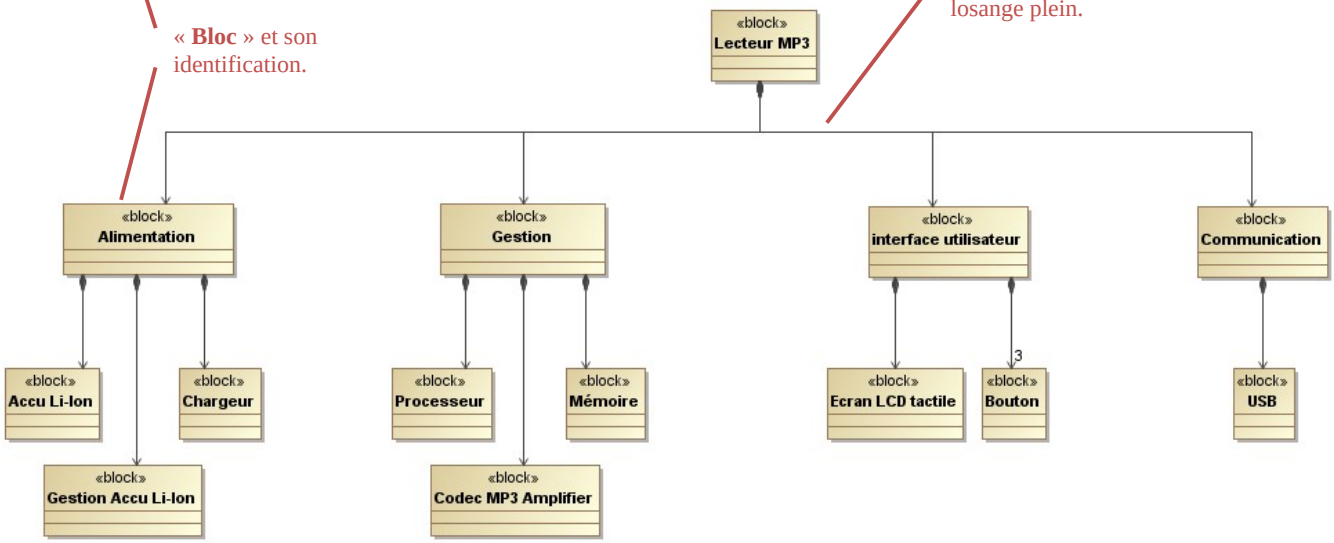
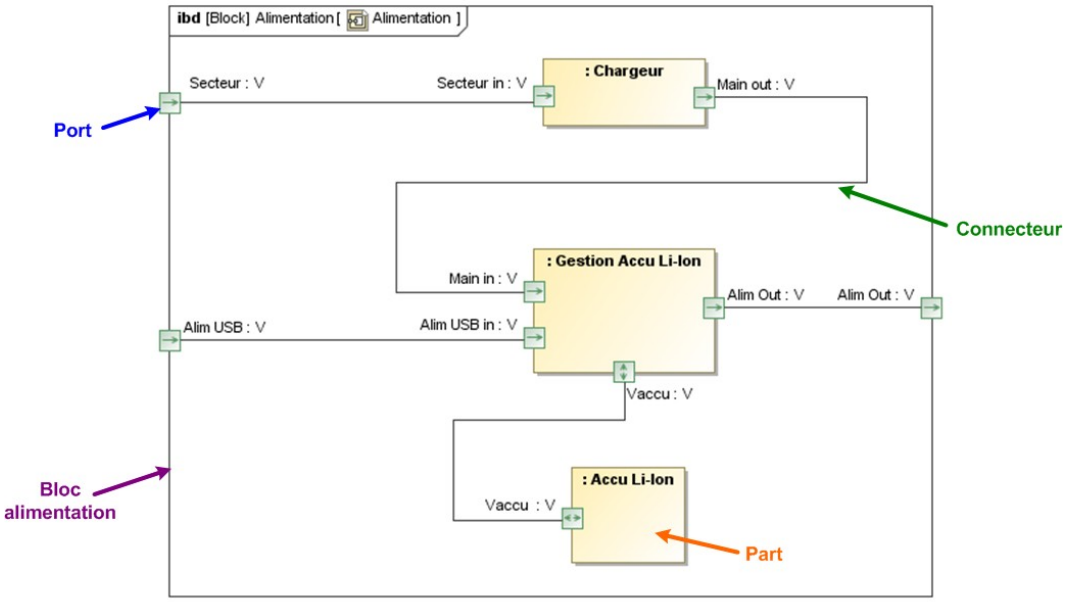


Diagramme de blocs internes (ibd – internal block diagram)

Le diagramme décrit la vue interne d'un bloc se basant sur le *bdd*. On effectue un zoom sur un bloc en apportant des précisions sur son organisation et sur la communication des différents éléments via les *flow ports*.
Un bloc peut avoir plusieurs ports unidirectionnel ou bidirectionnels, qui expriment la circulation de flux entre les blocs (énergie, fluides, données…).



FICHE-OUTIL 6

Diagramme de blocs internes

En anglais : *internal block diagram*
Notation SysML : *ibd*

C'est un diagramme statique.
Il est utilisé pour décrire l'architecture matérielle du système.

Il montre l'organisation interne d'un élément statique complexe.
Il représente les instances des *parts* d'un bloc (objets). L'IBD est cadré à l'intérieur des frontières du bloc concerné. Les circulations de flux (MEI) entre les *parts* s'effectuent grâce aux connecteurs qui relient leurs ports.
L'IBD d'un bloc est défini à partir du BDD correspondant. Un flux entre ou sort d'une *part* via un port.

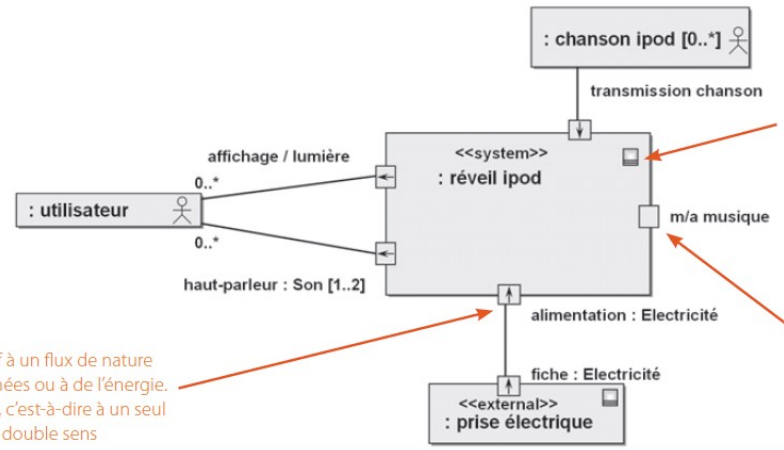
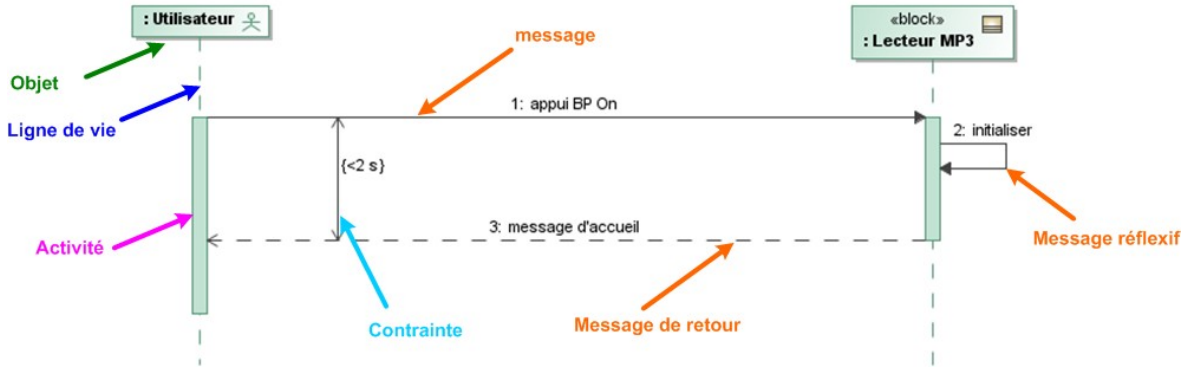


Diagramme de séquence (sd – sequence diagram)

Les cas d'utilisation permettent de décrire ce que le système doit être capable d'effectuer mais ils ne disent pas comment. Ce diagramme décrit le scénario des interactions dans le temps entre les acteurs et les objets. Il montre la chronologie des échanges issus **d'un cas d'utilisation**.



Chaque élément actif du système est représenté par un rectangle doté d'une « ligne de vie ». La chronologie des évènements se lit de haut en bas. Les lignes horizontales entre éléments sont des « messages », les messages étant des signaux, des événements ou des invocations d'opérations.

- un message synchrone (*flèche pleine*) bloque l'émetteur qui est dans l'attente d'une réponse.
- un message asynchrone (*flèche ouverte*) pour lequel l'émetteur n'attend pas de réponse pour continuer sa tâche.
- La flèche en pointillée représente un retour.
- La flèche qui boucle (*message réflexif*) permet de représenter un comportement interne.

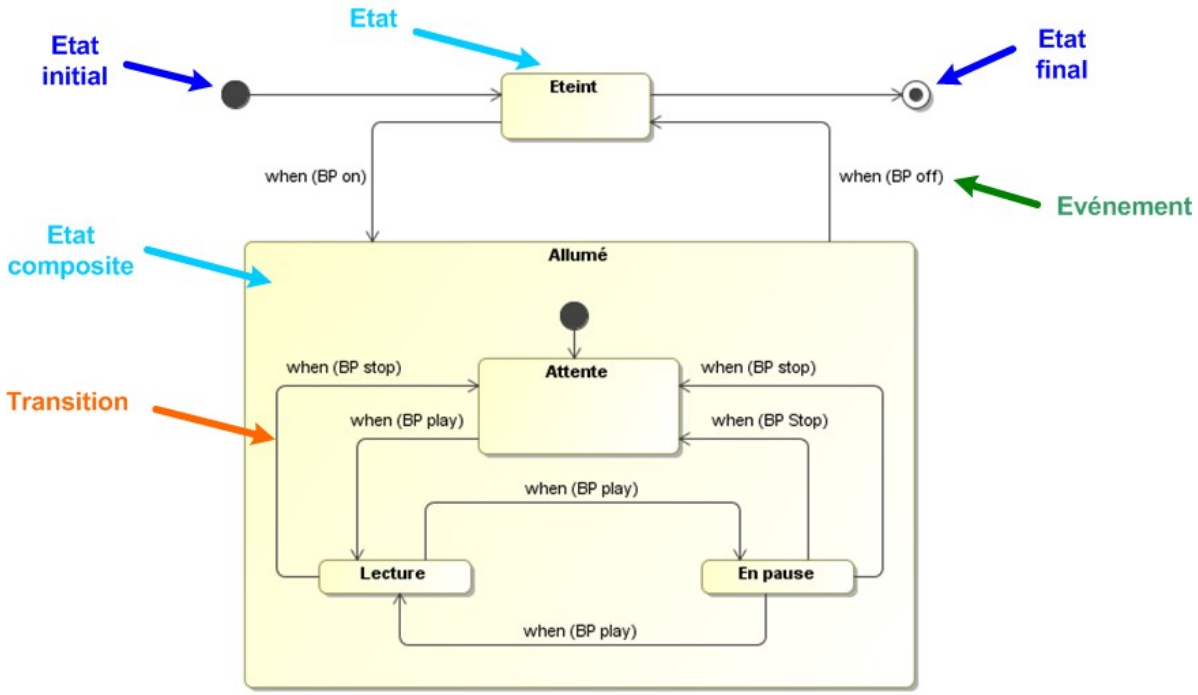
Dans un même diagramme de séquence, on peut indiquer différentes séquences possibles en fonction de conditions déclenchantes. On utilise pour cela un **fragment combiné**. Ces conditions sont appelées conditions de garde et sont indiquées entre crochet [ ].

Les principaux fragments combinés utilisés sont :

- **opt** (option) : le fragment s'exécute si la condition est vraie.
- **alt** (alternative) : en fonction de la condition, une partie seulement du fragment s'exécute.
- **loop** (boucle) : le fragment s'exécute en boucle selon la condition.
- **break** : si la condition est vraie, le fragment interrompt une boucle.
- **par** (parallèle) : les séquences séparées par une ligne pointillée s'exécutent en parallèle.

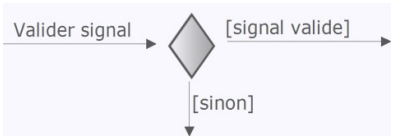
Diagramme d'état (stm – state machine)

Le diagramme d'état décrit les transitions entre les états et les actions que le système ou ses parties réalisent en réponse aux évènements. Il représente le fonctionnement séquentiel et permet de programmer un système.



La transition permet le changement d'état. Une transition peut être :

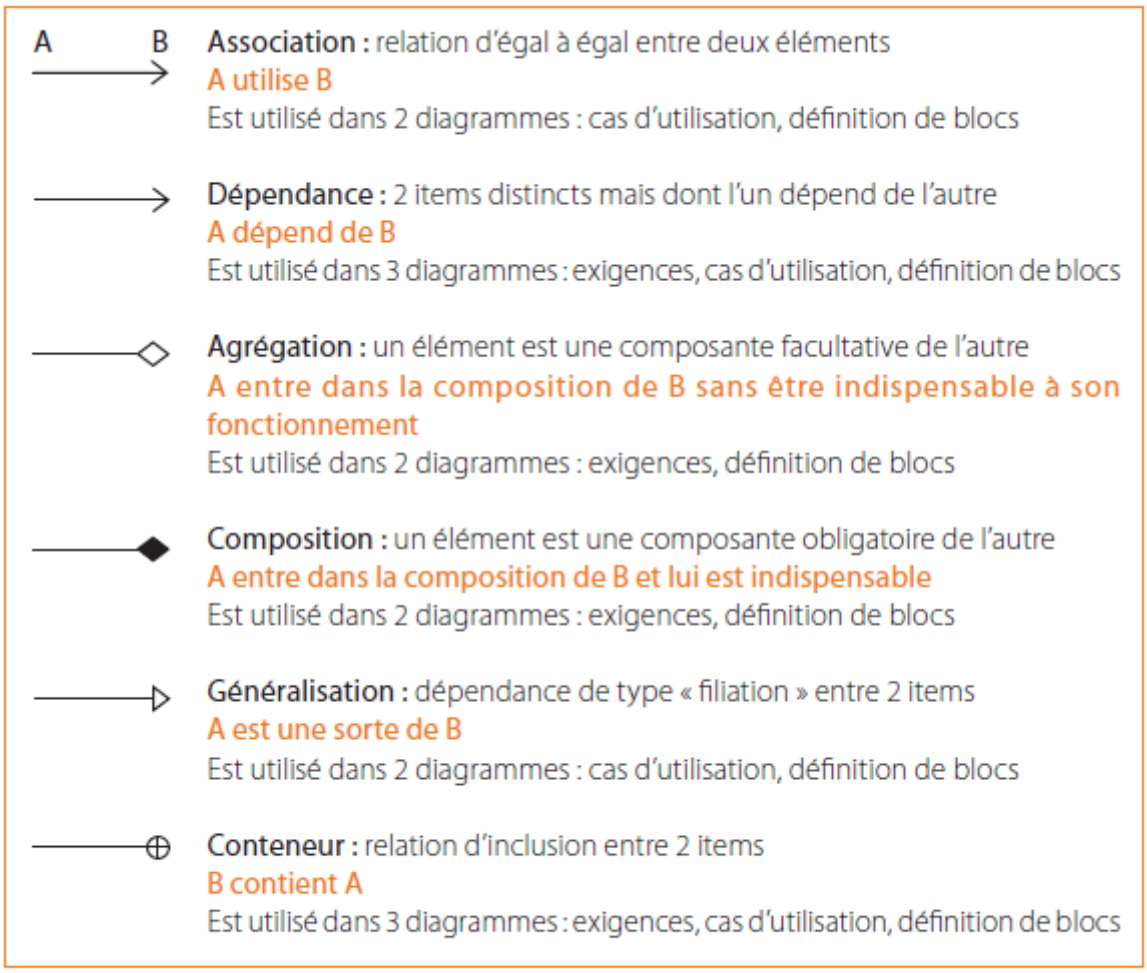
- automatique (flèche sans texte) : la fin de l'activité d'un état entraîne le passage à l'état suivant.
- sur évènement (when, after, etc ;) : l'occurrence de l'évènement entraîne le passage à l'état suivant.
- avec un point de décision comportant au minimum deux sorties, les critères de décision sont décrits entre crochets.



Les connecteurs

Selon le diagrammes, le langage SYSML propose différents types de connecteurs entre les élèments

Voici un tableau récapitulatif :



Les principales relations entre les blocs