

Langage SysML

Systems Modeling Language - SysML en abrégé - est un langage de modélisation spécifique au domaine de l'ingénierie système. Il permet la spécification, l'analyse, la conception, la vérification et la validation de nombreux systèmes et systèmes-de-systèmes. SysML se définit comme une extension d'un sous-ensemble d'UML (Unified Modeling Language).



1. Introduction

Les méthodes de l'Ingénierie Système (IS) reposent sur des approches de modélisation et de simulation pour valider les exigences, et pour vérifier ou évaluer le système. La modélisation a donc couramment été utilisée pour l'IS, que ce soit pour des représentations concrètes avec des plans ou modèles réduits, ou plus abstraites avec des systèmes d'équations.

En général l'IS tend à modéliser les aspects suivants du système: décomposition fonctionnelle, flux de données, et décomposition structurelle. Exemples de techniques de modélisation employées :

- Le diagramme de flux de données (DFD ou Data Flow Diagram) pour définir les données traversant un système et leurs traitements éventuels
- Le diagramme de flux fonctionnel de bloc (FFBD ou Functional Flow Block Diagram) proche du diagramme UML d'activité ou du « flowchart »

La modélisation avec le langage UML est une pratique bien établie dans l'industrie logicielle. Bien que le langage UML permette par son caractère à usage général d'adresser de nombreux besoins pour l'IS, il est nécessaire d'adapter ce langage de modélisation par la définition de « profils UML ».

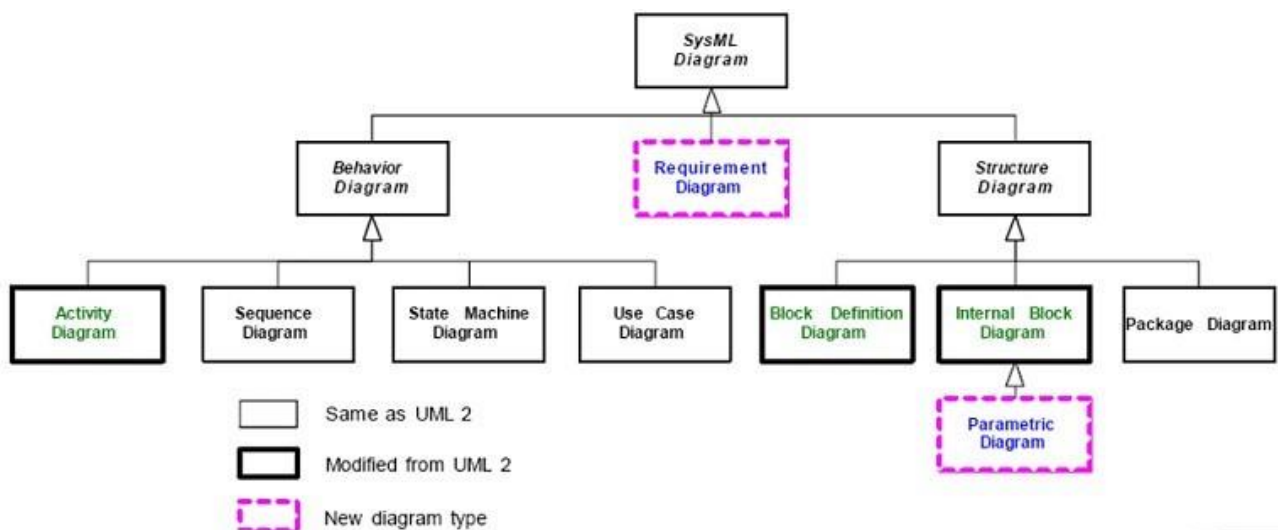
Le besoin de définir un langage basé sur UML pour l'IS a été initié en 2001 par l'organisation internationale de l'ingénierie système INCOSE (International Council on Systems Engineering).

SysML (Systems Modeling Language) est basé sur UML et remplace la modélisation de classes et d'objets par la modélisation de blocs pour un vocabulaire plus adapté à l'Ingénierie Système. Un bloc englobe tout concept logiciel, matériel, données, processus, et même la gestion des personnes.

Comme représenté sur le diagramme suivant, SysML réutilise une partie d'UML, et apporte également ses propres définitions (extensions SysML) : 4 structurels, 4 dynamiques, et le diagramme d'exigences.

Structurels :

- Le « Block Definition Diagram » (BDD) remplace le diagramme de classes
 - L' « Internal Block Diagram » (IBD) remplace le diagramme de structure composite
 - Le diagramme de paquetage reste inchangé
 - Le diagramme paramétrique est une extension SysML pour l'analyse de paramètres critiques du système
- Dynamiques :
- Le diagramme d'activités est légèrement modifié pour SysML
 - Les diagrammes de séquence, d'états, et de cas d'utilisations restent inchangés
- Le diagramme d'exigences est une extension SysML.



Ainsi SysML permet de fournir un référentiel central qui supporte les analyses du système requises par l'IS, à savoir la décomposition fonctionnelle, les flux de données, et la décomposition structurelle.

Chaque diagramme est nommé d'une façon bien précise et constitue un élément nommé du modèle.

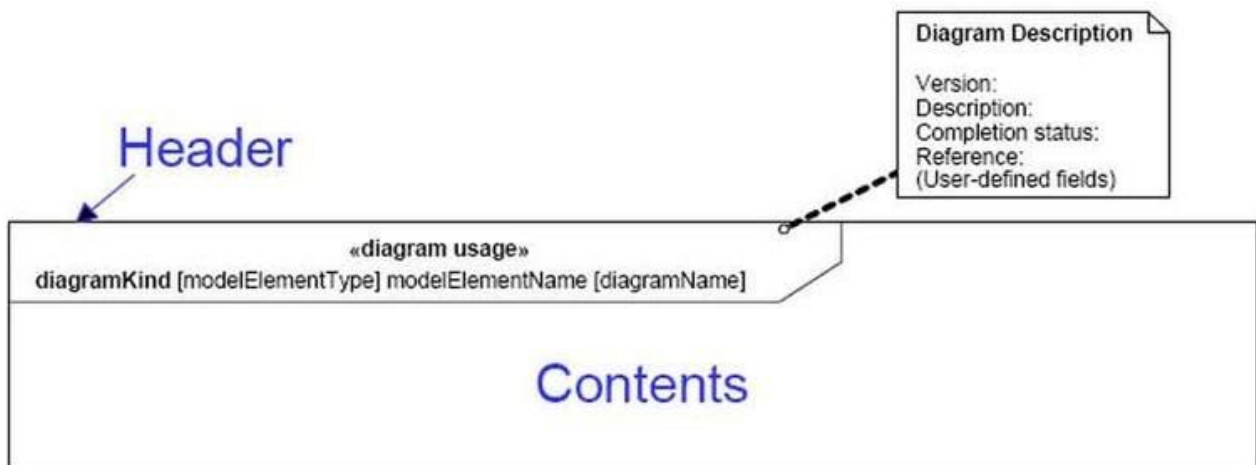
Pour cela SysML définit une en-tête standard à chaque diagramme qui contient obligatoirement :

- Le type de diagramme : act, bdd, ibd, sd, etc....
- Les éléments représentés dans le diagramme : packages, blocs, activités, etc....

- Le nom de l'élément modélisé.
- Le nom du diagramme ou de la vue représentée.

De plus, chaque diagramme dispose d'une description :

- Version.
- Description.
- Statut ou niveau d'avancement.
- Référence, etc...



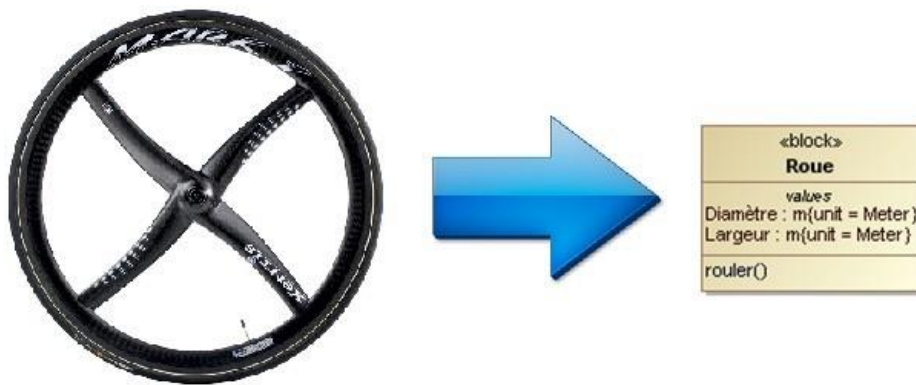
2. Présentation SysML

Les principaux concepts sur lesquels s'appuie le langage sysML pour modéliser les systèmes sont les suivants :

- Tout type d'élément faisant partie du système modélisé est décrit par un "block". Il est défini essentiellement par son nom, ses caractéristiques et les fonctions qu'il offre.



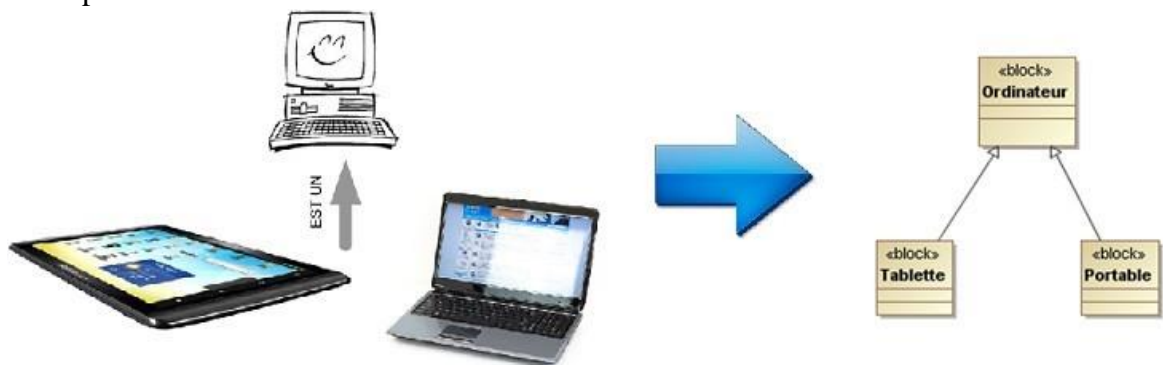
Exemple : le type "Téléviseur"



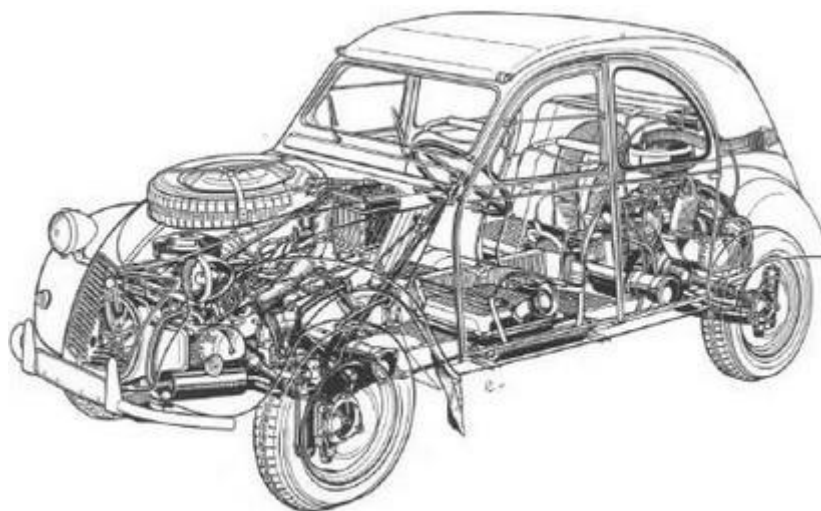
Exemple : le type "Roue" •

Un block peut être une spécialisation d'un block plus général.

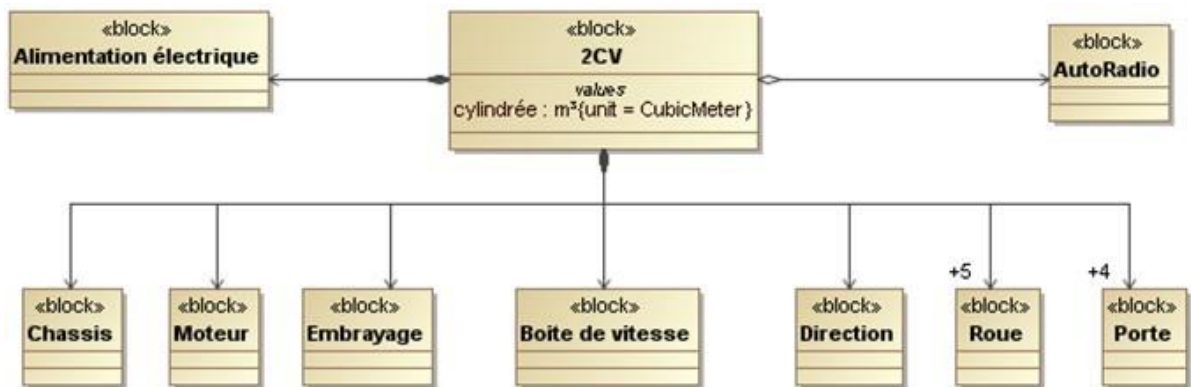
Exemple :



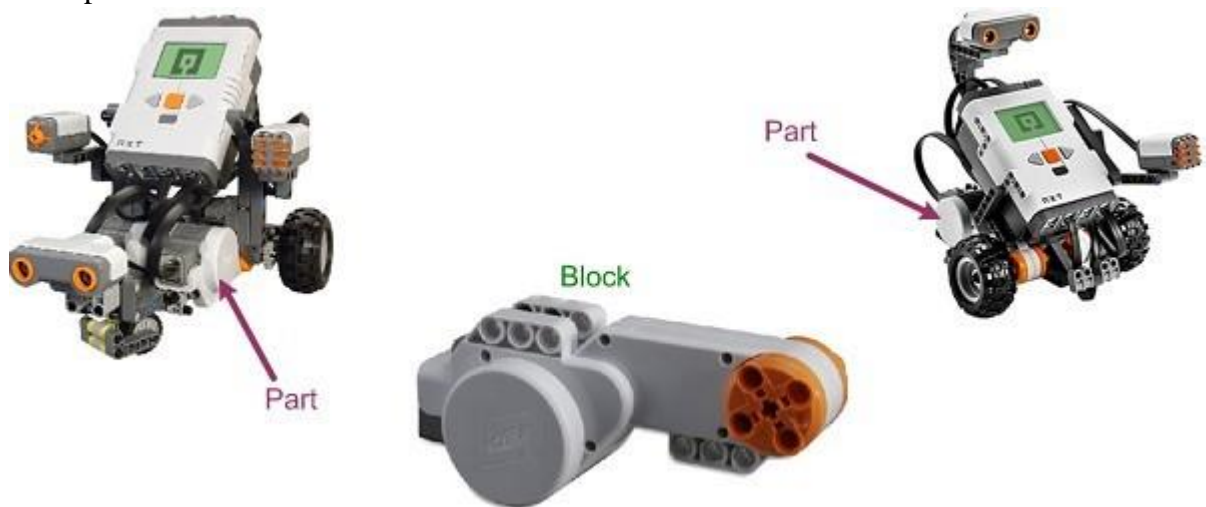
- Un block peut lui même être constitué de blocks (losange noir) et/ou susceptible d'en contenir certain (losange blanc). Exemple : un véhicule (2CV)



est symbolisé ainsi :



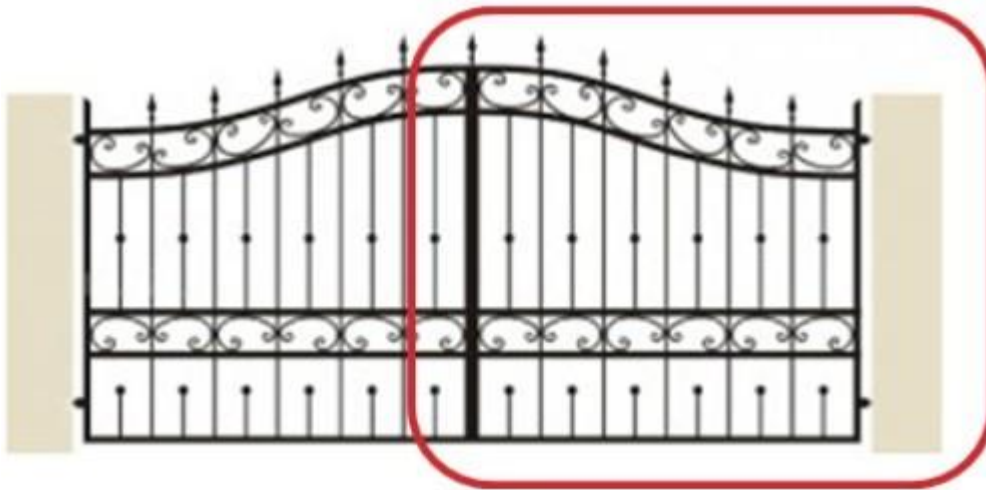
- Une "part" représente un block mis en œuvre dans un contexte donné.
Exemple : le block "servomoteur" du robot NXT



est symbolisé ainsi :



- Les parts sont liés entre eux pour réaliser une structure ou une fonction. Exemple : la partie droite d'un portail



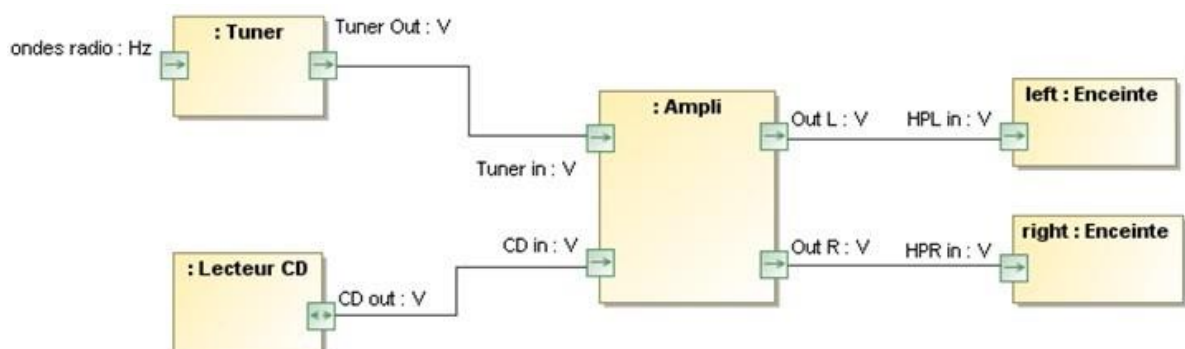
peut se représenter ainsi :



- Les parts interagissent par l'intermédiaire de ports (concept d'entrée/sortie). Exemple :
L'interconnexion des éléments d'une chaîne HI FI

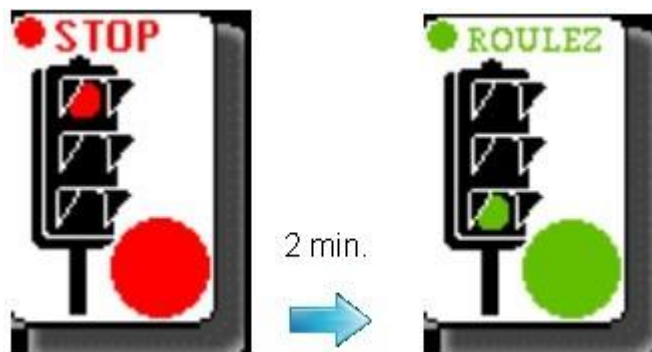


est représentée ainsi :

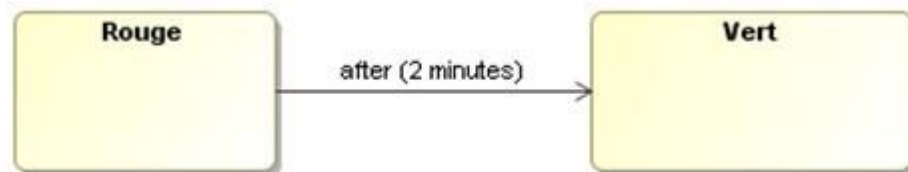


Remarque : Les parts "left" et "right" représentent 2 instances (exemplaires) du block "Enceinte".

- L'état d'un block évolue au fil des événements qui se produisent. Exemple : l'évolution de l'état d'un feu de croisement



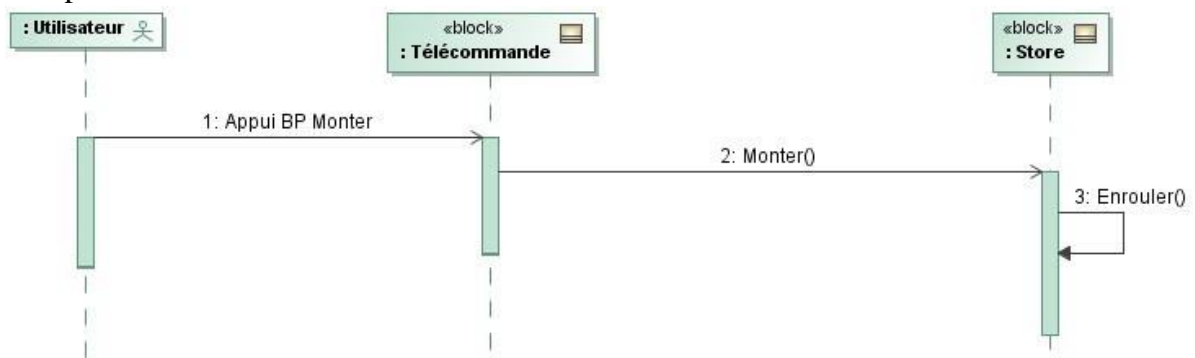
est modélisé ainsi :



- Les parts communiquent entre eux par l'envoi de message. La réception d'un message provoque le déclenchement d'une activité. Exemple : Dans le scénario suivant, l'ordre "monter" envoyé par la télécommande lors d'un appui sur le bouton poussoir, déclenchant l'enroulement du volet roulant motorisé.



se représente ainsi :



Le langage sysML propose 9 types de diagrammes destinés à représenter les aspects fonctionnel, structurel et comportemental d'un système.

Pour modéliser l'aspect fonctionnel :

- Le diagramme des cas d'utilisation (UCD)

- Le diagramme des exigences (RD)

Pour représenter l'aspect structurel :

- Du diagramme de définition de blocks (BDD)
- Du diagramme de block interne (IBD) Pour modéliser l'aspect comportemental :
- Le diagramme de séquence (SD)
- Le diagramme d'état (STM)

3. Description fonctionnelle

Trois diagrammes interviennent dans la modélisation fonctionnelle d'un système.

3.1. Le diagramme des cas d'utilisation (UCD)

Tout comme avec UML, on utilise les diagrammes de cas d'utilisations :

- Basé sur les interactions acteurs/système, pour identifier les acteurs et les cas d'utilisations d'un point de vue utilisation du système
- Pour choisir et identifier les frontières, le périmètre fonctionnel du système

Une description textuelle pour chacun des cas d'utilisations peut être rédigée avec les pré conditions, post conditions, scénario nominal, scénarios alternatifs et d'erreur.

- Il délimite la frontière entre ceux qui interagissent avec le système, les acteurs (humains, systèmes, flux d'énergie, de matière, etc.), et le système lui même ;
- Les cas d'utilisation présentent de façon organisée les fonctionnalités métier attendues du point de vue de l'utilisateur final ;
- Il associe les cas d'utilisation aux acteurs concernés. • Chaque cas d'utilisation fait l'objet d'une description soit textuelle, soit par un diagramme comportemental (diagramme d'état ou de séquence).

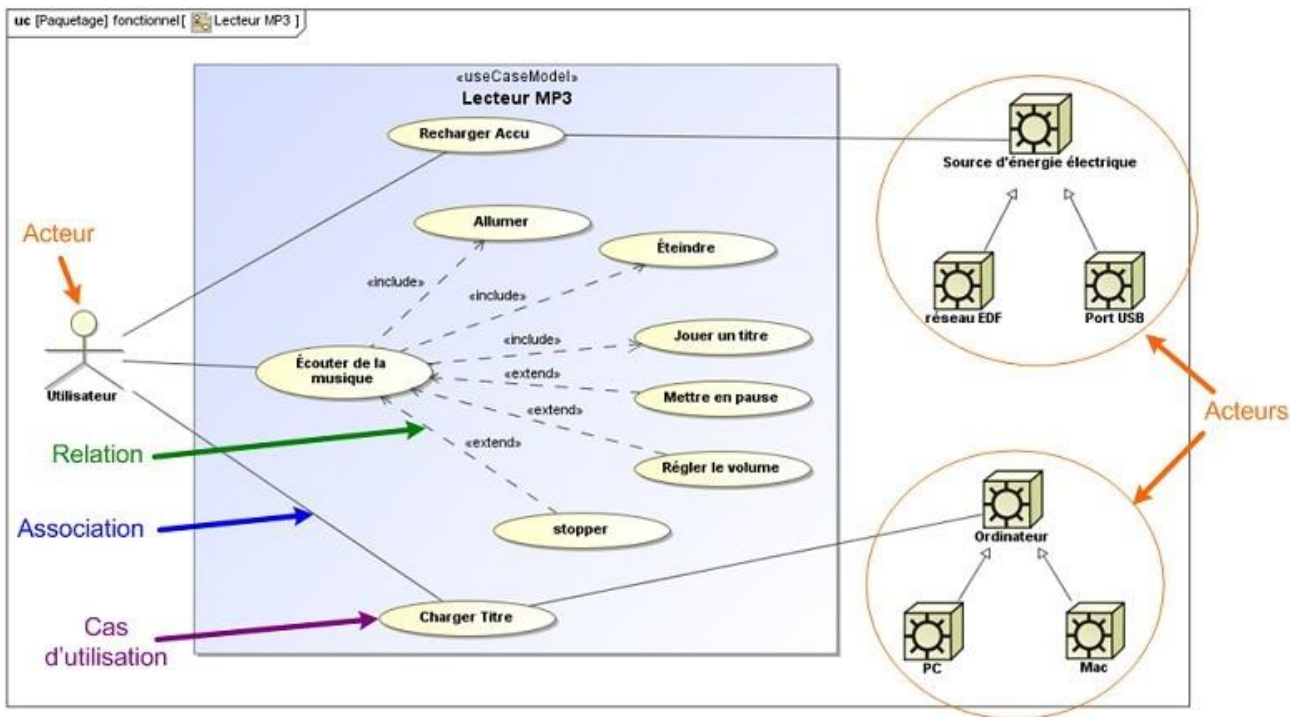
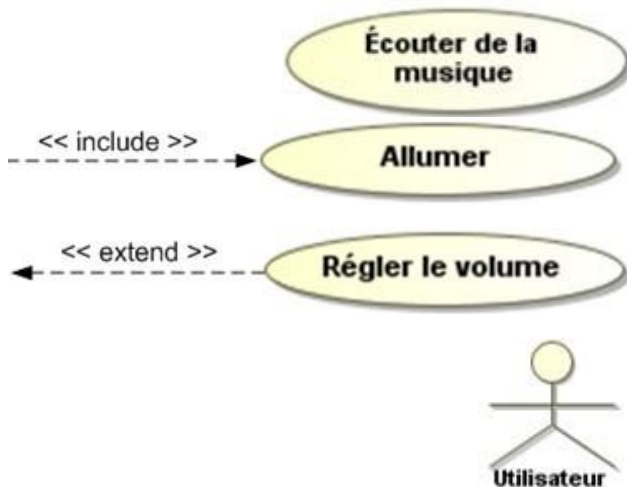


Diagramme des cas d'utilisation d'un lecteur MP3 (partiel)



Représente un service, une fonctionnalité attendue de l'utilisateur. Il symbolise son déroulement.

La relation «include» indique que "Allumer" fait partie du déroulement de "Écouter de la musique".

La relation «extend» indique que "Régler le volume" peut éventuellement se dérouler durant "Écouter de la musique".

Symbolise tout ce qui interagit avec le système qu'il s'agisse d'un utilisateur, d'un système, d'un flux d'énergie ou de matière.

Un cas d'utilisation représente les interactions entre un acteur et le système d'un point de vue « boîte noire », et comprend l'ensemble des scénarios identifiés (nominal, alternatifs, et d'erreurs). Ces scénarios peuvent être détaillés par une description textuelle et/ou par un diagramme de séquence.

3.2. Le diagramme des exigences

Que ce soit pour l'ingénierie système ou pour des réalisations logicielles, les exigences sont couramment utilisées pour formaliser les pré-requis du système, se traduisant par des fonctionnalités ou conditions qui doivent ou devraient être satisfaites par le système (selon les éventuelles priorités associées aux exigences).

Pour la maîtrise d'ouvrage (MOA), les exigences ont pour objectif d'assurer l'adéquation de la solution (le système réalisé) avec les besoins. Les exigences peuvent être formalisées et

catégorisées par centre d'intérêts ou aspects, par exemple différenciant les exigences fonctionnelles des exigences techniques (performance, fiabilité, ergonomie, etc.).

La formalisation des exigences peut être effectuée avec une feuille Excel, ou avec un outil spécialisé tel que DOORS ou CaliberRM. L'intérêt qu'offrent ces outils est la gestion des exigences dans une organisation structurée. Les exigences sont également utilisées pour la modélisation, par la création d'associations entre exigences et cas d'utilisations, blocs ou tout type d'élément du modèle, établissant une traçabilité. Il est possible avec l'outil Enterprise Architect de définir les exigences ou de les importer depuis un outil tel que DOORS, et de les associer avec les éléments du modèle.

SysML formalise les exigences et leur représentation, s'inspirant des fonctionnalités des outils actuellement disponibles sur le marché, par exemple EA, DOORS, CaliberRM. Ainsi SysML définit

une représentation graphique et visuelle des exigences textuelles, permet une organisation hiérarchique et l'association avec les éléments du modèle.

SysML définit de nouveaux types de d'associations (liens de dépendance stéréotypés) :

- Derive : une ou plusieurs exigences sont dérivées d'une exigence
- Satisfy: un ou plusieurs éléments du modèle (par exemple un bloc) permettent de satisfaire une exigence
- Verify: un ou plusieurs éléments du modèle (par exemple un « test case ») permettent de vérifier et valider une exigence
- Refine: un ou plusieurs éléments du modèle, par exemple un cas d'utilisation, redéfinit une exigence

SysML définit de nouveaux commentaires stéréotypés permettant d'associer une explication à des associations ou éléments du modèle :

- Problem: commentaire dont la description pose le problème ou le besoin qui a donné lieu à la création de l'association ou de l'élément associé
- Rationale: commentaire dont la description indique la raison ou la justification par rapport à l'élément ou l'association associé

Exemple d'associations et de commentaires des spécifications officielles de SysML (OMG) :

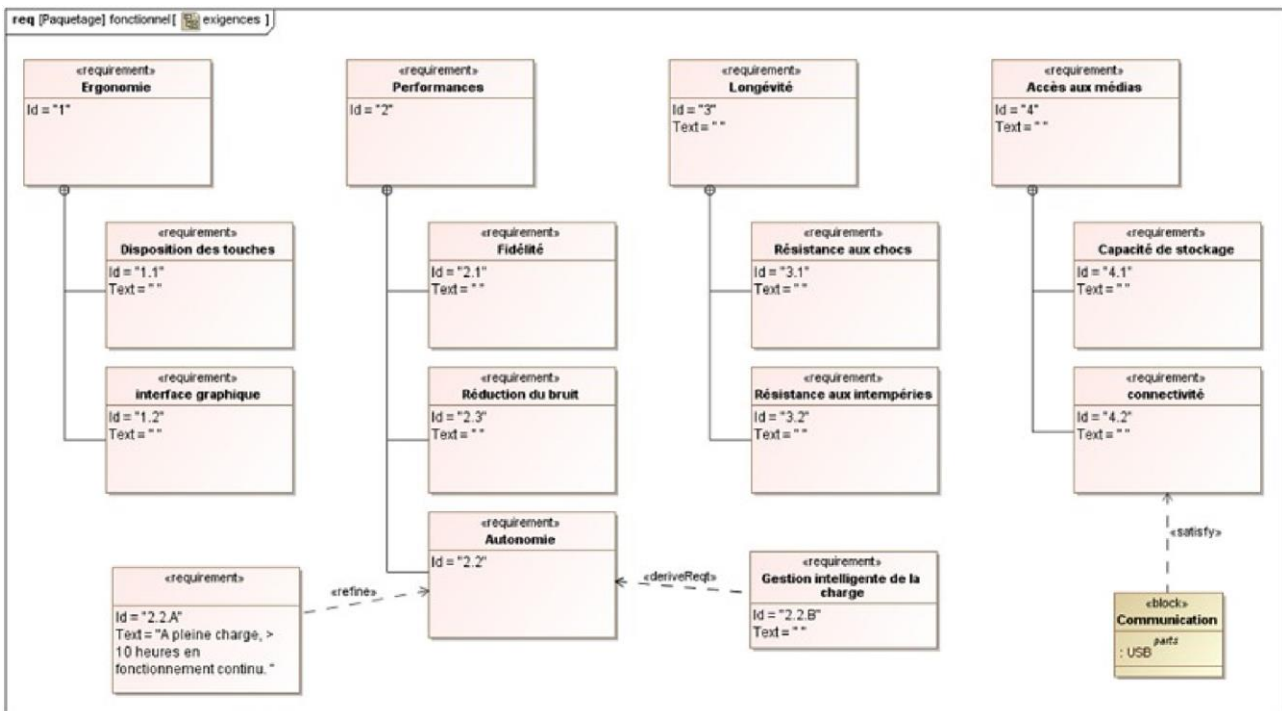
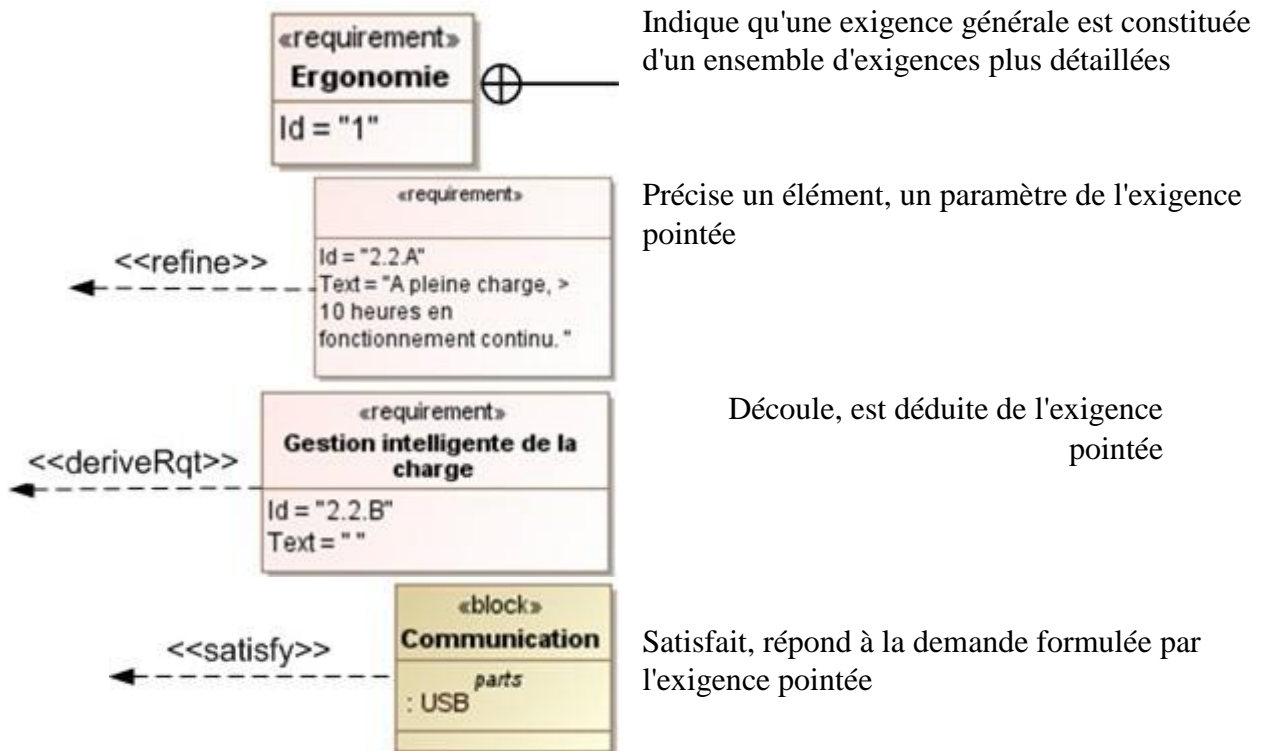


Diagramme d'exigence partiel d'un lecteur MP3



3.3. Le diagramme de contexte

Il complète éventuellement la description fonctionnelle en présentant tous les éléments externes qui influencent le système étudié et le système lui-même.

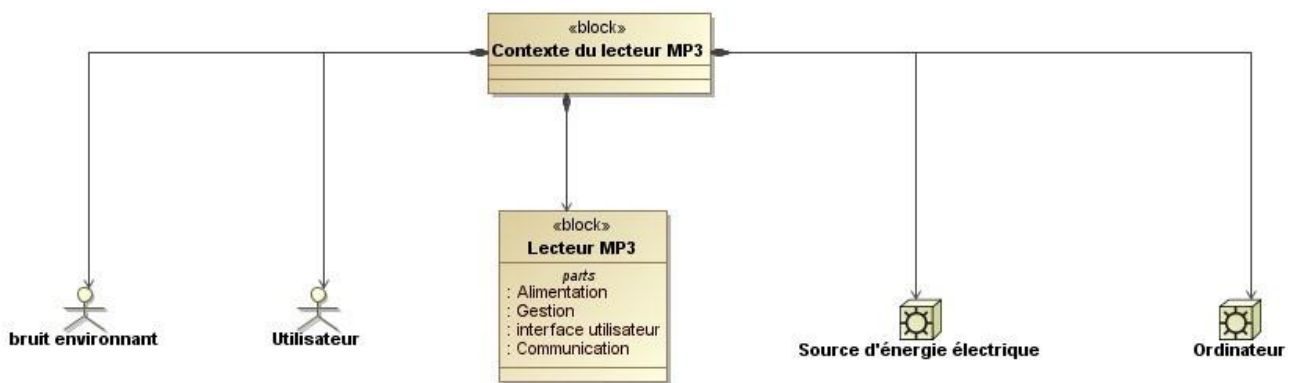


Diagramme de contexte d'un lecteur MP3

4. Description structurelle

Le langage sysML propose 2 diagrammes destinés à décrire la composition du système.

4.1. Le diagramme de définition des bocks (BDD)

Le diagramme de définition de bloc (BDD, ou Block Definition Diagram en anglais) représente la vue boîte noire d'un bloc. Ainsi le bloc principal et la hiérarchie des blocs qui le composent, qu'ils soient logiciels ou matériels, sont spécifiés dans ce diagramme.

Par rapport à UML, le BDD de SysML redéfinit le diagramme de classe en remplaçant les classes par des blocs.

Le BDD est similaire à la première page d'une notice de montage d'un meuble, indiquant la liste des éléments et des pièces à assembler avec leurs quantités respectives.

Il répertorie les constituants du système ou d'un block en précisant éventuellement leur rôle et leur quantité. Chaque block peut faire l'objet d'une description plus précise en indiquant ses constituants, ses propriétés, les opérations qu'il peut effectuer ainsi que les contraintes ou limites auxquelles il est soumis. Exemple :

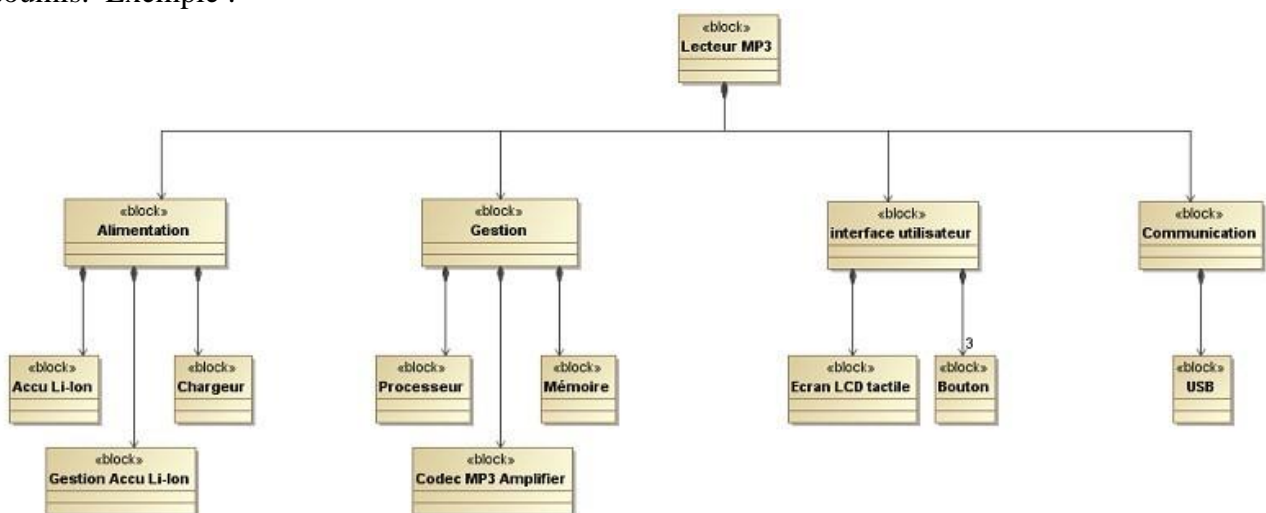


Diagramme de définition de blocks d'un lecteur MP3

Remarque : Dans le cas présent, Le lecteur MP3 dispose d'une interface utilisateur constituée d'un écran LCD tactile et de 3 boutons.

4.2. Le diagramme de block interne (IBD)

Le diagramme de bloc interne (IBD, ou Internal Block Diagram) décrit la vue interne d'un bloc ou vue boîte blanche, et se base sur le BDD pour assembler les blocs qui composent le bloc principal. Il représente les liens, les flux et les informations échangées entre les parties d'un block ou du système. Le cadre du diagramme représente le block lui-même ou le système.

Le bloc principal peut être représenté comme conteneur sur l'IBD ou être absent de ce diagramme. Les blocs qui le composent, définis dans le BDD, sont instanciés en parties (tout comme les parties dans un diagramme de structure composite avec UML2). Ces parties sont assemblées par des connecteurs qui relient leurs ports (ports standards avec interfaces exposées et/ou ports de flux).

Il est également possible de relier des parties directement entre elles, l'utilisation des ports étant optionnelle.

Par rapport à UML2, l'IBD de SysML redéfinit le diagramme de structure composite en ajoutant entre autre les ports de flux (« flow ports » en anglais).

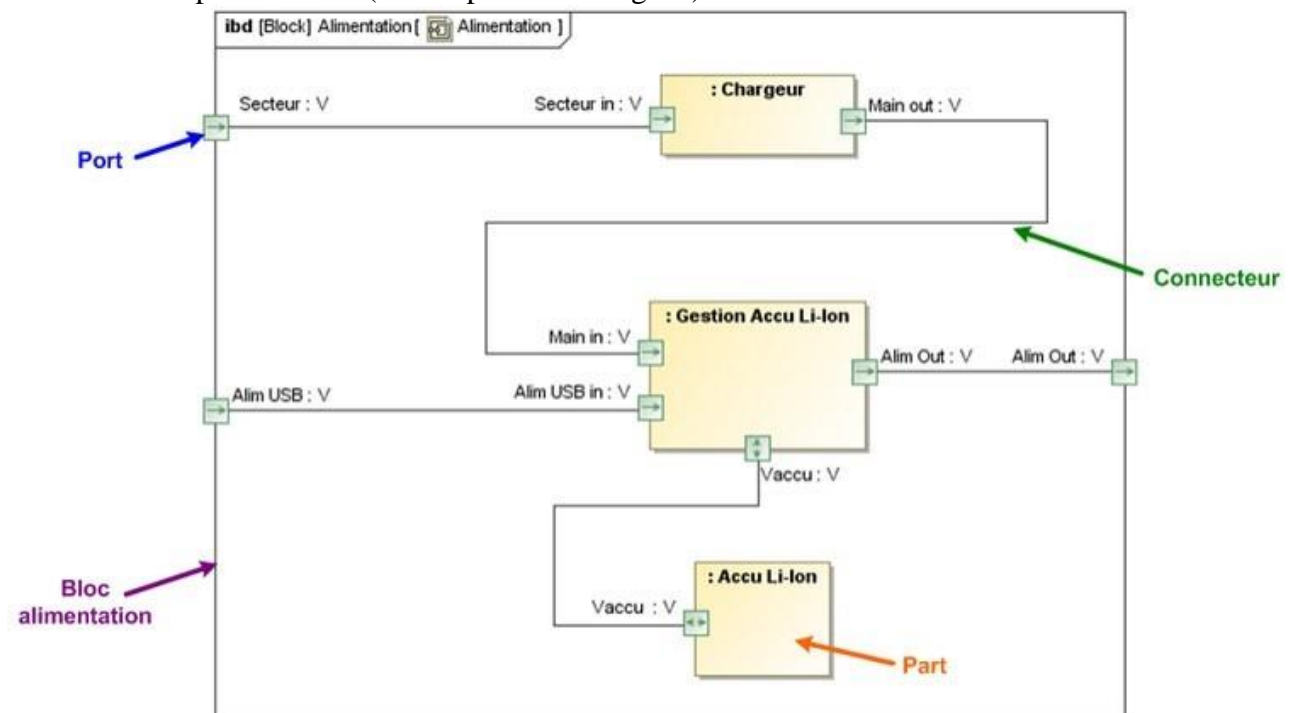




Diagramme de block interne du Block Alimentation (Lecteur MP3)

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

- ⑦ 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.
- ⑦ 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.

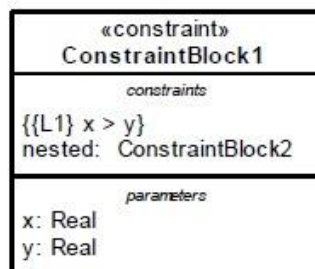
4.3. Le diagramme paramétrique

Le diagramme paramétrique est une nouveauté SysML qui redéfinit le diagramme interne de bloc SysML (lui-même basé sur le diagramme de structure composite UML2), et permet d'intégrer des analyses systèmes (performance, fiabilité, etc.) avec des blocs de contrainte.

Un bloc de contrainte représente une expression mathématique dont les paramètres peuvent faire référence à des éléments du système, par exemple des propriétés de blocs.

Exemples de blocs de contraintes : $\{F=m*a\}$, $\{a=dv/dt\}$

Dans un premier temps, de façon similaire à la création du diagramme BDD, les blocs de contraintes sont définis dans un diagramme de classe, et représentés comme suit :



Après avoir défini les blocs de contrainte, il faut générer un diagramme paramétrique :

- Des blocs de contraintes sont instanciés, donnant lieu aux propriétés de contrainte (ou `constraintproperty`), héritant ainsi des paramètres du bloc de contrainte (note : il n'y a pas de différenciation entre paramètres d'entrée et paramètres de sortie)
- Des propriétés systèmes (optionnellement liées à des blocs)
- Des connecteurs reliant les propriétés systèmes aux paramètres des propriétés de contrainte, ou reliant uniquement des paramètres des propriétés de contrainte

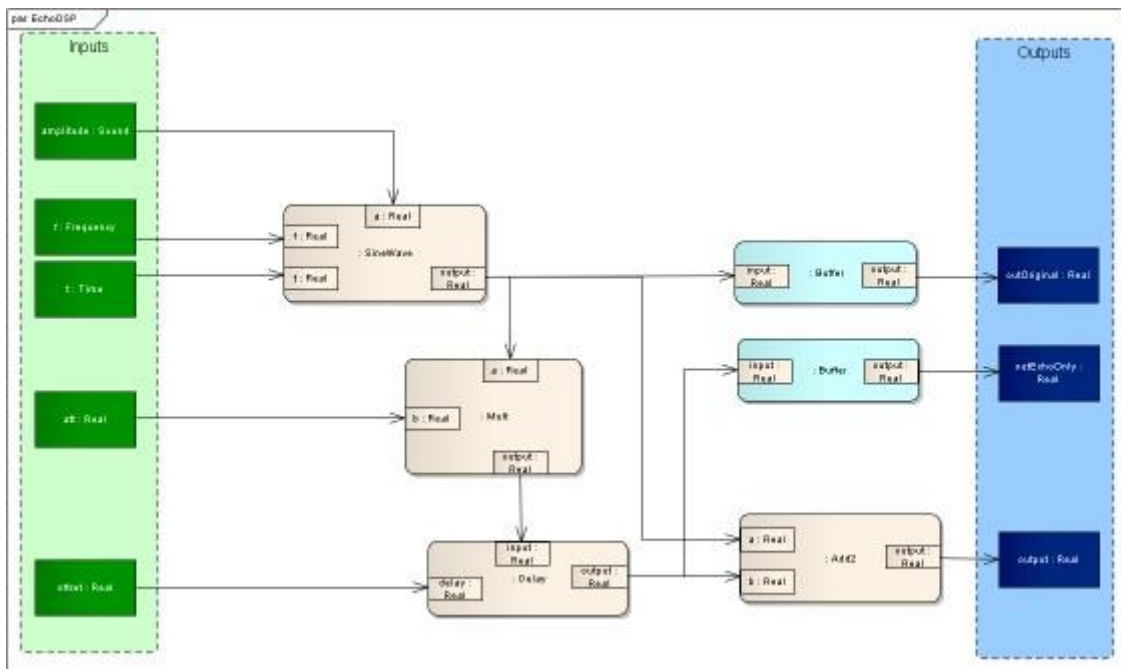


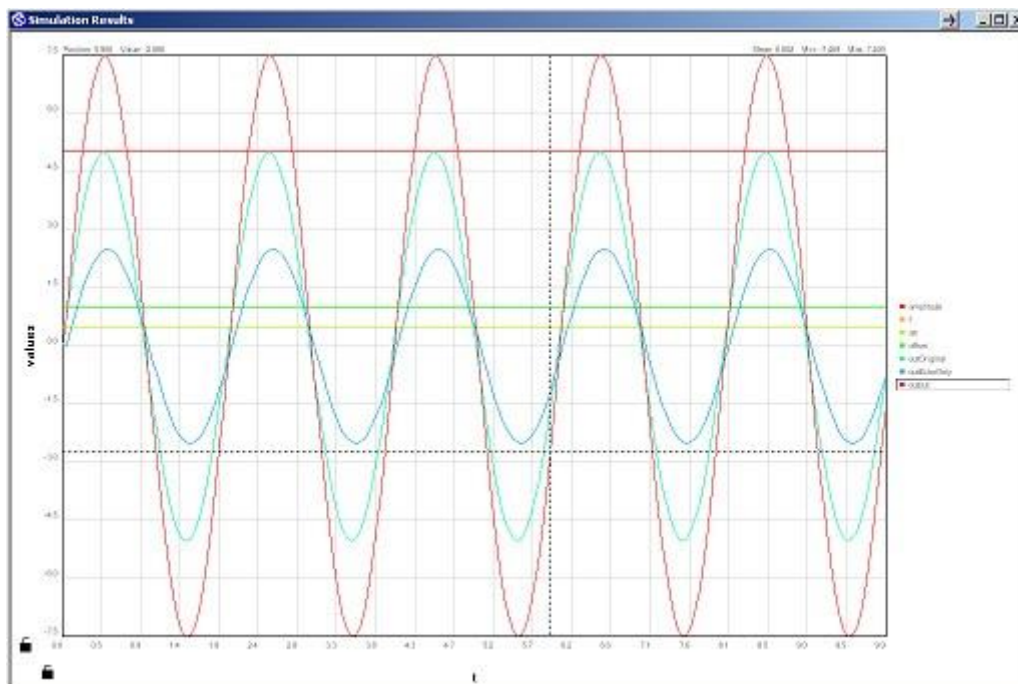
Diagramme paramétrique d'un lecteur MP3

Informations liées au diagramme paramétrique :

- Le diagramme comporte six propriétés de contrainte, dont deux propriétés instanciées du même bloc de contrainte : Buffer.
- Les paramètres systèmes sont catégorisés entre entrées et sorties (ex : amplitude et fréquence en entrée, output en sortie). Ces paramètres sont associés aux paramètres de propriétés de contrainte (par exemple la fréquence d'entrée f est reliée au paramètre f de la propriété de contrainte SineWave) • Certaines propriétés de contrainte sont reliées entre elle (ex : le paramètre SineWave.output est relié à Buffer.input et à Add.a)

Avec certains outils, ce diagramme peut être exécuté dans le cadre de simulation du système.

Il est possible sous Enterprise Architect de saisir des expressions pour chacun des blocs de contraintes (par exemple en VBScript ou JavaScript), ainsi que de renseigner différentes valeurs pour les paramètres systèmes. Ces expressions et valeurs peuvent être alors exécutées par le module de simulation Enterprise Architect comme illustré ci-dessous :



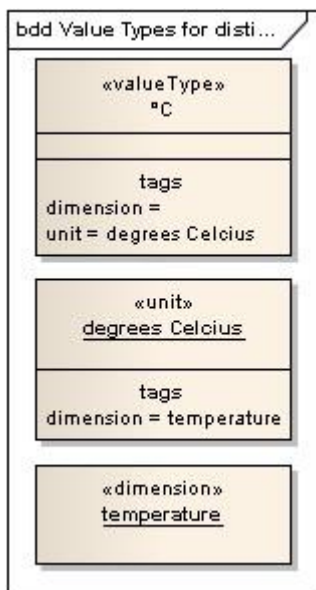
4.4. Value Types

Les Value Types sont une nouveauté SysML pour définir des types de valeurs réutilisables pour des propriétés du modèle, par exemple des blocs. De façon similaire à la modélisation UML où les attributs de classes peuvent être typés par d'autres classes, SysML permet de définir des propriétés de blocs typées avec des Value Type.

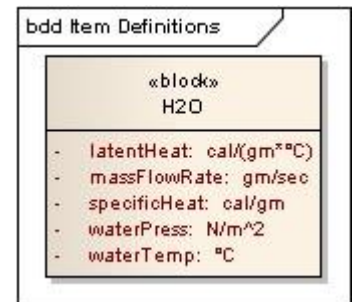
La Value Type a la particularité de contenir 2 propriétés optionnelles : une unité et une dimension. Les unités et dimensions sont des classes spécialisées et définies dans le modèle, par exemple :

- Value Type « Distance » avec unit = mètres et dimension = longueur. L'unit « mètres » est définie avec dimension = nulle.
- Value Type « °C » avec unit = degrés Celsius et dimension = nulle. L'unit « degrés Celsius »

est définie avec dimension = température :



L'exemple ci-contre montre l'utilisation de la valeur type « °C » par une propriété du bloc H2O (waterTemp : °C).



5. Description comportementale

Pour modéliser l'aspect dynamique d'un système et/ou de ses constituants, le langage sysML propose des diagrammes de comportement parmi lesquels on trouve :

5.1. Le diagramme de séquence (sd)

Le diagramme de séquence représente les éléments intervenant dans le scénario, ainsi que les échanges 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.

Dans un premier temps, on peut choisir de représenter les interactions entre l'acteur et le système (vue boîte noire). Par la suite il est possible de rentrer en détails avec un diagramme de séquence qui représente les blocs internes du système intervenant dans un scénario (pour un message émis par l'acteur, le diagramme décrit l'enchaînement des messages échangés entre les blocs internes du système) ; on parle ainsi de la vue boîte blanche (comportement du système).

Les éléments intervenant sont représentés par des lignes de vie (lifetime en anglais).

Ces lignes de vies peuvent être des instances de blocs, établissant un lien avec la modélisation structurelle du système, permettant ainsi d'établir une cohérence dans le modèle :

- On peut accéder aux propriétés du bloc d'une ligne de vie (et également aux diagrammes de BDD et d'IBD de ce bloc)
- Chaque message échangé peut être utilisé pour l'identification des opérations de blocs

L'ensemble des propriétés du diagramme de séquence utilisées en UML sont également disponibles avec SysML : messages synchrones ou asynchrones, opérateurs (ex : alt, loop, opt, par), références vers d'autres diagrammes de séquence (ex : naviguer de la vue boîte noire du scénario vers la vue boîte blanche), etc.

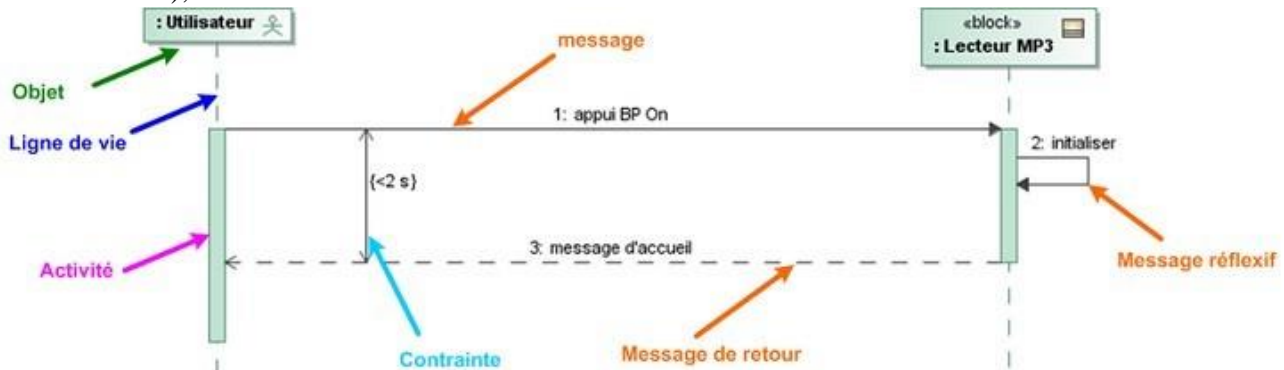


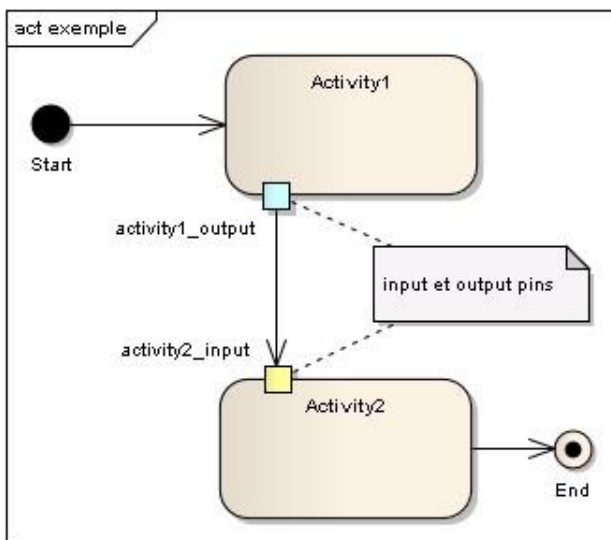
Diagramme de séquence du cas d'utilisation "Allumer" du lecteur MP3

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



5.2. Le diagramme d'activité

Le diagramme d'activité est utilisé pour représenter les étapes d'un traitement. Avec SysML, les « input et output pins » sont particulièrement utilisés pour représenter le type d'élément qui est requis en entrée d'une activité ou action, et ce qui est généré en sortie.



Si une action ou activité représente l'opération d'un bloc, on peut garantir la cohérence du modèle en s'assurant que ce qui est défini en entrée d'une activité corresponde à la signature de l'opération du bloc ou de son interface.

Toutes les propriétés des diagrammes d'activités UML sont également disponibles avec SysML.

SysML a rajouté quelques spécificités :

- Notion de contrôle pour activer ou désactiver les actions en cours.
- Spécification de la nature du débit sur le flot : système continu ou discret

- Définition de taux et de probabilité sur les flots

5.3. Le diagramme d'état (stm)

Le diagramme d'états est utilisé avec SysML de la même manière qu'avec UML2, c'est-à-dire qu'il permet de représenter le cycle de vie auquel doivent se conformer toutes les instances d'un bloc donné, ce au travers de la modélisation de tous les états possibles. Il modélise l'évolution de l'état d'un bloc en fonction des événements qui peuvent se produire.

Seuls les blocs qui sont importants d'un point de vue métier, ou qui sont de nature complexe, possèdent un diagramme d'état. Exemple :

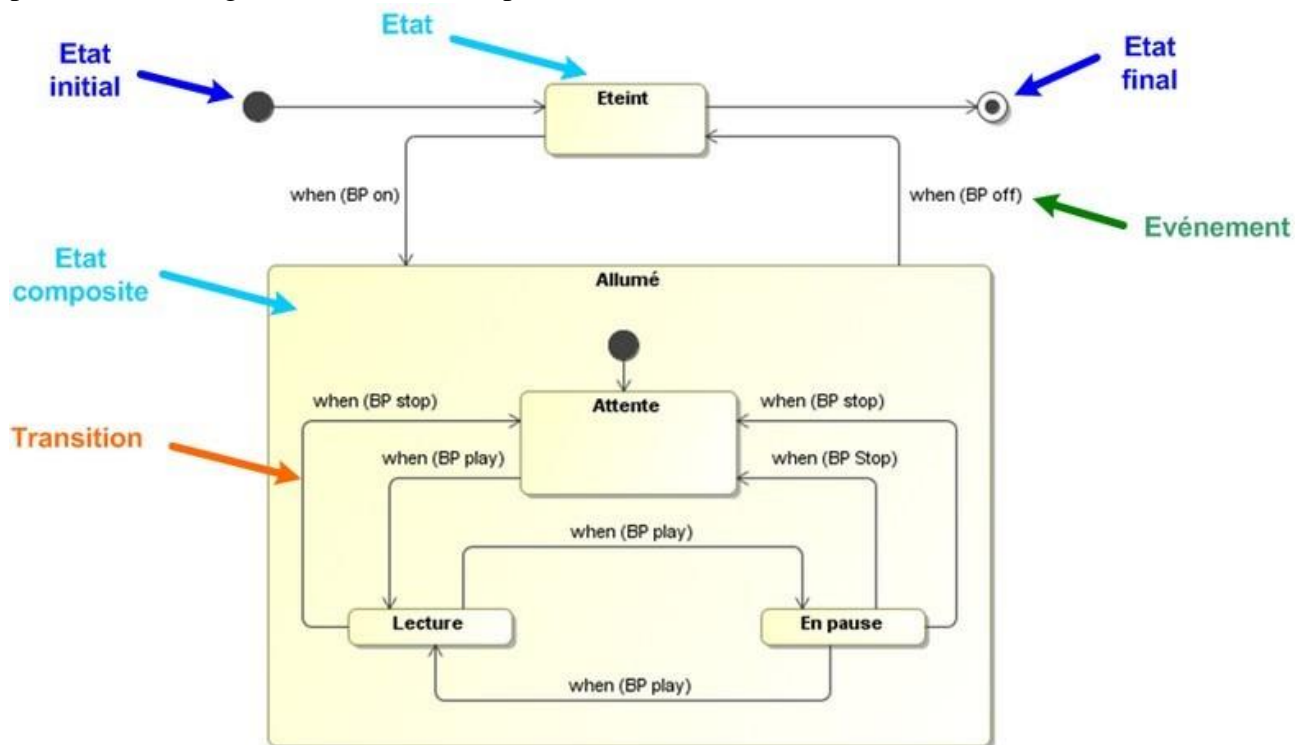


Diagramme état - transition partiel d'un lecteur MP3

Les propriétés du diagramme d'état UML sont également disponibles avec SysML : conditions sur événements, effets, activité durable, transitions, états composites, modularité, régions concurrentes, etc.

6. Approche SysML

Le SysML est un langage et non une méthode. Il est néanmoins possible d'établir un ordre pour comprendre et expliquer le fonctionnement des systèmes.

