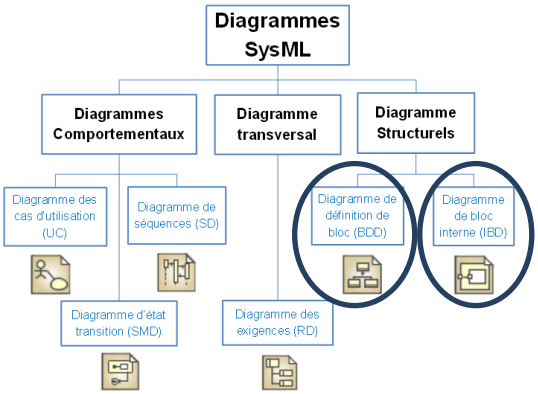
INGENIERIE SYSTEMES :
Compréhension - Exploitation des diagrammes SysML (SystemsModelingLanguage)

La modélisation structurelle

La modélisation structurelle d'un système est définie par :

- Le diagramme de définition de blocs
- Le diagramme de blocs internes



1. Diagramme de définition de blocs

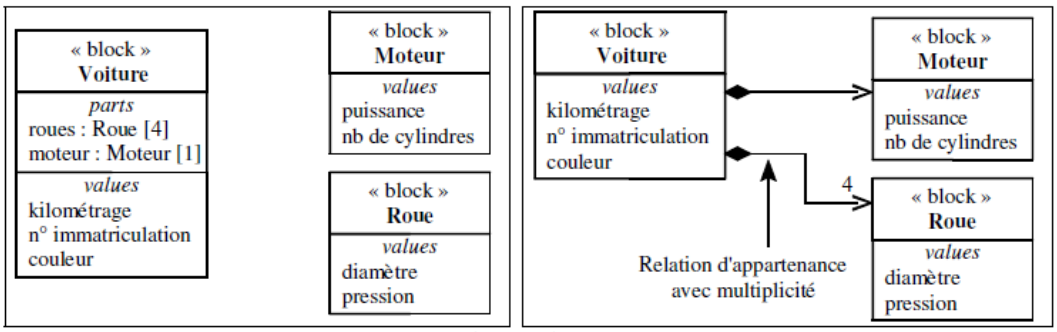
Le diagramme de définition de bloc (bdd) est utilisé pour représenter les blocs, leurs propriétés et leurs relations.

Le bloc SysML (block) constitue la brique de base pour la modélisation de la structure d'un système. Il permet de représenter, sous une forme générique, un système complet, un sous-système ou un composant élémentaire.

Les propriétés d'un bloc sont de deux types :

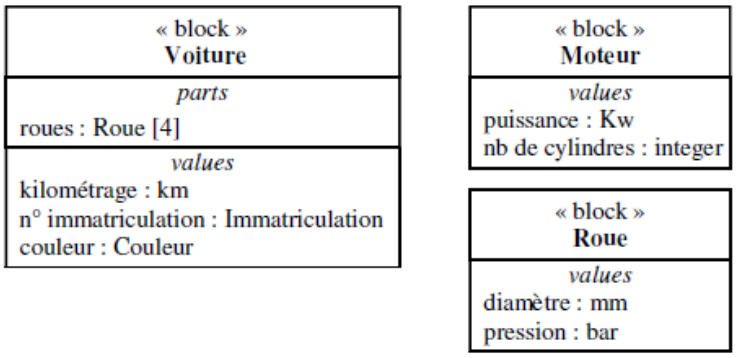
- les valeurs (value properties) qui décrivent des caractéristiques quantifiables en terme de valuetypes (domaine de valeur, dimension et unité optionnelles) ;
- les parties (part properties) qui désignent les blocs entrant dans sa composition

Si nous prenons l'exemple de la voiture qui a quatre roues et un moteur, une première représentation est donnée par l'une des deux figures suivantes.



Premier bdd du bloc voiture

Nous pouvons préciser le **type** des valeurs :



Deuxième bdd du bloc voiture avec quelques valeurs typées

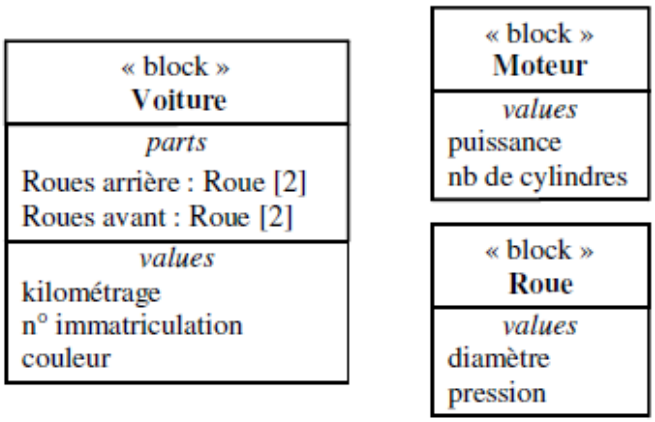
Remarques :

- Un bloc représente soit une entité physique, soit une entité logique ou conceptuelle.
- Le mot-clé «block» apparaît par défaut, mais il est aussi possible de définir de nouveaux mots-clés tels que « system », « subsystem », etc.

Instances de bloc :

Une instance de bloc désigne un objet partageant les propriétés du bloc, mais ayant une identité unique. Par exemple, chaque roue de diamètre X, gonflée à Y bar constitue une instance du bloc Roue.

Si nous considérons maintenant que les roues avant ont un rôle différent des roues arrière, parce qu'elles sont reliées au moteur (traction avant) et qu'elles peuvent avoir une pression différente, nous voulons les différencier en tant que parties. Nous obtenons alors la représentation suivante :

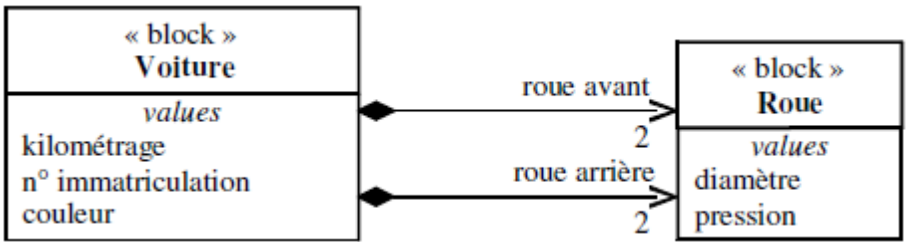


Troisième bdd du bloc voiture

1.1. Composition

La relation de composition entre blocs, dans laquelle un bloc représente le tout et les autres ses parties, est également représentable graphiquement. Le côté du tout est indiqué par un losange plein.

Du côté des parties, on peut indiquer un nom de rôle et une multiplicité (la valeur par défaut étant toujours 1). Le rôle permet notamment de distinguer les instances du bloc.

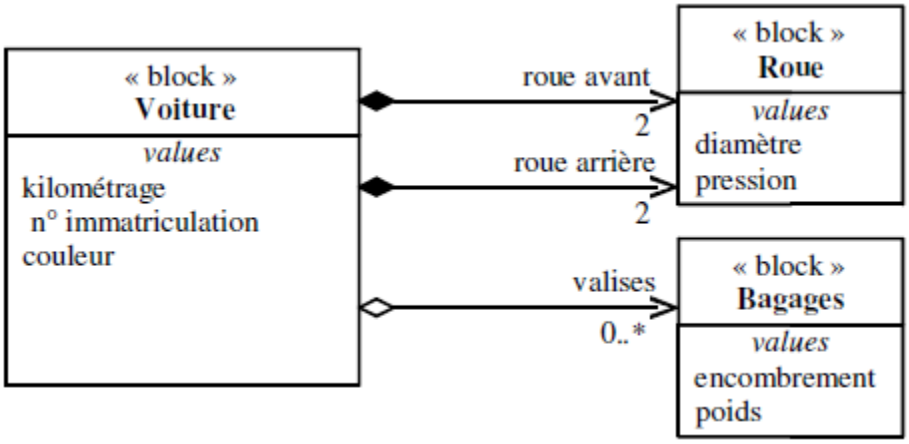


bdd du bloc voiture avec prise en compte de la composition

1.2. Agrégation

La relation d'agrégation (losange vide) est beaucoup moins forte que la relation de composition (losange plein). L'agrégation est utile pour représenter le fait que la présence du bloc est simplement optionnelle.

Si nous voulons exprimer le fait que la voiture contient éventuellement des bagages lors d'un départ en vacances, nous utiliserons l'agrégation et pas la composition.

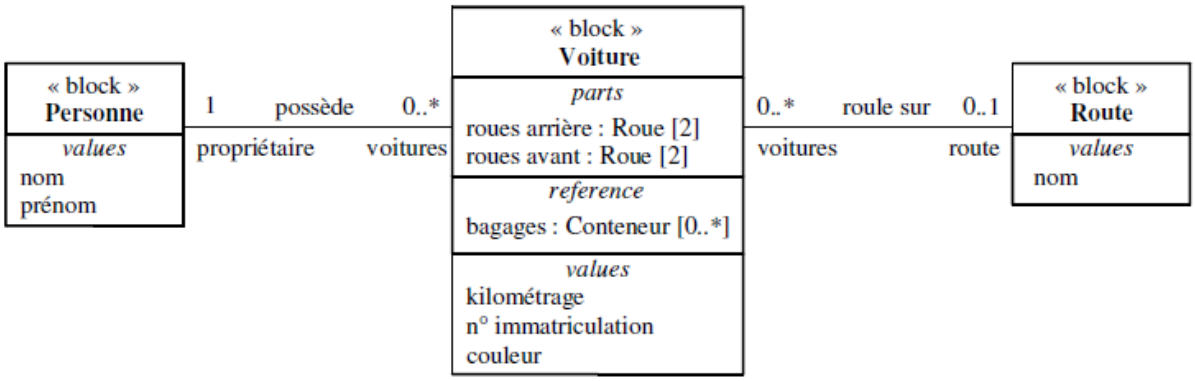


bdd du bloc voiture avec prise en compte de l'agrégation

1.3. Association

La relation d'association n'implique pas de contenance, comme la composition ou l'agrégation, mais une relation d'égal à égal. Par exemple, notre voiture roule habituellement sur une route (au sens large).

Elle est la propriété d'une personne, qui peut en posséder plusieurs et joue le rôle de propriétaire. Ces deux relations sont représentées par des associations en SysML (lignes simples).



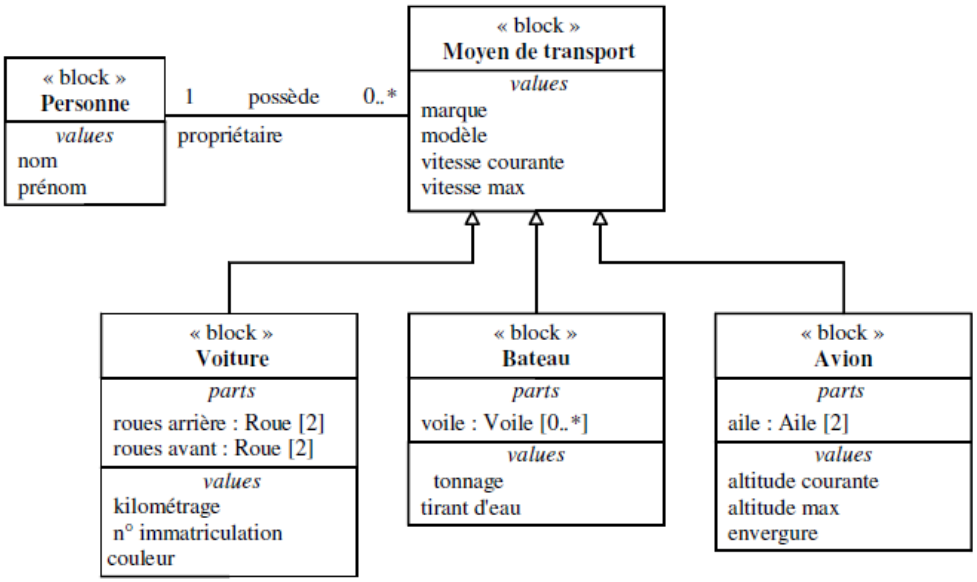
bdd du bloc voiture avec prise en compte de l'association

1.4. Généralisation

Toutes les définitions qui apparaissent dans un bdd peuvent être organisées dans une hiérarchie de classification. Le but est souvent de factoriser des propriétés communes (valeurs, parties...) à plusieurs blocs dans un bloc généralisé.

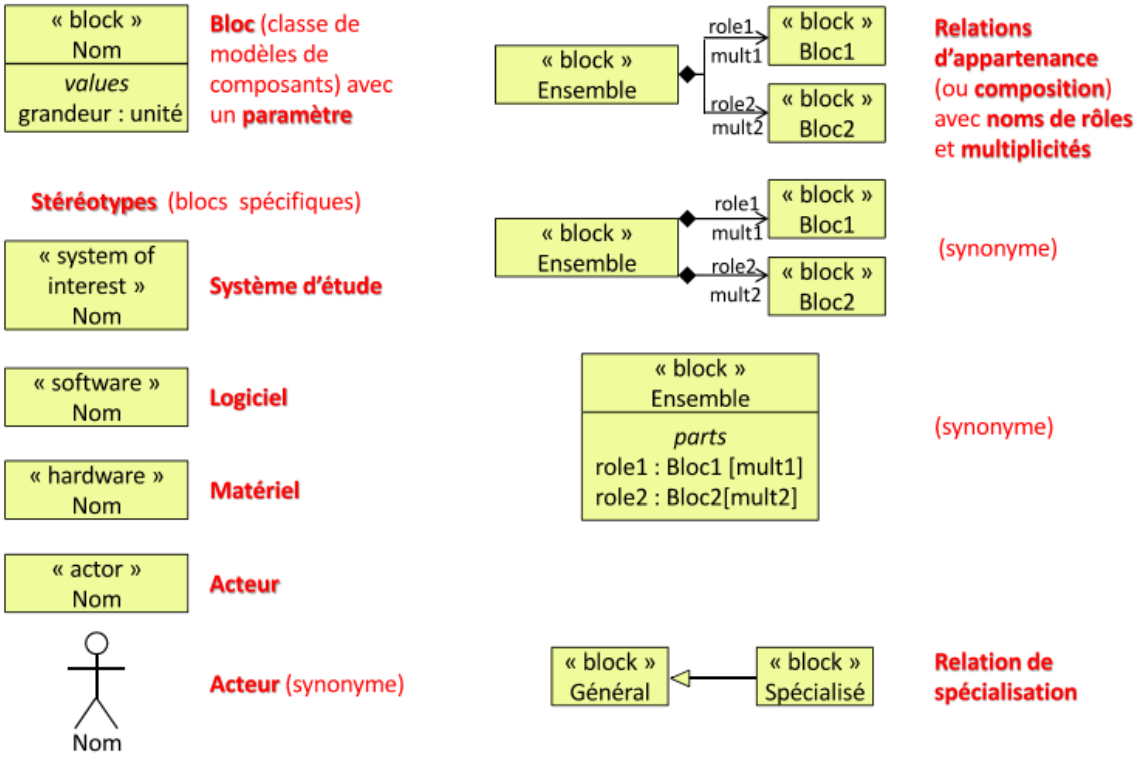
Les blocs spécialisés « héritent » des propriétés du bloc généralisé et peuvent comporter des propriétés spécifiques supplémentaires.

La généralisation en SysML se représente graphiquement par une flèche triangulaire pointant sur le bloc généralisé.



bdd avec relation de généralisation

1.5. Syntaxe



2. Diagramme de bloc interne

Le diagramme de bloc interne (ibd)décrit la structure interne du système en termes de parties, ports et connecteurs.

Il est utilisé pour décrire les connexions, les flots entre blocs grâce au concept de « port ».

Il représente l'intérieur d'un bloc :

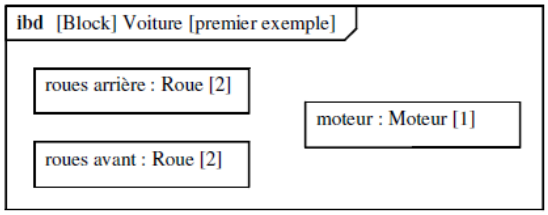
- ses composants (parts)
- leurs interfaces, et éventuellement celles du bloc (ports)
- les relations entre ces interfaces

...en accord avec sa définition (bdd)

2.1. Parts et références

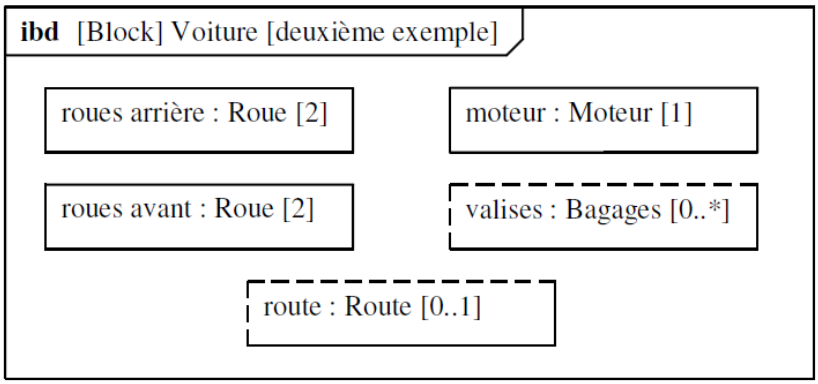
Dans l'exemple de la voiture, commençons par représenter les roues avant et arrière, ainsi que le moteur. La relation de composition du bdd se traduit de la façon suivante dans l'ibd :

- le cadre de l'ibd représente le bloc englobant. Il fournit le contexte pour tous les éléments du diagramme ;
- chaque partie du bloc résultant d'une relation de composition est représentée par un rectangle à l'intérieur du cadre. Son nom est défini ainsi : nom_partie : nom_bloc [multiplicité].



Premier ibd du bloc voiture

Les parts résultant d'une association simple ou d'une agrégation sont représentés par des rectangles tracés en pointillé. Ainsi, les valises et la route empruntée peuvent apparaître dans l'ibd.



Ibd du bloc voiture avec agrégation et association

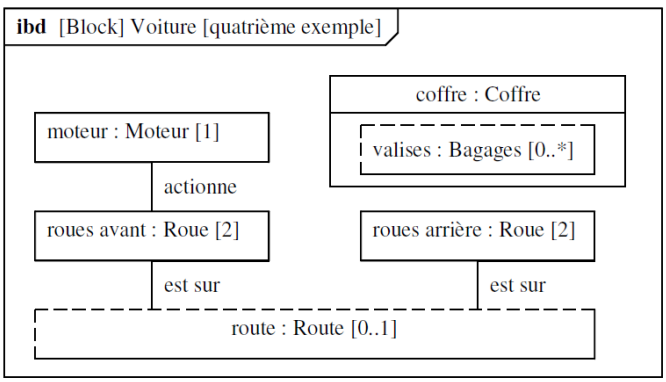
Il est important de noter qu'on peut représenter plusieurs niveaux de décomposition ou de référence sur un même ibd.

Dans notre exemple, si nous voulons exprimer le fait que les valises sont dans le coffre de la voiture, il suffit d'ajouter une partie coffre à la voiture et de déplacer la référence valises à l'intérieur de cette nouvelle partie.

2.2. Connecteur

Le connecteur est un concept structurel utilisé pour relier deux parts et leur permettre d'interagir.

Dans notre exemple, à l'intérieur du bloc Voiture, le moteur est relié aux deux roues avant, mais pas aux roues arrière. Les quatre roues sont (normalement...) reliées à la route.



Ibd du bloc voiture avec connecteurs

2.3. Ports et interfaces

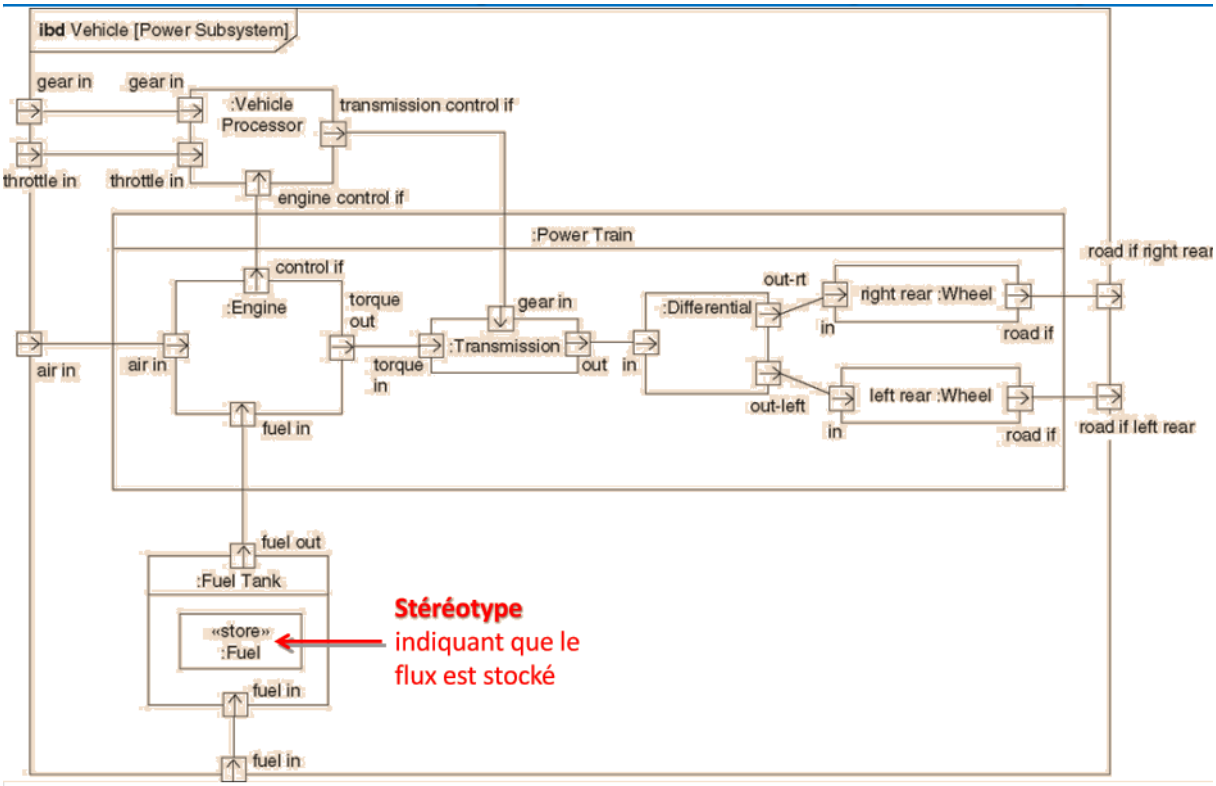
Le diagramme de bloc interne décrit également la logique de connexion, de services et de flux entre blocs grâce au concept de « port ».

Les ports définissent les points d'interaction offerts (provided) et requis (required) entre les blocs. Un bloc peut avoir plusieurs ports qui spécifient des points d'interaction différents. Les ports sont de deux types :

- les ports flux (flow port) qui autorisent la circulation de flux d'entités physiques entre les blocs, les entités physiques concernées étant la matière, l'énergie et/ou l'information.
- les ports standards qui autorisent la description de services logiques entre les blocs, au moyen d'interfaces regroupant des opérations.

Une combinaison des deux types de ports est souvent utile, mais les ports standards ne peuvent pas être connectés directement aux flow ports et réciproquement.

La figure ci-dessous représente l'ibd du bloc de puissance du véhicule



Ibd du bloc puissance du véhicule

Les ports sont représentés par des carrés chevauchant le bord du rectangle représentant le composant concerné. Ici, les flow ports sont tous atomiques, c'est-à-dire qu'ils sont traversés par un flux unidirectionnel entrant ou sortant, la direction étant simplement indiquée par une flèche à l'intérieur du carré.

La notation complète d'un flow port est : nom_port : nom_type [multiplicité].

Pour leur part, les ports standards sont simplement représentés par des carrés.

2.4. Compléments

- Il est possible de décrire les éléments de flux (item flows) qui circulent réellement sur les connecteurs. Cette description peut se révéler utile parfois car les flow ports définissent uniquement ce qui peut circuler. Par exemple l'indication énergie mécanique ou mélange air-essence sur la figure.
- Quand le flux traversant un port est bidirectionnelle, le flow port correspondant doit être modélisé comme un flow port composite (ou non atomique). On représente alors deux flèches opposées à l'intérieur du carré.
- Pour décrire un comportement basé sur l'invocation de services, le port standard est tout à fait adapté.

Cependant, au lieu d'affecter directement des opérations aux ports, il est plus intéressant de les regrouper en ensembles cohérents appelés interfaces.

2.5. Syntaxe