



# C1 : Modélisation des systèmes pluritechniques

## C1-2 : Analyse structurelle des systèmes

Émilien DURIF

Lycée La Martinière Monplaisir Lyon  
Classe de MPSI  
13 Septembre 2022



# Plan

- 1 **Présentation du langage SysML**
  - Intérêts et objectifs
  - Architecture du langage
  - Éléments de syntaxe
  - Présentation du support du cours
- 2 **Analyse du contexte et des exigences du système**
  - Diagramme de contexte
  - Diagramme des exigences
  - Diagramme des cas d'utilisation
- 3 **Analyse structurelle du système**
  - Diagramme de définition des blocs
  - Diagramme de blocs internes
  - Diagramme paramétrique
- 4 **Modélisation structurelle : chaîne d'info et d'énergie**
  - La chaîne d'information
  - La chaîne d'énergie
  - Les interfaces



# Plan

- 1 **Présentation du langage SysML**
  - Intérêts et objectifs
  - Architecture du langage
  - Éléments de syntaxe
  - Présentation du support du cours
- 2 **Analyse du contexte et des exigences du système**
  - Diagramme de contexte
  - Diagramme des exigences
  - Diagramme des cas d'utilisation
- 3 **Analyse structurelle du système**
  - Diagramme de définition des blocs
  - Diagramme de blocs internes
  - Diagramme paramétrique
- 4 **Modélisation structurelle : chaîne d'info et d'énergie**
  - La chaîne d'information
  - La chaîne d'énergie
  - Les interfaces



# Présentation du langage SysML

## intérêts et objectifs

Le chapitre d'introduction précédent a mis en évidence l'intérêt de développer un langage commun de modélisation tel que le . Modélisation **SysML** (**S**ystems **M**odeling **L**anguage) :

- formaliser et appréhender la conception d'un système.
- augmenter les capacités de conception des ingénieurs en minimisant les risques de retard ou de problèmes liés à l'élaboration du produit.
- mettre en commun les spécifications, contraintes et paramètres de l'ensemble du système pour aboutir directement à la simulation d'un système complexe et ainsi autoriser la prévision de ses performances.



## Présentation du langage SysML

### intérêts et objectifs

Le chapitre d'introduction précédent a mis en évidence l'intérêt de développer un langage commun de modélisation tel que le . Modélisation **SysML** (**S**ystems **M**odeling **L**anguage) :

- formaliser et appréhender la conception d'un système.
- augmenter les capacités de conception des ingénieurs en minimisant les risques de retard ou de problèmes liés à l'élaboration du produit.
- mettre en commun les spécifications, contraintes et paramètres de l'ensemble du système pour aboutir directement à la simulation d'un système complexe et ainsi autoriser la prévision de ses performances.



## Présentation du langage SysML

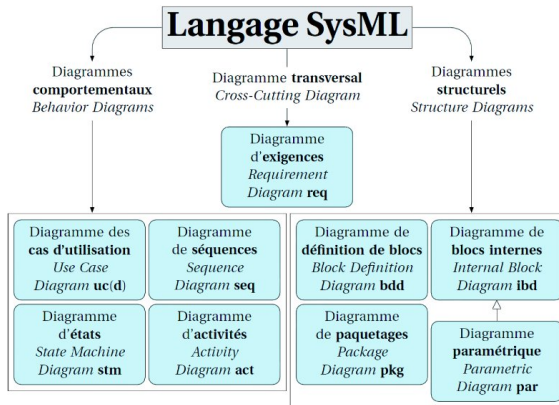
### intérêts et objectifs

Le chapitre d'introduction précédent a mis en évidence l'intérêt de développer un langage commun de modélisation tel que le . Modélisation **SysML** (**S**ystems **M**odeling **L**anguage) :

- formaliser et appréhender la conception d'un système.
- augmenter les capacités de conception des ingénieurs en minimisant les risques de retard ou de problèmes liés à l'élaboration du produit.
- mettre en commun les spécifications, contraintes et paramètres de l'ensemble du système pour aboutir directement à la simulation d'un système complexe et ainsi autoriser la prévision de ses performances.



## Architecture du langage

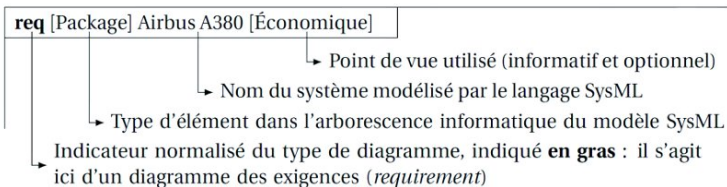


### Remarque

A ces neuf diagrammes peut s'ajouter le **diagramme du contexte** qui est un cas particulier du diagramme de définition des blocs et qui permet de situer le contexte du système étudié.





## Éléments de Syntaxe







## Éléments de Syntaxe

Liens	Significations et commentaires de la relation	Orientations
	<b>Contenance</b> : décompose une exigence en plusieurs autres plus faciles ensuite à identifier lors de la mise en place du système ou des tests.	Cercle contenant une croix du côté du conteneur
	<b>Association</b> : relie deux éléments considérés d'égale importance et indique qu'ils sont en lien sans en indiquer la nature	<ul style="list-style-type: none"><li>• Unidirectionnelle (une seule flèche) ;</li><li>• bidirectionnelle (sans flèche).</li></ul>






## Éléments de Syntaxe

Liens	Significations et commentaires de la relation	Orientations
— — — — →	Inclusion (extension, raffinement ou dérivation)	<ul style="list-style-type: none"><li>• Du cas d'utilisation global vers un cas d'utilisation partiel inclus avec le mot clé <i>include</i> pour l'<b>inclusion</b> ;</li><li>• du cas d'utilisation partiel vers le cas d'utilisation global avec le mot clé <i>extend</i> pour l'<b>extension</b> ;</li><li>• de l'exigence partielle vers l'exigence globale avec le mot clé <i>refine</i> pour l'ajout de précisions, par exemple des données quantitatives, pour le <b>raffinement</b> ;</li><li>• de l'exigence partielle vers l'exigence globale avec le mot clé <b>deriveReq</b> pour relier de manière dérivée des exigences de niveaux différents, par exemple entre un système et certains de ses sous-systèmes (<b>dérivation</b>).</li></ul>



## Éléments de Syntaxe

Liens	Significations et commentaires de la relation	Orientations
	<b>Généralisation</b> : spécialisation d'un élément (cas d'utilisa- tion, bloc, etc)	Pointe blanche orientée vers l'élément plus général.
	<b>Composition</b> : relie deux blocs et indique qu'un élément est structurel- lement indispensable à l'autre	Losange plein du côté du composé (ou système principal), l'autre extrémité du côté du composant.
	<b>Agrégation</b> : même rôle que la relation de com- position mais elle a un sens moins fort : en général, elle indique que le composant est présent de manière op- tionnelle	losange vide du côté du composé (ou système principal), l'autre extrémité du côté du com- posant.



## Éléments de Syntaxe

Pour chacun des liens, nous pourrons avoir différents types d'associations :

- **Extend** : le cas d'utilisation source est une extension possible du cas d'utilisation destination.
- **Include** : le cas d'utilisation source comprend obligatoirement le cas inclus.
- **Derive** : une ou plusieurs exigences sont dérivées d'une exigence.
- **DeriveReq** : permet de relier une exigence d'un niveau général à une exigence d'un niveau plus spécialisée mais exprimant la même contrainte.
- **Satisfy** : un ou plusieurs éléments du modèle permettent de satisfaire une exigence.
- **Verify** : un ou plusieurs éléments du modèle permettent de vérifier et valider une exigence.
- **Refine** : un ou plusieurs éléments du modèle redéfinissent une exigence.



## Éléments de Syntaxe

Pour chacun des liens, nous pourrons avoir différents types d'associations :

- **Extend** : le cas d'utilisation source est une extension possible du cas d'utilisation destination.
- **Include** : le cas d'utilisation source comprend obligatoirement le cas inclus.
- **Derive** : une ou plusieurs exigences sont dérivées d'une exigence.
- **DeriveReq** : permet de relier une exigence d'un niveau général à une exigence d'un niveau plus spécialisée mais exprimant la même contrainte.
- **Satisfy** : un ou plusieurs éléments du modèle permettent de satisfaire une exigence.
- **Verify** : un ou plusieurs éléments du modèle permettent de vérifier et valider une exigence.
- **Refine** : un ou plusieurs éléments du modèle redéfinissent une exigence.



## Éléments de Syntaxe

Pour chacun des liens, nous pourrons avoir différents types d'associations :

- **Extend** : le cas d'utilisation source est une extension possible du cas d'utilisation destination.
- **Include** : le cas d'utilisation source comprend obligatoirement le cas inclus.
- **Derive** : une ou plusieurs exigences sont dérivées d'une exigence.
- **DeriveReq** : permet de relier une exigence d'un niveau général à une exigence d'un niveau plus spécialisée mais exprimant la même contrainte.
- **Satisfy** : un ou plusieurs éléments du modèle permettent de satisfaire une exigence.
- **Verify** : un ou plusieurs éléments du modèle permettent de vérifier et valider une exigence.
- **Refine** : un ou plusieurs éléments du modèle redéfinissent une exigence.



## Éléments de Syntaxe

Pour chacun des liens, nous pourrons avoir différents types d'associations :

- **Extend** : le cas d'utilisation source est une extension possible du cas d'utilisation destination.
- **Include** : le cas d'utilisation source comprend obligatoirement le cas inclus.
- **Derive** : une ou plusieurs exigences sont dérivées d'une exigence.
- **DeriveReq** : permet de relier une exigence d'un niveau général à une exigence d'un niveau plus spécialisée mais exprimant la même contrainte.
- **Satisfy** : un ou plusieurs éléments du modèle permettent de satisfaire une exigence.
- **Verify** : un ou plusieurs éléments du modèle permettent de vérifier et valider une exigence.
- **Refine** : un ou plusieurs éléments du modèle redéfinissent une exigence.



## Éléments de Syntaxe

Pour chacun des liens, nous pourrons avoir différents types d'associations :

- **Extend** : le cas d'utilisation source est une extension possible du cas d'utilisation destination.
- **Include** : le cas d'utilisation source comprend obligatoirement le cas inclus.
- **Derive** : une ou plusieurs exigences sont dérivées d'une exigence.
- **DeriveReq** : permet de relier une exigence d'un niveau général à une exigence d'un niveau plus spécialisée mais exprimant la même contrainte.
- **Satisfy** : un ou plusieurs éléments du modèle permettent de satisfaire une exigence.
- **Verify** : un ou plusieurs éléments du modèle permettent de vérifier et valider une exigence.
- **Refine** : un ou plusieurs éléments du modèle redéfinissent une exigence.





## Éléments de Syntaxe

Pour chacun des liens, nous pourrions avoir différents types d'associations :

- **Extend** : le cas d'utilisation source est une extension possible du cas d'utilisation destination.
- **Include** : le cas d'utilisation source comprend obligatoirement le cas inclus.
- **Derive** : une ou plusieurs exigences sont dérivées d'une exigence.
- **DeriveReq** : permet de relier une exigence d'un niveau général à une exigence d'un niveau plus spécialisée mais exprimant la même contrainte.
- **Satisfy** : un ou plusieurs éléments du modèle permettent de satisfaire une exigence.
- **Verify** : un ou plusieurs éléments du modèle permettent de vérifier et valider une exigence.
- **Refine** : un ou plusieurs éléments du modèle redéfinissent une exigence.



## Éléments de Syntaxe

Pour chacun des liens, nous pourrons avoir différents types d'associations :

- **Extend** : le cas d'utilisation source est une extension possible du cas d'utilisation destination.
- **Include** : le cas d'utilisation source comprend obligatoirement le cas inclus.
- **Derive** : une ou plusieurs exigences sont dérivées d'une exigence.
- **DeriveReq** : permet de relier une exigence d'un niveau général à une exigence d'un niveau plus spécialisée mais exprimant la même contrainte.
- **Satisfy** : un ou plusieurs éléments du modèle permettent de satisfaire une exigence.
- **Verify** : un ou plusieurs éléments du modèle permettent de vérifier et valider une exigence.
- **Refine** : un ou plusieurs éléments du modèle redéfinissent une exigence.



## Présentation du support du cours





# Plan

- 1 **Présentation du langage SysML**
  - Intérêts et objectifs
  - Architecture du langage
  - Éléments de syntaxe
  - Présentation du support du cours
- 2 **Analyse du contexte et des exigences du système**
  - Diagramme de contexte
  - Diagramme des exigences
  - Diagramme des cas d'utilisation
- 3 **Analyse structurelle du système**
  - Diagramme de définition des blocs
  - Diagramme de blocs internes
  - Diagramme paramétrique
- 4 **Modélisation structurelle : chaîne d'info et d'énergie**
  - La chaîne d'information
  - La chaîne d'énergie
  - Les interfaces



## Diagramme de contexte

### Diagramme de contexte

- Extension non normalisée du langage **SysML** : définition des frontières de l'étude et la phase du cycle de vie dans laquelle on situe l'étude (il s'agit généralement de la **phase d'utilisation normale** du système).
- Ce diagramme permet de préciser, si possible de manière exhaustive, les acteurs et éléments environnants au système étudié. Il permet également de faire apparaître les différents acteurs ou éléments intervenant dans une exigence.



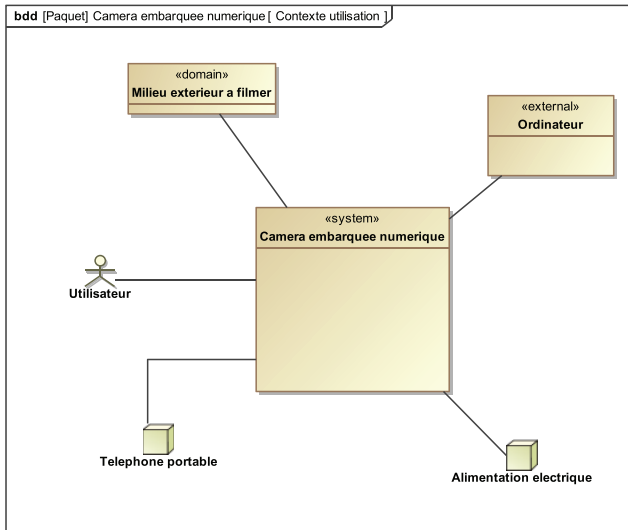
## Diagramme de contexte

### Diagramme de contexte

- Extension non normalisée du langage **SysML** : définition des frontières de l'étude et la phase du cycle de vie dans laquelle on situe l'étude (il s'agit généralement de la **phase d'utilisation normale** du système).
- Ce diagramme permet de préciser, si possible de manière exhaustive, les acteurs et éléments environnants au système étudié. Il permet également de faire apparaître les différents acteurs ou éléments intervenant dans une exigence.

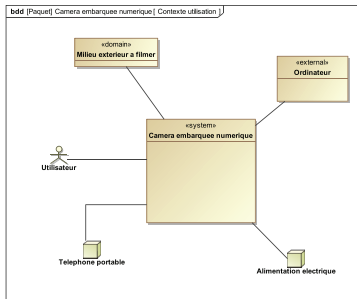


## Diagramme de contexte





## Diagramme de contexte



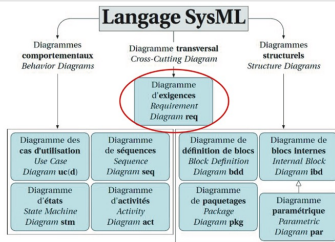




## Diagramme des exigences

### Exigence

- Une **exigence** permet de spécifier une capacité ou une contrainte qui doit être satisfaite par le système. Les exigences servent à établir un contrat entre le client et les réalisateurs du futur système (cahier des charges).
- De nombreux domaines peuvent être couverts, les plus classiques étant les exigences environnementales, économiques, fonctionnelles ou techniques.

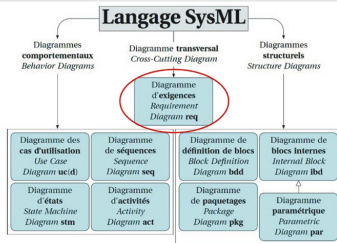




## Diagramme des exigences

### Exigence

- Une **exigence** permet de spécifier une capacité ou une contrainte qui doit être satisfaite par le système. Les exigences servent à établir un contrat entre le client et les réalisateurs du futur système (cahier des charges).
- De nombreux domaines peuvent être couverts, les plus classiques étant les exigences **environnementales**, **économiques**, **fonctionnelles** ou **techniques**.





## Diagramme des exigences

### Exigence

- Une **exigence** permet de spécifier une capacité ou une contrainte qui doit être satisfaite par le système. Les exigences servent à établir un contrat entre le client et les réalisateurs du futur système (cahier des charges).
- De nombreux domaines peuvent être couverts, les plus classiques étant les exigences **environnementales**, **économiques**, **fonctionnelles** ou **techniques**.

### Diagramme des exigences (req)

- Le diagramme des exigences, appelé *Requirement Diagram* (**req**) dans le langage **SysML**, est le seul diagramme transversal du langage **SysML**.
- L'objectif de ce diagramme est de modéliser les exigences devant être vérifiées par le système en liant les solutions mises en oeuvre sur le système avec les besoins définis dans le cahier des charges. Ce diagramme traduit, par des fonctionnalités ou des contraintes, ce qui doit être satisfait par le système.



## Diagramme des exigences

### Exigence

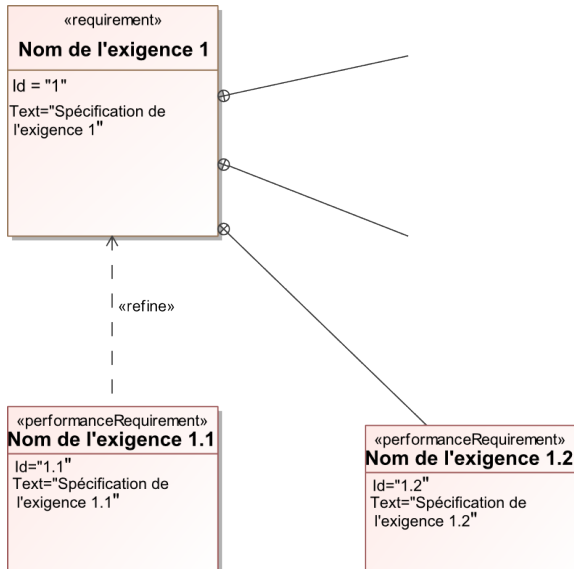
- Une **exigence** permet de spécifier une capacité ou une contrainte qui doit être satisfaite par le système. Les exigences servent à établir un contrat entre le client et les réalisateurs du futur système (cahier des charges).
- De nombreux domaines peuvent être couverts, les plus classiques étant les exigences **environnementales**, **économiques**, **fonctionnelles** ou **techniques**.

### Diagramme des exigences (req)

- Le diagramme des exigences, appelé *Requirement Diagram* (**req**) dans le langage **SysML**, est le seul diagramme transversal du langage **SysML**.
- L'objectif de ce diagramme est de modéliser les exigences devant être vérifiées par le système en liant les solutions mises en oeuvre sur le système avec les besoins définis dans le cahier des charges. Ce diagramme traduit, par des fonctionnalités ou des contraintes, ce qui doit être satisfait par le système.



## Diagramme des exigences





## Diagramme des exigences

### Remarque

- Le diagramme doit être le plus lisible possible et donc le plus simple (il est possible de réaliser plusieurs diagrammes pour alléger les schémas).
- Il est possible d'associer des propriétés aux exigences telles que :
  - une priorité, par exemple haute, moyenne ou basse;
  - une indication de la "source", par exemple client, législation ou concurrence;
  - un statut, par exemple proposé, validé, implanté;
  - ou, de manière générale, toute donnée pouvant se rapporter à une exigence devant être validée à un niveau du cycle de vie du produit.



## Diagramme des exigences

### Remarque

- Le diagramme doit être le plus lisible possible et donc le plus simple (il est possible de réaliser plusieurs diagrammes pour alléger les schémas).
- Il est possible d'associer des propriétés aux exigences telles que :
  - une **priorité**, par exemple haute, moyenne ou basse ;
  - une **indication de la "source"**, par exemple client, législation ou concurrence ;
  - un **statut**, par exemple proposé, validé, implanté ;
  - ou, de manière générale, toute donnée pouvant se rapporter à une exigence devant être validée à un niveau du cycle de vie du produit.



## Diagramme des exigences

### Remarque

- Le diagramme doit être le plus lisible possible et donc le plus simple (il est possible de réaliser plusieurs diagrammes pour alléger les schémas).
- Il est possible d'associer des propriétés aux exigences telles que :
  - une **priorité**, par exemple haute, moyenne ou basse ;
  - une **indication de la "source"**, par exemple client, législation ou concurrence ;
  - un **statut**, par exemple proposé, validé, implanté ;
  - ou, de manière générale, toute donnée pouvant se rapporter à une exigence devant être validée à un niveau du cycle de vie du produit.





## Diagramme des exigences

### Remarque

- Le diagramme doit être le plus lisible possible et donc le plus simple (il est possible de réaliser plusieurs diagrammes pour alléger les schémas).
- Il est possible d'associer des propriétés aux exigences telles que :
  - une **priorité**, par exemple haute, moyenne ou basse ;
  - une **indication de la "source"**, par exemple client, législation ou concurrence ;
  - un **statut**, par exemple proposé, validé, implanté ;
  - ou, de manière générale, toute donnée pouvant se rapporter à une exigence devant être validée à un niveau du cycle de vie du produit.



## Diagramme des exigences

### Remarque

- Le diagramme doit être le plus lisible possible et donc le plus simple (il est possible de réaliser plusieurs diagrammes pour alléger les schémas).
- Il est possible d'associer des propriétés aux exigences telles que :
  - une **priorité**, par exemple haute, moyenne ou basse ;
  - une **indication de la "source"**, par exemple client, législation ou concurrence ;
  - un **statut**, par exemple proposé, validé, implanté ;
  - ou, de manière générale, toute donnée pouvant se rapporter à une exigence devant être validée à un niveau du cycle de vie du produit.



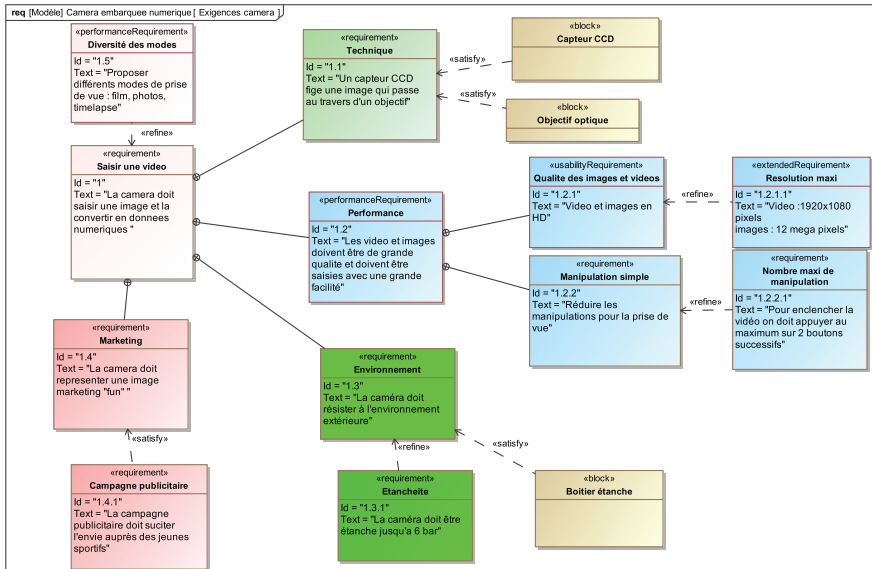
## Diagramme des exigences

### Remarque

- Le diagramme doit être le plus lisible possible et donc le plus simple (il est possible de réaliser plusieurs diagrammes pour alléger les schémas).
- Il est possible d'associer des propriétés aux exigences telles que :
  - une **priorité**, par exemple haute, moyenne ou basse ;
  - une **indication de la "source"**, par exemple client, législation ou concurrence ;
  - un **statut**, par exemple proposé, validé, implanté ;
  - ou, **de manière générale**, toute donnée pouvant se rapporter à une exigence devant être validée à un niveau du cycle de vie du produit.



## Diagramme des exigences

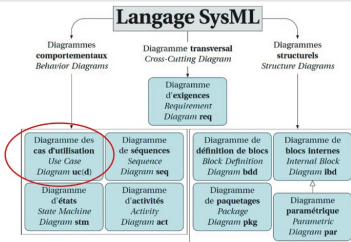




## Diagramme des cas d'utilisation

### Diagramme des cas d'utilisation (uc ou ucd)

- Le diagramme des cas d'utilisation est un *diagramme comportemental*, appelé **Use Case Diagram (uc ou ucd)** dans le langage SysML.
- L'objectif de ce diagramme est de montrer les fonctionnalités offertes par un système en identifiant les services qu'il rend : il permet donc de modéliser les exigences selon un point de vue complémentaire à celui exposé par le diagramme des exigences.
- L'énoncé d'un cas d'utilisation doit se faire hors technologie, puisque il est défini en termes de résultats attendus.

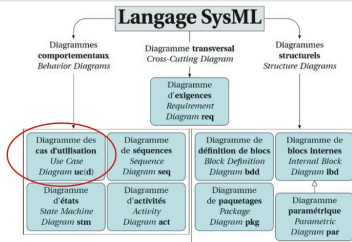




## Diagramme des cas d'utilisation

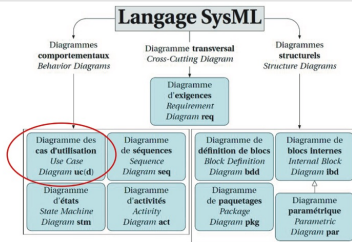
### Diagramme des cas d'utilisation (uc ou ucd)

- Le diagramme des cas d'utilisation est un *diagramme comportemental*, appelé **Use Case Diagram (uc ou ucd)** dans le langage SysML.
- L'objectif de ce diagramme est de montrer les fonctionnalités offertes par un système en identifiant les services qu'il rend : il permet donc de modéliser les exigences selon un point de vue complémentaire à celui exposé par le diagramme des exigences.
- L'énoncé d'un cas d'utilisation doit se faire hors technologie, puisque il est défini en termes de résultats attendus.



## Diagramme des cas d'utilisation (uc ou ucd)

- Le diagramme des cas d'utilisation est un *diagramme comportemental*, appelé **Use Case Diagram** (**uc** ou **ucd**) dans le langage **SysML**.
- L'objectif de ce diagramme est de montrer les fonctionnalités offertes par un système en identifiant les services qu'il rend : il permet donc de modéliser les exigences selon un point de vue complémentaire à celui exposé par le diagramme des exigences.
- L'énoncé d'un cas d'utilisation doit se faire hors technologie, puisque il est défini en termes de résultats attendus.





## Diagramme des cas d'utilisation

### Diagramme des cas d'utilisation (uc ou ucd)

- Le diagramme des cas d'utilisation est un *diagramme comportemental*, appelé **Use Case Diagram (uc ou ucd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de montrer les fonctionnalités offertes par un système en identifiant les services qu'il rend : il permet donc de modéliser les exigences selon un point de vue complémentaire à celui exposé par le diagramme des exigences.
- L'énoncé d'un cas d'utilisation doit se faire hors technologie, puisque il est défini en termes de résultats attendus.

### Éléments graphiques

- Les acteurs, entités extérieures au système et en interaction avec lui, sont représentés par le pictogramme "*bonhomme bâton*" et sont reliés à un ou plusieurs cas d'utilisation par une ligne simple appelée association.
- Les cas d'utilisation sont représentés sous forme d'ovales. Ils donnent les fonctionnalités du système et sont énoncés du point de vue de l'acteur.
- La frontière du système permet de symboliser les limites du modèle et est représentée par un simple rectangle englobant les cas d'utilisation, les acteurs étant à l'extérieur, à gauche si ils sont considérés comme "principaux", à droite si ils sont considérés comme "secondaires".





## Diagramme des cas d'utilisation

### Diagramme des cas d'utilisation (uc ou ucd)

- Le diagramme des cas d'utilisation est un *diagramme comportemental*, appelé **Use Case Diagram (uc ou ucd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de montrer les fonctionnalités offertes par un système en identifiant les services qu'il rend : il permet donc de modéliser les exigences selon un point de vue complémentaire à celui exposé par le diagramme des exigences.
- L'énoncé d'un cas d'utilisation doit se faire hors technologie, puisque il est défini en termes de résultats attendus.

### Éléments graphiques

- Les acteurs, entités extérieures au système et en interaction avec lui, sont représentés par le pictogramme "*bonhomme bâton*" et sont reliés à un ou plusieurs cas d'utilisation par une ligne simple appelée association.
- Les cas d'utilisation sont représentés sous forme d'ovales. Ils donnent les fonctionnalités du système et sont énoncés du point de vue de l'acteur.
- La frontière du système permet de symboliser les limites du modèle et est représentée par un simple rectangle englobant les cas d'utilisation, les acteurs étant à l'extérieur, à gauche si ils sont considérés comme "principaux", à droite si ils sont considérés comme "secondaires".



## Diagramme des cas d'utilisation

### Diagramme des cas d'utilisation (uc ou ucd)

- Le diagramme des cas d'utilisation est un *diagramme comportemental*, appelé **Use Case Diagram (uc ou ucd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de montrer les fonctionnalités offertes par un système en identifiant les services qu'il rend : il permet donc de modéliser les exigences selon un point de vue complémentaire à celui exposé par le diagramme des exigences.
- L'énoncé d'un cas d'utilisation doit se faire hors technologie, puisque il est défini en termes de résultats attendus.

### Éléments graphiques

- Les acteurs, entités extérieures au système et en interaction avec lui, sont représentés par le pictogramme "*bonhomme bâton*" et sont reliés à un ou plusieurs cas d'utilisation par une ligne simple appelée association.
- Les cas d'utilisation sont représentés sous forme d'ovales. Ils donnent les fonctionnalités du système et sont énoncés du point de vue de l'acteur.
- La frontière du système permet de symboliser les limites du modèle et est représentée par un simple rectangle englobant les cas d'utilisation, les acteurs étant à l'extérieur, à gauche si ils sont considérés comme "principaux", à droite si ils sont considérés comme "secondaires".



## Diagramme des cas d'utilisation

### Diagramme des cas d'utilisation (uc ou ucd)

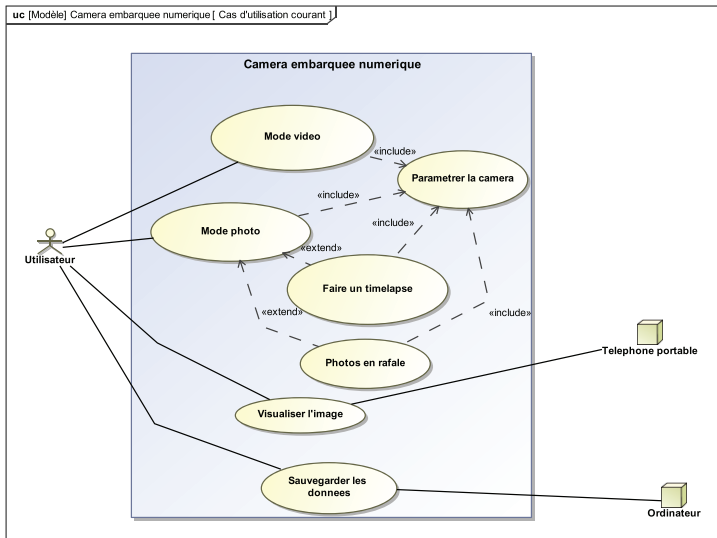
- Le diagramme des cas d'utilisation est un *diagramme comportemental*, appelé **Use Case Diagram (uc ou ucd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de montrer les fonctionnalités offertes par un système en identifiant les services qu'il rend : il permet donc de modéliser les exigences selon un point de vue complémentaire à celui exposé par le diagramme des exigences.
- L'énoncé d'un cas d'utilisation doit se faire hors technologie, puisque il est défini en termes de résultats attendus.

### Éléments graphiques

- Les acteurs, entités extérieures au système et en interaction avec lui, sont représentés par le pictogramme “*bonhomme bâton*” et sont reliés à un ou plusieurs cas d'utilisation par une ligne simple appelée association.
- Les cas d'utilisation sont représentés sous forme d'ovales. Ils donnent les fonctionnalités du système et sont énoncés du point de vue de l'acteur.
- La frontière du système permet de symboliser les limites du modèle et est représentée par un simple rectangle englobant les cas d'utilisation, les acteurs étant à l'extérieur, à gauche si ils sont considérés comme “principaux”, à droite si ils sont considérés comme “secondaires”.



## Diagramme des cas d'utilisation





## Diagramme des cas d'utilisation

### Remarque

Les fonctionnalités d'un système correspondent à des cas d'utilisation, c'est-à-dire à des services rendus par le système. Il n'apparaîtra donc pas ce qui ne peut être fait par des acteurs extérieurs : ainsi, par exemple, le lavage, la recharge, le recyclage, la réparation, etc. ne doivent pas apparaître si le système n'a pas été développé expressément pour cela.



# Plan

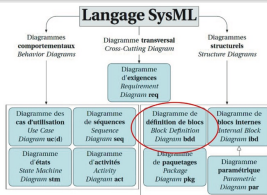
- 1 **Présentation du langage SysML**
  - Intérêts et objectifs
  - Architecture du langage
  - Éléments de syntaxe
  - Présentation du support du cours
- 2 **Analyse du contexte et des exigences du système**
  - Diagramme de contexte
  - Diagramme des exigences
  - Diagramme des cas d'utilisation
- 3 **Analyse structurelle du système**
  - Diagramme de définition des blocs
  - Diagramme de blocs internes
  - Diagramme paramétrique
- 4 **Modélisation structurelle : chaîne d'info et d'énergie**
  - La chaîne d'information
  - La chaîne d'énergie
  - Les interfaces



## Diagramme de définition des blocs

### Diagramme de définition des blocs (bdd)

- Le **diagramme de définition des blocs** est un *diagramme structurel* appelé **Block Definition Diagram (bdd)** dans le langage SysML.
- L'objectif de ce diagramme est de décrire le système via des blocs (blocks dans le langage SysML) et représentant des éléments matériels (cas le plus fréquent) mais également des entités abstraites (regroupement logique d'éléments) ou des logiciels.
- Ce diagramme représente les caractéristiques principales de chaque bloc ainsi que les liens entre eux : il permet donc une **modélisation de l'architecture du système**.

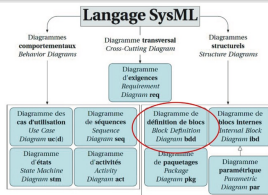




## Diagramme de définition des blocs

### Diagramme de définition des blocs (bdd)

- Le **diagramme de définition des blocs** est un *diagramme structurel* appelé **Block Definition Diagram (bdd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de décrire le système via des blocs (blocks dans le langage **SysML**) et représentant des éléments matériels (cas le plus fréquent) mais également des entités abstraites (regroupement logique d'éléments) ou des logiciels.
- Ce diagramme représente les caractéristiques principales de chaque bloc ainsi que les liens entre eux : il permet donc une **modélisation de l'architecture du système**.



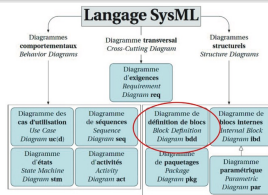




## Diagramme de définition des blocs

### Diagramme de définition des blocs (bdd)

- Le **diagramme de définition des blocs** est un *diagramme structurel* appelé **Block Definition Diagram (bdd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de décrire le système via des blocs (blocks dans le langage **SysML**) et représentant des éléments matériels (cas le plus fréquent) mais également des entités abstraites (regroupement logique d'éléments) ou des logiciels.
- Ce diagramme représente les caractéristiques principales de chaque bloc ainsi que les liens entre eux : il permet donc une **modélisation de l'architecture du système**.





## Diagramme de définition des blocs

### Diagramme de définition des blocs (bdd)

- Le **diagramme de définition des blocs** est un *diagramme structurel* appelé **Block Definition Diagram (bdd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de décrire le système via des blocs (blocks dans le langage **SysML**) et représentant des éléments matériels (cas le plus fréquent) mais également des entités abstraites (regroupement logique d'éléments) ou des logiciels.
- Ce diagramme représente les caractéristiques principales de chaque bloc ainsi que les liens entre eux : il permet donc une **modélisation de l'architecture du système**.

### Représentation graphique d'un bloc

D'un point de vue graphique, on représente un bloc par un rectangle avec le stéréotype *block* comprenant un titre et des compartiments étagés regroupant des propriétés particulières. Les plus significatives sont :

- *value* : exprime une caractéristique quantifiable ;
- *part* : représente ce qui compose le bloc (équivalent à un lien de composition ;



## Diagramme de définition des blocs

### Diagramme de définition des blocs (bdd)

- Le **diagramme de définition des blocs** est un *diagramme structurel* appelé **Block Definition Diagram (bdd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de décrire le système via des blocs (blocks dans le langage **SysML**) et représentant des éléments matériels (cas le plus fréquent) mais également des entités abstraites (regroupement logique d'éléments) ou des logiciels.
- Ce diagramme représente les caractéristiques principales de chaque bloc ainsi que les liens entre eux : il permet donc une **modélisation de l'architecture du système**.

### Représentation graphique d'un bloc

D'un point de vue graphique, on représente un bloc par un rectangle avec le stéréotype *block* comprenant un titre et des compartiments étagés regroupant des propriétés particulières. Les plus significatives sont :

- *value* : exprime une caractéristique quantifiable ;
- *part* : représente ce qui compose le bloc (équivalent à un lien de composition ;



## Diagramme de définition des blocs

### Diagramme de définition des blocs (bdd)

- Le **diagramme de définition des blocs** est un *diagramme structurel* appelé **Block Definition Diagram (bdd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de décrire le système via des blocs (blocks dans le langage **SysML**) et représentant des éléments matériels (cas le plus fréquent) mais également des entités abstraites (regroupement logique d'éléments) ou des logiciels.
- Ce diagramme représente les caractéristiques principales de chaque bloc ainsi que les liens entre eux : il permet donc une **modélisation de l'architecture du système**.

### Représentation graphique d'un bloc

D'un point de vue graphique, on représente un bloc par un rectangle avec le stéréotype *block* comprenant un titre et des compartiments étagés regroupant des propriétés particulières. Les plus significatives sont :

- *value* : exprime une caractéristique quantifiable ;
- *part* : représente ce qui compose le bloc (équivalent à un lien de composition ;



## Diagramme de définition des blocs

### Diagramme de définition des blocs (bdd)

- Le **diagramme de définition des blocs** est un *diagramme structurel* appelé **Block Definition Diagram (bdd)** dans le langage **SysML**.
- L'objectif de ce diagramme est de décrire le système via des blocs (blocks dans le langage **SysML**) et représentant des éléments matériels (cas le plus fréquent) mais également des entités abstraites (regroupement logique d'éléments) ou des logiciels.
- Ce diagramme représente les caractéristiques principales de chaque bloc ainsi que les liens entre eux : il permet donc une **modélisation de l'architecture du système**.

«block»

**Voiture**

*parts*

roue : Roue [4]

*values*

kilométrage

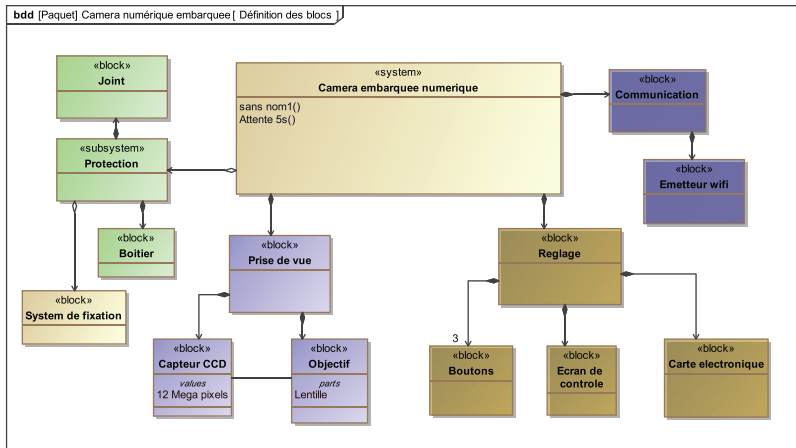


## Diagramme de définition des blocs



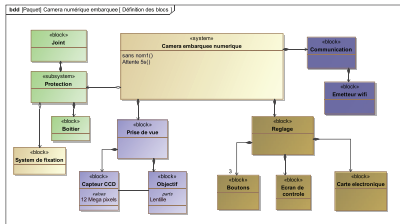


## Diagramme de définition des blocs





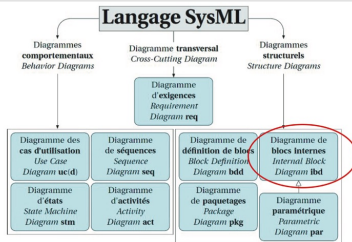
## Diagramme de définition des blocs





## Diagramme de blocs internes (ibd)

- Le **diagramme de blocs internes** est un *diagramme structurel* appelé **Internal Block Diagram (ibd)** dans le langage **SysML**.
- Le diagramme de blocs internes est rattaché à un bloc issu du diagramme de définition de blocs présenté dans la partie précédente, le cadre du diagramme représentant la frontière d'un bloc.
- Il introduit la notion fondamentale de “**port**” qui correspond à un point d'interaction avec l'extérieur du bloc.
- Les connecteurs (traits) entre les ports indiquent soit les associations soit les **flux de matière**, d'énergie et d'information entre les différents blocs.

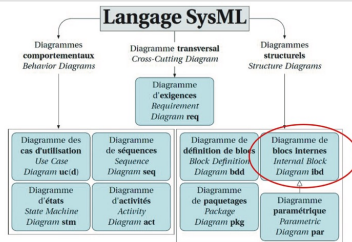




## Diagramme de blocs internes

### Diagramme de blocs internes (ibd)

- Le **diagramme de blocs internes** est un *diagramme structurel* appelé **Internal Block Diagram (ibd)** dans le langage SysML.
- Le diagramme de blocs internes est rattaché à un bloc issu du diagramme de définition de blocs présenté dans la partie précédente, le cadre du diagramme représentant la frontière d'un bloc.
- Il introduit la notion fondamentale de “port” qui correspond à un point d'interaction avec l'extérieur du bloc.
- Les connecteurs (traits) entre les ports indiquent soit les associations soit les flux de matière, d'énergie et d'information entre les différents blocs.



## Diagramme de blocs internes (ibd)

- 
- Diagramme de classification des diagrammes SysML. Le langage SysML se divise en trois catégories principales : Diagrammes comportementaux (Behavior Diagrams), Diagramme transversal (Cross-Cutting Diagram), et Diagrammes structurels (Structure Diagrams). Les diagrammes comportementaux incluent le Diagramme des cas d'utilisation (UML Use Case Diagram), le Diagramme d'états (State Machine Diagram), le Diagramme de séquences (Sequence Diagram), et le Diagramme d'activités (Activity Diagram). Le diagramme transversal est le Diagramme de Cross-Cutting. Les diagrammes structurels incluent le Diagramme de définition de blocs (Block Definition Diagram), le Diagramme de paquetages (Package Diagram), le Diagramme de blocs internes (Internal Block Diagram), et le Diagramme paramétrique (Parametric Diagram).
- ```

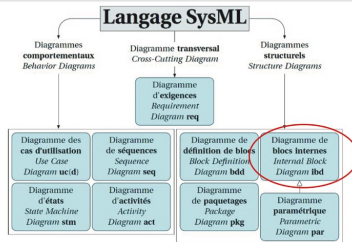
graph TD
    SysML[Langage SysML] --> BD[Diagrammes comportementaux  
Behavior Diagrams]
    SysML --> TCD[Diagramme transversal  
Cross-Cutting Diagram]
    SysML --> SD[Diagrammes structurels  
Structure Diagrams]
    BD --> UC[Diagramme des cas d'utilisation  
Use Case Diagram uc(d)]
    BD --> SM[Diagramme d'états  
State Machine Diagram stm]
    TCD --> REQ[Diagramme d'exigences  
Requirement Diagram req]
    SD --> BDD[Diagramme de définition de blocs  
Block Definition Diagram bdd]
    SD --> IBD[Diagramme de blocs internes  
Internal Block Diagram ibd]
    SD --> PKG[Diagramme de paquetages  
Package Diagram pkg]
    SD --> PAR[Diagramme paramétrique  
Parametric Diagram par]
    IBD --> PKG
  
```



## Diagramme de blocs internes

### Diagramme de blocs internes (ibd)

- Le **diagramme de blocs internes** est un *diagramme structurel* appelé **Internal Block Diagram (ibd)** dans le langage **SysML**.
- Le diagramme de blocs internes est rattaché à un bloc issu du diagramme de définition de blocs présenté dans la partie précédente, le cadre du diagramme représentant la frontière d'un bloc.
- Il introduit la notion fondamentale de “**port**” qui correspond à un point d'interaction avec l'extérieur du bloc.
- Les connecteurs (traits) entre les ports indiquent soit les associations soit les **flux de matière, d'énergie et d'information** entre les différents blocs.





## Diagramme de blocs internes

### Diagramme de blocs internes (ibd)

- Le **diagramme de blocs internes** est un *diagramme structurel* appelé **Internal Block Diagram (ibd)** dans le langage **SysML**.
- Le diagramme de blocs internes est rattaché à un bloc issu du diagramme de définition de blocs présenté dans la partie précédente, le cadre du diagramme représentant la frontière d'un bloc.
- Il introduit la notion fondamentale de “**port**” qui correspond à un point d'interaction avec l'extérieur du bloc.
- Les connecteurs (traits) entre les ports indiquent soit les associations soit les **flux de matière, d'énergie et d'information** entre les différents blocs.

### Représentation graphique

- Les ports se représentent par un carré placé sur le contour du bloc :
  - les ports **flux** : indiquent les échanges de matière, d'énergie et d'information entre blocs : ce type de port contient une flèche dont le sens (entrante, sortante ou bidirectionnelle) indique celui du flux;
  - les ports **standards** : indiquent la logique de commande et les interfaces d'un bloc : ce type de port ne contient pas d'indication particulière.



## Diagramme de blocs internes

### Diagramme de blocs internes (ibd)

- Le **diagramme de blocs internes** est un *diagramme structurel* appelé **Internal Block Diagram (ibd)** dans le langage **SysML**.
- Le diagramme de blocs internes est rattaché à un bloc issu du diagramme de définition de blocs présenté dans la partie précédente, le cadre du diagramme représentant la frontière d'un bloc.
- Il introduit la notion fondamentale de “**port**” qui correspond à un point d'interaction avec l'extérieur du bloc.
- Les connecteurs (traits) entre les ports indiquent soit les associations soit les **flux de matière, d'énergie et d'information** entre les différents blocs.

### Représentation graphique

- Les ports se représentent par un carré placé sur le contour du bloc :
  - les ports *flux* : indiquent les échanges de matière, d'énergie et d'information entre blocs : ce type de port contient une flèche dont le sens (entrante, sortante ou bidirectionnelle) indique celui du flux ;
  - les ports *standards* : indiquent la logique de commande et les interfaces d'un bloc : ce type de port ne contient pas d'indication particulière.



## Diagramme de blocs internes

### Diagramme de blocs internes (ibd)

- Le **diagramme de blocs internes** est un *diagramme structurel* appelé **Internal Block Diagram (ibd)** dans le langage **SysML**.
- Le diagramme de blocs internes est rattaché à un bloc issu du diagramme de définition de blocs présenté dans la partie précédente, le cadre du diagramme représentant la frontière d'un bloc.
- Il introduit la notion fondamentale de “**port**” qui correspond à un point d'interaction avec l'extérieur du bloc.
- Les connecteurs (traits) entre les ports indiquent soit les associations soit les **flux de matière, d'énergie et d'information** entre les différents blocs.

### Représentation graphique

- Les ports se représentent par un carré placé sur le contour du bloc :
  - les ports *flux* : indiquent les échanges de matière, d'énergie et d'information entre blocs : ce type de port contient une flèche dont le sens (entrante, sortante ou bidirectionnelle) indique celui du flux ;
  - les ports *standards* : indiquent la logique de commande et les interfaces d'un bloc : ce type de port ne contient pas d'indication particulière.



## Diagramme de blocs internes

### Diagramme de blocs internes (ibd)

- Le **diagramme de blocs internes** est un *diagramme structurel* appelé **Internal Block Diagram (ibd)** dans le langage **SysML**.
- Le diagramme de blocs internes est rattaché à un bloc issu du diagramme de définition de blocs présenté dans la partie précédente, le cadre du diagramme représentant la frontière d'un bloc.
- Il introduit la notion fondamentale de “**port**” qui correspond à un point d'interaction avec l'extérieur du bloc.
- Les connecteurs (traits) entre les ports indiquent soit les associations soit les **flux de matière, d'énergie et d'information** entre les différents blocs.

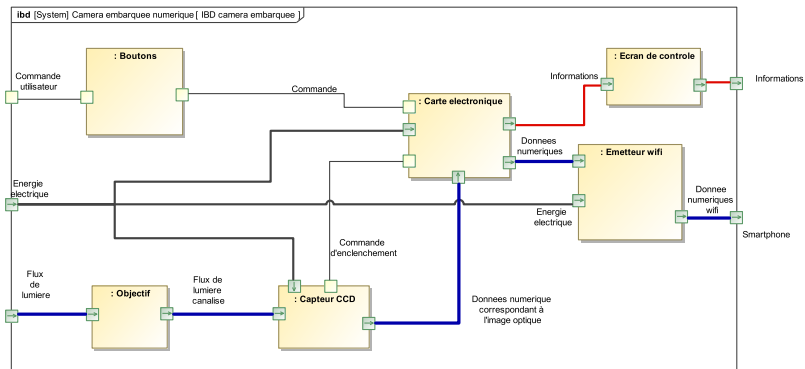
### Représentation graphique

- Les ports se représentent par un carré placé sur le contour du bloc :
  - les ports *flux* : indiquent les échanges de matière, d'énergie et d'information entre blocs : ce type de port contient une flèche dont le sens (entrante, sortante ou bidirectionnelle) indique celui du flux ;
  - les ports *standards* : indiquent la logique de commande et les interfaces d'un bloc : ce type de port ne contient pas d'indication particulière.





## Diagramme de blocs internes

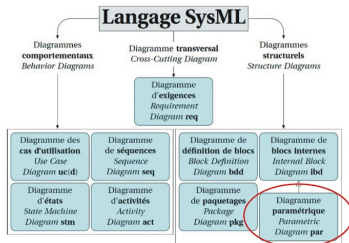




## Diagramme paramétrique

### Diagramme paramétrique (par)

- Le **diagramme paramétrique** est un *diagramme structurel* appelé **Parametric Diagram (par)** dans le langage SysML.
- Ce diagramme est une extension du diagramme de blocs internes (ibd présenté précédemment) et il partage donc les mêmes éléments graphiques.
- Il présente la particularité de pouvoir connecter entre elles des contraintes ajoutées au diagramme de blocs par le biais d'un bloc particulier, dit "**de contraintes**" (*constraint block*) qui contient des paramètres et une relation, en général *mathématique*, les reliant.

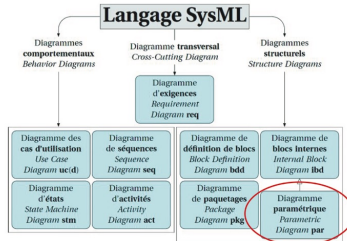




## Diagramme paramétrique

### Diagramme paramétrique (par)

- Le **diagramme paramétrique** est un *diagramme structurel* appelé **Parametric Diagram (par)** dans le langage SysML.
- Ce diagramme est une extension du diagramme de blocs internes (**ibd** présenté précédemment) et il partage donc les mêmes éléments graphiques.
- Il présente la particularité de pouvoir connecter entre elles des contraintes ajoutées au diagramme de blocs par le biais d'un bloc particulier, dit "**de contraintes**" (*constraint block*) qui contient des paramètres et une relation, en général *mathématique*, les reliant.

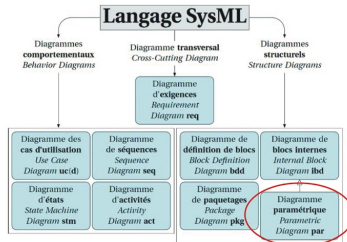




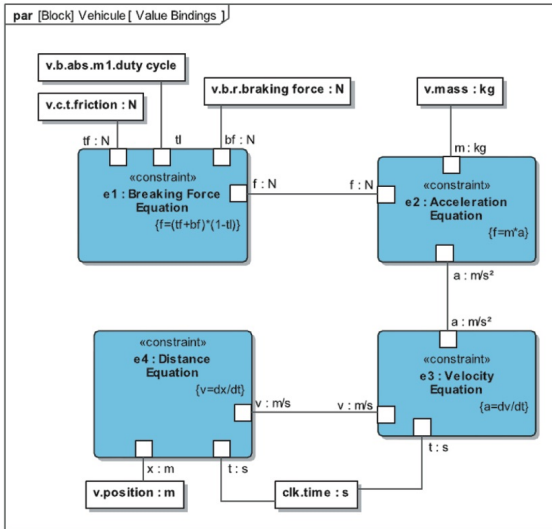
## Diagramme paramétrique

### Diagramme paramétrique (par)

- Le **diagramme paramétrique** est un *diagramme structurel* appelé **Parametric Diagram (par)** dans le langage SysML.
- Ce diagramme est une extension du diagramme de blocs internes (**ibd** présenté précédemment) et il partage donc les mêmes éléments graphiques.
- Il présente la particularité de pouvoir connecter entre elles des contraintes ajoutées au diagramme de blocs par le biais d'un bloc particulier, dit “**de contraintes**” (*constraint block*) qui contient des paramètres et une relation, en général *mathématique*, les reliant.



## Diagramme paramétrique





# Plan

- 1 **Présentation du langage SysML**
  - Intérêts et objectifs
  - Architecture du langage
  - Éléments de syntaxe
  - Présentation du support du cours
- 2 **Analyse du contexte et des exigences du système**
  - Diagramme de contexte
  - Diagramme des exigences
  - Diagramme des cas d'utilisation
- 3 **Analyse structurelle du système**
  - Diagramme de définition des blocs
  - Diagramme de blocs internes
  - Diagramme paramétrique
- 4 **Modélisation structurelle : chaîne d'info et d'énergie**
  - La chaîne d'information
  - La chaîne d'énergie
  - Les interfaces

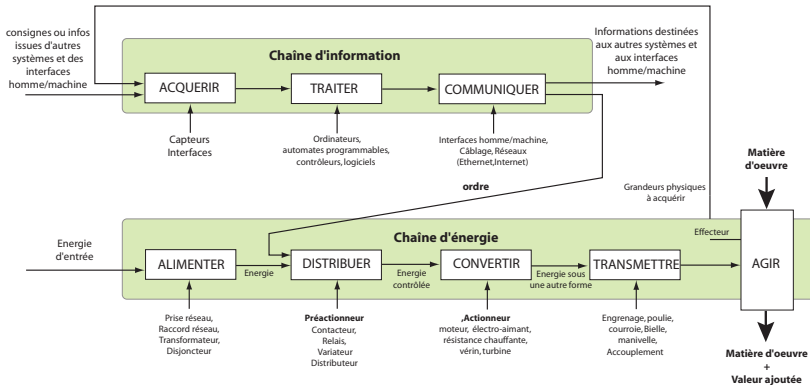


## Chaîne fonctionnelle

### Définition

Elle est constituée :

- d'une **chaîne d'information**,
- d'une **chaîne d'énergie**.





## Chaîne fonctionnelle

### Définition

Elle est constituée :

- **Chaîne d'information :**

- **acquérir** : capteurs, interfaces,
- **traiter** : automates programmables, ordinateurs,
- **communiquer** : interfaces, câblages, réseaux.

- **Chaîne d'énergie :**

- **alimenter** : prise réseau, raccord, batterie,
- **distribuer** : contacteurs, distributeurs, électro-vannes,
- **convertir** : moteur électrique, vérin,
- **agir** : tapis roulant, ventouse, roue, balais.



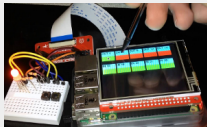


## Chaîne fonctionnelle : les interfaces

### Les interfaces

Les interfaces permettent :

- d'acquérir certaines données de la chaîne d'énergie vers la chaîne d'information (capteurs).
- d'acquérir ou de communiquer des informations de l'utilisateur du système



Interface de communication sur une *raspberry-pi*



Centrale inertielle de drone