Centre d'intérêt 3

Architecture des systèmes

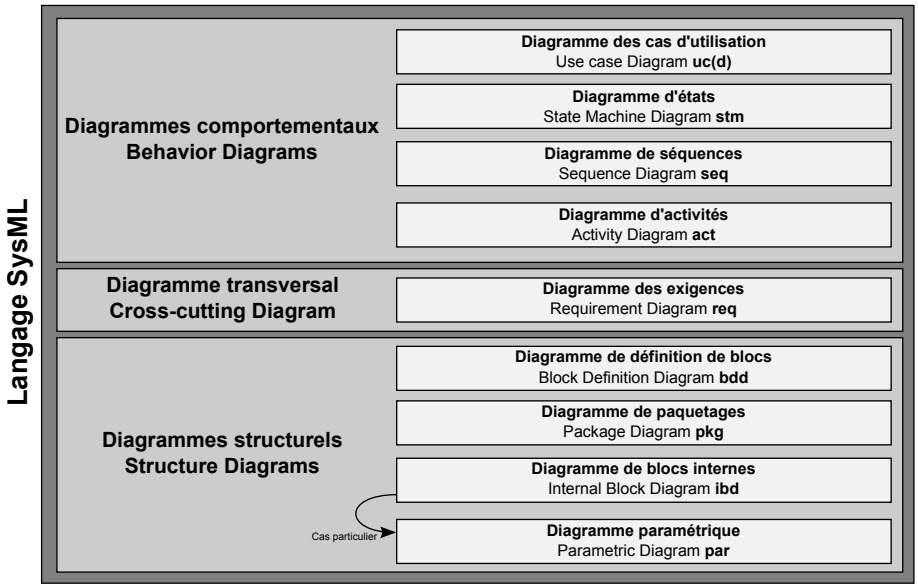
Diagrammes SysML

PSI - MP : Lycée Rabelais

Les systèmes techniques actuels sont d'une grande complexité. Ils incluent très souvent des technologies très variées : mécanique, électronique, optique, informatique... Cette pluri-technologie nécessite, lors de la conception du produit, un panel d'experts ayant, chacun d'eux, leur spécialité. Chacun de ces acteurs doit donc pouvoir participer à la conception du produit. Cela nécessite donc d'utiliser un langage commun afin de définir le cahier des charges.

Ce langage devra être suffisamment complexe pour décrire avec finesse les attentes du produit. Il devra également être suffisamment simple pour que chacun des acteurs puisse l'utiliser. C'est dans ce cadre qu'est né le langage SysML (**Sys**tems **M**odeling **L**anguage). Il fut développé au début des années 2000 et est standardisé depuis 2007.

Ce langage SysML a donc pour objectif de formaliser, de manière graphique, les spécifications associées à un système technique. Il permet de spécifier, de concevoir, de définir et d'analyser la structure d'un système. Il permettra également d'identifier les performances, les limites et l'environnement du système. **Il a donc, avant tout, un objectif de documentation !**



1 Les différents diagrammes SysML

Durant une phase de conception, il est courant d'aborder le problème selon trois points de vue :

- **Point de vue fonctionnel.** À travers ce point de vue, on cherche à déterminer les différentes exigences auxquelles le système devra répondre.
- **Point de vue comportemental.** L'objectif est ici de décrire la manière dont le système évoluera au cours du temps (lors de son utilisation).

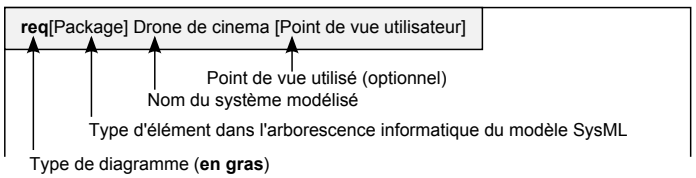
- **Point de vue structurel.** On cherche ici à savoir comment et par quels moyens (techniques) le système pourra répondre aux exigences.

Le langage SysML met à disposition, pour ces trois approches, des outils graphiques spécifiques. Le langage est donc développé autour de neuf diagrammes qui permettent donc de représenter le système sous différents points de vue.

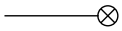
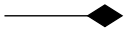
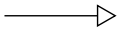
On retrouve donc les diagrammes détaillés ci-dessous.

- **Diagrammes comportementaux :**
  - Le diagramme des cas d'utilisation (*Use case Diagram* : **ucd**) permet de décrire les objectifs poursuivis par l'acteur primaire (souvent l'utilisateur) avec le système. Des acteurs secondaires peuvent également être indiqué sur le diagramme.
  - Le diagramme de séquence (*Sequence Diagram* : **seq**) décrit l'enchaînement des messages passés entre différents blocs en interaction.
  - Le diagramme d'état (*State Machine Diagram* : **stm**) décrit les différents états du système et les transitions entre ces états.
  - Le diagramme d'activité (*Activity Diagram* :**act**) décrit l'enchaînement des différentes tâches réalisées par le système. **(Il n'est pas au programme !)**
- **Diagrammes structurels :**
  - Le diagramme de définition de blocs (*Block Definition Diagram* : **bdd**) décrit l'architecture (les différents éléments) du système.
  - Le diagramme de blocs internes (*Internal Block Diagram* : **ibd**) décrit ici l'architecture à l'intérieur d'un bloc.
  - Le diagramme paramétrique (*Parametric Diagram* : **par**) est un cas particulier du diagramme de blocs internes mais cette fois-ci l'architecture n'est décrite qu'à partir des équations qui "gouvernent" le fonctionnement du bloc. **(Il n'est pas au programme !)**
  - Le diagramme de paquetages (*Package Diagram* : **pkg**) décrit l'organisation logique du modèle. **(Il n'est pas au programme !)**
- **Diagramme transversal :** C'est le diagramme des exigences (*Requirement Diagram* : **req**) qui permet de décrire ce que doit réaliser le système et les contraintes associées.

Peu importe le diagramme utilisé, des règles graphiques sont à respecter. Pour repérer le diagramme (et pour savoir de quel type de diagramme il s'agit), celui-ci dispose d'une cartouche (voir ci-dessous). Cette cartouche est renseignée de plusieurs éléments.



Les diagrammes sont ensuite complétés. Ils sont composés de rectangles, d'ellipses et de lignes. Le choix du type de lien permet d'exprimer la nature de la relation entre les éléments connectés. Le tableau ci-dessous récapitule les concepts liés à l'utilisation des différentes lignes.

Lien	Signification
	Ce lien exprime une relation de **contenance** ou d'**inclusion**. Elle permet de décomposer une exigence en plusieurs sous-exigences.
	Ce lien exprime une relation d'**association**. Elle permet de lier deux éléments entre-eux. Si cette relation est unidirectionnelle, il y aura une flèche en fin de ligne. Si la relation est bidirectionnelle, la ligne sera sans flèche.
	Ce lien exprime des relations de d'**inclusion**, d'**extension**, de **raffinement** ou de **dérivation** entre différents blocs dans un cas d'utilisation ou dans le diagramme des exigences. Cela permet de donner essentiellement une relation hiérarchique entre les éléments.
	Ce lien exprime une relation de **composition**. Cela signifie qu'un élément est structurellement indispensable à l'autre.
	Ce lien exprime une relation d'**agrégation**. Ce lien a la même signification que le précédent mais avec un sens moins fort. Elle indique donc qu'un élément est présent mais de manière optionnelle.
	Ce lien exprime une relation de **spécialisation**. Ce lien sera utilisé dans les diagrammes des cas d'utilisation pour caractériser en détail un utilisateur qui peut avoir différentes fonctions (par exemple l'utilisateur d'une voiture pourra être spécialisé comme un passager ou un conducteur).

2 Application : Drone de Cinéma



Pour réaliser des prises de vue aériennes en haute définition lors de la réalisation de films ou de documentaires, des drones sont fréquemment utilisés. Ces appareils sont souvent loués par la production à des entreprises spécialisées qui fournissent, outre le matériel, un pilote maîtrisant parfaitement le vol de l'engin. Ce mode prise de vue permet d'obtenir des prix très attractifs (en comparaison avec une solution en hélicoptère ou en avion).

Fonction	Critère	Niveau	Flexibilité
Assurer la qualité des images	Temps de visée d'une cible sans bouger	5s	mini
	Vitesse de déplacement maximale	1 m/s	+/- 0,2 m/s
Permettre de couvrir la zone de prise de vue	Taille de la zone	300m x 300m	
	Altitude maxi de vol	60m	mini
	Durée de vol	8h	mini
S'adapter à la caméra	Fixation universelle		
	Capacité de charge	1 kg	mini
Assurer le retour d'image au sol	Délai	mini	
	Résolution	854x480	
Résister à l'environnement	Déplacement admissible suite à un vent de 20 km/h	0,5 cm et 1,5° dans chaque direction	maxi
Être économe	Prix pour la mission (1 jour)	4000 €	maxi

Afin de pouvoir "gagner le contrat", il est nécessaire de répondre au cahier des charges du client mais aussi d'être compétitif vis-à-vis de la concurrence. L'ingénierie système est une discipline récente qui s'inscrit dans cette démarche. Elle est née dans les années 80 en partant du constat qu'à cette époque 85% des projets industriels présentent des dépassements de budget ou de délais.

Ces échecs sont attribués à la complexité technique et organisationnelle de plus en plus grande des projets. En effet, les systèmes techniques actuels font appel à de nombreuses technologies (mécanique, électronique, informatique, télécommunications ...). Leur conception nécessite l'intervention d'acteurs issus de divers métiers.

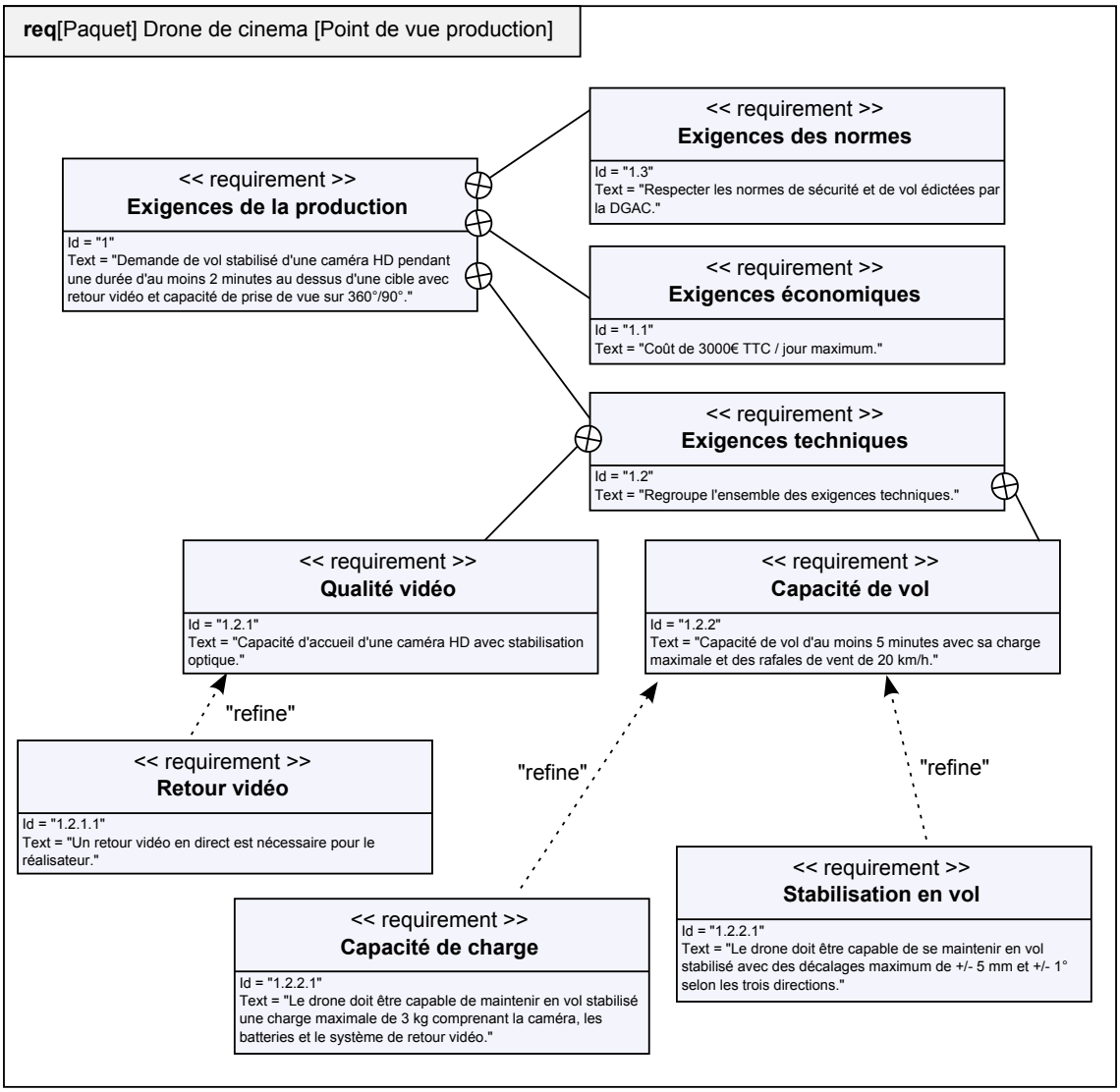
Le rôle de "l'ingénieur système" est d'utiliser des méthodes et des démarches interdisciplinaires permettant d'aborder les systèmes complexes pluri-technologiques afin de les concevoir, de les étudier, de vérifier leurs performances et de les améliorer.

C'est ce que nous allons tenter de faire ici à travers la description des différents diagrammes SysML.

2.1 Diagramme des exigences

Le diagramme des exigences (*Requirement Diagram* (**req**)) est le seul diagramme transversal du langage SysML. L'objectif de ce diagramme est de modéliser les attentes du client devant être vérifiées par le système. Ce diagramme traduit, par des fonctionnalités ou des contraintes, ce qui doit être satisfait par le système. Ces différentes exigences peuvent être reliées par des liens.

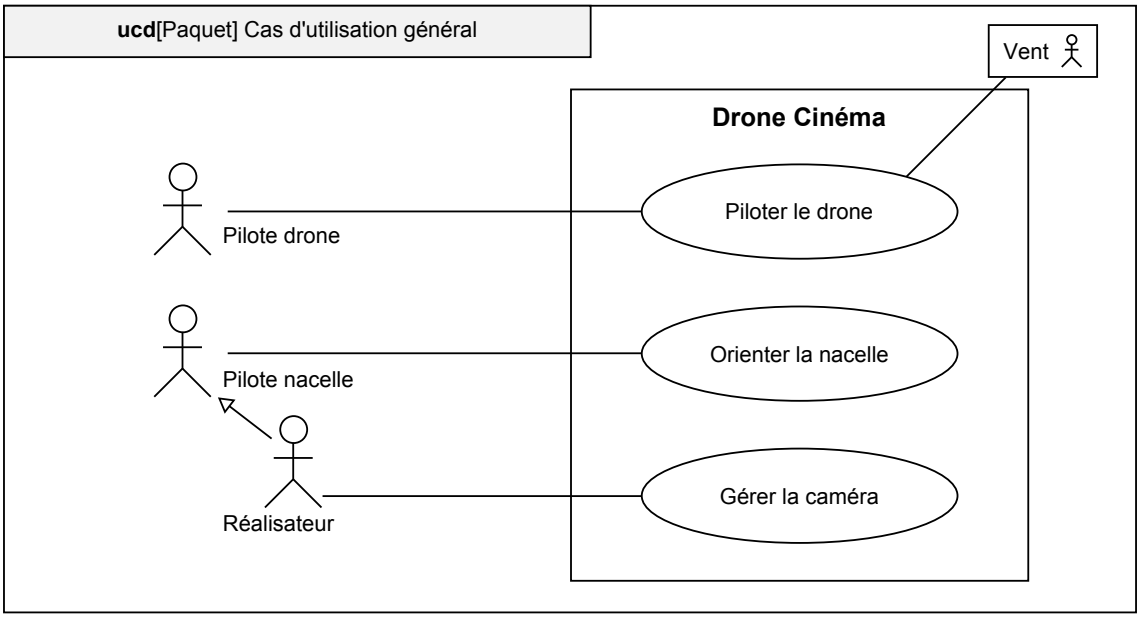
Diagramme des exigences pour le drone de cinéma



2.2 Diagramme des cas d'utilisation

Le diagramme des cas d'utilisation est un diagramme comportemental (*Use Case Diagram* (**uc** ou **ucd**)). Ce diagramme a pour objectif de montrer les fonctionnalités offertes par le système en identifiant les services. Il met donc en relation bidirectionnelle les différents acteurs et les services rendus par le système.

Diagramme des cas d'utilisation pour le drone de cinéma



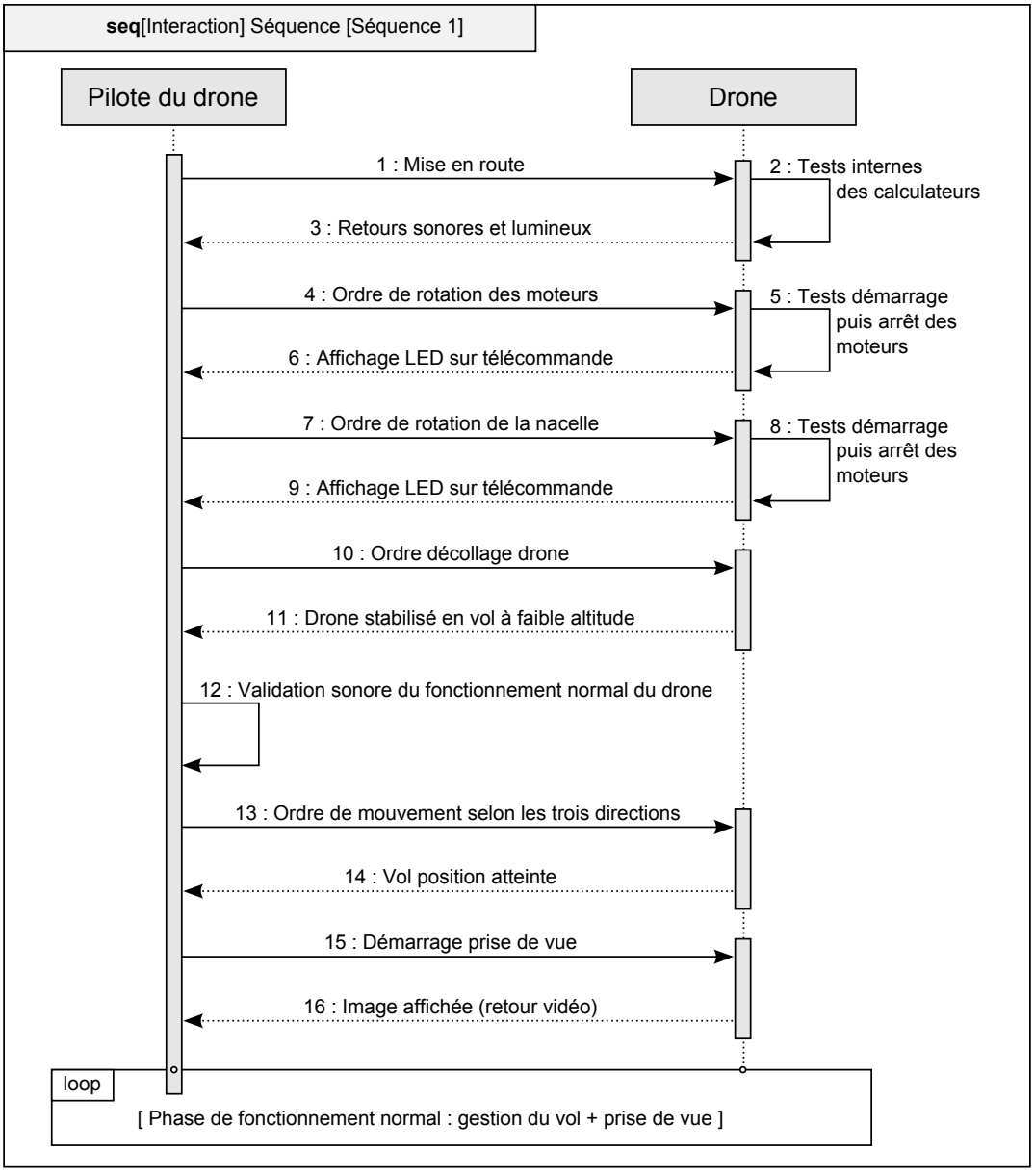
2.3 Diagramme de séquence

L'objectif du diagramme de séquence (*Sequence Diagram* (**seq**)) est de décrire les interactions entre plusieurs entités (acteurs, système ou sous-systèmes). Ce diagramme permet de représenter ces interactions avec un point de vue chronologique (même si la notion de temps n'apparaît pas).

Un diagramme de séquence est rattaché à un cas d'utilisation et décrit ce dernier en entier ou en partie. Il peut donc s'agir de la représentation de l'évolution du fonctionnement dans un cas "positif" (fonctionnement normal par exemple). Mais cela peut aussi être la représentation de l'évolution dans un cas "négatif" (cas d'une panne par exemple).

Graphiquement, ce diagramme est composé de plusieurs éléments. Des traits verticaux indiquent les différents acteurs (utilisateurs, système, sous-systèmes). Des traits horizontaux indiquent l'envoie et la réception d'information.

Diagramme de séquence pour le drone de cinéma



2.4 Diagramme de définition de blocs

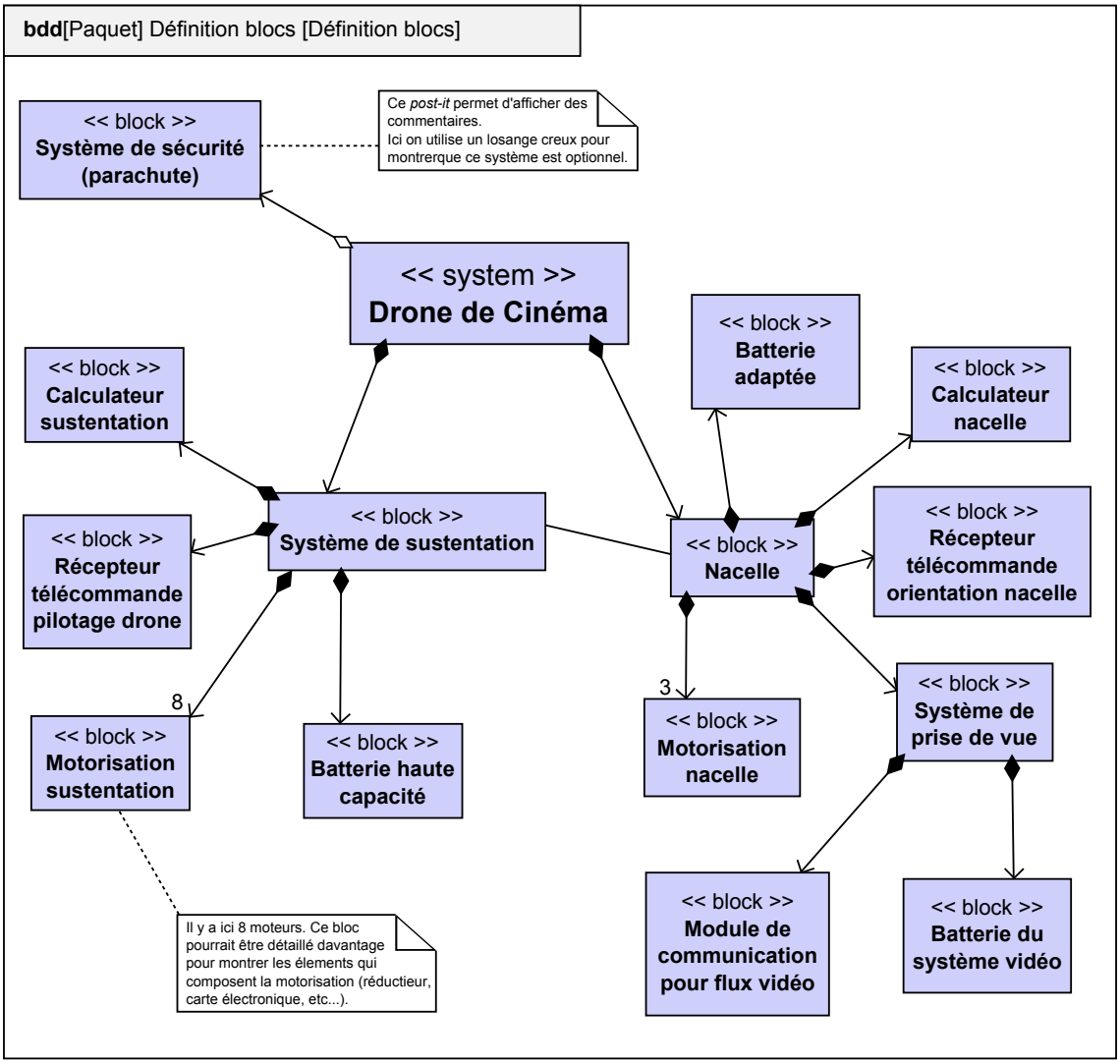
Le diagramme de définition de blocs (*Block Definition Diagram* (**bdd**)) est un diagramme structurel. L'objectif de ce diagramme est de décrire le système à l'aide de blocs, chaque bloc représentant un élément du système.

Ce diagramme représente (souvent) les caractéristiques principales de chaque bloc ainsi que les différents liens entre-eux. Il permet donc une modélisation complète de l'architecture.

Dans chacun des blocs, il est possible d'insérer des propriétés. Les grandeurs les plus classiquement utilisées sont : *value* et *part*. *Value* permet d'exprimer les constantes numériques liées au système. *Part* permet, quant à lui, de représenter les sous-éléments qui composent le bloc (des vis, un réducteur, etc...).

Les différents blocs doivent ensuite être reliés. Le lien utilisé devra montrer la dépendance d'un bloc par rapport à son vis-à-vis. En général, on utilisera un lien d'agrégation (trait avec losange plein ou vide). Ce lien permet de décomposer un éléments en sous-éléments et ainsi de suite. Il est également possible d'ajouter un nombre en bout de lien pour exprimer la multiplicité d'un bloc (s'il y a 8 moteurs identiques, on indiquera 8 mais on ne dessinera pas 8 blocs !).

Diagramme de définition de blocs pour le drone de cinéma



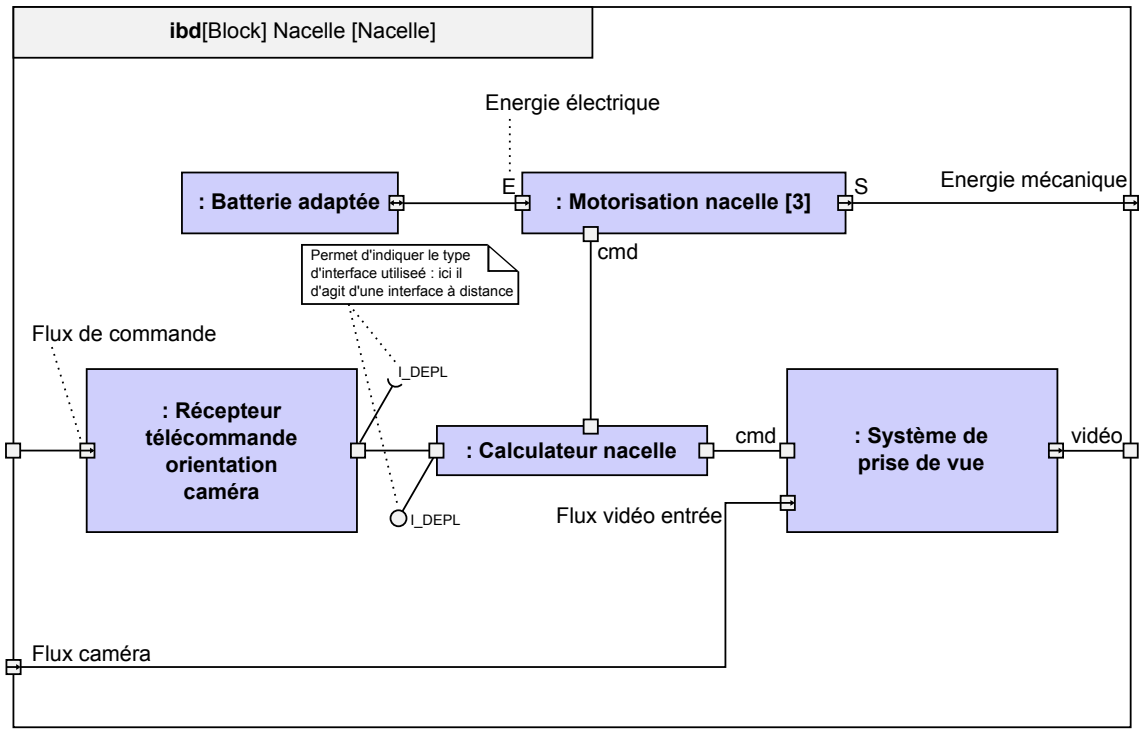
2.5 Diagramme de blocs internes

Le diagramme de blocs internes (*Internal Block Diagram* (**ibd**)) est également un diagramme structurel. L'objectif de ce diagramme est de décrire plus précisément le fonctionnement d'un bloc. Ce diagramme est donc rattaché à un bloc issu du diagramme de définition de blocs. Le cadre du diagramme représente ici la frontière du bloc.

Une notion importante dans ce bloc est la notion de **port** qui correspond à un point d'interaction avec l'extérieur du bloc (c'est là que viendront "s'attacher" les autres blocs du diagramme de définition). Les connecteurs entre les différents ports permettent de montrer le cheminement des différents flux (flux de matière, d'énergie ou d'information).

Les ports sont schématisés par des petits carrés en périphérie des blocs. Si le sens du flux est connu, on placera dans ce port un flèche (entrante, sortante ou bidirectionnelle) pour indiquer le sens du flux. On parlera alors de **ports de flux**. S'il n'y a pas de flux particulier, on parlera de **port standard** et dans ce cas-là, on ne place pas de flèche dans le port.

Diagramme de blocs internes pour l'élément nacelle du drone de cinéma



2.6 Diagramme d'états

Le diagramme d'états (*State Machine Diagram* (**stm**)) est un diagramme comportemental. Le diagramme d'état est rattaché à un bloc : ce bloc peut être le système entier, un sous-système ou n'importe quel élément.

Un état est une situation (d'une durée finie) durant laquelle un système exécute une activité ou est en attente d'un évènement. Le passage d'un état à un autre se fait en franchissant une transition.

Les états sont représentés graphiquement par des rectangles aux coins arrondis. Les transitions entre ces états sont représentées par des flèches orientées ayant un état de départ et un état d'arrivé. La transition sera franchie si l'état en amont est actif et si la transition est validée.

La transition s'écrit d'une manière générale sous la forme `évènement [condition de garde] / action`. Chacune des closes `évènement`, `condition de garde` ou `action` est facultative.

- `évènement` aussi appelé **déclencheur** est une expression booléenne qui représente l'apparition d'un évènement et provoque le franchissement de la transition. On utilise fréquemment les deux attributs suivants pour détailler les évènements :
  - *when('condition booléenne')* s'active à l'instant précis où la condition devient vraie. L'évènement est "*la condition booléenne devient vraie*". Cela est équivalent à la syntaxe ↑, parfois rencontrée, devant l'expression pour spécifier que l'on regarde spécifiquement le front montant associé à la condition.
  - *after(temps)* s'active à l'instant précis où le temps spécifié s'est écoulé.
- `condition de garde` permet de rajouter une condition au passage de la transition. Si l'évènement apparait, la condition de garde sera évaluée. Si la condition est respectée lorsque l'évènement apparaît alors la transition sera franchie,
- `action` permet d'exécuter une action à l'instant où le franchissement de la transition est effectuée.

Il est également possible de rajouter des évènements internes à chacun des états en ajoutant la ou les closes `entry`, `do` et `exit`. `Entry` indique que l'action est effectuée en entrant dans l'état (sauf indication contraire, elle n'est jamais interrompue). `Do` indique que l'action est réalisée dans l'état et s'arrête en sortant de l'état. `Exit` indique l'action qui sont effectuée en sortant de l'état (sauf indication contraire, elle n'est jamais interrompue).

Diagramme d'état du drone de cinéma

