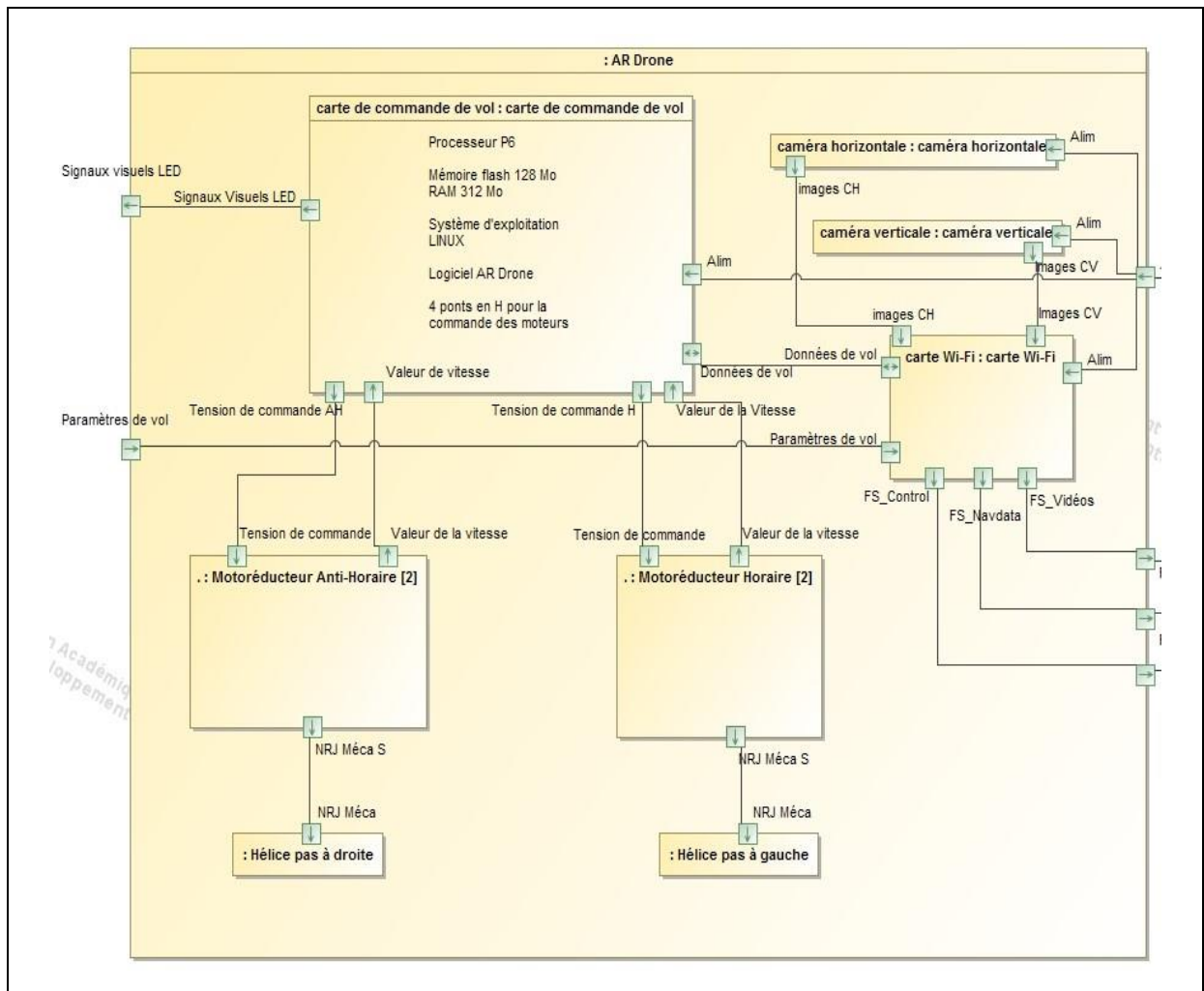


INGENIERIE SYSTEME

MODELISER ET PREVOIR LES PERFORMANCES DES SYSTEMES TECHNIQUES



1) Introduction à l'Ingénierie Système

L'ingénierie des systèmes est une approche scientifique interdisciplinaire, dont le but est de formaliser et d'appréhender la conception de systèmes complexes. L'ingénierie des systèmes se focalise sur la définition des besoins du client et des exigences fonctionnelles, détectés tôt dans le « cycle de vie du système », en documentant les exigences, puis en poursuivant avec la synthèse de la conception et la validation du système.

Au cours des 2 dernières décennies, l'informatique est de plus en plus présente dans les systèmes créés par l'homme afin de répondre à ses besoins. Les performances obtenues sont devenues habituelles et plus personne ne s'étonne de ce qu'on est capable de faire dans des domaines qui associent la technologie avec les besoins industriels aussi bien qu'avec les loisirs. Par exemple, l'AR Drone de Parrot qui va être présenté par la suite trouve des applications professionnelles pour le cinéma, voire la surveillance aussi bien que pour le jeu ou la publicité. Il n'en reste pas moins que les moyens mis en œuvre sont de haute technologie et ont nécessité plus de 5 ans de recherche à une équipe d'ingénieurs.

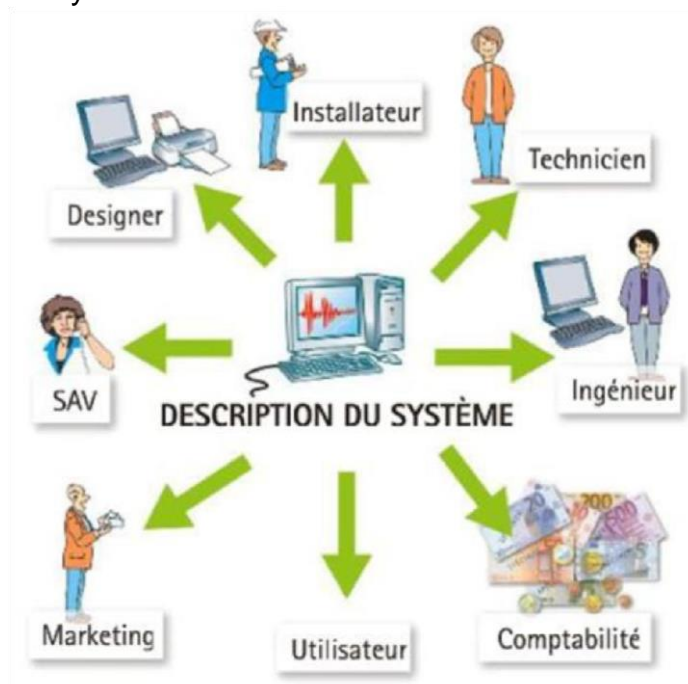
A noter qu'une fois la technologie développée, il est très courant de voir naître des systèmes équivalents. Par exemple de très nombreux drones de conception diverses ont vu le jour.

L'évolution permanente des technologies et le besoin naturel de nouveautés engendre la nécessité de créer de nouveaux produits, de nouveaux systèmes. La complexité des systèmes actuels impose des outils performants permettant la description à différents niveaux. Il est nécessaire de définir complètement les différents **composants physiques** du système ainsi que les **échanges d'information et d'énergie** entre ces composants.

Cette création s'appuie en grande partie sur le passé qui, avec l'ajout de nouveautés permet d'innover. La conception de système donne souvent lieu à une accumulation de documentations qui doivent toutes être croisées et mises à jour pour maintenir la cohérence et respecter les spécifications du système.



Le langage SysML est un moyen de Il permet ainsi d'éviter tout problème de regrouper dans un modèle commun à tous les communication qui engendrerai du retard corps de métiers, les spécifications, les dans l'élaboration du produit. contraintes, et les paramètres de l'ensemble du système.



2) Étapes du cycle de vie d'un système

La notion de « cycle de vie » est indissociable d'un système. Elle exprime les différentes étapes qui vont de l'analyse du besoin jusqu'à l'élimination et/ou le recyclage de ses constituants.



Analyser le besoin

« Un besoin est une nécessité ou un désir éprouvé par un utilisateur » NF X 50-150. Le client est sensible à l'évolution du contexte économique, social et environnemental ainsi qu'au degré d'innovation, le besoin évolue donc constamment. Exemple : le vélo à assistance électrique.



EMI/Philips de 1935/37
simplex



ISD City 3 – 2004
Batterie 36 V



VAE Tidal Force IO-Cruiser – Matra -2010
Moteur Matra Brushless DC
dans le moyeu arrière
Batterie Ni-MH 36 V 9 Ah
dans moyeu avant



VAE du futur.... ? ??
– Concept Peugeot –



Établir le cahier des charges fonctionnel (CdCF)

Avant d'imposer une solution, il faut se tourner vers le demandeur, pour aboutir de manière structurée à la solution. En effet, le but d'un projet est de satisfaire le besoin. Il faut exprimer

clairement les objectifs à atteindre d'un projet, afin d'éviter toute confusion entre vous et le demandeur.

Ce besoin doit être exprimé dès le lancement du projet. Il s'agit d'expliciter quelle est l'exigence fondamentale qui justifie la conception du produit.

Le cahier des charges fonctionnel est un document qui permet de formaliser avec précision le besoin du demandeur. En effet, le C.d.C.F. est un tableau de bord qui définit le projet et détaille les conditions dans lesquelles il doit être réalisé. C'est le lien de compréhension entre l'entreprise et le client.

La partie technique d'un C.d.C.F. doit se limiter à *énumérer les contraintes techniques avérées*.

Les contraintes de base sont économiques (les contraintes monétaires comme le budget de fonctionnement), environnementales (le caractère recyclable du produit, etc.), humaines (par exemple, dans le cas d'un jouet pour enfant, il doit être léger, ne pas contenir de petites pièces, etc.), industrielles (par exemple, il doit être fabriqué au Canada) et matérielles (par exemple, il doit spécifier les morceaux qui peuvent être remplacés, comme des piles, il doit préciser le recours à tel ou tel serveur d'applications).

- **Concevoir** : A partir du cahier des charges fonctionnel, le bureau d'études modélise le produit sous forme de maquette numérique.
- **Matières premières** : Le choix des matières premières est important dans le cycle de vie puisqu'il conditionne le prix du produit. Il faut également penser au recyclage du produit donc aux choix des matériaux.
- **Industrialiser** : L'industrialisation est le processus de fabrication de produits avec des techniques permettant une forte productivité du travail.
- **Homologuer** : L'homologation est la certification conforme d'un produit à une norme, ou une réglementation. En d'autres termes l'homologation garantit au consommateur que le produit qu'il achète correspond à ce qu'il est en droit d'en attendre.
- **Transporter** : Le transport du produit est un point important dans l'analyse du cycle de vie du produit.
- **Commercialiser – Utiliser le produit** : C'est là qu'on peut vérifier la pertinence du produit et la satisfaction du client.
- **Éliminer – Recycler** : L'objectif actuel est de recycler un maximum de composants mais beaucoup de matériaux ne peuvent pas être recyclés. Cette phase doit être

pensée dès le début en parallèle de tous les choix de conception et de matériaux. Elle fait partie des règles édictées par les lois du « Grenelle de l'environnement ».

3) Le Langage : SysML (System Modeling Language)

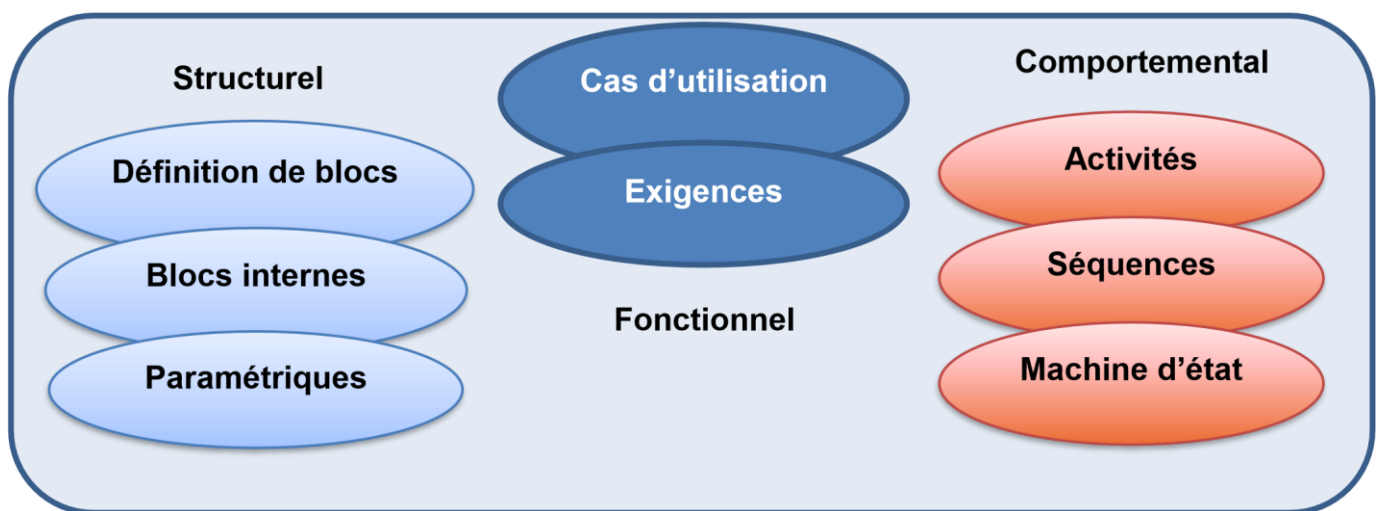
Le SysML, comme on le nomme couramment, est un ensemble de représentations graphiques permettant d'appréhender le système sous tous ses aspects, de façon très complète.

Chaque graphique est appelé **diagram** en anglais, et par abus de langage, nous le nommerons *diagramme* en français.

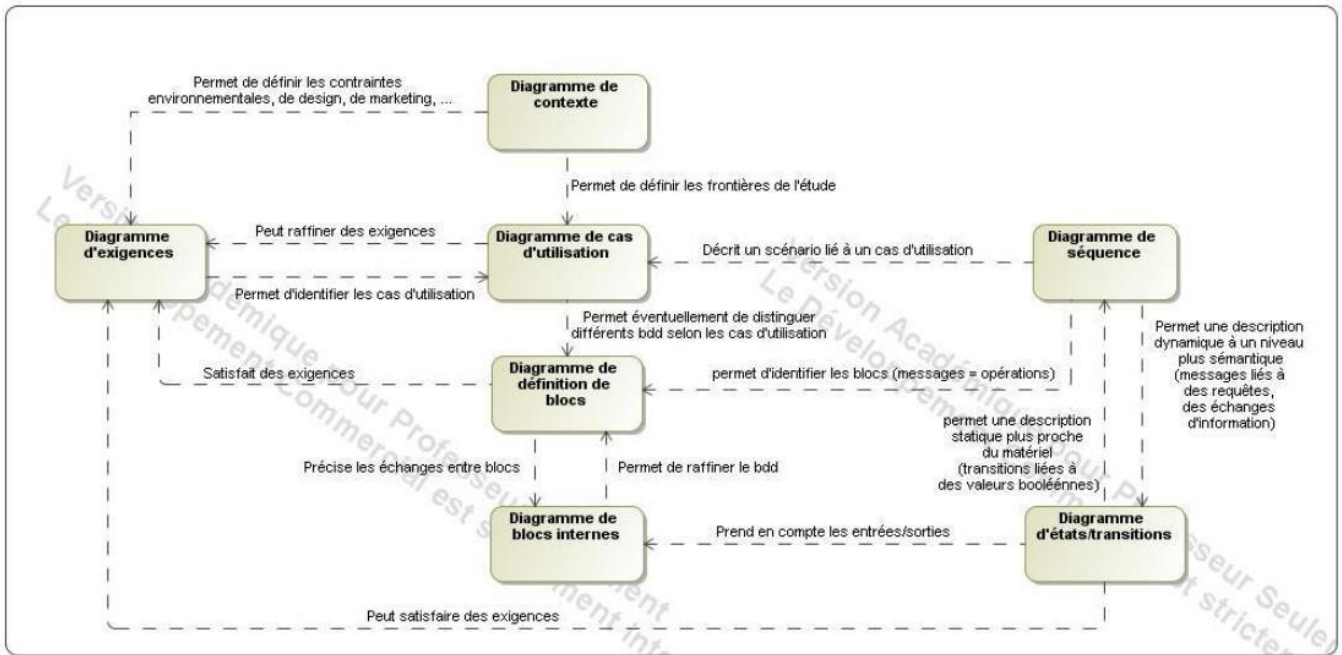


Les problèmes rencontrés avec un système étant de natures différentes, différents diagrammes sont nécessaires, dont l'ordre importe peu, puisqu'ils correspondent à des besoins différents. Certains diagrammes s'intéressent à la structure du système (diagrammes structurels) et d'autres à son fonctionnement (diagrammes comportementaux). Un autre diagramme différent s'occupe de décrire les performances attendues : le diagramme des exigences. Les cas d'utilisation définissent quant à eux à quoi sert le système.

Organisation du SysML par diagrammes : Le langage **SysML** s'articule autour de neuf types de diagrammes, seuls sept diagrammes seront étudiés dans ce cours, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système.



Le langage **SysML** s'articule autour de neuf types de diagrammes, seuls sept diagrammes seront étudiés dans ce cours, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système.



a) Diagramme transversal

Diagramme d'exigences (SysML Requirements Diagram).

□ Rôle

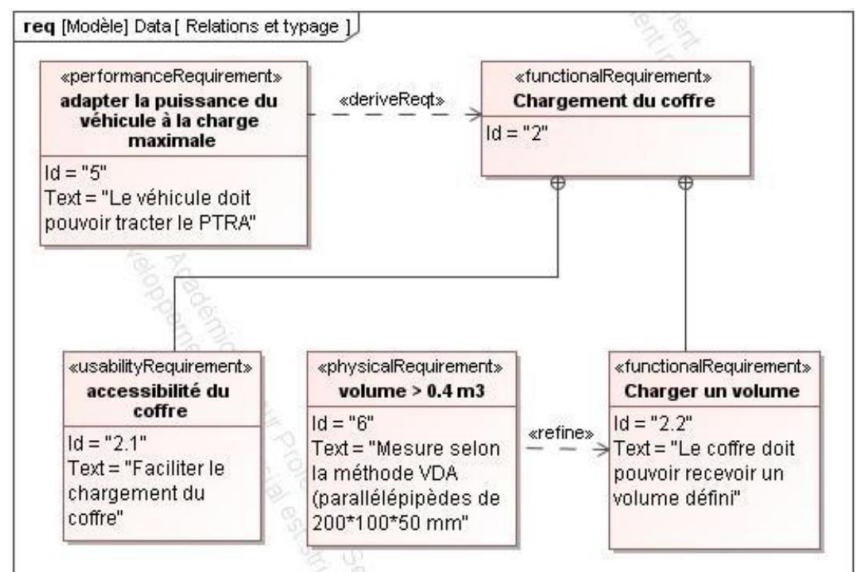
Représente toutes les exigences du système :

- exigences environnementales,
- exigences économiques,
- exigences fonctionnelles,
- exigences techniques,
- etc.

□ Limites et préconisation

Par souci de lisibilité, ne pas chercher à poser toutes les exigences.

Réaliser plusieurs diagrammes d'exigences si nécessaire. Regrouper les exigences techniques sur un seul diagramme par exemple, puis les autres groupes d'exigences sur d'autres diagrammes.



b) Diagrammes comportementaux

Diagramme de contexte

- Rôle :

Permettre de définir les frontières de l'étude, et en particulier de préciser la phase du cycle de vie dans laquelle on situe l'étude (généralement la phase d'utilisation).

Répondre à la question : "Quels sont les acteurs et éléments environnants du système ?".

- Limites et préconisation :

Ce diagramme devra bien sûr faire apparaître tous les acteurs intervenants dans le diagramme de cas d'utilisation, mais il fera aussi apparaître les différents acteurs ou éléments intervenant dans une exigence. Il n'y a aucune recommandation spécifique sur la manière dont il sera établi. Il pourra se faire par :

- une carte mentale,
- un bdd (diagramme de définitions de blocs) SysML,
- un ibd (diagramme de blocs internes) SysML.

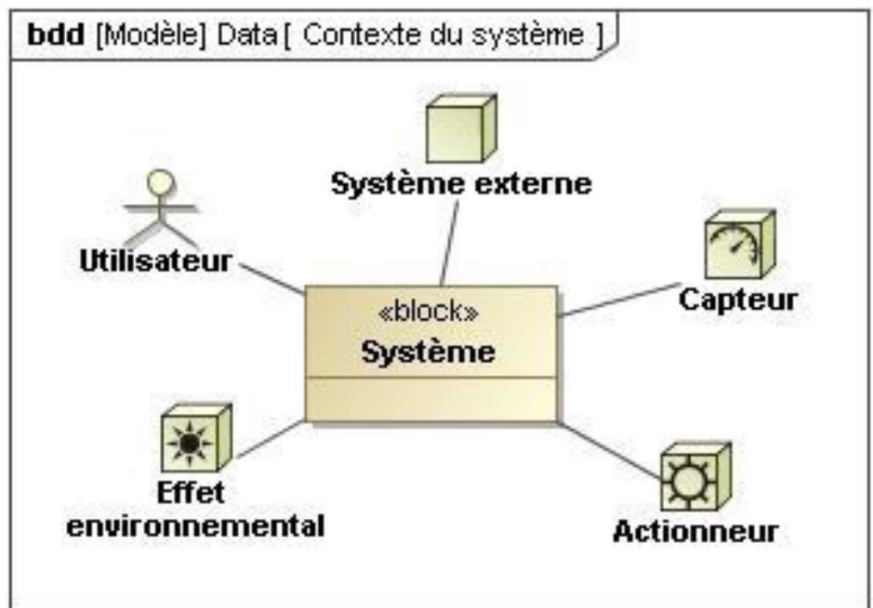


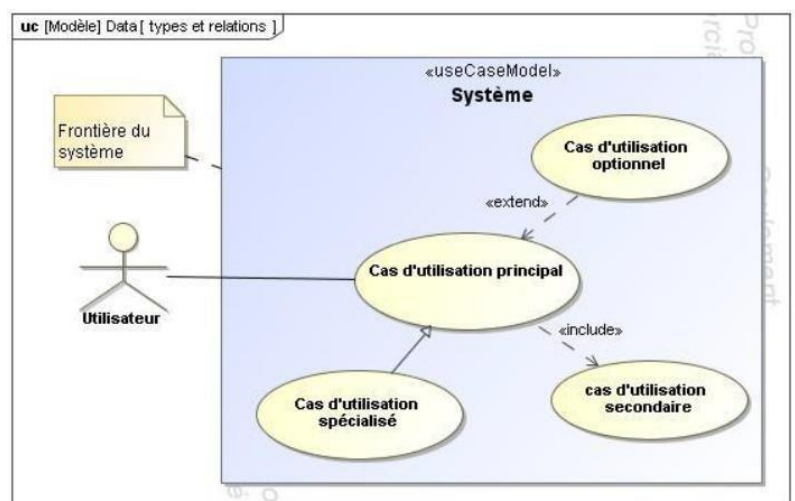
Diagramme de cas d'utilisation (SysML Use Case Diagram)

- Rôle :

Montrer les fonctionnalités offertes par le système. Répondre à la question : "quels services rend le système ?".

Fonctionnalité = cas d'utilisation = service rendu en autonomie d'un bout à l'autre par le système.

Le résultat est visible par l'acteur (entité extérieure en interaction avec le système).



- Limites et préconisation :

Ce diagramme devra bien sûr faire apparaître tous les acteurs intervenants dans le diagramme de cas d'utilisation, mais il fera aussi apparaître les différents acteurs ou éléments intervenant dans une exigence.

Diagramme de séquence (SysML Sequence Diagram)

□ Rôle :

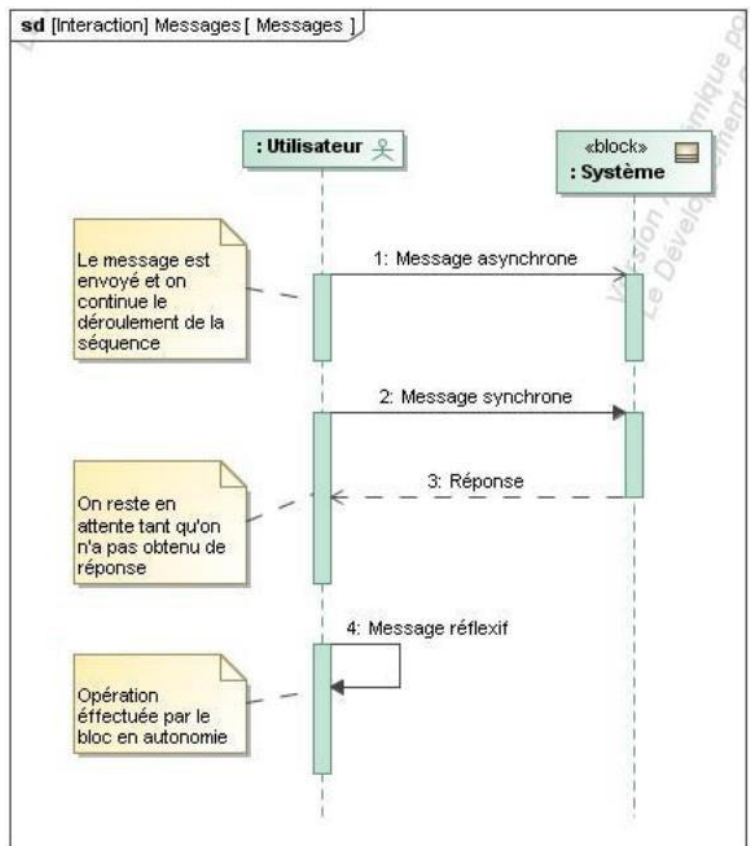
Décrire les scénarios correspondant aux cas d'utilisation, un cas d'utilisation est décrit par au moins un diagramme de séquence.

Répondre à la question : "Comment est réalisé ce cas d'utilisation ?".

Montrer les interactions entre différents éléments d'un point de vue séquentiel, enchaînement et nature des échanges.

Dans un premier temps il est préférable de faire un diagramme de séquence « système », celui-ci étant vu comme une boîte noire.

Pour des détails vous pouvez ensuite montrer les interactions au sein du système (décomposé en ses différents éléments).



□ Limites et préconisation :

Il existe les **fragments combinés** pour montrer des variantes dans un scénario.

Cela doit être utilisé avec parcimonie car les diagrammes de séquence ne sont pas des algorithmes.

Un scénario se décrit dans un cadre bien précis. Vous aurez donc des scénarios de réussite et des scénarios d'échecs (gestion des problèmes).

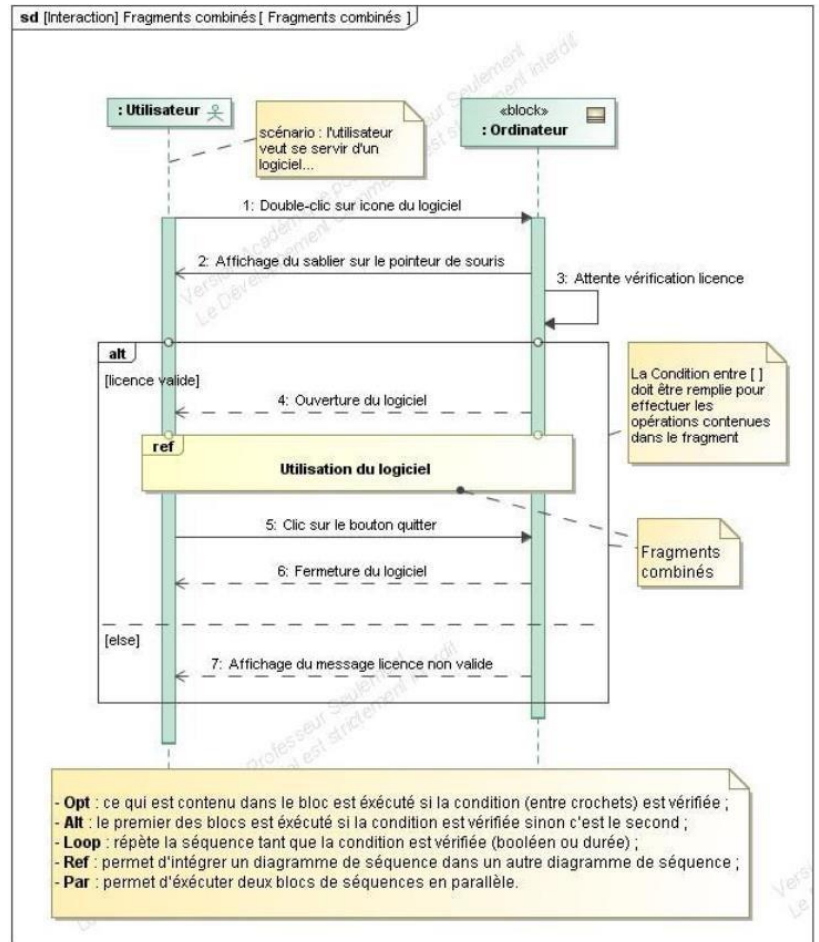


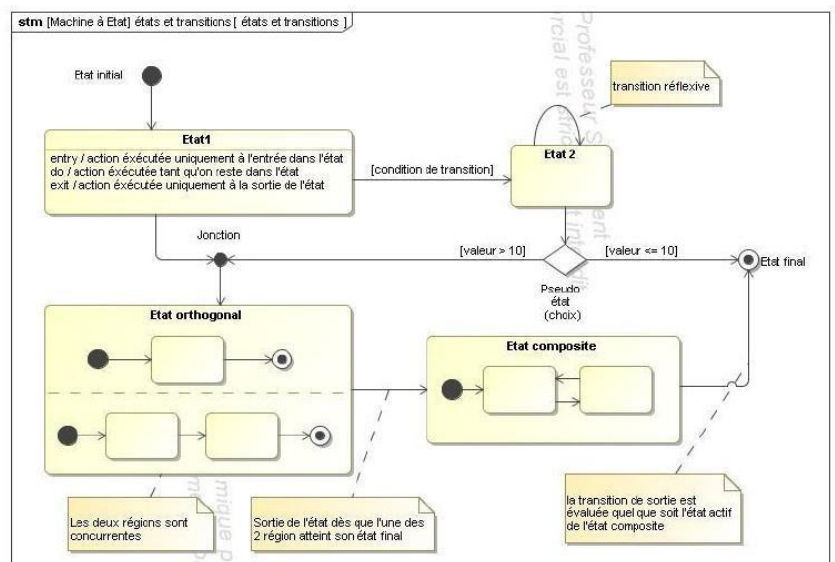
Diagramme d'états / transitions (SysML State Machine Diagram)

• Rôle :

Décrire le comportement d'un programme sous forme de machine d'états.

Montrer les différents états pris par le système (ou un sousystème) en fonction des interactions

Répondre à la question : "Comment représenter les différents états du système ?"



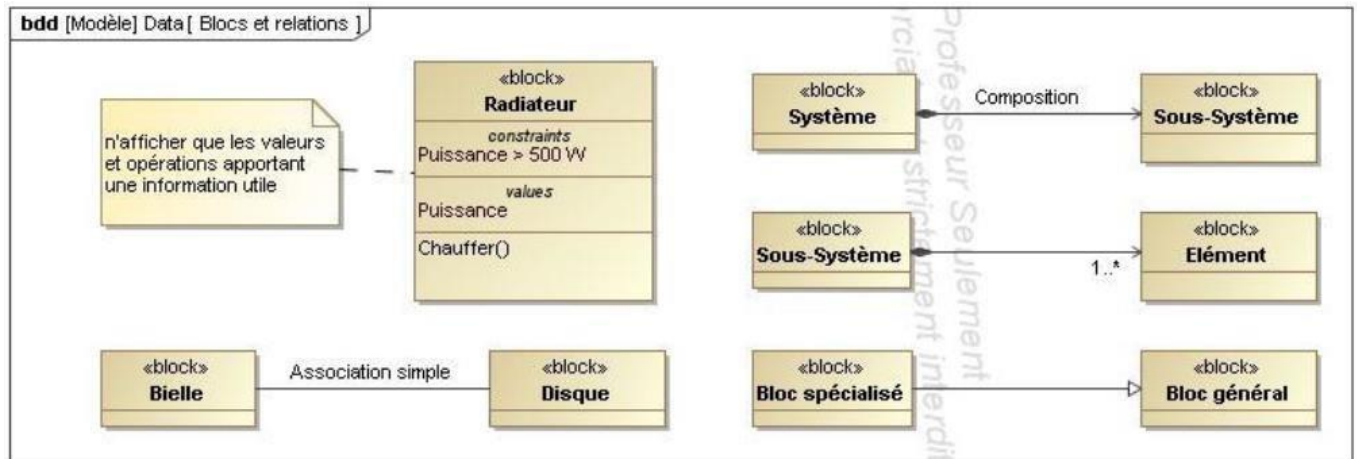
• Limites et préconisation :

Ce diagramme servira avant tout à décrire le fonctionnement d'un programme.

Ce diagramme trouve toute sa place en lien avec les logiciels de simulation comportementale (Matlab, LabView, Modelica,...) .

c) Diagrammes structurels

Diagramme de définition de blocs (SysML Block Definition Diagram)



- Rôle :

Montrer le système d'un point de vue composé/composant Répondre à la question "qui contient quoi?".

Montrer les caractéristiques principales de chaque bloc en faisant apparaître les opérations (rôles) et les propriétés (caractéristiques).

Permettre de représenter les liens entre les blocs de même niveau par une association (simple trait entre 2 blocs).

- Limites et préconisation :

La question du zoom est importante. Même si on peut descendre assez bas dans les détails, il ne sera pas pertinent en général de le faire. Ce diagramme est utile pour montrer les grosses briques du système.

Il n'est pas obligatoire de faire apparaître les propriétés et les opérations dans chaque bloc. Dans ce cas le diagramme est relativement pauvre en informations, mais il offre d'un coup d'oeil la structure du système.

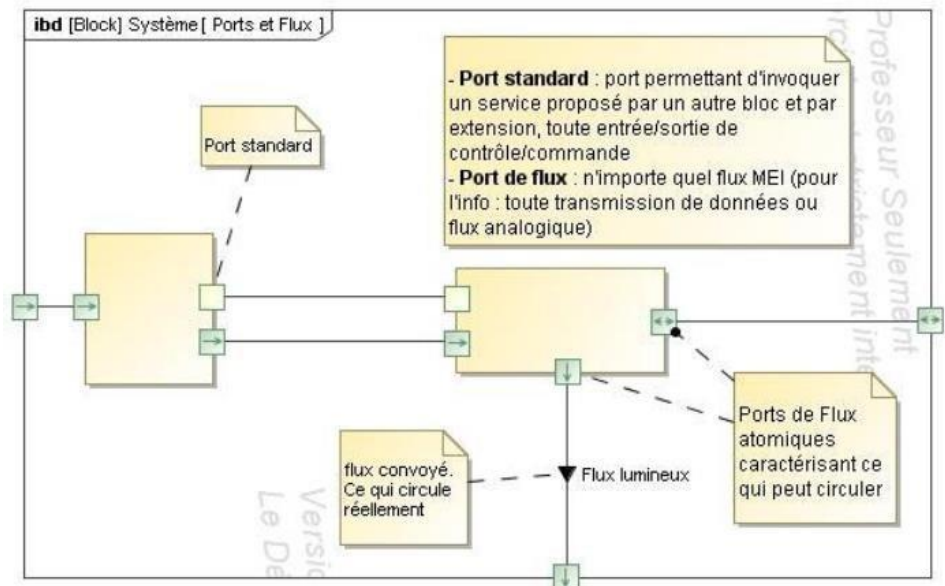
Diagramme de blocs internes (SysML Internal Block Diagram)

- Rôle :

Permettre de représenter les échanges de matière/information/énergie entre blocs de même niveau grâce aux ports de flux (petit carré avec une flèche).

Permettre de représenter les services invoqués par un autre bloc grâce aux ports standards (petit carré sans flèche), et par extension toute entrée/sortie de contrôle/commande.

Permettre de représenter les liens entre les blocs de même niveau.



- Limites et préconisation :

Il faut bien retenir que les liens se représentent entre blocs de même niveau, ils ne se contiennent pas.

Chaque bloc du BDD contenant d'autres blocs peut être représenté par un IBD.

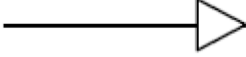
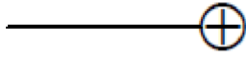
Attention à bien faire la différence entre port standard et port de flux.

Port standard : désigne une interface permettant d'invoquer un service / une opération

Port de flux : canal d'Entrée / Sortie par lequel transite de la matière, de l'énergie ou de l'information (MEI).

Définitions des relations dans les diagrammes.

A	B	
		Association : relation d'égal à égal entre 2 éléments. A utilise B - 3 diagrammes : cas d'utilisation, définition des blocs, blocs internes.
		Dépendance : l'un des deux éléments dépend de l'autre. A dépend de B - 3 diagrammes : cas d'utilisation, exigences, définition des blocs.
		Agrégation : Un élément est une composante facultative d'un autre. A entre dans la composition de B sans être indispensable à son fonctionnement. - 2 diagrammes : exigences, définition des blocs.
		Composition : Un élément est une composante obligatoire de l'autre. A entre dans la composition de B et lui est indispensable à son fonctionnement. - 2 diagrammes : exigences, définition des blocs.

	Généralisation : Dépendance de type 'filiation' entre 2 éléments. A est une sorte de B. - 3 diagrammes : exigences, définition des blocs, blocs internes.
	Conteneur : relation d'inclusion entre 2 éléments. B contient A - 2 diagrammes : exigences, définition des blocs.

4) Exemple 1 : drone AR Drone PARROT



a) Diagrammes fonctionnels :

Diagramme des cas d'utilisation : *Use case diagram* (uc)

Ce diagramme permet de définir **la ou les fonctions** du système étudié. Il peut être réduit à sa plus simple expression ou détaillé pour faire apparaître des fonctions secondaires.

Cas d'utilisation minimal. On voit ici défini un ACTEUR (Pilote), un cas d'utilisation (qui ne spécifie pas le but de la navigation : jouer, filmer...) et une relation liant l'acteur au système.



La question pourrait se poser de savoir si la station sol est un acteur car on achète l'AR Drone seul. Mais on achète aussi le logiciel et sans Iphone ou tablette, l'AR Drone ne peut pas fonctionner. Il est donc réaliste d'inclure la station sol dans le système. On peut alors détailler le cas d'utilisation ci-contre.

Cas d'utilisation un peu détaillé : des fonctions supplémentaires apparaissent.

Paramétrer le vol est « inclus » (**include**) dans contrôler l'AR Drone, donc obligatoire, alors que la diffusion des données de navigation et des flux vidéo est une « extension possible »

Dans ce diagramme apparaissent :

- Le système délimité par sa frontière
- Les acteurs (au sens large) : L'utilisateur d'une part et les différents éléments auxquels le système est en relation directe (environnement...)
- Les fonctions globales réalisées par le système
- Les relations logiques (« stéréotypes ») d'inclusion « include » ou « extend » d'extension possible.

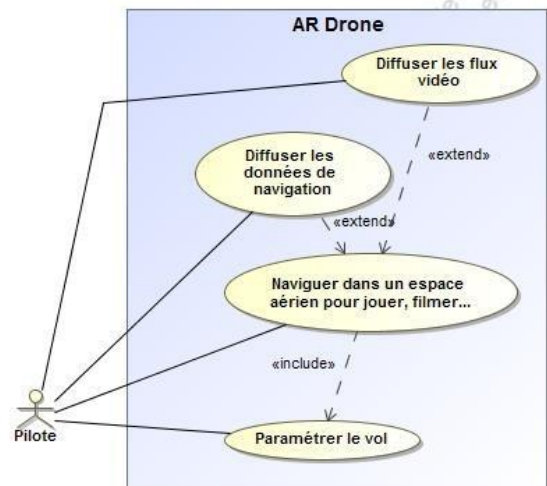
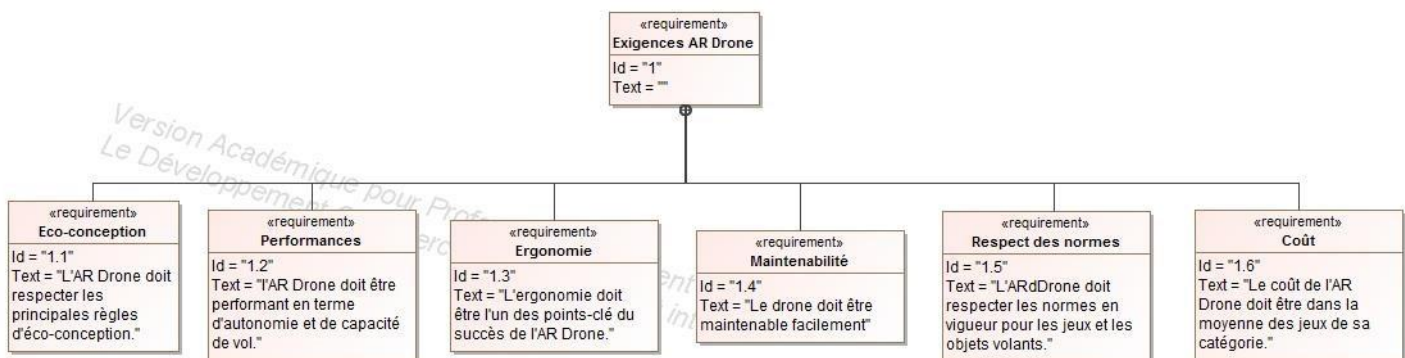


Diagramme d'exigences **Requirements diagram** (req)

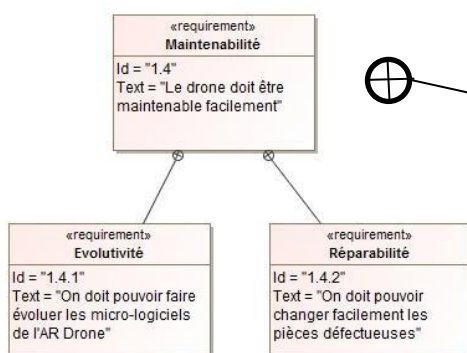


Il sert à :

- spécifier les exigences attendues vis-à-vis du système
- hiérarchiser les exigences (fonctionnelles, techniques, économiques, environnementales...)
- documenter et amener des précisions sur le contrôle de ces exigences.

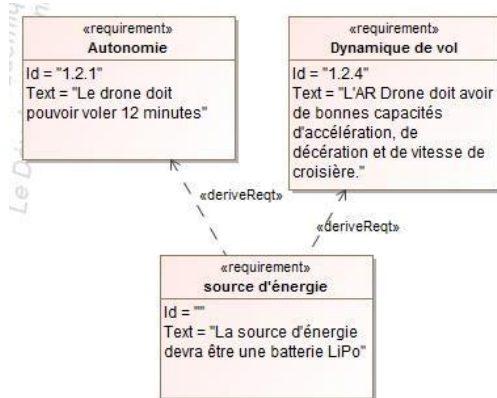
Une exigence possède un nom et un identifiant correct unique afin de pouvoir être validée et vérifiée.

➤ Contenance



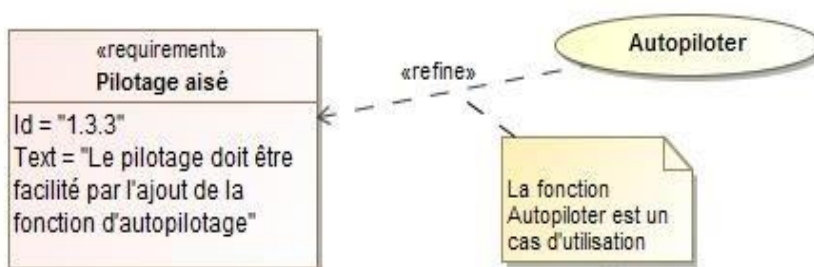
Une exigence complexe peut être décomposée par un lien de contenance

➤ Relation d'exigence dérivée : **Derive Requirement** « deriveReq »



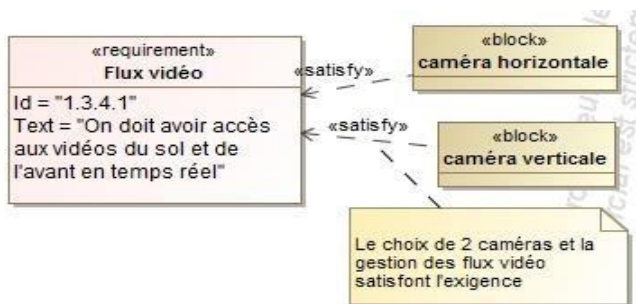
Une exigence technique (source d'énergie) peut découler d'une ou plusieurs exigences fonctionnelles (ou autres). Par exemple l'exigence d'avoir recours à une batterie Lithium Polymère vient de l'expérience « métier », et ce choix découle des exigences de performances d'autonomie et de dynamique de vol. On dit que le choix de la source d'énergie est une exigence dérivée des autres exigences.

➤ Relation de raffinement « refine »



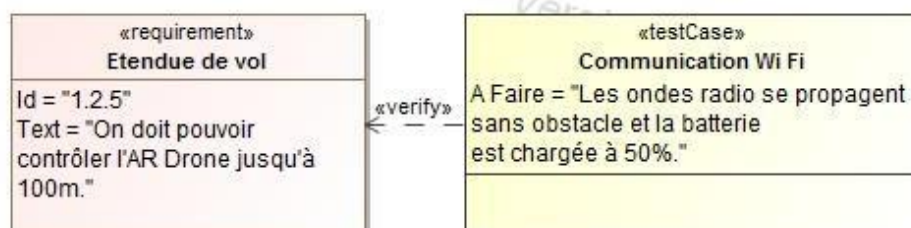
La relation de raffinement peut être utile pour décrire comment un ou plusieurs éléments de la modélisation peuvent préciser une exigence (un cas d'utilisation, un diagramme de séquence, etc...).

➤ Relation « satisfy »



Cette fois c'est un élément de structure (ou d'architecture), contenu dans un bloc – voir bdd, qui permet de satisfaire l'exigence.

➤ Relation « verify » : spécifie comment une exigence est contrôlée ou calculée par un modèle.



A Retenir : Le diagramme des exigences organise et décompose les impératifs du cahier des charges en créant des liens de dépendance visibles dans la numérotation. Plusieurs catégories peuvent être créées (fonctionnelles, économiques, techniques...)

b) Diagrammes structurels

Le bloc ou « *block* » en anglais est l'élément de base de la décomposition architecturale du système.

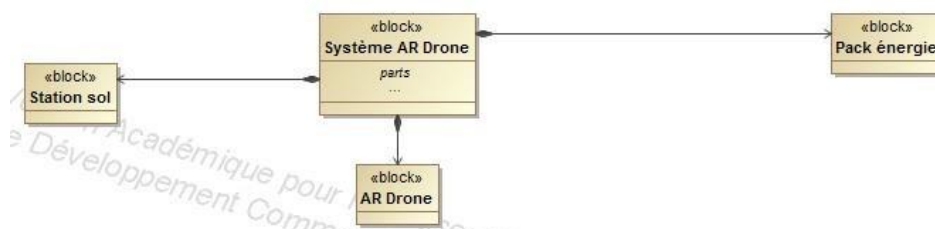
Un bloc peut être de nature logicielle (base de donnée, traitement...) ou matérielle (ensemble, composant...), ce qui est plutôt notre utilisation en CPGE.

3-1- Diagramme de définition de blocs : **Block definition diagram** (bdd)

Ce diagramme permet de définir la décomposition matérielle (et logicielle) du système en établissant une relation composant – composé en répondant à la question qui contient quoi ?

Dans les blocs peuvent être définis les rôles (**operations**) et les caractéristiques principales.

A un premier niveau, on pourrait trouver seulement :



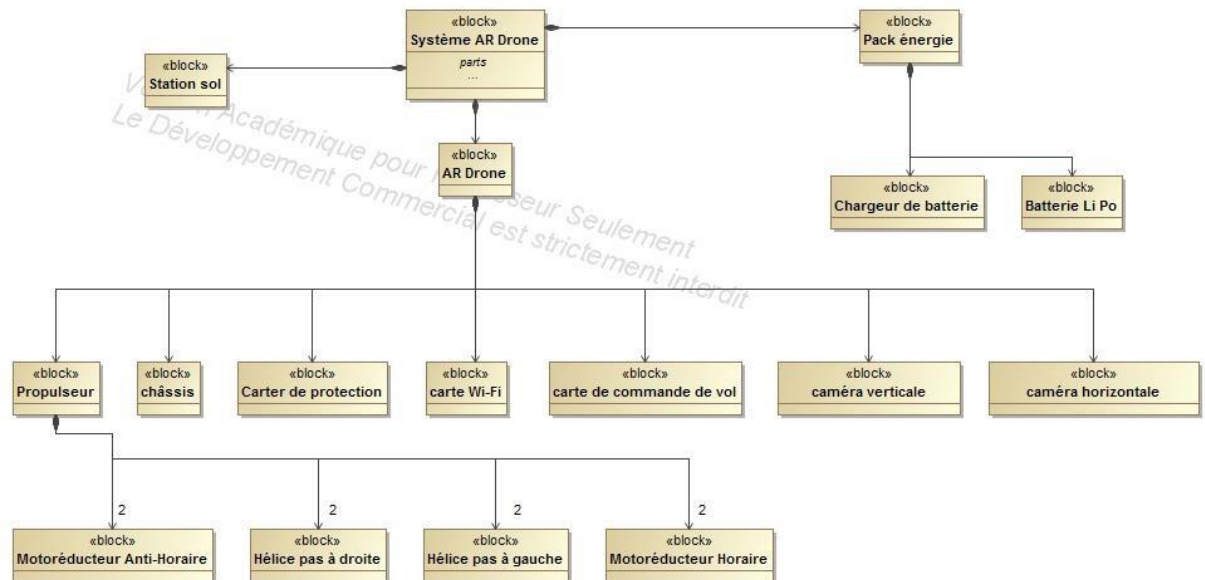
Un bloc comporte plusieurs zones graphiques distinctes après le titre, servant à définir des *paramètres* :

- Les éléments qui composent le bloc (**parts**)
- Des valeurs importantes à connaître (**values**)
- La définition de ports de communication (**flow ports, standard ports...**)

Il est très utile de ne détailler que ce qui est nécessaire et de faire des « zoom » si besoin, sans forcément tout représenter. L'architecture complète de l'AR Drone ci-après :

Les liens qui apparaissent s'appellent des liens de *composition* : **composition**.

Il existe aussi d'autres types de liens : *spécialisation* (ou **generalisation**), donnant un choix entre deux options : caméra câblée ou caméra sans fil, **agregation**, ...



Multiplicité : On remarque les chiffres 2 au-dessus de certains blocs. Cette valeur, qui vaut 1 par défaut désigne le nombre d'*instances* liées au bloc considéré. Par exemple, il y a 2 motoréducteurs à rotation anti-horaire. Ces 2 moteurs apparaîtront séparément dans le diagramme de blocs interne.

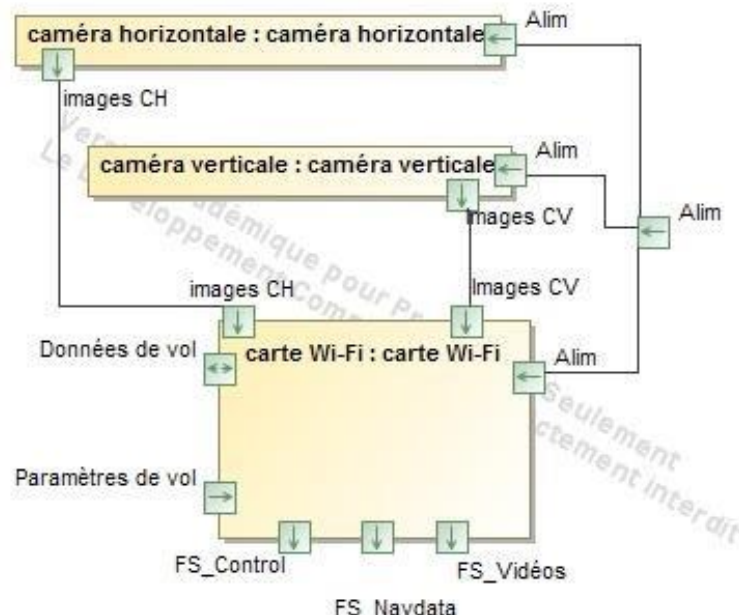
Un bloc représente une définition générale alors que l'instance du bloc représente l'objet réel. On dira qu'une instance de propulseur comporte 2 instances de moteurs à rotation anti-horaire, 2 hélices pas à droite, etc...

A retenir : Le bbd est une décomposition hiérarchique d'un système en sous-systèmes, jusqu'au composant et/ou programme informatique.

3-2- Diagramme de blocs internes : **Internal block diagram** (ibd)

L'ibd est associé à un bloc du bbd et permet de décrire les échanges entre les blocs grâce à des **ports** de communication :

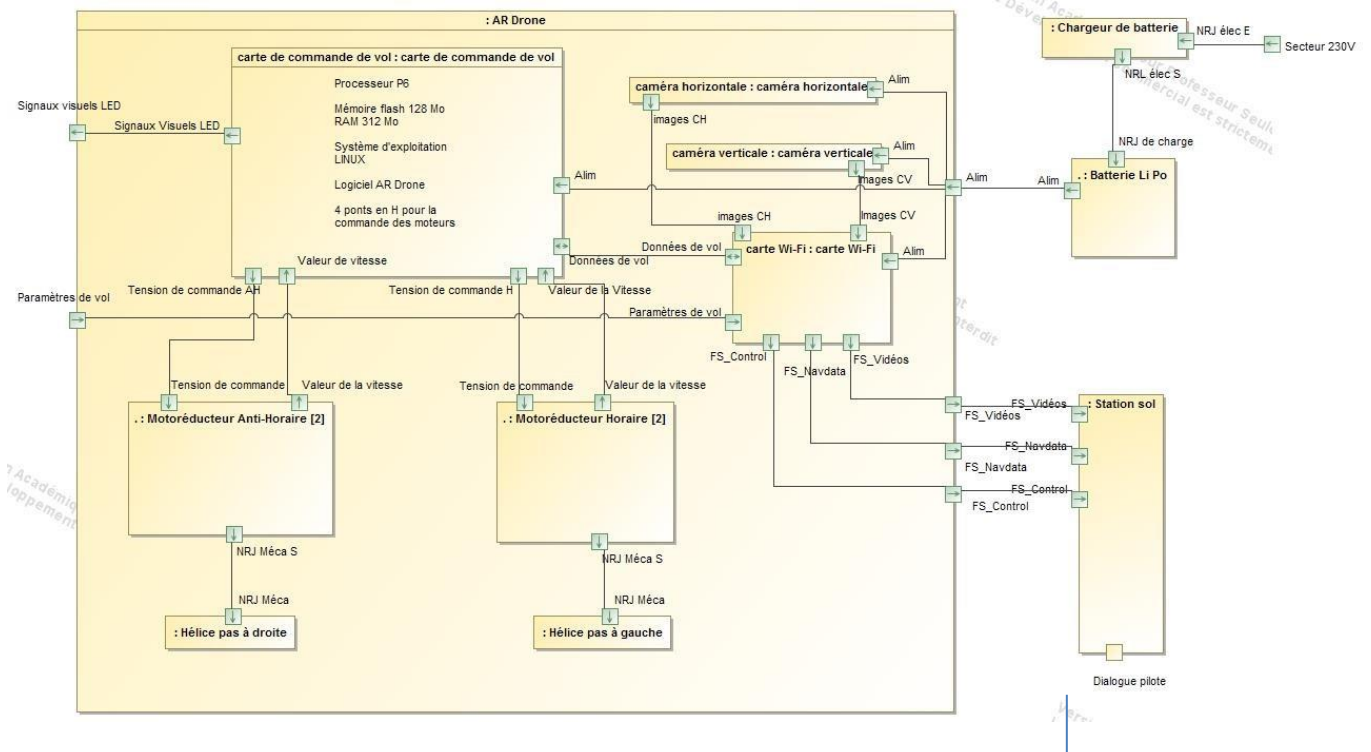
- Energie
- Information
- Matière



Par exemple ici ibd partiel de l'AR Drone qui montre les échanges entre la carte Wi-Fi et les caméras

2 types de ports :

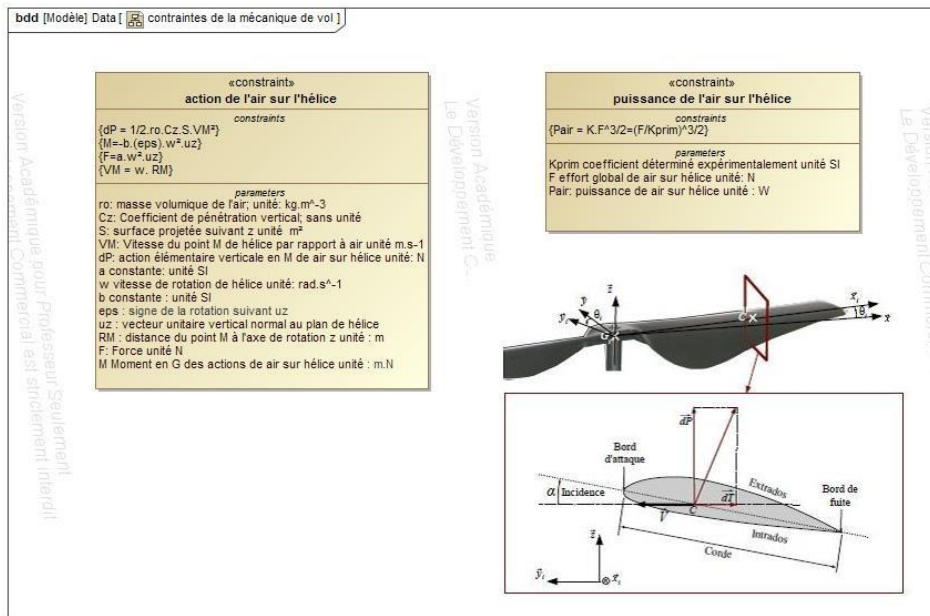
- **Flow port** : qui permet la communication de flux orienté (de données, d'énergie, de matière)
- **Standard port** : qui sert à faire circuler les ordres de commande (Marche/Arrêt...)



A retenir : L'ibd permet de détailler un block en faisant apparaître ses composants et les échanges d'énergie, information, matières et ordres de commandes extérieurs.

c) Diagramme paramétrique *Parametric diagram* (par)

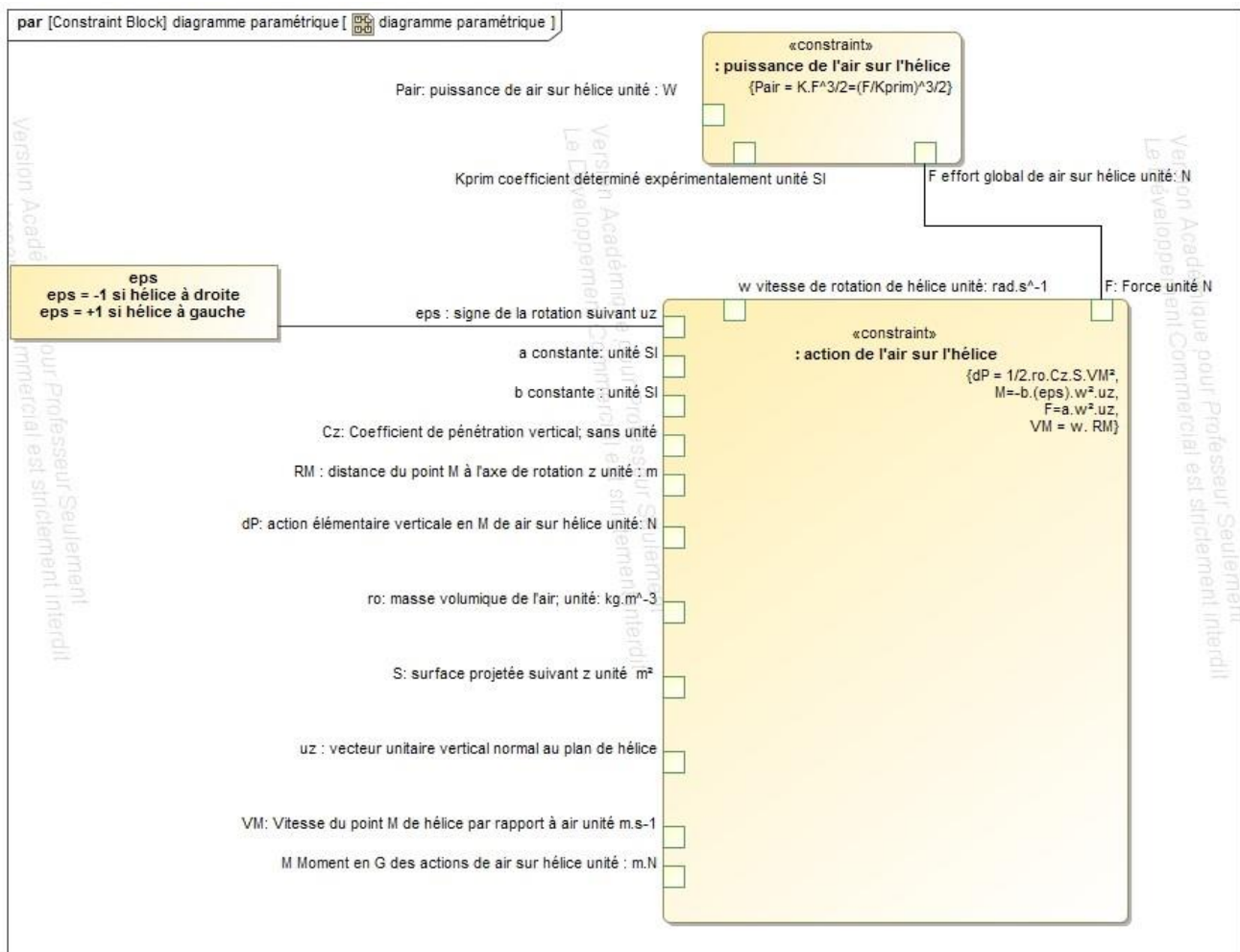
Le diagramme paramétrique est un diagramme de blocs particulier de SysML. Grâce à une zone de définition des contraintes (**constraints**), on peut définir la ou les lois de comportement physique de l'élément désigné dans le bloc. On lit ainsi par exemple le rapport de réduction, la traduction de la loi de roulement sans glissement, les relations entre effort et vitesse.... Les grandeurs qui apparaissent sont aussi définies, ainsi que leur unité (**parameters**). Ces formules peuvent servir à simuler le comportement par logiciel.



Exemple : AR
Drone contraintes de
la mécanique de vol
sur le propulseur

Ce qui donne le
diagramme
paramétrique suivant
avec les paramètres

représentés par des ports.

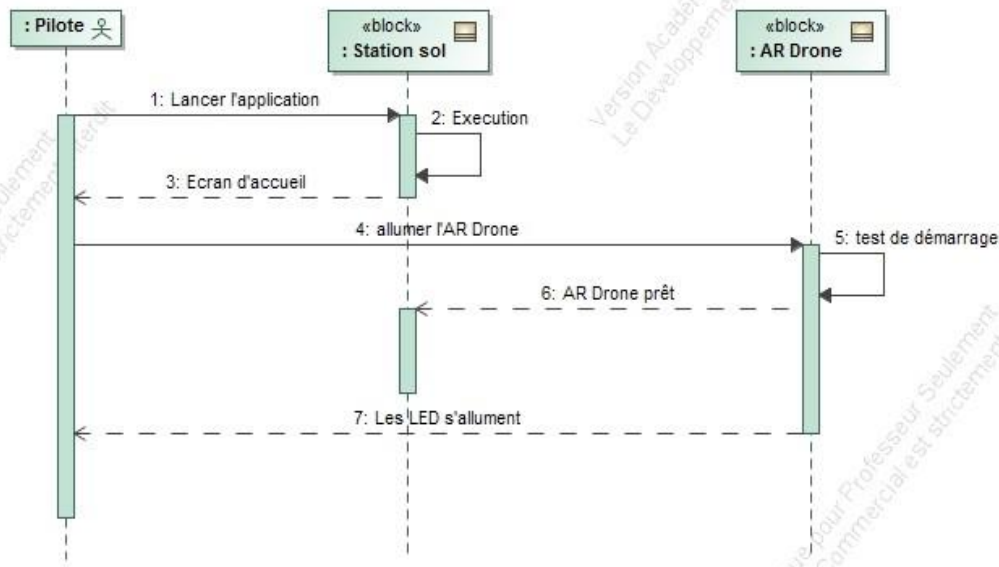


d) Diagrammes comportementaux :

Ces diagrammes servent à décrire comment fonctionne le système suivant différents points de vue externes ou internes avec plus ou moins de détails

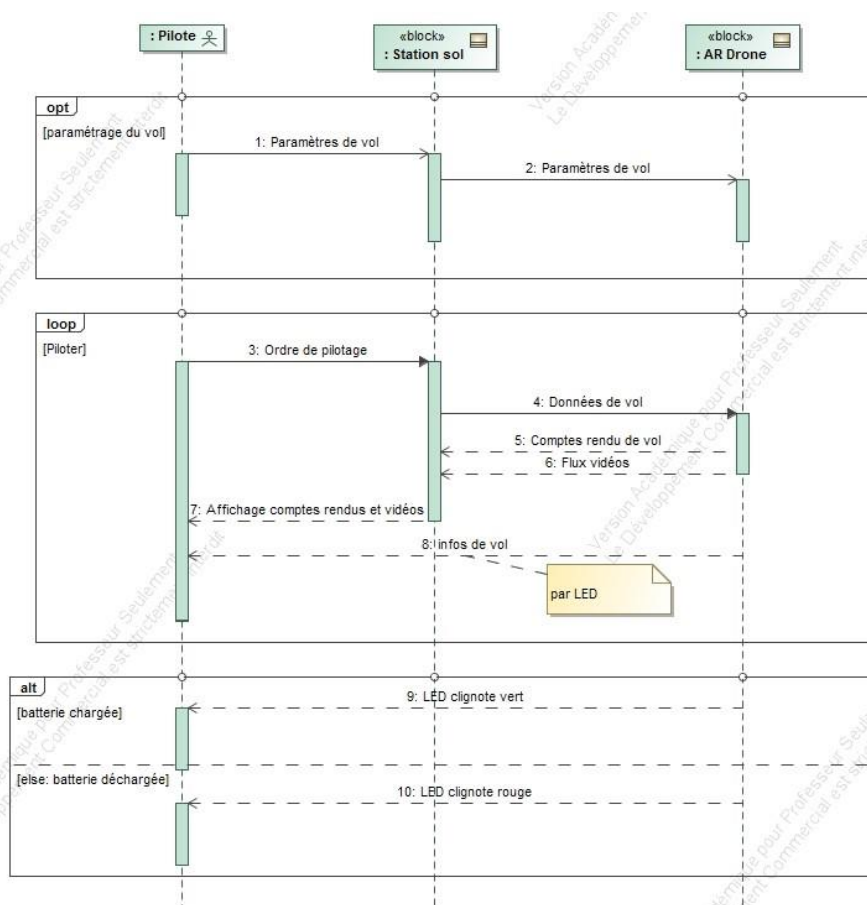
Diagramme de séquence : **Sequence diagram** (seq)

Ce diagramme décrit les échanges entre le (ou les) acteur(s) et le système ou entre blocs. Ces échanges sont des messages (en horizontal) entraînant des actions d'une certaine durée représentée verticalement. Les lignes pointillées verticales se nomment « lignes de vie ».



Certains messages attendent une réponse, ils sont appelés **synchrones** (ceux de l'exemple ci-contre), tandis que d'autres, **asynchrones** n'attendent pas de réponse de la part du destinataire. La forme des flèches diffère.

Une tâche interne à une entité correspond à un message **réflexif**.



De plus, les séquences d'échange peuvent comporter des pavés pour signifier qu'il y a :

- des options possibles (**opt**),
- une séquence qui se répète en boucle (**loop**)
- une séquence alternative (**alt**), c'est-à-dire qu'elle se déroule en fonction d'une condition.

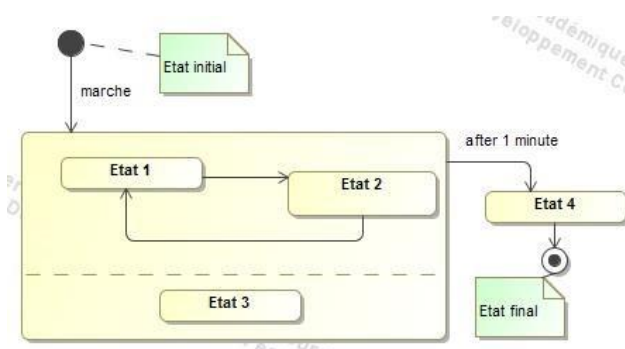
A retenir : Le diagramme de séquence donne la description des interactions entre les différentes entités du

système au cours du temps.

Diagramme d'états transition : **State machine diagram** (stm)

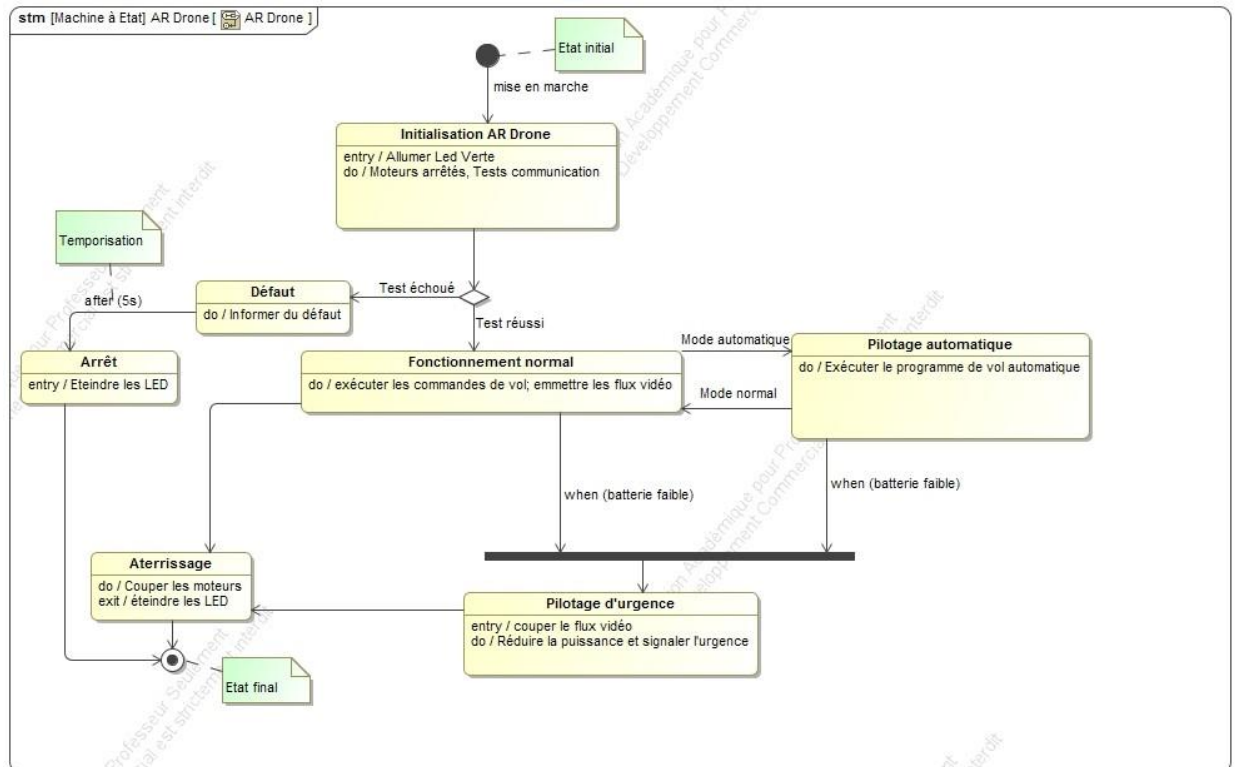
Le comportement du système, d'un sous-système ou d'un bloc est schématisé par des **états** et des **transitions** entre les états. Ce diagramme montre l'évolution des états du système en fonction des événements. A un état correspondent des actions, une activité ou une attente. Le symbole graphique est de type Block et l'état est décrit clairement. Une action (impulsionnelle) peut se faire à l'activation de l'état (entry)(ex : allumer voyant) ou en quittant l'état (exit) (ex : éteindre voyant), et l'activité (faire...) pendant que l'état est actif (do) (ex : tourner la grue à vitesse lente).

Les transitions entre les états sont de type **événement** (possibilité de front) ou une **condition de garde** (équation booléenne entre variables binaires). On peut aussi avoir un **changement** de valeur (when) ou l'attente d'une réponse à un appel donné dans l'état (call event). Par défaut, en l'absence de transition, le changement d'état se fait à la fin de l'activité.



Un **état initial** et un **état final** uniques sont nécessaires. Un seul état peut être actif à la fois, mais un **état composite à régions orthogonales** permet de faire évoluer deux ou plusieurs sousétats séparément. Un **état composite** représente un ensemble d'états qui sont décrits séparément, ce qui permet d'établir une certaine hiérarchie.

Exemple :



A Retenir : Une machine d'état décrit le fonctionnement séquentiel du système.

Ce diagramme sera étudié plus en détail dans le chapitre des systèmes à événements discrets à la fin de l'année.

Diagramme d'activités : **Activity Diagramm** (act)

La forme de ce diagramme peut être celle d'un algorithme qui représente l'enchaînement des tâches que le système réalise (activités). De plus chaque tâche peut être impartie à l'élément du système qui la réalise.

A priori hors programme, ce diagramme ne posera pas de problème de lecture si toutefois il était donné.

5) Compléments d'associations et relations entre blocks

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.

DeriveReqt : 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.

6) Exemple 2 : Le sécateur électrique PELLENC

Pour appréhender le langage SysML, nous nous appuierons sur un exemple suivant : **le sécateur électrique Lixion évolution (société PELLENC)**.

a) Analyser le besoin

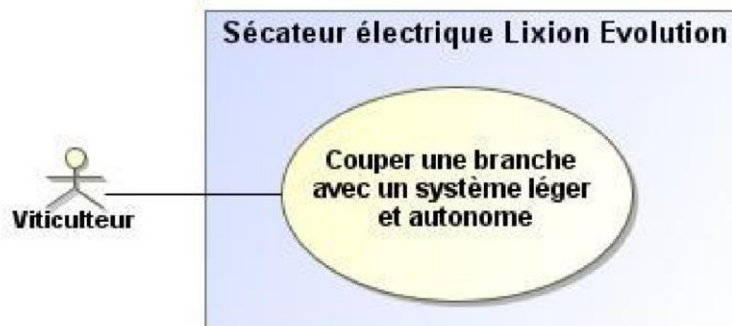
La période de la taille de la vigne dure environ 2 mois. Les viticulteurs coupent 8 à 10 heures par jour.



Pour réduire la fatigue de la main et du bras, la société PELLENC commercialise un sécateur électrique à commande électronique. Ce système se compose d'une valise contenant la partie commande PC (portée sur le dos de l'utilisateur) et alimentant un sécateur (tenu par la main de l'utilisateur) par un câble.

Diagramme de cas d'utilisation :

uc [Paquet] Diagrammes comportementaux [Diagramme de cas d'utilisation]



Cahier des Charges Fonctionnel (C.d.C.F.).

Il est extrait du tableau des exigences (SysML Requirements Table).

#	ID	Name	Text
1	3	Coût	< 1 400 € HT
2	1.2.1.1	Taille de la branche	Diamètre de La branche à tailler < 35 mm
3	1.2.3	Cadence de coupe	82 coupes par minute
4	1.2.2.1	Durée de coupe	0,5 s maximum
5	2.1.1	Temps d'utilisation	Autonomie jusqu'à 3 jours suivant l'utilisation
6	1.4.3.2	Diamètre	diamètre du manche < 40 mm
7	2.2.1.1	Masse maximale de la source énergie	2,5 kg
8	1.4.4.1	Masse maximale du sécateur seul	800 gr
9	5	Normes	le sécateur doit être conforme à l'article R233-100 du Code du Travail
10	11.1	Température extérieure	Le sécateur doit résister à des températures extérieures comprises entre -5 et 50 °
11	11.4.1	Humidité, poussière	Le système doit être étanche

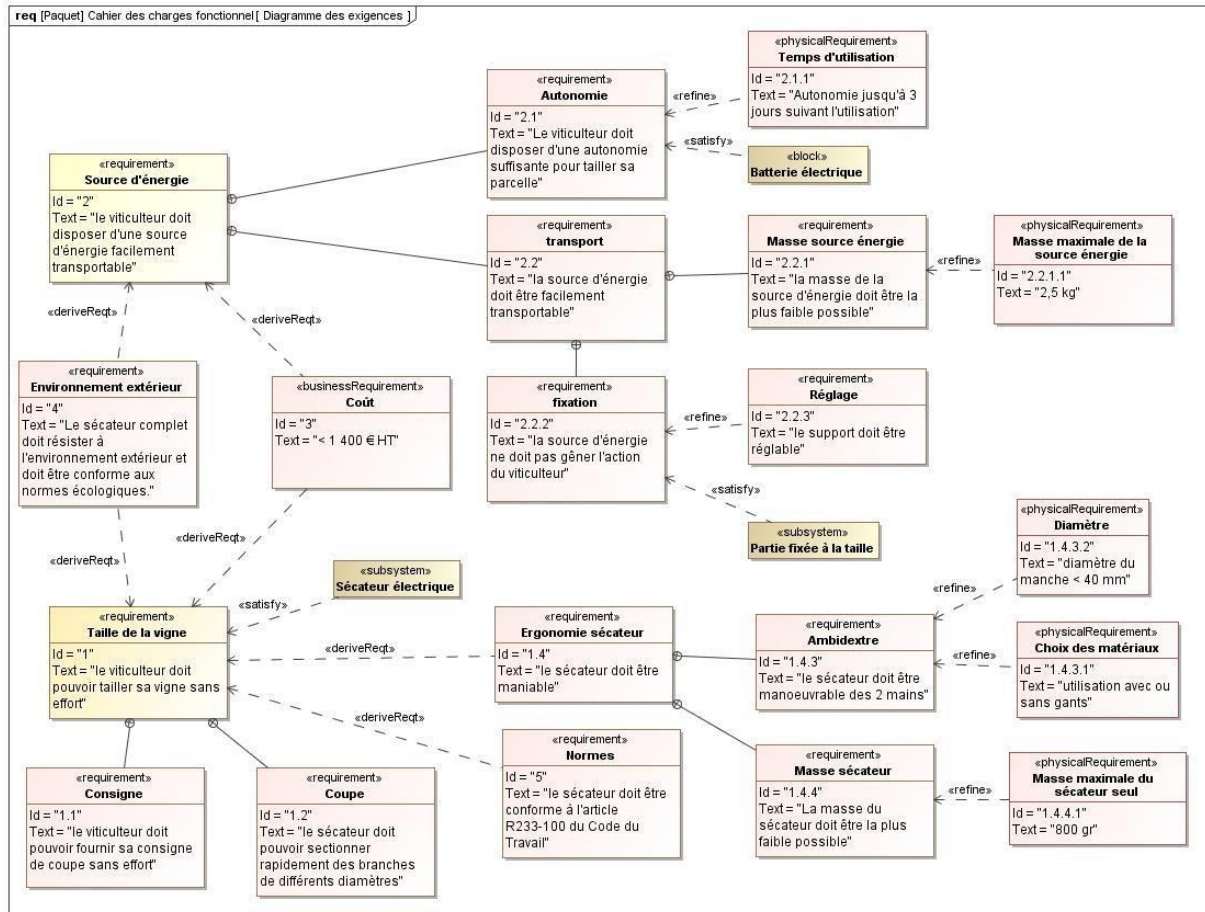
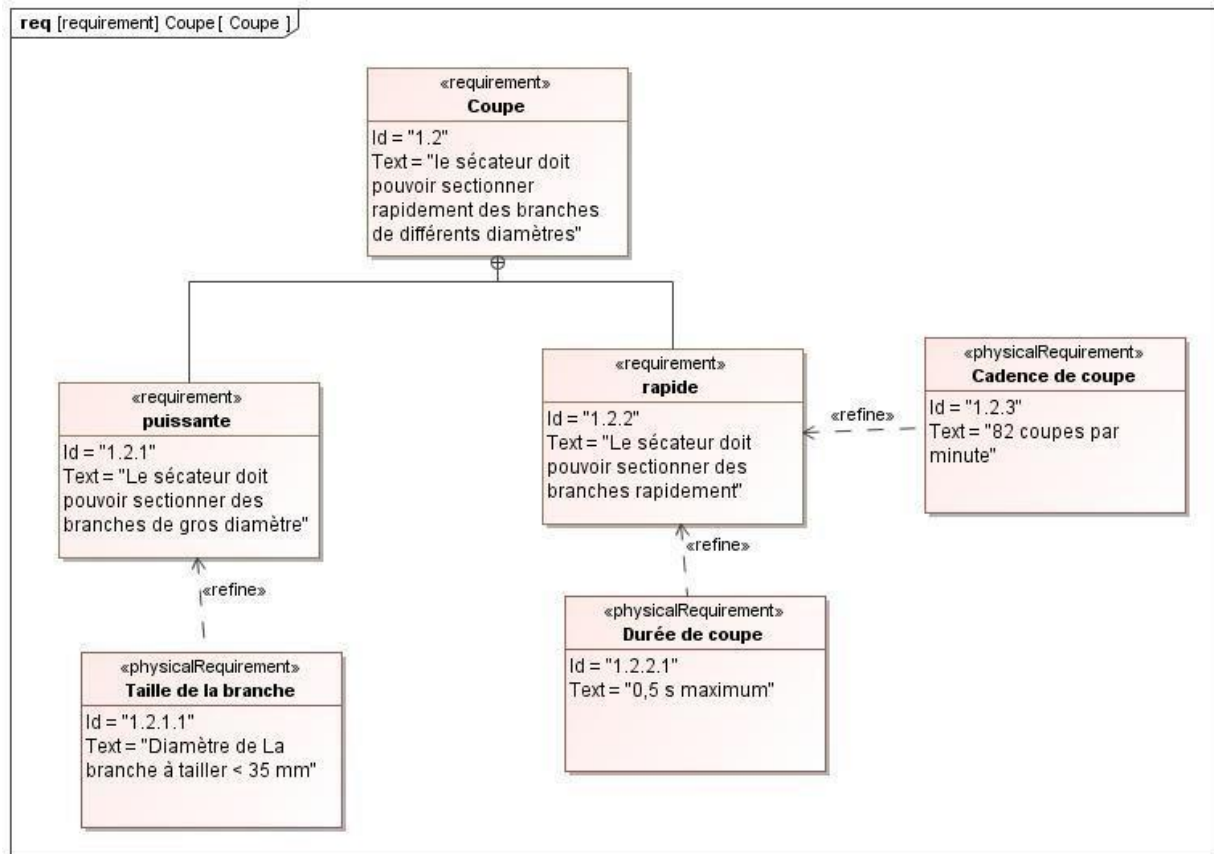


Diagramme d'exigences (coupe de la branche)



b) Concevoir le sécateur

Diagramme de définition de blocs du sécateur complet

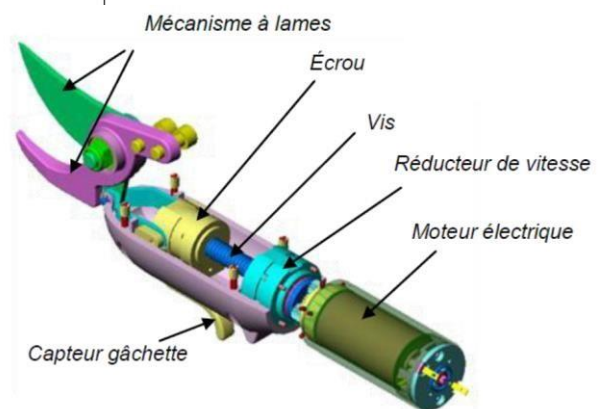
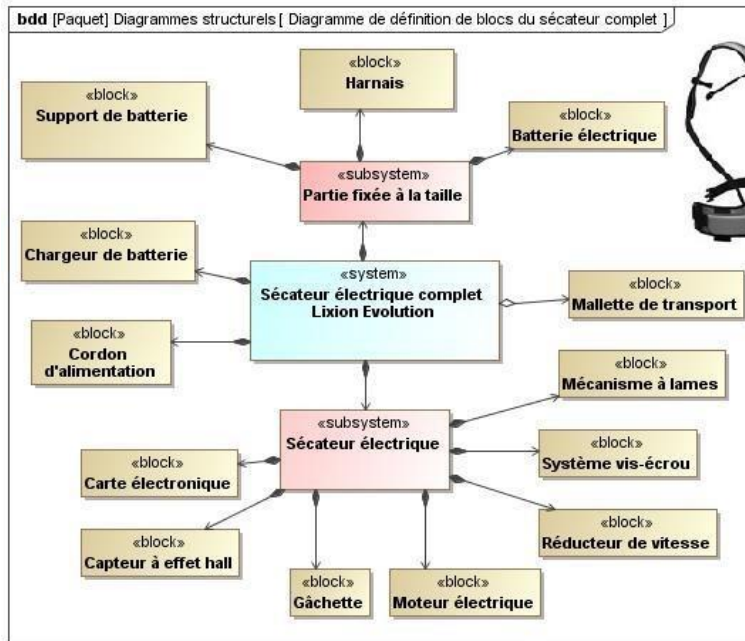


Diagramme de blocs internes du sécateur seul

