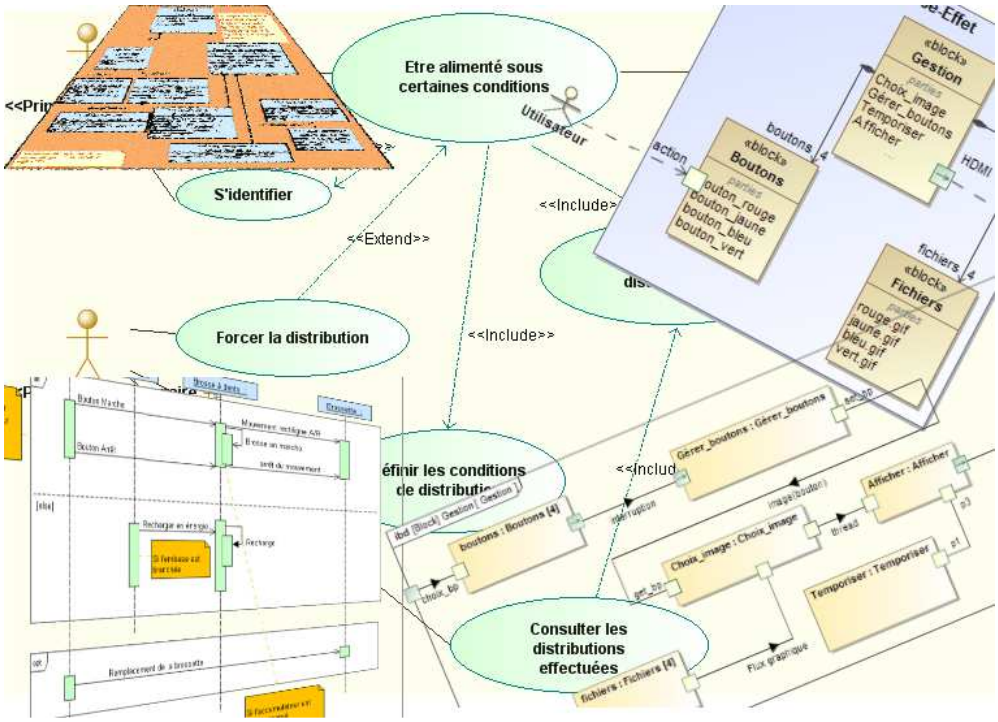
MODELISATION SYSML



Baccalauréat STI2D

- 2.2.2 : Représentation symbolique

Objectifs

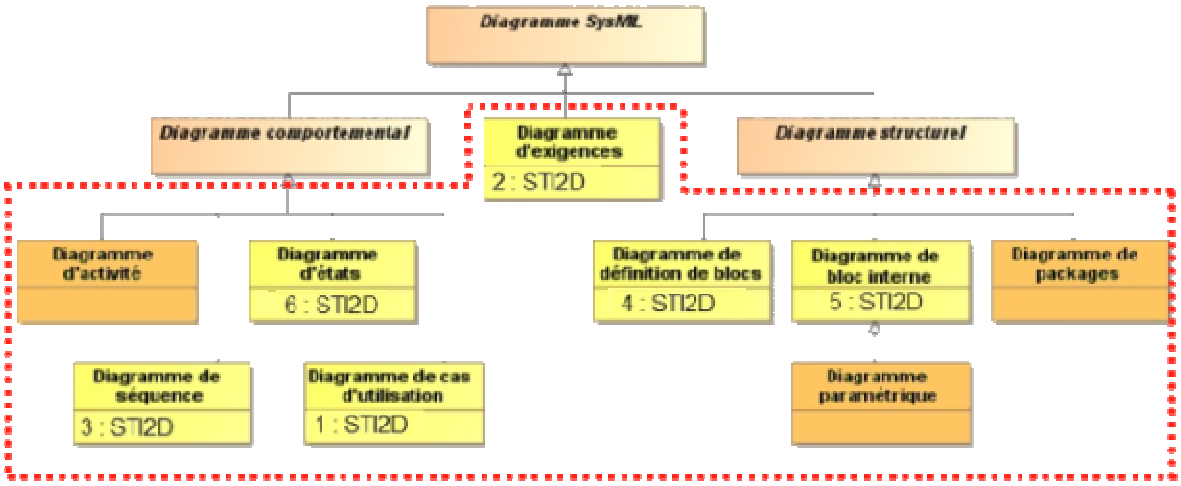
A la fin de la séquence, l'élève doit être capable

- d'interpréter et de compléter partiellement un diagramme d'exigence SysML à partir de la description d'un système ou du cahier des charges de celui-ci
- d'interpréter et de compléter partiellement un diagramme de cas d'utilisation
- d'interpréter et de concevoir un diagramme de séquence simple
- d'interpréter et de compléter partiellement un diagramme de bloc SysML à partir de la description d'un système ou du cahier des charges de celui-ci
- d'interpréter et de compléter partiellement un diagramme d'état

La modélisation SysML

"La transformation d'un besoin émergeant en la définition d'un système lui apportant une
solution met en oeuvre de multiples activités intellectuelles faisant passer progressivement
de concepts abstraits à la définition rigoureuse de produits. Il est nécessaire de s'appuyer
sur des représentations tant du problème que de ses solutions possibles à différents niveaux d'abstraction pour appréhender, conceptualiser, concevoir, estimer, simuler, valider, justifier des choix, communiquer. C'est le rôle de la modélisation !"

sources : SysML par l'exemple de Pascal Roques



SysML est un outil de modélisation dérivé d'UML qui, lui-même, est un outil dédié au développement informatique.

SysML (*System Modeling Language*) s'articule autour de 9 diagrammes dédiés à la représentation de concepts particuliers d'un système.

Ces 9 diagrammes décrivent :

- les exigences du système dérivés du cahier des charges
  - diagramme d'exigences
  - diagramme de cas
  - diagramme de séquence
- l'architecture du système
  - diagramme de package
  - diagramme de bloc
  - diagramme de bloc interne
- le cycle de vie des éléments et le fonctionnement
  - diagramme d'état
  - diagramme d'activité
- les liens entre les éléments de modélisation
  - diagramme paramétrique

Remarque : La formation STI2D limite la connaissance de la modélisation SysML à 6 diagrammes.

Rappels : Le cahier des charges

Le cahier des charges est un document essentiel à la réalisation d'un projet.
Il décrit précisément les besoins auxquels les intervenants doivent répondre : objectifs, cibles, concurrence, spécifications techniques, délais...

La norme NFX 50-150 formalise la rédaction d'un cahier des charges.

Les diagrammes SysML

Ce cours se limitera aux 6 diagrammes préconisés par le référentiel de STI2D.
Chaque type de diagramme sera associé à un ou deux exemples qui permettront de comprendre plus concrètement la mise en œuvre de cette modélisation.

- Diagramme de cas d'utilisation
- Diagramme d'exigences
- Diagramme de séquence
- Diagramme de bloc

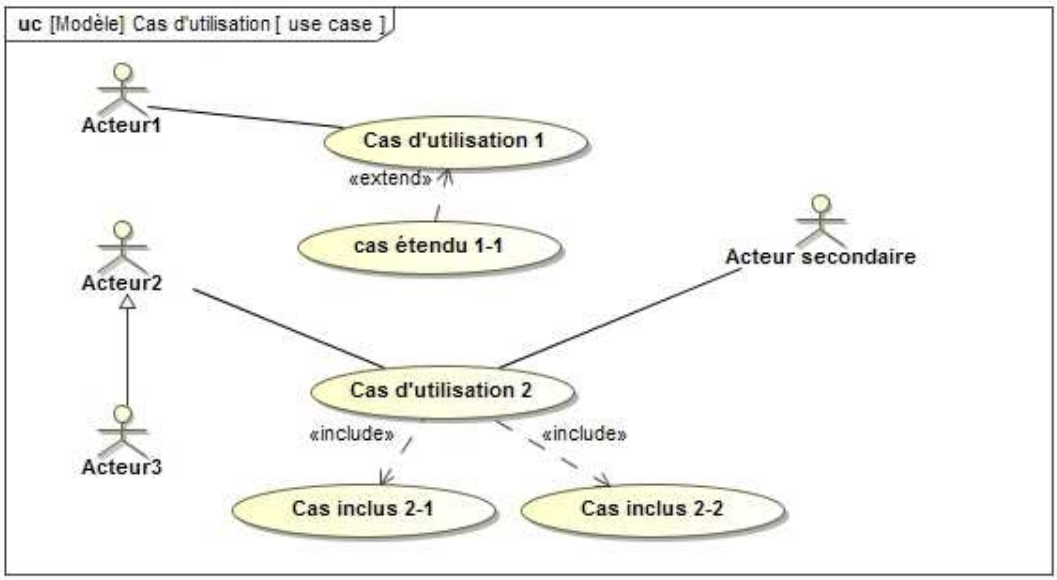
- Diagramme de bloc interne
- Diagramme d'état

Diagramme de cas d'utilisation (Use case Diagram)

Le diagramme de cas d'utilisation permet, comme son nom l'indique, d'expliciter comment un système ou un objet technique est utilisé.
Il explique les lots d'actions que réalisent les acteurs avec le système.
Par "lot d'action", on entend un ensemble d'action : paramétrer un système est un lot d'action, définir la langue d'utilisation de l'interface Homme machine est une action prise dans ce lot.
Chaque lot d'action représente un cas d'utilisation.

Un diagramme de cas d'utilisation, placé dans un cartouche qui spécifie le type de diagramme est composé :

- de cas d'utilisations
- d'acteurs (principaux ou secondaires)
- de liaisons



Les acteurs

Rôle joué par un utilisateur humain, un groupe d'humain, ou un autre système qui interagit directement avec le système étudié.

- L'acteur est extérieur au système et interagit avec lui.
- Un acteur participe à au moins un cas d'utilisation.
- Plusieurs acteurs peuvent participer au même cas d'utilisation
- Une même personne ou entité peut-être plusieurs fois acteur. Par exemple, un musicien peut créer la musique et l'entendre être aussi l'auditeur (c'est d'ailleurs préférable).

Les ***acteurs principaux***, placés à du diagramme agissent directement sur le système. Ils ont besoin d'utiliser le système.
Les ***acteurs secondaires*** à droite du diagramme, n'ont pas besoin d'utiliser le système, il s'agit généralement d'un autre système qui échange avec le système principal. Par exemple un réseau informatique sur lequel le système est connecté et pour lequel le système étudié n'est pas indispensable..

Les cas d'utilisation (Use case)

Chaque cas d'utilisation représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

Le service rendu à l'utilisateur par le système est le cas d'utilisation.
Chaque cas d'utilisation spécifie un comportement attendu du système considéré comme un tout, sans imposer le mode de réalisation de ce comportement. Il permet de décrire ce que le système fait, sans spécifier comment il le fait. Un cas d'utilisation doit être relié à au moins un acteur.

Les liaisons

Les liaisons permettent de relier les acteurs et les cas d'utilisation. Chaque type de liaison possède une représentation propre.

- **association :** il s'agit d'un trait simple qui lie l'acteur au cas d'utilisation. Permet de spécifier que le cas d'utilisation est utilisé par l'acteur
- **inclusion (« include ») :** le cas d'utilisation de base en incorpore explicitement un autre, de façon obligatoire. Par exemple, si le cas d'utilisation est "*Régler la tonalité*", ce cas inclus "*Régler les graves*" et "*Régler les aigües*".
- **extension (« extend ») :** le cas d'utilisation de base en incorpore implicitement un autre, de façon optionnelle, à un endroit spécifié indirectement dans celui qui procède à l'extension (appelé extension point). Par exemple, l'usage du casque audio est un fonctionnement optionnel dont l'utilisation n'est pas systématique en fonctionnement normal.
- **généralisation** : lorsque plusieurs acteurs ont besoin d'accéder au même cas d'utilisation on peut généraliser ces acteurs de sorte que le diagramme reste plus lisible.

Exemple 1 : Distributeur d'aliments

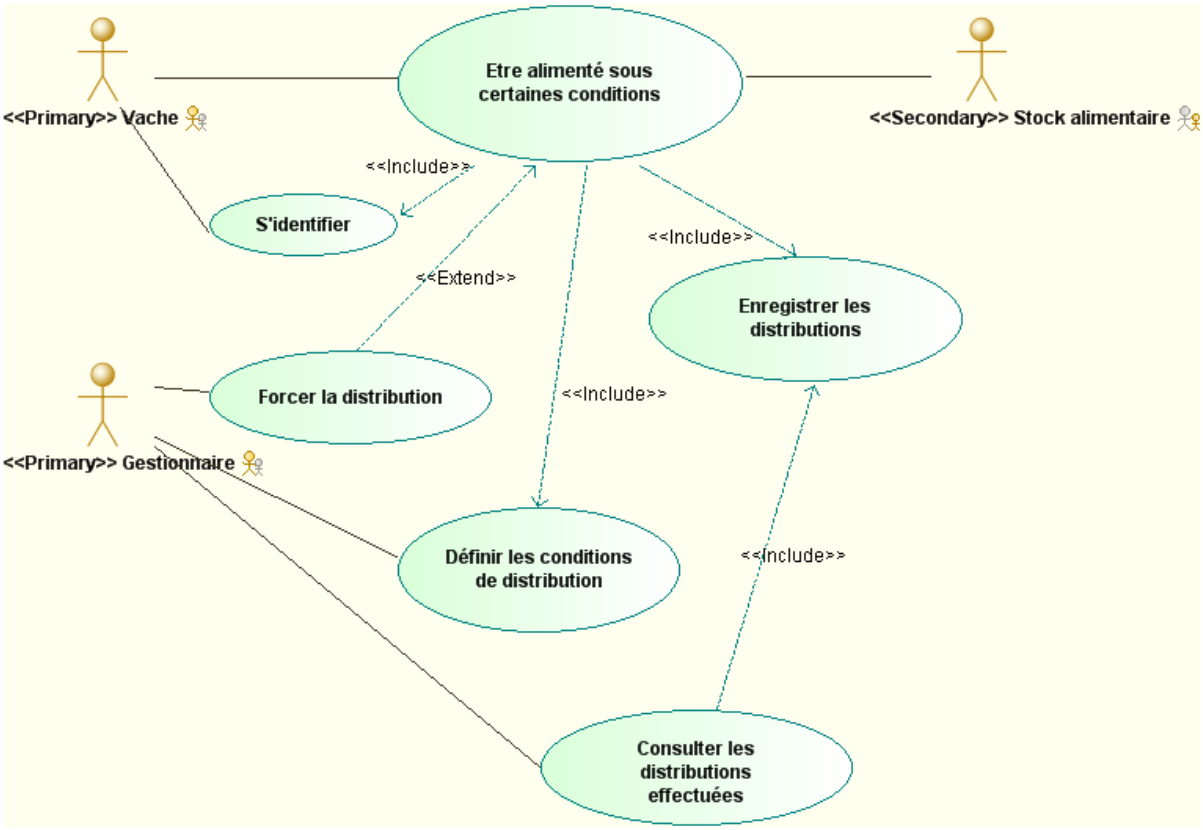
Ce diagramme des cas d'utilisation présente un distributeurs d'aliments automatique pour bovins. Le système doit permettre à la bête (une vache par exemple)

- de s'alimenter si les conditions le permettent
- de s'identifier

Il doit également permettre à l'éleveur :

- de paramétrer le système en particulier définir les conditions de distribution des aliments
- de consulter les statistiques de distribution
- et éventuellement de forcer la distribution

Le système s'appuie bien sûr sur un stock alimentaire mais ce dernier à une existence sans le système de distribution. Le stock est donc un acteur secondaire.



On remarque que :

- les verbes d'actions à l'infinitif décrivent les cas d'utilisation
- le forçage étant une option du cahier des charges, la liaison est de type EXTENSION
- pour que la nourriture soit distribuée de manière personnalisée, l'identification est un cas d'utilisation d'INCLUSION. Il en est de même pour l'enregistrement

Exemple 2 : Contrôle d'accès

Ce diagramme décrit les cas d'utilisation d'un système de contrôle d'accès à un local sécurisé par empreinte biométrique.

Les acteurs qui souhaitent accéder à la zone sont à la fois des utilisateurs autorisés, d'autres non-autorisés et l'administrateur du système. Dans le diagramme, tous les acteurs sont généralisés par "Utilisateur extérieur à la zone".

Le système agit sur une porte dont le fonctionnement doit être possible même sans ce système

Pour les utilisateurs hors zone qui souhaitent accéder, les cas d'utilisation sont :

- pouvoir présenter leur empreinte
- être avertis de l'autorisation d'accès
- enregistrer l'empreinte

L'administrateur du système doit pouvoir :

- superviser les accès
- enregistrer de nouveaux utilisateurs
- forcer le déverrouillage de la serrure

L'utilisateur dans le local doit pouvoir ressortir et donc déverrouiller la serrure.

La CNIL doit être avertie du stockage de données d'utilisateurs (Base de donnée). Elle est donc un acteur secondaire également.

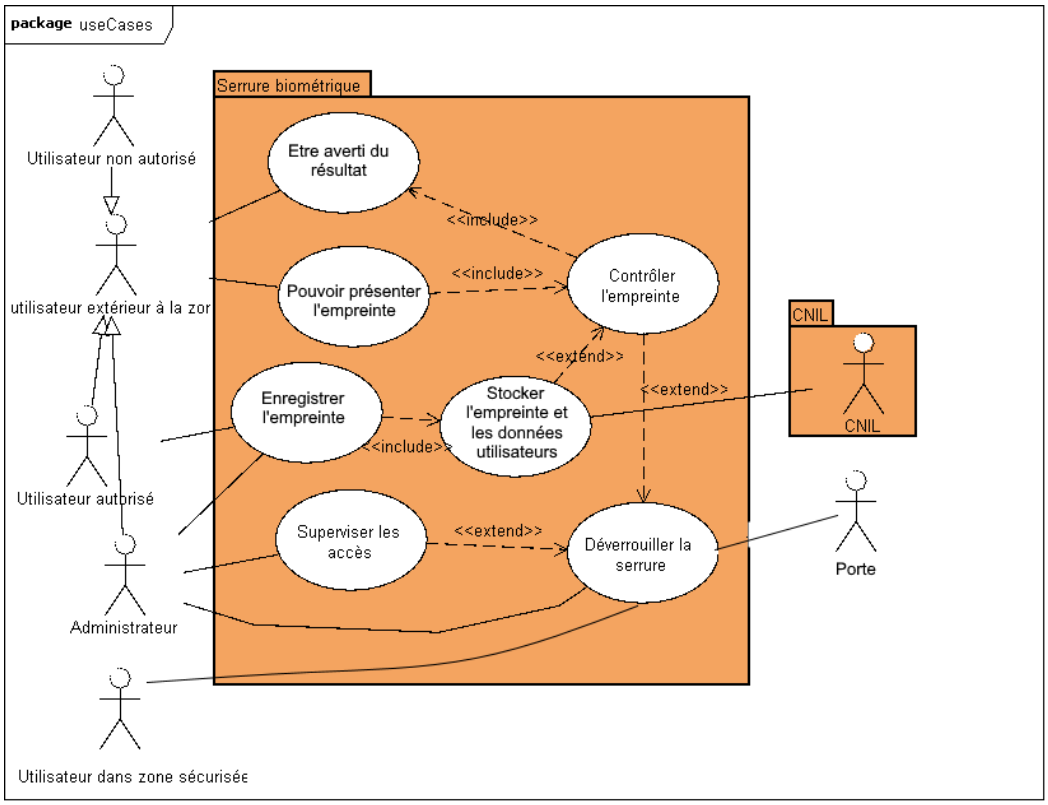
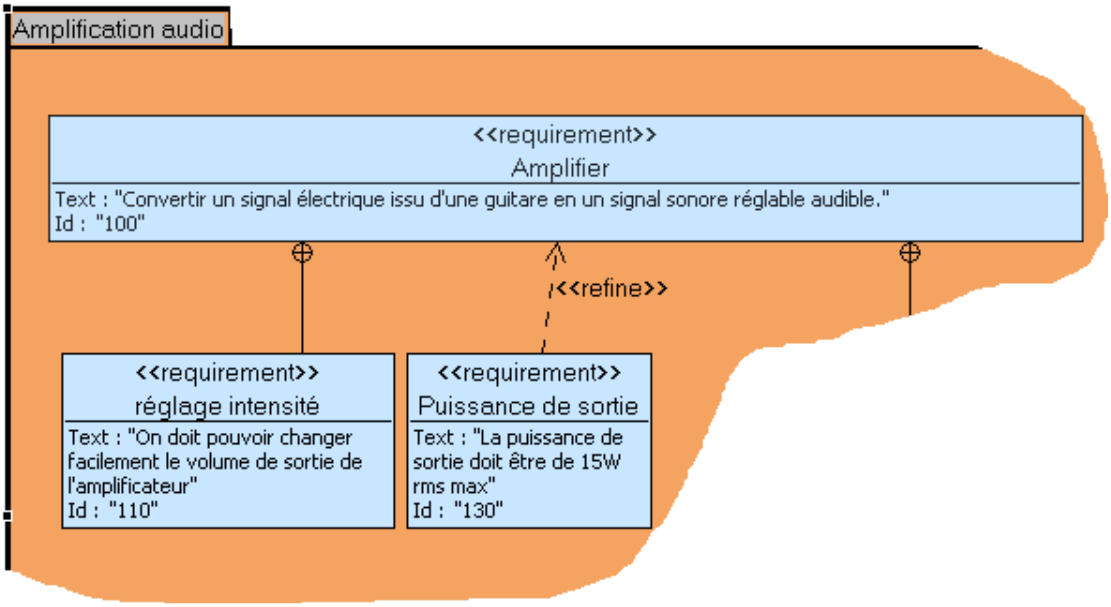


Diagramme d'exigences (Requirement diagram)

Le diagramme d'exigence décrit graphiquement une capacité ou une contrainte qui doit être satisfaite par un système. C'est une interprétation du cahier des charges.

Un diagramme d'exigence, placé dans un cartouche de type ***req***, comporte :

- Des exigences décrites dans un cadre d'exigence (Requirement)
- Des liaisons
- Parfois le diagramme est inclus dans un package.



Exigence

Chaque cadre d'exigence dispose de trois informations au moins :

- un nom : il s'agit d'une chaîne de caractères décrivant l'exigence. Elle doit être courte précise et de préférence unique
- le texte de description: il décrit et précise l'exigence
- l'identifiant : il doit être unique et de préférence hiérarchisé. Un diagramme SysML pouvant être compilé pour générer un programme une duplicité d'identifiant générerait automatiquement une erreur.

Liaison

3 types de liaisons sont vraiment fondamentales :

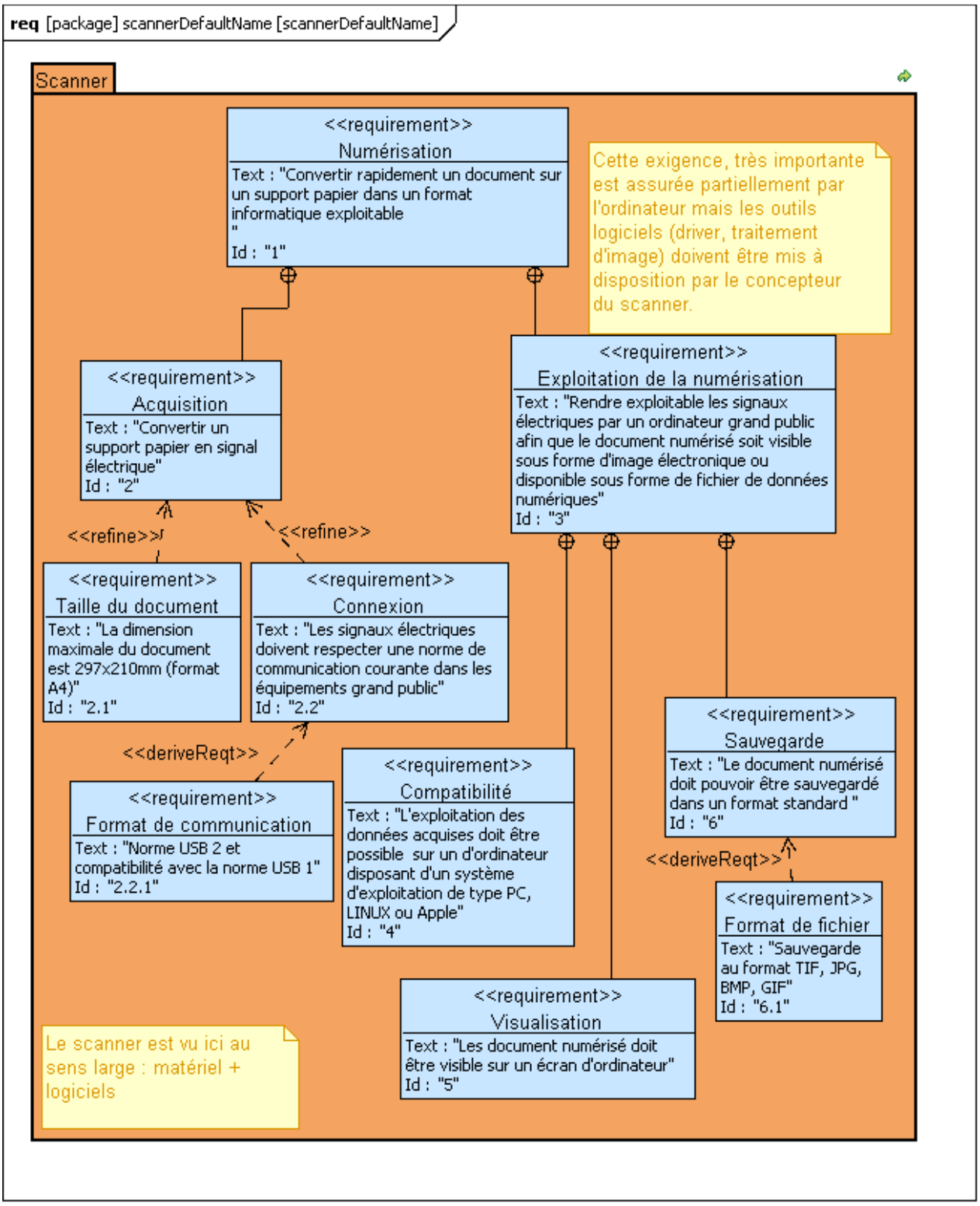
- la contenance (ligne terminée par un cercle contenant une croix du côté du conteneur) permet de décomposer une exigence composite en plusieurs exigences unitaires, plus faciles ensuite à tracer vis-à-vis de l'architecture ou des tests ;
- le raffinement (« refine ») consiste en l'ajout de précisions, par exemple de données quantitatives ;
- la dérivation (« deriveReqt ») consiste à relier des exigences de niveaux différents, par exemple des exigences système à des exigences de niveau sous-système, etc. Elle implique généralement des choix d'architecture.

Package

Si le système est trop important, il peut être décomposé en partie. Chaque partie serait alors placée dans un package et chaque package aurait son propre diagramme d'exigence.

Exemple n°1 : Scanner

Le scanner est vu ici comme une partie d'un système de traitement d'image. Le package se justifie tout à fait.



L'exigence principale sous-entend deux exigences secondaires :

- l'acquisition
- l'exploitation de la numérisation

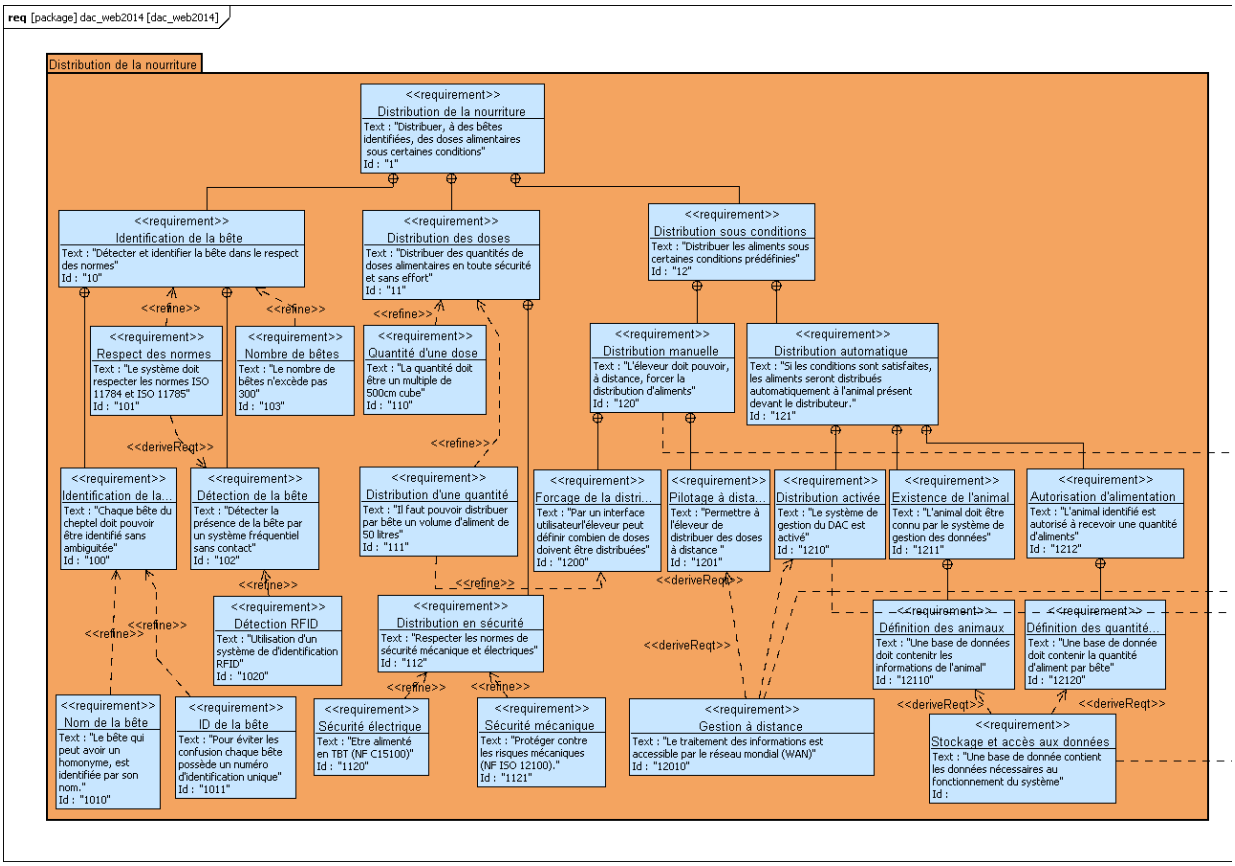
On remarque que les relations de raffinement précisent les exigences :

- taille maximale de l'acquisition : A4
- interfaces de connexions standards
- format d'acquisition

Les exigences dérivées sous-entendent des choix technologiques qui se justifient par l'usage de matériel informatique standard.

Exemple n°2 : Distributeur d'aliments

Il s'agit du distributeur d'aliments dont le diagramme de cas d'utilisation a été décrit précédemment.



La complexité du système implique plusieurs parties présentées dans des *package* différents (ici Distribution de la nourriture).

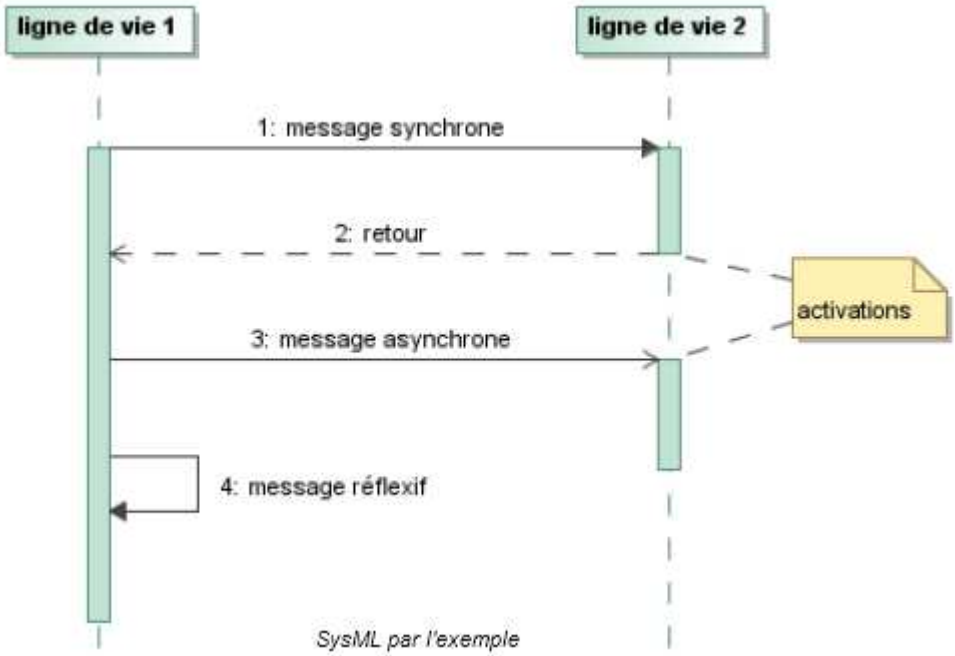
A remarquer :

- chaque exigence dispose d'un ID spécifique
- le numéro des ID est hiérarchisé
- le respect des normes étant une exigence, une relation de raffinement précise les normes
- une relation *copy* permet de reprendre dans un autre package une exigence déjà décrite

Diagramme de séquence

Le diagramme de séquence montre la chronologie des événements et des messages passés entre éléments (lignes de vie) au sein d'une interaction.

La progression temporelle est verticale et les éléments sont représentés horizontalement.



Les éléments sont les entités qui sont en mesure de délivrer des messages. Un utilisateur peut donc figurer dans le diagramme de séquence.

Il est recommandé de garder une certaine cohérence entre le diagramme de cas d'utilisation et le diagramme de séquence. Ainsi, l'acteur sera placé à gauche du diagramme.

Un diagramme simple peut se limiter aux lignes de vie et aux messages, mais souvent il faudra faire appel aux *fragments combinés* qui permettent d'affiner la représentation.

Lignes de vie

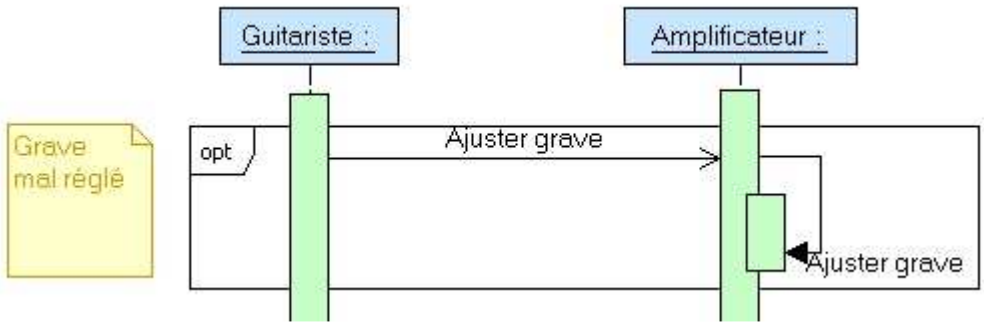
Comme cela a déjà été annoncé, la ligne de vie décrit une entité du système.
La progression chronologique se fait du haut vers le bas.
Lorsque l'entité s'active, le trait en pointillé est remplacé par un trait large. Lorsqu'un système n'est plus en fonction toutes les entités sont désactivées.

Messages

Ce sont des éléments de communication unidirectionnel entre les lignes de vie qui déclenchent une activité dans le destinataire. La réception d'un message provoque un événement chez le récepteur. La flèche pointillée représente un retour. Cela signifie que le message en question est le résultat direct du message précédent.

Un ***message synchrone*** (émetteur bloqué en attente de réponse) est représenté par une flèche pleine, alors qu'un ***message asynchrone*** est représenté par une flèche évidée.
La flèche qui boucle (***message réflexif***) permet de représenter un comportement interne.

Fragments combinés



Il s'agit d'une notation qui permet de décrire avec plus de finesse le cycle d'un système :

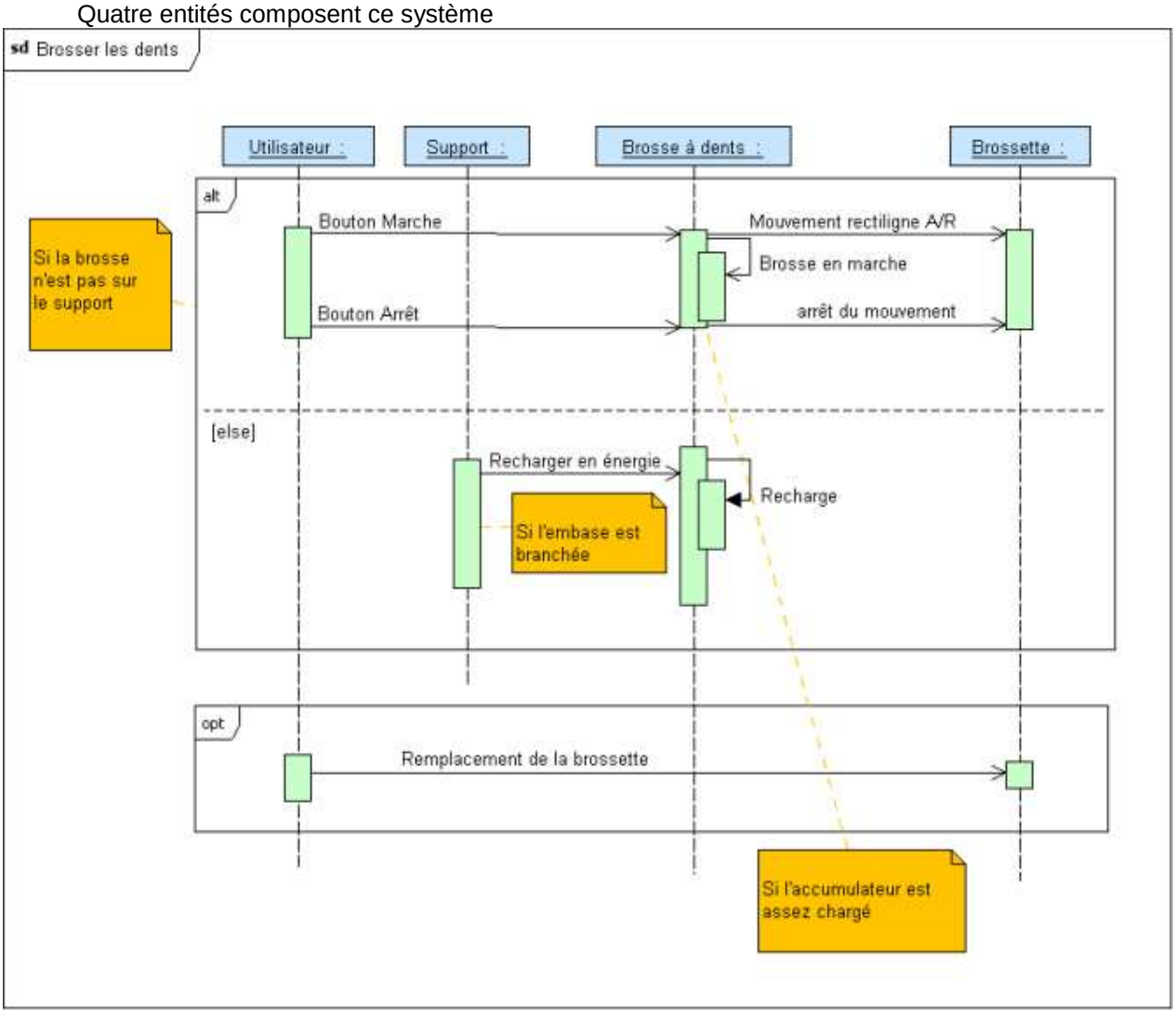
Les principaux opérateurs sont :

- **loop** : boucle. Le fragment peut s'exécuter plusieurs fois, et la condition de garde explicite l'itération;
- **opt** : optionnel. Le fragment ne s'exécute que si la condition fournie est vraie ;
- **alt** : fragments alternatifs. Seul le fragment possédant la condition vraie s'exécutera.

Exemple n°1 : Brosse à dent électrique

Exemple n°1 : Brosse à dent électrique

Quatre entités composent ce système

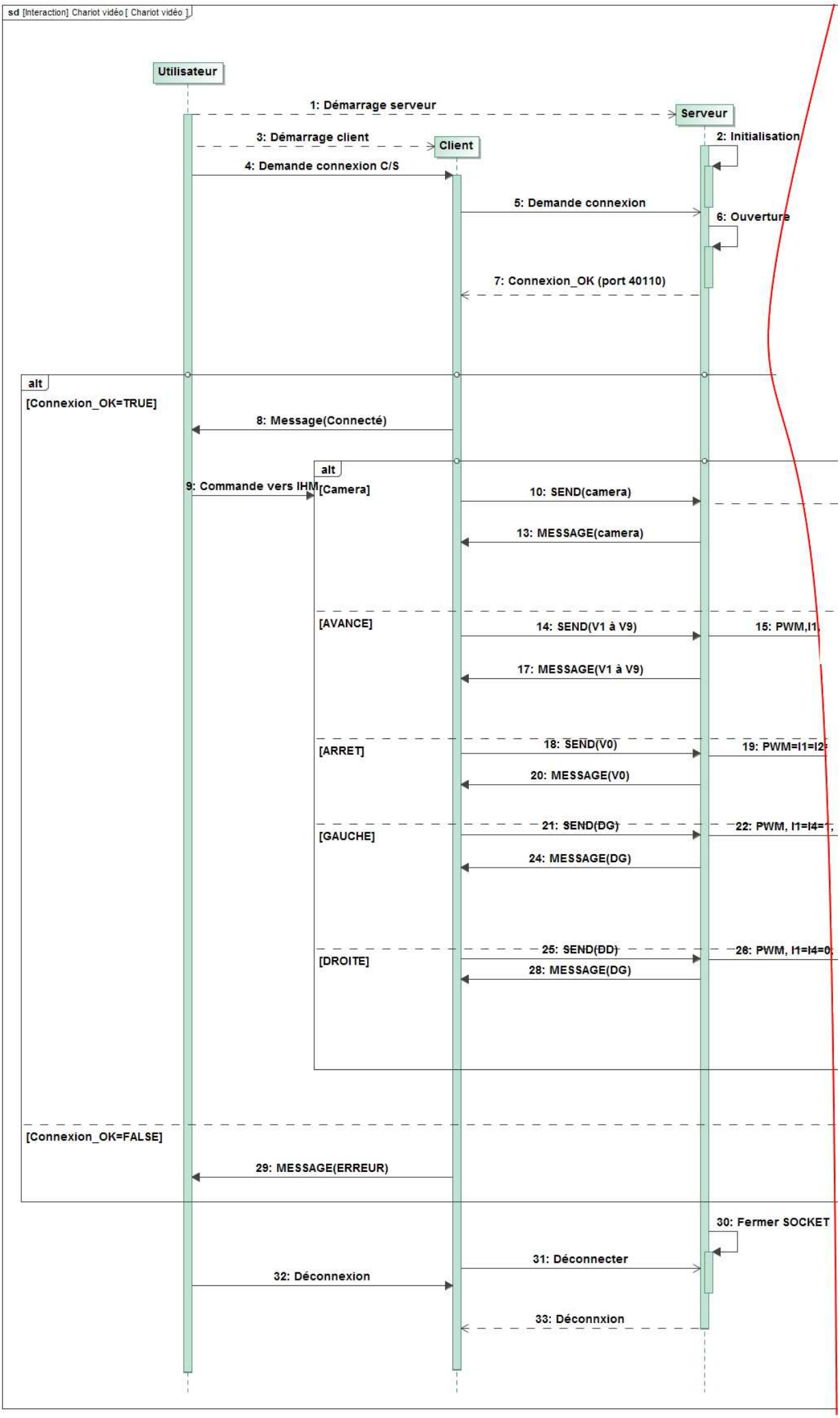


On remarque que :

- Le fragment combiné permet de définir le fonctionnement si la brosse est sur le support ou non
- Lorsque la brosse est sur son support, la ligne de vie de l'utilisateur s'arrête. Il n'y a pas d'utilisation dasn cette situation.

Exemple n°2 : Caméra pilotée

Le diagramme de séquence partiel ci-dessous correspond à un chariot supportant une caméra dont le déplacement et la transmission d'image se fait grâce à une liaison WIFI et une liaison Client/Serveur


sd [Interaction] Chariot vidéo [Chariot vidéo]
Utilisateur
Client
Serveur
1: Démarrage serveur
2: Initialisation
3: Démarrage client
4: Demande connexion C/S
5: Demande connexion
6: Ouverture
7: Connexion_OK (port 40110)
alt
[Connexion_OK=TRUE]
8: Message(Connecté)
alt
9: Commande vers IHM
[Camera]
10: SEND(camera)
13: MESSAGE(camera)
[AVANCE]
14: SEND(V1 à V9)
15: PWM,I1
17: MESSAGE(V1 à V9)
[ARRET]
18: SEND(V0)
19: PWM=I1=I2=
20: MESSAGE(V0)
[GAUCHE]
21: SEND(DG)
22: PWM, I1=I4=
24: MESSAGE(DG)
[DROITE]
25: SEND(DD)
26: PWM, I1=I4=0
28: MESSAGE(DG)
[Connexion_OK=FALSE]
29: MESSAGE(ERREUR)
30: Fermer SOCKET
31: Déconnecter
32: Déconnexion
33: Déconnxion

Ce diagramme présente clairement la mise en route du serveur (1) et du client (3) par l'utilisateur. Un message réflexif provoque l'initialisation de ce dernier (2)
Lorsque l'utilisateur sollicite la demande de connexion C/S (4), le client fait une demande de connexion au serveur en envoyant un message synchrone (5). Rien ne peut se produire au niveau du client tant qu'il n'y a pas eu de message de retour. Le serveur ouvre un socket lors de la phase d'ouverture de la liaison (6) puis autorise la liaison du client (7).
L'utilisateur peut alors piloter le chariot par un ensemble de messages. Un fragment combiné de type alt décrit les messages transmis dans ces cas (10, 14, 1821, 25).

Diagramme de définition de bloc

Le diagramme de définition de bloc (bdd : Block Definition Diagram) est un diagramme qui décrit la structure d'un système. Il est, avec le diagramme de bloc interne, un diagramme architectural.

Il décrit toutes les structures du système modélisé :

- logiques
- matérielles
- fonctionnelles
- ou la matière d'œuvre

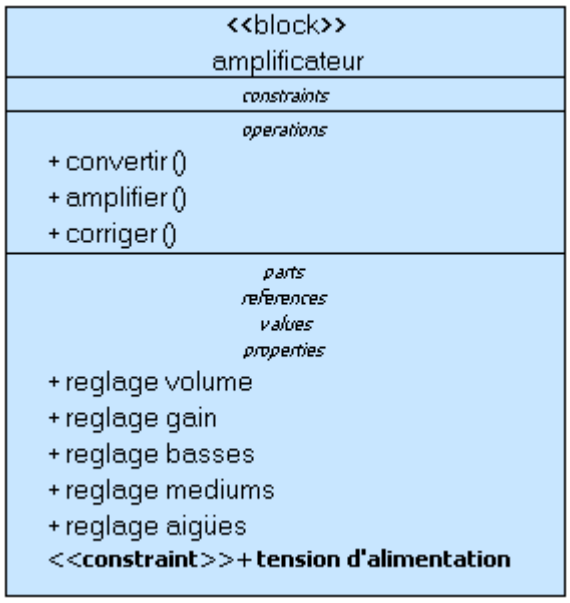
Le diagramme de définition de blocs est composé de blocs de fonction et de liaisons.

Bloc

Le bloc représente le système complet, ou sous système ou un élément du système. Cela peut même être un composant.

Un bloc est représenté graphiquement par un rectangle comportant plusieurs compartiments.

- Le compartiment supérieur contient le nom du bloc
- Le compartiment ***Operations*** détaille les actions réalisées par le bloc. Si le bloc représentait un petit programme, les opérations pourraient être les procédures de ce programme
- Le compartiment inférieur contient les ***paramètres*** qui définissent les instances du bloc : les parties, les références, les valeurs et/ou les propriétés.
- le compartiment ***constraint*** permet de définir les contraintes de la fonction.

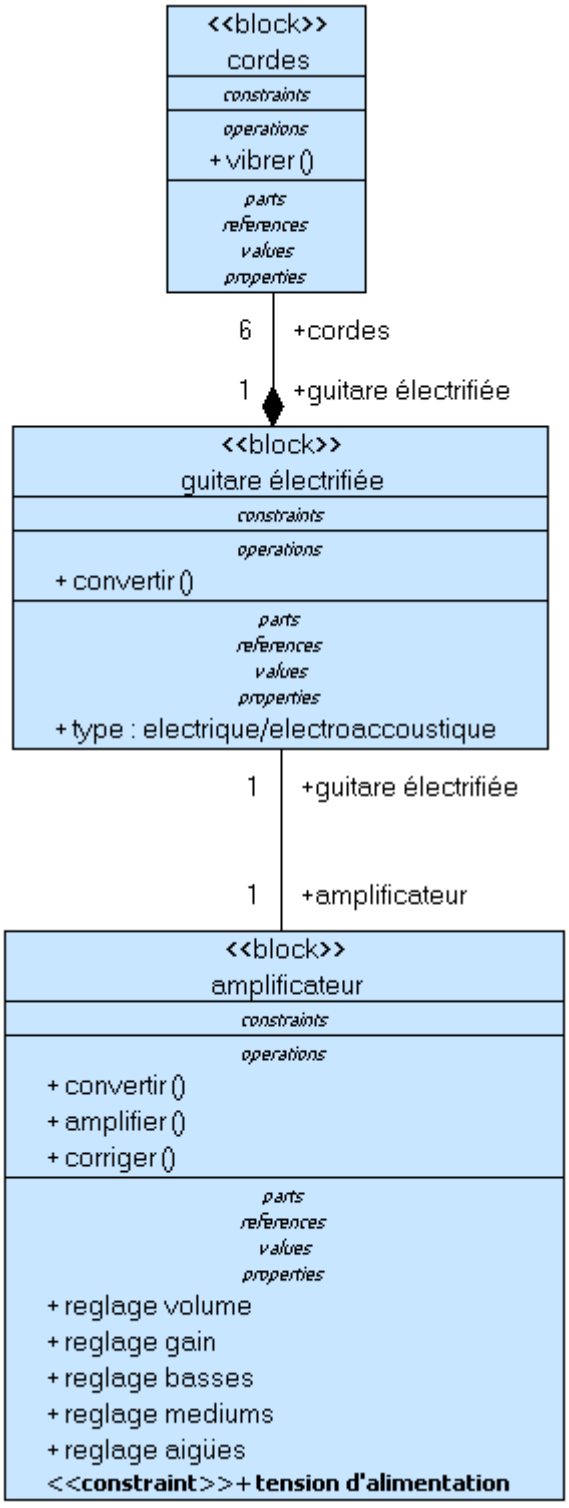


Dans l'exemple ci-dessus, l'amplificateur converti un signal électrique en signal sonore tout en assurant une amplification et une correction de tonalité.

Son fonctionnement est conditionné par l'alimentation en énergie électrique.

Les différents réglages constituent les différents paramètres.

Relations



Dans un diagramme de définition bloc, il y a 3 relations possibles :

- La ***composition***, l'un des bloc est le contenant et l'autre est le contenu, mais le contenant n'a pas de raison d'être sans le contenu. Exemple la guitare et les cordes de guitare : Sans cordes, la guitare n'a aucun intérêt mais les cordes sont contenues dans la guitare. Une relation de composition est représentée par un losange plein.
- ***L'association*** représentée par un trait simple. Elle lie deux blocs égaux. Si on prend comme exemple, l'ensemble guitare + amplificateur, chaque sous-système est indépendant, mais ils sont liés.
- ***L'agrégation*** est identique à la composition, à la différence prêt que le contenu a une existence sans le contenant. Par exemple une salle de classe est composée de tables et de chaises mais sans tables ni chaise la salle peut encore être utilisée. Une relation d'agrégation est représentée par un losange vide.

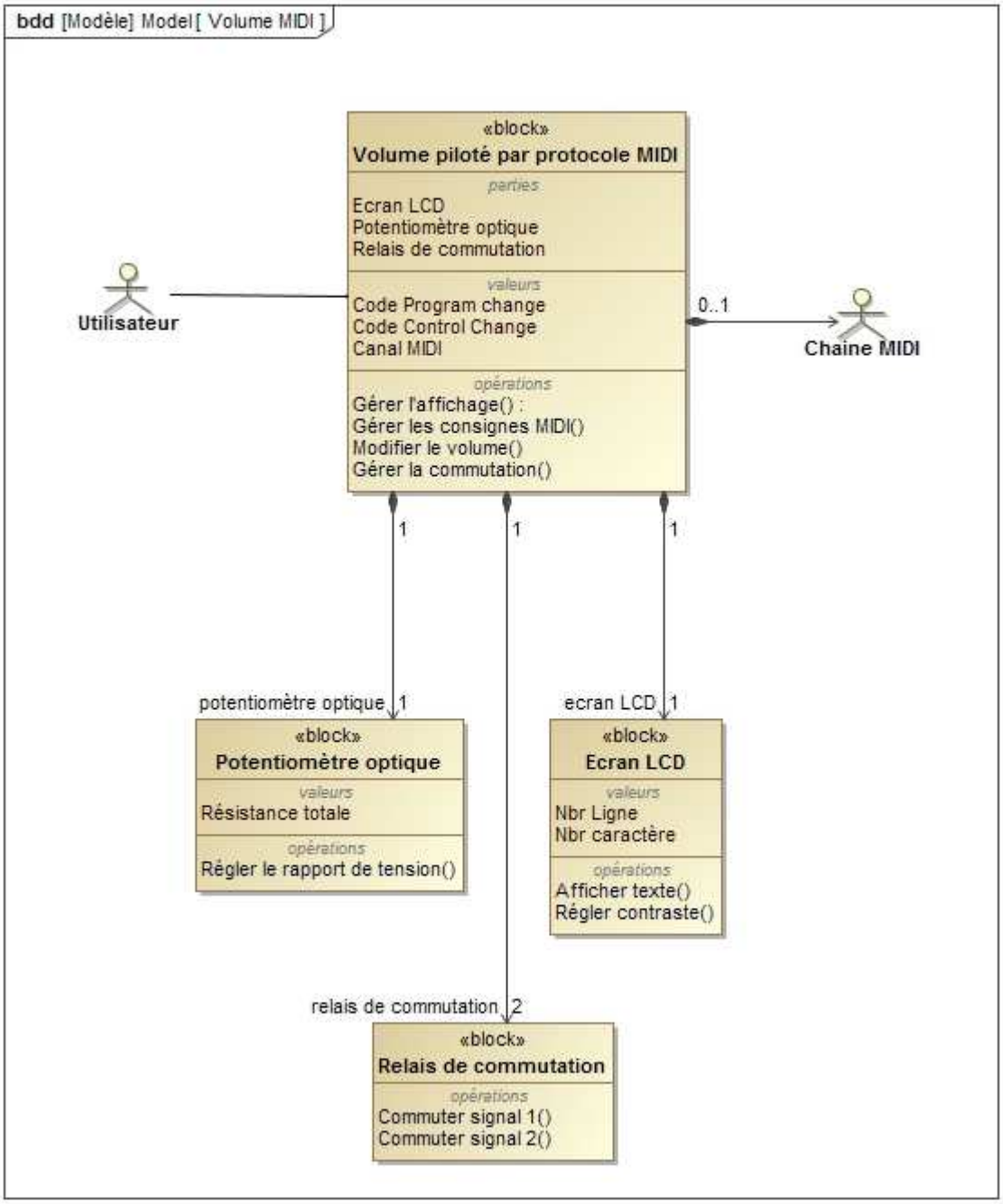
Les lignes de relation sont associées à des numéros. Dans l'exemple ci-contre, il y a 6 cordes sur la guitare électrique (dans la cas d'une guitare basse on mettrait généralement 4 cordes ou 12 pour une guitare 12 cordes). Le bloc CORDES est donc associé à 6 et le bloc guitare à 1.

Dans le cas de l'amplificateur pris comme exemple seule une guitare peut être branchée sur l'amplificateur d'où le chiffre "1" des deux cotés de la ligne.

Dans le cas où le nombre est indéterminé, on peut mettre une étoile (cas des tables dans une salle de classe.)

Exemple n°1 : Volume MIDI

Il s'agit ici d'un système permettant de régler le volume sonore d'un instrument de musique à partir d'un équipement intégré dans une chaîne de traitement MIDI (Musical Instrument Digital Interface).
L'utilisateur, qui peut ne pas utiliser le système (association), agit sur l'équipement intégré ou non dans une chaîne MIDI.



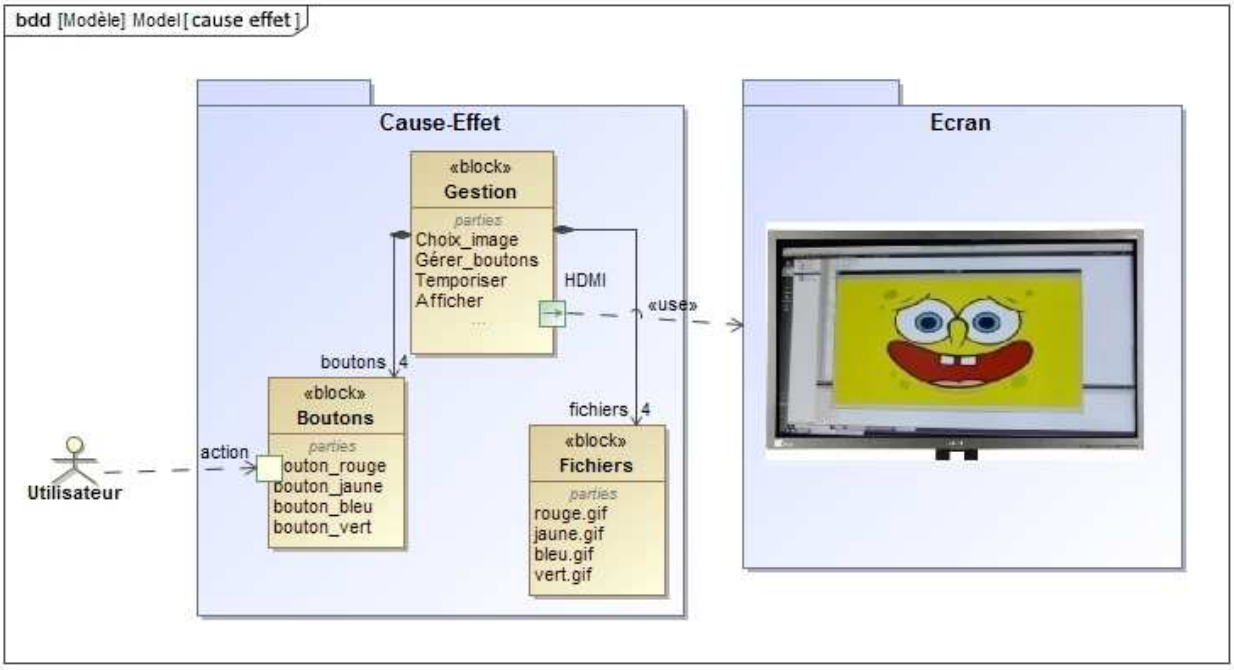
L'équipement possède :

- un affichage LCD de 2x16 caractères
- un potentiomètre optique qui assure la variation de la tension de sortie
- deux relais de commutation qui permettent de réaliser les fonction Mute (extinction du signal de sortie) ou Bypass (le signal d'entrée est directement envoyé sur la sortie).

Chaque ***block*** réalise des opérations et dispose de valeurs.

Exemple n°2 : Cause à effet

Il s'agit ici d'un système disposant de 4 boutons de couleur. La pression sur un bouton provoque l'affichage sur un écran, pendant une certaine durée, d'une image animée.



A remarquer :
Les ports qui permettent de spécifier les liaisons avec d'autres package

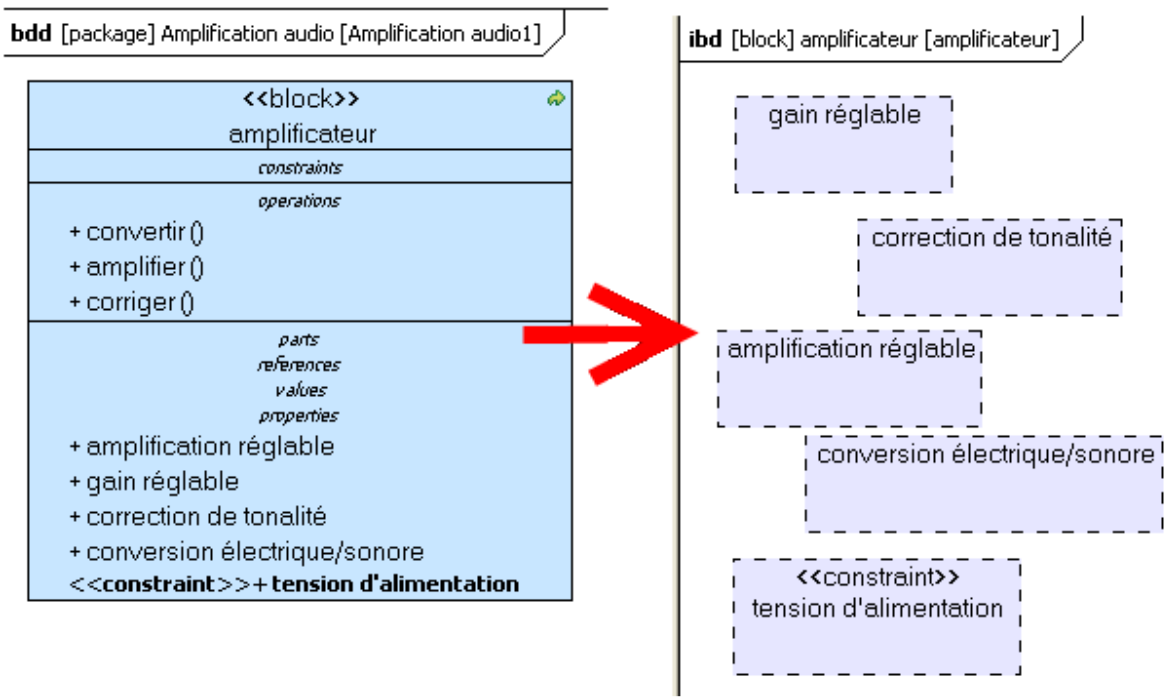
Diagramme de bloc interne

Le diagramme de bloc interne modélise la structure interne d'un bloc. Il décrit les flux (MEI) et interactions entre des blocs qui caractérisent les fonctions. Un diagramme de bloc interne est obligatoirement un descendant d'un diagramme de bloc. On reconnaît le diagramme de bloc interne par les trois lettres ***ibd*** qui surplombent le diagramme.

Les parties

Elles caractérisent des sous fonctions structurelles.

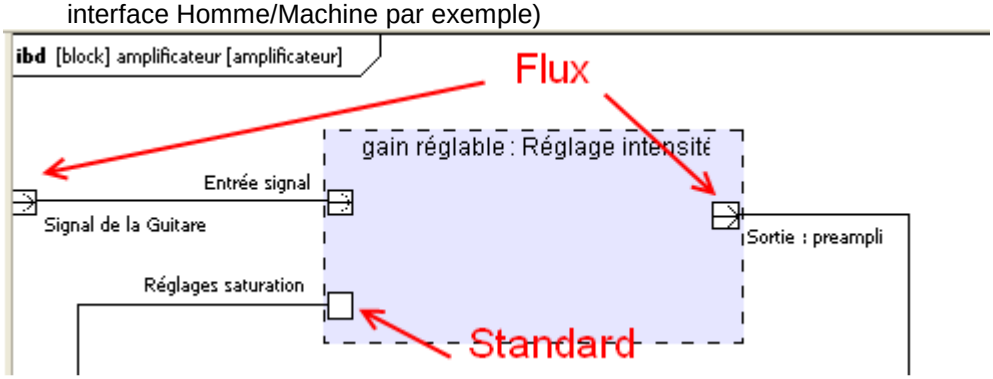
Les ***parts*** ou ***properties*** d'un bloc du diagramme de définition de bloc peuvent deviennent des parties du diagramme de bloc interne.



Les ports

Les ports sont des "points d'interaction" de la partie.
Ils peuvent être de deux natures :

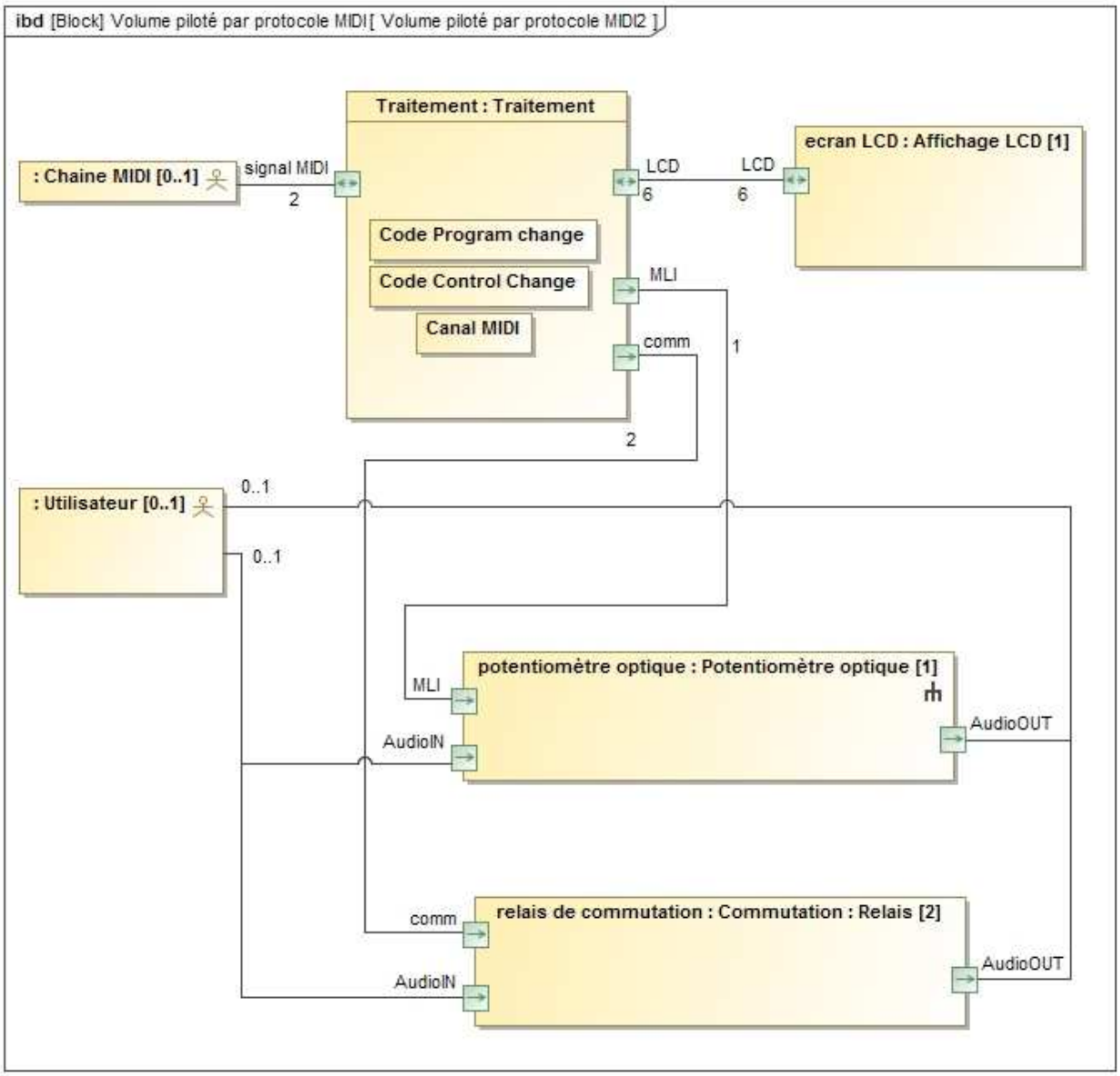
- **flux** : il peut s'agir de matière d'énergie ou d'information (MEI). Il faudra préciser le sens du flux (in, out, ou inout)
- **standard** : il peut s'agir de services logiques abstraits (interruption logicielle, ou interface Homme/Machine par exemple)



Les ports standards sont en général sans direction déterminée mais il est possible de définir une direction d'information grâce à une schématisation particulière

Exemple n°1 : Volume MIDI

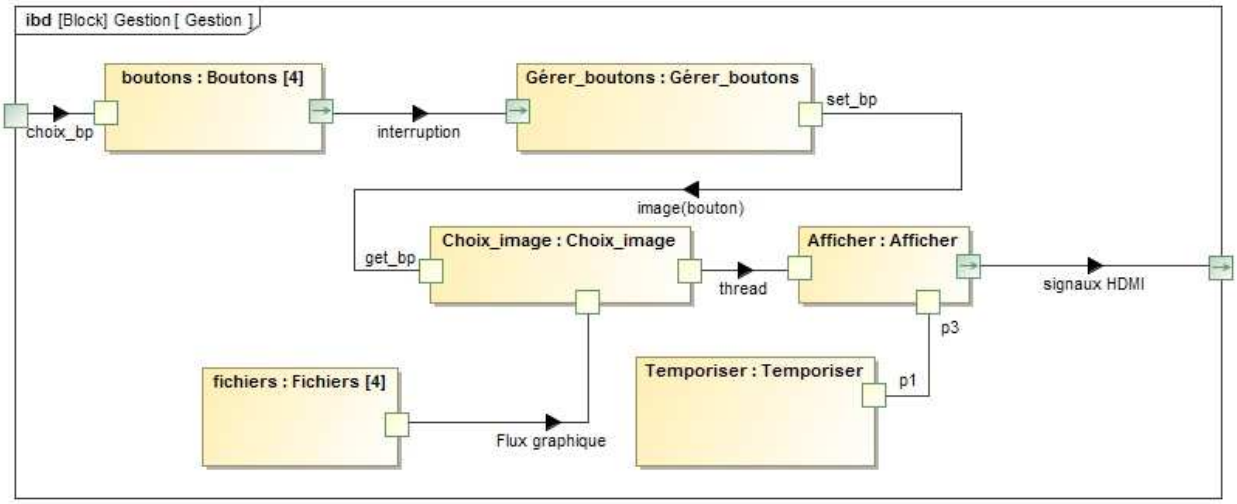
Ce diagramme de bloc interne (ibd) est directement dérivé du diagramme de définition de bloc précédent.



Chaque cadre représente une fonction interne du système et peut-être presque associé aux structures électroniques liées.

Exemple n°2 : Cause à effet

Ce diagramme de bloc interne (ibd) est directement dérivé du diagramme de définition de bloc précédent.



On peut remarquer :

Les signaux traités ici sont pour la plupart des signaux d'ordre logiciel. Il ne s'agit donc pas de flux de données mais plutôt de services logiques ce qui explique les ports standards.

Diagramme d'état

Le diagramme d'état décrit les transitions entre les états et les actions que le système ou ses parties réalisent en réponse à un événement.

Il s'agit d'une représentation séquentielle des états d'un système.

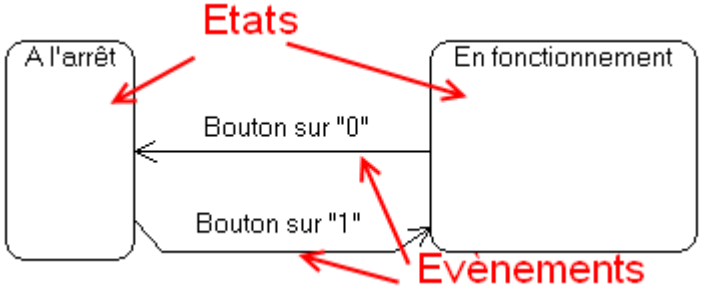
Le diagramme d'état se compose

- d'états
- de transitions
- d'évènements
- de conditions
- d'effets
- d'activités

Etat - transition

Un état (state) représente une situation d'un bloc fonctionnel pendant laquelle

- il satisfait une certaine condition
- il exécute une certaine activité



Les différents états sont reliés entre eux par des transitions.

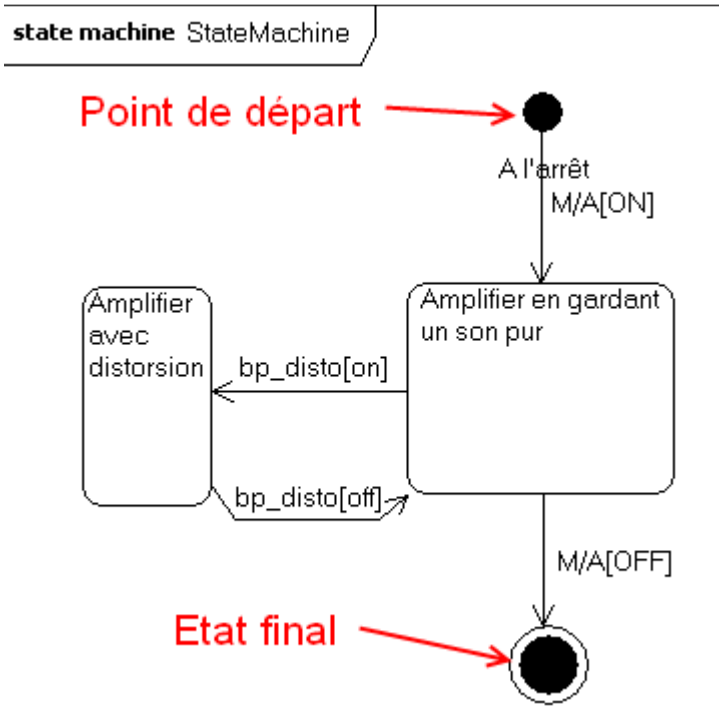
Une **transition** est liée à un événement. Lorsque l'événement se produit, il peut provoquer le changement d'état de la fonction qui lui est liée. Une transition peut être complétée par une condition (***guard***) notée entre croche, des paramètres notés entre parenthèses et d'une activité.

Ainsi la syntaxe complète de la transition serait : ***evenement[condition](paramètres)/activité***

Dans l'exemple ci-dessus le basculement du bouton marche-arrêt provoque un événement qui place

Evolution du diagramme d'état

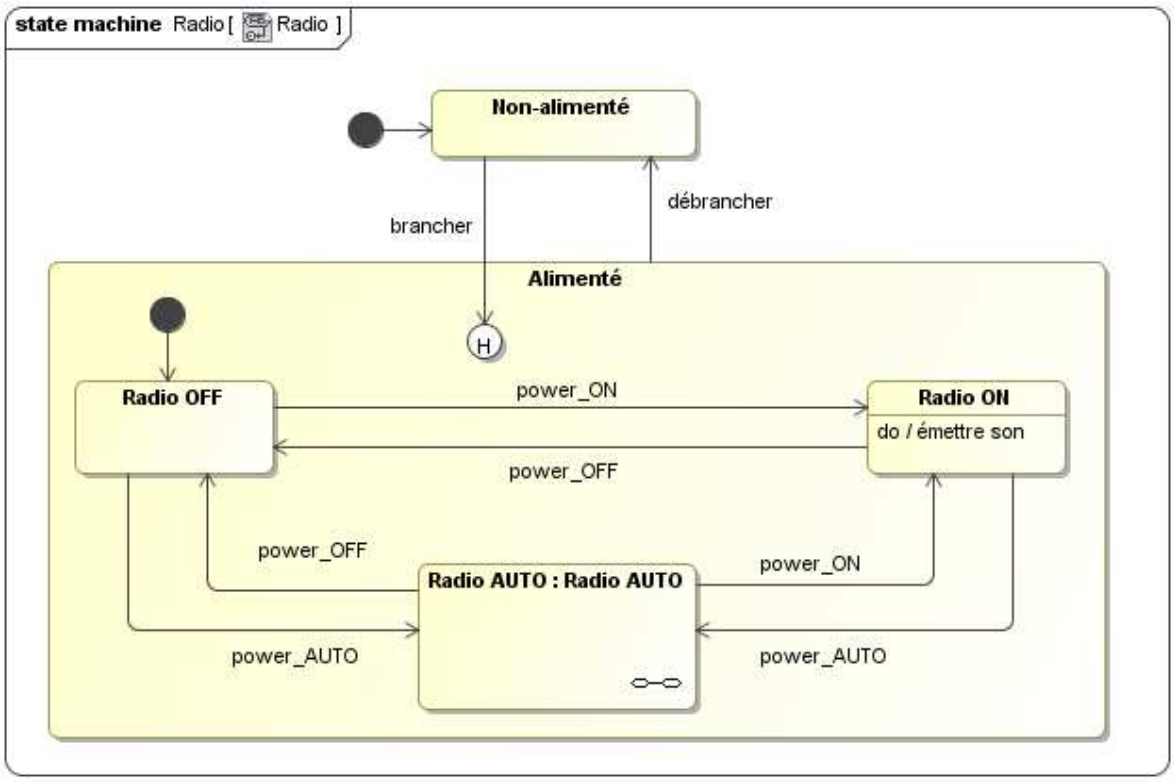
A la création de l'instance du graphe, un jeton virtuel est placé au point de départ. Si les conditions de la transition le permettent, le jeton quitte le point de départ et rejoint le premier état. L'entrée du jeton provoque l'exécution de la tâche d'entrée de l'état (***entry bahavior***).Pendant le temps de présence du jeton, l'activité (***do active***) de l'état se réalise. Quand une transition pour sortir de l'état est vérifiée, le jeton quitte l'état et provoque l'exécution de l'activité de l'action de sortie (***exit behavior***). Le graphe reste actif tant que le jeton n'a pas rejoint la sortie.



Si plusieurs transitions peuvent être franchies, une seule le sera de manière non déterministe. Il faut donc être attentif à la gestion de ces situations.

Les super-états, pseudo-états et mémoire

Le diagramme d'état peut comporter des super-états (***submachine state***) qui encapsulent d'autres diagrammes d'états. Ils permettent de "factoriser" des transitions déclenchées par le même événement et amenant vers le même état cible (brancher, débrancher) tout en spécifiant des transitions particulières entre les sous-états.



sources : SysML par l'exemple de Pascal Roques aux éditions Eyrolles

La mémorisation est modélisée par un pseudo état ***History*** symbolisé par la lettre H. L'activation de ce pseudo-état permet au super état de se souvenir du dernier sous-état qui était actif avant une transition sortante.

Dans l'exemple ci-dessus, grâce au pseudo état ***History*** le fait de débrancher la RADIO provoque la mémorisation des paramètres programmés. Lorsque l'appareil sera à nouveau rebranché les paramètres mémorisés seront repris.

On peut définir qu'une mémorisation (***History***) se propage quelque soit le niveau de profondeur des diagrammes d'état. On utilise alors le ***Deep History*** représenté par H*.