SysML, un langage modèle

JEAN-PIERRE LAMY [1]

Nouveauté remarquée dans le programme de STI2D, SysML se veut un outil cohérent de modélisation des systèmes pluritechnologiques actuels. Mais il vient quelque peu bousculer les habitudes que nous avions, chacun dans notre domaine, dans les descriptions fonctionnelle, structurelle et comportementale des systèmes techniques. Afin de nous éclairer, voici une analyse fonctionnelle menée avec les diagrammes SysML et appliquée au produit Hemomixer.

mots-clés
lycée technologique, modélisation, outil et méthodes

Le langage SysML

On le sait, un système est un ensemble de composants qui interagissent d'une manière organisée pour accomplir une finalité commune. L'organisation structurale du système et ses interactions avec l'environnement lui donnent sa signification et lui permettent d'atteindre sa finalité. Avec le progrès de la science et de la technologie, les systèmes deviennent de plus en plus complexes, parce qu'ils sont constitués d'un nombre important de composants de natures différentes, parce que leur résolution fait intervenir des spécialistes de disciplines diverses, enfin parce qu'il est difficile de prévoir leur comportement par des méthodes réductionnistes. Le comportement global d'un système émerge des interactions simples de ses constituants, mais il est beaucoup plus riche que la somme des comportements individuels. Ce phénomène génère de la complexité qui ne pourra être maîtrisée que par des approches systémiques transdisciplinaires.

Aujourd'hui, des approches comme l'ingénierie système proposent un ensemble de démarches méthodiques associées à des outils qui permet de maîtriser la compréhension, le développement et l'exploitation de systèmes techniques complexes. Ces outils permettent de donner une représentation de la réalité sous des formes simplifiées selon les besoins du moment. Nous appellerons ces différentes visions simplifiées des *modèles*. Ces modèles sont en grande majorité représentés sous forme symbolique par des outils à base de langages graphiques. Ces langages – instrument de la pensée et de la communication – sont mis ici au service de la conception et de la compréhension d'artefacts par des équipes pluridisciplinaires. SADT, les outils classiques de l'analyse fonctionnelle (outils de la méthode APTE, Fast…) et le Grafcet – pour ne prendre qu'eux – que nous connaissions déjà font partie de ces langages. Cependant, un ensemble d'outils disparates, aussi performants soient-ils, ne donnera jamais une vision globale cohérente de la chose représentée. L'idée moderne est de fédérer plusieurs langages spécialisés au sein d'un « méta-outil ». Une vision pertinente du système pourra ainsi émerger de ses différentes composantes spécialisées liées par un métalangage garant de la cohérence. C'est l'objectif de SysML (*Systems Modeling Language*).

SysML n'est pas une méthode, mais un ensemble d'outils graphiques définis par un métalangage qui offrent au concepteur toutes les facilités pour construire un modèle à forte cohérence sémantique. Il permet de spécifier les systèmes, de concevoir, définir et analyser leur structure et leur fonctionnement dynamique, de simuler leur comportement afin de valider leur faisabilité avant leur réalisation. Il intègre les composants physiques de toutes technologies, les programmes, les données et les énergies, les personnes, les procédures et les flux divers.

Quelles sont alors les fonctionnalités d'un tel outil de représentation ?

- **Il facilite la collaboration** transdisciplinaire de tous les spécialistes des corps de métier concernés en proposant un ensemble lié d'outils de représentation suffisamment universels pour qu'ils puissent être compris par tout le monde, et suffisamment expressifs pour qu'il puissent représenter toutes les composantes hautement spécialisées ainsi que les différents points de vue d'un même système.
- **Il permet la mise à jour**, le stockage et, surtout, le partage ainsi que l'interprétation faciles des informations.
- **Il permet la modélisation** du système à toutes les étapes de son cycle de développement et de vie en représentant de manière quasi exhaustive les principaux éléments de modèle suivants :
  – l'expression des besoins et des contraintes ;
  – la représentation de l'organisation structurée des composants ;
  – la définition précise de chaque composant (propriétés structurelles et comportementales) ;
  – la description du comportement attendu du système au cours des différentes phases d'utilisation.
- **Il permet l'intégration** et la mise en relation cohérente des différentes composantes techniques dans un même modèle, par exemple les liaisons entre un programme informatique et des actionneurs mécaniques.
- **Il permet la validation** de solutions par une simulation basée sur des diagrammes paramétriques.

SysML peut être mis en œuvre par des ingénieurs en phase de conception d'un nouveau système pour élaborer soit des modèles normatifs qui guident la construction soit des modèles prédictifs pour valider des solutions en amont de la construction. Mais SysML

[1] Professeur agrégé de génie électrique en STS IRIS au lycée Diderot (75)

s'utilise également pour décrire un système existant avec des modèles cognitifs qui permettront d'analyser et comprendre des constructions – ce qui est le cas le plus fréquent en enseignement secondaire. Il va de soi – mais il est utile de le rappeler – que les démarches d'analyse fonctionnelle (de conception ou d'exploration) que nous évoquons et que nous connaissons ne changent pas avec l'apparition de ce nouvel outil.

Mon but ici n'est pas de décrire de manière séquentielle la syntaxe de chacun des diagrammes SysML, mais plutôt de montrer leur usage dans un contexte, car la modélisation d'un système comporte plusieurs fois le même type de diagrammes dont la signification dépend du point de vue de l'analyse et de son modeleur. En revanche, je montrerai leurs contributions respectives au cours d'une démarche d'analyse d'un système existant en privilégiant le factuel par rapport au normatif. Par ailleurs, dans un souci de clarté, je limiterai mes descriptions et remarques à ce qui me paraît essentiel dans l'étude de cas. Le lecteur trouvera les compléments d'information dans la norme éditée par l'OMG (Object Management Group) ou bien dans l'article de Didier Fagnon et Stéphane Gaston, « SysML : les diagrammes », en page 100 de ce même numéro.

Je me propose de montrer sur un exemple concret comment utiliser de manière cohérente les différents diagrammes spécialisés en fonction des contextes et des aspects qui doivent être décrits. La modélisation a été effectué avec l'outil Case (*Computer Aided Software Engineering*) MagicDraw de la société No Magic, dont le lecteur pourra télécharger librement une version d'essai (voir « Pour aller plus loin » en encadré).

La modélisation de l'Hemomixer

L'Hemomixer est un système d'assistance pour la collecte de sang. Conçu et fabriqué par la société Hemopharm, il est proposé en version didactisée par la société Didastel Provence (voir « Pour en savoir plus » en encadré).

Pour limiter au maximum les erreurs, j'ai choisi d'appliquer pour cet appareil une méthode incrémentale simplifiée, inspirée de celles utilisées en conception (*Unified Process*).

Toutes les informations sur le système ainsi que ses caractéristiques techniques sont puisées dans la notice technique et les documents ressources qui accompagnent le produit didactisé.

Les choix effectués pour l'élaboration des modèles présentés résultent d'un parti pris ; d'autres propositions sont certainement possibles. Au contraire des mathématiques, la voie qui mène à « la solution » n'est pas quasi unique et pavée de difficultés. George E. P. Box ne disait-il pas : « Tous les modèles sont faux, mais certains sont utiles » ?

Le contexte est défini par l'ensemble des acteurs et autres systèmes qui échangent des flux avec l'Hemomixer. Il faut bien sûr préciser les conditions d'utilisation : le contexte correspondant à la phase « maintenance » est différent de celui de la phase « exploitation normale » ou de celui de la phase « exploitation pédagogique ». Nous étudierons ici la phase « exploitation normale ».

Nous procéderons d'abord à une analyse externe du produit, puis, dans un second temps, à son analyse interne.

L'analyse du besoin

L'analyse du besoin a pour objectif de savoir « dans quel but » le produit est construit. Elle constitue une étape fondamentale permettant de poser un problème en termes de finalités, et sert de base pour élaborer le cahier de charges du produit. Même si nous ne sommes pas ici en phase de conception, cette analyse va nous permettre de reconstituer le cahier des charges fonctionnel à partir des fonctionnalités décrites par le constructeur.

La modélisation du contexte d'exploitation normal avec les diagrammes BDD et IBD

Le contexte d'exploitation de l'Hemomixer est défini par l'ensemble des acteurs et autres systèmes situés dans son environnement qui échangent des flux (matière, énergie, information) avec lui. De nature structurelle, il est ici représenté par un diagramme de définition de blocs (BDD, *Block Definition Diagram*) et un diagramme de blocs internes (IBD, *Internal Block Diagram*).

Le BDD définit le contexte composé des acteurs et des systèmes du tableau 1 et du graphe 2 qui entretiennent des relations avec l'Hemomixer 3. L'IBD représente le réseau des connexions qui permet la circulation des flux échangés 4.

Ici, l'infirmière échange des informations (ordres et comptes rendus) avec l'Hemomixer grâce à des interfaces (clavier, écran, voyant) liées à un port standard dont je ne détaillerai pas ici les symboliques. Les connecteurs qui relient les composants sont représentés avec des épaisseurs qui dépendent du flux transporté. Trait fin pour l'information, moyen pour l'énergie, épais pour la matière. Le donneur, qui fait partie du « second cercle » des interacteurs sur la figure 2, est quand même représenté sur le diagramme car il agit (indirectement) sur le système et réciproquement.

Interacteur	Point d'interaction	Nature de l'échange
Poche de sang	Plateau	Énergie (force, mouvement oscillant)
Tuyau	Clampeur	Énergie (force de pincement)
Table support	Pieds	Énergie (force de réaction)
Infirmier	Écran, clavier	Informations
PC	RS 232	Informations

1 Les interacteurs directs

2 Le contexte de l'Hemomixer en exploitation normale

La modélisation fonctionnelle avec le diagramme des cas d'utilisation

Un diagramme des cas d'utilisation (*use case diagram*) répertorie les fonctions d'usage que le système offre à chacun de ses acteurs utilisateurs afin de satisfaire leurs besoins. Il ne doit pas préciser comment il assure ces services.

Pour élaborer le diagramme des cas d'utilisation 5, j'ai simplement relevé les fonctions d'usage dans le descriptif du produit, puis je les ai assemblées.

Un cas d'utilisation représente un service offert par le système à un ou plusieurs acteurs de son environnement. Il est défini par une fonction dans une ellipse reliée à l'acteur concerné.

Si l'exécution d'un service inclus obligatoirement celle d'un sous-service, ce dernier peut être extrait et représenté de manière autonome : il suffira de préciser l'inclusion par une relation orientée (flèche en pointillé) stéréotypée *include*. Cette possibilité permet de factoriser des cas d'utilisation comme « assurer l'interruption du prélèvement », commun à « assurer la sécurité et le confort du donneur » et « prélever une quantité déterminée de sang ».

Le diagramme de contexte qui définit les acteurs bénéficiaires des services du système doit bien sûr, à ce stade, exister. Il sera, au besoin, complété au fur et à mesure de l'étude incrémentale des besoins et de l'apparition éventuelle de nouveaux acteurs…

Il faut aussi se limiter aux fonctions essentielles en évitant une décomposition trop fine. Le diagramme sert de base à la compréhension et à la conception du système. Les détails fonctionnels plus fins seront décrits dans le diagramme des exigences.

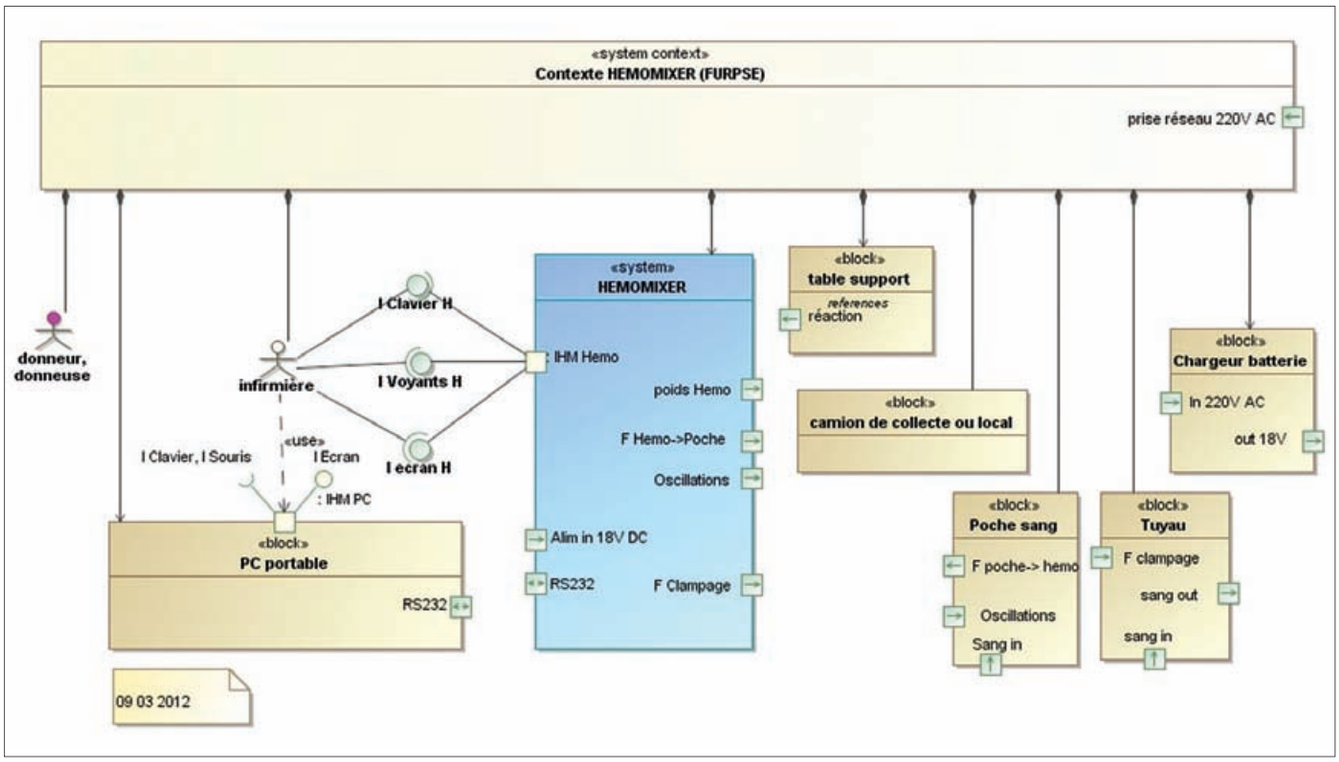
Attention :

- Les cas d'utilisation définissent ce que le système doit faire, et non comment il le fait. Ils représentent le comportement fonctionnel externe du système. Il faut respecter ici les règles d'objectivité – que nous connaissons déjà – qui prévalent dans l'analyse fonctionnelle et l'analyse systémique : définir ce que l'on cherche en termes de finalités, sans a-priori de solutions, et placer l'objet dans son environnement d'utilisation pour exprimer les buts des relations qu'il y crée.
- Les cas d'utilisation ne représentent pas ce que les acteurs doivent faire, par exemple « appuyer sur un bouton », « ouvrir une porte », « consommer de l'énergie », qui ne sont pas des services rendus. On peut, à la limite, définir les fonctions que l'acteur peut réaliser grâce au système, mais toujours en termes de services rendus, par exemple « conduire » une voiture.

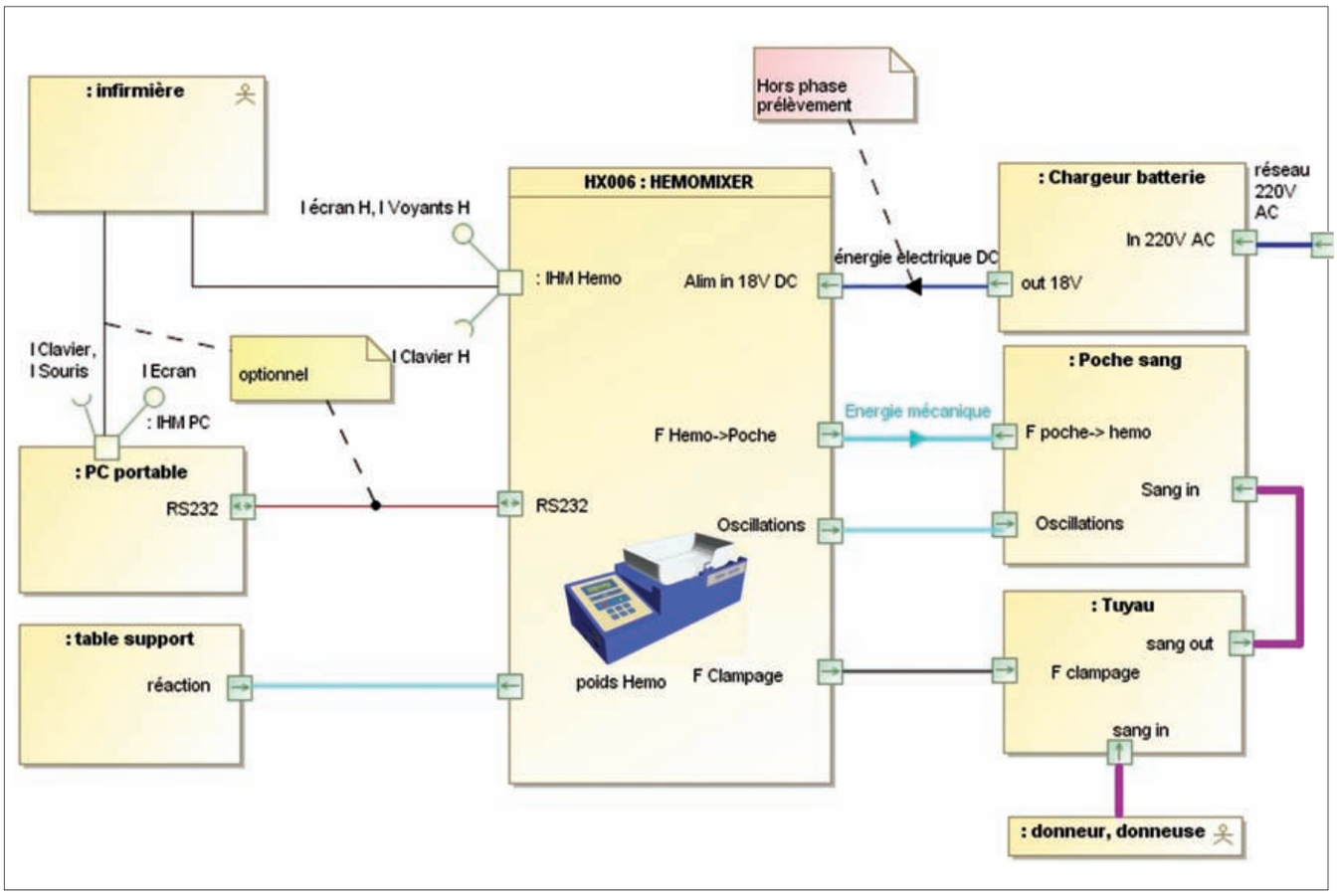
Les compléments à la modélisation fonctionnelle avec le diagramme des exigences

Un diagramme des exigences (*requirement diagram*) répertorie en les classant les affinements des fonctions d'usage et les différentes contraintes et conditions qui doivent être respectées par le système afin qu'il puisse fonctionner correctement mais qui ne sont pas des buts principaux.

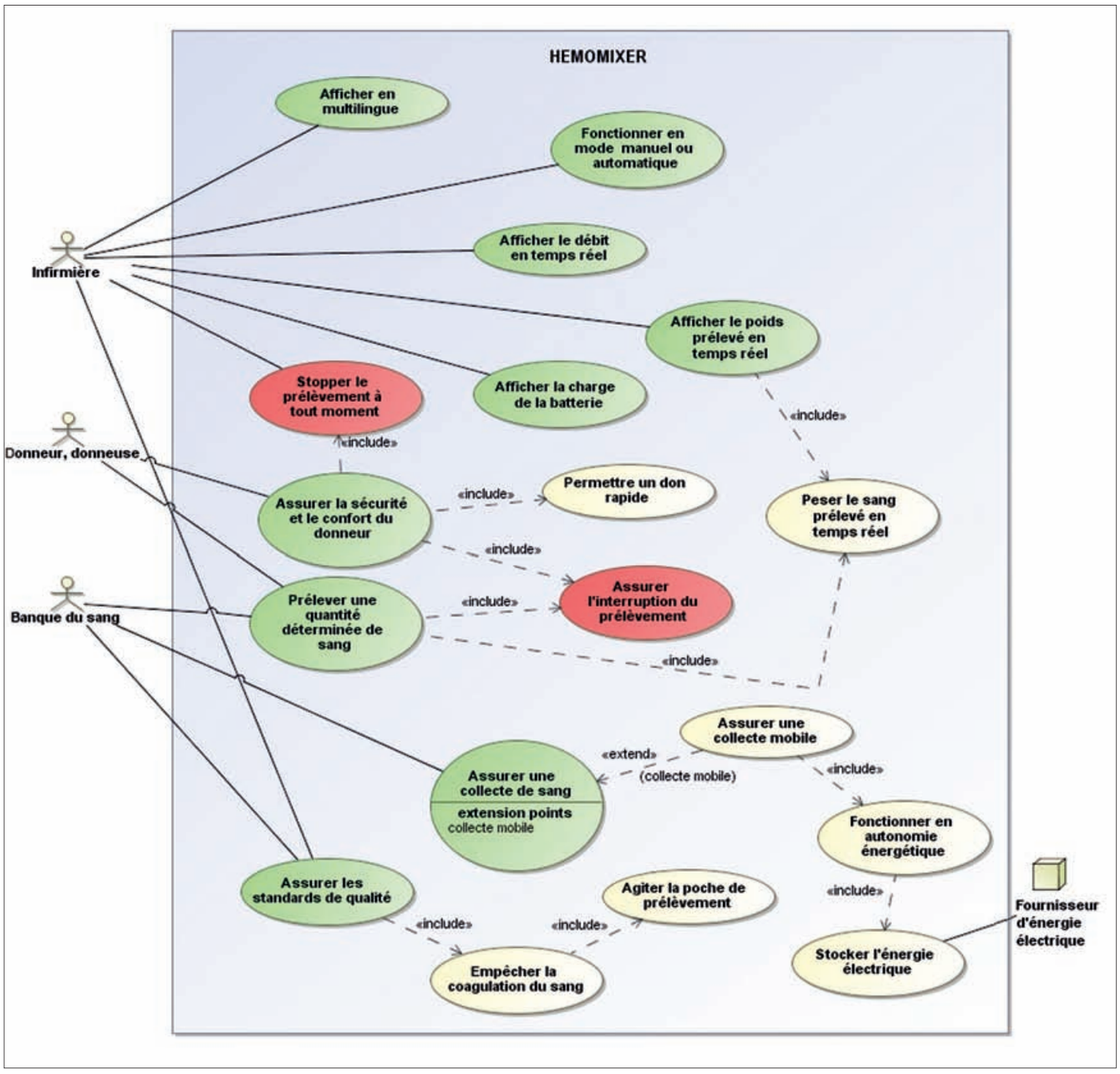
Le système doit vérifier des exigences multiples qui décrivent principalement les conditions de fonctionnement et d'utilisation, les normes à respecter, les exigences de recyclage en fin de vie, les composants et matériels imposés par le client, les caractéristiques physiques imposées et les niveaux de performance attendus. Il faut aussi apporter les précisions nécessaires à la définition complète des fonctions d'usage définies dans le diagramme des cas d'utilisation. On élabore alors le diagramme des exigences. Précisons que ce travail est très long et coûteux, qu'il fait appel à des experts et est souvent sujet à désaccords.



3 Le BDD du contexte de l'Hemomixer en exploitation normale



4 L'IBD du contexte de l'Hemomixer en exploitation normale



5 Le diagramme des cas d'utilisation

On peut relever les exigences sur le descriptif technique du produit, et les classer en trois familles :

- Les exigences fonctionnelles **6**
- Les contraintes technologiques **7**
- Les contraintes opératoires **8**

Les exigences sont représentées sous forme textuelle et structurées de manière arborescente.

Pour assurer la traçabilité et le suivi des exigences tout au long de la modélisation, des relations permettent de préciser les éléments du système concernés par les exigences.

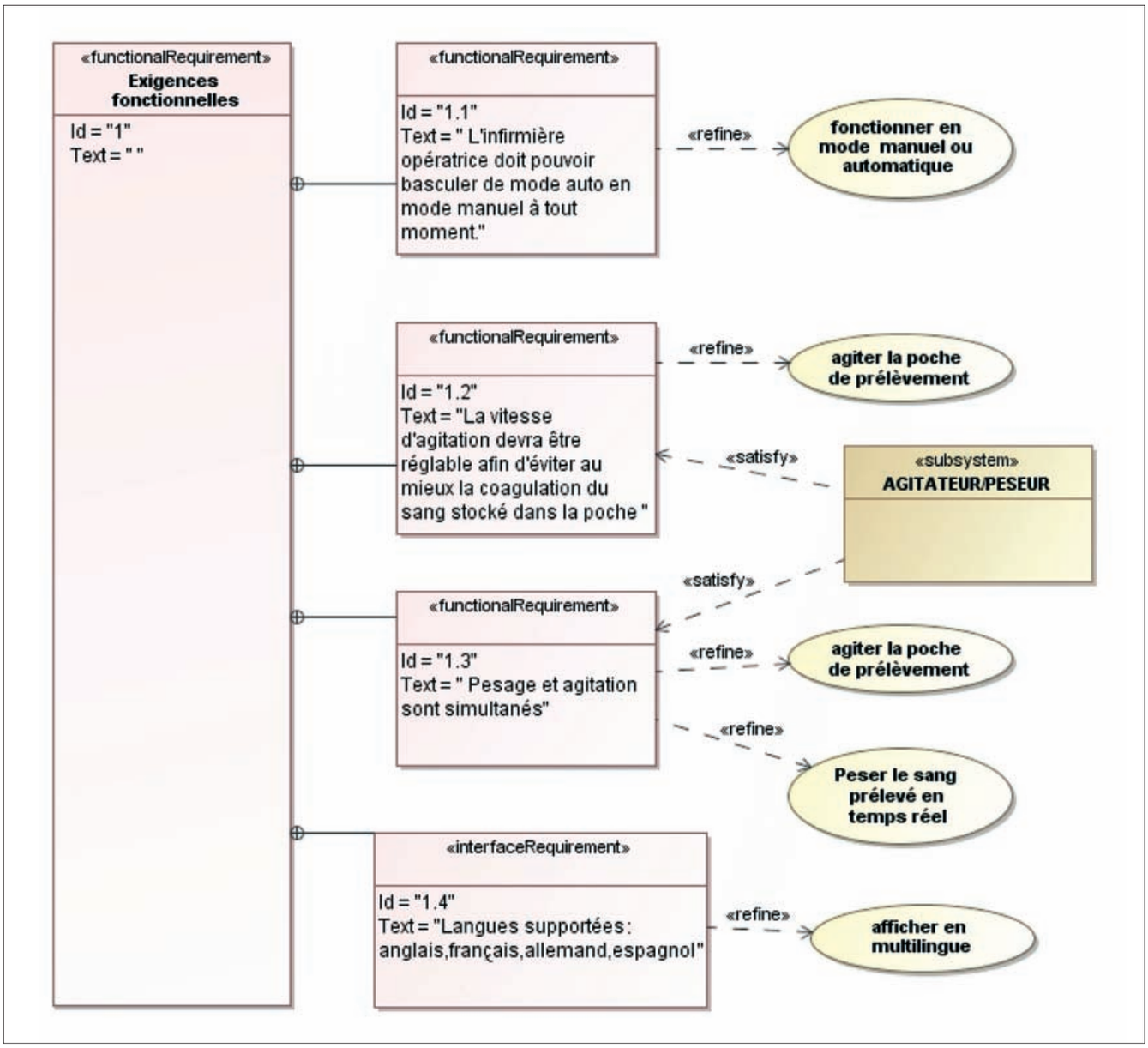
Les exigences de ce type de diagramme apportent des précisions (relation *refine*) aux cas d'utilisation définis en **5**. Par exemple, le cas d'utilisation « fonctionner en mode manuel ou automatique » s'affine avec l'exigence « l'infirmière doit pouvoir commuter sur un des deux modes à tout moment ».

Le sous-système « agitateur-peseur » qui sera défini ultérieurement apparaît dans ce diagramme parce qu'il satisfait (relation *satisfy*) l'exigence « pesage et agitation simultanées ».

SysML définit ainsi plusieurs types d'associations (liens de dépendance stéréotypés) :

Derive : une exigence est dérivée d'une exigence.

Satisfy : un élément du modèle (par exemple un bloc) permet de satisfaire une exigence.



6 **Le diagramme des exigences fonctionnelles**

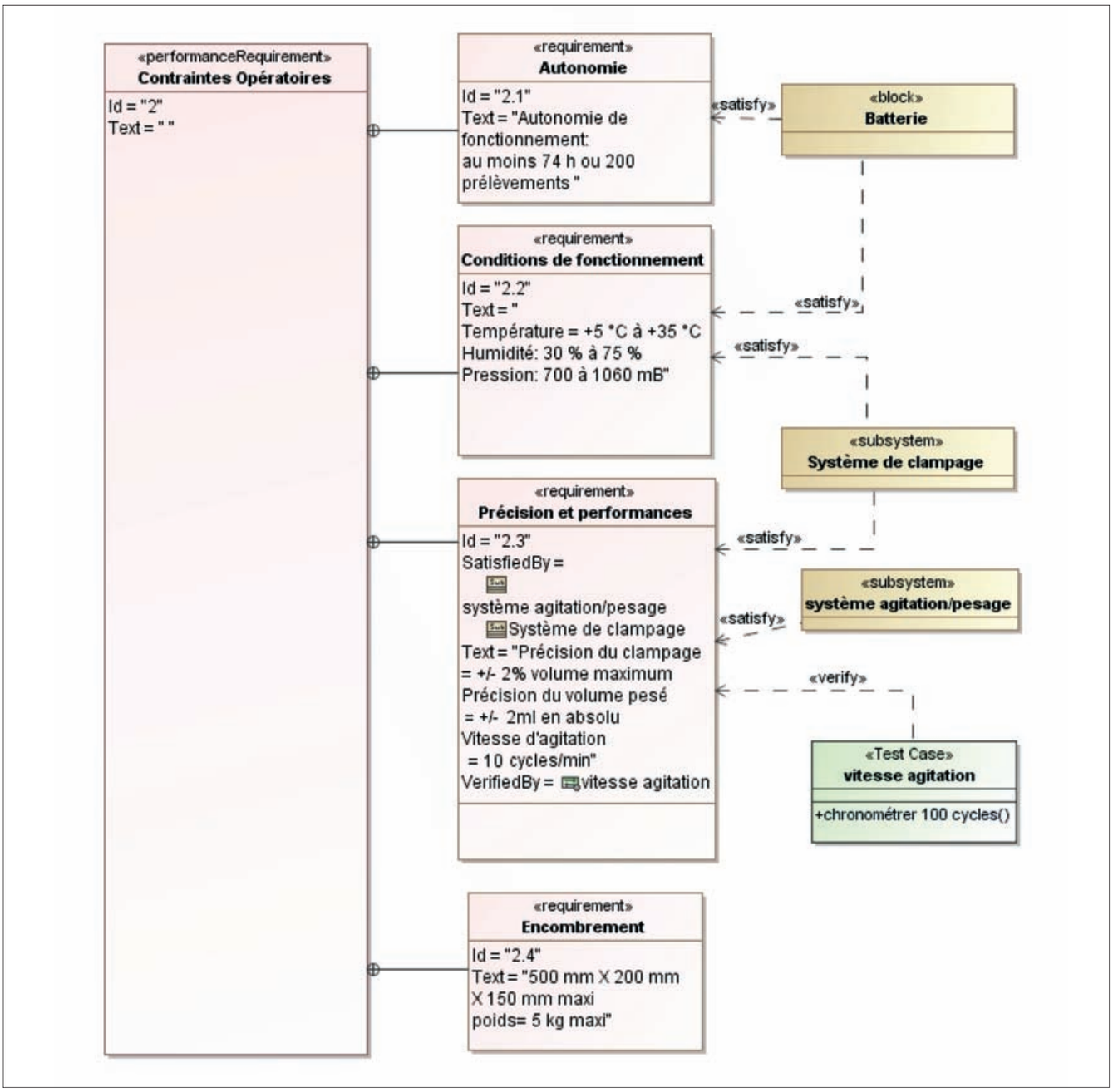
Verify : un élément du modèle (par exemple un *test case*) permet de vérifier et valider une exigence.
Refine : un élément du modèle, par exemple un cas d'utilisation, affine une exigence.

La modélisation du comportement avec le diagramme de séquence

Un diagramme de séquence (*sequence diagram*) représente les interactions classées selon un ordre chronologique des acteurs et du système, ou des parties internes du système au cours du déroulement d'un scénario. En effet, pour atteindre ses objectifs, un système ne reste pas inerte, il agit en déclenchant des suites d'activités dont le résultat observable constitue son *comportement*. Le comportement produit des flux sur les ports de sortie. Le système répond à des sollicitations de son environnement. Il consomme des flux sur les ports d'entrée.

Dans la phase de saisie des besoins, le concepteur devra spécifier les comportements attendus du système lorsqu'il assure les fonctions d'usage définies dans le *use case*. Le système dont on n'est pas censé connaître encore la structure est considéré ici comme une « boîte noire » dont les comportements seront représentés sous forme d'une suite séquentielle d'échanges avec les acteurs dans des diagrammes de séquence **9** nommés ici diagrammes de séquence système ou encore *black box sequence diagrams*.

Un diagramme de séquence représente un déroulement des interactions du système et des acteurs



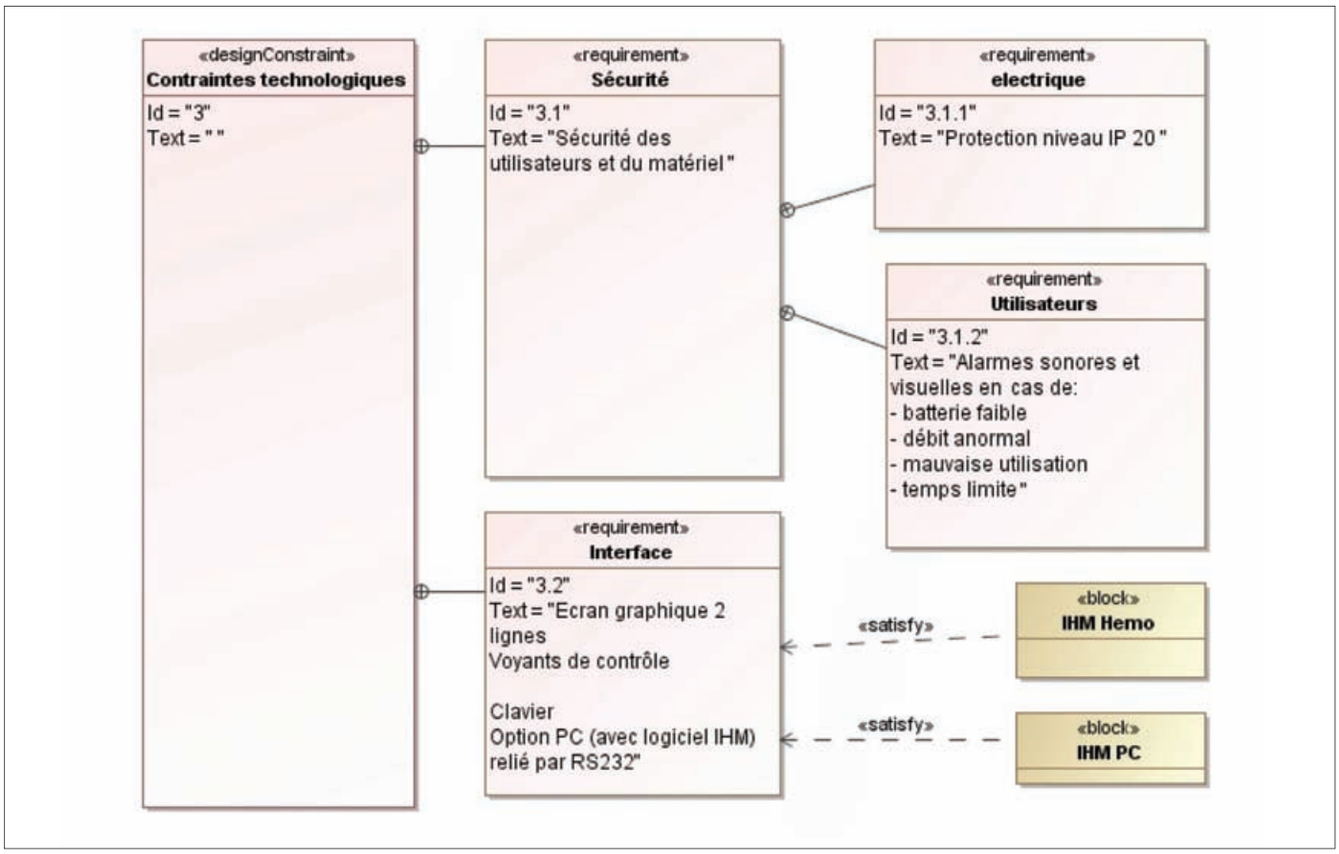
7 Le diagramme des exigences technologiques (contraintes)

concernés au cours de la réalisation d'un service, ici « prélever une quantité déterminée de sang en mode automatique ».

Des objets représentent les acteurs concernés et la boîte noire système. Le temps s'écoule vers le bas du diagramme le long des lignes de vie associées à chaque acteur. Les interactions sont représentées par des échanges de « messages » symbolisés par des traits fléchés horizontaux tirés entre les lignes de vie des correspondants. Un message est un élément de communication dont l'occurrence déclenche une activité chez le récepteur. Il apporte éventuellement avec lui une ou plusieurs données. Les rectangles ajoutés sur les lignes de vie représentent les périodes d'activité de l'acteur ou du système correspondants.

Attention :

- Le comportement (« normal ») décrit ici peut comporter des variantes. Dans ce cas, d'autres diagrammes de séquence exposeront les scénarios alternatifs.
- Il y a autant de diagrammes de séquence que de cas d'utilisation, et, pour un cas donné, autant de diagrammes que de scénarios possibles. Leur nombre devenant vite pléthorique, il faut se limiter au strict



8 **Le diagramme des exigences opératoires (contraintes)**

nécessaire, c'est-à-dire aux cas complexes sujets à des interprétations ambiguës.

- Les systèmes physiques sont de nature causale, c'est-à-dire que leurs réactions sont postérieures aux *stimuli* provocateurs. Les messages de réponse seront donc toujours postérieurs aux messages de sollicitation.
- Le nom d'un message représente l'action que le récepteur doit exécuter lorsqu'il reçoit le message.

☞ *Remarques :*

- Dans cette phase de spécification, le diagramme de séquence représente un comportement externe du système qui réagit aux sollicitations de son environnement. Il sera aussi très largement utilisé, par la suite, pour décrire les suites d'échanges de messages entre les parties internes du système durant des séquences temporelles d'actions appelées activités. C'est alors une vision *white box* qui sert de spécification pour réaliser les programmes de contrôle-commande du système.
- Le diagramme de séquence qui définit le besoin est souvent élaboré avec le client.

Après avoir mené l'analyse fonctionnelle externe du produit (dans sa phase d'exploitation normale), on peut passer à une étude interne du système. Elle procède par itération et en faisant des allers-retours entre les diagrammes. C'est le cœur même de la méthode. Car, bien évidemment, tous les diagrammes sont liés entre eux.

L'analyse technique de l'Hemomixer

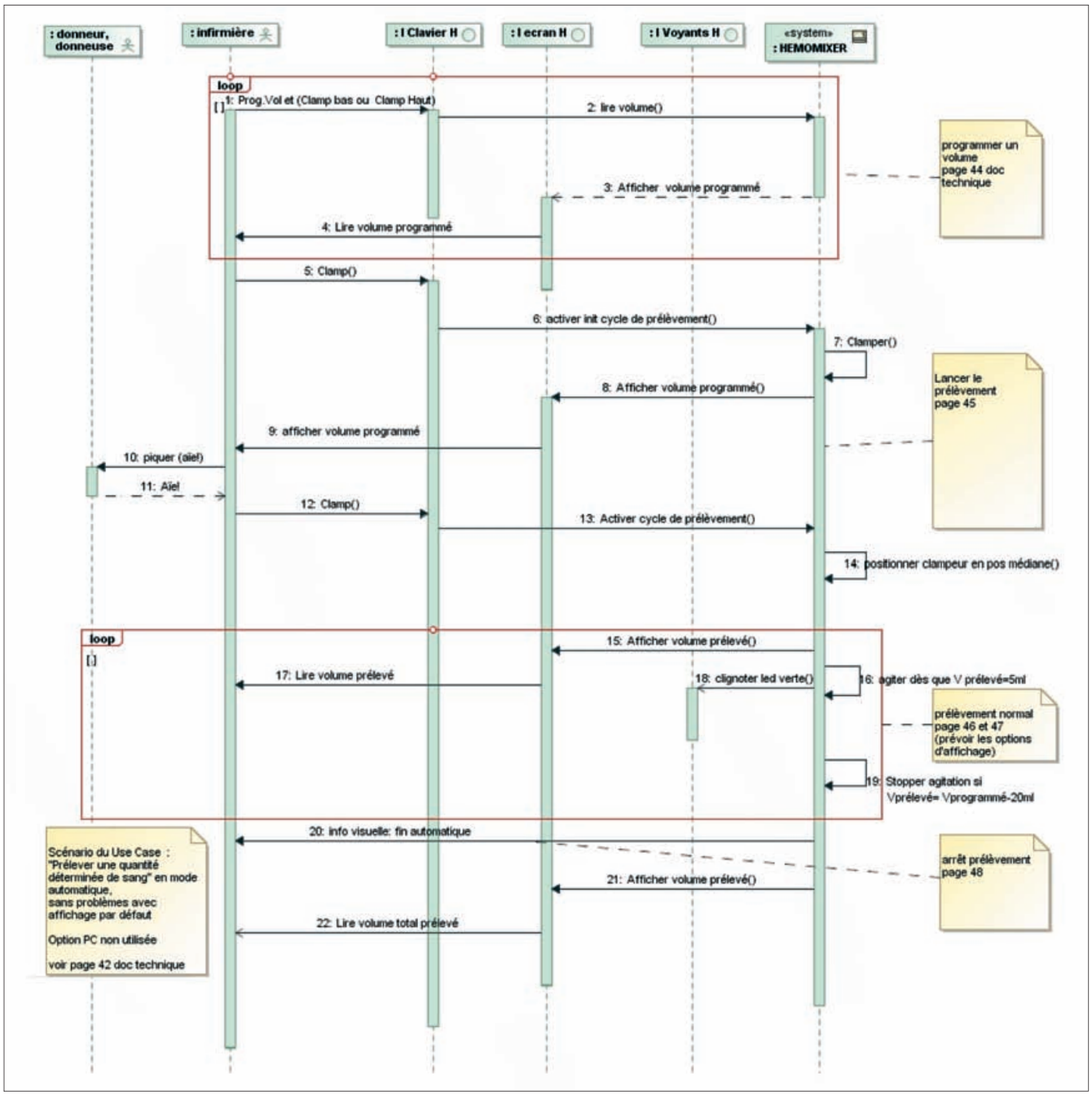
L'existence du système autorise une analyse descendante interne pour le décrire en observant la manière dont il est construit. On tâchera de construire nos diagrammes en les justifiant et en explicitant leurs liens avec les autres éléments du modèle global déjà définis.

La modélisation structurelle avec les BDD et IBD

L'étude des plans de l'Hemomixer, des animations multimédias interactives fournies avec ce dernier et son observation directe permettent d'élaborer un BDD **10** qui traduit son architecture globale, puis l'IBD correspondant **11**.

Guidé par les cas d'utilisation, le concepteur a imaginé cette architecture où les constituants assurent chacun une ou plusieurs fonctions d'usage : l'élaboration de ces diagrammes est orientée « objet » et gouvernée par le « fonctionnel ».

Le BDD représente un modèle générique de définition. Il traduit fidèlement la structure arborescente du système composé récursivement de sous-systèmes. Chaque type de composants, matériel ou logiciel,



9 Le diagramme de séquence « système »

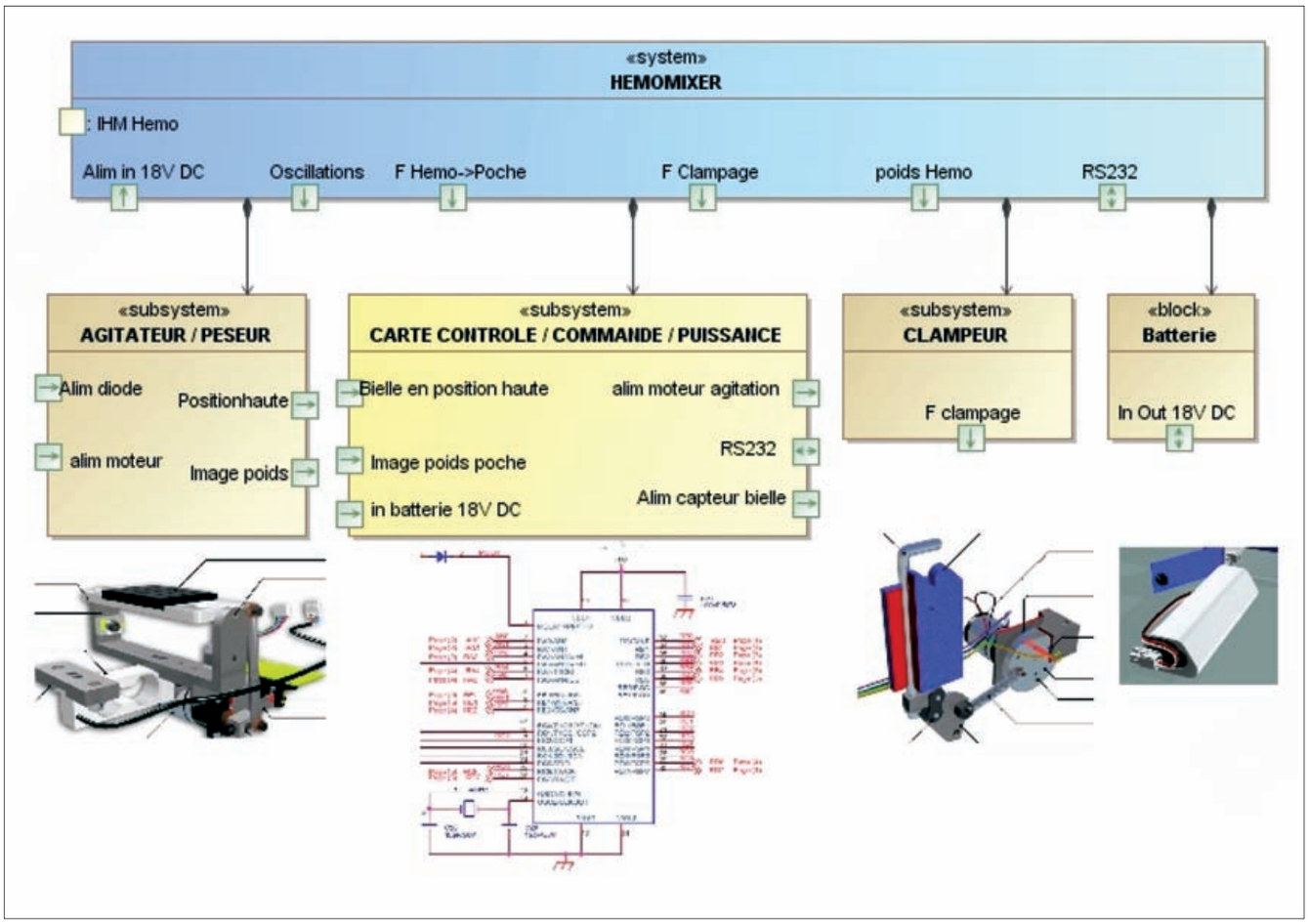
est représenté par un bloc qui encapsule toutes ses composantes caractéristiques regroupées en familles stockées dans quatre « compartiments » : partie (*part property*), référence (*shared property*), données (*value property*) et contraintes (*constraint property*)

Les ports de flux (*flow ports*) situés sur la frontière d'un bloc permettent la circulation de flux (*item flow*) de matière, d'énergie ou d'information entre le bloc et l'extérieur.

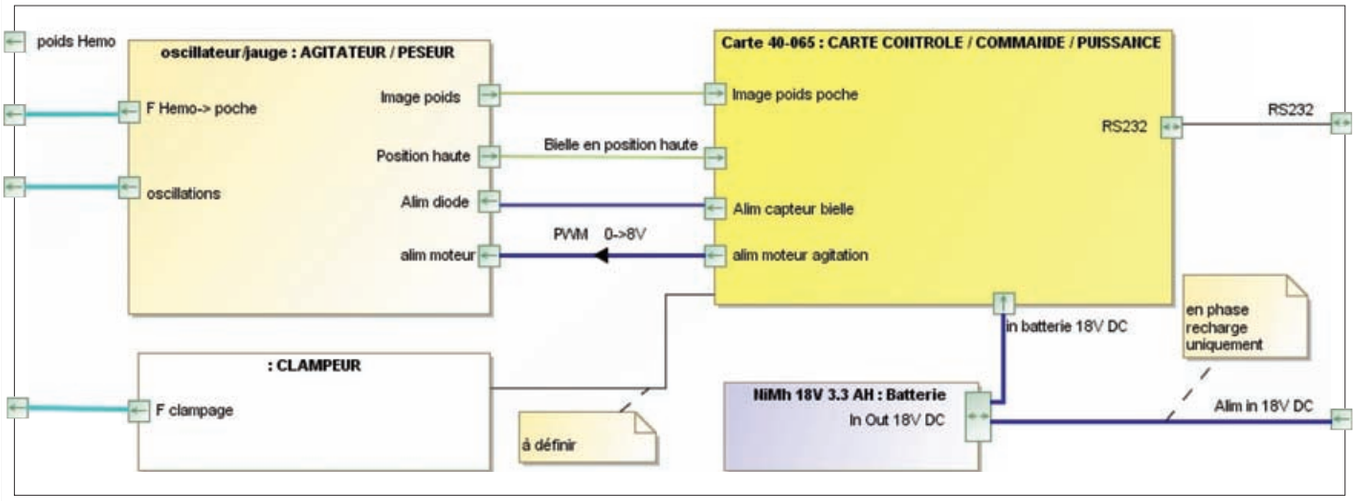
Si plusieurs composants sont quasi identiques et qu'ils sont de la même famille, alors un seul bloc suffit à les typer ; il suffit ensuite d'indiquer leur multiplicité par un nombre posé sur le lien de composition.

Un jeu de liens permet de construire l'arbre de la structure par assemblage des constituants ; on utilise principalement le symbole de composition orientée (flèche pleine au losange noir), qui signifie : « est composé de » ou « appartient à ».

On peut considérer que le BDD est similaire à la première page d'une notice de montage d'un meuble, indiquant la liste des éléments et des pièces à assembler avec leurs quantités respectives.



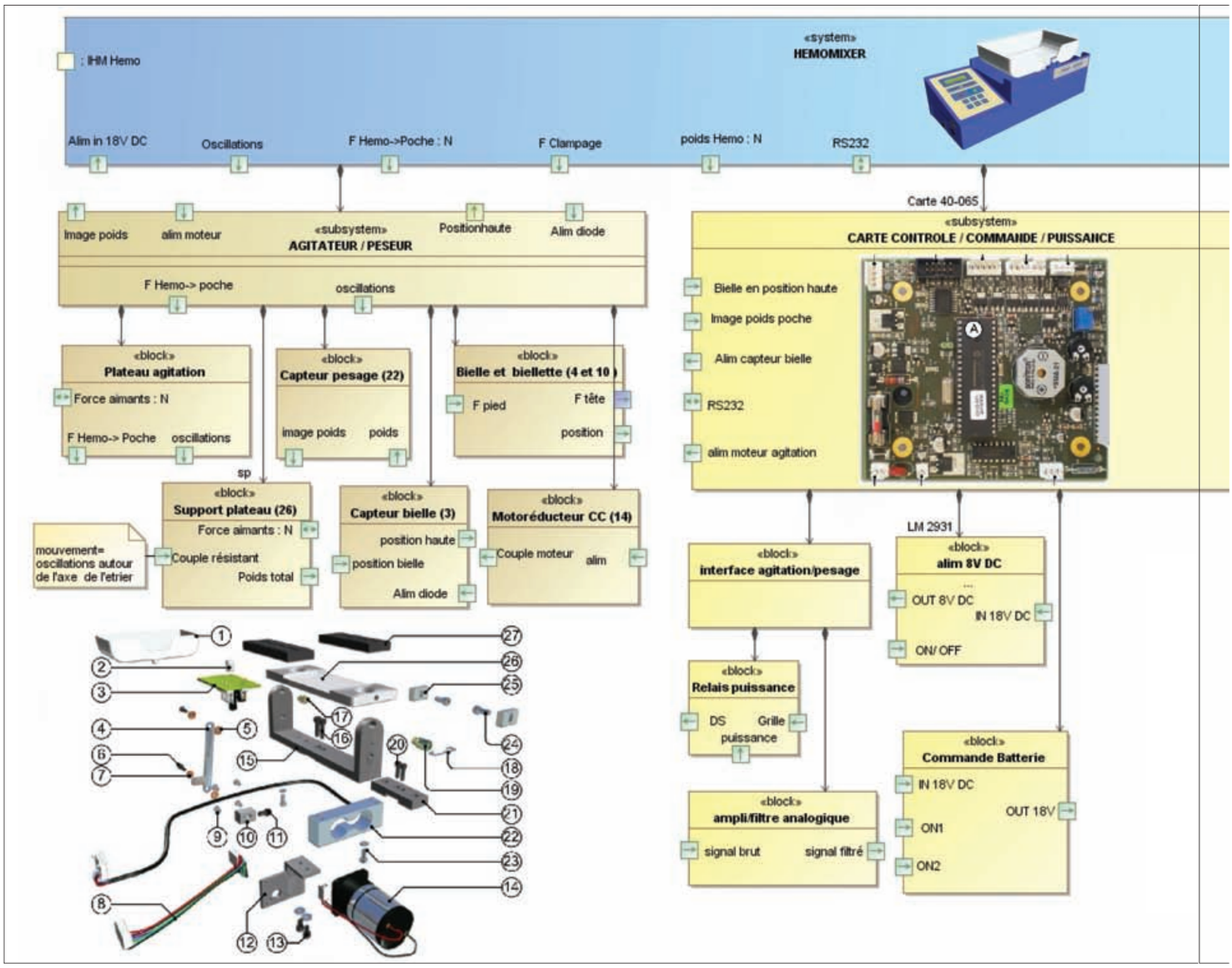
10 **Le BDD global de l'Hemomixer**



11 **L'IBD global de l'Hemomixer (vue « boîtes noires »)**

Attention : La sémantique du modèle est laissée au soin du concepteur, c'est à lui de respecter les règles de cohérence et d'homogénéité en évitant les mélanges de genre.

L'IBD représente l'architecture interne de l'instanciation d'un bloc (système ou sous-système) typé dans un BDD, il met en scène l'organisation de ses composants (*parts*) qui échangent à travers leurs ports reliés par un réseau de *connecteurs* les flux vitaux de matière, d'énergie et d'information afin de faire « vivre » le système. Chaque partie du bloc « radiographié » est engendrée par son bloc de définition décrit dans le



12 **Le BDD global détaillé de l'Hemomixer**

BDD ; elle en possède toutes les propriétés. Son nom propre (par exemple, « carte 40-065 » pour la carte de contrôle-commande) lui confère son statut d'objet concret.

Ce diagramme permet, entre autres choses, de vérifier la continuité des différentes chaînes de flux pour valider le modèle avant de poursuivre l'analyse.

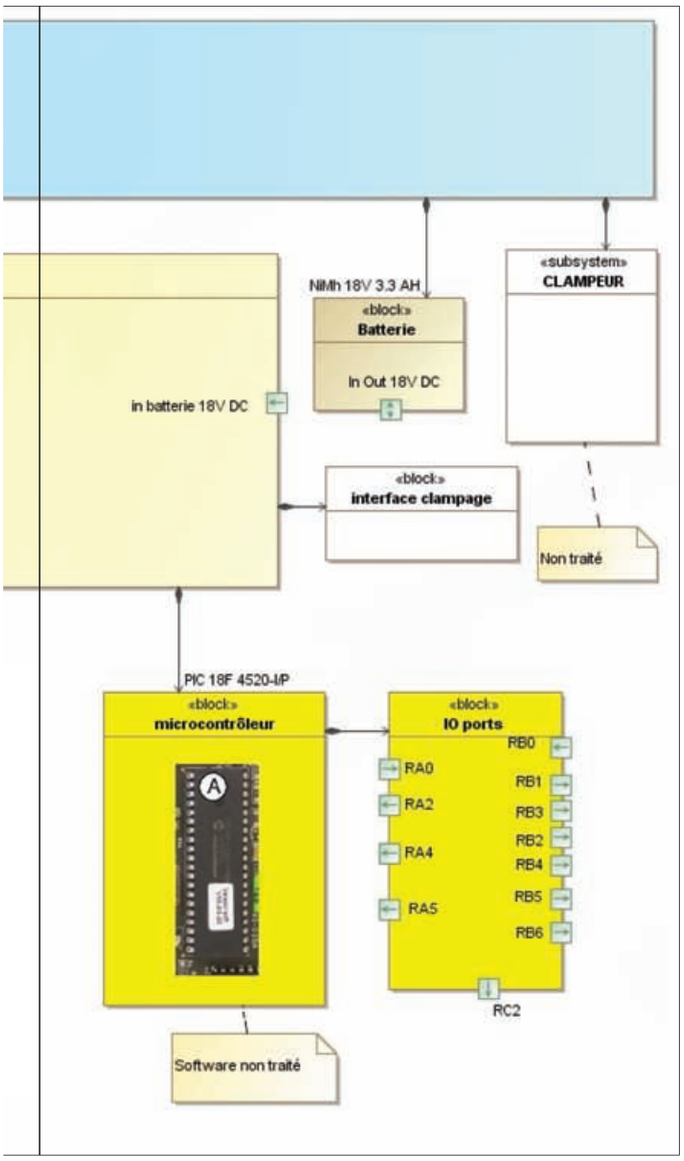
☞ *Remarques :*

- Les bords d'un IBD représentent la frontière du bloc qui est décrit ; les ports de flux qui permettent les entrées/sorties de flux doivent y figurer.
- Il serait judicieux, à ce stade, et d'établir quelques diagrammes de séquence qui représentent les échanges principaux entre la carte de commande et les deux sous-systèmes mécaniques au cours des principaux scénarios.

☞ *Attention :* Il ne faut pas confondre le bloc, qui représente une définition, avec la partie, qui est un objet réel construit sur le modèle du bloc, ce dernier en étant en quelque sorte le moule.

L'analyse se poursuit en affinant les sous-systèmes, que l'on peut représenter soit sur des diagrammes séparés soit sur un même diagramme, comme en **12**, lorsque le système n'est pas trop volumineux.

L'Hemomixer est un système à l'architecture classique : la carte de commande est physiquement séparée de la partie opérative. Sur les 27 composants de la vue éclatée de l'agitateur-peseur fournie dans la documentation technique, notre modèle en retient six. Ce choix est-il approprié ? La décomposition doit mettre en évidence des parties qui assurent chacune



une fonction essentielle au fonctionnement du système, ni plus, ni moins. Le vrai problème du modeleur est de définir le bon niveau de granularité ; la bonne solution n'existant pas, il faut simplement que l'outil permette de corriger et d'affiner si besoin. Ici, rien n'empêche de décider la modélisation complète du motoréducteur ; il suffit de déterminer ses *parts*, leurs interactions, etc. En général, un bloc « motoréducteur » bien renseigné suffit au concepteur… sauf s'il est concepteur de motoréducteurs.

L'absence de description détaillée du clampeur ne bloque pas les travaux sur le reste du système si l'on considère provisoirement le clampeur comme une boîte noire. Il faudra néanmoins rapidement déterminer ses ports de communication. Les noms des parties instanciées apparaissent à l'extrémité des relations de composition.

Pour aller plus loin

Hemomixer

En ligne

Le site du fabricant Hemopharm :

www.hemopharm.fr/fr/produit-idss_2-idc_1-automate-de-prelevement.html

Le site de Didastel Provence, le distributeur du produit didactisé, pour en télécharger la notice :

www.didastel.fr/Produit.php?nom_produit=HMX

En rayon

PÉREZ (Valérie) et BONTEMPS (Philippe) « Du sang neuf pour l'enseignement technologique », *Technologie,* n° 176, nov.-déc. 2011

Sys3ML

En ligne

Le site officiel, pour télécharger la norme :

www.sysml.org/

Un site intéressant avec de nombreuses références sérieuses :

www.uml-sysml.org/

Le site de l'association de promotion de SysML en France, qui se réjouit de l'introduction de SysML dans les programmes de STI2D. Elle met en garde contre les mauvais exemples qui circulent sur le Web et propose sur son blog une initiation à SysML à partir du cas concret du lave-linge :

www.sysml-france.fr

Le site de MagicDraw (AGL de la société No Magic), pour télécharger MagicDraw UML et son plugin SysML avec une licence provisoire de test :

www.magicdraw.com

ROQUES (Pascal), *SysML par l'exemple : Un langage de modélisation pour systèmes complexes,* Eyrolles, 2009, disponible uniquement en version PDF à télécharger :

http://izibook.eyrolles.com

UML

En ligne

L'excellent cours UML de Laurent Audibert :

http://laurent-audibert.developpez.com/Cours-UML/

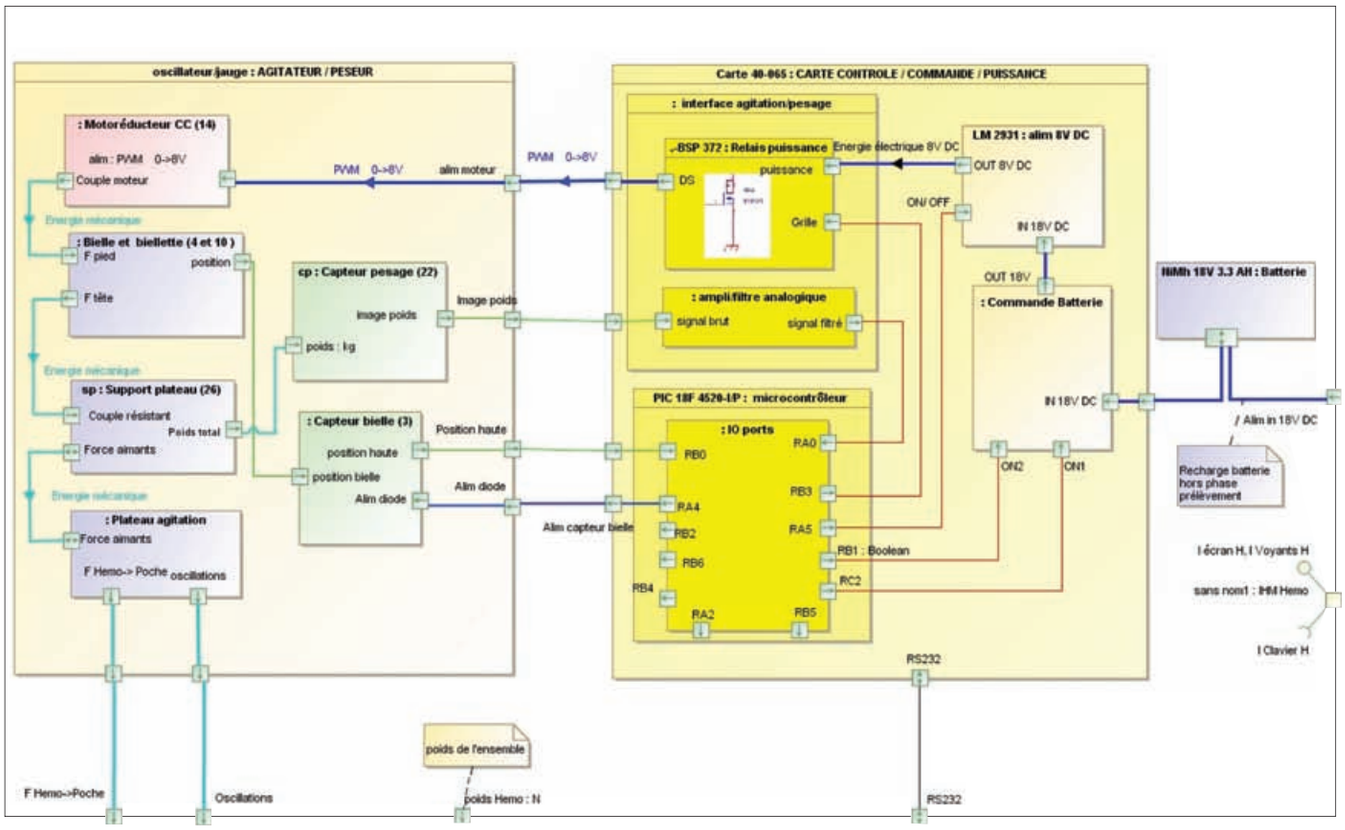
En rayon

L'ouvrage fondateur. Toutes les références sur les diagrammes d'UML, par leurs auteurs eux-mêmes :
BOOCH (Grady), RUMBAUGH (James), JACOBSON (Ivar), *Le Guide de l'utilisateur UML,* Eyrolles, 2000

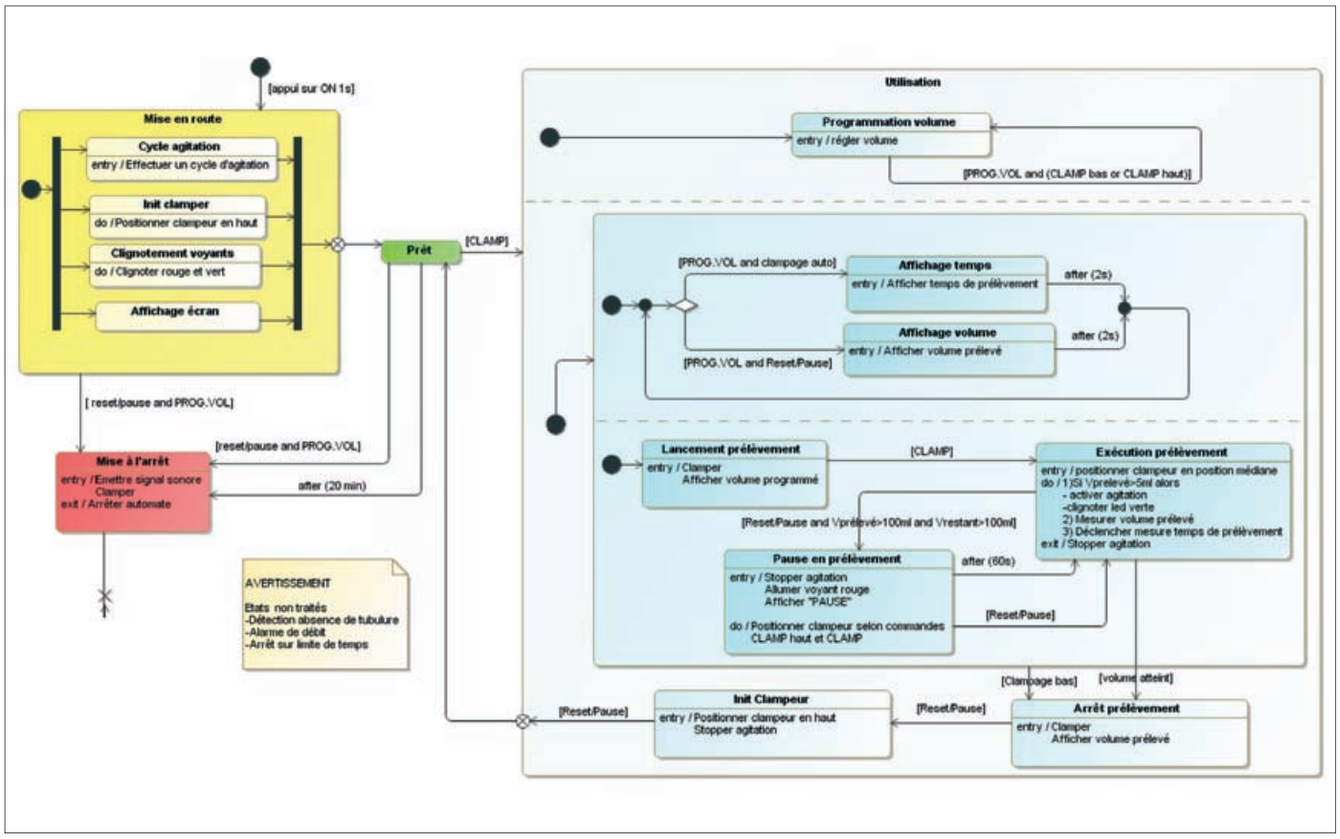
Méthodes

NOLAN (Brian), BROWN (Barclay), BALMELLI (Laurent) *et al., Model Driven System Development with Rational Products,* IBM, coll. « Redbooks », 2008. Développée par IBM, cette méthode moderne de conception guidée par le modèle s'appuie fortement sur SysML. Téléchargement en PDF gratuit :

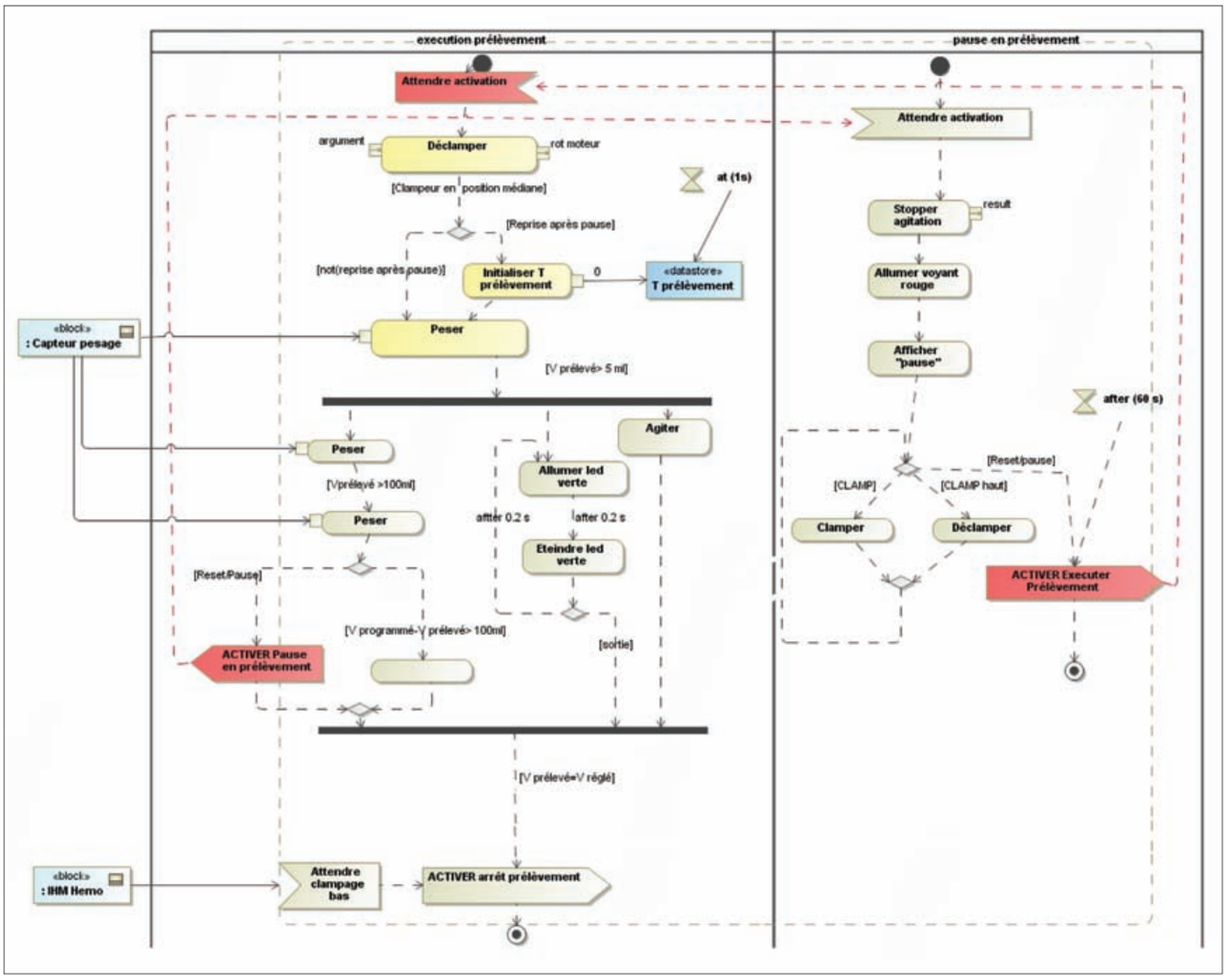
www.redbooks.ibm.com/redbooks/pdfs/sg247368.pdf



13 L'IBD global détaillé de l'Hemomixer (vue « boîtes blanches »)



14 Le diagramme d'état général de l'Hemomixer



15 **Deux activités synchronisées**

Attention : Au cours d'une phase d'analyse, le modeleur doit extraire ses éléments de modélisation exclusivement des plans, des notices techniques et, si possible, de l'observation directe du système.

L'IBD 13, structurellement identique à celui en 11, exprime un niveau plus détaillé de l'Hemomixer. Les composants internes étant maintenant parfaitement définis, les chaînes peuvent être vérifiées complètement. Le clampeur n'est pas représenté sur ce diagramme ; le logiciel de pilotage implémenté dans le microcontrôleur pourrait quant à lui y figurer.

Les composants internes du système étant maintenant déterminés, il faut représenter la manière dont ils interagissent ainsi que les effets de leurs activités collaboratives sur le comportement global du système pour vérifier que ce dernier est conforme aux attentes.

Trois diagrammes SysML permettent de spécifier ces comportements :

- Le diagramme de séquence (déjà élaboré 9)
- Le diagramme d'état
- Le diagramme d'activité (non spécifié dans le document d'accompagnement STI2D)

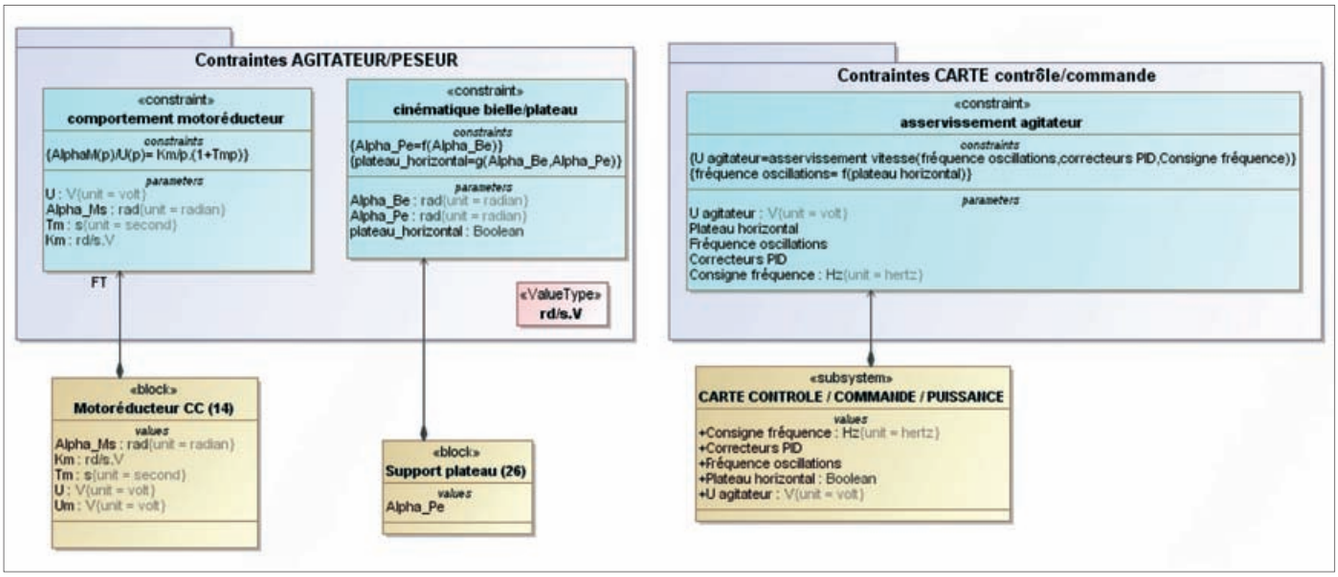
La modélisation du comportement dynamique avec le diagramme d'état

C'est un automate à état fini qui décrit la succession des états du système (ou d'un composant) pendant son utilisation. On considérera qu'un état correspond à des phases reproductibles de la vie du système caractérisées par une invariance d'une partie de ses variables d'état ou de ses activités.

Les différents états de l'Hemomixer identifiés dans la documentation technique figurent sur le diagramme d'état (*state diagram*) 14.

Il s'agit ici d'un diagramme très classique à quatre états : mise en route, prêt, utilisation, et mise à l'arrêt.

Un état est soit actif soit inactif ; l'état initialement actif est symbolisé par un disque noir. Les transitions relient les états entre eux et indiquent les voies d'évolution possibles.



16 **Le BDD préparatoire au diagramme paramétrique**

Un événement ou (inclusif) une garde associée à chaque transition orientée permettent de déclencher le franchissement de la transition lorsque l'état source est actif ; dans ce cas l'état cible devient actif. Des activités, c'est-à-dire des suites d'actions, peuvent être activées soit au moment de l'entrée dans l'état (*entry/*), soit pendant toute la durée de l'état (*do/*), soit enfin au moment de la sortie de l'état (*exit/*).

Un état peut contenir des sous-états, qui sont soumis aux règles d'évolution générale à la condition que l'état conteneur soit actif. Des symboles complémentaires permettent de synchroniser (barres noires) ou d'assurer l'indépendance de sous-états (état conteneur composite : séparé en zones parallèles par un segment pointillé).

Ce diagramme définit complètement les différentes phases de fonctionnement du composant.

Il servira principalement à la conception générale des logiciels de contrôle-commande, et en particulier à définir leur architecture globale, en général de nature multitâche. Mais il peut aussi servir à définir un mode d'emploi – c'est d'ailleurs à partir du mode d'emploi détaillé qu'on l'a élaboré. Les automaticiens pourront, le cas échéant, représenter un Gemma (Guide d'Étude des Modes de Marche et d'Arrêt) sans aucun problème, car il représente une instance particulière du graphe d'état.

Attention : Les diagrammes de séquence ne conviennent pas pour décrire des activités. Ces dernières sont associées aux états en tant qu'éléments moteurs générateurs d'événements qui provoquent des changements d'état.

Remarque : Les diagrammes de séquence pourront tracer sur ce support balisé des parcours qui représenteront chacun un scénario de fonctionnement. À ce titre, le diagramme de séquence **9** représente le déroulement du cas d'utilisation « prélever une quantité déterminée de sang en mode automatique » dans le cas où tout se passe bien et avec affichage par défaut ; le lecteur pourra suivre ce parcours sur le diagramme d'état. D'autres diagrammes de séquence, par exemple dans le cas où le donneur demande une pause pendant le prélèvement, compléteront utilement l'analyse dynamique, qui est ici simplement ébauchée. C'est au modeleur de maintenir la cohérence entre les diagrammes qui devront être corrélés en permanence tout au long de la modélisation.

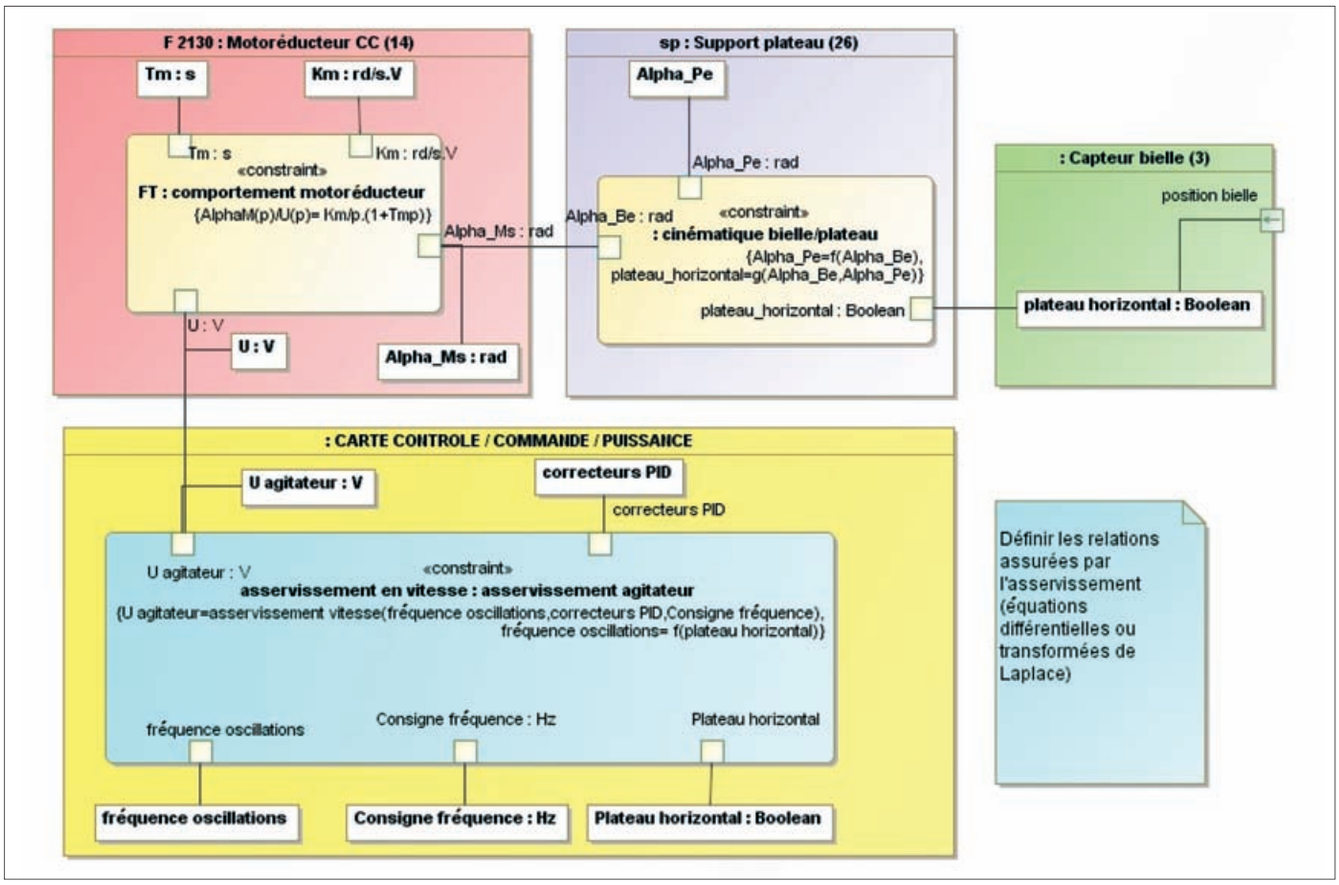
La modélisation du comportement dynamique avec le diagramme d'activité

Non spécifié dans le document officiel d'accompagnement STI2D, le diagramme d'activité (*activity diagram*) est un automate à états finis qui permet de décrire une activité en représentant le séquencement temporel des actions qui la composent. Il est suréquipé en éléments syntaxiques qui lui permettent de traiter les événements, spécifier les flux d'activation ainsi que les flux de traitement des données.

Le diagramme d'activité **15** représente deux activités synchronisées par des événements et qui s'exécutent pendant les activités des états « exécution prélèvement » et « pause en prélèvement » du diagramme des états généraux **12**.

Cousin du Grafcet, le diagramme d'activité est toutefois plus riche que ce dernier.

Les deux activités sont représentées dans des couloirs différents (*swimlanes*) qui portent leur nom. Les couloirs permettent de définir des partitions (bloc, activité, états) auxquelles seront allouées les suites d'opérations. Ce dispositif qui permet de lier des éléments appartenant à des diagrammes différents (un processeur et son programme applicatif, par exemple) apporte de la cohérence au modèle.



17 Le diagramme paramétrique

Les éléments syntaxiques, très nombreux, se classent en trois groupes : les nœuds d'actions qui décrivent les actions élémentaires (par exemple, *Déclamper*), les nœuds d'objets qui permettent de faire transiter et de stocker les données d'objets (*flow*) traités par les nœuds d'action (par exemple, *T prélèvement : « datastore »*), enfin les nœuds de contrôle, qui définissent le déroulement des traitements (le losange de choix) et leur synchronisation (barre).

Des nœuds d'action particuliers permettent l'échange de signaux d'événements entre les activités.

Les blocs producteurs et consommateurs des données traitées qui figurent sur le diagramme apportent leur part de cohérence au modèle en facilitant la compréhension globale des processus. Le diagramme indique, par exemple, que le capteur de pesage défini en **12** et **13** fournit de l'information à l'action « peser ».

Ces diagrammes seront directement traduits en tâches logicielles par le programmeur. Nous avons là un vrai outil de spécification avant codage de la commande.

La modélisation comportementale avec le diagramme paramétrique

Même si ce diagramme n'est pas évoqué dans le document d'accompagnement STI2D, il est intéressant de montrer ses possibilités. Le diagramme paramétrique (*parametric diagram*) est un diagramme de structure qui sert à exprimer les relations qui lient les grandeurs physiques caractéristiques du système. Il se construit en deux étapes, que nous allons suivre pour représenter les phénomènes mécaniques associés à la fonction d'agitation :

❶ Définition des paramètres physique et des équations qui les lient **16**

Les relations physique sont définies dans un BDD par des blocs stéréotypés *« constraint »*, qui sont des *parts* des blocs (système physiques) qu'ils paramètrent.

Les relations peuvent s'exprimer soit en langage « libre », soit en langage OCL (*Object Contraint Language*), soit dans le langage imposé par l'outil Case si celui-ci assure la simulation, donc l'interprétation, des relations.

❷ Élaboration du diagramme paramétrique **17**

Le principe est analogue à celui utilisé pour l'élaboration d'un IBD. Les contraintes définies à l'étape ❶ sont instanciées ; on a choisi de les ranger dans les instances des composants dont elles décrivent les « contraintes ». Les paramètres des relations sont des attributs (*« value »*) des composants correspondants : ils représentent soit des données constantes (par exemple *Tm* signifie

Ce qu'il faut retenir des 6 diagrammes étudiés en STI2D

BDD : Un diagramme de définition de blocs représente une classification arborescente des types de composants d'un système. Chaque famille de composants est représentée par un bloc qui encapsule toutes ses propriétés. C'est un diagramme de description structurelle.
IBD : Un diagramme de blocs internes représente l'architecture interne de l'instanciation d'un bloc (système ou sous-système) typé dans un BDD, il met en scène l'organisation des composants qui échangent les flux vitaux de matière, d'énergie et d'information pour faire « vivre » le système. C'est un diagramme de description structurelle.
Use case diagram : Un diagramme des cas d'utilisation répertorie les fonctions d'usage que le système offre à chacun de ses acteurs utilisateurs afin de satisfaire leurs besoins. Il ne doit pas préciser comment il assure ces services. C'est un diagramme de spécification fonctionnelle.
Requirement diagram : Un diagramme des exigences répertorie en les classant les affinements des fonctions d'usage et les différentes contraintes et conditions qui doivent être respectées par le système afin qu'il puisse fonctionner correctement mais qui ne sont pas des buts principaux. Il faut prendre en compte les exigences fonctionnelles, les contraintes technologiques et les contraintes opératoires. C'est un diagramme de spécification générale.
Sequence diagram : Un diagramme de séquence représente les interactions classées selon un ordre chronologique soit des acteurs et du système (boîte noire) soit des parties internes du système (boîte blanche) au cours du déroulement d'un scénario. Il sert principalement à l'élaboration des programmes de contrôle/commande. Il donne une description comportementale.
State diagram : Un diagramme d'état représente une situation durant la vie d'un bloc. Un état peut contenir des sous-états qui sont soumis aux règles d'évolution générale à la condition supplémentaire que l'état conteneur soit actif. Ce diagramme définit complètement les différentes phases de fonctionnement du composant. Il donne une description comportementale.

« constante de temps du moteur ») soit des variables qui caractérisent les phénomènes physiques mis en œuvre au cours du fonctionnement.

Attention :
– Il s'agit bien d'un diagramme de structure. Les relations mathématiques entre paramètres ne sont pas orientées, il n'est pas question ici d'entrées/sorties ;
– Il ne faut pas confondre avec un IBD, les connecteurs du diagramme paramétrique ne transportent pas de flux, ce sont des connections logiques qui indiquent des équivalences entre des valeurs de paramètres.

Remarque : Ce diagramme permet de valider des choix de conception en simulant directement les équations lorsque l'outil le permet ou en utilisant des logiciels de simulation multiphysique comme Scilab. Notons que le *plugin* Paramagic de MagicDraw assure ce service.

Conclusion

À propos de SysML

SysML permet de modéliser des artefacts existants pour en comprendre le fonctionnement, mais aussi des artefacts inexistants pour les concevoir ; dans ce cas, il couvre les représentations correspondant à toutes les phases de développement.

Il doit rester en prise avec le concret, en harmonie avec tous les principes de la systémique. C'est un outil pour les techniciens et les ingénieurs.

Il permet de représenter de façon cohérente et complète des systèmes tout en maîtrisant la complexité qui en résulte. Il est toutefois évident que c'est le modeleur qui donne du sens au modèle.

SysML est un modèle contemplatif, mais certains AGL (Ateliers de Génie Logiciel) comme MagicDraw intègrent des modules de simulation.

Il garde encore en héritage d'UML des particularités fortement typées « logiciel » qui complexifient inutilement le langage et peuvent nuire à la compréhension. J'ai d'ailleurs ici volontairement simplifié quelques règles, par exemple la distinction entre les *parts* qui sont des instances de rôle plutôt que des instances de bloc, sans pour autant porter atteinte aux principes de lisibilité, simplicité, pertinence et universalité.

Cet article n'est pas exhaustif ; il y aurait encore beaucoup à exposer, comme la notion de réutilisabilité, favorisée par la possibilité de création de bibliothèques de composants (moteurs, capteurs, microcontrôleurs, logiciels, normes…) rangés dans des paquetages et réutilisables à volonté.

Systémique *versus* analytique

L'approche analytique compartimente non seulement les connaissances elles-mêmes, mais, ce qui est plus grave, leur acquisition. Elle satisfait pleinement les « spécialistes purs » qui vivent en harmonie avec eux-mêmes dans leur pré carré, garant de ce que Bourdieu appelait la « distinction ».

L'approche systémique, au contraire, permet de relier progressivement des connaissances multidisciplinaires dans un modèle unique et cohérent qui leur donne du sens en les mettant en relation. Dans cet esprit, les outils d'aide à la conception comme SysML permettent l'intégration de nombreux composants hautement spécialisés et favorisent ainsi le partage entre spécialistes éclairés : c'est non plus « je pense », mais « nous pensons » autour d'un projet commun.

En 1975, Joël de Rosnay, dans son célèbre *Macroscope* (Seuil), exposait dans le détail – avec 40 années d'avance – tous ces grands principes repris aujourd'hui, à la lettre, par les méthodes de l'ingénierie moderne. Les nouveaux programmes de STI2D favorisent l'engagement dans cette voie. Malgré toutes les difficultés qui pavent ce nouveau chemin, je suis persuadé qu'il est celui de la réussite des jeunes générations d'ingénieurs et de techniciens.

Merci de m'avoir lu jusqu'ici ! Et restons bien conscients que SysML n'est qu'un outil, et que ce sont les hommes impliqués dans le projet qui font la différence, non les instruments qu'ils utilisent. ■