Source : ens-cachan

SysML : un langage pour la modélisation des systèmes

La modélisation des systèmes avec le langage SysML est au coeur des nouveaux programmes d'enseignement des Sciences de l'Ingénieur. Cette ressource présente quelques-uns des objectifs de SysML et les principaux concepts à connaître pour comprendre et utiliser ce langage.

1. Introduction

Les nouveaux programmes d'enseignement des Sciences de l'Ingénieur accompagnent deux des évolutions notables de l'ingénierie :

- le caractère *pluritechnologique* de nombreux produits industriels actuels, dont l'étude nécessite des approches transversales mobilisant plusieurs disciplines ;
- le développement de la *simulation* dans la prévision des performances d'un produit.

La mise en oeuvre d'un enseignement transversal nécessite d'utiliser des *descriptions communes* aux différentes disciplines mises en jeu, et le rôle central de la simulation suggère que ces descriptions puissent mener, autant que faire se peut, à des *modèles exécutables* à l'aide de logiciels dédiés. C'est pourquoi les concepteurs des nouveaux programmes ont choisi d'utiliser un unique outil de description adapté à ces contraintes : le *langage SysML* (pour *Systems Modeling Language*, "langage pour la modélisation des systèmes").

Cette ressource présente ce qu'est la modélisation des systèmes, puis met en évidence la façon dont SysML permet de mener à bien cette tâche ; une présentation plus détaillée, incluant les principaux éléments syntaxiques du langage, est téléchargeable en suivant le lien au bas de cette ressource.

2. La modélisation des systèmes

2.1. "Systèmes" : un point de vue systémique

Dans les nouveaux programmes, l'étude des produits industriels est abordée selon une approche systémique. Selon cette approche, un *système* est défini comme un ensemble de composants en interaction ; le comportement du système ne résulte pas seulement des comportements individuels de ses composants, mais aussi et surtout de la façon dont ils interagissent entre eux.

La systémique est une approche complètement transversale, qui possède de nombreux domaines d'utilisation. Son application à la conception et à l'analyse des produits industriels s'appelle l'ingénierie des systèmes. Dans ce cadre, la notion de système s'étend sur plusieurs niveaux : un composant peut très bien être vu comme un système et décomposé à son tour, et on le qualifiera alors de *sous-système* (Figure 1).

Source : ens-cachan

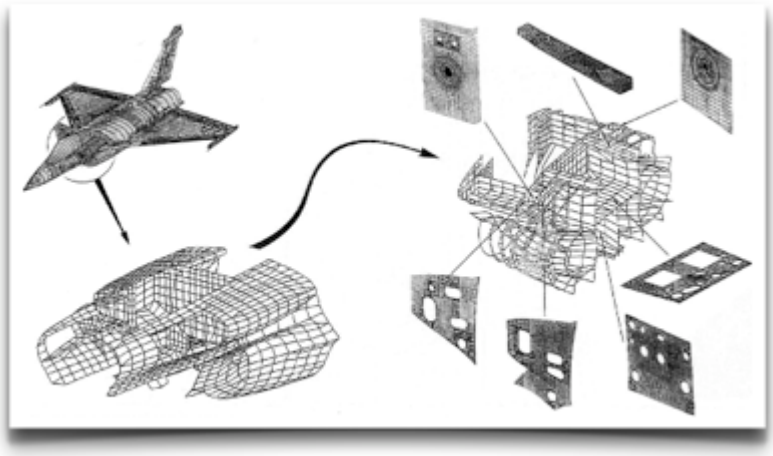

Figure 1 : Illustration de la décomposition d'un produit en composants sur plusieurs niveaux (image de Dassault Aviation).

L'ingénierie des systèmes offre un cadre adapté à l'étude des produits pluritechnologiques, et se prête bien à l'utilisation intensive de la simulation.

2.2. La modélisation : pour simuler et pour communiquer

La *modélisation* des systèmes a deux objectifs principaux : *simuler* leur comportement et *communiquer* des descriptions. En Sciences de l'Ingénieur, elle peut s'effectuer selon trois grands points de vue complémentaires (Figure 2) :

1. le point de vue *fonctionnel*, qui consiste à décrire les actions effectuées par le système pour répondre à la question *"A quoi sert-il ?"* ;
2. le point de vue *structurel* qui, dans le cadre de la systémique, consiste à décrire les composants du produit et de son environnement ainsi que les relations entre ces composants, pour répondre aux questions *"De quoi est-il composé ?"* et *"Comment est-il organisé ?"* ;
3. le point de vue *comportemental*, qui consiste à modéliser le produit et son environnement au sein d'une théorie afin de répondre, par la simulation, à la question *"Quelles sont ses performances ?"*.

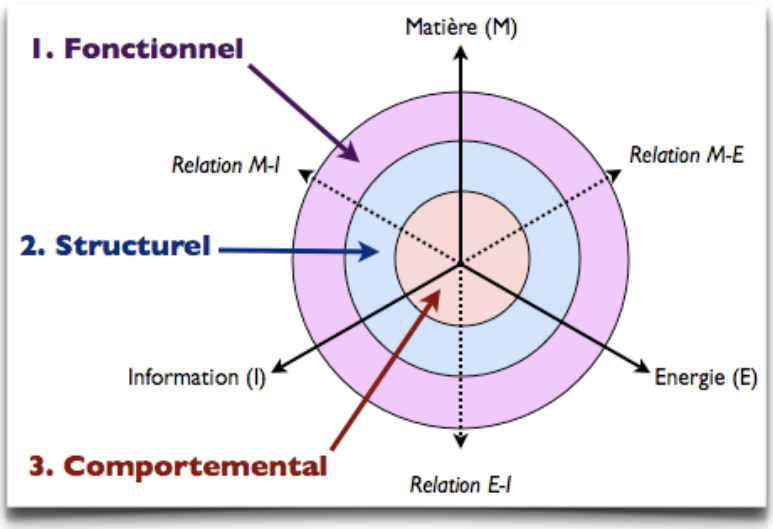

Figure 2 : Les trois grands points de vue des Sciences de l'Ingénieur : fonctionnel, structurel et comportemental.

Source : ens-cachan

Une particularité de SysML est de regrouper ces trois familles de représentations au sein d'un unique modèle *"multi-points de vue"*. Ce modèle est généralement construit à l'aide d'un logiciel dédié qui le stocke dans une base de données, et peut ensuite générer automatiquement un document correspondant à un point de vue particulier. Une telle approche est dite *basée sur un modèle*, par opposition aux approches *basées sur des documents* qui reposent sur une collection de modèles *"mono-point de vue"* disjoints. L'utilisation d'un modèle unique présente deux avantages notables :

- cela assure la *cohérence* des données car les règles de SysML donnent à chaque élément du modèle une définition unique, construite en rassemblant les informations issues de ses différentes représentations, et interdisent à celles-ci de se contredire. Cela permet de réduire à la fois le risque d'erreurs et le temps passé à la vérification des données par rapport aux approches basées sur des documents qui ne bénéficient pas d'un tel garde-fou ;
- cela facilite l'usage intensif de la *simulation* car un modèle SysML peut regrouper toutes les informations permettant de modéliser le système, de simuler son comportement, et de comparer les résultats aux exigences afin de valider (ou non) des solutions.

Ces avantages sont analogues à ceux que procure, en conception mécanique, la génération automatique de vues 2D à partir d'une maquette numérique 3D réalisée dans un logiciel de CAO (Figure 3(a)) par rapport au dessin direct, fût-il assisté par ordinateur, d'un ensemble de vues 2D séparées les unes des autres (Figure 3(b)).

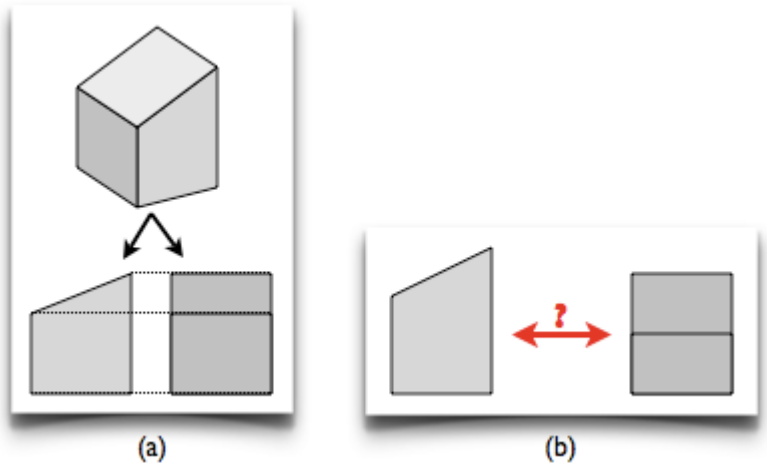

Figure 3 : (a) Un modèle dont des vues cohérentes sont extraites ; (b) un ensemble de vues dessinées indépendamment, et donc susceptibles de se contredire.

Naturellement, ces fonctionnalités sont d'autant plus avantageuses que le projet est de grande taille (nombreux composants, nombreuses relations, travail collaboratif...) et ne sont pleinement présentes que dans le cadre de l'utilisation d'un logiciel ; cependant, comme nous allons le voir, l'aspect structuré de SysML et le gain de rigueur qui en découle peuvent tout à fait être mis en évidence sur des exemples très simples, et rien n'empêche de concevoir et de communiquer un modèle SysML sous la forme de documents papier.

3. Un aperçu du langage SysML

SysML est un langage, c'est-à-dire un ensemble de signifiés (les éléments du modèle) et de signifiants (la façon dont ils sont représentés). Ce langage est essentiellement graphique : les signifiants sont des *boîtes* reliées par des *lignes*, ces dernières étant orientées ou non selon les cas. Il obéit aux quelques règles suivantes :

- Les boîtes et les lignes possèdent des *types*, indiqués par des conventions graphiques ou des mots-clés ; le type d'une boîte indique s'il s'agit du modèle d'un composant, d'un comportement... tandis que le type d'une ligne précise la nature de la relation entre les boîtes.
- De plus, les boîtes portent des *noms* servant à identifier les éléments du modèle qu'elles représentent : deux boîtes portant le même nom sont deux représentations (*a priori* différentes) du *même élément*, et ces représentations ne peuvent pas se contredire (mais elles peuvent se compléter).
- SysML possède une structure *hiérarchique*, cohérente vis-à-vis de la systémique : de nombreuses boîtes peuvent contenir d'autres boîtes reliées par des lignes, et le modèle entier est lui-même une boîte.
- La représentation du contenu d'une boîte s'appelle un *diagramme SysML*. Les diagrammes, tout comme les boîtes qu'ils représentent, possèdent des types qui déterminent la nature de l'information que l'on y trouve (structurelle, comportementale...) ; il en existe neuf. Chaque diagramme est obligatoirement dessiné dans un cadre muni d'un en-tête qui précise son type, suivi du type et du nom de la boîte représentée, et enfin d'une description libre (Figure 4).

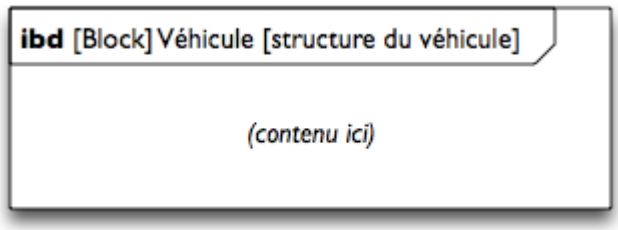

Figure 4 : Un exemple de cadre et d'en-tête d'un diagramme SysML

Ainsi, un diagramme SysML est, par définition, une *vue du modèle*, c'est-à-dire une représentation graphique d'une partie du modèle (en l'occurrence, d'une boîte) selon un point de vue particulier (donné par son type) ; l'en-tête permet d'identifier cette vue avec précision. Réciproquement, un modèle SysML est représenté par une *collection de diagrammes liés entre eux*.

- Enfin, *SysML n'impose pas l'exhaustivité :* un diagramme ne représente que les éléments utiles pour atteindre l'objectif (qu'il s'agisse de communiquer ou de simuler), et l'absence de représentation d'un élément sur un diagramme donné n'implique nullement que cet élément n'existe pas (il peut tout à fait figurer sur d'autres diagrammes). Deux diagrammes représentant un même élément de façon différente doivent donc être considérés comme *complémentaires* ; en revanche, s'ils s'avèrent *contradictoires*, alors il s'agit d'une faute, comme expliqué plus haut.

La suite de la ressource propose un rapide aperçu des neuf types de diagrammes en prenant l'exemple d'une automobile. Pour chaque diagramme, nous présentons son objectif, un exemple très simple permettant d'expliquer la signification des boîtes et des lignes, et un aperçu des relations possibles avec les autres diagrammes. Nous avons choisi de classer les diagrammes selon ce qui nous semble être leur point de vue dominant : fonctionnel, structurel ou comportemental, en nous appuyant sur la signification qu'ont ces termes dans les référentiels de Sciences de l'Ingénieur. Cependant, certains diagrammes combinent plusieurs de ces points de vue et nous avons donc dû faire des choix...

Source : ens-cachan

3.1. Les diagrammes fonctionnels

SysML propose deux types de diagrammes permettant d'effectuer une modélisation fonctionnelle. Le premier s'appelle *diagramme des cas d'utilisation* et est noté *uc* (pour *use case diagram*).

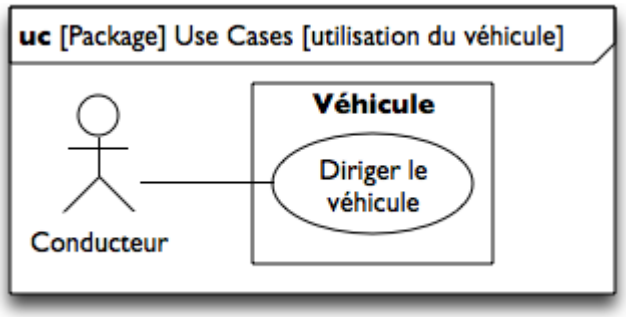

Figure 5 : Un diagramme des cas d'utilisation d'une automobile.

Un diagramme des cas d'utilisation comporte toujours quatre types d'éléments (Figure 5) :

- la *frontière* séparant le système de son environnement, représentée par un rectangle muni du nom du système (ici *Véhicule*) ;
- un ou plusieurs *cas d'utilisation*, qui sont des expressions d'actions pouvant être réalisées en utilisant le système (ici *Diriger le véhicule*) ;
- un ou plusieurs *acteurs*, qui sont des entités extérieures au système, réalisant et/ou subissant les actions décrites par les cas d'utilisation (ici *Conducteur*) ; en pratique, un acteur modélise généralement un humain ou un système automatisé ;
- une ou plusieurs *relations* entre ces éléments ; en particulier, une ligne entre un acteur et un cas d'utilisation signifie que l'acteur *participe* au cas d'utilisation, c'est-à-dire qu'il réalise et/ou subit l'action correspondante (SysML ne distingue pas les deux cas de figure). Plusieurs acteurs peuvent participer à un même cas d'utilisation, et réciproquement.

Ce diagramme permet également de classer les acteurs, de classer les cas d'utilisation et d'établir des relations entre ces derniers. Il se limite toutefois à l'*expression* des cas d'utilisation, et ne permet pas de les *caractériser* par des critères chiffrés, pas plus qu'il ne caractérise les acteurs.

Le second diagramme, nommé *diagramme des exigences* et noté *req* (pour *requirement diagram*), permet de spécifier, hiérarchiser et documenter des exigences, c'est-à-dire des attentes portant sur le système ou sur son comportement ; il s'agit d'une notion vaste, incluant par exemple toutes sortes de caractéristiques chiffrées, des choix technologiques imposés, le respect de normes ou de réglementations...

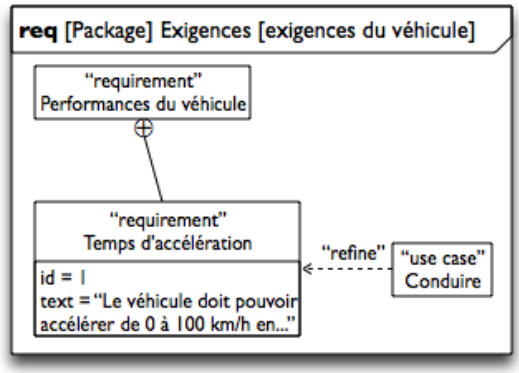

Figure 6 : Un diagramme des exigences d'une automobile.

Source : ens-cachan

Le diagramme de la Figure 6 représente le contenu d'un *package* nommé *Exigences* et :

- *définit deux exigences*, nommées *Performances du véhicule* et *Temps d'accélération* ; seule cette dernière est explicitée ;
- *hiérarchise* ces deux exigences : le trait continu terminé par un symbole "plus cerclé" (⊕) signifie que l'exigence *Temps d'accélération* est *contenue* dans l'exigence *Performances du véhicule* ;
- définit une *allocation*, c'est-à-dire une relation entre deux éléments définis sur des diagrammes différents : la flèche pointillée munie du mot-clé *"refine"* signifie que le cas d'utilisation *Conduire*, défini dans la Figure 5 et réutilisé ici, *raffine* l'exigence *Temps d'accélération*, c'est-à-dire que cette exigence caractérise ce cas d'utilisation.

Les allocations sont très utiles pour documenter les exigences ; outre la caractérisation des cas d'utilisation, elles permettent de mettre les exigences en relation avec les composants qui les satisfont, avec les modèles de comportements qui permettent de vérifier si elles sont satisfaites, ou encore d'en assurer la traçabilité. SysML permet également de définir des types d'exigences particuliers afin de classer celles-ci.

3.2. Les diagrammes structurels

En SysML, la modélisation structurelle repose essentiellement sur deux diagrammes utilisés conjointement. Le premier, appelé *diagramme de définition de blocs* et noté *bdd* (pour *"block definition diagram"*), sert à définir, classer et hiérarchiser les *blocs*, qui sont des modèles de composants --- ou, plus précisément, de *classes de composants* (Figure 7).

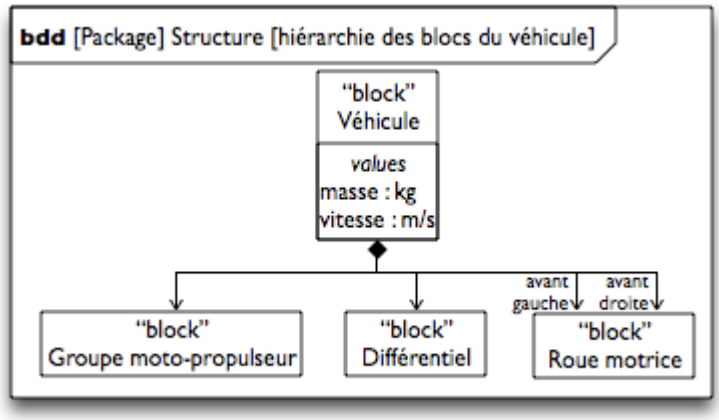

Figure 7 : Un diagramme de définition des blocs d'une automobile.

Ainsi, la Figure 7 représente le contenu d'un *package* nommé *Structure* et définit :

- quatre *blocs* (remarquons que la décomposition n'est pas exhaustive !) ;
- deux *caractéristiques* du bloc *Véhicule*, toutes deux pourvues d'une unité ;
- quatre *relations d'appartenance* des blocs *GMP*, *Différentiel* et *Roue motrice* au bloc *Véhicule* ; les deux flèches dirigées vers le bloc *Roue motrice* signifient qu'un composant de classe *Véhicule* contient deux composants de classe *Roue motrice*, et chacune de ces flèches est munie d'un *rôle*, c'est-à-dire d'un nom donné à la relation d'appartenance, permettant de distinguer ces deux composants.

Ce diagramme permet également d'affiner la définition des blocs grâce à des *stéréotypes*, c'est-à-dire des blocs "spécialisés" munis de noms dédiés. Les plus courants sont les blocs logiciels (*"software"*), les blocs matériels (*"hardware"*), les acteurs figurant sur le diagramme des cas d'utilisation (*"actor"*) et le système lui-même (*"system"* ou *"system of interest"*).

Source : ens-cachan

Le second, nommé *diagramme interne d'un bloc* et noté *ibd* (pour *"internal block diagram"*), modélise la structure interne d'un bloc : il représente les composants de ce bloc ainsi que les flux de matière, d'énergie et/ou d'information pouvant circuler entre ces composants (il existe un deuxième type de relations entre composants, nommé *services*, que nous n'abordons pas ici). Les composants sont modélisés par des *instances* des blocs ; leur type et leur nombre doivent être cohérents avec la définition donnée dans le diagramme précédent.

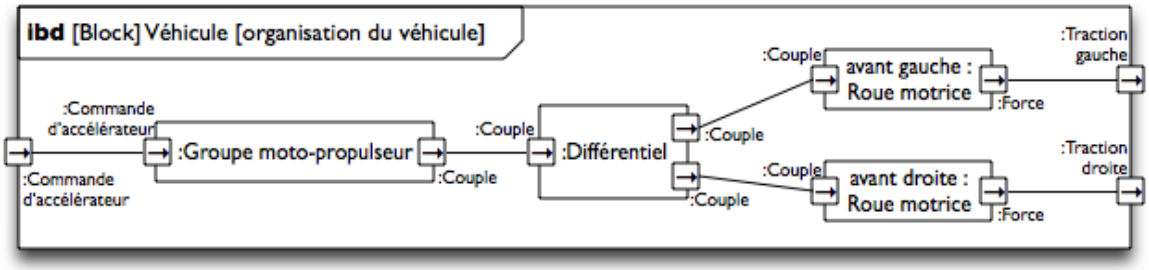

Figure 8 : Un diagramme interne d'un bloc d'une automobile.

Par exemple, la Figure 8 représente la vue interne du bloc *Voiture* défini sur la Figure 7 et :

- modélise les composants par des *instances de blocs*, nommées par un symbole "deux points" (:) suivi du nom du bloc dont elles sont issues, et précédé du rôle dans le cas où celui-ci est défini (ici, les deux instances du bloc *Roue motrice* possèdent les rôles respectifs *avant gauche* et *avant droite*) ;
- définit des *ports*, c'est-à-dire des interfaces permettant aux composants d'échanger des flux avec leurs environnements, représentés par des carrés chevauchant le bord du rectangle ; ici, les ports sont nommés d'après le type de flux qui les traverse et possèdent une direction (entrante ou sortante) ;
- représente les *flux possibles* par des connecteurs joignant les ports.

Ce mécanisme permet de modéliser la structure de systèmes sophistiqués avec une grande rigueur, toujours selon le même schéma : *définir des blocs dans un bdd, puis les instancier dans un ibd*. Leurs deux utilisations les plus courantes sont l'utilisation *"boîte blanche"*, consistant à modéliser la structure du produit comme sur les Figures 7 et 8, et l'utilisation *"boîte noire"*, consistant à définir des éléments du milieu extérieur dans un *bdd* (pouvant inclure, par exemple, les acteurs) puis à représenter les flux que ceux-ci peuvent échanger avec le produit dans un *ibd*. Dans les deux cas, il est important de garder à l'esprit qu'un *ibd* ne représente que des flux *possibles*, permis par la structure du système, et non les flux effectivement échangés à un certain instant de la vie du système : ce n'est pas un diagramme fonctionnel ou comportemental !

La modélisation structurelle joue généralement un rôle central en SysML. Elle peut notamment être liée :

- à la *modélisation comportementale*, en allouant des modèles de comportements aux blocs (ou à leurs instances) ou en spécifiant des relations mathématiques entre leurs caractéristiques, notamment dans le but de réaliser des simulations ;
- à la *modélisation fonctionnelle*, en faisant participer les acteurs (qui sont des blocs spécialisés !) aux cas d'utilisation, et en allouant les blocs (ou leurs instances) aux exigences qu'ils satisfont.

SysML offre enfin un troisième type de diagramme structurel, qui n'apporte aucune information sur le produit ou son environnement proprement dits, mais sert à classer les différentes données contenues dans un modèle : le *diagramme des paquets*, noté *pkg* (pour *"package diagram"*). Ce diagramme permet de définir des *paquets* ou *packages*, qui sont des éléments pouvant contenir d'autres éléments du modèle (cas d'utilisation, exigences, blocs, autres paquets...) selon un principe tout à fait analogue à celui des dossiers informatiques ; comme nous venons de le voir, de nombreux diagrammes SysML de haut niveau représentent le contenu d'un paquet, et le paquet "principal" contenant toutes les données (analogue au répertoire racine d'un disque dur) s'appelle tout

simplement... le *modèle*. L'organisation des données en paquets prend toute son utilité pratique lorsque l'on utilise un logiciel de modélisation.

3.3. Les diagrammes comportementaux

SysML offre quatre types de diagrammes comportementaux, appartenant à deux catégories bien distinctes. L'un de ces diagrammes permet de représenter graphiquement des *systèmes d'équations* reliant les caractéristiques du modèle. Les trois autres permettent de modéliser un comportement par une *suite d'étapes* (actions, états...) munie de règles d'évolution, et prennent généralement la forme de graphes orientés. Outre leur rôle descriptif, certains de ces diagrammes peuvent être utilisés pour réaliser des simulations.

Le diagramme paramétrique

La modélisation paramétrique consiste à construire des *systèmes de relations* (équations, éventuellement différentielles, ou inéquations) entre les grandeurs (physiques ou logiques) qui caractérisent les modèles des composants et des flux, afin de réaliser des simulations. En pratique, ces relations proviennent généralement de l'expression des lois d'une théorie ou de modèles de comportements de composants ; SysML n'étant pas un outil de modélisation physique, c'est au modélisateur qu'il revient de choisir et d'écrire ces relations, bien que certains logiciels proposent des bibliothèques susceptibles de faciliter cette tâche. Ici, nous nous intéressons par exemple au système d'équations suivant :

$$v(t) = \int_0^t a(\tau)\mathrm{d}\tau \qquad (1)$$

$$F(t) = ma(t) \qquad (2)$$

$$F(t) = F_\mathrm{g}(t) + F_\mathrm{d}(t) \qquad (3)$$

Pour représenter ce système d'équations en SysML, on procède en deux temps. Premièrement, on représente les différentes relations, ainsi que les grandeurs qui y interviennent, dans des *blocs de contraintes* (*constraint blocks*). Cette représentation s'effectue dans un *diagramme de définition de blocs* que l'on construit selon le même principe que s'il s'agissait de blocs "standard" (Figure 9).

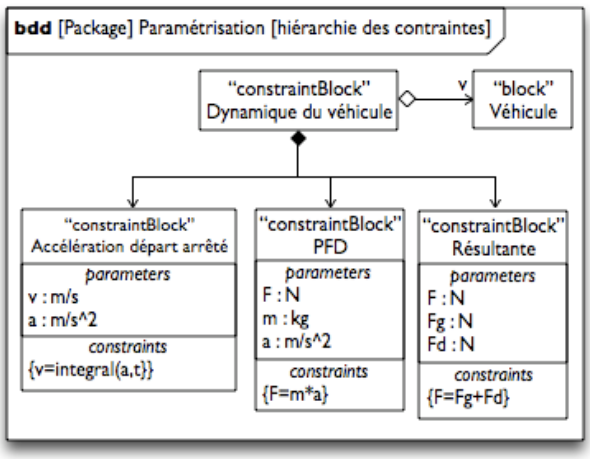

Figure 9 : Un diagramme de définition des blocs de contrainte d'une automobile, en vue d'une modélisation paramétrique.

Source : ens-cachan

Par exemple, le diagramme ci-dessus définit :

- quatre *blocs de contraintes*, dont trois sont caractérisés par des variables (*parameters*) et des relations entre ces variables (*constraints*), qui correspondent aux trois équations ci-dessus prises indépendamment les unes des autres, et sont écrites dans le langage du logiciel qui réalisera la simulation ;
- une *relation d'appartenance* de chacun des trois blocs caractérisés au bloc *Dynamique du véhicule* ;
- une *relation de référence* représentée par la flèche munie d'un losange évidé, qui signifie que le bloc de contrainte *Dynamique du véhicule* peut faire appel aux caractéristiques du bloc *Véhicule* (défini sur la Figure 7). Cette relation est munie d'un *nom de rôle*, à savoir *v*.

Dans un deuxième temps, on construit un *diagramme paramétrique*, noté *par*, qui représente graphiquement le système formé par ces contraintes et leurs paramètres (Figure 11). Pour cela, on *instancie* les blocs de contrainte, de la même façon que l'on instancie des blocs dans un *ibd*.

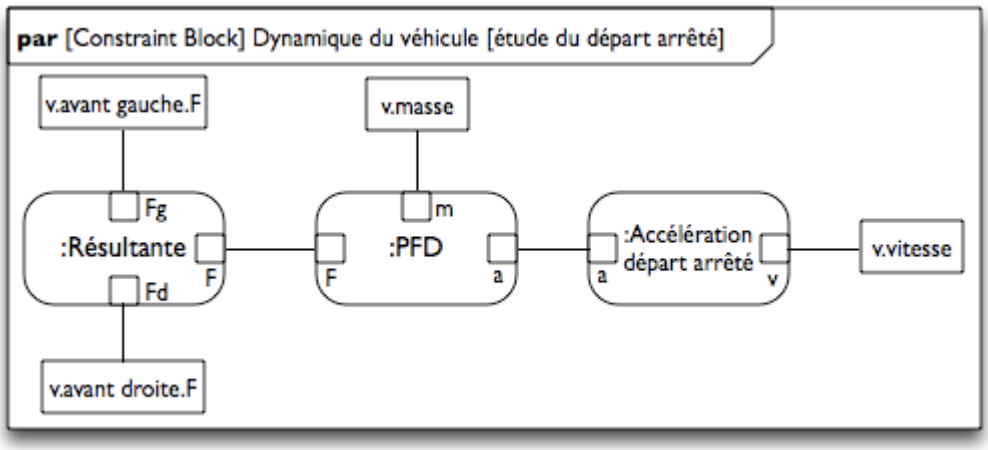

Figure 10 : Un diagramme paramétrique d'une automobile.

Le diagramme ci-dessus représente l'intérieur du bloc de contrainte *Dynamique du véhicule* défini sur la Figure 9. Il est construit de façon tout à fait similaire à un *ibd* :

- il définit trois *contraintes*, qui sont des instances des blocs de contraintes définis sur la Figure 9, et les représente par des rectangles à coins arrondis nommés selon la convention habituelle ; les *paramètres* intervenant dans l'expression de ces contraintes sont également représentés et nommés conformément à la définition des blocs de contraintes ;
- il définit quatre *références à des caractéristiques de blocs* ; ces références sont représentées par des rectangles portant le nom de la caractéristique précédé des rôles (noms des relations de référence ou d'appartenance) permettant de remonter jusqu'au bloc concerné (ici, la caractéristique *F* n'a pas été définie ; tout le reste est défini sur les Figures 7 et 9) ;
- enfin, il relie les *paramètres égaux* (entre deux contraintes ou entre une contrainte et une référence) afin de représenter la structure du système d'équations.

L'examen des Figures 9 et 10 montre que ce diagramme représente bien le système des équations (1) à (3) ; une fois les valeurs numériques des paramètres renseignées, il sera possible de réaliser une simulation à l'aide d'un logiciel approprié. Il est important de noter qu'*aucun des liens de la Figure 10 n'est orienté :* un diagramme paramétrique représente un système de relations mathématiques, et non un algorithme de résolution (qui n'existe d'ailleurs pas toujours !). Un tel modèle est dit *acausal* ; s'il est mathématiquement bien formulé (ce qui relève de la responsabilité du modélisateur), alors on pourra réaliser une simulation d'après ce modèle.

Source : ens-cachan

Les diagrammes basés sur des suites d'étapes

L'autre grande catégorie de modélisation comportementale offerte par SysML fait appel à une suite d'étapes et de règles d'évolution, représentées par des *graphes orientés* : elle est donc *causale*. SysML propose trois modèles de comportements de ce type, correspondant chacun à un point de vue différent, et représentés par un diagramme différent.

Le premier diagramme, nommé *diagramme d'activités* et noté *act* (pour *activity diagram*), permet de modéliser un comportement sous la forme d'une suite d'*actions* transformant des *flux* de matière, d'énergie et/ou d'information (Figure 11). Un tel modèle s'appelle une *activité* et contient, la plupart du temps, une description comportementale mais aussi fonctionnelle.

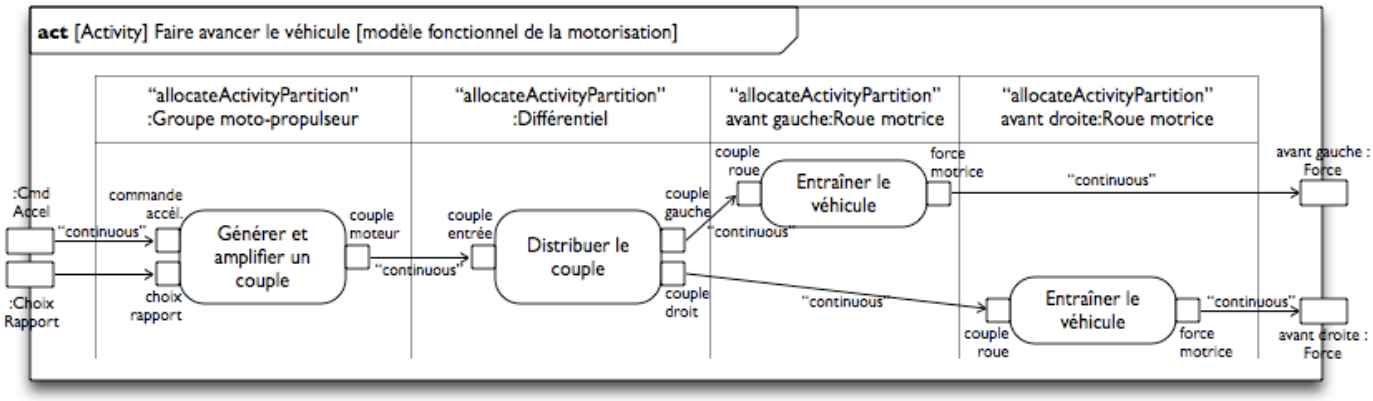

Figure 11 : Un diagramme d'activité d'une automobile.

Le diagramme de la Figure 11 représente une activité nommée *Faire avancer le véhicule* et définit :

- quatre *actions*, représentées par des rectangles dont les coins sont arrondis ;
- la *matière d'oeuvre entrante et sortante* de chacune de ces actions, représentée par des carrés situés sur le bord extérieur des actions ;
- la matière d'oeuvre entrante et sortante de l'activité entière, représentée par des rectangles chevauchant le cadre du diagramme ;
- les *flux* de la matière d'oeuvre au cours de l'activité, représentés par des flèches reliant les carrés ou rectangles ; ces flèches sont munies du mot-clé *"continuous"* indiquant qu'il s'agit de flux continus, à l'exception de celle qui représente le choix du rapport (par défaut, les flux sont supposés discrets) ;
- des *allocations* des actions aux modèles de composants (*ie.* aux instances des blocs), en partitionnant l'activité en colonnes appelées *swimlanes* ("lignes d'eau") portant chacune le nom d'une instance d'un bloc.

Les diagrammes d'activités possèdent une sémantique très riche, adaptée à des descriptions *à la fois fonctionnelles et comportementales*. Ils offrent de nombreuses possibilités dont la Figure 11 ne donne qu'un infime aperçu : ils permettent de représenter des flux continus ou discrets, d'imposer des conditions sur le déclenchement des actions (qui peuvent être activées et désactivées au cours du temps), de spécifier des exécutions parallèles ou des branchements conditionnels, d'inclure une activité dans une autre, de coupler plusieurs activités entre elles... Les actions sont souvent allouées aux instances des blocs. Ce diagramme est généralement utilisé à des fins purement descriptives, mais peut parfois servir à réaliser des simulations si les comportements sont modélisés de façon suffisamment détaillée.

Le second diagramme, nommé *diagramme d'états* et noté *stm* (pour *state machine diagram*), permet de modéliser un comportement par un *automate fini* (parfois appelé *machine à états*), c'est-à-dire une suite d'*états*

reliés par des *transitions* pilotées par des *événements*, supposés instantanés et de durée négligeable (Figure 12) ; la règle de base est que le système change d'état lorsque ces événements surviennent.

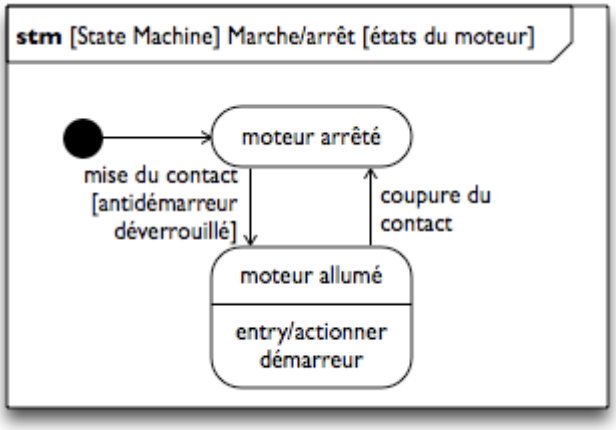

Figure 12 : Un diagramme d'états d'une automobile.

Le diagramme de la Figure 12 représente un automate fini nommé *Marche/arrêt* et définit :

- *deux états* nommés *moteur arrêté* et *moteur allumé* ; ce dernier possède un *comportement d'entrée* repéré par le mot-clé *entry*, qui est appelé au moment où cet état devient actif et n'est pas défini sur ce diagramme ;
- un *pseudo-état* indiquant l'état initial, représenté par un disque noir ;
- deux *transitions* entre les états ; toutes deux possèdent un *événement déclencheur* (respectivement *mise du contact* et *coupure du contact*) et l'une d'entre elles possède de plus une *condition de garde* indiquée entre crochets (*antidémarrage déverrouillé*). Un changement d'état s'effectue lorsque l'événement déclencheur de la transition se produit, sous réserve que la condition de garde (s'il y en a une) soit vraie à l'instant où il se produit.

Les automates finis permettent de modéliser des comportements selon un point de vue complémentaire à celui des activités : l'enchaînement des états est piloté par les *événements* mentionnés dans les transitions, tandis que l'enchaînement des actions est uniquement piloté par leur *durée d'exécution*. Les deux modèles présentent cependant des possibilités syntaxiques comparables (exécutions parallèles ou branchements conditionnels, inclusion, couplages...) et peuvent tous deux être alloués à des blocs ou à leurs instances (mais l'allocation des automates n'est pas représentée par des *swimlanes*, contrairement à celle des activités). Les automates sont régis par des règles précises et leur niveau de détail est généralement assez élevé pour permettre la réalisation de simulations.

Enfin, le *diagramme de séquence*, noté *sd* (pour *sequence diagram*), permet de modéliser des *interactions entre des instances de blocs* (c'est-à-dire des modèles de composants du produit et de son environnement), selon un ordre chronologique et sans détailler leurs comportements individuels. Ces interactions sont modélisées par des *messages*, définis en SysML comme une action qui déclenche un comportement prédéfini chez son destinataire.

Source : ens-cachan

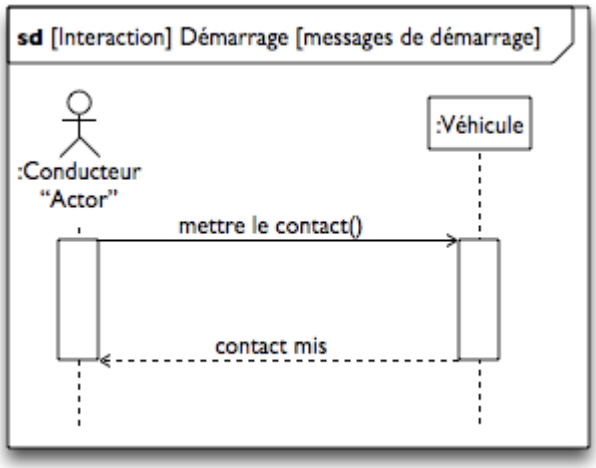

Figure 13 : Un diagramme de séquence d'une automobile.

Le diagramme de la Figure 13 :

- représente *deux instances de blocs* dont l'un est un acteur ;
- attribue à chaque instance une *ligne de vie*, c'est-à-dire une ligne pointillée symbolisant l'écoulement du temps (vers le bas) ;
- représente *deux messages*, *mettre le contact()* et *contact mis* ;
- représente les *comportements individuels* des deux instances par des rectangles blancs, sans les détailler ; c'est dans le cadre de ces comportements que les messages sont échangés.

Les diagrammes de séquences sont souvent associés à des diagrammes d'activités ou d'états couplés entre eux, en raison de la complémentarité des points de vue : en effet, une activité ou un automate fini représentent généralement le comportement individuel d'un composant sans détailler ses interactions avec l'extérieur, tandis qu'un diagramme de séquence représente uniquement les interactions. Les interactions sont explicitement liées à la modélisation structurelle, puisque des instances de blocs y figurent ; de plus, comme tout modèle de comportement, elles peuvent être allouées à la modélisation fonctionnelle, et il est notamment fréquent d'allouer une interaction à un cas d'utilisation afin de développer celui-ci.

Source : ens-cachan

4. Bilan

Dans cette ressource, nous avons mis en évidence les points suivants :

1. SysML permet de modéliser un produit et son environnement par un *système*, c'est-à-dire un ensemble de composants qui interagissent entre eux.
2. Cette modélisation est *multi-points de vue* et inclut notamment des aspects fonctionnels, structurels et comportementaux.
3. Un modèle SysML est représenté par un ensemble de *boîtes* et de *lignes*, munies de *types*, de *noms* permettant de les identifier, et d'une *hiérarchie* (une boîte contient d'autres boîtes et des lignes).
4. Les boîtes et les lignes sont assemblées au sein de *diagrammes* interdépendants : un même élément, repéré par son nom, peut figurer sur des diagrammes différents, qui apportent alors des informations complémentaires.
5. Il existe *neuf types de diagrammes*. Ces types, ainsi que les relations possibles entre les éléments représentés par ces diagrammes, sont récapitulés sur la Figure 14 (le diagramme des paquets a été omis dans un souci de simplicité).
6. Un diagramme doit toujours représenter l'intérieur d'une boîte. Cependant, cette représentation peut être partielle (on ne représente que ce dont on a besoin).

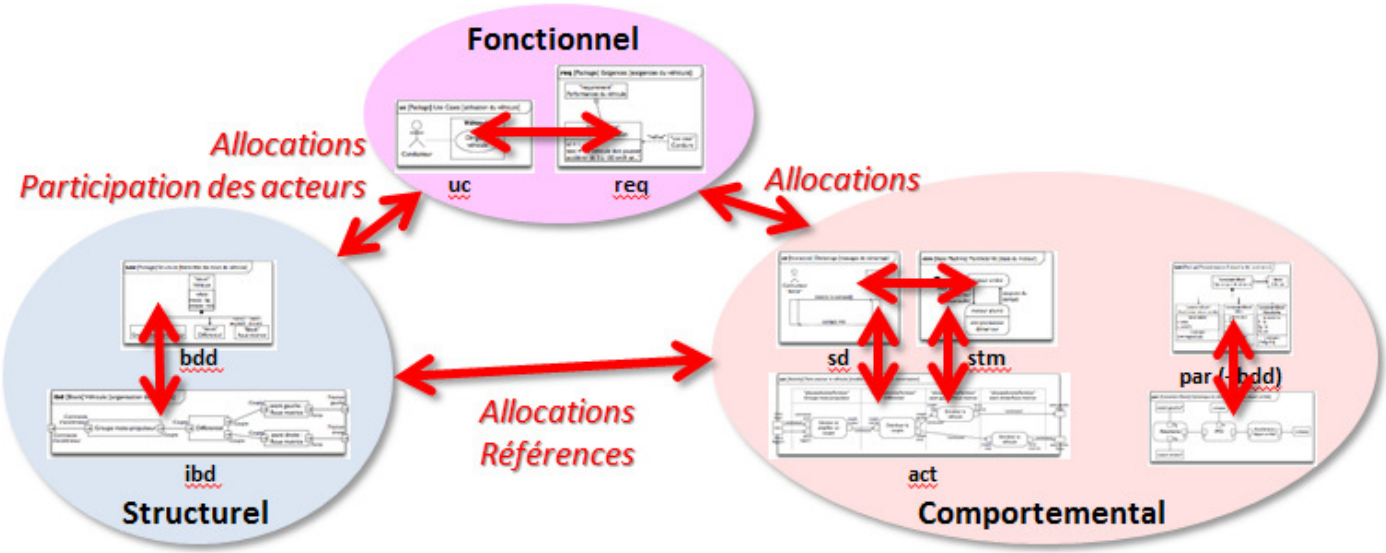

Figure 14 : Récapitulatif des neuf types de diagrammes SysML (sans le diagramme des paquets) ainsi que des relations possibles entre les contenus de ces diagrammes.

Pour un exposé plus complet des possibilités offertes par SysML et un récapitulatif des principaux éléments syntaxiques pouvant figurer sur chaque diagramme, vous pouvez télécharger le support de la formation LIESSE sur SysML, proposée par Lionel Gendre et Jean-Marie Virely (environ 4 Mo).