Modélisation de systèmes complexes avec SysML

Pascal Roques

Préface de Jean-Michel Bruel

EYROLLES

© Groupe Eyrolles, 2013, ISBN : 978-2-212-13641-8

Préface

Enfin un livre sur SysML en français, qui plus est écrit par Pascal Roques qui a commis les livres sur UML les plus utilisés dans la francophonie à ma connaissance. Voilà qui va réjouir bon nombre de mes collègues enseignants qui commencent à intégrer SysML dans leurs cursus et bon nombre d'ingénieurs fatigués de ne trouver que des articles de recherche ou de vulgarisation utilisant souvent les mêmes exemples, pour la plupart issus des documents de l'*Object Management Group*, à l'origine de SysML.

La frontière entre le logiciel et le matériel est de plus en plus étroite. L'automobile en est l'illustration la plus frappante car elle touche tout le monde (le métier de garagiste est en pleine évolution !). Toutefois, l'aéronautique avait commencé avant elle. Et pour sévir depuis de nombreuses années dans le milieu de la formation toulousaine, Pascal Roques ne le sait que trop bien. Il est fort opportun qu'il n'ait pas attendu trop longtemps pour mettre à profit ses talents de pédagogue et d'auteur pour nous concocter un livre simple, clair et abordable qui, j'en suis persuadé, n'est que le premier d'une série d'autres qui germent sûrement déjà dans son esprit fertile. Le monde des informaticiens et celui des « systémistes » sont souvent restés étanches et les barrières culturelles vont être longues à faire tomber. SysML, s'il n'est qu'une notation au service d'une meilleure communication entre ces deux mondes, n'en reste pas moins un passage obligé pour toute personne amenée à participer au développement de systèmes complexes, où les notations classiques trouvent leurs limites.

Pascal a fait appel à un universitaire pour sa préface. Mais ne vous y trompez pas. Ce livre va au-delà du manuel que l'on indique en référence dans son support de cours, ou celui duquel on tire ses exemples (voire ses idées d'examens, chers étudiants !). Les relations industrielles très variées de l'auteur sont utilisées pleinement dans l'ouvrage que vous avez entre les mains pour à la fois nous expliquer simplement les tenants et les aboutissants de cette notation nouvelle, mais aussi coller à une réalité de terrain qui satisfera aussi bien les enseignants, les étudiants que les concepteurs de systèmes. Les premiers y trouveront des exemples concrets pour illustrer leurs recher-

ches et leurs enseignements, les derniers y trouveront un décodage clair et didactique des subtilités de la spécification. Tous trouveront la plupart des réponses aux interrogations et aux points d'ombre qui subsistent dans la spécification OMG de SysML. Je vous souhaite une très bonne lecture !

Jean-Michel Bruel
Professeur des universités
Université de Toulouse – IUT de Blagnac
Membre de l'institut de recherche en informatique de Toulouse
(Unité mixte de recherche CNRS)

Table des matières

Avant-propos

Objectifs du livre

L'ingénierie système est une démarche méthodologique générale qui englobe l'ensemble des activités adéquates pour concevoir, faire évoluer et vérifier un système apportant une solution économique et performante aux besoins d'un client tout en satisfaisant l'ensemble des parties prenantes. Depuis longtemps, les ingénieurs système ont utilisé des techniques de modélisation. Parmi les plus connues, on trouve SADT et SA/RT, qui datent des années 1980, ainsi que de nombreuses approches basées sur les réseaux de Pétri ou les machines à états finis. Mais ces techniques étaient limitées par leur portée et leur expressivité ainsi que par la difficulté de leur intégration avec d'autres formalismes et avec les exigences.

L'essor d'UML dans le domaine du logiciel et l'effort industriel de développement d'outils qui l'accompagne ont naturellement conduit à envisager son utilisation en ingénierie système. Cependant, du fait de sa conception fortement guidée par l'objectif du passage à la programmation objet, le langage était, tout au moins dans ses premières versions, peu adapté à la modélisation des systèmes complexes et donc au support de l'ingénierie système.

La version 2 d'UML, officialisée en 2005, a introduit plusieurs nouveaux concepts et diagrammes utiles pour la modélisation système. En particulier, le diagramme de structure composite avec les concepts de classe structurée, partie, port et connecteur, permet maintenant de décrire l'interconnexion structurelle interne d'un système complexe. Les avancées du diagramme de séquence permettent également de décrire des scénarios d'interaction de façon descendante en ajoutant progressivement des niveaux d'architecture. Mais il reste toujours la barrière psychologique du vocabulaire orienté logiciel : classe, objet, héritage, etc.

La communauté de l'ingénierie système a voulu définir un langage commun de modélisation adapté à sa problématique, comme UML l'est devenu pour les informaticiens. Ce nouveau langage, nommé SysML, est fortement inspiré de la version 2 d'UML, mais ajoute la possibilité de représenter les exigences du système, les éléments non logiciels (mécaniques, hydrauliques, capteurs…), les équations physiques, les flux continus (matière, énergie, etc.) et les allocations.

La version 1.0 du langage de modélisation SysML a été adoptée officiellement par l'OMG le 19 septembre 2007. Depuis, trois révisions mineures ont été publiées : SysML 1.1 en décembre 2008, SysML 1.2 en juin 2010 et SysML 1.3 (la version courante à l'heure où nous écrivons ce livre) en juin 2012.

Mon ambition dans cet ouvrage introductif est de vous faire découvrir ce nouveau langage de modélisation pour l'ingénierie système. Fort de mon passé de consultant en modélisation dans de nombreux domaines, de ma pratique pédagogique en tant que formateur UML et SysML depuis plus de 15 ans, ainsi que de mon expérience d'auteur sur UML (plus de 50 000 exemplaires vendus), j'espère parvenir à vous faire apprécier la richesse et les atouts du langage SysML.

Ce livre s'adresse avant tout aux professionnels de l'ingénierie système, c'est-à-dire à tous ceux qui ont en charge des systèmes complexes, incluant du logiciel et du matériel, que ce soit dans l'aéronautique, le spatial, l'automobile, l'énergie, le transport, l'armement, etc. Même si une connaissance préalable du langage de modélisation UML est un atout supplémentaire pour bien saisir toutes les subtilités de SysML, cela n'est absolument pas un prérequis.

Structure de l'ouvrage

Après une brève introduction sur la problématique de l'ingénierie système, je détaillerai l'historique du langage SysML. Je présenterai ensuite les principes du processus de modélisation appliqué dans le livre.

La partie I de l'ouvrage concerne la modélisation des exigences. Nous verrons que SysML innove en permettant de modéliser les exigences du système et surtout de les relier ensuite aux éléments structurels ou dynamiques de la modélisation, ainsi qu'à d'autres exigences de niveau sous-système ou équipement. Nous apprendrons également à mettre en œuvre la technique des cas d'utilisation, déjà présente en UML. Nous verrons enfin une première application du diagramme de séquence pour décrire les interactions entre le système « boîte noire » et son environnement (les acteurs).

La partie II concerne la modélisation structurelle. Nous apprendrons à utiliser le concept universel de « bloc » proposé par SysML pour modéliser tout élément struc-

turel, ainsi que les deux types de diagrammes associés. Nous verrons tout d'abord comment définir les éléments structurels de base de notre modèle dans le diagramme de définition de blocs. Nous apprendrons ensuite à décrire la décomposition des éléments complexes avec le diagramme interne de bloc. Nous verrons enfin comment structurer notre modèle en *packages*, à des fins de travail en équipe ou de réutilisation.

La partie III concerne la modélisation dynamique. Nous verrons toute la puissance du diagramme d'états, pour modéliser le cycle de vie des éléments fortement réactifs, ainsi que celle du diagramme d'activité, qui permet de modéliser avec précision des algorithmes et des procédures complexes.

La partie IV concerne la modélisation transverse. SysML permet de décrire des équations grâce au nouveau diagramme paramétrique. Nous verrons également comment décrire plusieurs types de liens de traçabilité entre éléments de modélisation, et en particulier comment mettre en œuvre le concept fondamental d'allocation.

Une première annexe récapitule les acronymes et les définitions. Une seconde annexe présente des diagrammes complémentaires réalisés avec des outils de modélisation différents de l'outil principal (MagicDraw) : *Rhapsody*, *Enterprise Architect*, *TopCased et Artisan Studio*.

Remerciements

Cet ouvrage n'aurait jamais vu le jour sans les encouragements et les retours motivants des lecteurs de mes précédents livres sur UML, mais aussi des nombreux stagiaires des cours SysML que je donne depuis plus de six ans déjà. Ils m'ont donné l'énergie et l'envie de partager une fois encore mes connaissances et mon expérience à travers ce support que j'apprécie tant.

Merci à mes relecteurs techniques pour leurs judicieuses remarques :

- Tony Cornuaud (ingénieur en informatique, évangéliste SysML chez Magneti Marelli à Châtellerault) ;
- Christophe Surdieux (ancien de Valtech, maintenant chez Altran) ;
- Olivier Casse (expert en langages et outils de modélisation de systèmes embarqués, ancien d'I-Logix/Telelogic et Atego/Artisan) ;
- Éric Panier (évangéliste SysML chez Nexter Electronics à Toulouse).

Merci à la société NoMagic (http://www.nomagic.com/) de m'avoir fourni une licence de l'outil MagicDraw (maintenant Cameo Systems Modeler) avec son plug-in SysML que je trouve excellent, et grâce auquel j'ai réalisé la majeure partie des diagrammes du livre.

Merci aussi à Éric Sulpice et Muriel Shan Sei Fan des Éditions Eyrolles pour leur témoignage renouvelé de confiance. Un grand bravo également à l'équipe des éditrices qui a contribué notablement à la réussite de ce projet.

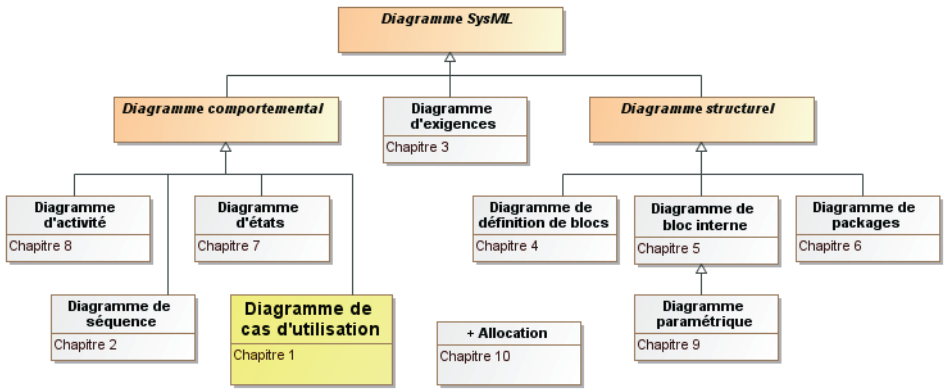
Enfin, comment ne pas terminer par un clin d'œil affectueux à celle qui partage ma vie depuis maintenant plus de dix ans, et dont l'énergie m'aide à avancer. Merci à Sylvie, la femme qui m'accompagne...

Pascal Roques, mars 2013
Formateur / Consultant senior (PRFC)
pascal.roques@prfc.fr

1

Le diagramme de cas d'utilisation

Ce chapitre présente le diagramme de cas d'utilisation, qui fournit une description de haut niveau des fonctionnalités du système.



Notation de base

Il est souhaitable de représenter les services attendus du système à l'étude par un modèle de cas d'utilisation. Ce modèle contient un ou plusieurs diagrammes de cas d'utilisation, montrant les interactions fonctionnelles entre les acteurs et le système.

B.A.-BA Cas d'utilisation

Un cas d'utilisation (*use case*, ou UC) représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier. Chaque cas d'utilisation spécifie un comportement attendu du système considéré comme un tout, sans imposer le mode de réalisation de ce comportement. Il décrit ce que le futur système devra faire, sans spécifier comment il le fera.
Nommez les cas d'utilisation par un verbe à l'infinitif suivi d'un complément, du point de vue de l'acteur (et non du système).

B.A.-BA Acteur

Rôle joué par un utilisateur humain ou un autre système qui interagit directement avec le système étudié. Un acteur participe à au moins un cas d'utilisation. Un cas d'utilisation doit être relié à au moins un acteur.

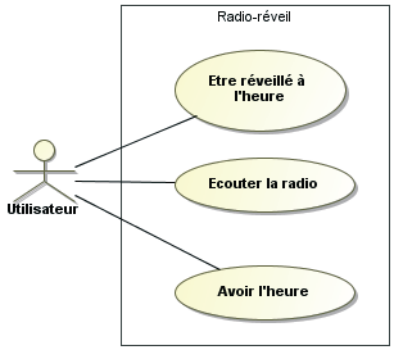
Le diagramme de cas d'utilisation est un schéma qui montre les cas d'utilisation (ovales) reliés par des associations (lignes) à leurs acteurs (icône du *stick man*, ou représentation graphique équivalente). Chaque association signifie simplement « participe à ».

Pour notre étude de cas, une première version du diagramme de cas d'utilisation consiste à considérer un seul acteur (l'utilisateur) connecté à un unique cas d'utilisation (`Être réveillé à l'heure`).

Figure 1–1
Acteur et cas d'utilisation

Ensuite, on peut se dire que l'utilisateur, alors qu'il est réveillé, est susceptible d'utiliser le radio-réveil en tant que simple radio ou horloge. Chaque cas d'utilisation doit bien représenter un service autonome rendu par le système et fournissant un résultat observable et intéressant pour l'acteur concerné. Si l'on dessine un rectangle englobant autour des cas d'utilisation pour matérialiser le radio-réveil, on obtient la figure suivante.

Figure 1–2
Acteur et cas d'utilisation (suite)



ATTENTION Cas d'utilisation

Une erreur fréquente consiste à vouloir descendre trop bas en termes de granularité. Un cas d'utilisation représente un ensemble de séquences d'actions réalisées par le système, et le lien entre ces séquences d'actions est précisément l'objectif métier de l'acteur. Le cas d'utilisation ne doit donc pas se réduire systématiquement à une seule séquence, et encore moins à une simple action.
Limitez à 20 le nombre de vos cas d'utilisation de base (en dehors des cas inclus, spécialisés, ou des extensions). Avec cette limite arbitraire, on reste synthétique et on ne tombe pas dans le piège de la granularité trop fine des cas d'utilisation.

Les acteurs candidats sont systématiquement :

- les utilisateurs humains directs : faites donc en sorte d'identifier tous les profils possibles, sans oublier l'administrateur, l'opérateur de maintenance, etc. ;
- les autres systèmes connexes qui interagissent aussi directement avec le système étudié, souvent par le biais de protocoles bidirectionnels.

La recommandation commune consiste à faire prévaloir l'utilisation de la forme graphique du *stick man* pour les acteurs humains et une représentation rectangulaire pour les systèmes connectés (si l'outil de modélisation le permet…).

ATTENTION Acteurs

Ne confondez pas rôle et entité concrète. Une même entité concrète peut jouer successivement différents rôles par rapport au système étudié, et être modélisée par plusieurs acteurs. Réciproquement, le même rôle peut être tenu simultanément par plusieurs entités concrètes, qui seront alors modélisées par le même acteur.

Nous appelons acteur principal celui pour qui le cas d'utilisation produit un résultat observable. Par opposition, nous qualifions d'acteurs secondaires les autres participants du cas d'utilisation. Ces derniers secondaires sont souvent sollicités pour des

informations complémentaires ; ils peuvent uniquement consulter ou informer le système lors de l'exécution du cas d'utilisation.

Une bonne pratique consiste à faire figurer les acteurs principaux à gauche des cas d'utilisation et les acteurs secondaires à droite. Si nous ajoutons les émetteurs radio en tant qu'acteurs secondaires pour les cas d'utilisation du radio-réveil liés à la radio, nous obtenons la figure suivante.

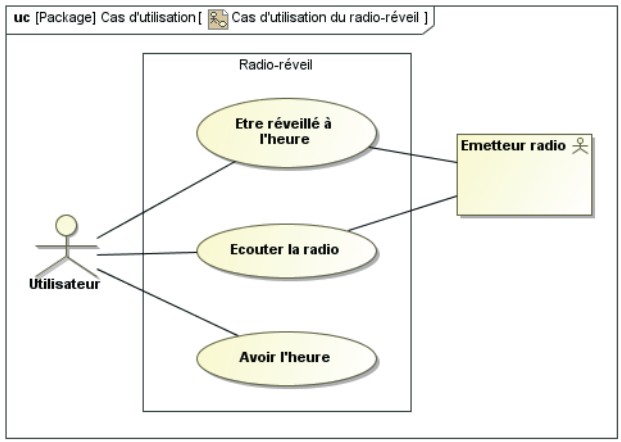


Figure 1–3
Acteurs principal humain et secondaire non humain

B.A.-BA Cartouche de diagramme

Un cartouche positionné en haut à gauche du diagramme dans un pentagone sert à spécifier le type de diagramme SysML, le type et le nom de l'élément concerné, ainsi que le nom du diagramme.
Dans l'exemple précédent, il s'agit d'un diagramme de cas d'utilisation (`uc`) nommé `Cas d'utilisation du radio-réveil`, concernant l'élément `Cas d'utilisation` de type `Package`.

Le cartouche général du diagramme de cas d'utilisation est de la forme :

uc `[package ou bloc] nom de l'élément [nom du diagramme]`

Décrire les cas d'utilisation

Chaque cas d'utilisation doit être décrit textuellement (à l'aide d'un plan type). On peut également associer à chaque cas d'utilisation un ou plusieurs diagrammes de séquence, comme expliqué dans le chapitre qui suit.

La fiche de description textuelle d'un cas d'utilisation n'est pas normalisée par SysML. Nous préconisons pour notre part la structuration suivante.

- Sommaire d'identification (obligatoire) : avec titre, résumé, dates de création et de modification, version, responsable, acteurs…
- Description des scénarios (obligatoire) : pour décrire le scénario nominal, les scénarios (ou enchaînements) alternatifs, les scénarios (ou enchaînements) d'échec, mais aussi les préconditions (l'état du système pour que le cas d'utilisation puisse démarrer) et les postconditions (ce qui a changé dans l'état du système à la fin du cas d'utilisation).

B.A.-BA **Scénario**

Un scénario représente une succession particulière d'enchaînements, s'exécutant du début à la fin du cas d'utilisation, un enchaînement étant l'unité de description de séquences d'actions. Un cas d'utilisation contient en général un scénario nominal et plusieurs scénarios alternatifs (qui se terminent de façon normale) ou d'erreur (qui se terminent en échec).
On peut d'ailleurs proposer une définition différente pour un cas d'utilisation : « ensemble de scénarios d'utilisation d'un système reliés par un but commun du point de vue de l'acteur principal ».

- Exigences non fonctionnelles (optionnel) : pour ajouter, si c'est pertinent, des informations telles que fréquence, volumétrie, disponibilité, fiabilité, intégrité, confidentialité, performances, concurrence, etc. Préciser également les contraintes d'interface homme-machine comme des règles d'ergonomie, une charte graphique, etc.

Compléments

Classification des acteurs

On pourrait imaginer distinguer les cas d'utilisation du radio-réveil selon que l'utilisateur est endormi ou déjà réveillé. En effet, c'est l'utilisateur endormi qui souhaite être réveillé à l'heure, alors que c'est l'utilisateur éveillé qui va écouter la radio ou regarder l'heure. Une version alternative du diagramme de cas d'utilisation est donnée à la figure suivante.

On peut même aller un peu plus loin : grâce au dispositif de projection au plafond, l'utilisateur (à moitié) endormi peut lui aussi lire l'heure ! On pourrait ainsi considérer que tout utilisateur peut avoir l'heure, qu'il soit endormi ou réveillé, mais que seul l'utilisateur endormi veut se faire réveiller et que seul l'utilisateur éveillé peut choisir d'écouter la radio.

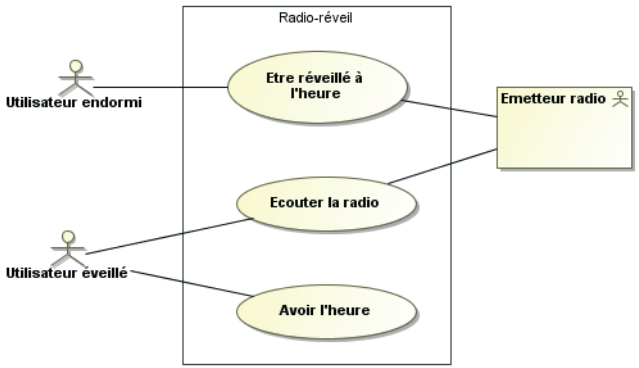


Figure 1–4
Première variante du diagramme de cas d'utilisation

Pour éviter d'avoir deux acteurs principaux connectés au même cas d'utilisation (utilisateur endormi et utilisateur éveillé), SysML permet de créer un acteur généralisé `Utilisateur` qui factorise les comportements communs aux deux acteurs. On dit alors que les acteurs `Utilisateur endormi` et `Utilisateur éveillé` sont des spécialisations de l'acteur `Utilisateur`. Chacun possède ses propres cas d'utilisation spécifiques.

B.A.-BA Acteur généralisé

Deux acteurs ou plus peuvent présenter des similitudes dans leurs relations aux cas d'utilisation. On exprime ce concept en créant un acteur généralisé, qui modélise les aspects communs aux différents acteurs concrets. L'acteur généralisé, considéré comme abstrait, est noté en italique.
Les acteurs spécialisés sont utiles pour exprimer des sous-profils particuliers d'un profil général.

Le diagramme suivant est toutefois un peu « tiré par les cheveux », la distinction entre utilisateurs endormi et éveillé n'étant pas indispensable (et s'apparentant plutôt à la notion d'état…). Il nous sert à illustrer les concepts d'acteur spécialisé et généralisé, mais nous recommanderions plutôt le diagramme simple de la figure 1-3 sur un vrai projet !

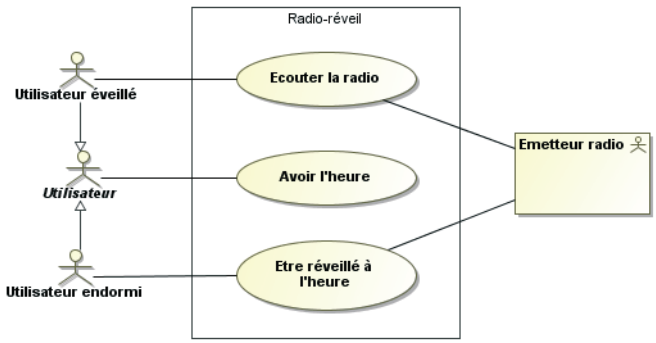


Figure 1–5
Variante du diagramme avec généralisation d'acteurs

Relations entre cas d'utilisation

Pour affiner le diagramme de cas d'utilisation, SysML définit trois types de relations standardisées entre cas d'utilisation :

- une relation d'inclusion, formalisée par le mot-clé `«include»` : le cas d'utilisation de base en incorpore explicitement un autre, de façon obligatoire ;
- une relation d'extension, formalisée par le mot-clé `«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*) ;
- une relation de généralisation/spécialisation (flèche blanche) : les cas d'utilisation descendants héritent de la description par leur parent commun. Chacun d'entre eux peut néanmoins comprendre des interactions spécifiques supplémentaires.

Essayons de donner quelques exemples sur notre étude de cas.

Le cas d'utilisation *`Avoir l'heure`* pourrait se spécialiser suivant que la lecture de l'heure se fait directement sur le radio-réveil ou alors au plafond. Le cas d'utilisation généralisé serait alors considéré comme abstrait et noté en italique. On pourrait également connecter le cas d'utilisation `Avoir l'heure au plafond` à un acteur secondaire supplémentaire représentant le plafond de la chambre…

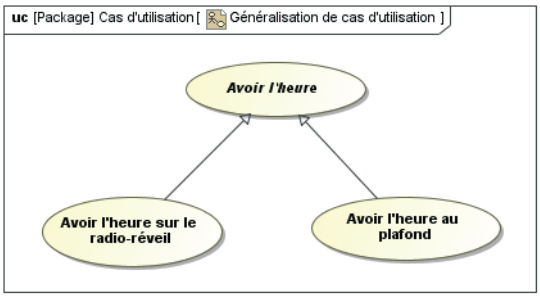
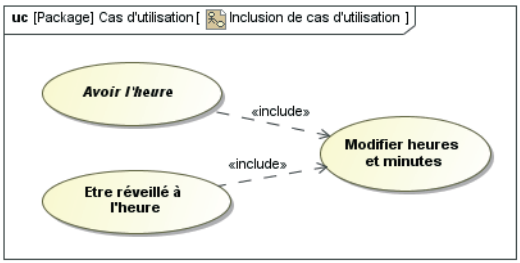


Figure 1–6
Généralisation/spécialisation de cas d'utilisation

Les cas d'utilisation *`Avoir l'heure`* et `Être réveillé à l'heure` impliquent tous les deux que les heures et les minutes sont modifiables, afin de mettre à jour l'heure d'alarme et/ou l'heure courante. On serait donc tenté de créer un nouveau cas d'utilisation pour ne déclarer et décrire qu'une seule fois un comportement réutilisable, tel que la modification de l'horodatage.

Le diagramme suivant illustre la relation d'inclusion ainsi proposée.

Figure 1–7
Inclusion de cas d'utilisation



Attention, ce schéma montre en fait une grosse erreur de modélisation. En effet, il spécifie qu'à chaque exécution de l'un des deux cas d'utilisation de base, on exécute également le cas d'utilisation inclus, ce qui est manifestement complètement faux… La relation `«include»` n'exprime pas une potentialité, mais bien une inclusion obligatoire, tout à fait inappropriée dans ce cas précis.

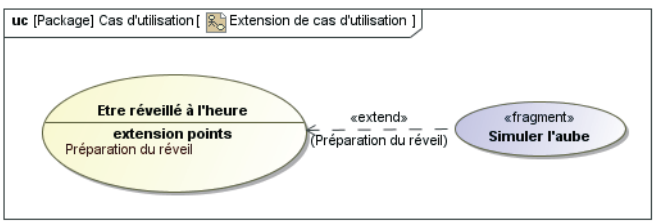
Pour la relation d'extension, souvent mal utilisée dans la pratique elle aussi, nous pourrions prendre en compte une fonctionnalité optionnelle, telle que la simulation d'aube (la lumière augmente progressivement pendant 30 à 90 min avant l'heure de réveil : le contact de la lumière avec les paupières prépare doucement l'organisme au réveil…).

Ce nouveau cas d'utilisation peut être stéréotypé `«fragment»` afin d'exprimer le fait qu'il ne s'exécutera jamais de façon autonome, mais toujours dans le cadre d'un autre cas d'utilisation. Il s'agit d'une bonne pratique préconisée par de nombreux spécialistes, mais qui n'est pas obligatoire.

B.A.-BA Stéréotype

Les mots-clés SysML comme `«include»` et `«extend»` sont notés entre guillemets typographiques. Nous pouvons également créer nos propres mots-clés, tels que `«fragment»` (pour indiquer qu'un cas d'utilisation n'est qu'un fragment factorisé par d'autres cas d'utilisation) ou `«secondaire»` (pour indiquer qu'un cas d'utilisation est moins important que les autres). Ces mots-clés inventés par les modélisateurs s'appellent alors des stéréotypes.

Figure 1–8
Extension de cas d'utilisation



Une version complexe du diagramme de cas d'utilisation, incorporant toutes les relations possibles entre acteurs et cas d'utilisation, est donnée par le schéma suivant. Précisons une fois de plus qu'il s'agit d'un exemple de notation et surtout pas d'une recommandation méthodologique. Souvenez-vous, par exemple, que la relation d'inclusion est erronée dans le cas présent. La relation de généralisation entre acteurs est probablement superflue, de même que celle entre les cas d'utilisation. En fait, nous considérons que le diagramme 1-3 est tout à fait suffisant pour notre étude de cas !

ATTENTION Relations entre UC

N'abusez pas des relations entre cas d'utilisation (inclusion, extension, généralisation) : elles rendent parfois les diagrammes de cas d'utilisation trop difficiles à décrypter pour les experts métier qui sont censés les valider.

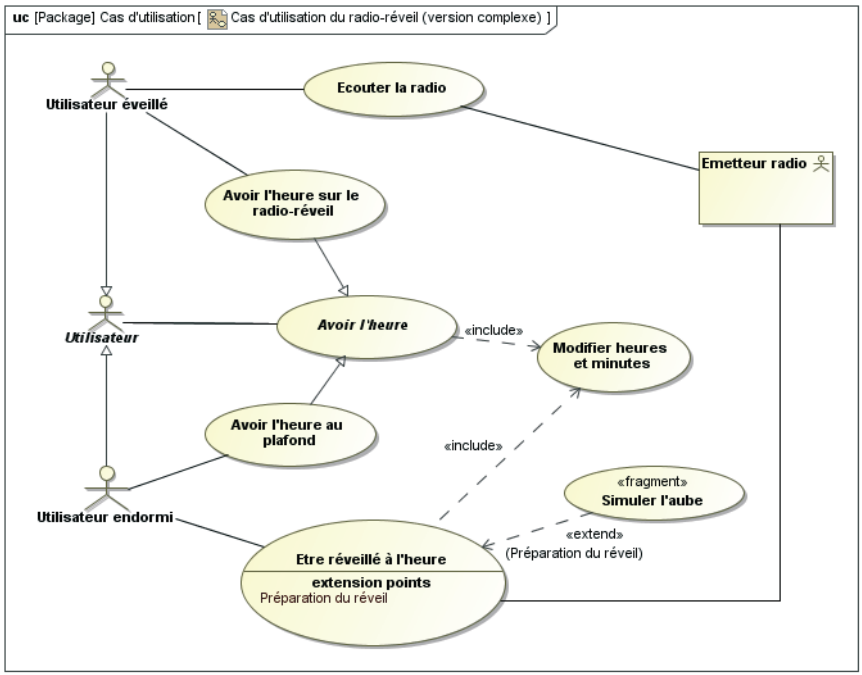


Figure 1–9 Version complexe (et peu recommandable !) du diagramme de cas d'utilisation

Partie I

La modélisation des exigences

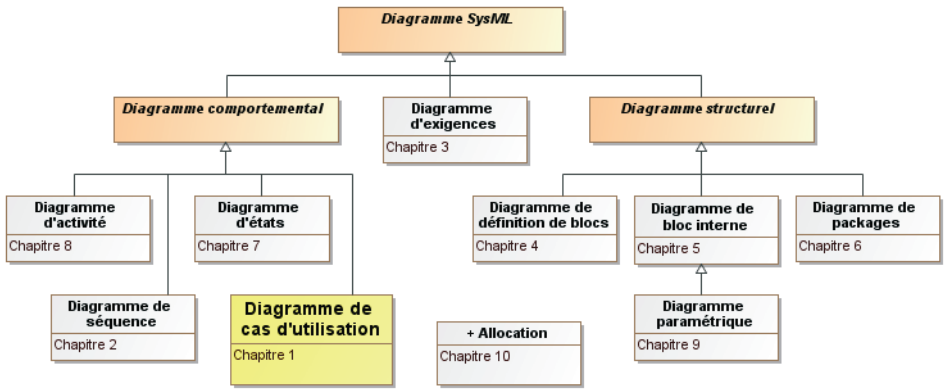
La partie I de l'ouvrage concerne la modélisation des exigences.

Nous apprendrons tout d'abord à mettre en œuvre la technique des cas d'utilisation, déjà présente en UML et très utilisée. Nous présenterons ensuite une première application du diagramme de séquence au niveau du système « boîte noire ». Nous verrons enfin que SysML innove en permettant de modéliser les exigences système et surtout de les relier ensuite aux éléments structurels ou dynamiques de la modélisation, ainsi qu'à des exigences de niveau sous-système ou équipement.

1

Le diagramme de cas d'utilisation

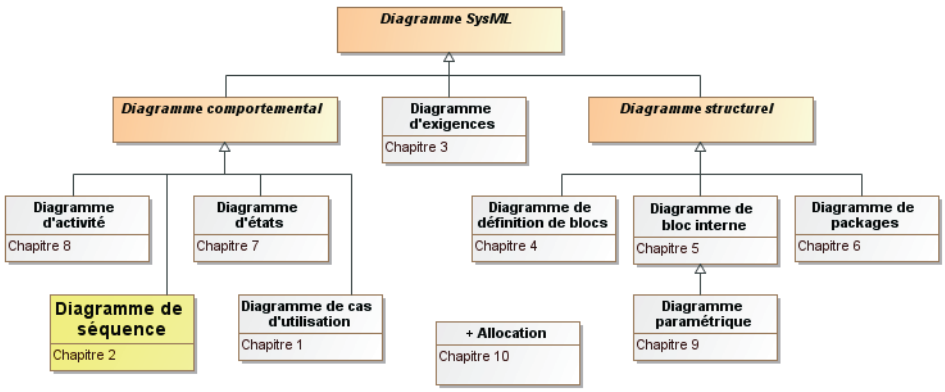
Ce chapitre présente le diagramme de cas d'utilisation, qui fournit une description de haut niveau des fonctionnalités du système.



2

Le diagramme de séquence

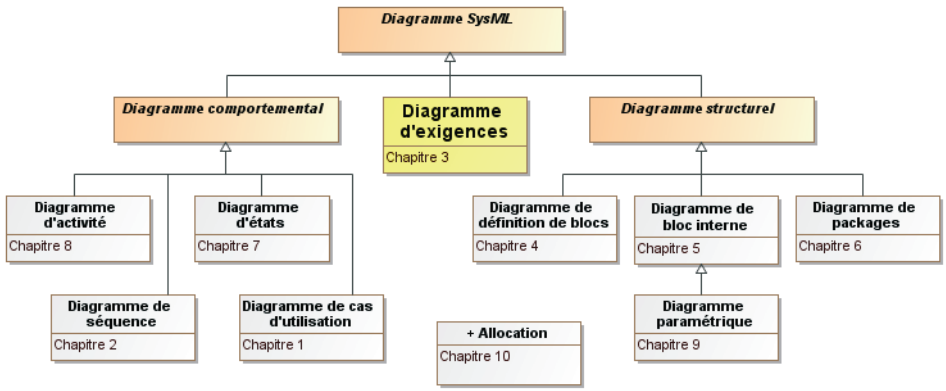
Ce chapitre présente le diagramme de séquence. Pour documenter les cas d'utilisation, la description textuelle est utile, car elle permet de communiquer facilement avec les utilisateurs et de s'entendre sur le vocabulaire « métier » employé. En revanche, le texte présente des désavantages puisqu'il est difficile de montrer comment les enchaînements se succèdent, ou à quel moment les acteurs secondaires sont sollicités. En outre, la maintenance des évolutions s'avère souvent fastidieuse. Il est donc recommandé de compléter la description textuelle par un ou plusieurs diagrammes de séquence SysML. Ces derniers seront ensuite utilisés pour montrer des interactions entre les éléments d'architecture, comme nous le verrons au chapitre 10.



3

Le diagramme d'exigences

Ce chapitre présente le diagramme d'exigences. Il capture les hiérarchies d'exigences, ainsi que leurs relations de dérivation, de raffinement, de satisfaction et de vérification. Ces relations fournissent la capacité de relier les exigences les unes aux autres, ainsi qu'aux éléments de conception et aux cas de tests.



Partie II

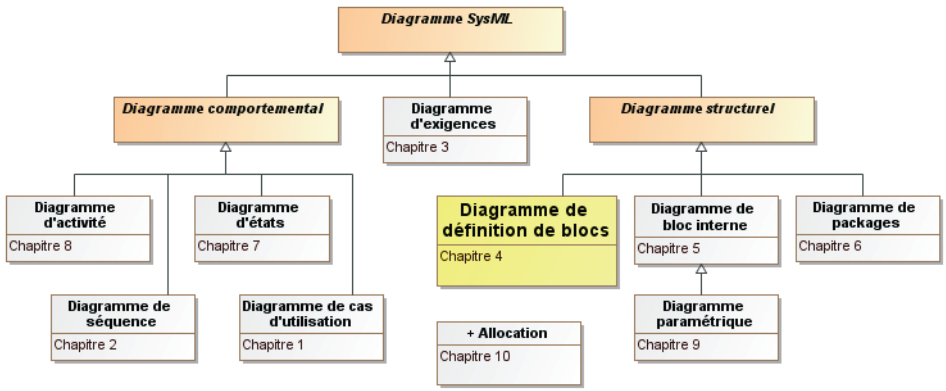
La modélisation d'architecture

La partie II concerne la modélisation structurelle. Nous apprendrons à utiliser le concept universel de « bloc » proposé par SysML pour modéliser tout élément structurel, ainsi que les deux types de diagrammes associés. Nous verrons tout d'abord comment définir les éléments structurels de base de notre modèle dans le diagramme de définition de blocs. Nous apprendrons ensuite à décrire la décomposition des éléments complexes avec le diagramme de bloc interne. Nous verrons enfin comment structurer notre modèle en packages, à des fins de travail en équipe ou de réutilisation.

4

Diagramme de définition de blocs

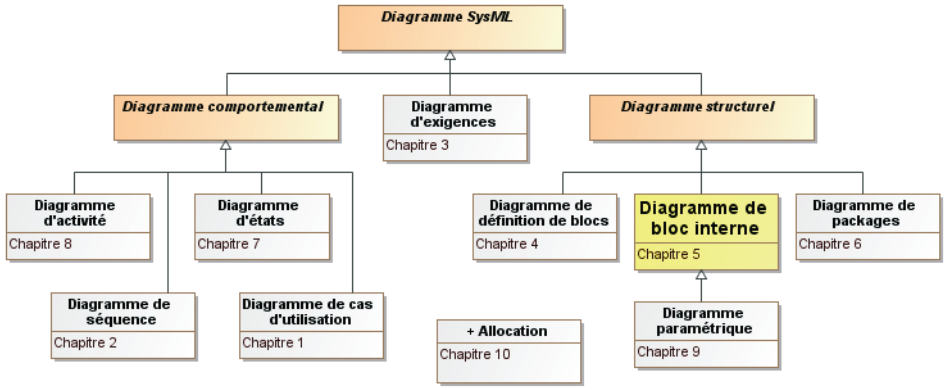
Ce chapitre présente le diagramme de définition de blocs. Le bloc SysML (block) *constitue la brique de base pour la modélisation de la structure d'un système. Il peut représenter un système complet, un sous-système ou un composant élémentaire. Les blocs sont décomposables et peuvent posséder un comportement. Le diagramme de définition de blocs (*block definition diagram *ou bdd) décrit la hiérarchie du système et les classifications système/composant.*



5

Le diagramme de bloc interne

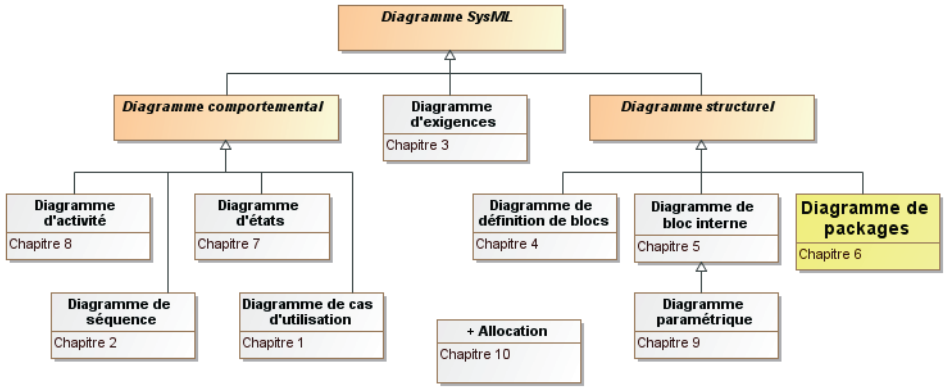
*Ce chapitre présente le diagramme de bloc interne (*internal block diagram *ou ibd), qui décrit la structure interne du système en termes de parties, ports et connecteurs.*



6

Le diagramme de packages

Ce chapitre présente le diagramme de packages, qui montre l'organisation du modèle et les éventuelles relations entre les packages.



Partie III

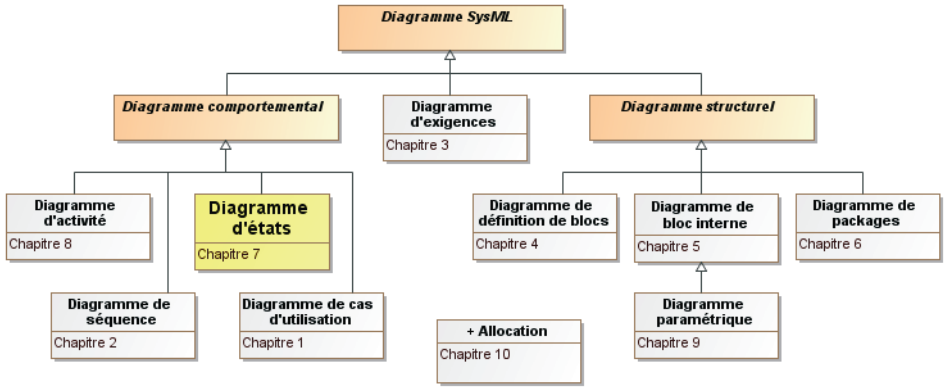
La modélisation dynamique

La partie III concerne la modélisation dynamique. Les diagrammes comportementaux incluent le diagramme de cas d'utilisation, le diagramme d'activité, le diagramme de séquence et le diagramme d'états. Nous avons déjà vu le diagramme de cas d'utilisation au chapitre 2 et le diagramme de séquence au chapitre 3. Nous présenterons dans cette partie toute la puissance du diagramme d'états, pour modéliser le cycle de vie des éléments fortement réactifs, ainsi que celle du diagramme d'activité, qui permet de modéliser avec précision des algorithmes complexes.

7

Le diagramme d'états

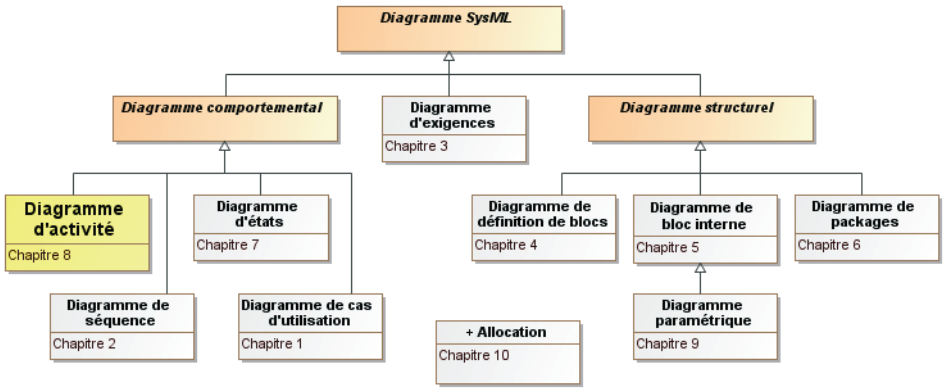
Ce chapitre présente le diagramme d'états, qui décrit les transitions entre états et les actions que le système ou ses parties réalisent en réponse aux événements.



8

Le diagramme d'activité

Ce chapitre présente le diagramme d'activité, qui figure les flots de données et de contrôle entre les actions. Il est utilisé majoritairement pour l'expression de la logique de contrôle et d'entrées/sorties.



Partie IV

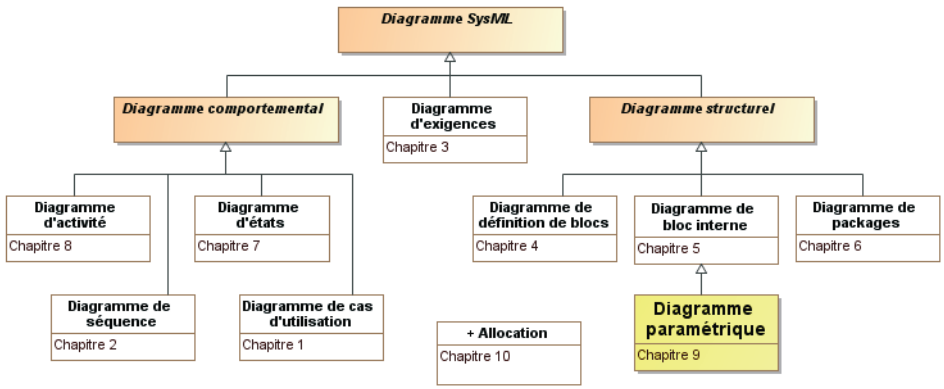
La modélisation transverse

La partie IV concerne la modélisation transverse. SysML permet de décrire plusieurs types de liens de traçabilité entre éléments de modélisation, et en particulier de mettre en œuvre le concept fondamental d'allocation. Nous verrons également comment décrire des équations grâce au nouveau diagramme paramétrique.

9

Le diagramme paramétrique

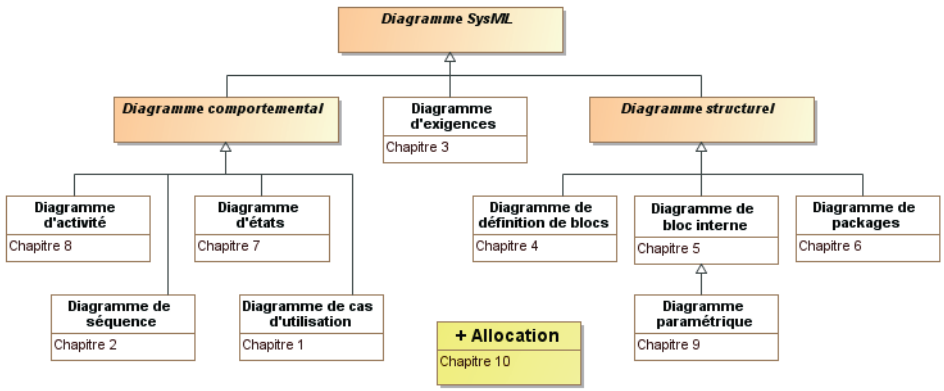
Ce chapitre présente le diagramme paramétrique, qui sert à figurer des contraintes sur les valeurs de paramètres système tels que performance, fiabilité, masse, etc. Il s'agit d'une spécialisation du diagramme de bloc interne où les seuls blocs utilisables sont des contraintes entre paramètres permettant de représenter graphiquement des équations et des relations mathématiques. Ce nouveau diagramme fournit ainsi un support précieux pour les études d'analyse système.



10

Allocation et traçabilité

Ce chapitre présente le concept d'allocation et ses possibilités de représentation. SysML inclut en effet un mécanisme général pour figurer différents types d'allocations, incluant l'allocation de fonctions à des composants, de composants logiques à des composants physiques, ainsi que du software au hardware.



Index