Modélisation SysML

Date : 30 septembre 2010

Auteur : Guillaume FINANCE

UML, langage de modélisation très répandu pour les développements logiciels, a été utilisé et adapté pour définir un langage de modélisation des systèmes : **SysML** ou **Systems Modeling Language**. Dans cet article nous présentons l'Ingénierie Système, un bref historique sur SysML, puis nous abordons l'utilisation des modèles SysML.

Le but de cet article est de présenter de façon non-exhaustive le langage SysML.

Ingénierie Système

L'**Ingénierie Système** (**IS**), ou Systems Engineering en anglais (SE), est une démarche méthodologique pour répondre à des problèmes complexes par la réalisation de solutions logicielles et matérielles.

L'Ingénierie Système s'adresse aux secteurs suivants : systèmes embarqués (ex : encodage/décodage audio et vidéo, set top box), automobile, ferroviaire, aéronautique, espace, militaire, télécoms, médical, production d'énergie, etc.

Les méthodes de l'Ingénierie Système (IS) reposent sur des approches de **modélisation** et de **simulation** pour valider les exigences ou pour évaluer le système. La modélisation a donc été couramment utilisée pour la décomposition fonctionnelle, les flux de données, et la décomposition structurelle du système, faisant appel à des techniques telles que :

- Le diagramme de flux de données (DFD ou Data Flow Diagram) pour définir les données traversant un système et leurs traitements éventuels
- Le diagramme de flux fonctionnel de bloc (FFBD ou Functional Flow Block Diagram) proche du diagramme UML d'activité

Les spécifications issues de l'IS relèvent souvent d'une documentation dense due à une **approche orientée documentation** (« document-based approach »), qui peut amener à une sélection inconsistante de différents types de diagrammes ou notations.

L'alternative consiste à effectuer une transition vers une « **approche orientée modèles** » (« model-based systems engineering » ou MBSE), permettant de réaliser un modèle cohérent du système, stocké et géré dans un référentiel avec un outil tel qu'Enterprise Architect de Sparx Systems.

Cette approche permet la réalisation d'un *ensemble organisé* de modèles, s'appliquant à différents niveaux de granularités tels que l'aspect opérationnel (contexte et utilisation du système), l'aspect fonctionnel (structure et sous-fonctions du système), et l'aspect physique (architecture). La modélisation permet de maitriser la complexité du système étudié, car chaque modèle donne accès à une représentation abstraite de différents aspects du système.

SysML : Historique

La modélisation avec le langage **UML** est une pratique bien établie dans l'industrie logicielle. Bien que le langage UML permette par son caractère à usage général d'adresser de nombreux besoins pour l'IS, il est nécessaire de l'adapter par la définition de « profils UML ». Plusieurs projets ont été menés dans ce cadre sur des domaines spécifiques : MARTE, System on a Chip.

Le besoin de définir un langage basé sur UML pour l'IS a été initié en 2001 par l'organisation internationale de l'ingénierie système **INCOSE**, qui s'est mise en relation avec l'**OMG**, organisme responsable d'UML. Dès lors, plusieurs membres de l'industrie (BAE, Motorola, Boeing…), éditeurs d'outils (IBM, Sparx Systems…), universités et organisations ont travaillés sur la définition du langage de modélisation système **SysML ou Systems Modeling Language**.

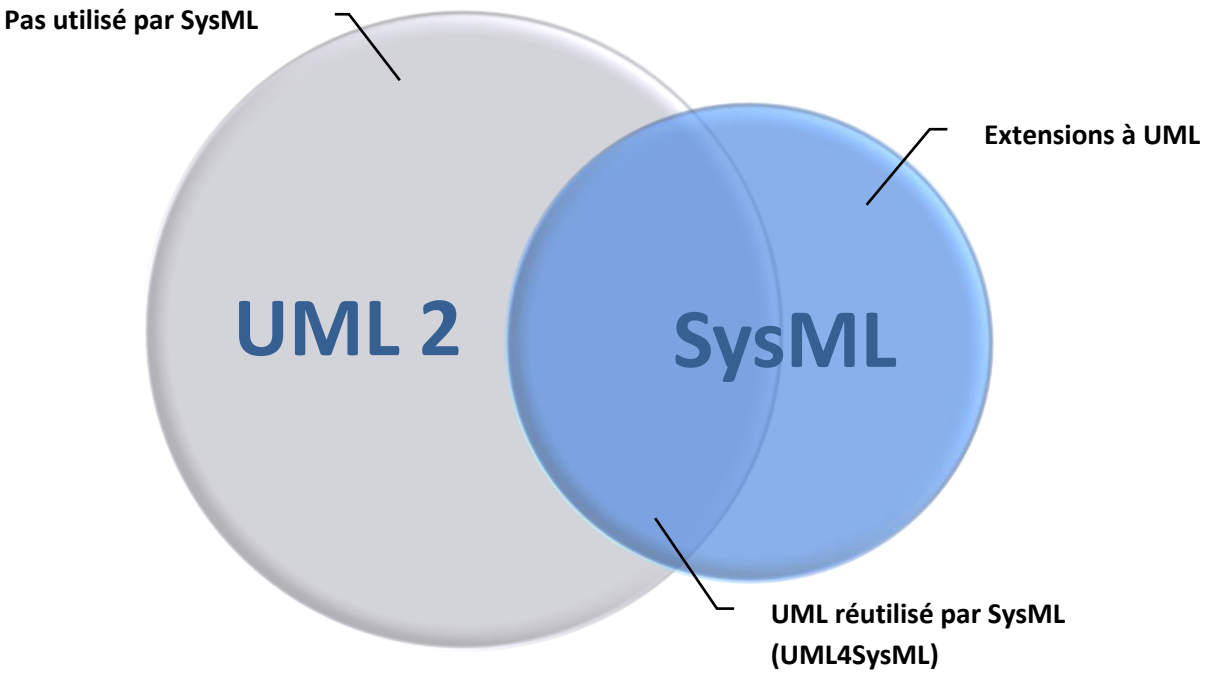
- 07/2006 : OMG annonce l'adoption de SysML
- 09/2007 : SysML v1.0
- 12/2008 : SysML v1.1
- 06/2010 : SysML v1.2 (version actuelle)

SysML : Présentation

SysML est basé sur UML et remplace la modélisation de classes par des **blocs** pour un vocabulaire plus adapté à l'Ingénierie Système. Un bloc englobe tout concept logiciel, matériel, données, processus, et même gestion des personnes.

Comme indiqué sur le diagramme suivant, SysML réutilise une partie d'UML2 (UML4SysML), et apporte également ses propres définitions (extensions SysML). SysML n'utilise donc pas les treize diagrammes d'UML2, proposant ainsi un ensemble de diagrammes adaptés à l'IS.

SysML offre l'avantage d'être accessible aux développeurs logiciels qui vont retrouver de nombreuses similitudes avec UML2. SysML permet de produire des spécifications dans un langage unique pour des équipes hétérogènes, responsables de la réalisation des blocs matériels et logiciels. Les connaissances sont ainsi modélisées dans un référentiel unique qui améliore la communication entre les différentes équipes participantes, est accessible à tous, et permet la réutilisation des blocs réalisés.



SysML propose les diagrammes suivants :

- Diagrammes structurels
  - Le « Block Definition Diagram » (BDD) remplace le diagramme de classes
  - L' « Internal Block Diagram » (IBD) remplace le diagramme de structure composite
  - Le diagramme paramétrique est une extension SysML pour l'analyse de paramètres critiques du système
  - Le diagramme de paquetage reste inchangé
- Diagrammes dynamiques
  - Le diagramme d'activités est légèrement modifié pour SysML
  - Les diagrammes de séquence, d'états, et de cas d'utilisations restent inchangés
- Le diagramme d'exigences est une extension SysML

Cet article se poursuit par un aperçu de chacun des diagrammes proposés par SysML, en commençant par la structure du système, suivi de la modélisation dynamique, et en terminant par les nouveautés de SysML.

Cet ordre ne définit en aucun cas la méthodologie à suivre pour modéliser un système ; le sujet de cet article est la présentation du langage SysML et non les méthodologies appliquées aux projets.

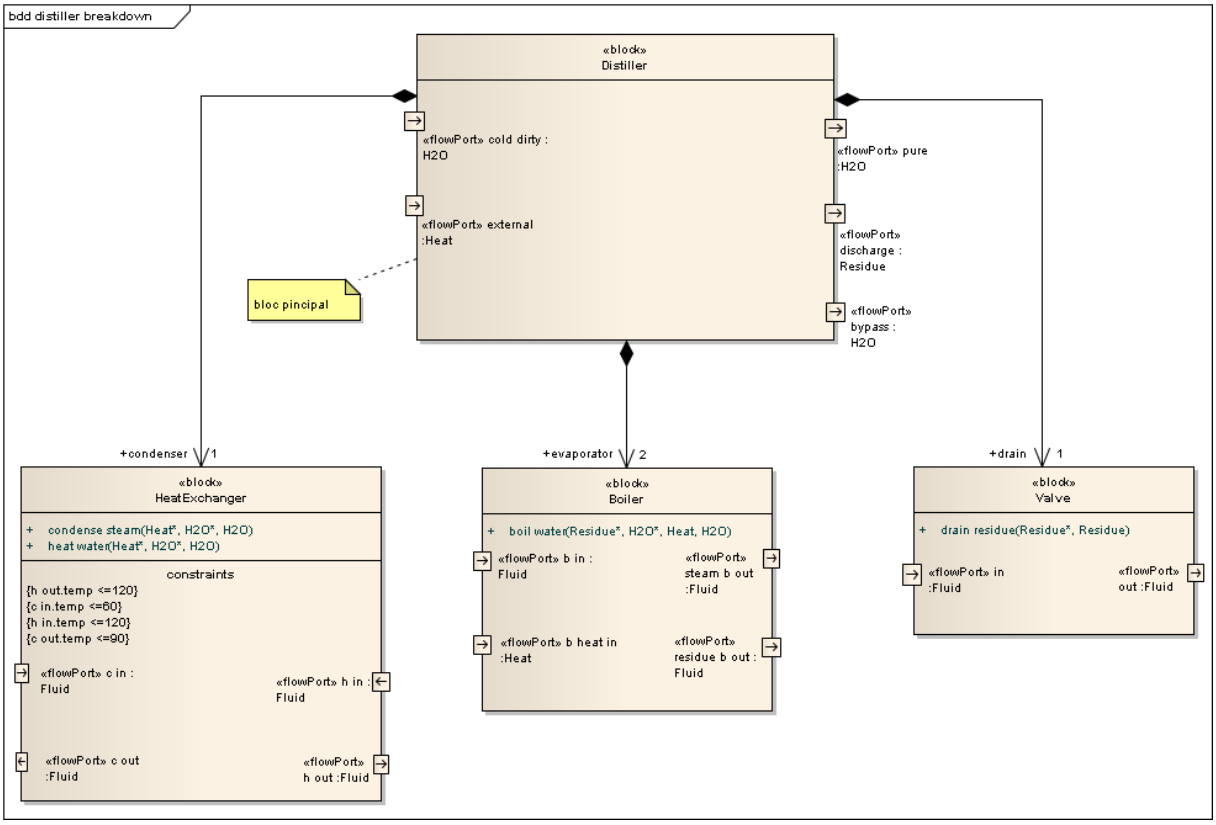
Modélisation structurelle

BDD : Block Definition Diagram

Le diagramme de définition de bloc (BDD, ou Block Definition Diagram en anglais) représente la vue boîte noire d'un bloc. Ainsi le **bloc principal** et la hiérarchie des blocs qui le composent, qu'ils soient logiciels ou matériels, sont spécifiés dans ce diagramme.

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.

Par rapport à UML, le BDD de SysML redéfinit le diagramme de classe en remplaçant les classes par des blocs. Le diagramme BDD ci-dessous provient de l'exemple OMG du purificateur d'eau.



Informations liées au BDD :

- Les blocs sont représentés par des classes UML stéréotypées « block ».
- Le bloc principal définit le purificateur d'eau (Distiller) composé de 3 blocs :
  - un échangeur de chaleur (HeatExchanger) qui a un rôle de condensateur (condenser)
  - deux bouilloires (Boiler) qui ont tous deux un rôle d'évaporateur (evaporator)
  - une soupape (Valve) qui a un rôle de drain
- Les trois blocs font physiquement partie du bloc principal, car les liens utilisés sur le diagramme sont des agrégations fortes ou compositions, représentées par un losange plein. Si un bloc n'en faisait pas physiquement partie, on parlerait alors d'une *référence*, et l'association utilisée serait une agrégation simple, représentée par un losange vide.
- Le port de flux (flow port) est une nouveauté SysML ; il représente ce qui peut circuler en entrée et/ou en sortie d'un bloc, que ce soit des données, de la matière ou de l'énergie. Ainsi le BDD indique que les entrées du bloc « Distiller » sont de l'eau froide et de la chaleur externe.

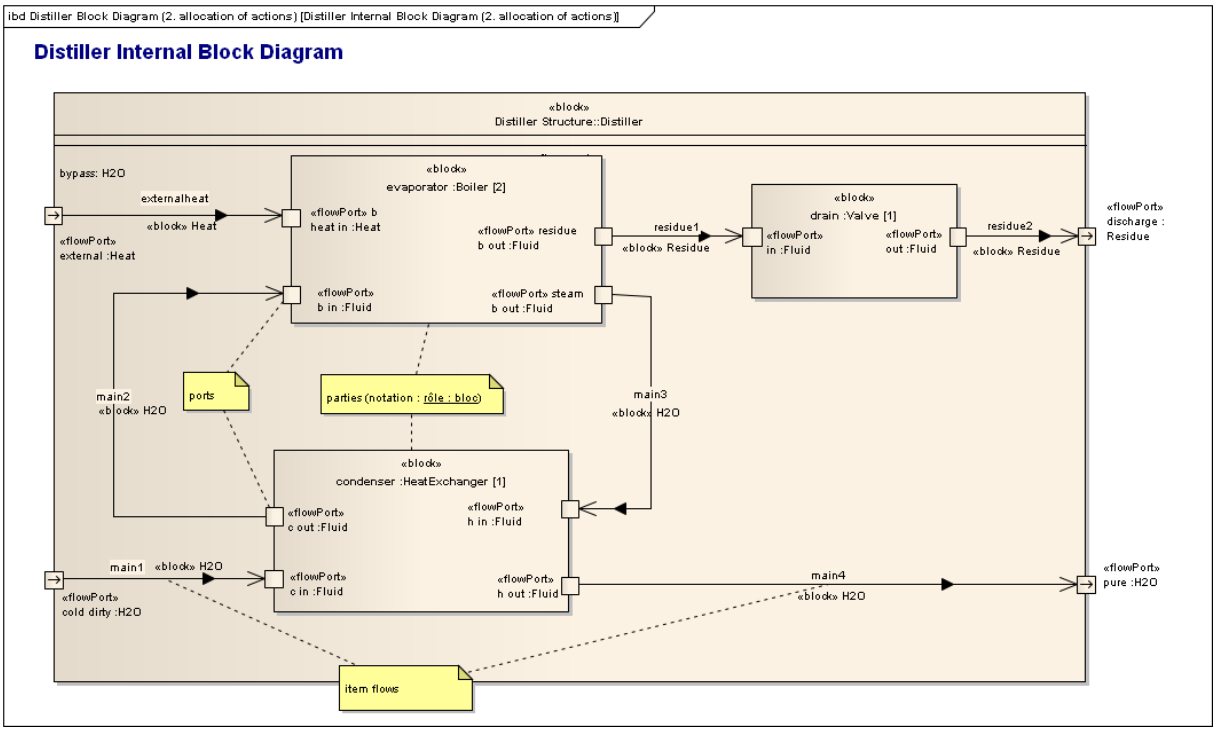
IBD : Internal Block Diagram

Le diagramme de bloc interne (IBD, ou Internal Block Diagram) décrit la vue interne d'un bloc ou vue boîte blanche, et se base sur le BDD pour représenter l'assemblage final des blocs qui composent le bloc principal.

Les blocs composant le bloc principal et définis dans le BDD, sont **instanciés** en **parties**. Ces parties sont assemblées par des **connecteurs**, les reliant directement ou au travers de leurs **ports** (ports standards avec interfaces exposées et/ou ports de flux).

Par rapport à UML2, l'IBD de SysML redéfinit le diagramme de structure composite en ajoutant entre autre les ports de flux.

Exemple de diagramme IBD pour le purificateur d'eau.



Informations liées à l'IBD :

- Le bloc Distiller du BDD est copié sur ce diagramme
- Les blocs du BDD qui composent le Distiller sont instanciés en **parties** sur l'IBD, et sont intitulées comme suit : « rôle : nom du Bloc ». Le rôle d'une partie doit être cohérent avec les associations d'agrégations du BDD ; ainsi on retrouve le rôle « drain » défini sur l'agrégation du bloc Valve du BDD dans l'IBD via la partie « drain : Valve ».
- La **multiplicité** spécifiée sur une agrégation du BDD est cohérente avec l'IBD, dont la multiplicité est représentée sur les parties entre crochets, par exemple : « evaporator : Boiler [2] »
- Tous les ports du bloc principal sont reliés aux ports des parties internes par des connecteurs. Par exemple l'eau froide du Distiller est transmise en entrée à l'échangeur de chaleur.
- La direction d'un flow port peut être définie en entrée, sortie, ou entrée/sortie.
- Alors que les ports indiquent ce qui peut passer par un bloc, les **flux d'éléments** ou « **item flows** » définissent ce qui passe entre deux blocs reliés par un connecteur. Par exemple l'IBD spécifie que l'élément « externalHeat », de type Heat (bloc), circule entre le port d'entrée du Distiller et le port d'entrée du Boiler.

Value Types

Les Value Types sont une nouveauté SysML pour définir des types de valeurs réutilisables par des propriétés du modèle, par exemple des blocs. De façon similaire à la modélisation UML où les attributs de classes peuvent être typés par d'autres classes, SysML permet de définir des propriétés de blocs typées avec des Value Type.

La Value Type a la particularité de contenir deux propriétés optionnelles : une **unité** et une **dimension**. Exemple : Value Type « °C » définie avec unit = 'degrés Celsius' et sans dimension. La unit « degrés Celsius» est définie avec dimension = 'température'.

Diagramme de paquetage

Le diagramme de paquetage permet de structurer le modèle tout comme avec UML.

Modélisation dynamique

La modélisation de l'aspect dynamique du système avec SysML repose sur une sélection de quatre diagrammes UML2 : diagrammes de cas d'utilisations, de séquence, d'activité, et d'états.

Parmi ces diagrammes, seul le diagramme d'activité comporte quelques modifications pour SysML.

Diagramme de cas d'utilisations

Tout comme avec UML, on utilise les diagrammes de cas d'utilisations pour identifier les acteurs et les cas d'utilisations d'un point de vue utilisation du système, interactions acteurs/système.

Les cas d'utilisations permettent de définir les frontières, le périmètre fonctionnel du système.

Diagramme de séquence

Le diagramme de séquence représente les éléments intervenant dans un scénario ou un processus, ainsi que les messages échangés dans un ordre chronologique.

Les éléments intervenant sont représentés par des **lignes de vie** (lifetime en anglais) pouvant être des instances de blocs du modèle. Cette instanciation de blocs établit un lien avec la modélisation structurelle du système donc une cohérence dans le modèle :

- On peut accéder aux propriétés du bloc d'une ligne de vie
- Chaque message échangé peut être utilisé pour l'identification des opérations de blocs

L'ensemble des propriétés du diagramme de séquence utilisées en UML sont également disponibles avec SysML : messages synchrones ou asynchrones, opérateurs (ex : alt, loop, opt, par), références vers d'autres diagrammes de séquence, etc.

Diagramme d'activité

Le diagramme d'activité est utilisé pour représenter les étapes d'un processus, impliquant en général les « input et output pins » qui correspondent respectivement au type d'élément requis en entrée d'une activité ou action, et à celui généré en sortie.

Si une action ou activité correspond à l'opération d'un bloc, il est alors possible de vérifier que les types d'éléments définis en entrée et en sortie de cette activité soient cohérents avec la signature de l'opération du bloc ou de son interface.

Toutes les propriétés des diagrammes d'activités UML sont également disponibles avec SysML.

SysML a rajouté quelques spécificités :

- Notion de contrôle pour activer ou désactiver les actions en cours (Control Value)
- Spécification de la nature du débit sur le flot : système continu ou discret
- Définition de taux et de probabilité sur les flux de contrôle ou d'objets

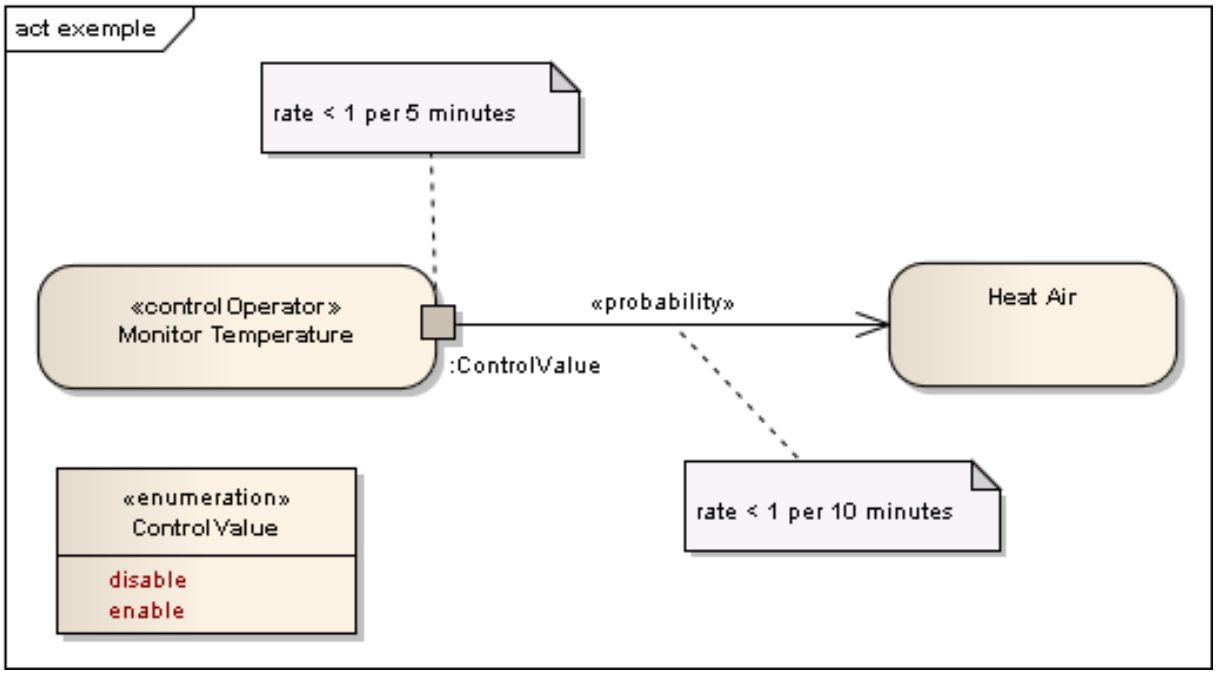


Diagramme d'états

Le diagramme d'états est utilisé avec SysML de la même manière qu'avec UML2, c'est-à-dire qu'il permet de représenter le cycle de vie auquel doivent se conformer toutes les instances d'un bloc donné, ce au travers de la modélisation de tous les états possibles.

Seuls les blocs qui sont importants d'un point de vue métier, ou qui sont de nature complexe, devraient avoir un diagramme d'état.

Les propriétés du diagramme d'état UML sont également disponibles avec SysML : conditions sur évènements, effets, activité durable, transitions, états composites, régions concurrentes, etc.

Nouveautés SysML

Exigences

Que ce soit pour l'ingénierie système ou uniquement pour des réalisations logicielles, les exigences sont couramment utilisées pour formaliser les pré-requis du système, se traduisant par des

fonctionnalités ou conditions qui doivent ou devraient être satisfaites par le système (selon les éventuelles priorités associées aux exigences).

Pour la **maîtrise d'ouvrage** (MOA), les exigences ont pour objectif d'assurer l'adéquation de la solution (le système réalisé) avec les besoins. Les exigences peuvent être formalisées et catégorisées, par exemple différenciant les exigences fonctionnelles des exigences techniques (performance, fiabilité, ergonomie, etc.).

La formalisation des exigences peut être effectuée avec une feuille Excel, ou avec un outil spécialisé tel que DOORS ou CaliberRM. L'intérêt qu'offrent ces outils est la gestion des exigences dans une organisation structurée. Les exigences sont également utilisées pour la modélisation, par la création d'associations entre exigences et cas d'utilisations, blocs ou tout type d'élément du modèle, établissant la **traçabilité du modèle**. Il est possible avec l'outil Enterprise Architect de définir les exigences ou des les importer depuis un outil tel que DOORS, et de les associer avec les éléments du modèle.

SysML formalise les exigences et leur représentation, s'inspirant des fonctionnalités des outils actuellement disponibles sur le marché. Ainsi **SysML définit une représentation graphique et visuelle des exigences textuelles, leur organisation hiérarchique, et des associations avec les éléments du modèle.**

SysML définit aussi de nouveaux types de d'associations (liens de dépendance stéréotypés) :

- *Derive* : une ou plusieurs exigences dérivent d'une exigence
- *Satisfy* : un ou plusieurs éléments du modèle permettent de satisfaire une exigence
- *Verify* : un ou plusieurs éléments du modèle permettent de vérifier et valider une exigence
- *Refine* : un ou plusieurs éléments du modèle, par exemple un cas d'utilisation, redéfinissent une exigence

SysML définit de nouveaux commentaires stéréotypés permettant d'associer une explication à des associations ou éléments du modèle :

- *Problem* : commentaire dont la description pose le problème ou le besoin
- *Rationale*: commentaire dont la description indique la raison ou la justification par rapport à l'élément ou l'association associé

Exemple de représentation des exigences sous EA pour le système de purificateur d'eau :

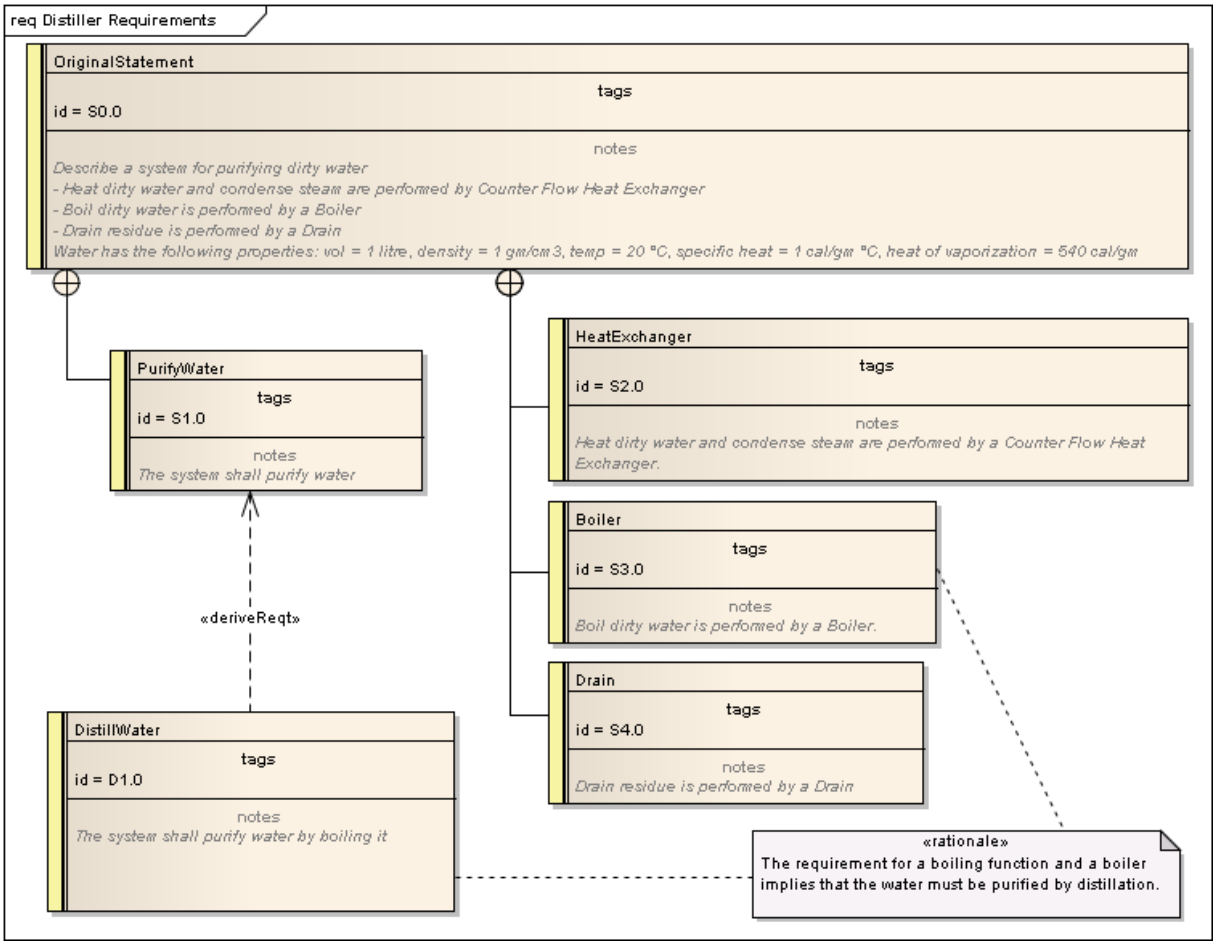


Diagramme paramétrique

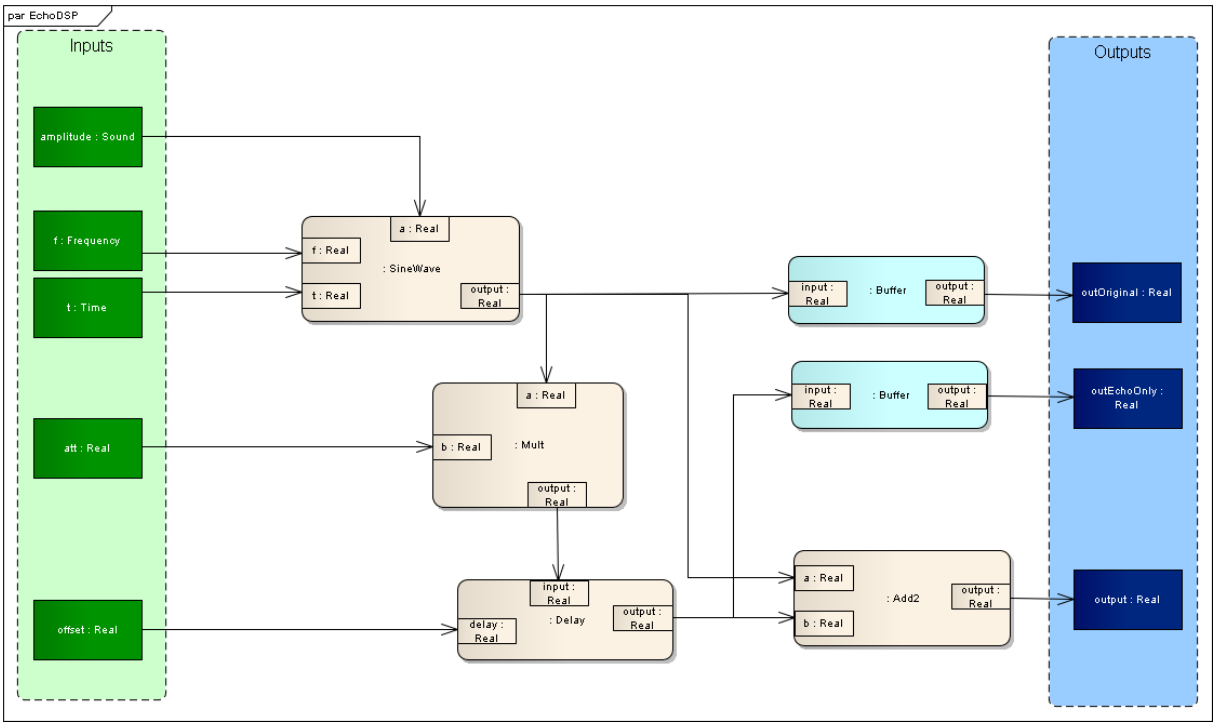
Le diagramme paramétrique permet d'intégrer des analyses systèmes (performance, fiabilité, etc.) par des **blocs de contrainte**.

Un bloc de contrainte représente une expression mathématique dont les paramètres peuvent faire référence à des éléments du système, par exemple aux propriétés de blocs.

Dans un premier temps, de façon similaire à la création du diagramme BDD, les blocs de contraintes sont définis dans un diagramme de classe. Un diagramme paramétrique peut alors être créé :

- Les blocs de contraintes sont instanciés, donnant lieu aux **propriétés de contrainte** (ou **constraint property**) qui héritent ainsi de leurs paramètres (note : il n'y a pas de différentiation entre paramètres d'entrée et paramètres de sortie)
- Des propriétés systèmes (optionnellement liées à des blocs)
- Des connecteurs reliant l'ensemble des propriétés systèmes et paramètres des propriétés de contrainte

Diagramme paramétrique basé sur un exemple de lecteur MP3 proposé par l'éditeur Sparx Systems (Enterprise Architect) :

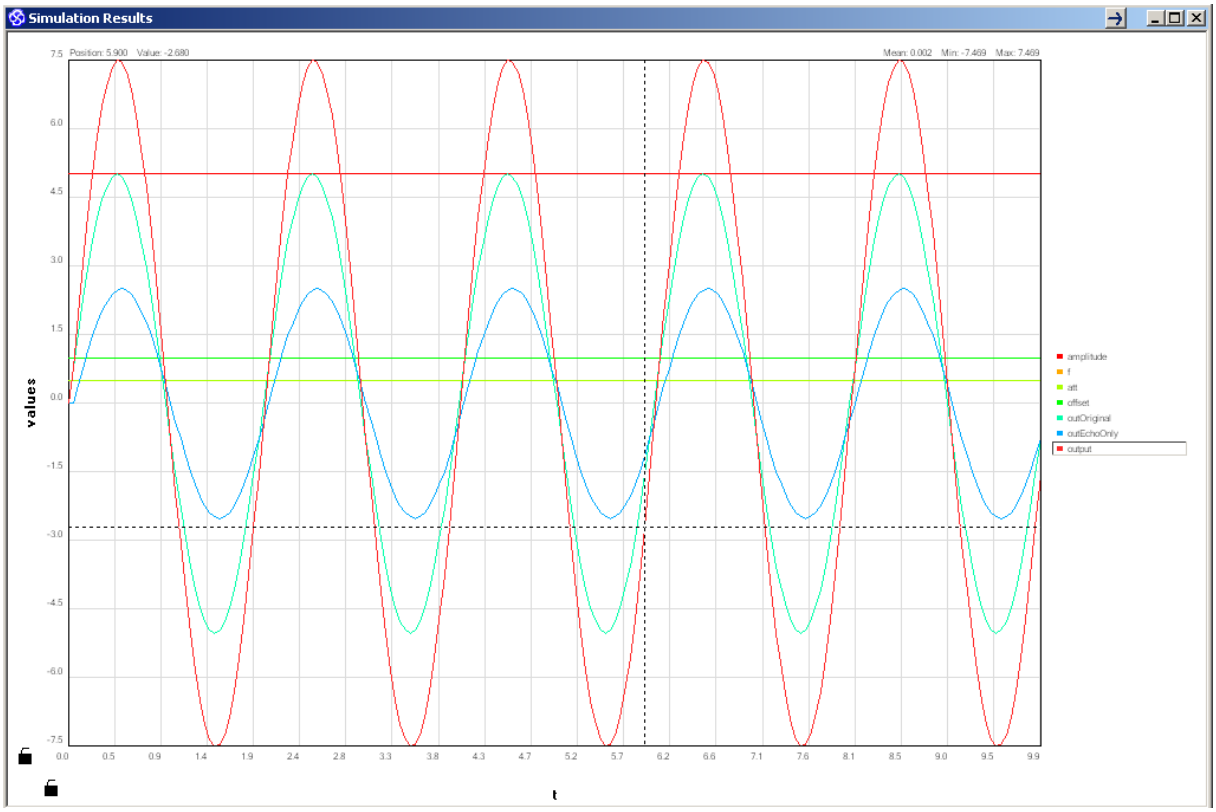


Informations liées au diagramme paramétrique :

- Le diagramme comporte six propriétés de contrainte.
- Les paramètres systèmes sont catégorisés entre entrées (vert) et sorties (bleues). Ces paramètres sont associés aux paramètres des propriétés de contrainte (par exemple la fréquence d'entrée f est reliée au paramètre f de la propriété de contrainte SineWave)
- Certaines propriétés de contrainte sont reliées entre elle (ex : le paramètre SineWave.output est relié à Buffer.input et à Add.a)

Certains outils permettent d'utiliser ce diagramme dans le cadre de **simulation**.

Il est possible avec Enterprise Architect de saisir des expressions pour chacun des blocs de contraintes (par exemple en VBScript ou JavaScript), ainsi que de renseigner les valeurs des paramètres systèmes. L'exécution du module de simulation est illustrée ci-dessous :



Allocations

Le concept d'allocation est repris du vocabulaire des ingénieurs systèmes, indiquant un ensemble d'éléments associés dans un environnement structuré. La modélisation système implique des tentatives d'allocations entre éléments du système. Cet ensemble d'allocations est utilisé pour générer une matrice pour vérifier que les parties du système sont correctement intégrées.

La création d'allocations permet de maintenir une cohérence entre les éléments du système, en particulier entre les modèles dynamiques et les modèles structurels.

Outils SysML

Le langage SysML a été intégré par de nombreux éditeurs d'outils commerciaux ou open source:

- Sparx Systems Enterprise Architect (plugin SysML ou version Ultimate requise)
- IBM Rational Software Modeler (plugin d'une société tierce disponible)
- MagicDraw (plugin SysML requis)
- Open source : TopCased (environnement Eclipse)