Modélisation des exigences en UML/SysML

Nicolas Belloir, Jean-Michel Bruel, Raphaël Faudou

Modélisation des exigences en UML/SysML

Nicolas Belloir, Jean-Michel Bruel et Raphaël Faudou

Résumé

L'ingénierie des exigences est une activité primordiale dans tout développement de systèmes. En ingénierie système (IS) tout ingénieur est formé à la capture et à l'ingénierie des exigences. L'INCOSE et l'OMG, quand ils ont développé SysML [3] à partir d'UML [1], ont introduit explicitement la notion d'exigence comme élément de modélisation. Un diagramme ainsi qu'une table des exigences leur sont même dédiés. Nous expliquons dans cet article comment les exigences sont modélisées, et surtout comment elles sont liées avec le reste de la modélisation structurelle et comportementale. Nous apportons notre analyse sur les manques constatés par les industriels et dressons quelques pistes d'améliorations futures.

Mots-clés : Exigences, ingénierie système, modèles, UML, SysML, traçabilité, MBSE

1. INTRODUCTION

La phase de recueil des exigences est une étape clé pour la réussite de tout projet. Elle vise à faire une traduction/représentation des besoins du client en un formalisme plus précis qu'un cahier des charges par exemple. Elle pose les prémices de la spécification. En ingénierie logicielle, le langage de modélisation standard *de facto* est UML [1]. Force est de constater qu'il ne dispose pas (ou peu) de moyen de formaliser et surtout de gérer les exigences, ce qui impose souvent de passer par un logiciel spécialisé tel que *Doors*^{TM}. À l'opposé de ce qui se pratique parfois dans le domaine du logiciel, l'ingénierie système, de par la complexité des systèmes qu'elle traite, ne peut faire l'économie d'un moyen de prise en compte des exigences. Elle doit également passer d'approches purement centrées documentation textuelle vers des approches utilisant des langages graphiques et orientées modèles [2]. Dans ce cadre, l'OMG et l'INCOSE ont développé le langage SysML^{TM} [3] comme une extension d'UML dédiée aux systèmes complexes. Il s'agit d'un langage généraliste pour l'ingénierie système. Les retours d'expérience industriels ont montré qu'il a été utilisé avec succès dans les domaines de l'aéronautique (par exemple, Airbus), du spatial (par exemple, CNES, Airbus Defence and Space), du ferroviaire (par exemple, Alstom), etc. Nous renvoyons le lecteur pour plus de précision à la série de conférences organisées par l'association SysML-France dont les présentations sont en ligne [4]. SysML se veut un langage dédié aux ingénieurs systèmes avec leurs préoccupations, leur culture et leurs contraintes propres qui diffèrent de celles d'un logiciel : il n'est pas besoin de typer toutes les données (car on ne va pas générer du code), il n'y a pas de concepts objets ou très peu, car l'accent est davantage mis sur la spécification que sur la description fine du comportement. Il permet la prise en compte d'un univers plus proche de la physique avec des flux continus notamment et de l'énergie, et ne se limite pas seulement à des données discrètes. Il permet la formalisation et la gestion des exigences via notamment un diagramme spécialisé et une table des exigences. Il offre également des moyens de traçabilité entre les différents éléments des modèles permettant de modéliser la couverture des exigences par le modèle. SysML s'inscrit donc pleinement dans les préconisations de l'INCOSE [5] ou plus récemment du rapport Potier sur le logiciel embarqué [6] : « [...] une véritable industrialisation des processus d'ingénierie basée modèle est indispensable, avec une définition claire des transitions système / logiciel / hardware et une prise en compte des différents partages industriels au sein de plateformes de développement cohérentes et communes et implique donc le choix de standards communs [...] ».

Le présent article décrit les concepts de la gestion des exigences de SysML, discute de la manière de les utiliser, puis résume les outils disponibles liés à ce langage. En conclusion, sont rappelées les limitations et les pistes d'évolution future.

2. PRÉSENTATION DES CONCEPTS

Définition

SysML définit une exigence comme une aptitude ou une condition qui doit être satisfaite :

> " *A requirement specifies a capability or condition that must (or should) be satisfied. A requirement may specify a function that a system must perform or a performance condition a system must achieve.*" [3, p.139].

Le langage fournit pour cela des artefacts graphiques de modélisation basés sur l'expression des exigences sous forme textuelle, d'une part, et sur les relations entre ces exigences et les autres éléments du modèle, d'autre part. Le diagramme des exigences a pour but de fournir une représentation graphique des exigences. Il peut être complété ou venir en complément d'une représentation tabulaire des exigences.

Nous présentons dans la Figure 1 une représentation partielle du métamodèle des exigences.
Une exigence est construite comme une classe en appliquant un stéréotype nommé <<*requirement*>>. Il est caractérisé *a minima* par un identifiant unique (*id*) et une zone de texte décrivant l'exigence (*text*). En effet, il est possible d'ajouter des propriétés aidant à mieux caractériser une exigence. Certains outils intègrent nativement dans les exigences des propriétés telles que la priorité (haute, basse), le risque, la source (technique, marketing, légale), le statut (approuvée, proposée), la méthode de vérification, etc.

Organisation
L'ingénierie des exigences aboutit généralement à un ensemble organisé d'exigences, que ce soit en termes de fonctionnelles/non fonctionnelles, de prioritaires/secondaires, etc. Le principal support de SysML à cette organisation, outre la possibilité de les annoter par des stéréotypes, consiste à utiliser les paquetages (*packages*). Plusieurs types d'organisation sont possibles : par niveau d'abstraction, par point de vue, etc. À noter que la spécification SysML évoque dans ses annexes (non normatives) un exemple de classification des exigences (*performance*, *functional*, *system*) correspondant à ce que suggère l'INCOSE [7] en appliquant un profil. Cette organisation en paquetage est héritée d'UML et est pratique pour séparer les préoccupations. Par ailleurs, SysML introduit la notion de <<*ViewPoint*>> qui permet de spécialiser la notion de paquetage. On peut par exemple avoir un point de vue « *requirement analysis* » pour rester au plus près des phases identifiées par l'INCOSE. Ainsi, c'est plutôt l'approche « stéréotype » qui sera utilisée pour typer des exigences tandis que pour faire face à leur complexité (grand nombre par exemple) on utilisera plutôt les paquetages afin de rester dans des ordres de grandeurs plus faciles à gérer et à se partager en équipe.

Relations entre exigences
Les exigences peuvent être organisées et liées entre elles de différentes manières. Tout d'abord, il est possible de décomposer une exigence en sous-exigences. On utilise pour cela la relation de contenance (*containment*) d'UML. Cela permet de décomposer une exigence complexe en une hiérarchie de sous-exigences. Pour que l'exigence complexe soit vérifiée, il faudra que l'ensemble des sous-exigences le soit. Il est possible ensuite de dire d'une exigence qu'elle est dérivée (« déclinée » selon la traduction officielle) d'une autre. On utilise pour cela la relation <<*deriveReqt*>>. Cela permet lors de l'analyse de déterminer un ensemble d'exigences dérivant d'une exigence source, notamment à des niveaux d'analyse plus fins. La relation <<*satisfy*>> décrit de son coté qu'un élément de modèle de conception ou d'implémentation satisfait à une ou plusieurs exigences. Il ne faut pas la confondre avec la relation <<*verify*>> qui définit quel cas de test (élément de modèle stéréotypé <<*testCase*>>) vérifie une exigence. En modélisation système, il peut être nécessaire de réutiliser une exigence. C'est le cas par exemple lorsque ces dernières sont issues des standards (ARP4754, CENELEC, ECSS, ...) ou normes (ISO, ...). Ou encore dans le cas de contraintes maison sur la conception (maintenabilité, coût, réutilisabilité, ...). Dans ce cas l'exigence recopiée l'est sans modification possible et, pour cela, SysML définit la relation <<*copy*>> permettant de matérialiser qu'une exigence est directement réutilisée et est donc non modifiable. Pour finir, il est possible d'utiliser la relation générique <<*trace*>> pour lier une exigence à tout autre élément de modèle. Cependant la sémantique de cette relation est peu définie et il convient de l'utiliser en dernier recours en lui préférant les autres relations ci-avant.

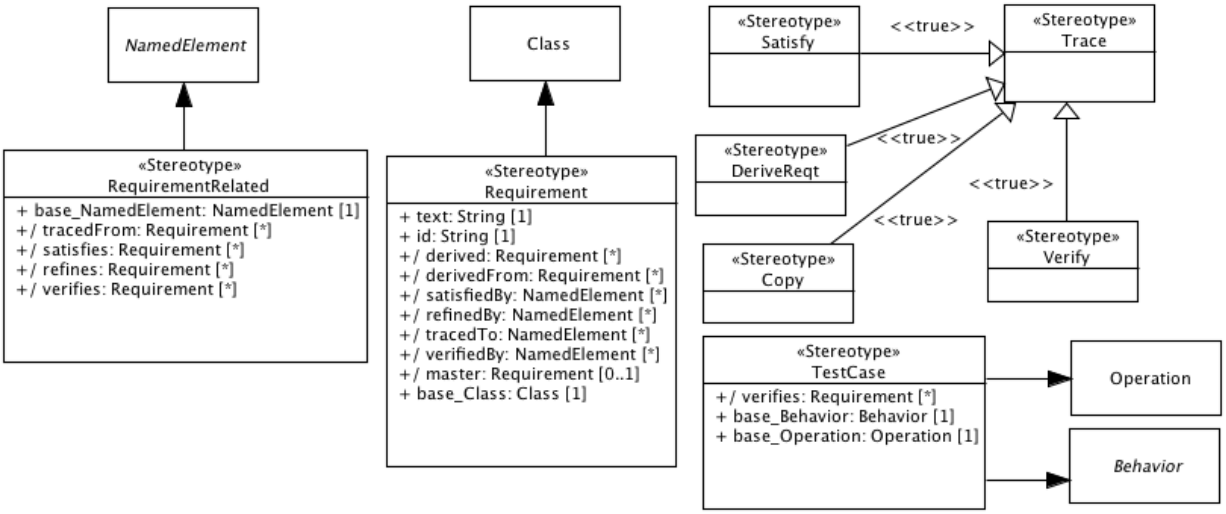

Figure 1 : Métamodèle des exigences SysML de Papyrus

Annotations et stéréotypes

Il est possible d'apporter un complément d'information sur les éléments d'un modèle à l'aide d'annotations. SysML propose deux annotations standard utilisables de façon transverse dans tous les modèles SysML : *rationale* pour en préciser les raisons d'un choix de modélisation et *problem* pour identifier un problème potentiel anticipé.

Profitant des capacités d'extension d'UML, il est possible d'étendre le champ sémantique des exigences en définissant des types particuliers d'exigences. Pour cela il faudra utiliser le mécanisme des profils UML en définissant de nouveaux stéréotypes. On pourra, par exemple, définir deux stéréotypes <<*Functional Requirement*>> et <<*Non Functional Requirement*>> pour respectivement identifier des exigences fonctionnelles et non fonctionnelles.

Illustration

Nous présentons ici un exemple d'utilisation des exigences en SysML. La partie du système modélisée concerne un réseau de capteur déployé dans un environnement dans lequel les capteurs doivent être autonomes et donc économiser leur batterie. Un appareil mobile sert à la collecte des données relevées par le réseau de capteurs. Il s'identifie auprès des capteurs lorsqu'il rentre dans leur périmètre réseau. La Figure 2 a été construite à partir d'une exigence initiale appelée *Longévité*. Celle-ci a donné deux sous-exigences concernant, d'une part, la limitation des communications et, d'autre part, la limitation de l'alimentation des appareils de mesure. À partir de l'analyse de l'exigence concernant la limitation de la communication, a été dérivée une exigence spécifiant que, pour les communications entre le collecteur et un capteur, il ne doit y avoir qu'une et une seule communication de type identification. Lors de cette analyse, un problème avait été identifié. L'annotation (stéréotypée <<*rationale*>>) exprime la solution retenue face à ce problème. Enfin, le block *Capteur Mobile* exprime la prise en compte de l'exigence *1.1.1* via la relation <<*satisfy*>>. La Figure 3 montre la représentation tabulaire des exigences appelée table des exigences. La représentation diagrammatique peut s'avérer complexe à utiliser sur des projets opérationnels de grande taille (500 exigences et plus). La table des exigences prend alors toute son utilité, car elle est plus simple à manipuler pour assurer la gestion et la traçabilité d'un grand nombre d'exigences. Ces deux figures ont été réalisées en utilisant le modeleur intégré à *Polarsys IDE (Integrated Development Environment)*.

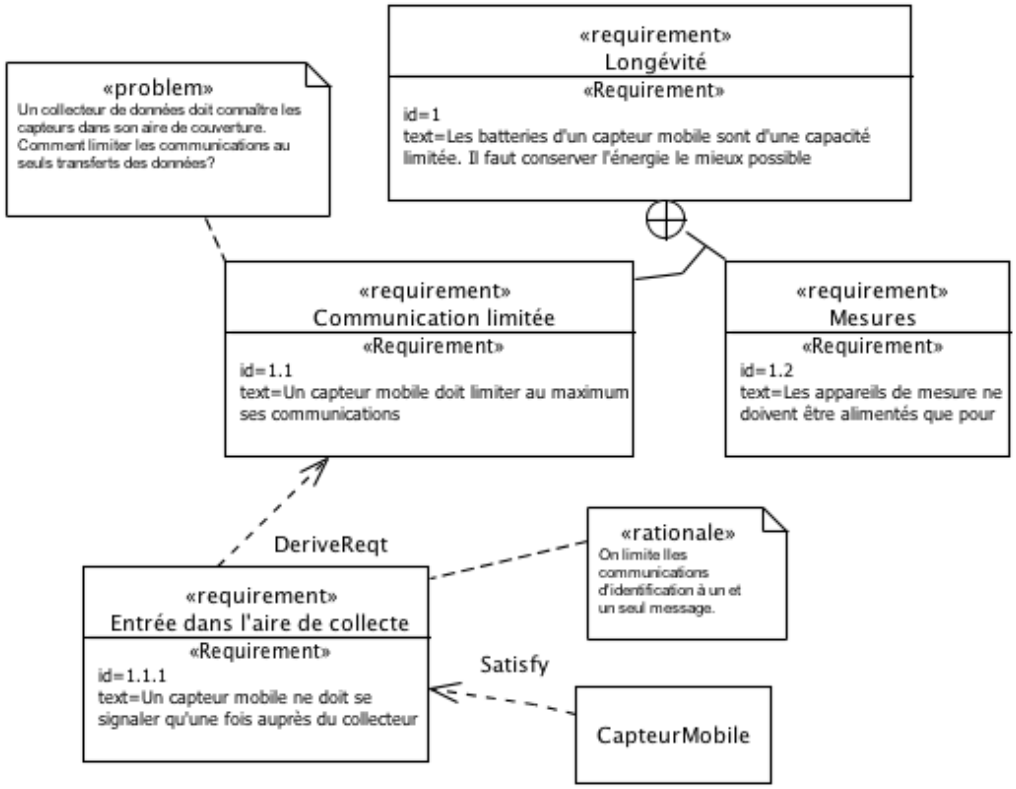

Figure 2 : Une partie du diagramme des exigences

	[Label]	name	text	id	derived	derivedFrom	satisfiedBy
1	<<Requirement>> <Class>..	Longévité	Les batteries d'un capteur...	1			
2	<<Requirement>> <Class> E	Entrée dans l'aire de...	Un capteur mobile ne doit..	1.1.1		<<Requirement>> <Clas...	<<Block>> <Class> C
3	<<Requirement>> <Class...	Mesures	Les appareils de mesure ...	1.2			
4	<<Requirement>> <Class>..	Communication limi...	Un capteur mobile doit ...	1.1	<<Requirement>> <Class> E		

Figure 3 : La table des exigences correspondante

3. DÉMARCHE D'UTILISATION DE SYSML POUR LES EXIGENCES

S'il est une chose à bien comprendre, c'est que SysML, à l'instar d'UML, n'est pas une méthode ou un processus. Il s'agit uniquement d'un langage. Il est certes entendu que certains diagrammes sont destinés à être utilisés à tel ou tel moment du cycle de développement d'un système. C'est le cas du diagramme des exigences. Cependant, comme le montre la relation *<<satisfy>>*, les exigences ont vocation à être liées aux autres éléments du modèle. Ainsi, les éléments de modèle liés aux exigences peuvent être manipulés à d'autres étapes que durant le recueil des exigences. En effet, le travail sur les exigences est par nature itératif. Il y a par exemple une étape de recueil des exigences, puis une étape d'analyse des exigences recueillies qui va permettre de raffiner les exigences identifiées et de les allouer aux éléments de solution.

Afin de bien comprendre cette nuance, il convient de regarder plus généralement le principe des modélisations UML/SysML. En effet, la première approche de ces langages se fait généralement à travers la réalisation ou la consultation de diagrammes. Or, les diagrammes ne sont que des représentations parcellaires et graphiques du modèle complet du système que l'on considère. Il faut les voir comme une vue du modèle. Et pour un même élément du modèle, on peut vouloir des représentations plus ou moins précises voire guidées par un prisme particulier. Les relations entre éléments de modélisation, et plus particulièrement avec les exigences, sont fortement encouragées par les mécanismes de dépendances et de traçabilité (notamment introduits et mis en avant dans SysML, cf. Fig. 4) pour répondre aux bonnes pratiques de vérification et d'analyse d'impact, qui nécessitent d'avoir relié les éléments pour pouvoir suivre les changements. Ces bonnes

pratiques deviennent des recommandations et même des exigences quand on est dans un contexte industriel critique, avec des autorités de certification à qui il faudra rendre compte.

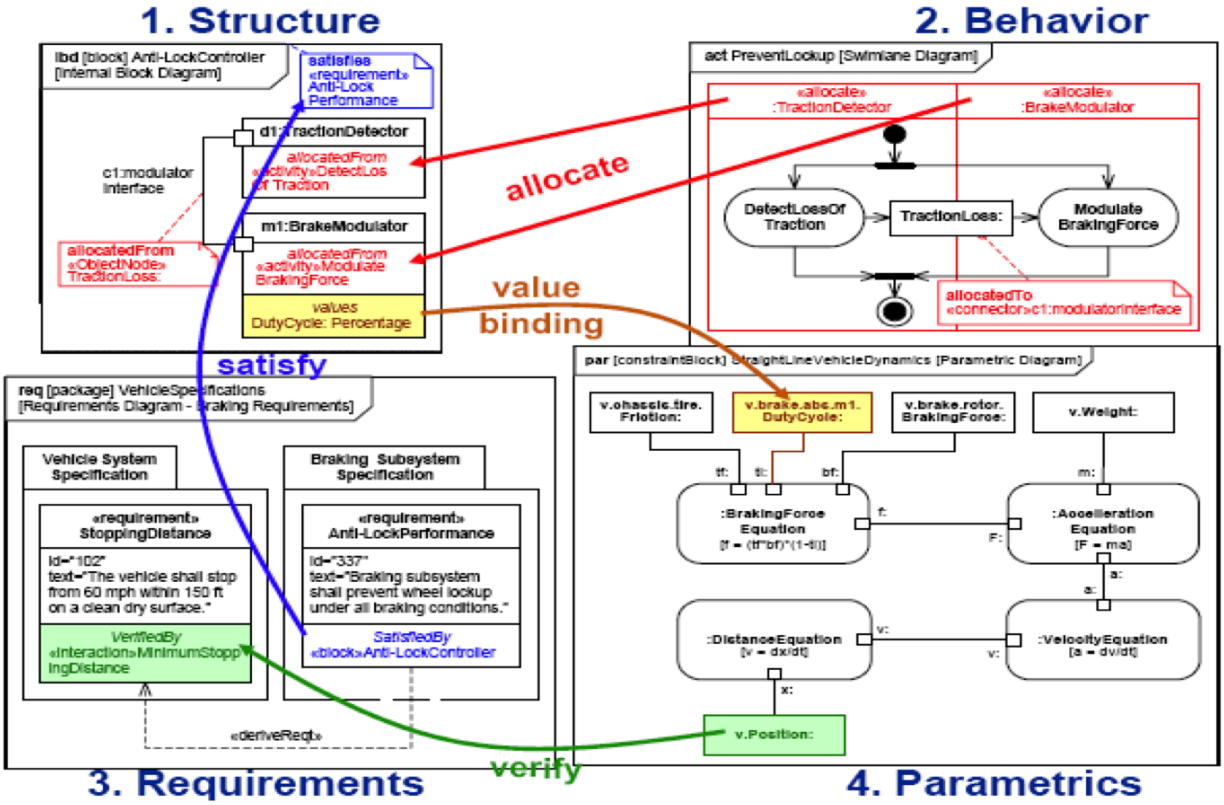

Figure 4 : Liens entre éléments de modèle [8]

Pour résumer, voici une liste non exhaustive de l'intérêt de modéliser les exigences de manière explicite en SysML :

- Un modèle d'exigence spécialisé permet de centraliser la vérification que les exigences ont été prises en compte par au moins un élément de modélisation ;
- SysML fournit des moyens d'exprimer formellement ces liens de traçabilité ;
- SysML distingue deux familles de liens :
  . entre exigences (par exemple *containment*, *derive*, *copy* et *trace*)
  . entre exigences et implémentation (par exemple *satisfy*, *verify* et *refine*)
- SysML supporte trois représentations : graphique, tabulaire ou arborescente ;
- Les mécanismes d'annotation permettent de capturer les raisons des choix de conception (*rationale*).

4. OUTILS

La complexité inhérente aux systèmes complexes impose l'utilisation d'outils complets et performants permettant la mise en œuvre de la notation utilisée. À ce propos, SysML présente l'avantage d'être assisté par une large palette d'outils qui commencent à devenir matures même s'ils sont encore considérés comme trop complexes par la majorité des ingénieurs systèmes. Ces outils sont majoritairement dérivés d'outils orientés UML qui ont été adaptés pour assister le langage. Cela est rendu possible, car SysML peut-être vu comme un profil UML (sa spécification est fournie comme telle par l'OMG).

Nous n'avons pas trouvé dans la littérature d'étude comparative complète de ces outils. Par contre, il existe plusieurs listes les référençant telles que celle présentée par Tim Wielkiens [9]. Parmi les outils les plus utilisés, citons Cameo (anciennement Magic Draw), IBM Rhapsody, Enterprise Architect et Artisan Studio pour les outils commerciaux. Concernant les logiciels libres, on peut citer l'initiative européenne Polarsys (née au sein du projet ITEA2 OPEES) visant à

fournir une infrastructure (voir l'utilisation de l'éditeur Papyrus de Polarsys IDE dans la Figure 5) et une gouvernance pour héberger à long terme des composants *open source* de systèmes embarqués et critiques.

Les outils SysML disponibles ne se limitent pas à offrir une seule plateforme de modélisation graphique et d'édition de diagrammes SysML. Ils fournissent tous, plus ou moins, des fonctionnalités avancées comme la gestion des versions et/ou de la modélisation de groupe, la mise en ligne ou la génération d'un document de spécification commun aux divers intervenants, la vérification des modèles, l'extensibilité du langage et de plus en plus la mise à disposition d'outils de simulation.

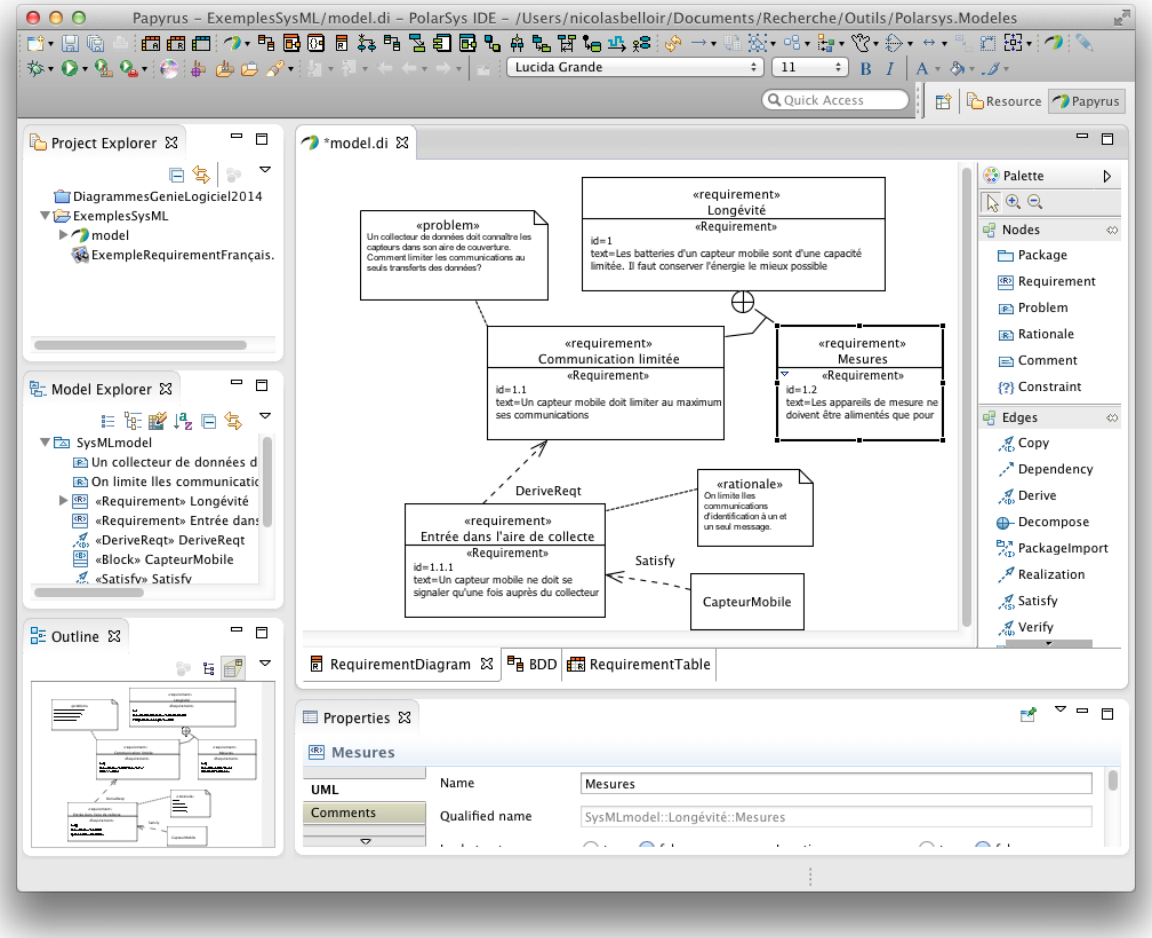

Figure 5 : Utilisation de l'éditeur Papyrus de Polarsys IDE

5. LIMITATIONS ET PISTES FUTURES

Nous l'avons déjà exprimé en amont mais il est fondamental d'insister sur ce point : SysML est un langage et non une méthode. Il faut le voir comme une boîte à outils générale dans laquelle on piochera des outils en fonction des besoins. Dans la problématique liée à l'ingénierie des exigences, ce qu'apporte SysML c'est un moyen de formaliser les exigences, et de, plus tard dans lors du développement, lier les éléments de modèle à la ou les exigences qui ont conduit à la définition de cet élément de modèle.

SysML ne repose pas sur une méthode ce qui signifie qu'il n'offre pas tout seul de moyen à l'identification et à l'émergence des exigences comme peuvent le faire d'autres approches. Nous l'avons d'ailleurs comparé à des méthodes et langages plus

axés sur l'identification et l'émergence ou a des méthodes couplant son utilisation à leur processus d'explicitation [10]. Concernant l'ingénierie des exigences, SysML reste limité à de nombreux égards. Par contre il peut être utilement combiné avec d'autres approches telles que Kaos [11]. Dans un tutoriel présenté à RE'2013 [12], nous avons montré comment des approches orientées but (*Goal-Oriented Approaches*) comme Kaos pouvaient être utilisées en amont, en phase d'explicitation des exigences. Nous avons établi un certain nombre de correspondances avec les concepts et relations de traçabilité SysML qui permettent de réaliser, par transformation de modèles, une forte intégration des deux approches. Nous travaillons également avec l'équipe de Régine Laleau à l'utilisation des méthodes formelles pour préciser encore plus ces liens [13]. Il est de plus possible d'étendre SysML pour assister entièrement une méthode. Par exemple, SysML/Kaos intègre les concepts Kaos au langage SysML de manière à bénéficier des avantages de Kaos pour l'élicitation des exigences [14].

ReqCycle[1] vient combler deux limitations de SysML. La première concerne le fait que SysML n'introduit la notion d'exigence qu'au sein du langage lui-même, sans possibilité de faire référence (autrement que par des annotations) aux exigences qui ont été définies dans d'autres langages (Simulink, Modelica, Scade, document, Excel, DOORS...). ReqCycle est capable de référencer ces exigences et de les agréger à celles capturées dans un ou plusieurs modèles SysML. La deuxième provient de ce que SysML considère les exigences sous leur seul angle textuel. ReqCycle ouvre la voie vers le référencement d'exigences qui ne sont pas formalisées par du texte, telles que des propriétés de bloc ou des transitions de machine à états, en reposant sur la théorie « Property-Based Requirement » [15]. Dans cette optique, certains éléments de modèle peuvent être considérés comme des exigences et peuvent donc potentiellement être dérivés d'autres exigences. Les relations de traçabilité de SysML, qui marquent clairement la distinction entre un élément « Exigence » et un élément de modèle, sont impactées par cette généralisation de la notion d'exigence.

6. CONCLUSION

Le langage SysML est un langage standardisé dédié à la modélisation des systèmes complexes et construit par spécialisation du langage UML. Il apporte de réelles nouveautés qui, si elles furent pensées en amont pour l'ingénierie système, peuvent tout à fait trouver leur place dans la modélisation de systèmes logiciels. Plus particulièrement, la prise en compte des exigences à travers un diagramme spécialisé, mais également la traçabilité entre les exigences et le reste des éléments du modèle, permettent de créer des modèles dont les éléments sont reliés entre eux ainsi qu'aux exigences. L'autre point fort de SysML repose sur sa capacité d'extension issue des profils UML. Elle permet de l'adapter à un champ sémantique proche des besoins de chaque entreprise. D'autre part, reposant sur un métamodèle, on peut également l'intégrer à des processus de transformations de modèles. Il est possible de coupler son utilisation à des méthodes et éventuellement à des outils (dépend de l'outil) plus en amont du cycle de modélisation, utilisant ainsi de manière conjointe des outils puissants à la fois pour le recueil des exigences et pour leur expression. Enfin, de par sa standardisation, ce langage est soutenu par un nombre d'outils aussi bien payants que gratuits qui lui permettent une large diffusion. Cette diffusion va par ailleurs s'amplifier, en France, puisque le langage a été choisi comme langage support auprès de la formation continue dans les lycées techniques par le nouveau programme STI2D [16] ou encore depuis la rentrée 2013 en première année de classe préparatoire.

RÉFÉRENCES

[1] Object Management Group: *UML V2.4.1* ; Disponible ici : http://www.omg.org/spec/UML/, 2011
[2] INCOSE, *Systems Engineering Vision 2020* ; Disponible ici : http://www.incose.org/ProductsPubs/products/sevision2020.aspx, 2009
[3] Object Management Group : *SysML V1.3* ; Disponible ici : http://www.omg.org/spec/SysML/ ; 2011
[4] Association SysML-France, *blog de l'association*, http://www.irit.fr/SysML/
[5] S. Friedenthal, R. Griego et M. Sampson : *INCOSE Model-Based Systems Engineering Initiative* ; 2007 ; Disponible ici : http://goo.gl/8zyvn6
[6] Dominique Potier : *Rapport sur les briques génériques du logiciel embarqué* ; 2013 ; Disponible ici : http://goo.gl/QlkV44
[7] INCOSE, *SE Handbook* ; Disponible ici : http://goo.gl/h5nJcd
[8] OMG, *OMG SysML Tutorial* ; Disponible ici : http://www.omgsysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf

[1] https://www.eclipse.org/proposals/polarsys.reqcycle/

[9] Tim Wielkiens : *Popular SysML modeling tools* ; Disponible ici : http://list.ly/list/23A-popular-sysml-modeling-tools
[10] Manzoor Ahmad, João Araújo, Nicolas Belloir, Jean-Michel Bruel, Christophe Gnaho, Régine Laleau et Farida Semmak : *Self-adaptive systems requirements modelling: four related approaches comparison* ; in Proceedings of the Comparing Requirements Modeling Approaches (CMA@RE'13) Workshop, tenu dans le cadre de IEEE International Conference on Requirements Engineering 2013, pages 37-42, Rio de Janeiro, Brésil, 16 juillet 2013
[11] KAOS. http://en.wikipedia.org/wiki/KAOS_(software_development)
[12] João Araujo et Jean-Michel Bruel : *Model-Based Systems Requirements*. IEEE Requirements Engineering Conference Tutorial. 2013. Disponible ici : http://www.slideshare.net/jmbruel/modelbased-systems-requirements
[13] Manzoor Ahmad, Jean-Michel Bruel, Régine Laleau et Christophe Gnaho : *Using RELAX, SysML and KAOS for Ambient Systems Requirements Modeling*. Dans Procedia Computer Science, Science Direct, Numéro spécial : The 3rd International Conference on Ambient Systems, Networks and Technologies (ANT) Proceedings, vol. 10, p. 474-481, août 2012
[14] Christophe Gnaho et Farida Semmak : *Une extension SysML pour l'ingénierie des exigences non-fonctionnelles orientée but ;* dans Ingénierie des Systèmes d'Information, pages 9-32, Lavoisier, 2010
[15] Patrice Micouin : *Toward a property-based requirements theory: System requirements structured as a semilattice* ; Systems Engineering, vol. 11, issue 3, pages 235-245, 2008
[16] Jean-Pierre Lamy : *SysML, un langage modèle ;* in Technologies, Canopi éditions, n°179, pages 32-48, avril 2012

BIOGRAPHIES

Nicolas Belloir est maître de conférences à l'Université de Pau et des Pays de l'Adour (UPPA) depuis 2005. Il est membre de l'équipe MOVIES du Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour (LIUPPA). Il est titulaire d'un DEA de l'Université Paul Sabatier obtenu en 1999. Il a ensuite travaillé comme ingénieur d'étude pour la société Transiciel, effectuant des missions dans le domaine du temps réel embarqué jusqu'en 2001. Il a ensuite rejoint l'UPPA pour effectuer sa thèse et a soutenu son doctorat en informatique en 2004. Ses recherches sont menées dans le domaine du génie logiciel et portent sur l'utilisation des langages semi-formels. Actuellement, il étudie l'application des techniques d'Ingénierie Dirigée par les Modèles aux systèmes complexes. Il est secrétaire de l'association SysML-France.

Jean-Michel Bruel est Professeur des Universités à l'Université de Toulouse depuis 2008. Il est membre de l'équipe MACAO (Models, Architectures, Components, Agility and professes) de l'Institut de Recherche en Informatique de

Toulouse (IRIT - UMR CNRS 5505). Il a obtenu son doctorat informatique de l'Université Paul Sabatier (Toulouse) en décembre 1996. De septembre 1997 à août 2008, il a été Maître de Conférences à l'Université de Pau et des Pays de l'Adour. Ses recherches portent sur le Génie Logiciel et plus particulièrement l'utilisation des techniques d'Ingénierie des Modèles (IDM) pour le développement des systèmes distribués complexes. Il a défendu son Habilitation à Diriger des Recherches en décembre 2006. Il est également l'un des co-fondateurs de l'association SysML-France.

Raphaël Faudou. Après une formation d'ingénieur informatique à l'ENSEEIHT (Toulouse), Raphaël Faudou débute sa carrière en septembre 1994 et devient consultant UML à partir de 1999. En 2003 il rejoint le groupe Atos Origin à Toulouse comme architecte logiciel et intervient sur des équipements du programme Airbus A380, puis pilote la participation d'AtoS à la plateforme de modélisation open source TOPCASED dès 2005. À partir de 2007, il prend progressivement en charge l'innovation, la définition des méthodes et outils concernant le département « Systèmes Embarqués ». Il se forme à l'ingénierie des systèmes et plus particulièrement à la définition d'architecture des systèmes, avec plusieurs missions d'accompagnement SysML en milieu industriel de 2009 à 2013. En novembre 2013, il quitte AtoS et créé la société Samares Engineering qui se positionne en expert modélisation auprès des architectes de systèmes. Raphaël Faudou intervient désormais auprès des bureaux d'études industriels, anime un groupe de travail MBSE à l'AFIS sur le suivi des exigences en phase de définition d'architecture, et intervient auprès des écoles d'ingénieurs toulousaines pour promouvoir le déploiement de l'ingénierie pilotée par les modèles dans le cadre des systèmes complexes.