Modélisation des Exigences pour les Systèmes Auto-adaptatifs : Intégration des Techniques Relax/Kaos/SysML

Manzoor Ahmad, Jean-Michel Bruel
University of Toulouse
CNRS/IRIT
F-31062 Toulouse Université Cedex,France
manzoor.ahmad,bruel@irit.fr

Christophe Gnaho, Régine Laleau
University of Paris-Est Créteil
LACL
94010 Créteil Cedex, France
laleau,gnaho@u-pec.fr

Résumé

Les systèmes dynamiques adaptatifs (*Dynamic Adaptive Systems* – DAS) peuvent modifier leur comportement de manière autonome (y compris en cours d'exécution) en fonction de l'évolution de leur environnement. Nous partons du principe que pour bien développer le caractère adaptatif de ces systèmes, il est important de bien identifier les exigences sur lesquelles peuvent porter cette adaptation (par opposition aux invariants). Des approches basées sur les buts peuvent aider à l'élaboration des exigences pour les DAS, compte tenu de l'incertitude inhérente à ces systèmes. Nous proposons dans cet article de combiner trois techniques complémentaires pour atteindre ce but : Relax, un langage pour exprimer les exigences adaptables, Kaos, une approche orientée but, et SysML [1], la notation dédiée à la modélisation des systèmes.

Mots Clés: Ingénierie des Exigences, Relax, Systèmes Dynamiques Adaptatifs, Langage Spécifique au Domaine, Exigences Non Fonctionnelle

1 Introduction

La plupart des travaux sur l'ingénierie des exigences (RE) pour les systèmes dynamiques adaptatifs (DAS) supposent que les exigences existent déjà . L'accent est ainsi mis sur le suivi des exigences et du raisonnement à propos de l'exactitude des adaptations [1]. Les conditions environnementales de ces systèmes ont tendance à changer et ils doivent s'adapter en conséquence à ces conditions changeantes. Il est indéniable qu'un nombre croissants de systèmes logiciels doivent s'adapter aux changements de leur environnement ou de leurs exigences afin de continuer à remplir leur fonction (assistance à domicile, adaptation au contexte, mobilité, etc.).

Les approches orientées buts (*goal-based approaches*) peuvent être utilisées pour systématiquement modéliser les exigences des DAS. Ces derniers sont traités comme une collection de systèmes cibles avec des conditions environnementales variables. Les besoins de chaque système cible sont donc modélisés, et la logique adaptative qui sert de transition entre les configurations est traitée comme des considérations distinctes [2].

Nos précédents travaux autour des exigences [3] ont abouti au langage Relax [4] pour modéliser les exigences non fonctionnelles (*Non Functional Requirements* – NFRs) qui peuvent être relâchées. Relax est un langage de l'ingénierie des exigences qui traite de l'incertitude dans les exigences et qui permet aux exigences d'être temporairement assouplies pour s'adapter aux conditions environnementales changeantes. Cette "relax-ation" est offerte dans le cas où les exigences non critiques peuvent être partiellement assouplies afin de satisfaire des exigences plus essentielles à court terme. Un défi fondamental dans le développement des DAS est de savoir comment gérer l'incertitude posée par les domaines d'application respectifs. Par conséquent,

1. http ://www.sysml.org/specs/

nous considérons chaque NFR comme "relax-able" (ou pouvant varier) afin de pouvoir utiliser la notion de but dans notre approche intégrée.

Des travaux similaires peuvent être trouvés dans la littérature concernant les systèmes adaptatifs, qui commencent par la définition des besoins jusqu'à leur opérationalisation. Par exemple, *Awareness Requirements* – AwReqs [5] est l'un d'entre eux. Les exigences y sont caractérisées syntaxiquement comme des exigences qui se réfèrent à d'autres exigences (hypothèses de domaine) et par leur réussite ou leur échec à l'exécution. Les AwReqs sont représentés dans un langage formel et peuvent être directement contrôlées par un environnement de suivi des exigences. Dans [6], les auteurs présentent FLAGS (*Fuzzy Live Adaptation Goals for Self-Adaptive Systems*), un modèle de but innovant qui généralise le modèle de KAOS, ajoute la notion de but d'adaptation pour intégrer les contre-mesures d'adaptation, et favorise l'adaptation autonomique.

L'objectif est donc de modéliser les exigences de ce DAS par la fusion d'une approche orientée buts qui se base (et étend) le méta-modèle de SYSML et de trouver un lien avec notre DSL pour RELAX. La suite de cet article est organisée de la façon suivante : la section 2 décrit le contexte de ce travail ; la section 3 décrit plus précisément notre contribution ; la section 4 conclue et dresse les priorités pour les suites à donner à ce travail préliminaire.

2 Contexte

2.1 SysML

SYSML est un langage généraliste de modélisation des systèmes. Il prend en charge la spécification, l'analyse, la conception, la vérification et validation d'une large gamme de systèmes et de systèmes-de-systèmes. Ces systèmes peuvent inclure du matériel, des logiciels, des informations, des processus, le personnel et les installations. Il comprend notamment une syntaxe graphique pour représenter les exigences textuelles et les relier aux autres éléments de modèles (comme les éléments physique, avec des liens d'allocation, ou les cas d'utilisation par des liens de satisfaction par exemple). Le diagramme des exigences modélise les hiérarchies des exigences et leurs déclinaisons éventuelles, et les relations du type <<`satisfy`>> et <<`verify`>> permettent de relier une exigence à un élément de modèle. Le diagramme des exigences constitue un pont entre les outils de gestion des exigences typiques et la modélisation système.

2.2 DSL pour RELAX

Typiquement, les exigences textuelles prescrivent le comportement à l'aide d'un verbe *SHALL* qui définit une fonctionnalité qu'un logiciel doit toujours fournir. RELAX prend la forme d'un langage naturel structuré, comprenant des opérateurs spécialement conçus pour capturer l'incertitude [7]. Pour les systèmes auto-adaptatifs cependant, l'incertitude de l'environnement peut signifier qu'il n'est pas toujours possible d'atteindre tous les états *SHALL*. Ou bien encore l'incertitude du comportement peut autoriser un compromis entre deux états *SHALL* pour favoriser un état non critique au profit des autres, etc. À cet effet, RELAX identifie deux types d'exigences : celles qui peuvent être relâchées et les autres (appelées invariants). Les opérateurs RELAX sont introduits pour donner plus de souplesse dans la façon dont la contrainte doit être prise en compte. Plus précisément, pour les besoins qui peuvent être partiellement insatisfaits, l'introduction d'une opérateur alternative, temporelle ou ordinale doit atteindre cette possibilité. Ces opérateurs définissent des contraintes sur la façon dont une exigence peut être assouplie au moment de l'exécution. En outre, il est important d'indiquer quels sont les facteurs

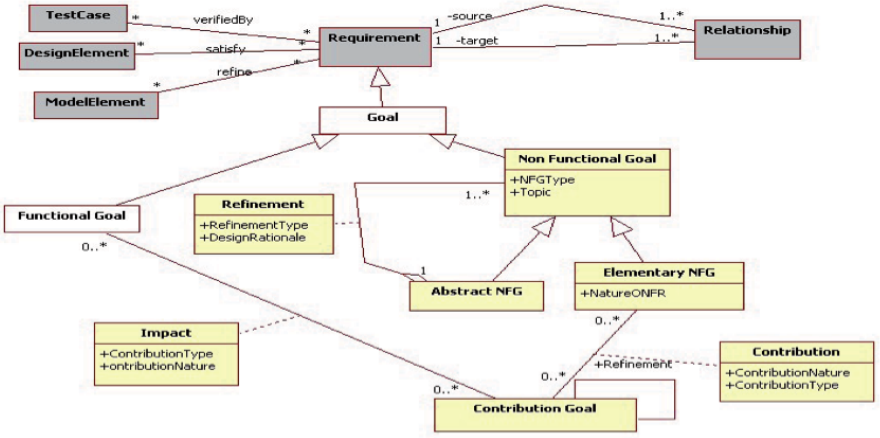

FIGURE 1 – Extended SysML Meta-Model [9]

d'incertitude qui justifient un assouplissement de ces exigences, ce qui nécessite un comportement adaptatif. Cette information est spécifiée en utilisant les mot clés : MON (*monitor* qui précise ce qu'on cherche à monitorer, comme une position), ENV (*environment* qui précise les éléments sur lesquels le système à la main, comme un capteur de pression ou un radar), REL (*relationship* qui indique le lien entre les deux précédents) et DEP (*dependency* pour préciser les liens entre exigences).

Nos travaux précédents sur RELAX ont consisté à la définition d'un langage dédié (*Domain Specific Language* – DSL) pour les systèmes auto adaptatifs [3]. En utilisant notre DSL, les exigences non fonctionnelles au format textuel sont transformées en éléments de modèle SYSML (format graphique) sous forme de diagramme d'exigence dans un premier temps [2]. L'outil eclipse XText [3] est utilisé comme générateur pour l'éditeur de code de RELAX. XText permet la description simple de DSL à partir de la définition de sa grammaire et fournit de manière automatisée un *Integrated Development Environnement* (IDE) pour ce DSL (éditeur, générateur de code prêt à l'emploi, etc.). Nous avons écrit un générateur de code capable de traiter des modèles créés avec l'éditeur et de générer le modèle SYSML sous la forme d'un fichier XMI. Notre éditeur RELAX, COOL [8] a été développé en utilisant la grammaire RELAX en tant que méta-modèle.

Les limites de notre DSL pour RELAX sont étendues par les concepts de SYSML/KAOS [9]. En SYSML/KAOS, les NFRs sont exprimés de manière plus riche, sous la forme de buts. Les relations entre les exigences y sont également définies de manière plus complète et précise (relations de raffinement, l'identification et résolution des conflits, les impacts positifs/négatifs et direct/indirect). Ici, les exigences d'invariant sont capturées par le concept de buts fonctionnels (FG) alors que les exigences assouplies par celui de buts non fonctionnels (NFG).

2.3 SysML/KAOS

Le modèle SYSML/KAOS est une extension du modèle des exigences de SYSML par l'intégration des concepts de buts de KAOS [10]. Tout d'abord, KAOS permet d'exprimer plusieurs modèles (goal, agent, object, behavioral models) et les relations entre eux. Deuxièmement,

2. La génération de diagramme paramétriques est également à l'étude.
3. http ://www.eclipse.org/Xtext/documentation

il fournit un ensemble puissant et complet de concepts pour spécifier des modèles de but. Ceci permet la conception des hiérarchies de but avec un haut niveau d'expressivité qui peut être considéré à différents niveaux d'abstraction. En effet, comme SysML est une extension d'UML, il fournit des concepts pour représenter les besoins et les relier aux éléments du modèle, permettant la définition des liens de traçabilité entre les exigences et les modèles du système. Cependant l'ensemble des concepts de SysML pour la modélisation des exigences n'est pas aussi étendue que dans les modèles de but. Le modèle SysML/Kaos permet de modéliser à la fois les exigences fonctionnelles et exigences non-fonctionnelles [11]. Dans cet article, nous mettons l'accent sur les concepts liés aux exigences non fonctionnelles.

La Figure 1 montre des concepts non fonctionnels sous la forme de boîtes jaunes (en bas de la figure), les boîtes grises représentant les concepts de SysML (en haut de la figure). L'instanciation du méta-modèle nous permet d'obtenir une hiérarchie des besoins non-fonctionnels sous forme de buts. Les buts non fonctionnels (*Non-fonctionnas Goals* – NFG) sont organisés en hiérarchies de raffinement. Un NFG est soit un NFG abstrait, soit un NFG élémentaire. Un but qui ne peut pas être raffiné est un but élémentaire. Le raffinement d'un but abstrait par des buts abstraits ou élémentaire est représenté par une classe d'association de raffinement. Un NFG abstrait peut contenir plusieurs combinaisons de sous-buts (abstraits ou élémentaires). La relation de raffinement devient une classe d'association entre un NFG abstrait et ses sous-buts. Il peut être spécialisé pour représenter des relations du type AND/OR. À la fin du processus de raffinement, il est nécessaire d'identifier et d'exprimer les alternatives pour satisfaire les buts élémentaires. Pour cela, nous considérons le concept de but de contribution (*Meta-Class Contribution Goal*). Un but de contribution capture une manière possible de satisfaire un but élémentaire. La classe d'association `Contribution` décrit les caractéristiques de la contribution. Il fournit deux propriétés : `ContributionNature` et `ContributionType`. Le premier spécifie si la contribution est positive ou négative, la seconde spécifie si la contribution est directe ou indirecte. Une contribution positive (resp. négative) aide positivement (resp. négativement) à la satisfaction d'un but élémentaire. Une contribution directe décrit une contribution explicite au but non-fonctionnel élémentaire. Une contribution indirecte décrit une sorte de contribution qui est une contribution directe à un but donné, mais induit une contribution inattendue à un autre but. Le concept d'impact est utilisé pour relier les buts non fonctionnels aux buts fonctionnels. Il est basé sur le fait qu'un but de contribution a un effet sur un but fonctionnels.

3 Fusion des approches

Notre contribution consiste à fusionner les techniques et les approches décrites précédemment afin d'obtenir une description détaillée et robuste des exigences du système et de son contexte. En nous basant sur le fait que les techniques orientés buts peuvent aider à définir les exigences des DAS, nous donnons quelques propositions.

La première proposition concerne les facteurs d'incertitude, en particulier les attributs ENV et MON de Relax, qui sont particulièrement importants pour documenter si le système possède les moyens de suivre les aspects critiques de l'environnement. En recueillant ces attributs ENV et MON, nous pouvons construire un modèle de l'environnement dans lequel le système va fonctionner, ainsi qu'un modèle qui définit la façon dont le système surveille son environnement. Ceci étant dit, SysML/Kaos peut aider Relax en injectant plus d'informations sous forme d'effets positifs/négatifs et directs/indirects.

La deuxième proposition concerne les méta-modèle. La grammaire de Relax agit comme un méta-modèle pour notre DSL, tandis que SysML/Kaos a étendu le méta-modèle de SysML avec le concept de but. Comme les deux méta-modèles sont proches du méta-modèle de SysML,

Concepts	SysML/KAOS	SysML	RELAX
Requirements Description	Goals	Textual Requirements	Enhanced Version of Textual Requirements
Relationship	AND, OR	<<verify>> <<refine>>	REL
Dependency/Impact	Contribution Nature: Positive Negative Contribution Type: Direct (Explicit) Indirect (Implicit) **b/w NFG and FG**	<<derive>> <<contain>>	DEP: Positive Negative
Monitoring	<<contribution goal>>	<<satisfy>>	MON
Tools	Eclipse based SysML/KAOS Editor	Eclipse/Papyrus/Topcased/	Eclipse based COOL RELAX editor

FIGURE 2 – Relationship b/w Different Concepts

les liens entre les deux langages vont être simples. Nous sommes donc convaincus que l'outillage de notre approche combinée ne sera pas un problème. En outre, nous fournirons une forte cohérence entre les modèles. Cela peut être assuré grâce à l'utilisation des méthodes formelles que fournissent des outils de vérification. Nous avons déjà développé une méthode [12] pour dériver des spécifications formelles B à partir de modèles SysML/Kaos qui peuvent être étendues dans l'approche combinée.

La troisième proposition concerne les relations entre SysML/KAOS, SysML et RELAX. Dans la Figure 2, nous indiquons comment plusieurs concepts clés sont pris en compte dans les différentes techniques sélectionnées. La plupart du temps, les concepts ne sont pas entièrement couverts (e.g. <<`satisfy`>> pour le suivi dans SysML, ce stéréotype est utilisé entre un block et une exigence), mais nous avons indiqué dans le tableau le plus proche mécanisme qui prend en charge les concepts. Il s'agit d'une première étape avant de faire une correspondance plus formelle par les méta-modèles (cf. [13] pour plus de détail). SysML/Kaos dispose d'un outil appelé SysML/Kaos *editor*, SysML dispose d'un certain nombre d'outils, par exemple : eclipse [4],papyrus [5], topcased [6] etc. et pour Relax, nous avons développé un éditeur COOL[8] qui est basé sur Eclipse.

4 Conclusion et perspectives

Les systèmes auto-adaptatifs peuvent modifier leur comportement à l'exécution en réponse aux changements de leur conditions environnementales. Pour ces systèmes, les exigences non-fonctionnelles jouent un rôle important, et il faut identifier au plus tôt les exigences qui sont concernées par ces caractéristiques adaptatives. Pour élaborer les exigences des DAS, les approches basées sur les buts jouent un rôle important compte tenu de l'incertitude inhérente à ces systèmes. Relax est un langage d'ingénierie des exigences pour exprimer les exigences adaptables et permettre d'introduire la flexibilité dans les exigences non-fonctionnelles. Nous proposons d'améliorer la modélisation des exigences des DAS en fusionnant deux approches

4. http ://www.eclipse.org/
5. http ://www.papyrusuml.org
6. http ://www.topcased.org/

complémentaires par l'utilisation de l'approche existante orienté but, pour laquelle le méta-modèle SysML est étendu avec des concepts de but et par celle de Relax.

Nous pensons que SysML/Kaos peut compléter Relax en injectant des informations pertinentes supplémentaires (e.g., comme celles concernant les impacts positifs/negatifs ou directs/indirects). Les deux approches traitent des exigences au plus tôt dans le cycle de vie. Notre travail sur Relax n'est qu'une brique de base d'un plan plus large [14]. Dans cet article nous avons ébauché les bases d'une intégration possible plus importante de ces deux approches. L'intégration des techniques Kaos, SysML et Relax permettent à chaque technique de bénéficier des avancées de l'autre. Une voie en cours d'exploration concerne, pour Relax, l'utilisation des informations recueillies dans les attributs ENV, MON et REL pour générer automatiquement les diagrammes paramétriques de SysML. La suite de ce travail va être de définir des règles de transformations ou faire une correspondance entre les méta-modèles de Relax et de SysML/Kaos.

Références

[1] Betty H. C. Cheng, Pete Sawyer, Nelly Bencomo, Jon Whittle. A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty, MODELS '09, Springer-Verlag Berlin, Heidelberg 2009.

[2] Ji Zhang, Betty H. C. Cheng. Model-based development of dynamically adaptive software, ICSE 06 : Proc. of the 28th int. conf. on Software engineering, NY USA, ACM (2006).

[3] Manzoor Ahmad, First Step towards a Domain Specific Language for Self Adaptive Systems, NOTERE, May 31- June 2 2010 Tozeur Tunisia, pp. 285–290.

[4] Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty H.C. Cheng, and Jean-Michel Bruel. Relax : Incorporating Uncertainty into the Specification of Self-Adaptive systems, Proceedings of the 2009, 17th IEEE International Requirements Engineering Conference, Pages : 79-88.

[5] Vítor E. Silva Souza, Alexei Lapouchnian, William N. Robinson, John Mylopoulos. Awareness requirements for adaptive systems, SEAMS '11 Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, New York USA.

[6] Luciano Baresi, Liliana Pasquale, Paola Spoletini. Fuzzy Goals for Requirements-Driven Adaptation, 18th IEEE International Requirements Engineering Conference (RE) 2010, Sydney Australia.

[7] J. Whittle, P. Sawyer, N. Bencomo, and B. H. C. Cheng. Reassessing languages for requirements engineering of selfadaptive systems, In RE Workshop for SOCCER08, Tec. Report, 2008.

[8] Sarah Gatti, Jacob Geisel, Simon Labondance, Julien Pages. COOL Relax Editor, Internal Report M2ICE, U. Toulouse II Le Mirail 2011.

[9] Yassir Belkaid, Christophe Gnaho, Farida Semmak. Création de l'éditeur Kaos-SysML sous Topcased, Rapport Tacos, Juillet 2010.

[10] Axel Van Lamsweerde. Requirements Engineering : From System Goals to UML Models to Software Specifications. Wiley, 2009.

[11] Christophe Gnaho, Farida Semmak. Une extension SYSML pour l'ingénierie des exigences non-fonctionnelles orientée but. Revue Ingénierie des Systèmes d'Information, Vol 16/1, 23 pages, 2011.

[12] Régine LALEAU, Farida SEMMAK, Abderrahman MATOUSSI, Dorian PETIT, Ahmed HAMMAD, Bruno TATIBOUET. "A first attempt to combine SysML requirements diagrams and B." Innovations in Systems and Software Engineering, Springer, 6(1-2) : 47-54, 2010.

[13] Manzoor Ahmad, Jean-Michel Bruel, Régine Laleau et Christophe Gnaho. Using RELAX, SysML and KAOS for Ambient Systems Requirements Modeling. The 3rd International Conference on Ambient Systems, Networks and Technologies August 27-29, 2012, Niagara Falls, Ontario, Canada.

[14] Jean-Michel Bruel, Nicolas Belloir, Manzoor Ahmad, SPAS : un profil SysML pour les systèmes auto-adaptatifs, 15ème Colloque National de la Recherche en IUT (CNRIUT), Lille, 2009.