Une première approche de traçabilité entre modèles d'exigences non-fonctionnelles et spécifications abstraites Event-B

Abderrahman Matoussi — Régine Laleau

Université Paris-Est, LACL, IUT Sénart Fontainebleau,
Dpt Informatique, Route Hurtault, 77300 Fontainebleau, France
{abderrahman.matoussi, laleau}@u-pec.fr

RÉSUMÉ. *Le cadre général de notre projet vise à définir un couplage entre un modèle d'exigences exprimé en SysML/KAOS et des spécifications formelles abstraites Event-B. Les buts fonctionnels sont la base de la dérivation de spécifications abstraites Event-B tandis que les buts non-fonctionnels sont injectés dans les modèles abstraits Event-B obtenus afin de les compléter et de les enrichir. Dans cet article, nous présentons les différentes manières de prendre en compte les buts non-fonctionnels et de leurs impacts dans les modèles abstraits Event-B. Nous définissons également des règles de traçabilité entre les exigences non-fonctionnelles et les différents éléments Event-B afin de faciliter la gestion de l'évolution de ces exigences.*

ABSTRACT. *The general framework of our project aims to define a coupling between a requirements model expressed in SysML/KAOS and an abstract Event-B formal specification. The functional goals are the basis for the derivation of abstract Event-B specifications, while non-functional goals are injected into the obtained abstract Event-B models in order to complete and enrich them. In this paper, we present different ways to take into account non-functional goals and their impact into the abstract Event-B models. We define also traceability rules between the non-functional requirements and the different Event-B elements in ordre to improve the management of the evolution of these requirements.*

MOTS-CLÉS : *Ingénierie des exigences, SysML/KAOS, Event-B, But non-fonctionnel, Traçabilité.*

KEYWORDS: *Requirements engineering, SysML/KAOS, Event-B, Non-functional goal, Traceability*

1. Introduction

Dans les méthodes d'ingénierie des exigences, deux types d'exigences peuvent être identifiés. Les exigences fonctionnelles spécifient les fonctions que le système à concevoir doit être en mesure d'effectuer. Les exigences non-fonctionnelles quant à elles capturent les propriétés ou les contraintes sous lesquelles le système à concevoir doit fonctionner telles que les aspects de performance, de qualité ou de sécurité. Les pratiques industrielles actuelles consistent à spécifier les exigences fonctionnelles dès les premières phases de développement logiciel et à laisser la prise en compte des exigences non-fonctionnelles au niveau de l'implémentation. Ce choix est souvent justifiée par l'énorme pression (en terme de temps) pour le déploiement rapide du logiciel [Cysneiros et Kushniruk2003]. Pourtant, la prise en compte des exigences non-fonctionnelles dès les phases initiales du processus de développement augmentent les chances de succès des systèmes logiciels. Dans ce contexte, [Neto *et al.*2000] présente quelques problèmes bien connus dûs à l'omission des exigences non-fonctionnelles durant les premières phases du processus de développement.

Néanmoins, l'analyse et la spécification des exigences non-fonctionnelles n'est pas toujours évidente puisque généralement il est difficile de les exprimer d'une façon objective et mesurable. De plus, les exigences non-fonctionnelles sont plus difficiles à traiter parce que leur impact n'est pas généralement localisé sur une partie du système, mais s'étend sur l'ensemble du système [Aurum et Wohlin2005]. Elles peuvent donc avoir un impact sur plusieurs exigences fonctionnelles et peuvent, de plus, être en conflit les unes avec les autres. Ainsi, nous pensons qu'une distinction entre les deux types d'exigences dès la phase d'analyse des besoins peut être utile. C'est pourquoi notre méthode est basée sur la définition de trois modèles. Le premier modèle décrit les exigences fonctionnelles, le second modèle décrit les exigences non-fonctionnelles et le dernier présente l'impact des exigences non-fonctionnelles sur les exigences fonctionnelles. Afin de spécifier ces différents modèles, [Laleau *et al.*2010, Gnaho et Semmak2010] ont proposé la méthode SysML/KAOS. C'est un profil SysML qui s'inspire de : (i) la méthode KAOS [van Lamsweerde2009] pour représenter les buts fonctionnels ; (ii) la méthode NFR framework [Chung *et al.*2000] pour représenter les buts non-fonctionnels et leurs impacts. L'objectif de nos travaux est de proposer un couplage entre ces modèles et des spécifications formelles abstraites Event-B. Pour cela, nous avons défini dans [Matoussi *et al.*2010a, Matoussi *et al.*2011] une approche qui se sert du premier modèle (décrivant les buts fonctionnels) pour dériver des spécifications formelles abstraites Event-B. Cet article continue ce travail en prenant en compte les buts non-fonctionnels et leurs impacts. Ils vont se traduire par différents éléments Event-B qui vont compléter et enrichir les modèles abstraits Event-B, obtenus à partir des buts fonctionnels. Cet article se focalise ainsi sur la description des différentes manières pour assurer leur prise en compte dans les modèles abstraits Event-B. Pour cela, nous définissons des règles de traçabilité entre les exigences non-fonctionnelles et les différents éléments Event-B afin de faciliter la gestion de l'évolution de ces exigences.

La suite de l'article est organisée comme suit. La section 2 présente quelques préliminaires sur SysML/KAOS et la méthode Event-B. La section 3 présente un aperçu général de l'approche proposée. La section 4 illustre l'approche avec un exemple. Les travaux liés sont présentés dans la section 5. La section 6 conclut le papier en présentant aussi les perspectives.

2. Préliminaires

2.1. *SysML/KAOS*

KAOS (Keep All Objectives Satisfied) est une méthode pour l'ingénierie des exigences qui résulte des travaux de recherche [van Lamsweerde2009] menés à l'Université de Louvain, en collaboration avec l'Université d'Oregon. Cette méthode permet aux analystes de construire des modèles des exigences en s'appuyant sur un raisonnement orienté par les buts. Le modèle des exigences KAOS se compose de cinq sous-modèles fortement liés par des règles de cohérence (le modèle de buts, le modèle objets, le modèle des responsabilités, le modèle des opérations et le modèle des comportements). Le modèle central dans KAOS est *le modèle de buts* qui décrit les buts du système et son environnement. Ce modèle est organisé dans une hiérarchie obtenue grâce au raffinement AND/OR de buts de plus haut niveau (les buts stratégiques) vers des buts de bas niveau (les exigences). Un raffinement AND implique que la conjonction des sous-buts est une condition suffisante pour réaliser le but parent. Un raffinement OR associe un but à des sous-buts alternatifs pour lesquels l'accomplissement du but de plus haut niveau exige l'accomplissement d'au moins un de ses sous-buts. KAOS offre un ensemble de patrons de raffinement [van Lamsweerde2009] qui décomposent les buts. Ces patrons ne peuvent être utilisés que dans le contexte de différentes tactiques tel que le raffinement *par jalons* (MILESTONE en anglais) qui consiste à identifier les sous-buts comme des étapes successives dans le temps permettant de satisfaire le but de plus haut niveau. Les buts dans KAOS sont classifiés en buts de type *Achieve*, *Maintain*, *Cease* ou *Avoid*. Un but *Achieve* (resp. *Cease*) spécifie une propriété que le système réalisera (resp. ne réalisera pas) nécessairement un jour dans le futur. Un but *Maintain* (resp. *Avoid*) spécifie une propriété qui doit être toujours vraie (resp. fausse). Un principe de la méthodologie KAOS est d'arrêter le raffinement lorsqu'il est possible d'assigner chaque exigence à un agent particulier. Le lecteur peut se référer à [van Lamsweerde2009] pour une description plus détaillée de ces notions.

Cependant, KAOS ne considère que la phase d'analyse des exigences, à la différence de SysML par exemple qui permet d'assurer une continuité depuis l'analyse des exigences jusqu'à l'implémentation. SysML [Friedenthal *et al.*2008] se définit comme un langage de modélisation pour l'analyse et la spécification de systèmes complexes. SysML est un profil UML, qui utilise certains diagrammes UML et propose aussi des extensions. Une des extensions proposées concerne la prise en compte des exigences. Plusieurs concepts sont proposés pour la représentation des exigences (fonctionnelles ou non-fonctionnelles) et la traçabilité avec d'autres éléments de spécification. Malgré

ses apports, l'ensemble des concepts proposés dans SysML n'est pas aussi riche que dans les autres méthodes d'ingénierie des exigences : (i) les exigences sont spécifiées de manière informelle sous forme de texte ; (ii) la sémantique des relations entre exigences (comme par exemple la relation de contenance) n'est pas définie de manière précise, ce qui entraîne des ambiguïtés [Goknil *et al.*2011].

Un des avantages de SysML est qu'on peut l'étendre facilement en utilisant le mécanisme de profils, comme en UML. Ainsi, [Laleau *et al.*2010] proposent d'étendre SysML tout d'abord avec les concepts du modèle de buts de KAOS, en étendant le méta-modèle des exigences de SysML. Le méta-modèle étendu (le méta-modèle SysML/KAOS) permet de construire des hiérarchies d'exigences fonctionnelles sous forme de buts. [Gnaho et Semmak2010] complètent par la suite le méta-modèle SysML/-KAOS par les concepts liés aux buts non-fonctionnels et l'impact de ces buts sur les buts fonctionnels, en s'inspirant de la méthode NFR framework [Chung *et al.*2000]. Pour supporter ces extensions, un outil SysML/KAOS [Gnaho et Semmak2010] a été développé dans l'environnement TOPCASED [1].

2.2. *Event-B*

La méthode B classique [Abrial1996] est une méthode formelle de modélisation et de construction de logiciels. Elle s'appuie sur les concepts mathématiques de la théorie des ensembles. Event-B [Abrial2010] est une évolution de la méthode B, adaptée à la modélisation de systèmes par raffinement. Une spécification Event-B est décomposée en deux parties : (i) *le contexte* qui contient la partie statique du modèle telle que les ensembles énumérés, les constantes et les axiomes ; (ii) *la machine* qui contient la partie dynamique telle que les variables et les événements. L'approche Event-B permet non seulement de modéliser le système, mais aussi son environnement à travers la notion d'observation des événements. Chaque événement est décrit sous la forme d'action gardée et il est susceptible d'être déclenché quand sa garde devient vraie. L'état du système change en fonction de l'action associée à l'événement. Cependant, il est nécessaire de vérifier que chaque événement dans le système préserve les propriétés d'invariance du modèle Event-B. Cette vérification passe par la réalisation des obligations de preuve correspondantes. En Event-B, le raffinement est un processus qui transforme une spécification abstraite et non-déterministe en un système concret et déterministe qui préserve les fonctionnalités de la spécification originale. Pendant le raffinement, les événements abstraits sont raffinés pour pouvoir prendre en compte de nouvelles variables. De nouveaux événements peuvent aussi être observés. Les obligations de preuves sont générées à chaque étape de raffinement afin d'assurer la cohérence du raffinement. Event-B fournit actuellement un logiciel libre sous la forme d'une plate-forme (basée sur « Eclipse ») de spécification et de preuve, appelé RODIN[2].

1. http ://www.topcased.org/
2. http ://www.event-b.org/

La Figure 1 donne une vue du méta-modèle Event-B[3] en se focalisant sur le package « Machine ». La méta-classe **Machine** représente les machines Event-B et l'association réflexive **refines** exprime qu'une machine peut en raffiner une autre. L'association **sees** exprime le fait qu'une machine Event-B peut accéder à des contextes Event-B. De plus, une machine Event-B peut contenir des variables, des invariants, un variant et plusieurs événements. Un événement peut contenir des paramètres, des gardes, des témoins (Witness) et des actions. Un événement peut raffiner un autre événement.

3. Un aperçu général de l'approche

Nous avons proposé dans [Matoussi *et al.*2010a, Matoussi *et al.*2011] une approche constructive dans laquelle des modèles formels abstraits exprimés en Event-B sont construits progressivement à partir des buts fonctionnels SysML/KAOS. Dans cet article, nous complétons les modèles abstraits Event-B obtenus en les enrichissant avec les buts non-fonctionnels. Pour cela, nous décrivons différentes manières de prendre en compte les buts non-fonctionnels et de leurs impacts dans ces modèles abstraits Event-B, obtenus à partir des buts fonctionnels. Des règles de traçabilité (les boîtes grises de la Figure 1) sont également définies entre les buts non-fonctionnels SysML/-KAOS et les éléments Event-B. Cette traçabilité explicite permet de faciliter la gestion de l'évolution des exigences non-fonctionnelles.

3.1. *Modèle de buts SysML/KAOS (buts fonctionnels et non-fonctionnels)*

Un but fonctionnel SysML/KAOS peut être un but « Achieve » ou sa variante duale « Cease ». Un but « Achieve » prescrit des comportements attendus où une certaine ***TargetCondition*** doit être tôt ou tard établie quand une autre condition (***CurrentCondition***) est vérifiée dans l'état actuel du système (cet état actuel est arbitraire). Un but « Achieve » dans SysML/KAOS est dénoté comme suit : *Achieve[**TargetCondition** From **CurrentCondition**]*. La description temporelle et informelle de cette notation est la suivante, sachant que le préfixe ***CurrentCondition*** est optionnel (autrement dit, il peut être vrai) : *[Si **CurrentCondition** alors] tôt ou tard **TargetCondition***.

Comme le montre la Figure 1, un but non-fonctionnel peut être soit un but abstrait (méta-classe **Abstract NFG**), soit un but élémentaire (méta-classe **Elementary NFG**). Un but élémentaire peut être associé à un but de contribution (méta-classe **Contribution Goal**) qui représente une manière dont le but élémentaire peut être satisfait. Le concept **Impact** permet de relier les buts non-fonctionnels aux buts fonctionnels. Il représente le fait qu'un but de contribution a un effet sur un but fonctionnel.

---

3. http ://wiki.event-b.org/index.php/EMF_framework_for_Event-B

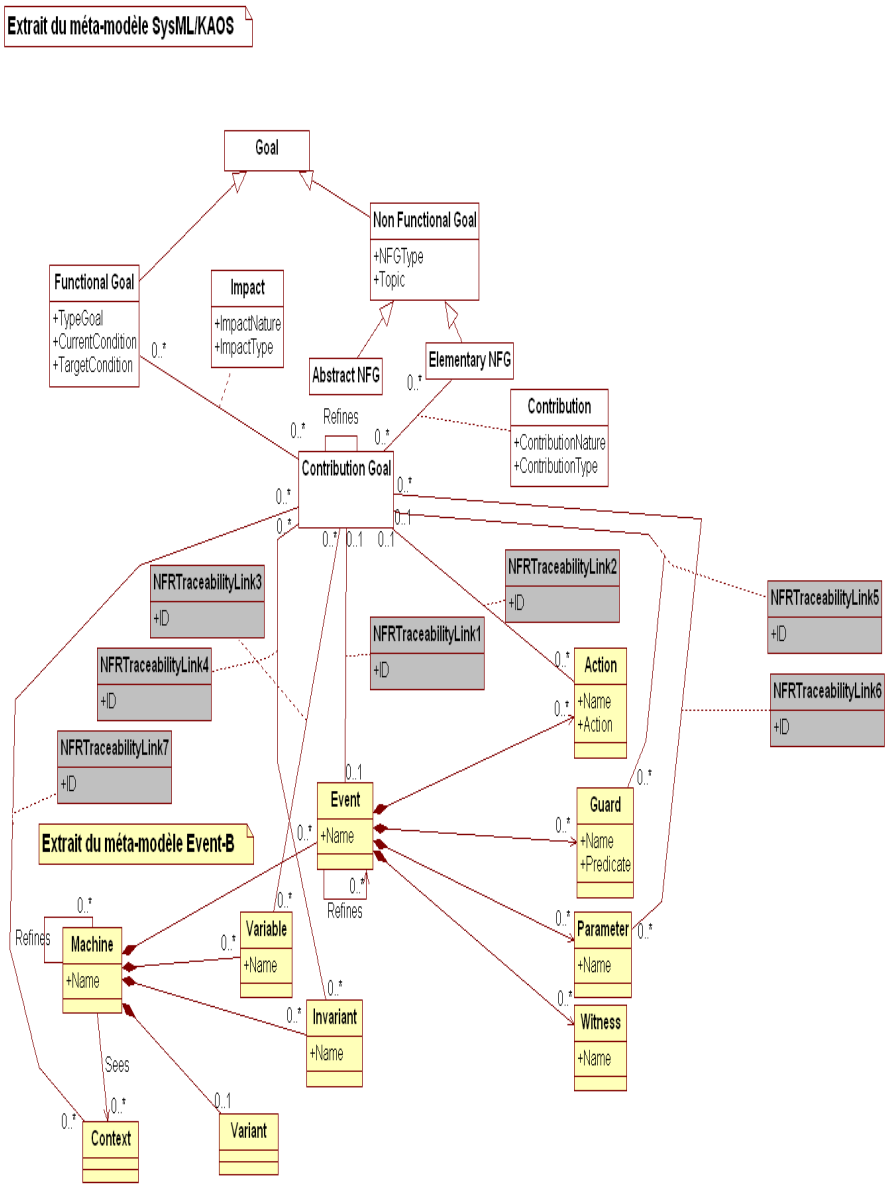

Figure 1. *Un aperçu général des liens entre les méta-modèles SysML/KAOS et Event-B pour les buts non-fonctionnels*

Pour mieux illustrer ces concepts, on se réfère au modèle de buts SysML/KAOS extrait d'une étude de cas réalisée dans le cadre du projet de recherche, TACOS[4]. L'étude de cas [Matoussi *et al.*2010b] consiste à spécifier un composant de localisation qui est une partie critique d'un système de transport terrestre. La Figure 2 montre un extrait du modèle de buts SysML/KAOS d'un composant de localisation. Ce modèle de buts est obtenu par la construction en parallèle du modèle de buts fonctionnels et du modèle de buts non-fonctionnels comme le souligne [Gnaho et Semmak2010]. Dans ce qui suit, nous décrivons chaque but d'une façon informelle en langage naturel **InformalDef**.

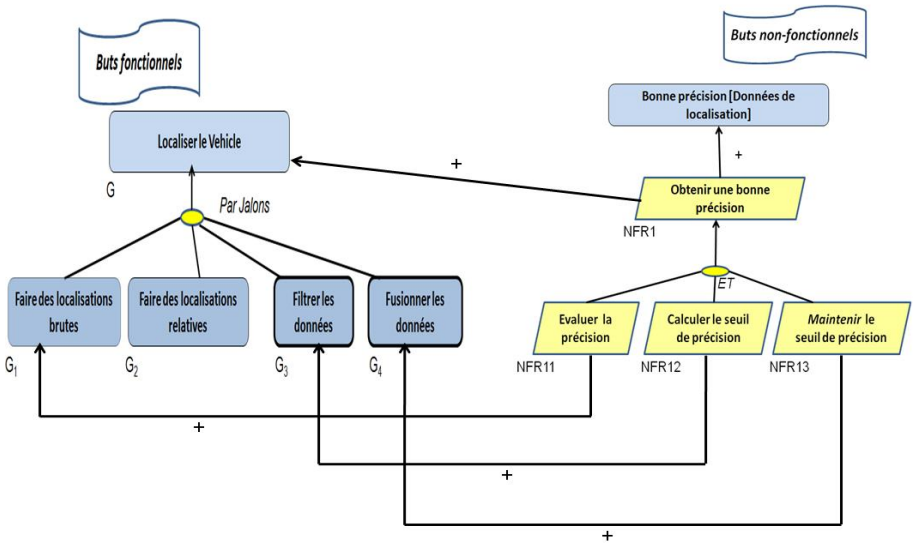

Figure 2. *Un extrait du modèle de buts SysML/KAOS (fonctionnels et non-fonctionnels) du composant de localisation*

La Figure 2 montre que le but fonctionnel le plus stratégique G est défini en SysML/KAOS comme suit :

Functional Goal G : Achieve [Localiser le Véhicule]
InformalDef : Le véhicule doit être localisé.

Ce but G est raffiné en quatre sous-buts selon le raffinement par jalons :

Functional Goal G_1 : Achieve [Faire des localisations brutes]
InformalDef : D'abord, des localisations brutes sont faites.

Functional Goal G_2 : Achieve [Faire des localisations relatives]
InformalDef : Puis, des localisations relatives sont faites.

4. http ://tacos.loria.fr

Functional Goal G_3 : Achieve [Filtrer les données]
InformalDef : Ensuite, toutes les données de localisations sont filtrées et vérifiées en se basant sur les localisations relatives.

Functional Goal G_4 : Achieve [Fusionner les données]
InformalDef : Enfin, toutes les données sont fusionnées afin d'obtenir une localisation finale.

La Figure 2 montre que le but de contribution *Obtenir une bonne précision* est une contribution directe positive au but élémentaire non-fonctionnel *Bonne précision [Données de localisation]*. Ce but de contribution $NFR1$ est raffiné en trois buts de contribution $NFR11$, $NFR12$, $NFR13$. Les impacts de ces différents buts de contribution sur les buts fonctionnels sont définis comme suit :

Contribution Goal $NFR1$: *Obtenir une bonne précision*
InformalDef : Il impacte positivement et directement le but fonctionnel G.

Contribution Goal $NFR11$: *Evaluer la précision*
InformalDef : Il impacte positivement et directement le but fonctionnel G_1.

Contribution Goal $NFR12$: *Calculer le seuil de précision*
InformalDef : Il impacte positivement et directement le but fonctionnel G_3.

Contribution Goal $NFR13$: *Maintenir le seuil de précision*
InformalDef : Il impacte positivement et directement le but fonctionnel G_4.

3.2. *Transformation des buts*

Notre approche de transformation des buts fonctionnels [Matoussi *et al.*2011] repose sur l'idée que chaque niveau i ($i \in [1..n]$) dans la hiérarchie de buts fonctionnels SysML/KAOS est représenté comme un modèle Event-B M_i qui raffine le modèle Event-B M_{i-1} lié au niveau $i-1$. Nous représentons chaque but fonctionnel comme un événement Event-B où : (i) la ***CurrentCondition*** du but est considérée comme la garde de l'événement ; (ii) La partie **then** traduit la ***TargetCondition*** de ce but. Jusqu'à présent, les autres éléments d'Event-B (l'initialisation, les variables et leurs invariants de typage, et les contextes) sont manuellement complétés par le concepteur. Il serait possible de les dériver à partir du modèle objet KAOS qui contient tous les concepts utilisés dans la définition des buts. Comme le modèle objet KAOS est un diagramme de classes UML, l'idée est de réutiliser par exemple les travaux UML-B définies par [Mammar et Laleau2006, Snook et Butler2006]. Le lecteur peut se référer à [Matoussi *et al.*2011, Matoussi *et al.*2010a] qui décrivent en détails la traduction des buts fonctionnels.

Un but non-fonctionnel, et plus précisément les but de contributions associés, peut être pris en compte en Event-B de différents manières :

– *Un événement* : Un but de contribution peut se traduire en Event-B par un nouvel événement. Cela concerne surtout les buts de contribution exprimant qu'une opération

doit être réalisée.

– *Des actions dans des événements existants* : Un but de contribution peut se traduire par une nouvelle action d'un événement lié à un but fonctionnel déjà existant. Ce type de traduction concerne souvent les buts fonctionnels impactés par la nécessité de satisfaire une action exprimée par le but de contribution.

– *Des gardes dans des événements existants* : Un but de contribution peut se traduire par une nouvelle garde d'un événement lié à un but fonctionnel déjà existant.

– *Des paramètres dans des événements existants* : Lorsqu'il impacte une action ou une garde d'un événement existant, un but de contribution peut faire apparaître de nouvelles variables locales (les paramètres).

– *Des variables* : Un but de contribution peut ajouter de nouvelles variables.

– *Des invariants* : L'ajout de nouvelles variables implique au moins l'ajout des invariants de typage de ces variables. De plus, si le but de contribution exprime une propriété à maintenir, il se traduit alors par un nouvel invariant (une contrainte).

– *Des éléments d'un contexte* : Un but de contribution peut se traduire dans le contexte Event-B par l'ajout de constantes, d'ensembles ou d'axiomes.

En se basant sur la forme de l'impact des buts de contribution sur les buts fonctionnels en SysML/KAOS, un seul but de contribution peut se traduire par un ou plusieurs concepts Event-B à la fois. Une autre remarque importante concerne les buts de contribution exprimant des propriétés de vivacité. Ces buts ne peuvent pas être exprimés comme des invariants Event-B mais seront considérés plutôt comme des expressions temporelles LTL vérifiables par model-checking, en utilisant par exemple le model-checker ProB [Leuschel et Butler2003].

4. Illustration de la traçabilité sur l'exemple

Pour une meilleure illustration, les règles de traçabilité sont appliquées sur le modèle de buts SysML/KAOS du composant de localisation (voir la Figure 2). Pour chaque niveau de la hiérarchie de buts SysML/KAOS, nous décrivons dans ce qui suit comment assurer la traçabilité entre les buts (fonctionnels et non-fonctionnels) et les modèles Event-B.

4.1. *Modèle abstrait*

4.1.1. *Transformation des buts fonctionnels*

Un modèle abstrait Event-B *Localisation* (dans la Figure 3) est associé à ce niveau le plus abstrait de la hiérarchie du modèle de buts SysML/KAOS. Dans ce modèle Event-B, un événement appelé **Localiser le Véhicule** traduit ce but ; c'est-à-dire qu'il va décrire « le travail » à faire pour atteindre le but G, en termes de substitutions généralisées. La localisation d'un véhicule consiste à obtenir une *estimated_loc* qui est un couple (latitude, longitude). A ce niveau d'abstraction, il n'est pas néces-

MACHINE Localisation
SEES TypeSets
VARIABLES
`estimated_loc`
`precision` // Variable créée par NFR1
INVARIANTS
`inv1 :` $estimated\_loc \in LATITUDE \times LONGITUDE$
`invNFR1 :` $precision \in \mathbb{N}$ // Invariant de typage pour NFR1
EVENTS
Initialisation
begin
`act1 :` $estimated\_loc := null \mapsto null$
`actiniNFR1 :` $precision := 0$
end
Event *Localiser le Véhicule* $\widehat{=}$ // Evénement impacté par NFR1
begin
`act1 :` $estimated\_loc :\in (LATITUDE \setminus \{null\}) \times (LONGITUDE \setminus \{null\})$
`actNFR1 :` $precision :\in \mathbb{N}_1$
end
END

Figure 3. *Le modèle Event-B abstrait*

CONTEXT TypeSets
SETS
`SUBCOMPONENTS`
`SUBSENSORS`
CONSTANTS
`null`, `LATITUDE`, `LONGITUDE`
`null`, `speed`, `accel`
AXIOMS
`axm1 :` $partition(SUBCOMPONENTS, \{gps\}, \{wifi\})$
`axm2 :` $LATITUDE = \mathbb{N} \cup \{null\}$
`axm3 :` $LONGITUDE = \mathbb{N} \cup \{null\}$
`axm4 :` $partition(SUBSENSORS, \{speed\}, \{accel\})$
END

Figure 4. *Le context Event-B TypeSets*

saire de préciser la façon dont cette information est calculée. Ainsi, nous utilisons la substitution généralisé non déterministe à travers le symbole $:\in$ qui spécifie un choix non borné : *estimated_loc* peut prendre n'importe quelle valeur dans l'ensemble $((LATITUDE \setminus \{null\}) \times$(LONGITUDE $\setminus \{null\})$). La valeur $null$ sert à l'initialisation du système. Notons qu'à ce niveau d'abstraction l'événement **Localiser le Véhicule** peut toujours se produire. Par conséquent, sa garde est toujours vraie. Les ensembles en majuscules sont des ensembles abstraits utilisés pour typer les variables et sont décrits dans le contexte Event-B *Type_sets* comme le montre la Figure 4.

4.1.2. *Prise en compte des buts non-fonctionnels*

Comme l'impact du but de contribution $NFR1$ sur le but fonctionnel G consiste en une donnée à satisfaire, l'événement Event-B traduisant le but G **Localiser le Véhicule** ajoute une nouvelle action *actNFR1* qui va assurer que la valeur de la précision est positive ($precision :\in \mathbb{N}_1$). Une nouvelle variable *precision*, un nouvel invariant de typage *invNFR1* et une initialisation de la variable *actiniNFR1* sont aussi ajoutés.

4.2. *Premier raffinement*

4.2.1. *Transformation des buts fonctionnels*

Un modèle de raffinement Event-B *Localisation1* (voir la Figure 5) est associé à ce premier niveau de la hiérarchie du graphe de buts SysML/KAOS. Le sous-buts G_1, G_2, G_3 et G_4 sont représentés par quatre événements Event-B **Faire des localisations brutes**, **Faire des localisations relatives**, **Filtrer les données** et **Fusionner les données**, respectivement. Le premier renvoie un ensemble de couples (latitude, longitude), un pour chaque composant utilisé pour la localisation brute d'un véhicule. Le second renvoie un autre ensemble de couples (latitude, longitude), un pour chaque composant utilisé pour la localisation relative d'un véhicule. En se basant sur les valeurs relatives renvoyées, le troisième événement filtre l'ensemble retourné des valeurs brutes en choisissant les valeurs acceptables. Le dernier événement renvoie la localisation finale calculée à partir des valeurs retournées de **Filtrer les données**.

4.2.2. *Prise en compte des buts non-fonctionnels*

Comme l'impact du but de contribution $NFR11$ sur le but fonctionnel **Faire des localisations brutes** consiste à une donnée à satisfaire, l'événement Event-B **Faire des localisations brutes** ajoute une nouvelle action *actNFR11* qui va assurer la satisfaction de cette donnée. Comme le montre la Figure 5, une nouvelle variable *precision_brute*, un nouvel invariant de typage *invNFR11* et une initialisation de la variable *actiniNFR11* sont également ajoutés.

Une fois les localisations relatives faites, le but de contribution $NFR12$ indique qu'il faut calculer un seuil de précision pour que le but fonctionnel **Filtrer les données** puisse assurer le filtrage des données. Comme le montre la Figure 5, ce but de contribution va se traduire alors en Event-B par un nouvel événement Event-B **Calculer le seuil de précision** incluant l'action *actNFR12*, une nouvelle variable *seuil_precision*, un nouvel invariant de typage *invNFR12* et une initialisation de la variable *actiniNFR12*. De plus, une nouvelle garde *grdNFR12* est ajoutée dans l'événement **Filtrer les données** pour s'assurer que cet événement ne peut se déclencher qu'après l'achèvement de l'événement **Calculer le seuil de précision**.

Le but de contribution $NFR13$ indique quant à lui qu'il faut s'assurer que la localisation finale retournée par le but fonctionnel **Fusionner les données** doit être toujours au moins égale au seuil de précision déjà calculé. Comme le montre la Figure 5, cette

forme d'impact se traduit en Event-B par l'ajout de : (i) un nouvel invariant *invNFR13* exprimant cette propriété ; (ii) une nouvelle action *actNFR13* dans l'événement Event-B **Fusionner les données** afin de respecter l'invariant *invNFR13*.

MACHINE Localisation1
REFINES Localisation
SEES TypeSets
VARIABLES
`subcomponents_loc`, `sensors_loc`, `kept_loc`, `merged_loc`
`precision_brute` // Variable créée par NFR11
`seuil_precision` // Variable créée par NFR12

INVARIANTS
`inv1` : $subcomponents\_loc \in SUBCOMPONENTS \rightarrow (LATITUDE \times LONGITUDE)$
`inv2` : $sensors\_loc \in SUBSENSORS \nrightarrow (LATITUDE \times LONGITUDE)$
`inv3` : $kept\_loc \in SUBCOMPONENTS \nrightarrow (LATITUDE \times LONGITUDE)$
`inv4` : $merged\_loc \in LATITUDE \times LONGITUDE$
`inv5` : $estimated\_loc = merged\_location$
`invNFR11` : $precision\_brute \in SUBCOMPONENTS \rightarrow \mathbb{N}$
`invNFR12` : $seuil\_precision \in \mathbb{N}$
`invNFR13` : $(merged\_loc \in (LATITUDE \setminus \{null\}) \times (LONGITUDE \setminus \{null\})) \Rightarrow$
$(precision >= seuil\_precision)$ // Invariant pour NFR13

EVENTS
Initialisation
begin
`act2` : $subcomponents\_loc :\in SUBCOMPONENTS \rightarrow (\{null\} \times \{null\})$
`act3` : $sensors\_loc :\in SUBSENSORS \rightarrow (\{null\} \times \{null\})$
`act4` : $kept\_loc :\in SUBCOMPONENTS \rightarrow (\{null\} \times \{null\})$
`act5` : $merged\_loc := null \mapsto null$
`actiniNFR11` : $precision\_brute :\in SUBCOMPONENTS \rightarrow 0$
`actiniNFR12` : $seuil\_precision := 0$

Event *Faire des localisations brutes* $\widehat{=}$ // Evénement impacté par NFR11
begin
`act1` : $subcomponents\_loc :\in SUBCOMPONENTS \rightarrow ((LATITUDE \setminus \{null\}) \times$
$(LONGITUDE \setminus \{null\}))$
`actNFR11` : $precision\_brute :\in SUBCOMPONENTS \rightarrow \mathbb{N}$
end

Event *Faire des localisations relatives* $\widehat{=}$
when
`grd1` : $subcomponents\_loc \in SUBCOMPONENTS \rightarrow ((LATITUDE \setminus \{null\}) \times$
$(LONGITUDE \setminus \{null\}))$
then
`act1` : $sensors\_loc : |(sensors\_loc' \in SUBSENSORS \nrightarrow ((LATITUDE \setminus \{null\}) \times$
$(LONGITUDE \setminus \{null\}))) \wedge sensors\_loc' \neq \varnothing$
end

Event *Calculer le seuil de précision* $\widehat{=}$ // Evénement ajouté traduisant NFR12
when
`grd1` : $sensors\_loc \in SUBSENSORS \nrightarrow ((LATITUDE \setminus \{null\}) \times$
$(LONGITUDE \setminus \{null\})) \wedge sensors\_loc \neq \varnothing$
then
`actNFR12` : $seuil\_precision :\in \mathbb{N}_1$
end

Event *Filtrer les données* $\widehat{=}$

when

grd1 : $sensors\_loc \in SUBSENSORS \nrightarrow ((LATITUDE \setminus \{null\}) \times (LONGITUDE \setminus \{null\})) \land sensors\_loc \neq \varnothing$

grdNFR12 : $seuil\_precision \in \mathbb{N}_1$

then

act1 : $kept\_loc :\in \mathbb{P}_1(subcomponents\_loc)$

end

Event *Fusionner les données* $\widehat{=}$ // Evénement impacté par NFR13

when

grd1 : $kept\_loc \in \mathbb{P}_1(subcomponents\_loc)$

grd2 : $subcomponents\_loc \in SUBCOMPONENTS \rightarrow ((LATITUDE \setminus \{null\}) \times (LONGITUDE \setminus \{null\}))$

then

act1 : $merged\_loc :\in (LATITUDE \setminus \{null\}) \times (LONGITUDE \setminus \{null\})$

actNFR13 : $precision : |(precision' \in \mathbb{N}_1) \land (precision' >= seuil\_precision)$

end

Figure 5. *Le premier modèle de raffinement Event-B*

4.2.3. *Obligations de preuves de raffinement*

Dans Event-B, les informations sur l'ordonnancement des événements peuvent être exprimées dans les gardes et les actions des événements en utilisant des variables de contrôle. Dans notre approche [Matoussi *et al.*2011], nous avons proposé une structure de raffinement Event-B pour les patrons de raffinement des buts fonctionnels SysML/KAOS parmi lesquels le patron *par jalons*. Chaque structure de raffinement Event-B proposée est justifiée en construisant des modèles mathématiques ensemblistes basés sur la sémantique des traces du développement Event-B [Abrial2010] et en identifiant les obligations de preuve systématiques. Dans notre exemple, l'événement abstrait **Localiser le Véhicule** est raffiné par la séquence de nouveaux événements (relatifs aux buts fonctionnels et non-fonctionnels) comme suit :

(**Faire des localisations brutes** ; **Faire des localisations relatives** ; **Calculer le seuil de précision** ; **Filtrer les données** ; **Fusionner les données**) *Refines* **Localiser le Vehicule**

Les injections des buts non-fonctionnels dans la machines Event-B peuvent avoir des effets sur les obligations de preuves déjà prouvées au niveau des buts fonctionnels. Pour cela, nous sommes en train d'étudier les différents cas d'injection des buts non-fonctionnels dans les machines Event-B. Cette étude a révélé jusqu'à maintenant un certain nombre de règles, de recommandation et de techniques permettant de préserver au maximum les obligations de preuves déjà faites sans avoir à les refaire.

5. Travaux similaires

Notre approche vise à assurer la traçabilité entre les exigences et les spécifications formelles. Dans ce qui suit, nous présentons quelques travaux qui se situent dans le même contexte que le notre.

[Jastram *et al.*2009] présente une approche assurant la traçabilité entre les exigences exprimées en langage naturel (NLRs) et les modèles Event-B. L'approche commence par considérer les exigences liées à l'environnement qui sont prises en compte dans Event-B comme étant des structures de données (des ensembles, des constantes et des variables). Les exigences de sécurité quant à elles sont considérées comme des invariants Event-B tandis que les exigences de vivacité sont exprimées comme des expressions temporelles LTL vérifiables par le model-checker ProB. De plus, l'approche explore manuellement les verbes figurant dans les exigences afin de dériver les noms des événements Event-B. Les invariants et les formules LTL guident alors la création des gardes et des actions de ces événements. Ces événements peuvent concernés donc plusieurs exigences à la fois. Une extension de l'approche est récemment présentée dans [Jastram *et al.*2010] qui se base essentiellement sur WRSPM comme modèle de référence pour l'application des méthodes formelles dans le développement des exigences de l'utilisateur. Quelques autres travaux [Hassan2009, Devroey2010, Ponsard et Dieul2006] ont essayé de se servir de l'approche GORE afin de dériver des modèles formels. La plupart de ces travaux, hormis [Devroey2010], ne mettent pas l'accent sur une traçabilité explicite entre les phases d'analyse des exigences et de spécification. Le travail de [Devroey2010], inspirée du [Ponsard et Dieul2006], définit des liens de traçabilité verticales entre une partie du modèle des exigences KAOS et les machines Event-B. La première étape de l'approche consiste à dériver une machine et un contexte Event-B initiaux afin de représenter l'ensemble du modèle objet KAOS en reprenant les règles de transformation présentées en UML-B [Snook et Butler2006]. Une traçabilité verticale est créée alors entre les différents éléments du modèle initial Event-B (constantes, événements...) et les différents éléments du modèle objet KAOS (objet abstrait, attribut du domaine, association...). Dans la deuxième étape, la machine Event-B initiale est décomposée en plusieurs machines Event-B où chaque machine est lié à un « agent » KAOS. La dernière étape de l'approche consiste à créer un raffinement Event-B pour chaque machine liée à « agent » KAOS dans laquelle l'événement de type *interne* est lié à une exigence ou une attente sous la responsabilité de l'agent KAOS en question.

Si la traçabilité définie par [Jastram *et al.*2010] entre les exigences et la machine Event-B abstraite est bien claire, l'absence d'une méthode structurant les exigences rend la traçabilité avec les autres niveaux de raffinement Event-B plus difficile et ambigüe. Pourtant, les auteurs soulignent qu'une hiérarchisation des verbes extraites des exigences peut aider à résoudre le problème. La traçabilité présentée par [Devroey2010] quant à elle reste partielle parce qu'il ne considère pas toutes les parties du modèle de buts KAOS, mais seulement les buts opérationnels (les buts feuilles). Par conséquent, le modèle formel n'inclut aucune information sur les buts non-opérationnels et surtout sur le type de raffinement de buts. De plus, ces travaux ne définissent pas des frontières

entre les exigences fonctionnelles et non-fonctionnelles. Dans cet article, nous avons exploré comment faire face à ces problèmes en utilisant une nouvelle approche qui assure la traçabilité entre tout le modèle de buts SysML/KAOS et le modèle formel en mettant l'accent sur la séparation entre les exigences fonctionnelles de celles non-fonctionnelles. Ainsi, ce que nous présentons peut être très utile en pratique pour vérifier que : (i) toutes les exigences SysML/KAOS (fonctionnelles et non-fonctionnelles) sont représentées dans le modèle Event-B ; (ii) la majorité des éléments dans le modèle Event-B ont des correspondants dans SysML/KAOS.

6. Conclusion et perspectives

Dans cet article, nous avons proposé une première solution pour prendre en compte les buts non-fonctionnels et leur impact sur les buts fonctionnels afin de compléter et enrichir les modèles abstraits Event-B, obtenus à partir des buts fonctionnels. Notre proposition consiste à étiqueter les éléments de la spécification Event-B qui sont liés aux buts non-fonctionnels. Ces étiquettes permettent de localiser dans la spécification formelle les impacts des buts non-fonctionnels sur les buts fonctionnels. Les différents liens de traçabilité ainsi définis peuvent faciliter la gestion de l'évolution des exigences non-fonctionnelles. D'un autre côté, l'utilisation de SysML/KAOS comme point d'entrée pour notre approche peut nous permettre de bénéficier des apports de SysML en terme de traçabilité depuis l'analyse des exigences jusqu'à l'implémentation. C'est un point que nous sommes en train d'étudier.

Un outil support SysKAOS2EventB [Matoussi *et al.*2011] a été développé. Il permet de générer, à partir des buts fonctionnels exprimés en SysML/KAOS, les spécifications abstraites Event-B. Nous comptons étendre cet outil afin de prendre en compte les buts non-fonctionnels. La suite du travail va consister également d'une part à améliorer et compléter les résultats actuels, d'autre part à appliquer l'approche sur des études de cas plus complexes.

7. Bibliographie

[Abrial1996] Abrial J. R., *The B-Book : Assigning programs to meanings*, Cambridge University Press, 1996.

[Abrial2010] Abrial J. R., *Modeling in Event-B : System and Software Engineering*, Cambridge University Press, 2010.

[Aurum et Wohlin2005] Aurum A., Wohlin C., *Engineering and Managing Software Requirements*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[Chung *et al.*2000] Chung L., Nixon B. A., Yu E., Mylopoulos J., *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishers, 2000.

[Cysneiros et Kushniruk2003] Cysneiros L. M., Kushniruk A., « Bringing Usability to the Early Stages of Software Development », *RE*, IEEE Computer Society, 2003, p. 359-360.

[Devroey2010] Devroey X., « Building a bridge between Goal-Oriented Requirements with KAOS and Event-B System Specifications », Master's thesis, Facultés Universitaires Notre-

Dame de la Paix Faculté d'Informatique, Namur, Belgium, 2010.

[Friedenthal *et al.*2008] Friedenthal S., Moore A., Steiner R., *A Practical Guide to SysML : The Systems Modeling Language*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[Gnaho et Semmak2010] Gnaho C., Semmak F., « Une extension SysML pour l'ingénierie des exigences dirigée par les buts », *INFORSID'10*, 2010, p. 277-292.

[Goknil *et al.*2011] Goknil A., Kurtev I., Berg K., Veldhuis J.-W., « Semantics of trace relations in requirements models for consistency checking and inferencing », *Softw. Syst. Model.*, vol. 10, 2011, p. 31–54, Springer-Verlag New York, Inc.

[Hassan2009] Hassan R., « Formal Analysis and Design for Engineering Security (FADES) », PhD thesis, Virginia Polytechnic Institute and State University, Virginia, USA, 2009.

[Jastram *et al.*2009] Jastram M., Leuschele M., Bendisposto J., Russo A. G., « Mapping Requirements to B models », rapport n° 104, 2009, DEPLOY project.

[Jastram *et al.*2010] Jastram M., Hallerstede S., Leuschel M., Russo A. G., « An Approach of Requirements Tracing in Formal Refinement », Leavens G. T., O'Hearn P. W., Rajamani S. K., Eds., *VSTTE*, vol. 6217 de *LNCS*, Springer, 2010, p. 97-111.

[Laleau *et al.*2010] Laleau R., Semmak F., Matoussi A., Petit D., Hammad A., Tatibouët B., « A First Attempt to Combine SysML Requirements Diagrams and B », *ISSE*, vol. 6, n° 1-2, 2010, p. 47-54.

[Leuschel et Butler2003] Leuschel M., Butler M. J., « ProB : A Model Checker for B », Araki K., Gnesi S., Mandrioli D., Eds., *FME*, vol. 2805 de *LNCS*, Springer, 2003, p. 855-874.

[Mammar et Laleau2006] Mammar A., Laleau R., « A Formal Approach Based on UML and B for the Specification and Development of Database Applications », *Automated Software Engg.*, vol. 13, 2006, p. 497–528, Kluwer Academic Publishers.

[Matoussi *et al.*2010a] Matoussi A., Gervais F., Laleau R., « An Event-B formalization of KAOS goal refinement patterns », rapport n° TR-LACL-2010-1, 2010, LACL, University of Paris-Est.

[Matoussi *et al.*2010b] Matoussi A., Gervais F., Laleau R., « Specification of a Localization Component Driven by a Goal-Based Approach : Some Lessons We Learned », Davies J., Silva L., da Silva Simão A., Eds., *SBMF*, vol. 6527 de *Lecture Notes in Computer Science*, Springer, 2010, p. 177-193.

[Matoussi *et al.*2011] Matoussi A., Gervais F., Laleau R., « A Goal-Based Approach to Guide the Design of an Abstract Event-B Specification », *Proceedings of the 16th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, IEEE Computer Society. To appear, 2011.

[Neto *et al.*2000] Neto J. D. M. S., Leite J. C. S., Cysneiros L. M., « Non-Functional Requirements for Object-Oriented Modeling », *WER*, 2000, p. 109-125.

[Ponsard et Dieul2006] Ponsard C., Dieul E., « From Requirements Models to Formal Specifications in B », *REMO2V'2006, Luxembourg*, 2006.

[Snook et Butler2006] Snook C., Butler M., « UML-B : Formal modeling and design aided by UML », *ACM Trans. Softw. Eng. Methodol.*, vol. 15, 2006, p. 92–122, ACM.

[van Lamsweerde2009] van Lamsweerde A., *Requirements Engineering : From System Goals to UML Models to Software Specifications*, Wiley, 2009.