Du cahier des charges à sa spécification

Imen Sayar, Jeanine Souquières

Du Cahier des Charges à sa Spécification

Imen Sayar et Jeanine Souquières

LORIA – CNRS UMR 7503 – Université de Lorraine
Campus Scientifique, BP 239
F-54506 Vandœuvre lès Nancy cedex

Firstname.Lastname@loria.fr

Résumé. Le cahier des charges est un document de référence tout au long du développement d'un système. Sa place commence par sa compréhension et sa structuration. Les liens entre ce document et la spécification en Event-B définie en termes de ses différents raffinements nous amène à utiliser des outils de vérification et de validation. Nous présentons quelques leçons sur le développement de la machine d'hémodialyse.

Mots clés. Cahier des charges, développement, spécification, raffinement, validation, vérification, outils.

1 Introduction

La compréhension des besoins est indispensable pour démarrer un processus formel de développement de logiciels. La qualité du cahier des charges, appelé *CdC* dans la suite du papier, affecte celle du logiciel obtenu. Ce document est souvent illisible et difficile à utiliser par les parties prenantes tout au long du processus. Des études conduites par le Standish Group montrent qu'une mauvaise qualité des exigences entraîne des difficultés dans le développement et mène à des échecs et à des coûts importants en temps et en argent. Un soin accordé au *CdC* aide à réduire l'écart entre ce document et sa spécification formelle. Il améliore la qualité du logiciel final.

Notre approche commence par une étape de re-structuration des besoins. Celle-ci a pour objectif l'obtention d'un document lisible. Nous travaillons sur le *CdC* du client et le ré-écrivons sous forme de phrases courtes étiquetées en utilisant les recommandations d'Abrial [9] et l'outil ProR [4]. Différentes sortes de diagrammes sont disponibles pour aider à la compréhension des besoins [7].

L'activité de vérification a pour objectif d'assurer la correction du modèle formel développé. La validation sert à montrer que le modèle satisfait les besoins du client. Dans notre approche, ces activités aident à améliorer la qualité du *CdC* et sa spécification formelle en Event-B [1]. Nous prenons en compte la validation en tant que processus rigoureux préparé dès le traitement des besoins et tout au long du développement de la spécification formelle [8]. Nous introduisons des paramètres au *CdC* structuré avec ProR ; ces paramètres sont automatiquement mis à jour par la plateforme Rodin [2].

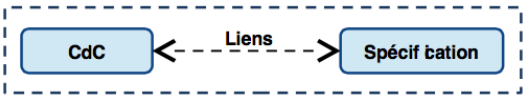
La section 2 décrit rapidement notre approche et les outils utilisés. La section 3 présente quelques leçons retirées de plusieurs études de cas [1]. Nous les illustrons par un système de l'hémodialyse [6]. La section 4 conclut et décrit la suite de ce travail.

2 Notre approche

Un système informatique est composé de :
- son *CdC*, décrit à l'aide de l'outil ProR,
- sa spécification formelle, décrite en Event-B et
- les liens entre ce *CdC* et sa spécification formelle.

L'évolution de l'ensemble est mise à jour en permanence. Elle s'affine grâce aux liens entre le *CdC* et sa spécification ; elle est aidée par les différents outils disponibles.

1. `http://dedale.loria.fr`



2.1 Le *CdC*

Le cahier des charges, informel au démarrage du développement, est ré-écrit sous le forme de phrases courtes et étiquetées. Il est défini par une suite de phrases ; une phrase peut contenir des termes formels intégrés dans le texte informel. Ces termes proviennent de la spécification formelle associée au *CdC*. Ses opérations sont celles des types utilisés, suites, ensembles et produit cartésien noté CP.

CdC is Sequence(*Sentence*)

Sentence is CP [ID: *TEXT*,
[Order: *INTEGER*,]
Description : *Set*(*Term*),
[*Sequence*(*Sentence*),]
[*Set*(*Term_ Type*)]]

Term is Informal_ Term | *Formal_ Term*

Term_ Type is Fact | *Functionality* | *Behavior* | *Obligation*

2.2 La spécification

La correspondance entre le *CdC* et sa spécification Event-B est définie à l'aide de :

- *ID*. Il s'agit d'un commentaire.
- *Order*. L'ordre entre les événements dans la spécification est décrit par les gardes des événements.
- *Informal_ Term*. Il s'agit d'un commentaire.
- *Formal_ Term*. Un terme introduit dans le *CdC* s'exprime par une clause Event-B.
- *Term_ Type*. Le type d'un terme est :
  - *Fact* pour des constantes, des ensembles et des variables,
  - *Functionality* pour des événements,
  - *Behavior* pour l'animation et la simulation de machines,
  - *Obligation* pour des axiomes, invariants et gardes.

2.3 Liens entre le *CdC* et sa spécification

Un terme formel introduit dans la spécification est introduit dans le *CdC*. Il s'agit de définir :

- Le lien entre ce terme formel et sa définition informelle dans le *CdC* ; il s'agit du glossaire.
- Le lien entre ce terme formel et sa définition en Event-B.

Ces liens sont automatiquement mis à jour et disponibles tout au long du développement. Initialement, nous n'avons pas de spécification formelle et il n'y a pas de terme formel dans le *CdC*. Au cours du développement, des termes formels sont intégrés dans le *CdC*. Les informations pour la validation sont extraites de chaque phrase, en tenant compte du type de chaque terme de cette phrase dans le modèle formel associé.

2.4 Outils de vérification et validation

La vérification. Après développement de la spécification formelle, nous prouvons sa correction via des preuves au sens mathématiques. Les obligations de preuve, OPs, sont générées automatiquement ou semi-automatiquement par les outils de preuve sous Rodin.

La validation. Cette activité est utilisée tout au long du processus de développement avec l'outil ProB [5] sous Rodin. Des informations nécessaires à la validation de la future spécification sont extraites de chaque besoin. Elles sont mises à jour par des termes formels au fur et à mesure du développement.

3 Leçons retirées des différentes études de cas

L'utilisation des outils aident à améliorer le processus de développement, que ce soit avec ProR ou avec les outils de vérification et de validation. Le *CdC* et ses liens sont automatiquement mis à jour avec l'évolution de la spécification. Notre propos est illustré par le développement d'un système de l'hémodialyse [6], système critique, hybride et interactif contrôlant sa machine. Nous abordons les points suivants :

– rôle des alarmes,
– forme particulière des besoins ; ils présentent des anomalies,
– rôle de l'invariant de collage de la spécification relativement au triplet <CdC, Liens, Spécification>.

3.1 Compréhension et analyse des besoins

La prise en compte des propriétés de sécurité, de vivacité et de terminaison sont examinées lors de la compréhension du *CdC*. Certaines ambiguïtés et imprécisions sont détectées très tôt dans le développement.

Exemple.

- Les propriétés de sécurité sont clairement exprimées dans le *CdC* de l'hémodialyse.
- La notion d'alarme dans ce système est à clarifier :
  - Existe-t'il une alarme commune à toutes ses composantes ?
  - Existe-t'il une alarme spécifique à chaque composante ? Peut-on alors déceler plusieurs alarmes différentes à un instant donné ?

3.2 Evolution du développement à l'aide d'un patron

La présentation des besoins de la machine de l'hémodialyse utilise une forme particulière. Elle nous a conduit à définir plusieurs patrons [3] ou modèle générique pour le triplet <CdC, Liens, Spécification>.

Exemple. Regardons le besoin *R-8* :

> *R-8 During initiation, if the software detects that the pressure at the AP Arterial Pressure transducer falls below the lower pressure limit, then the software shall stop the BP Blood Pumping and execute an alarm signal.*

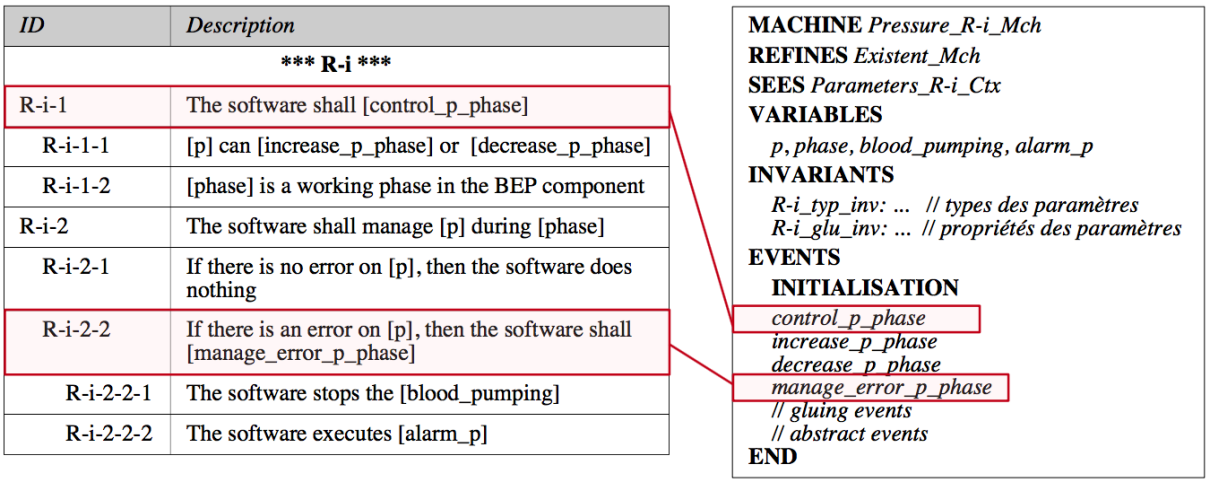
Ce besoin a la même forme que la plupart des besoins de ce cahier des charges. Nous le décrivons de la manière générale suivante :

> *R-i [While/During] an actual phase, if the software detects an error on p [for more/less than n seconds], then the software shall stop the BP Blood Pumping and execute an alarm signal.*

avec ses paramètres instanciés :

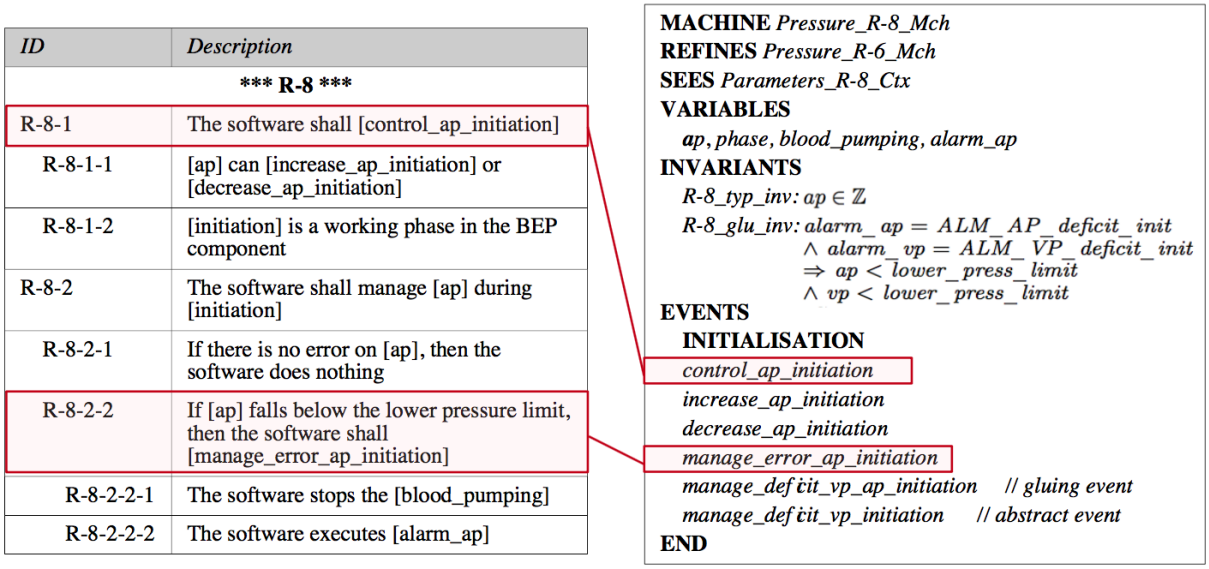
- *i* désigne le numéro du besoin ; il s'agit du numéro *8* dans le besoin *R-8* ;
- *actual phase* désigne les trois phases possibles de thérapie : *initiation*, *connecting the patient* et *reinfusion*. Dans le besoin *R-8*, il s'agit de l'*initiation*, terme informel dans le *CdC*, et *phase*, terme formel dans la spécification. C'est une variable qui a comme valeur *initiation*.
- *p* désigne le texte *the pressure at the AP transducer*. Dans le besoin *R-8*, il s'agit du terme formel *ap* dans le *CdC* et dans la spécification ;
- *an error on p* correspond au texte *p falls below the lower pressure limit*. Dans le besoin *R-8*, *error on p* est traduit par le texte *ap falls below the lower pressure limit* dans le *CdC*. Pour la spécification, il est explicité par la garde de l'événement *manage_error_ap_initiation* :

  grd2 : ap < lower_press_limit
- *n* secondes. Ce paramètre n'est pas utilisé dans le besoin *R-8*.



ID	*Description*
	***** R-i *****
R-i-1	The software shall [control_p_phase]
R-i-1-1	[p] can [increase_p_phase] or [decrease_p_phase]
R-i-1-2	[phase] is a working phase in the BEP component
R-i-2	The software shall manage [p] during [phase]
R-i-2-1	If there is no error on [p], then the software does nothing
R-i-2-2	If there is an error on [p], then the software shall [manage_error_p_phase]
R-i-2-2-1	The software stops the [blood_pumping]
R-i-2-2-2	The software executes [alarm_p]

FIGURE 1 – Modèle générique



ID	*Description*
	***** R-8 *****
R-8-1	The software shall [control_ap_initiation]
R-8-1-1	[ap] can [increase_ap_initiation] or [decrease_ap_initiation]
R-8-1-2	[initiation] is a working phase in the BEP component
R-8-2	The software shall manage [ap] during [initiation]
R-8-2-1	If there is no error on [ap], then the software does nothing
R-8-2-2	If [ap] falls below the lower pressure limit, then the software shall [manage_error_ap_initiation]
R-8-2-2-1	The software stops the [blood_pumping]
R-8-2-2-2	The software executes [alarm_ap]

FIGURE 2 – Instanciation pour le besoin *R-8*

Le modèle générique du triplet <CdC, Liens, Spécification> est décrit dans la figure 1.

CdC. Il est présenté dans la partie gauche de la figure 2.

Spec. Elle raffine la MACHINE *Pressure_R-6_Mch*, voir la partie droite de la figure 2, avec :
- *Un invariant de collage.* Il décrit les prédicats de liaison entre les nouvelles variables et les variables du composant raffiné. Le prédicat *R-8_glu_inv* est présenté dans la figure 2.
- *Les événements abstraits*, comme *manage_deficit_vp_initiation.*
- *Les événements liés au collage* des besoins *R-6* et *R-8*, comme *manage_deficit_vp_ap_initiation.*

Liens. Au cours du développement du besoin *R-8* et de son modèle Event-B, les liens sont automatiquement mis à jour via ProR :
- chaque nouveau terme formel introduit dans le besoin *R-8* est présenté dans la table 1 et
- chaque nouveau terme formel défini dans sa spécification est présenté dans la figure 2.

Les informations nécessaires à la validation pour chaque nouveau terme formel sont présentées dans la

table1.

Formal_ Term	*Informal_ Term*
control_ap_initiation	control the pressure at the AP transducer during initiation
ap	the pressure at the AP transducer
increase_ap_initiation	increase the pressure at the AP transducer during initiation
decrease_ap_initiation	decrease the pressure at the AP transducer during initiation
initiation	initiation
manage_error_ap_initiation	manage error on the pressure at the AP transducer during initiation
blood_pumping	the blood pumping
alarm_ap	an alarm on the pressure at the AP transducer

TABLE 1 – Glossaire pour *R-8*

ID	*Term_ Type*	*Sentence*	*Event-B Model*
R-8-1	Functionality	control_ap_initiation	Pressure_R-8_Mch
...	...	...	...
R-8-2-2	Fact	ap	Pressure_R-8_Mch
	Functionality	manage_error_ap_initiation	
	Behavior	ap falls below the lower pressure limit → manage_error_ap_initiation	

TABLE 2 – Prise en compte de la validation pour le besoin *R-8*

3.3 Amélioration du *CdC* à partir de la spécification

Le développement de la spécification formelle permet de détecter des lacunes dans le *CdC* avec les outils de vérification et de validation. Ces lacunes concernent des oublis, des imprécisions et des incohérences. Elles n'ont pas été détectées dans l'étape de compréhension et analyse des besoins.

Exemple. Regardons le besoin *R-9*.

R-9 While connecting the patient, if the software detects that the pressure at the VP transducer exceeds + 450 mmHg for more than 3 seconds, then the software shall stop the BP and execute an alarm signal.

ID	*Description*
init1	The [blood_pumping] is [BPStopped]
init2	There is no pressure at the [ap]
init3	There is no pressure at the [vp]
init4	All the alarms are disabled

MACHINE *Pressure_R−9_Mch*
REFINES *Pressure_R−8_Mch*
VARIABLES
ap, *phase*, *alarm_ap*, *blood_pumping*, *vp*, *alarm_vp*
EVENTS
INITIALISATION
init_act1: *blood_pumping* := *BPStopped*
init_act2:...
...
END

FIGURE 3 – Etat initial

Son modèle Event-B est défini par un raffinement de la *MACHINE Pressure_R-8_Mch*. Nous avons détecté les anomalies suivantes :

- *Invariant de collage.* Dans le document initial, chaque besoin est isolé. Il est décrit séparément des autres besoins et ne tient pas compte de possibles interactions entre eux.

- *Etat initial.* L'activité de validation nécessite un état initial. Cet état n'a pas été mentionné explicitement dans le *CdC*. Nous avons proposé l'ajout de phrases décrivant cet état, voir figure 3.
- *Omission.* L'alarme peut être commune à toutes les composantes du système ou bien définie pour chaque composant. Les preuves aident à détecter les cas prévus dans la spécification. Par exemple, le choix d'une alarme générale signifie qu'il n'y a pas de changement de valeur de sa variable abstraite.

4 Conclusion et perspectives

Dans ce papier, nous avons abordé la gestion des besoins, ceux-ci sont décrits dans une forme identique [6]. Le document décrit les cas anormaux et présente la gestion des alarmes. L'utilisation et la ré-utilisation d'une spécification existante aborde le rôle de l'invariant de collage dans la spécification.

Les outils disponibles dans la plateforme Rodin ont un rôle important tout au long du processus de développement. Nous utilisons l'outil ProR pour la ré-écriture des besoins, pour leur hiérarchisation et pour la mise à jour du *CdC* et des liens avec sa spécification. Nous utilisons les outils de vérification et de validation à savoir les générateurs d'obligations de preuve, les prouveurs et l'animateur/model-checker ProB pour assurer la correction de la spécification.

Pour la suite de notre travail, il est important de décrire rigoureusement les paramètres intervenant dans la prise en compte d'un besoin par rapport à un système existant. Ces paramètres servent à définir des patrons afin de réutiliser des modèles existants et corrects. L'étude de l'apport de l'étape d'analyse sert à la détection de lacunes dans le système existant.

Dans notre approche, nous étudions des systèmes hybrides. Il serait important de prendre en compte les contraintes de l'environnement et de ses hypothèses relativement aux nouveaux besoins.

Références

[1] Rodin platform, http ://wiki.event-b.org.

[2] J-R. Abrial, M. J. Butler, S. Hallerstede, T. Son Hoang, F. Mehta, and L. Voisin. Rodin : an open toolset for modelling and reasoning in Event-B. *STTT*, 12(6) :447–466, 2010.

[3] T. S. Hoang, A. Fürst, and J.-R. Abrial. Event-B Patterns and their Tool Support. *Software and System Modeling*, 12(2) :229–244, 2013.

[4] M. Jastram. ProR, an Open Source Platform for Requirements Engineering based RIF. *SEISCONF*, 2010.

[5] M. Leuschel and M. J. Butler. ProB : A Model Checker for B. In K. Araki, S. Gnesi, and D. Mandrioli, editors, *International Symposium of Formal Methods Europe, Pisa, Italy, Proceedings*, volume 2805 of *LNCS*, pages 855–874. Springer, 2003.

[6] A. Mashkoor. The Hemodialysis Machine Case Study. In *Abstract State Machines, Alloy, B, TLA, VDM, and Z - 5th International Conference, ABZ 2016, Linz, Austria, Proceedings*, pages 329–343. Springer International Publishing, 2016.

[7] C. Ponsard, R. Darimont, and A. Michot. Combining Models, Diagrams and Tables for Efficient Requirements Engineering : Lessons Learned from the Industry. In *Actes du XXXIIIème Congrès INFORSID, Biarritz, France, May 26-29*, pages 235–250, 2015.

[8] Imen Sayar and Jeanine Souquières. La Validation dans le Processus de Développement. In *Actes du XXXIVème Congrès INFORSID, Grenoble, France, May 31 - June 3*, pages 67–82, 2016.

[9] W. Su, J.-R. Abrial, R. Huang, and H. Zhu. From Requirements to Development : Methodology and Example. In *13th International Conference on Formal Engineering Methods, Durham, UK*, pages 437–455, 2011.