

A First Attempt to Express KAOS Refinement Patterns with Event B

Abderrahman Matoussi, Frédéric Gervais, and Régine Laleau

LACL, Université Paris-Est

{abderrahman.matoussi, frederic.gervais, laleau}@univ-paris12.fr

1 Motivation

It is now recognised that goals play an important role in requirements engineering process, and consequently in systems development process. Whereas specifications allow us to answer the question "WHAT the system does", goals allow us to address the "WHY, WHO, WHEN" questions [1]. Up to now, the development process associated with formal methods, including Event B, begins at the specification level. Our objective is to include requirements analysis within this process, and more precisely the KAOS method. Existing work [5, 4] that combine KAOS with formal methods generate a formal specification model from a KAOS requirements model. We aim at expressing KAOS goal models with a formal language (Event B), hence staying at the same abstraction level. Thus we take advantage from the Event B method: (i) it is possible to use the method during the whole development process and (ii) we can benefit from the industrial maturity of tools supporting the method. This paper presents, through an example, the outlines of a constructive approach in which Event B models are built incrementally from KAOS goal models, driven by goal refinement patterns.

2 Goals in KAOS

KAOS (Knowledge Acquisition in autOMated Specification) [2, 3] is a methodology to implement goal-based reasoning. A goal defines an objective the system should meet, usually through the cooperation of multiple agents such as devices or humans. KAOS is composed of several sub-models related through inter-model consistency rules: (i) the central model is the *goal model* which describes the goals of the system and its environment; (ii) the *object model* defines the objects (agents ,entity...) of interest in the application domain; (iii) the *agent responsibility model* takes care of assigning goals to agents in a realisable way; (iv) the *operation model* details the operation an agent has to perform to reach the goals he is responsible for. KAOS offers a lot of refinement patterns [1] that decompose goals. These patterns can only be used in the context of different tactics defined in KAOS such as *milestone-driven tactics*, i.e. identifying milestone states that must be reached to achieve the target predicate, and *case-driven tactics*, i.e. identifying different cases to satisfy the goal. The sub-goals $G_1, G_2, \dots, G_n (n \geq 2)$ refine a goal G iff the following conditions hold:

1. $G_1 \wedge G_2 \wedge \dots \wedge G_n \models G$ (entailment)
2. For each $i, j: j \neq i. G_j \not\models G_i$ (minimality)
3. $G_1 \wedge G_2 \wedge \dots \wedge G_n \not\models false$ (consistency)

In this work in progress, we focus on refinement patterns defined only with first-order logic. Patterns with LTL temporal logic will be studied in further work. Thus, the general form of the assertions associated to the patterns is $P \rightarrow Q$ where P and Q are predicates. Symbol \rightarrow denotes the classical logical implication.

Let us take a simple example managing the subscription to a summer school. The main goal G states that each person who has subscribed must receive a participation receipt: $Subs \rightarrow PRcpt$. This goal is refined into three sub-goals according to the milestone-driven tactics:

- (G_1) each subscription implies a payment: $Subs \rightarrow Payt$
- (G_2) for each payment a bill should be issued: $Payt \rightarrow Bill$
- (G_3) a receipt must be submitted whenever the bill is provided: $Bill \rightarrow PRcpt$

The case-driven tactics is applied to refine the goal G_1 into two sub-goals depending on the participant status (either student or professor):

- ($G_{1.1}$) $Subs \wedge Stud \rightarrow StudFee$
- ($G_{1.2}$) $Subs \wedge Prof \rightarrow ProfFee$

3 Expressing Goals in Event B

The objective of our work is to express a KAOS goal model with Event B. We start to study the most used refinement patterns: the milestone-driven and case-driven tactics.

Since a KAOS goal means that a property must be established, the main idea is to represent each goal as a B event and the property as the post-condition of this B event.

Thus, goal G can be translated into an abstract B event as follows:

$$\mathbf{EvG} \triangleq \text{SELECT } True \text{ THEN } Subs_B, PRcpt_B : (Subs_B \subseteq PRcpt_B) \text{ END}$$

The THEN part of \mathbf{EvG} is the translation into Event B of the logical formula associated to G . Even if this translation is obviously always possible, it is not straightforward and necessarily depends on the B representation of the KAOS object model.

At this most abstract level, the guard of \mathbf{EvG} is always set to *True* to express that the event is always feasible. The definitive guard is built during the refinement process.

First refinement: applying the milestone-driven tactics. All sub-goals are translated into new events using the same rules as for \mathbf{EvG} . For instance, (G_1) is translated by:

$$\mathbf{EvG}_1 \triangleq \text{SELECT } True \text{ THEN } Subs_B, Payt_B : (Subs_B \subseteq Payt_B) \text{ END}$$

The abstract event **EvG** is refined by strengthening its guard. This latter is the conjunction of the post-conditions of each sub-goal:

```

EvG  $\triangleq$ 
  SELECT ( SubsB  $\subseteq$  PaytB)  $\wedge$  (PaytB  $\subseteq$  BillB)  $\wedge$  (BillB  $\subseteq$  PRcptB)
  THEN SubsB, PRcptB : (SubsB  $\subseteq$  PRcptB) END

```

Second refinement: applying the case-driven tactics. In the same way as for the milestone-driven tactics, the sub-goals are translated by new events and **EvG₁** is refined as carried out for **EvG**.

```

EvG1.1  $\triangleq$ 
  SELECT True
  THEN SubsB, StudB, StudFeeB : ((SubsB  $\cap$  StudB)  $\subseteq$  StudFeeB) END

```

However, a faithful representation of this tactics requires the following additional constraints:

- A new invariant: $(Stud_B \cup Prof_B) = Subs_B \wedge (Stud_B \cap Prof_B) = \emptyset$
- A new proof obligation: $(StudFee_B \cup ProfFee_B) \subseteq Payt_B$

Verification of the KAOS refinement conditions. Proof obligations of Event B allow most of the KAOS refinement conditions to be verified. However, for some KAOS patterns as the case-driven tactics, additional constraints must be identified.

4 Further work

The paper shows that it is possible to express KAOS goal models with Event B and furthermore, this B representation is quite close to the KAOS one. The current work is still partial and we are actively working on its extensions. Future work will be concerned with (i) generalising our method to other refinement patterns presented in [1]; (ii) considering LTL temporal operators; (iii) considering the three other sub-models of a KAOS model.

References

1. R. Darimont and A. van Lamsweerde. Formal Refinement Patterns for Goal-Driven Requirements Elaboration. In *SIGSOFT '96*, pages 179–190, San Francisco, California, USA, October 1996. ACM SIGSOFT.
2. A. van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In *RE 2001*, pp. 249–263, Toronto, Canada, August 2001. IEEE Computer Society.
3. E. Letier. Reasoning About Agents in Goal-Oriented Requirements Engineering. Ph.D. Thesis, <ftp://ftp.info.ucl.ac.be/pub/thesis/letier.pdf>, 2001.
4. H. Nakagawa and K. Taguchi and S. Honiden. Formal specification generator for KAOS. In *ASE 2007*, pages 531–532, Atlanta, Georgia, USA, November 2007. ACM.
5. C. Ponsard and E. Dieul. From Requirements Models to Formal Specifications in B. In *REMO2V'2006*, Luxembourg, June 2006.