From Requirements Models to Formal Specifications in B

Christophe Ponsard and Emmanuel Dieul

[1] CETIC Research Center, Charleroi (Belgium) - cp@cetic.be
[2] Siemens Transportation Systems (France) - emmanuel.dieul@ts.siemens.fr

Abstract. The development of critical systems requires a high assurance process from requirements to the running code. Formal methods, such as B, now provide industry-strength tools to develop abstract models, refine them in more concrete models and finally turn them into code. A major remaining weakness in the development chain is the gap between textual or semi-formal requirements and formal models. In this paper, we explore how to cope with this problem using a goal-oriented approach to elaborate a pertinent model, including regulation modelling, and turn it into a high quality abstract formal specification.

1 Introduction

Several fields such as transportation, health care, finance are increasingly relying on complex systems mixing hardware and software, interacting tightly with human users and constrained by a number of national/international regulations and standards. Achieving high assurance is difficult but critical as failure can lead to catastrophic consequences ranging from loss of profit to human lives.

Formal methods (FM) are based on rigorous mathematical reasoning. They have shown their ability to produce such systems for large industrial problems (such as Paris metro line 14 using B). The deployment of such methods are far from trivial for a number of reasons such as the mathematical expertise required, the poor communicability with the customer, the lack of indirect "return on investment" of increased time spend earlier in the project. As the formal development chain matures, the major problem is now at the "gap" between the requirements documents and the initial formal specification [2]. In current practice, the formal specification is elaborated based on informal or semi-formal requirements, the validation of the specification is then difficult due to the inability for the customer to understand the formal model, to link them with initial requirements and to bring regulations into the picture.

Some practical solutions have been explored such as the extraction of UML views [4] or the inclusion of traceability links [3]. Although interesting, those are going "backward" to the requirements level. In this paper, we develop an approach where formal methods can be introduced during requirements engineering (RE) to build a model combining semi-formal and formal notations from which a specification can then be derived in a more constructive, "forward way" [9].

In the present context of regulation modelling, this design process is interesting because it can be guided to comply with given regulations and can produce a rich set of traceable properties (called goals hereafter) the system has to ensure within its environment. Those can be explicit regulation constraints on the system under design, organisational constraints on agents in the environment, technical requirements in a specific design, etc.

The rest of this paper is structured as follows. In section 2, the KAOS goal-oriented framework will be used to show how system goals can be captured, refined and analyzed formally while preserving communicability and taking regulations into account. The B method [1] which is which clearly evolving towards system level [2], will be used as formal specification language. Section 3 will show how a set of initial B machines can be derived from the requirements model. Section 4 will summarize benefits, current limits and future work.

To illustrate our purpose, we choose the train domain which is very rich in regulations (e.g. new office of rail regulation in UK, ministerial service of rail regulation in Belgium) and standards (e.g. Cenelec 50126/128/129). The excerpt used as running example is related to the operation of platform screen doors (borrowed from [5]).

1.1 Requirements Modelling and Analysis

A KAOS requirements model is composed of four sub-models: (i) the central model is the *goal model* which captures and structures the assumed and required properties (including regulations); (ii) the *object model* captures the relevant vocabulary to express the goals; (iii) the *agent model* takes care of assigning goal to agent in a realizable way; (iv) the *operation model* details, at state transitions level, the work an agent has to perform to reach the goals he is responsible for.

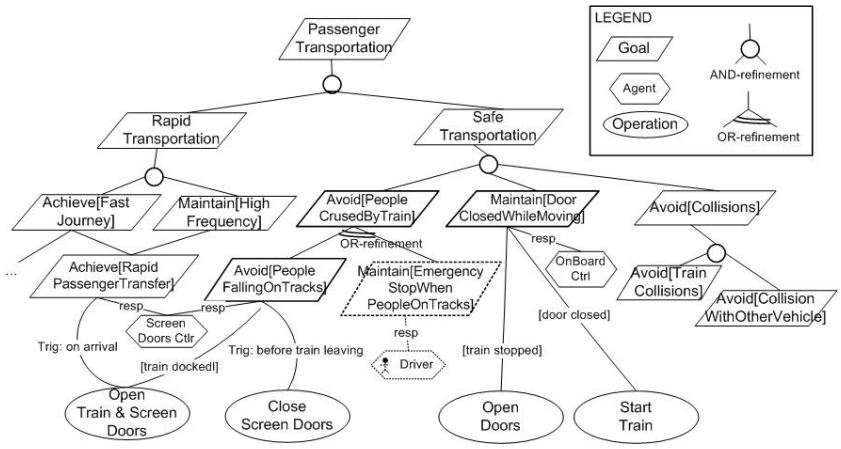

Fig. 1. Goals related to Platform Screen Doors

Model elaboration usually starts from key properties of the system to-be. Those are expressed using *goals* which are statements of intent about some sys-

tem (existing or to-be) whose satisfaction in general requires the cooperation of some of the agents forming that system. Figure 1 shows the goals related to the operation of the platform screen doors. High level goals are related to progress properties and passenger safety. Safety issues are refined (*AND-refinement*) considering the avoidance of dangers such as people falling off the train or on the tracks. For the latter, two alternatives are considered (*OR-refinement*): in a manual system, the driver can be responsible for this by monitoring/anticipating such events and triggering an emergency stop when needed. In the automated design considered here, this is not possible: the selected alternative is to achieve track isolation and access through screen doors. The design choice can be made with respect to the satisfaction of higher level (generally non functional) goals but can also be prescribed by a regulation authority (e.g. decision to generalise screen doors in the French metro).

Those goals are described informally in natural language and are optionally formalized in a real-time temporal logic [9]. Keywords such as *Achieve*, *Avoid*, $Maintain$ are used to name goals according to the temporal behaviour pattern they prescribe. For example the goal $Maintain[DoorsClosedWhileMoving]$ can be formalized as follows:

Goal Avoid[PeopleFallingOnTracks]
$(\forall pf : Platform)\ (\neg(\exists tr : Train)\ docked(tr, pf)) \Rightarrow pf.doors = CLOSED$

In the formalization process, pertinent entities/relationships/attributes (eg. Train, docked, doors,...) are identified and result in the constructive elaboration of an object model. Due to a lack of space, this model is not illustrated graphically here.

To enforce the system behaviours, goals are under the responsibility of a number of *agents* which are active components, such as humans, devices, legacy software or software-to-be components, that play some role towards goal satisfaction. Some agents are part of the system to design whereas others define its environment. Some may also be prescribed by regulation authorities. Those agents control the operation through a number of operations which are strengthened in order to satisfy the goals. For example, the $Achieve[FastPassengerTransfer]$ goal results in a trigger on the door opening, while the $Avoid[PeoplefallingOn Tracks]$ goal results in an additional precondition to open the doors. This can be formalized as follows (where $@P \Leftrightarrow \bullet\neg P \wedge P$, $\bullet$ being the previous state)).

Operation OpenPlatformDoors
Input $pf : Platform$
Output $pf : Platform/doors$
DomPre $pf.doors = CLOSED$
DomPost $pf.doors = OPEN$
ReqTrig for $FastPassengerTransfer : @\exists(tr : Train)\ docked(tr, pf)$
ReqPre for $PeopleFallingOnTracks : \exists(tr : Train)\ docked(tr, pf)$

2 Deriving Formal Specifications

In order to derive initial B machines from the requirements model, we propose here a basic algorithm focusing on safety properties.

As agents are the active entities able to perform operations, a B MACHINE is associated with each KAOS agent. Of course, this initial design can be refined later in B to develop or map to a finer grained system, reuse existing components, etc. Each machine is composed of the following elements:

- necessary SETS, VARIABLES, CONSTRAINT to model the agent attributes and the operations arguments
- INITIALISATION information, also present in goal refinements
- INVARIANTS for all maintain goals under the agent responsibility
- OPERATIONS the agent has to perform. Domain and strengthened pre- and postconditions are merged. Trigger conditions are not considered but can be safely ignored as they should imply the precondition.

The resulting machine for the $PlatformController$ is the following:

MACHINE PlatformController
SETS $PLATFORM; DOORSTATE = \{open, closed\}$
VARIABLES $doors, docked$
INVARIANT
$doors \in PLATFORM \rightarrow DOORSTATE \ \wedge \ docked \in TRAIN \times PLATFORM \ \wedge$
$\forall pf \cdot (pf \in PLATFORM \rightarrow \neg \exists tr \cdot (tr \in TRAIN \wedge (tr, pf) \in docked)$
$\rightarrow doors(pf) = closed)$
INITIALISATION $doors := PLATFORM \times \{closed\} \parallel docked = \{\}$
OPERATIONS
openPlatformDoors $(pf) \doteq$
PRE $pf \in PLATFORM \wedge doors(pf) = closed \wedge$
$\exists tr \cdot (tr \in TRAIN \wedge (tr, pf) \in docked)$
THEN $doors := doors \cup \{pf \mapsto open\}$
END
...

3 Benefits, current limitations and future work

So far, our current experiments confirmed a number of benefits:

- *early validation.* The operational requirement model can be animated through the generation of finite state machines [8]. The system behaviour can directly be checked by a domain expert, possibly using domain specific display and controls (note this component can be reused from the application domain, avoiding the cost of development and validation). Relevant goals can also be automatically monitored (see figure 2).
- *better initial models.* Although proof obligations are not discharged (the mapping is still partial and unproved), the effort spent in early modelling and validation results in easier proofs (i.e. managed by the automated prover).
- *communication.* Formal language can always be hidden behind semi-formal notations. Useful diagrams such as the object model or the agent interactions are easier to generate at this step than from B models.
- *better traceability.* The goal model provides a rich set of properties which can be traced throughout the project life. A look at the above untagged B machine shows how difficult it is to discover the related requirements.
- *natural complementarities.* Both KAOS and B have a notion of refinement and are based on a constructive approach. KAOS at the property level and B at the machine level.

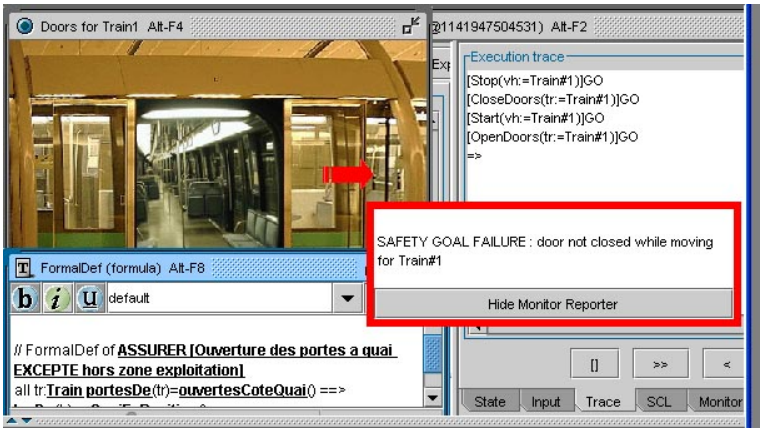

Fig. 2. Requirements Animation for Platform Screen Doors

The current mapping is still partial and we are actively working on its improvement. Future work will move to Event B [5] which is more appropriate to reason at system level and for liveness properties. This work is done in connection with RODIN project [7]. At tool level, we plan to develop a B connector between the Objectiver/FAUST[6] toolset and the RODIN open platform.

Regarding regulations, method support for regulation modelling should be further elaborated. Interesting issues to cover is the verification of regulation enforcement and the analysis of possibly conflicting regulations in order to merge them (from national regulation to European regulations such as in the Euro-Interlocking group in the railway domain).

Acknowledgement

This work is financially supported by the European Union (ERDF and ESF) and the Walloon Region (DGTRE).

References

1. J. R. Abrial, *The B-Book: Assigning programs to meanings*, Cambridge University Press, 1996.
2. J.R. Abrial, *B: past, present, future (in French)*, 2002.
3. Jeremy Dick, *Formalising the informal: Linking formal methods to informal requirements*, Proc. FMICS'04, Linz (Austria), 2004.
4. Akram Idani and Yves Ledru, *Object Oriented Concepts Identification from Formal B Specifications*, Proc. FMICS'04, 2004.
5. IST-1999-11435, *Matisse: Methodologies and technologies for industrial strength systems engineering - practitioners handbook*, 2003.
6. Objectiver/FAUST, *http://www.objectiver.com + http://faust.cetic.be*, 2004.
7. The RODIN project, *http://rodin-b-sharp.sourceforge.net/*, 2005.
8. H. Tran Van, A. van Lamsweerde, P. Massonet, and C. Ponsard, *Goal-oriented requirements animation*, 12th IEEE Int.Req.Eng.Conf., Kyoto, September 2004.
9. A. van Lamsweerde, *Goal-oriented requirements engineering: A guided tour*, Proc. RE'01 - 5th IEEE International Symposium on Requirements Engineering, 2001.