Formal Domain Modeling: From Specification to Validation

Atif Mashkoor

Formal Domain Modeling: From Specification to Validation *

Atif Mashkoor

LORIA – DEDALE Team – Nancy Université
Campus Scientifique, BP 239,
F-54506, Vandœuvre lès Nancy, France
{firstname.lastname}@loria.fr

Abstract. The main theme of this research is to study and develop techniques for the modeling of software controlled safety critical systems. In this work, we formally specify different entities, phenomena and their inter-relationships, and specially non functional properties related to land transportation systems with refinement based approach at domain level. We also introduce a stepwise validation process to maintain seamlessness between environment and its captured models. We apply our results on two safety critical case studies.

1 Introduction

Having good understanding of an application domain is a crucial prerequisite to develop software within that domain. The understanding of a domain is referred to as a domain model. A domain model is a conceptual model of a system which describes various entities, phenomena and their inter-relationships, along with their important static and dynamic properties of the domain. The domain model may be expressed in the form of requirements, specifications, or architectural references.

According to [1], if domain models and requirements of software are not formally expressed, software correctness can not be meaningfully achieved. Safety is also one of the major factors which can not be overlooked while designing complex and critical systems. The development of correct and safe systems can be difficult and error prone with traditional software development methods. However, use of formal methods, in order to ensure their correctness and to structure their development from domain modeling to implementation, can significantly help system development.

Formal languages are notoriously difficult to read for the non-initiated. Furthermore, well-written specifications often introduce abstract objects and operations that have no intuitive concrete counterpart. Hence, validation has to wait. This implies that the development of the model requires an uncomfortable level of trust.

The pivotal concept of formal methods such as B [2] is the notion of refinement and its relation to correctness. The assessment of the correctness of a piece of code,

* This work has been partially supported by the ANR (National Research Agency) in the context of the TACOS project, whose reference number is ANR-06-SETI-017 (http://tacos.loria.fr), and by the Pôle de Compétitivité Alsace/Franche-Comté in the context of the CRISTAL project (http://www.projet-cristal.org).

its verification, is no more a unique big process step but it is broken down into small pieces along with the whole development process. The proof of correctness is the sum of the proofs of small assertions (invariant preservation, well-formedness, existence of abstraction function, etc.) associated to each refinement. Likewise, the technique of animation could be used with each refinement step to break its validation into smaller assessments.

Introduction of validation into refinement based processes yields two advantages: First, early detection of problems in the models (say, misunderstanding about a certain behavior) should be easier and inexpensive to correct. Second, users can be involved into the development right from the start.

In this work, we focus on animation technique which is the "execution" of a specification as a mean to validate it. Thanks to the development of tools, like Brama [3], it is possible to animate specifications in B or Event-B [4] before they reach an implementation stage. However, there are restrictions on the kind of specifications that can be animated. Non-constructive definitions, infinite sets, or complex quantified logic expressions are among the list of restrictions.

We devise a technique to animate abstract specifications by systematic transformations. The product of the transformations is a specification which may be non provable, but which is guaranteed to have the same behavior as the formally correct initial specification. This goal is achieved through the design of a set of transformation heuristics whose correction is rigourously asserted and a rigorous process.

The main aim of the research is two fold: first, formal modeling of data, behaviors, protocols, interaction between elements, and non-functional properties of transportation domain and second, to incorporate a stepwise validation technique into the overall modeling process.

The presentation of the paper is organized as follows: Next section presents the language and tool we use: Event-B and Brama. Then we present a stepwise specification and validation process for domain modeling. Two case-studies are described thereafter to show how we have implemented our approaches. Finally, we conclude that what should now be completed to have a technique that could be used as standard practice and our future research endeavors.

2 Tools

This section introduces the tools which we use for specification and validation.

2.1 Event-B

Event-B is a formal method for system-level modeling and analysis of large reactive and distributed systems. Main features of Event-B are the use of set theory and first order logic for modeling systems, the use of refinement to represent systems at different levels of abstraction and the use of mathematical proof to verify consistency between refinement levels. Event-B is provided with tool support in the form of an Eclipse-based IDE called RODIN[1] which is a platform for writing and proving Event-B specifications.

[1] http://sourceforge.net/projects/rodin-b-sharp/

2.2 Brama

Brama is an animator for Event-B specifications. It is an Eclipse based plug-in for the RODIN platform which can be used in two complementary modes. Either Brama can be manually controlled from within the RODIN or it can also be connected to a Flash graphical animation through a communication server; it then acts as the engine which controls the graphical effects.

3 Stepwise specification and validation

In order to verify complex systems against their requirements, the breakdown of overall system into different levels of abstraction is highly recommended. This can be achieved by their stepwise development. To specify our domain models we follow the same principle. We start from abstract requirements, we gradually refine them to achieve concrete and fine grained description of the model, and verify each refinement step against the specification constructed in the previous refinement.

We use two different notions of refinement to specify our models: horizontal refinement and vertical refinement. In horizontal refinement the details are added to the model while remaining at the same level of abstraction. However in vertical refinements we add the description to the model while making a leap to the next abstraction level.

Once a domain model has been specified and verified, an important question arises: does it accurately capture the environment? While proof tools guarantee the consistency of the specification, they are of little help to check if it is the true representation of the environment. Like the verification of the model can be broken down into smaller proofs associated with each refinement step, we use the technique of animation at each refinement step to break its validation into smaller assessments.

Animation of specification is not that straightforward because it heavily depends upon tools. Any limitation of the tool will be a restriction on the class of animatable specifications. To validate a specification which does not belong to this class, we need to "bring it in." We do this by applying transformation rules which are designed to keep the behavior unaltered, possibly at the expense of other properties.

Experimenting with Brama on two Event-B safety critical systems–a formal domain model of land transportation [5] and a situated multi-agent platooning system [6]–we have dressed up a typology of five general cases:

1 Brama forbids finite clauses in axioms
2 Brama interprets quantifications as iterations
3 Brama cannot compute dynamic functional bindings in substitutions
4 Brama does not compute analytically defined functions
5 Brama has limited communication with its external graphical environment

This lead us to design 10 transformation heuristics [7] expressed following the pattern shown by figure 1.

We designed the heuristics to preserve the behavior of the specification, *not* its formal properties. In particular, the transformed specification may not be provable. The correctness of the transformation is then a crucial issue.

Since heuristics cannot be "proven" within B formal logic system, we relied on the mathematical tradition of *rigorous arguments*. For this to work, we need a basic assumption: the initial specification text must have been formally verified. Most of the arguments given in the **Justification** clause of heuristic rely on this hypothesis.

Heuristic pattern

Symptom: What reveals the situation e.g. Brama error message
Transform: The expression schema in the original specification and its transformed counterpart
Caution: Description of the applicability conditions, possible effects, and precautions to follow
Justification: A rigorous argument about the validity of the transformation

Fig. 1. The heuristic pattern

A *verified* specification must be the starting point of the validation process. The application of the heuristics will "downgrade" it to a non provable specification. Running the animation may uncover some mistakes. These entail the modification of the *initial* specification, which then must be verified, and transformed again for proceeding with the validation. This is summed-up in figure 2.

4 Case studies

This section describes two case studies which were the incentive for this work. Both specifications concern the domain of land transport systems. They are part of cooperative projects. TACOS is an effort to integrate components and non functional properties into formal requirement specifications. In particular, safety critical properties must be assessed and formalized. CRISTAL is a joint project with the industrial goal of designing urban mobility systems based on autonomous vehicles. As these systems interact with humans and operate on public space, the certification issue is a major problem.

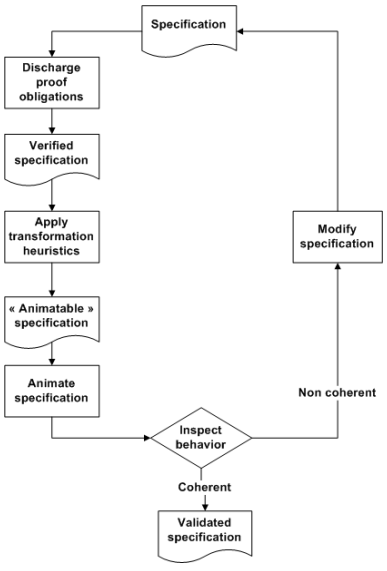

Fig. 2. The validation process

4.1 TACOS

The specification in this case study is about the modeling of the land-transportation domain. In the model, we want to express properties that any system working within the domain is expected to meet and maintain.

In this specification effort, the focus is on the formal definition of concepts, constraints and properties, rather than on the implementation of a particular system. Refinement is then used to introduce new notions; the proof

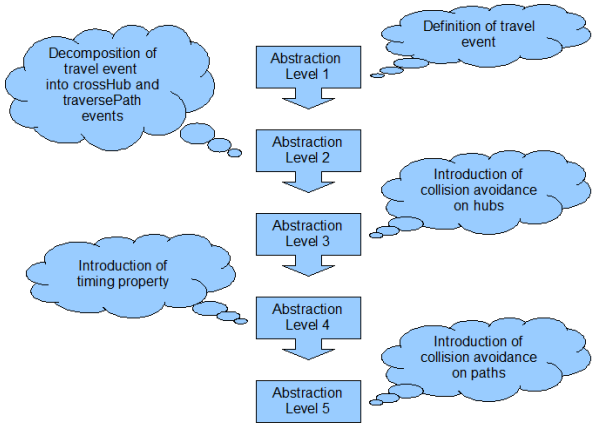

Fig. 3. Levels of abstraction of transport domain model

obligations serve to guarantee the consistency of the model. Our devised stepwise validation technique was used for the validation of the model whose details can be found in [8].

The current specification consists of 8 refinements (3 horizontal and 5 vertical). It is organized into five abstraction levels which are summarized by figure 3. A detailed description of the model can be found in [5].

This specification exhibits several properties which call for validation, namely:

- complex data which constraint behaviors (following a route),
- protocols and iterations (travel as sequence of stages, hub crossing protocol), and
- non deterministic interaction between elements (autonomous vehicles).
- several non-functional properties, such as collision avoidance, timing, etc.

4.2 CRISTAL

The second case study deals with a specification of *platooning*. Platooning is a mode of moving where vehicles are synchronized and follow one another closely. A platoon can be seen as a road-train where cars are linked by software instead of hardware. A local Event-B specification of the model has been written [6] as an effort to make it amenable to the formal techniques required by certification.

The specification consists of five machines (one abstract and four vertical refinements). Contrary to the previous case study, the structure of the development of this case study can be interpreted as a sequence of refinements toward an implementation. Each refinement decomposes some events to make explicit a part of the general computation.

We mainly use this case study to experiment our devised transformation heuristics for its validation. The underlying rationale was to observe the animation of a model based on kinematic laws specified by heavy use of functions and also to compare the results of the validation by animation with the results of validation by simulations that had been previously made.

5 Conclusion and future work

Using a modeling language which is not conceived particularly for domain engineering was a challenging task. Though we stumbled upon some shortcomings in Event-B (for example, lack of temporal constraints, lack of notion of sequences, etc.) yet, the general philosophy has been well suited to our purpose. The notions of events and non determinism allow us easy modeling of independent vehicles without any assumption other than their common property: they move. The strong safety constraints we have considered are also easily modeled. Modeling of other non functional properties, such as, collision avoidance, timing, etc. also did not pose great difficulties. All was done through standard refinement techniques. We are thus encouraged to proceed further with enrichment of current domain models specially with inclusion of more non functional properties, such as, oscillation, hooking/unhooking, etc.

While arguing about the relationship between refinement based modeling and its stepwise validation, we discovered that not every refinement step is animatable. This is consistent with using animation as a kind of quality-assurance activity during development. We believe that one animation per abstraction level is sufficient. In fact, the first refinement of a level may often have a non-determinism too wide to allow for meaningful animation (concept introduction), but subsequent refinements get the definitions of the new concept precise enough to allow animation.

The list of heuristics is not closed yet. In future this is expected to grow as we model and validate new properties of the domain. Manual application of these heuristics to specification is tedious, cumbersome and may be error prone if not applied carefully. Therefore we are planning to write a plug-in/tool which can apply these transformations automatically to specifications.

References

1. Bjørner, D.: Development of transportation systems. In: International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISOLA). (2007)
2. Abrial, J.R.: The B Book. Cambridge University Press (1996)
3. Servat, T.: BRAMA: A New Graphic Animation Tool for B Models. In: B 2007: Formal Specification and Development in B, Springer-Verlag (2006) 274–276
4. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press (2009)
5. Mashkoor, A., Jacquot, J.P., Souquières, J.: B événementiel pour la modélisation du domaine: application au transport. In: Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL'09), Toulouse, France (2009) 1–19
6. Lanoix, A.: Event-B specification of a situated multi-agent system: Study of a platoon of vehicles. In: 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE), IEEE Computer Society (2008) 297–304
7. Mashkoor, A., Jacquot, J.P.: Incorporating Animation in Stepwise Development of Formal Specification. Research Report INRIA-00392996, LORIA, Nancy, France (2009) http://hal.inria.fr/inria-00392996/en/.
8. Mashkoor, A., Jacquot, J.P., Souquières, J.: Transformation Heuristics for Formal Requirements Validation by Animation. In: 2nd International Workshop on the Certification of Safety-Critical Software Controlled Systems - SafeCert'09, York, UK (2009)