On the importance of explicit domain modelling in refinement-based modelling design. Experiments with Event-B*

Y. Aït-Ameur[1], I. Ait-Sadoune[2], P. Casteran[3], P. Gibson[4], K. Hacid[1], S. Kherroubi[5], D. Méry[5], L. Mohand-Oussaid[2], N. K. Singh[1], and L. Voisin[6]

[1] INP-ENSEEIHT/ IRIT, Université de Toulouse, Toulouse, France
[2] CentraleSupelec/LRI/Paris Saclay University, Compus de Paris-Saclay, France
[3] LABRI, Université de Bordeaux, Bordeaux, France
[4] Télécom Sud Paris, France
[5] LORIA,Université de Lorraine & Telecom Nancy, Nancy, France
[6] Systerel, Aix-En-Provence, France
yamine@enseeiht.fr

1 Context

Although several authors like P. Zave and M. Jackson[11, 17], D. Bjorner[5], A. Van Lamsweerde [13] have drawn the attention of system designers on the necessity to handle domain knowledge, while designing systems, it is still a major concern nowadays. The IMPEX[7] project, funded by the French ANR national research agency, addresses the problem of making explicit domain knowledge in formal system developments using refinement and proof based formal methods. It advocates the use and formalisation of ontologies as models for domain knowledge. The Event-B[1] modelling technique has shown its usefulness to support the various developments. In this paper, we briefly describe the approach[3] and the case studies developed in the context of this project.

2 Formal ontologies as domain models

Gruber defines an ontology as *an explicit specification of a conceptualization* [7]. Another definition relies on the notion of dictionary. [12] considers a domain ontology as a *formal and consensual dictionary of categories and properties of entities of a domain and the relationships that hold among them.* Here, an entity represents any concept belonging to the considered domain. *Dictionary* entails two major concepts. First, it makes explicit the existence, through a constructive definition or declaration, of entities in the domain under consideration and second any entity or relationship described in this ontology is directly referenceable independently of other entities or relationships. Reference is carried by a symbol defining an identifier. Each *description* of each entity or relationship

*This work was supported by grant from the French national research agency - ANR ANR-13-INSE-0001 (The IMPEX Project http://impex.gforge.inria.fr (or http://impex.loria.fr).

[7] http://impex.gforge.inria.fr (or http://impex.loria.fr)

is formally *stated* using an *ontology modelling language* equipped with a formal semantics. It allows automatic reasoning and consistency checking. Event-B is a good candidate to support such formal descriptions.

2.1 Ontologies as theories in Event-B[2]

In the context of IMPEX, we have identified two approaches to define ontologies as formal theories. They use two different modelling processes: shallow and deep.

- The **shallow modelling** approach consists in formalising the ontology concepts directly in the target modelling language without keeping trace of the structure of the ontology modelling language concepts [16]. One way to integrate the ontology concepts into a specific formal method development process is to express the ontology modelling languages constructs into the target formal language by means of transformation rules. In our case, a shallow modelling approach consists in encoding the ontology concepts (classes, properties, ...) directly in an Event-B context using abstract sets, constants and axioms.
- The **deep modelling** approach consists in formalising the ontology concepts together with the concepts of the modelling language that were used to define the ontology concepts [9, 8]. Here, ontologies are defined as instances of ontology models. Two steps are required. First, an ontology model is formalised and then ontologies are defined as specific models corresponding to the defined ontology model. In our approach, we consider that both ontology modelling concepts and ontologies are explicitly modelled. These concepts have been formalised in Event-B. More precisely, as we consider ontologies as theories, we have used Event-B contexts to formalise such concepts.

The OntoEventB plug-in. The OntoEventB plug-in[8] [16] has been developed to automatically support the translation of ontologies models, described using ontology description languages such as OWL [4] or PLIB [10], into Event-B contexts. It takes as input an ontology description file and generates, according to the selected approach (shallow or deep), the corresponding Event-B contexts. The OntoEventB plug-in is integrated it into Rodin. To use the OntoEventB plug-in in your Rodin platform instance, you must install the plug-in by using the Install New Software menu item.

2.2 The IMPEX approach[3]

As mentioned above, ontologies have been chosen as a framework for modelling domain knowledge. Additionally, annotation relationships have been set up to link system models concepts to its semantic description unit provided by the ontologies. In this way, it becomes possible to consider properties of the domain in system models. As a consequence, domain knowledge is made explicit in such system models. Domain properties together with reasoning capabilities become accessible from the system models.

[8] OntoEventB update site : `http://wdi.supelec.fr/OntoEventB-update-site/`

2.3 *A priori* or *a posteriori* handling of domain models

When domain models are formalised, it is possible to take into account the expressed domain concepts and properties in system models. Two different situations depending on the availability of the considered system models may occur.

- In the **a priori** case, we consider that domain models are available before the system models are produced. In this case, when designing system models, domain concepts (axioms or theorems) are borrowed to define the system model concepts as being *subsumed* by domain concepts. Note that the subsumption mechanism available in ontology-based models allows a designer to borrow only relevant concepts and properties from an ontology. The two first examples of Section 3 report on an a priori approach.
- The **a posteriori** case occurs when system models are already designed. In this case, these models are *re-factored* using explicit references to ontology concepts. This mechanism uses specific references, based on explicit mapping definitions, to borrow ontology concepts inside the re-factored models. The last example of Section 3 reports on an a posteriori approach.

3 Case studies

In this section, we briefly present some experiments we have conducted using both a priori and a posteriori approaches to explicitly handle domain knowledge.

Embedded systems[14] The embedded system under consideration is a nose gear velocity update function. It is responsible of estimating the velocity of an aircraft while moving on the ground. Hence, it is suitable to highlight the need for identifying and integrating explicit semantics. A single explicit, requirement is defined. `EXFUN-1`: *While the aircraft is on the ground, the estimated velocity shall be within 3 km/hr of the true velocity of the aircraft at some moment within the past 3 seconds.* Along with `EXFUN-1`, we have systematically extracted several other implicit/derived requirements from this requirements description. An a priori model [14] of the Nose Gear Velocity system has been developed with six Event-B refinements. The second refinement introduces the interrupt service routine (ISR) responsible for updating the rotation counter and recording the service request time. As per the system description, *NGRotations* counter is a 16-bit counter. However, it is observed that *NGRotations* counter can be modelled as an *always incrementing* counter – taking into account possible diameters of an aircraft wheel, a 16-bit rotations counter is more than enough for the longest existing runway. The explanations are based on the strong requirement to avoid overflow during execution of the system: $\pi * WHEEL\_DIAMETER * NGRotations \leq LONGEST\_WORLD\_RUNWAY$ or equivalently the maximal distance of the aircraft is less than the longest world runway which is *Qamdo Bamda Airport, China, 5,500 m* following the Internet. It means that the condition $NGRotations \leq 2^{15} - 1$ and a 16-bit counter is largely sufficient. The validation of the choice of the size is based on a knowledge borrowed from an ontology. The proof of absence of overflow is then obtained automatically as long as the prover is able to handle the fact.

Electronic voting systems[6] In *Applying a Dependency Mechanism for Voting Protocol Models Using Event-B*' [6], the case study presents a method for reusing general concepts from an e-voting domain model in the formal development of specific systems within the same domain. The approach is refinement-based and thus the development is a sequence of models, moving from the abstract to the concrete. By following different refinement sequences, a family of e-voting systems can be produced that share common properties from the e-voting domain. This is illustrated in the study by a development which branches (through refinement) into two different e-voting system families. The e-voting domain knowledge is explicitly represented in the Event-B contexts. The first Event-B context introduces only the elements necessary to build an initial abstract machine for the phase of behavior associated with recording votes i.e.: sets, constants and static properties such as *Electors, Choices, Envelopes, PollStation, Representatives, Bulletins* .... As this abstract machine is refined it is necessary to extend the initial context with new conceptual elements from the e-voting domain. These correspond to specific features (increments of behavior) which the concrete system needs to offer; and they are added to the Event-B contexts as required.

Medical systems[15] Here, we describe the a posteriori approach for developing a medical protocol and we revisit the ECG interpretation protocol case study [15]. Our aim is to use domain knowledge explicitly in the development of a medical protocol including two different models: *domain model* and *system model*. The domain model describes the common medical concepts, relationships, properties and axioms related to biomedical, disease, diagnosis, anatomy, clinical procedure using several available medical ontologies (e.g. GALAN, OpenCyc, WordNet, UMLS, SNOMED-CT, FMA and Gene Ontology).The system model describes the stepwise clinical procedure for assessing the medical protocol. The Event-B models both domain and system. Domain knowledge is described in an Event-B context using ontology relations to capture the clinical procedure of the medical protocol. Note that both the domain model and system model are linked through *annotation*, in which the system model uses all axioms and theorems expressed in the domain ontology model. This model combination allows us to verify new properties related to domain knowledge within the enriched design medical protocol. We have used the ECG medical protocol for developing refinement-based formal models to systematically analyse whether the formalisation complies with certain medically relevant protocol properties. Moreover, this approach allows us to identify possible anomalies and to improve the quality of the medical protocol.

4 Conclusion

Our results show that it is possible to handle formally and explicitly domain knowledge in formal system developments with Event-B and the Rodin platform. Ontologies have been formalised within Event-B as theories and a Rodin plug-in has been developed for this purpose. Moreover, the a priori and a posteriori scenarios have been set up to define system model annotations. For the future, we plan to investigate two research directions. The first one relates to the study

of the properties of the annotation relationships, possibly modelled as Galois connections, and the second one concerns the study of domain knowledge for dynamic concepts like actions, events or transitions. Finally, experimenting the proposed approach in other application engineering areas is also planned.

References

1. Abrial, J.R.: Modeling in Event-B - System and Software Engineering. Cambridge University Press (2010)
2. Aït-Ameur, Y., Gibson, J.P., Méry, D.: On implicit and explicit semantics: Integration issues in proof-based development of systems. In: 6th International Symposium, ISoLA 2014. LNCS, vol. 8803, pp. 604–618. Springer (2014)
3. Aït Ameur, Y., Méry, D.: Making explicit domain knowledge in formal system development. Sci. Comput. Program. 121, 100–127 (2016)
4. Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L., et al.: Owl web ontology language reference. W3C recommendation 10 (2004)
5. Bjørner, D.: Manifest domains: analysis and description. Formal Asp. Comput. 29(2), 175–225 (2017)
6. Gibson, J.P., Kherroubi, S., Méry, D.: Applying a dependency mechanism for voting protocol models using event-b. In: International Conference, FORTE. Part of the 12th DisCoTec 2017. pp. 124–138 (2017)
7. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. 5(2) (Jun 1993)
8. Hacid, K., Aït-Ameur, Y.: Strengthening MDE and formal design models by references to domain ontologies. A model annotation based approach. In: 7th International Symposium, ISoLA 2016. LNCS, vol. 9952, pp. 340–357 (2016)
9. Hacid, K., Aït-Ameur, Y.: Annotation of engineering models by references to domain ontologies. In: Model and Data Engineering - 6th International Conference, MEDI 2016. LNCS, vol. 9952, pp. 234–244. Springer (2016)
10. ISO: Industrial automation systems and integration - parts library - part 42: Description methodology: Methodology for structuring parts families. ISO ISO13584-42, International Organization for Standardization, Geneva, Switzerland (1998)
11. Jackson, M., Zave, P.: Domain descriptions. In: Proceedings of IEEE International Symposium on Requirements Engineering, RE 1993, San Diego, California, USA, January 4-6, 1993. pp. 56–64
12. Jean, S., Pierra, G., Aït-Ameur, Y.: Domain Ontologies: A Database-Oriented Analysis. In: Web Information Systems and Technologies. pp. 238–254. Lecture Notes in Business Information Processing, Springer Berlin Heidelberg (2007)
13. van Lamsweerde, A.: Requirements engineering in the year 00: a research perspective. In: Proceedings of the 22nd International Conference on on Software Engineering, ICSE 2000, Limerick Ireland, June 4-11, 2000. pp. 5–19. ACM (2000)
14. Méry, D., Sawant, R., Tarasyuk, A.: Integrating domain-based features into event-b: A nose gear velocity case study. In: Model and Data Engineering - 5th International Conference, MEDI 2015. LNCS, vol. 9344, pp. 89–102. Springer (2015)
15. Méry, D., Singh, N.K.: Medical protocol diagnosis using formal methods. In: Foundations of Health Informatics Engineering and Systems - 1st International Symposium, FHIES 2011. Selected Papers. LNCS, vol. 7151, pp. 1–20. Springer (2012)
16. Mohand-Oussaïd, L., Aït-Sadoune, I.: Formal modelling of domain constraints in event-b. In: Model and Data Engineering - 7th International Conference, MEDI 2017. LNCS, vol. 10563, pp. 153–166. Springer (2017)
17. Zave, P., Jackson, M.: Four dark corners of requirements engineering. ACM Trans. Softw. Eng. Methodol. 6(1), 1–30 (1997)