Modelling Control Process and Control Mode with Synchronising Orthogonal State Machines

Colin Snook[1]

Electronics and Computer Science
University of Southampton
Southampton, England

Abstract

In this short paper we describe early work on a case study concerning a power window control unit. We use UML-B state machines to simultaneously model both the cyclic processing schedule and the mode of control behaviour. We find this a useful way to visualise the model, particularly when the state machines are animated via the Pro-B animator. We verify the state machines using the Event-B proof tools. We envisage new developments to the UML-B tool set to improve support for this modelling technique. The motivation for this simple but powerful form of modelling is the immediate benefit and low cost of entry making industrial adoption of formal models more attractive to industry.

Keywords: UML-B, State Machine, Control

1 Introduction

Development of correct software meeting the intended requirements is a primary goal of industry when developing embedded control systems. The Event-B method provides a mechanism of achieving this using formal refinements. During the development process, choosing and constructing a useful refinement chain is a difficult problem for an industrial practitioner who is not practiced in these techniques. Our aim is to make the Event-B method more accessible and easier to integrate into the development methods currently used in industry. Embedded control systems typically utilise a cyclic schedule to

[1] Email: cfs@ecs.soton.ac.uk

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

approximate a state oriented continuous behaviour as a sequential process executed at discrete time intervals. We propose a modelling approach using orthogonal state machines to represent the state oriented behaviour and the cyclic schedule.

2 Background

UML-B [7,8] is a visual 'front-end' for the Event-B notation [3] and includes a state machine diagram editor. Tool support for UML-B is provided by a plug-in to the Rodin platform [1]. State machines may be refined by adding nested state machines [4] and can be animated via a plug-in [6] that utilises the Pro-B [2] model checker and animator. The state machine refinement supported by UML-B allows the model to be progressively developed in stages. This improves understanding, and hence validity, as features can be laid down in small steps while the Rodin provers ensure consistency. Invariants can be added to states (i.e. the state becomes an implicit antecedent for the property) providing another mechanism to clearly state our understanding of the model with further consistency checking via the Rodin provers. Animation of the state machine diagrams allows us to test that they behave as we expected. However, although UML-B provides a strong graphical representation of the Event-B models, it may not be sufficient alone, to integrate with industrial development methods.

Requirement Refinement Modelling (RRM) Diagrams [5] have been proposed by Satpathy and Ramish to provide a graphical view of the requirements refinement process for embedded control processors. RRM Diagrams illustrate both process control flow and data flow abstractions and provide a good representation for integrating with industrial development methods but are, at present, informal. In this research, we use UML-B to construct RRM diagrams and to provide tool support for them.

3 Studied Case: Power Window

The case study is an automobile powered window having the following requirements.

R1 Both driver and passenger can control window movement using separate up/down switches.

R2 A driver command overrides a passenger command

R3 When the glass reaches an end-stop further commands in that direction have no impact.

R4 When the window is moving up and an obstacle is detected the window moves down for a fixed duration

R5 If a button is pressed and released before a threshold time limit then the window continues moving in that direction until it reaches an end-stop.

Fig. 1 shows a RRM diagram during the development of a power window controller of a vehicle. It has *v_dri* and *v_pass* as inputs (standing for validated driver and validated passenger commands). The box, *select_command*, is an activity which selects one of them and calls it *sel_comm*. The selected command is then executed to determine the values of *up_out* and *down_out*. These outputs determine whether the window goes up or down.

4 Interpretation as a UML-B Model

The controller executes in two orthogonal dimensions. Firstly, the controller executes in a cyclic loop, acquiring inputs, processing them and then making some decisions about control. Simultaneously and independently the controller can be thought of as progressing in a control modes dimension. In this dimension the control responds to inputs by changing its state and consequent behaviour. The control modes behaviour progresses every time the processing loop is ready to make decisions about control (i.e. *execCommand* in Fig. 2).

The refinements are summarised as follows:

PW0 Introduces the cyclic process, *getValidCommand; selectCommand; execCommand* (Fig. 2).

PW1 The select command transition is split into 2 cases, representing the selection of a driver window command and the selection of a passenger window command.

PW2 This refinement introduces the notion of modes of control. This is done by adding a state machine that is orthogonal to the previous one (Fig. 3). Some variables are also introduced to model the conditions that influence the control modes. The modes of control introduced in this refinement, concern the response to the detection of an obstacle. The transitions of the new state machine all represent different cases of the execute command transition from the cyclic process state machine. Hence *execCommand* is split into its different cases which synchronise with the transitions of the new control mode state machine (Fig. 4). State invariants express properties that should hold in the state, *obstacle_mode*. These help to ensure that the model has been constructed correctly.

PW3 In this refinement, further details of the modes of control are added. This is achieved by adding a nested state machine to the normal mode thus elaborating the behaviour in the absence of obstacles. The transitions of this new state machine represent different cases of the *execCommand* transition with no obstacle present. Hence the *EC_no_obstacle* transition in the process state machine is again split into all these new cases to achieve

synchronisation with the new state machine.

PW4 This refinement introduces more detail about the control when going up or down (in the absence of obstacles). This is done by adding nested state machines to the *Going_up* and *Going_dn* states. The two new state machines have a similar pattern. Notice that support for state machine instantiation would have avoided replicating these state machines.

PW5, PW6 and PW7 These refinements provide the mechanisms used to obtain and decode the input switches and their possible combinations. This is done by adding a nested state machine in the executed process state, hence refining the model of the acquisition process.

All refinements were proved using the Rodin provers. The majority of proofs were automatic, some requiring trivial interactions which could be made automatic by configuring the prover preferences or simply invoking a meta-prover.[2] In a few cases, the proof tree was pruned and the prover directed along a different path resulting in a quick and simple proof but one that the automatic provers could not find. Proof uncovered many simple 'typo' mistakes in the model. Animation helped us understand and demonstrate the model and is also seen as a very useful tool for model validation, but our first choice for debugging was to use the prover. The UML-B model and Event-B translation with proofs are available as a Rodin archive [9].

5 Future Work

We plan to improve support for this style of modelling in UML-B. The synchronisation of orthogonal state machines is cumbersome because UML-B has limited support for synchronising transitions. Synchronisation is achieved by name matching. However this restricts synchronisation to one to one relationships between the transitions of the two orthogonal state machines. Hence synchronisation is cumbersome because every possible control mode transition had to be replicated in the control process state machine. We would like to have a generic representation for a set of mode transitions any one of which may be synchronised with. A more flexible version of UML-B is currently being developed which has stronger integration with Event-B. In this version, transitions contribute their behaviour to events. Several transitions may contribute to the same event and a particular transition may contribute to several events. In our situation synchronisation would then be achieved by having each transition in the mode state machine contribute to a correspond-

[2] Proof statistics can be misleading due to the configurability of the Rodin automatic prover preferences. However, out of 964 proof obligations, initially 37 required interactive proof. This was reduced to 6 interactive proofs after adding the Relevance Filter meta prover and raising the priority of the Atelier-B ML prover

ing event and a single transition in the process state machine contribute to all of these events. We also plan to research into ways to support instantiation of state machines. This would have avoided the need to repeat the nested state machines introduced in refinement step PW4.

Control systems are often developed using Simulink/Stateflow models obtained from the requirements in an ad-hoc way. RRM diagrams correspond well with Simulink/Stateflow models. RRM control flows define the sequencing of Simulink components while RRM data flows correspond to Simulink data flows. We intend to extend UML-B to derive Simulink/Stateflow models automatically. Since Simulink/Stateflow models are widely used in control applications, this would be a significant contribution to the development of control applications in industry.

6 Conclusion

RRM diagrams model both control and data-flow abstractions of control systems. Information from them helps guide the refinement path. In this research, we use UML-B to provide tool support for RRM diagrams, manually interpreting the RRM diagrams as orthogonal, synchronising UML-B state machines and utilising UML-B's support for refinement via translation to Event-B to verify the models using the Event-B provers. We also animate the diagrams via the Pro-B animation engine in order to validate the models.

References

[1] Butler, M., Hallerstede, S.: The rodin formal modelling tool. BCS-FACS Christmas 2007 Meeting - Formal Methods In Industry, London. (December 2007)

[2] Leuschel, M., Butler, M.: ProB: A model checker for B. In Araki, K., Gnesi, S., Mandrioli, D., eds.: FME 2003: Formal Methods. LNCS 2805, Springer-Verlag (2003) 855–874

[3] Metayer, C., Abrial, J.R., Voisin, L.: Event-B Language. Rodin deliverable 3.2, EU Project IST-511599 -RODIN (May 2005)

[4] Said, M. y., Butler, M. and Snook, C.: Language and Tool Support for Class and State Machine Refinement in UML-B. In: FM2009 - 16th International Symposium on Formal Methods, 2-6th November 2009, Eindhoven. pp. 579-595.

[5] Satpathy M, Ramesh S.: Formal Foundation to Systematic Development of Simulink/Stateflow models. In Proc. of the Dagstuhl Seminar on *Refinement based methods for the construction of dependable systems*, 2009

[6] Savicks, V., Snook, C. and Butler, M.: Animation of UML-B Statemachines. Technical Report (http://eprints.ecs.soton.ac.uk/18261/1/TBFMsmAnim.pdf) and presented at Rodin User and Developer Workshop, Sept, 2010.

[7] Snook, C., Butler, M.: UML-B: Formal modeling and design aided by UML. ACM Trans. Softw. Eng. Methodol. **15**(1) (2006) 92–122

[8] Snook, C., Butler, M.: UML-B and Event-B: An integration of languages and tools. In: The IASTED International Conference on Software Engineering - SE2008. (February 2008)

[9] Snook, C.: Power window case study - UML-B model and Event-B translation with proofs. Rodin Archive (http://eprints.ecs.soton.ac.uk/22287/)

Fig. 1. RRM diagram showing initial abstract representation of power window model

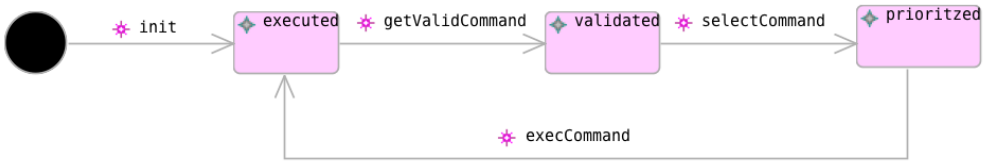

Fig. 2. State machine in the process dimension

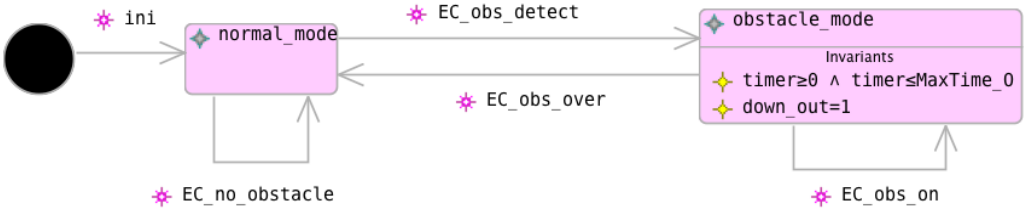

Fig. 3. State machine in the control modes dimension showing refinement of the *execCommand* transition from Fig 2 by orthogonal state machine

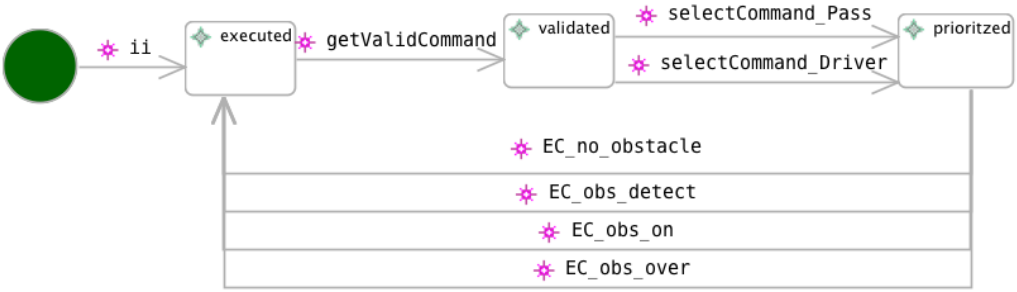

Fig. 4. Refined State machine in the control process dimension showing split execute commands to synchronise with the transitions of the orthogonal control modes state machine (Fig. 3)