Heterogeneous Model Composition in ModHel'X: the Power Window Case Study

Frédéric Boulanger, Christophe Jacquet, Cécile Hardebolle, and Ayman Dogui

Supélec E3S – Computer Science Department
`firstname.lastname@supelec.fr`
`http://wwwdi.supelec.fr/software`

Abstract. This paper describes an heterogeneous model of a power window which is available on the ReMoDD repository. This model uses timed finite state machines for modeling the controller of the power window, synchronous data flows for modeling the mechanical part of the window, and discrete events for modeling the communications between the components on the car's bus. An important aspect of this model is the specification of the semantic adaptation between the heterogeneous parts of the model. This semantic adaptation is made for data, control, and time. The semantic adaptation of control and time relies on the TESL library which is an implementation of the model of time used in the ModHel'X platform. The model can be run using a simulation scenario with a graphical display of the outputs. The semantic adaptation can be disabled in order to show how it affects the behavior of the model. The demo can also be run with a graphical interface and a user in the loop.

1 Introduction

Using the most suitable set of concepts for modeling a system allows for compact and understandable models. A modeling paradigm with a restricted set of concepts helps staying focused on the problem that must be solved and reduces accidental complexity. It may even allow the use of formal analysis tools on the model. However, because of its efficient but limited set of concepts, such a paradigm cannot be used for modeling a complex system, with numerous and heterogeneous components. In such a case, *several modeling paradigms* are necessary to model the different components of the system, to study the system under different points of view and at different levels of abstraction, or to perform different kinds of analysis [1]. Making those different modeling paradigms cooperate in the global model of the system is therefore essential.

Multi-paradigm modeling [2] requires well-defined semantics for modeling languages as well as mechanisms for composing heterogeneous models. Supélec's Department of Computer Science is developing ModHel'X, a platform for building and executing heterogeneous models. It constructs the behavior of a model by combining the observations of the behavior of black box components according to a model of computation. More details can be found in [1,3].

ModHel'X has been inspired by Ptolemy [4] but focuses on the modeling of the *semantic adaptation* between models of computation. Semantic adaptation is the "glue" that is necessary at the boundary between two parts of a model that use different modeling paradigms so that the resulting composed model has a well-defined semantics. The problem of semantic adaptation, and the approach we use for solving it are described in [5,6] and illustrated on former versions of the power window case study. In this article we present a new version of this model, which relies on TESL (Tagged Events Specification Language) — a model of time we developed as an extension with time tags of a restriction of CCSL (Clock Constraints Specification Language [7]) — for specifying semantic adaptation. This model of time is described in a technical report [8].

In the following, we first present the power window system, then we show how it is modeled using several modeling paradigms in ModHel'X, and we present the different kinds of semantic adaptation needed in the model.

2 The Power Window System

Figure 1 shows the structure of the power window system. The switch sends user commands to the controller through the bus. The controller receives the commands from the switch as well as information about the end stops and obstacle detector from the electro-mechanical part. It sends commands to the electro-mechanical part to switch the motor on or off.
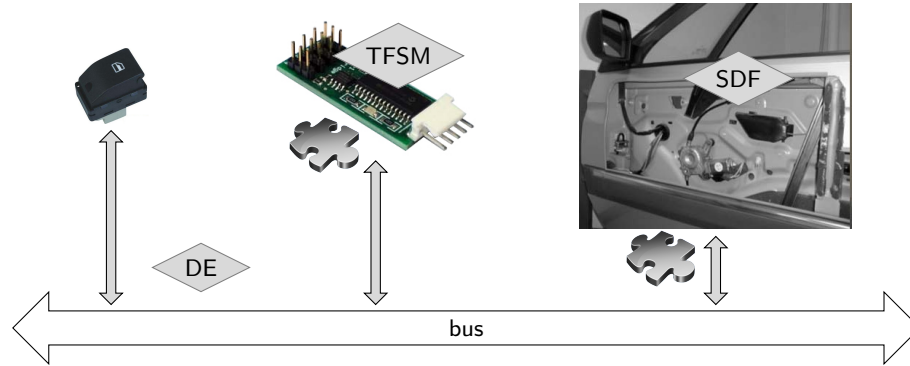

Fig. 1. The power window example

Since commands which transit on the bus occur at specific instants and carry data, we model the bus using the Discrete Events (DE) model of computation. The behavior of the switch is not modeled explicitly, and we will either use a scenario player to simulate the actions of the user, or a graphical user interface with virtual buttons. The controller is naturally modeled using a state machine. To implement an automatic impulse mode to completely close or open the window when the user presses the button for a short time (less than 500 ms), we need timed transitions, so we use the Timed Finite State Machine (TFSM) model of computation. The continuous dynamics of the electromechanical part of the

window could be modeled using differential equations. However, since we do not yet have a continuous solver in ModHel'X, we use a discretized model consisting of difference equations, which corresponds to the Synchronous Data Flow (SDF) model of computation.

Semantic adaptation is necessary at the boundary between heterogeneous parts of the model, and it is represented by puzzle pieces on the figure.

3 The Power Window in ModHel'X

Figure 2 shows the structure of the model of the power window in ModHel'X. The root model uses the DE model of computation, and its structure contains five blocks. On the left and right sides, the scenario and display are used to feed the model with events and to display the results. Similar components are used for interfacing the model with the simulation interface in the interactive version.
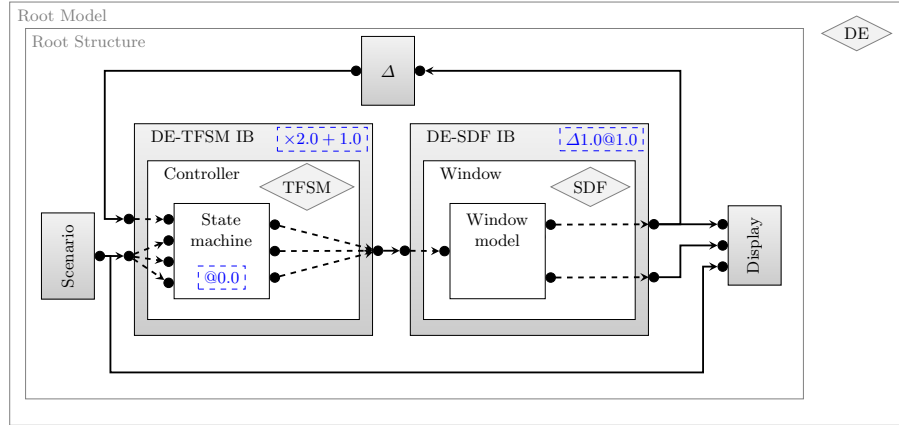
Fig. 2. Structure of the model of the power window

The model of the controller, which uses the TFSM model of computation, and the model of the window, which uses the SDF model of computation, are embedded in interface blocks (respectively DE-TFSM IB and DE-SDF IB). Interface blocks is the mechanism used in ModHel'X to specify semantic adaptation between heterogeneous models. The block labeled Δ in the top part of the model is a delay which models the time taken by the end stop information produced by the window to reach the input of the controller. This delay is modeled explicitly in order to avoid instantaneous feedback, which is not yet supported by our implementation of the DE model of computation. We detail the role of the two interface blocks with respect to semantic adaptation in the next section.

4 Semantic Adaptation

The concept of semantic adaptation has been detailed and illustrated on the power window model in [6]. In particular, we have shown that semantic adapta-

tion can be decomposed along three axes: the adaptation of data, the adaptation of the control flow, and the adaptation of time scales. In this paper, we show how the semantic adaptation of time and control can be specified at a more abstract level using the TESL model of time (see [8] for more details on TESL).

4.1 Adaptation between DE and TFSM

The adaptation of data between DE and TFSM consists in transforming DE events into input symbols for the state machine, and output symbols from the state machine into DE events. This is achieved by decoding the value of input DE events into symbols and encoding symbols into values for the output DE events. On Figure 2, the dashed arrows between the pins of the interface blocks and the pins of their internal model represent this adaptation of data between models of computation. The interface block implements this generic decoding/encoding policy. The details which are specific to a model are set using properties of the pins. Here, the scenario produces events with values -1, 0 and 1, which are decoded into input symbols cmd_up, cmd_stop, and cmd_down of the controller. Similarly, the properties set on the output pins of the interface block make it encode the output symbols motor_up, motor_stop, and motor_down of the controller into events with respective values 1, 0 and -1, which will be interpreted accordingly by the SDF model of the window.

Semantic adaptation between DE and TFSM is also necessary with respect to time. The timed transitions in the state machine are expressed according to a time scale which is not necessarily the same as the time scale used in the discrete event model. In this example, in order to illustrate this aspect, we choose arbitrarily to make time advance twice as fast in TFSM than in DE, with an offset of one unit of time. This is represented by the "$\times 2.0 + 1.0$" label in a dashed frame on the figure and it is realized by a TESL tag relation between the clock of the DE model and the clock of the state machine. The controller has an initial transition which is used to initialize the output, and we must specify at what time this transition occurs. We choose to trigger it at time 0.0 on the TFSM time scale, as indicated by the "@0.0" label in a dashed frame on the state machine. These timing labels are interpreted by the interface block and translated into calls to the TESL Java API.

The last aspect of semantic adaptation between DE and TFSM is the control flow. When the controller enters a state from which a timed transition starts, it must be updated at the current time plus the delay of the transition, even if it receives no input symbol at this time. For this, the interface block uses the TESL Java API to create an implication relation between the clock of the state machine and the clock of the DE model, and to request the occurrence of an event on the TFSM clock at the expiration time of delay transitions.

4.2 Adaptation between DE and SDF

The semantics of SDF implies that a model can be updated only when it has all its data available in input. The semantic adaptation of data here consists in

maintaining the current value of the input each time an event is received from the controller, and to feed the SDF model with this input value each time it must be updated. On the output side, we choose to produce events only when the value of an SDF output changes.

The SDF model of computation has no notion of time. However, an SDF model may have been designed in such a way that its behavior is correct with respect to the rest of the system only if it is updated periodically on some time scale. In this example, we choose to update the SDF model periodically on the DE time scale. The role of the interface block is therefore to update the SDF model at each period, but also to take care of not updating the SDF model at other instants even if some data is available on input, because such updates would break the periodic pattern. The "$\Delta 1.0@1.0$" label in a dashed frame specifies that the DE-SDF IB interface block should use the TESL API to request an update of the SDF model every 1.0 unit of time starting at date 1.0.

It is important to note that the two interface blocks used in this model implement *generic adaptation patterns*. They define standard ways of composing DE with SDF and DE with TFSM. Their particular behavior in this example is obtained by interpreting the structure of the adaptation relations between the pins of the interface block and the pins of its internal model, and by taking into account specific properties the user has set for these pins and models.

5 Simulation

Get `http://modhelx.vze.com/2013-09-05_ModHelX_PowerWindow.zip`, then run demos.powerwindow.PowerWindowDisplay.java to obtain the result shown on the left of Figure 3. If an argument is given when running this example, the program switches to a very basic semantic adaptation between the DE model and the SDF model, which leads to a completely different behavior, and illustrates the importance of explicit semantic adaptation.

Running PowerWindowGUI.java should display the graphical interface shown on the right of the figure. One can click on the triangle-shaped buttons to close and open the window. They turn green when the motor is on. One can click on the "obstacle" checkbox to have an obstacle appear in the way of the window. One can check that the controller makes the window move down before stopping.
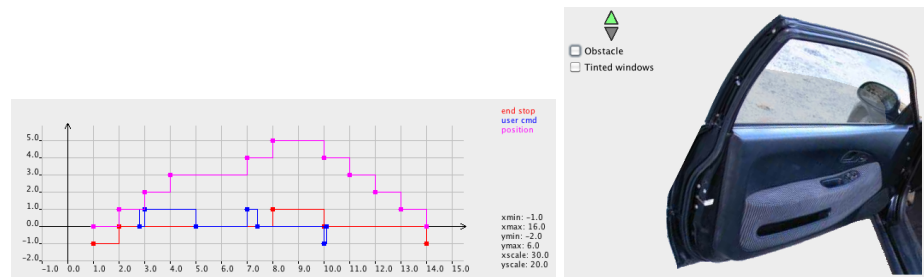
Fig. 3. Results of the execution of the model

6   Conclusion

This model illustrates how three different modeling paradigms (Discrete Events, Timed Finite State Machines and Synchronous Data Flows) can be used to model different parts of a global model in the most suitable way. It also illustrates how important is the explicit modeling of the semantic adaptation between heterogeneous models. The source code has been made available on ReMoDD.

In this implementation, semantic adaptation is provided as reusable interface blocks which can be parameterized (for instance, the update period of the SDF model is given by setting a property of the interface block) and which model a particular policy of adaptation. However, although the time and control aspects of semantic adaptation are specified using the TESL time model, the adaptation of data is written in Java, and the interaction with the TESL solver is made through its Java API. This creates a semantic gap between the implementation and the abstract notions involved in the modeling of semantic adaptation. We are currently working on the design of a domain-specific language for modeling semantic adaptation in order to provide the designers of a model with a more suitable level of description and to allow eventually the verification of properties on the composition of heterogeneous models.

References

1. Hardebolle, C., Boulanger, F.: Exploring multi-paradigm modeling techniques. SIMULATION **85**(11/12) (November/December 2009) 688–708
2. Mosterman, P.J., Vangheluwe, H.: Computer Automated Multi-Paradigm Modeling: An Introduction. SIMULATION **80**(9) (2004) 433–450
3. Hardebolle, C., Boulanger, F.: Multi-formalism modelling and model execution. Intl Journal of Computers and Applications **31**(3) (July 2009) 193–203
4. Eker, J., Janneck, J.W., Lee, E.A., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y.: Taming heterogeneity – The Ptolemy approach. Proceedings of the IEEE **91**(1) (January 2003) 127–144
5. Boulanger, F., Dogui, A., Hardebolle, C., Jacquet, C., Marcadet, D., Prodan, I.: Semantic adaptation using CCSL clock constraints. In Kienzle, J., ed.: Models in Software Engineering. Volume 7167 of Lecture Notes in Computer Science., Springer-Verlag (2012) 104–118
6. Boulanger, F., Hardebolle, C., Jacquet, C., Marcadet, D.: Semantic adaptation for models of computations. In Caillaud, B., Carmona, J., Hiraishi, K., eds.: Proceedings of the 11th International Conference on Application of Concurrency to System Design, IEEE Computer Society (2011) 153–162
7. Mallet, F., Deantoni, J., André, C., De Simone, R.: The Clock Constraint Specification Language for building timed causality models. Innovations in Systems and Software Engineering **6**(1-2) (March 2010) 99–106
8. Boulanger, F., Hardebolle, C., Jacquet, C., Prodan, I.: Modeling time for the execution of heterogeneous models. Technical report 2013-09-03-DI-FBO, Supélec E3S - Computer Science Department (September 2012) `http://wwwdi.supelec.fr/software/downloads/InternalReports/Report_2012-09-03-DI-FBO.pdf`.