Model-Checking the ETCS Ceiling Speed Monitor Function Using a SysML Model

2014-07-01

Table of Contents

Figures and Tables

Figures

Tables

1 Introduction

We study the SysML model of the ceiling speed monitoring (CSM) function using the Tina model-checking toolbox. We base our analysis on a model provided by the University of Bremen that was slightly extended with information on the environment of the system. The same function was studied by the University of Rostock using a software testing approach.

The cornerstone of our approach is an automatic transformation from SysML models to Time Petri Net. We can then use the resulting formal model to check several temporal logic formulas.

2 Model Transformation from UML to Time Petri Nets

This section gives some background information on the translation from SysML to Time Petri Net (TPN) used in our study. This transformation is based on a mapping from UML (Unified Modeling Language) to TPN described in the PhD thesis of Ning Ge [2]. The transformation can also take into account realtime properties defined using the MARTE profile (Modeling and Analysis of Real Time and Embedded systems).

By nature, UML is intended to be a general purpose modeling language and, as such, it integrates different modeling viewpoints through the definition of a large class of diagram elements. In this work, we select a core subset of UML diagrams and diagram elements for modeling real-time software architecture and behavior, and focus on the semantic mapping from the UML model to the verification model.

We briefly describe the different elements supported in our translation.

2.1 Architecture Model

The purpose of architecture model is to connect different sub-system behavior models and create a system-level model, by means of communication media. The objective of the mapping is to replace each architecture model's entities by its relevant behavior model. We rely on the composite structure diagram as the architecture model. Composite structure diagrams specify the internal structure of a class, including its interaction points to other parts of the system, and the architecture of all parts managed by this class. They are used to explore run-time instances of interconnected instances collaborating over communications links.

2.2 Behavioral Model

The mapping semantics for behavioral model covers both activity and state machine diagrams.

Activity diagrams express the coordination between lower-level behaviors using constraints on the possible sequence of actions. In this context, actions can be triggered because other actions finish executing; because objects and resources become available; or because external events occur. The main elements in UML activity diagram behavior model are control nodes, actions, objects, and connection elements.

2.3 Principles of Semantic Mapping

The mapping from UML-MARTE to TPN preserves the semantics of the input language. A particularity of the approach is that, for efficiency reason, the transformation is driven by the set of real-time properties that should be checked on the resulting model. For instance, in order to

reduce the size of the state space explored during the verification phase, the behavior of some elements irrelevant to the target property can be abstracted. The transformation conforms to the following principles:

- The resulting TPN models should be easy to analyze, meaning that the semantics mapping should allow the use of high-level abstraction methods during model checking.
- In order to keep the transformation simple, we use a compositional approach where the resulting system is obtained by composing the interpretation of all its elements. Then, to optimize the result, we apply a state space reduction phase that eliminates the elements irrelevant to the verification.

3 Model Transformation from SysML to Time Transition System

Instead of the thirteen diagrams available in UML 2, the SysML includes only nine diagrams, including:

- the Block Definition Diagram (BDD), replacing the UML 2 class diagram
- the Internal Block Definition Diagram (IBDD), replacing the UML 2 composite structure diagram
- the Parameter diagram, a SysML extension to analyze critical system parameters
- the Package diagram remained unchanged.

The behavior diagrams includes:

- the activity diagram, slightly modified from UML 2
- the sequence, state machine, and use case diagrams remain unchanged.

The requirement diagram is a SysML extension to describe functional, performance, and interface requirements.

In order to reuse the existing transformation form UML to TPN [2] to build a mapping from SysML, we have redefined the mapping semantics for the block diagram as structure models. The mapping semantics for the activity and state machine diagrams are left unchanged. Some of the semantics mapping have also been modified in order to take into account some of the modeling convention adopted in the OpenETCS project.

Also, the target model is now an extension of Time Petri Net with priorities and typed variables called TTS, for Time Transition System. The data handling ability of TTS is used to model the guards and actions on integer and float variables found in a SysML diagram.

4 Description of the Ceiling Speed Monitoring Model and its Extension

The detailed CSM SysML model is described in [1] and available in the 05-Work folders. The model was edited, and later extended, using Papyrus. The main addition made to the existing

model for the CSM function is the definition of the environment; mainly the possible actions on the current speed of the train resulting from acceleration or deceleration orders.

The combination of a test environment model and an optional test driver model provides a deterministic model

4.1 Description of the Environment Model

The environment model is defined in the TestEnvironment block. It is described using a state machine diagram as shown in Fig. 1.

The system under test is activated with an initial speed (SimulatedTrainSpeed) and may enter into the composite state Running. The Running state encapsulates three possible behavior. Either the speed remains unchanged (state Normal), or the train accelerates (state Acc), or the train decelerates (state Dec). The guards defined on the transitions found inside the composite state Running are described in Table 1. The DriveCommand is the command sent by the driver which contains three modes: keeping speed (DriveCommand = 0), acceleration (DriveCommand = 1) and Deceleration (DriveCommand = 2).

Table 1. Transitions between Running States

From / To	Normal	Acc	Dec
Normal		DriveCommand != 1	DriveCommand != 2
Acc	DriveCommand == 1		
Dec	DriveCommand == 2		

The actions on each transition (the "do behaviors") are described in Table 2. At the moment we use dummy values for the acceleration and braking parameters of the train. We have chosen a fix acceleration of 2 km/h per 100 units of time (u.t.) and a constant braking factor of 2 km/h per 400 u.t..

Table 2. Do Behaviors in Running States

Normal	
Acc	SimulatedTrainSpeed = SimulatedTrainSpeed + 2;
Dec	SimulatedTrainSpeed = SimulatedTrainSpeed - 2;

The effect of the environment on the system is described by the transitions between states Running, EBrake (emergency brake), SBrake (service brake) and STOP (simulatedTrain Speed <= 1). The transition guards between environment states are described in Table 3. The transitions are controlled by the commands EmergencyBrakeCommand and ServiceBrakeCommand generated from the train control system.

The do behaviors in environment states are described in Table 4. The deceleration of emergency brake is set at 10 km/h per 200 u.t.. The deceleration of service brake is set at 5 km/h per 200 u.t..

4.2 Description of the Driver Model

Table 3. Transitions between Environment States

From / To	Running	EBrake	SBrake	STOP
Running		ServiceBrakeCommand == 0 && EmergencyBrakeCommand == 0	ServiceBrakeCommand == 0 && EmergencyBrakeCommand == 0	
EBrake	EmergencyBrakeCommand == 1		ServiceBrakeCommand == 0 && EmergencyBrakeCommand == 1	
SBrake	ServiceBrakeCommand == 1 && EmergencyBrakeCommand == 0	ServiceBrakeCommand == 1 && EmergencyBrakeCommand == 0		
STOP	SimulatedTrainSpeed <= 1			

Table 4. Do Behaviors in Environment States

Running	
EBrake	SimulatedTrainSpeed = SimulatedTrainSpeed - 10;
SBrake	SimulatedTrainSpeed = SimulatedTrainSpeed - 5;
STOP	

In addition to the model of the train behavior we have added a model of the driver (and of the track description) that can be used to force a particular scenario. In this context, a scenario is a timed annotated sequence of acceleration and deceleration orders. One possible scenario can be obtained using the model defined in the TestDriver block of Fig. 2. This test describes a situation where the driver accelerates for a time T1 (DriveCommand = 1) before decelerating for a time T2.

5 Model Checking the Resulting TTS

We provide two verification scenarios for testing the formal system obtained from the transformation of the CSM model. Each scenario is available as a TTS "file" (actually a folder called tpn.tts) inside the 05-Work folder.

5.1 Scenario 1

Scenario 1 includes the following initial values for the parameters:

- SimulatedTrainSpeed = 110
- V_mrsp = 120
- SBAvailable = true
- DriveCommand = 1. The initial drive command is acceleration.
- T1 transition in Driver model has an effect behavior DriveCommand = 1. The time duration for the initial behavior (acceleration) is 20000.
- T2 transition in Driver model has an effect behavior DriveCommand = 0. The time duration for the behavior before keeping speed (acceleration) is 10000.

We give in the table below the number of reachable states (or markings) of the resulting TPN. A marking is defined by a particular value for each system variable and for each internal state of

the blocks. This gives a rough idea of the complexity of checking reachability properties on the system. "Classes" take into account timing constraints on top of the markings; hence there is always more classes than markings. The generation of the whole state space takes for Scenario 1 takes less than 24 seconds (system time: 23.350s).

markings	domains	classes	transitions
399	455880	456926	978970

5.2 Scenario 2

Scenario 2 includes the following initial values of parameters:

- SimulatedTrainSpeed = 0
- V_mrsp = 160
- SBAvailable = true
- DriveCommand = 1. The initial drive command is acceleration.
- T1 transition in Driver model has an effect behavior DriveCommand = 2. The time duration for initial behavior (acceleration) is 200000.
- T2 transition in Driver model has an effect behavior DriveCommand = 0. The time duration for the behavior before keeping speed (deceleration) is 100000.

The size of the state graph for scenario 2 is shown in the table below. The generation of the whole state space takes less than 26s (system time of 25.912s).

markings	domains	classes	transitions
474	700129	700472	1201679

6 Verification of Requirements

We have used our model-checking toolbox to check the properties stated in the work by Univ. Bremen [1]. These properties are a direct translation into temporal logic of the requirements found on the Subset 026 documents. Since the CSM model does not take into account the possible activation and de-activation of the CSM, we have not dealt with three requirements. (By default, the CSM is always activated.) We provide an interpretation of the nine remaining requirements using LTL, see Table 5. To simplify the the LTL formula, we have used simple names for naming the relevant variables. Full names, integrating information on the hierarchy, should be used when model-checking the actual systems in Tina.

Table 5. LTL

Requirement	LTL Formula
req_01	EmergencyBrakeCommand ∧ SBAvailable=0 ∧ SimulatedTrainSpeed <= V_mrsp ∧ RevocationEmergencyBrake=0
req_02	ServiceBrakeCommand ∧ SBAvailable=0 ∧ SimulatedTrainSpeed <= V_mrsp
req_03	ServiceBrakeCommand ∧ SBAvailable=0 ∧ SimulatedTrainSpeed gt (V_mrsp + dV_sbi) ∧ SimulatedTrainSpeed lt (V_mrsp + dV_ebi)
req_08	⋄ ((NORMAL ∧ SimulatedTrainSpeed <= V_mrsp) U (NORMAL ∧ SimulatedTrainSpeed gt V_mrsp + dV_warning ∧ SimulatedTrainSpeed <= V_mrsp + dV_sbi))
req_09	⋄ ((NORMAL ∧ SimulatedTrainSpeed <= V_mrsp) ∧ ((NORMAL ∧ SimulatedTrainSpeed <= V_mrsp) U (NORMAL ∧ SimulatedTrainSpeed gt V_mrsp + dV_sbi ∧ SimulatedTrainSpeed <= V_mrsp + dV_ebi)))
req_10	⋄ ((NORMAL ∧ SimulatedTrainSpeed <= V_mrsp) ∧ ((NORMAL ∧ SimulatedTrainSpeed <= V_mrsp) U (NORMAL ∧ SimulatedTrainSpeed gt V_mrsp + dV_sbi)))
req_11	⋄ ((OVERSPEED ∧ SimulatedTrainSpeed <= V_mrsp) ∧ ((OVERSPEED ∧ SimulatedTrainSpeed <= V_mrsp) U (NORMAL ∧ SimulatedTrainSpeed gt V_mrsp + dV_sbi ∧ SimulatedTrainSpeed <= V_mrsp + dV_ebi)))
req_12	⋄ ((OVERSPEED ∧ SimulatedTrainSpeed <= V_mrsp) ∧ ((OVERSPEED ∧ SimulatedTrainSpeed <= V_mrsp) U (NORMAL ∧ SimulatedTrainSpeed gt V_mrsp + dV_sbi)))
req_13	⋄ ((WARNING ∧ SimulatedTrainSpeed <= V_mrsp) ∧ ((WARNING ∧ SimulatedTrainSpeed <= V_mrsp) U (WARNING ∧ SimulatedTrainSpeed gt V_mrsp + dV_ebi)))

References

[1] Cécile Braunstein, Jan Peleska, Uwe Schulze, Felix Hübner, Wen ling Huang, Anne E. Haxthausen, and Linh Vu Hong. A SysML test model and test suite for the ETCS ceiling speed monitor. Technical report, Univ. Bremen, 2014.

[2] Ning Ge. *Property Driven Verification Framework: Application to Real-Time Property for UML-MARTE Software Designs*. Ph.d., Institut National Polytechnique de Toulouse, Toulouse, 2014.

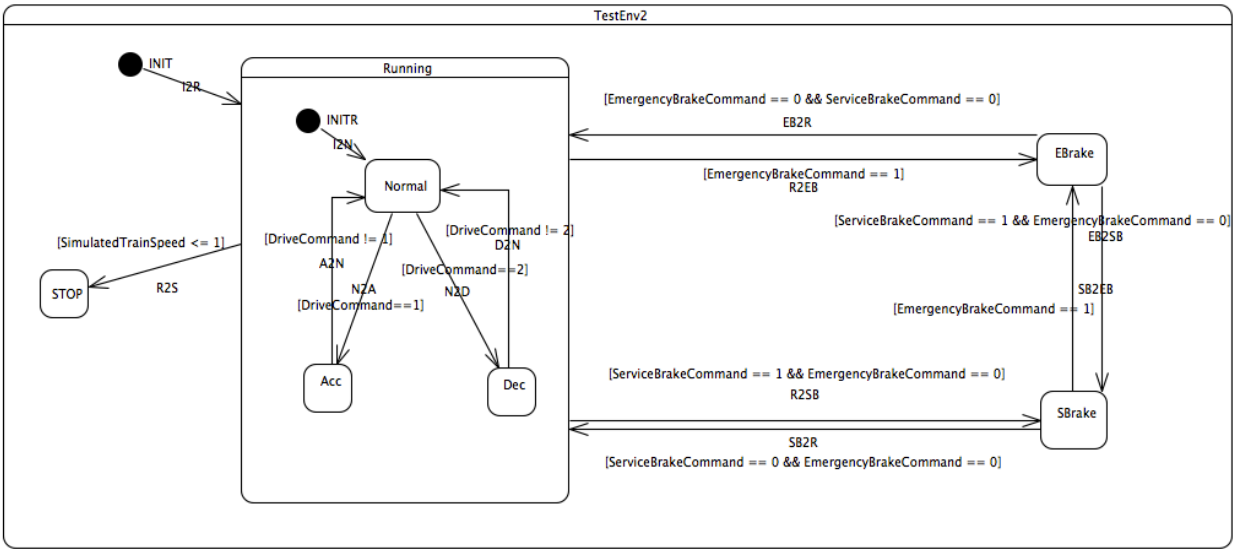

Figure 1. Test Environment Model

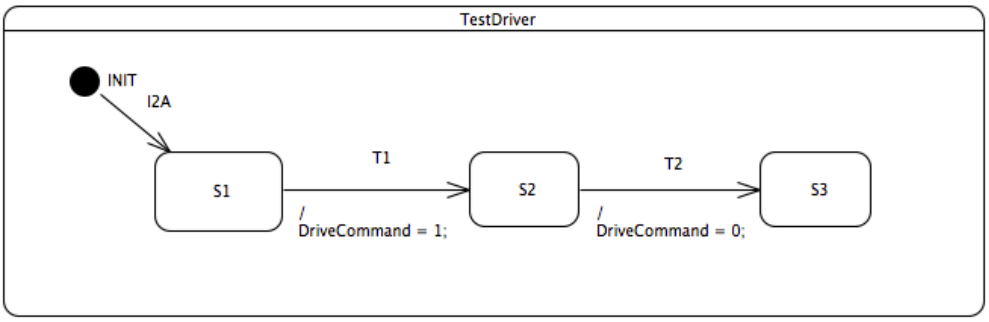

Figure 2. Test Driver Model