Powertrain Model Assessment for Different Driving Tasks through Requirement Verification

Anders Andersson[1] Lena Buffoni[2]

[1]Swedish National Road and Transport Research Institute, Sweden, anders.andersson@vti.se
[2]IDA, Linköping University, Sweden, lena.buffoni@liu.se

Abstract

For assessing whether a system model is a good candidate for a particular simulation scenario or choosing the best system model between multiple design alternatives it is important to be able to evaluate the suitability of the system model. In this paper we present a methodology based on finite state machine requirements verifying system behaviour in a Modelica environment where the intended system model usage is within a moving base driving simulator. A use case illustrate the methodology with a Modelica powertrain system model using replaceable components and measured data from a Golf V. The achieved results show the importance of context of requirements and how users are assisted in finding system model issues.

Keywords: system model assessment, requirement modelling, Modelica, finite state machine, powertrain validations

1 Introduction

With the increasing complexity of cyber-physical systems, determining whether a particular system design alternative fulfils all the requirements that are imposed on the system under development can no longer be done manually and requires formalizing the requirements into some computable form. Verifying the validity of a system design through simulation will reduce the risk of modelling errors and allow to evaluate the suitability of the model for a particular purpose.

In the context of this paper, we illustrate the validation process on a powertrain model with an intended use in a driving simulator. A simulator model is validated before conducting a driving simulator study as well as over the whole evaluation time period and in particular whenever a developer changes parts of a model, to guarantee that the model is suitable for the intended driving tasks.

One common way to test a powertrain is to use driving cycles. For the use case in this paper we have logged a driver for two different driving cycles, the Artemis Road Driving Cycle and the 130 km/h variant of the Artemis Motorway Driving Cycle. Using this logged data it is possible to run the model offline and we use these datasets to verify that the model is working as intended using requirements.

The physical model and the requirement model are both written in Modelica (Modelica Association, 2014). Using the same language to express both the requirement and the design model simplifies the co-simulation of the two. The declarative nature of Modelica lends itself well to the description of the requirement model and the component based nature of the verification framework allows to quickly create different configurations for testing.

The paper is organized as follows: Section 2 presents the use case that will be used to illustrate the methodology, Section 3 describes the requirement model, Section 4 illustrates the setup for the whole verification framework, used in Section 5 to show the model validation process, and finally the conclusion and future work are discussed in Section 6.

2 Use Case

To acquire data for the powertrain system models and the requirement model the vehicle propulsion laboratory at Linköping University was used. In this laboratory chassis dynamometers are connected to the wheel hubs of the test vehicle, in this case a Golf V, measuring signals such as the torque and rotational speed at the driving wheels. Used setup, shown in Figure 1, is further described in (Öberg *et al*, 2013).

Figure 1. The Golf V used for measuring powertrain data connected to chassis dynamometers at the vehicle propulsion laboratory at Linköping University.

2.1 Driving Task Specification

The system models are used to simulate the powertrain during various driving tasks. Examples of such driving tasks are driving monotonously on a motorway with low traffic or city driving which typically includes more accelerations and driver input. Driving tasks can be represented by driving cycles. Thus, to connect our conclusions to real driving, two different driving cycles were used, the 130 km/h variant of the Artemis Motorway Driving Cycle and the Artemis Road Driving Cycle, see (Andre, 2004). In the laboratory, the driver was asked to drive according to the chosen driving cycle as close as possible.

Gathered data was used to parameterize the system model and also as a test case to evaluate the requirement model's performance. Since the requirement model should ensure that the system model captures the driving cycle characteristics and thus it is suitable for the represented driving task.

3 Requirement Model

Alongside the system model for the powertrain, we define the requirement model. It is important to note that the requirement model should not impact the physical model of the system and therefore has read-only access to the information necessary for the verification.

3.1 Requirement Modelling in Modelica

To represent requirements in Modelica, we use the following conventions (Schamai *et al*, 2014):

- A requirement is identified by extending the partial Requirement interface.
- A requirement is associated with a status and a set of properties to reason on the status.

A status can take the following values:

- `VIOLATED` when the conditions of the requirement are not fulfilled by the design model;
- `NOT_VIOLATED` when the conditions of the requirement are fulfilled by the design model;
- `NOT_APPLICABLE` when the requirement does not apply, for instance a requirement that describes the behavior of a vehicle when switching gears cannot be verified in a scenario where the vehicle is always in first gear. This is important to identify requirements that were never tested during a simulation.

It is important to note that the status of a requirement evolves over time, and that the status of the requirement at the last instant of the simulation cannot be used to determine whether the requirement has been violated earlier in the simulation. For this reason, each requirement is also associated with the following variables:

- `hasBeenVerified` indicates if a requirement has ever been checked during a simulation run
- `hasBeenViolated` indicates if a requirement has been violated during a simulation run

These predicates can be used to analyze the simulation results.

There is no unique way to specify a requirement model, but in this paper we choose to represent requirement as finite state machines (Thiele *et al*, 2015) because this allows to intuitively map the state of the system through inputs to the 3 possible states of the requirement. Other alternatives would have been representing requirements as conditional equations or using a dedicated library (for example (Otter *et al*, 2015)).

In is important to note that when modeling requirements as state-machines the clock frequency is a key design choice. In the current implementation the transitions are triggered by a clock event and thus a zero crossing is not detected. As a consequence, an event with a frequency shorter than the clock interval can possibly be missed. For the simulations in this study case we set the clock period to 0.5.

3.2 Powertrain Use Case Requirements

For this case study we have selected a set of 4 different requirements where three of them are related to system model validity and one is related to model logics. A textual description of the requirement set is given in Table 1.

Table 1. Textual requirement description.

Requirement ID	Description
`req1`	The accelerator and clutch pedal value should always be between 0 and 1.
`req2`	When driving calmly there should be a limited amount of gear changes per minute when using an automatic gearbox.
`req3`	When the car is moving the difference between modeled and measured engine RPM should be below an acceptable error when the clutch is not used.
`req4`	When the car is moving the difference between modeled and measured vehicle speed should be small over time.

The textual descriptions of requirements in Table 1 can be ambiguous. For instance it is unclear, whether "between 0 and 1" is an open or a closed interval. Formalizing these requirements as a computable model removes such ambiguities. As described in the previous

section, each requirement in Table 1 is modeled by a finite state machine. For an example of how these requirement finite state machines look, see Figure 2.

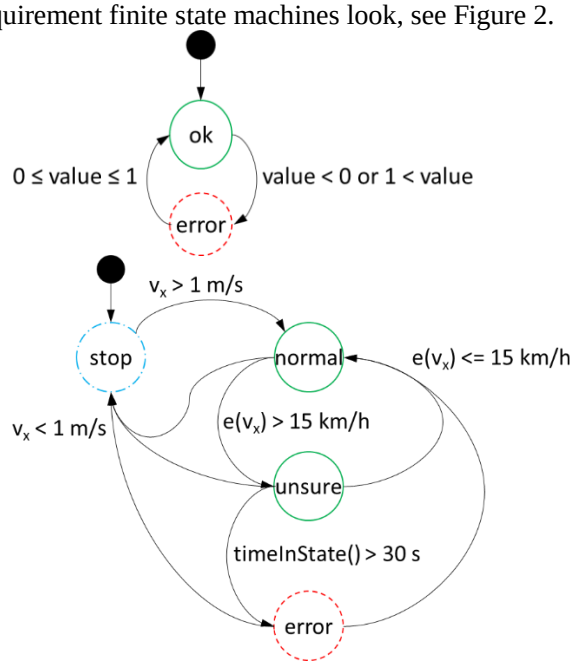

Figure 2. Finite state machines used to model **`req1`** above and **`req4`** below. The requirement status corresponds to state circle format where a green line means **`NOT_VIOLATED`**, a red dashed line means **`VIOLATED`** and a blue dash dotted line means **`NOT_APPLICABLE`**.

One of the advantages of using a component-based language to model requirements is the possibility of hierarchically composing smaller requirements into more complex requirements. For instance, `req1` is a combination of two requirements: one requirement for the acceleration pedal and another requirement for the clutch pedal. These are requirements on the inputs of the system and are grouped together for convenience purposes.

The model below is the description of the state machine for `req4` in Modelica. It corresponds to the second state machine in Figure 2. We can see here that the requirement inherits from the partial model Requirement and the different states of the requirement are represented by the states of the state machine, e.g. when the vehicle is stopped, the requirement cannot be verified and is therefore `NOT_APPLICABLE`.

```
model VehicleSpeed
  extends Requirement;
  Modelica.Blocks.Interfaces.RealInput vx;
  Modelica.Blocks.Interfaces.RealInput vx_ref;
  inner Integer y;
  Real e_vx;

  block VehicleStopped
    outer Modelica.Blocks.Interfaces.IntegerOutput y;
  equation
    y = ReqStatus.NOT_APPLICABLE;
  end VehicleStopped;
  VehicleStopped stop;

  block NormalDriving
    outer Modelica.Blocks.Interfaces.IntegerOutput y;
  equation
    y = ReqStatus.NOT_VIOLATED;
  end NormalDriving;
  NormalDriving normal;

  block DetectedError
    outer Modelica.Blocks.Interfaces.IntegerOutput y;
  equation
    y = ReqStatus.NOT_VIOLATED;
  end DetectedError;
  DetectedError unsure;

  block PersistingError
    outer Modelica.Blocks.Interfaces.IntegerOutput y;
  equation
    y = ReqStatus.VIOLATED;
  end PersistingError;
  PersistingError error;

equation
  status = y;
  e_vx = abs(vx_ref - vx);
  initialState(stop);
  transition(stop,normal,vx >= 1);
  transition(normal,stop,vx < 1,
    immediate=false,reset=true,
    synchronize=false,priority=1);
  transition(unsure,stop,vx < 1,immediate=false);
  transition(error,stop,vx < 1,immediate=false);
  transition(normal,unsure,e_vx >= 15/3.6,
    priority=2,immediate=false);
  transition(unsure,normal,e_vx <15/3.6,
    priority=2,immediate=false);
  transition(unsure,error,timeInState() >= 30,
    priority=3,immediate=false,reset=true,
    synchronize=false);
  transition(error,normal,e_vx <15/3.6,
    priority=2,immediate=false);
end VehicleSpeed;
```

The requirement model is encapsulated in a single Modelica component (the green box in Figure 3), which is then connected with the rest of the verification setup.

As requirements are parameterizable, the same requirement can be instantiated several times in a requirement model. For instance, the requirement `WithinLimits` used to verify that an input stays within certain boundaries is used twice in `req1` both for the clutch and the accelerator pedal values.

4 Verification Model

A verification model (Schamai *et al*, 2014) contains:

- The design alternative chosen to model the system
- A requirement model
- A particular scenario used for the verification

For this case study we test the set of requirements presented in Table 1 on two different versions of a powertrain model to test how each model performs and whether it is suitable for the target application. The general setup framework is shown in Figure 3.

With this setup we can thus verify different configurations of the model against different scenarios represented by different logged data. The requirement model is in this setup not changed. In Figure 3 we can also see the parts of the physical system representing the vehicle. These models will not be changed and thus in our test the same test vehicle is used, e.g. vehicle mass is not changed between different simulation setups.

In this paper we test 2 possible design alternatives in 2 scenarios for the set of requirements defined in Section 3, resulting in the total of 4 verification models.

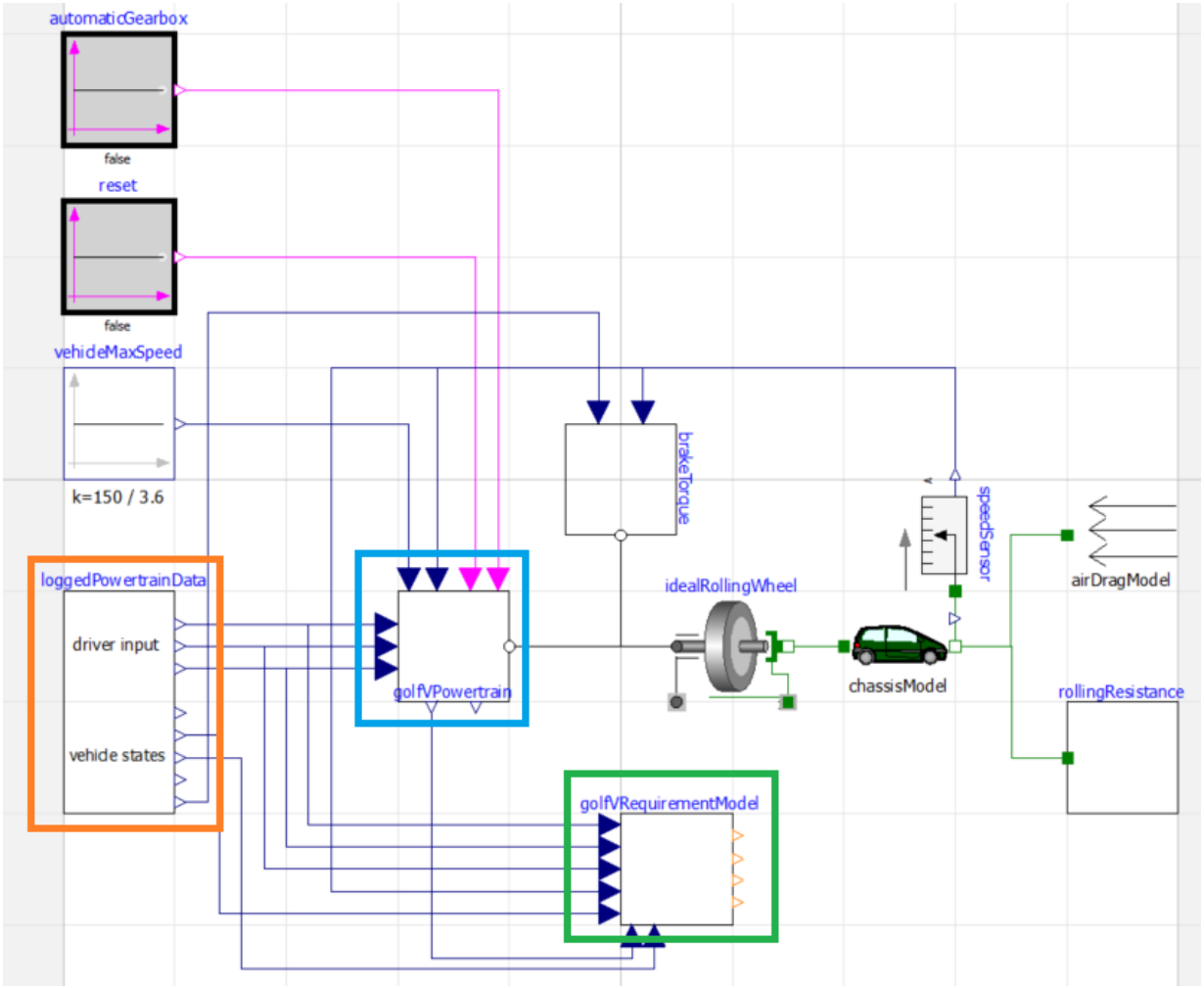

Figure 3. The complete model in OpenModelica with replaceable parts. The orange box marks the driver input which is modified depending on which data needs to be tested. The blue box is the model under test and is this is where the two different powertrain models are tested. The green box is the requirement model.

4.1 Design Alternatives

When testing different design alternatives, it is important to be able to interchange different parts of the system i.e. powertrain model easily. In Modelica a structured way to do this is to use replaceable components (Modelica Association, 2014), (Fritzson, 2014). Parts of the verification model are declared as replaceable and are instantiated with different types of components when the verification scenarios are generated.

To verify our approach we have chosen to compare two different powertrain models. These two models were created by two students and for information on the powertrain model version 1 (v1) see (Andersson *et al*, 2016) and on the powertrain model version 2 (v2) see (Zetterlund, 2015). The second model is meant to be an improved version of the first model. One of the major improvements is better performance for lower gear driving.

One of the goals in this case-study is to check whether model v2 is a more accurate model.

4.2 Verification Scenarios

To verify the powertrain system models two driving cycles were used. These are the 130 km/h variant of the Artemis Motorway Driving Cycle and the Artemis Road Driving Cycle. Data from these driving cycles are collected from the chassis dynamometers and the vehicle CAN bus in the vehicle propulsion laboratory.

Using logged data from the chassis dynamometers the powertrain models should produce the same or similar vehicle response as in the vehicle in the chassis dynamometers. Since the used driving cycle then highly influence the test of the vehicle we have chosen to use two easier tests which will test the powertrain under calm conditions.

The Artemis Motorway Driving Cycle is calm driving with one part in the middle of the test with a reduction in speed. The Artemis Road Driving Cycle is a more dynamic test and excites more dynamics in the powertrain model where for example there are three

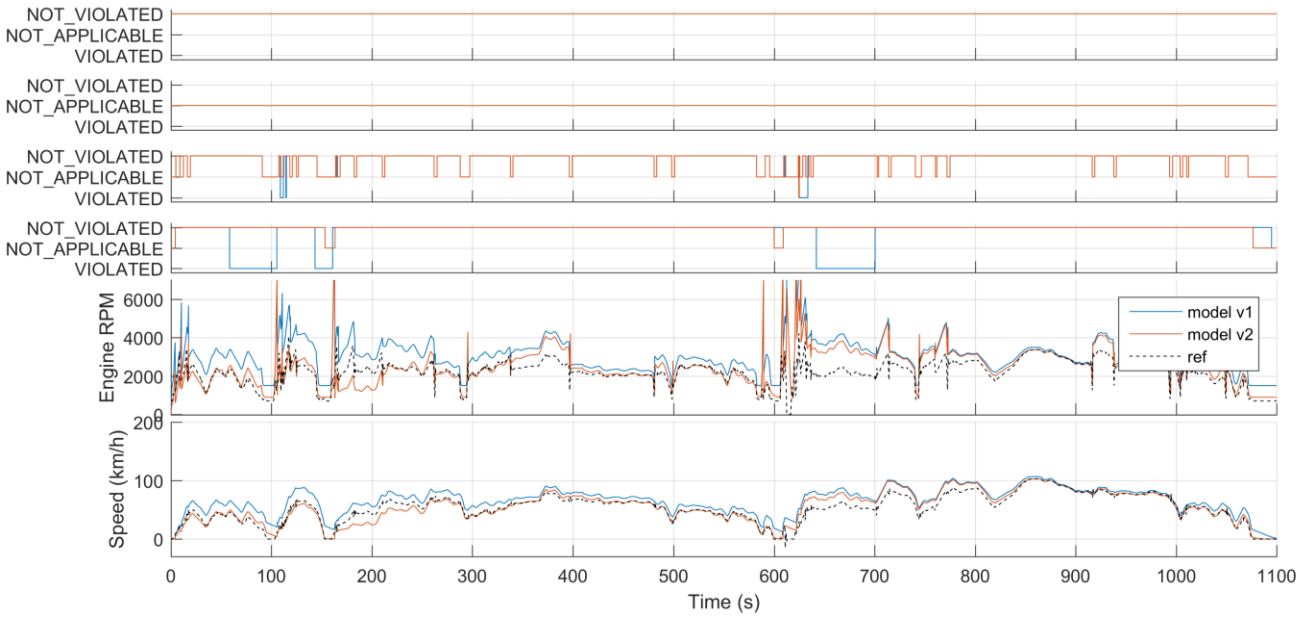

Figure 4. The powertrain model v1 and v2 during the 130 km/h variant of the Artemis Motorway Driving Cycle. The top four figures are the requirements one to four and the lower to figures show the difference in engine RPM end vehicle speed.

stops to zero vehicle speed. This means that if a powertrain model is accurate enough for a driving task when we verify our model the conclusion should be that if the requirements are passed for the Artemis Motorway Driving Cycle the model can be used for simulator studies when motorway driving is tested.

In our scenario the logged data is part of the simulation which means that a requirement can be checked during simulation and thus cancel the simulation if a requirement is violated.

Our goal is to assess whether the models can be used with the represented scenarios.

5 Simulation and Results

When running the different setups a Python interface to OpenModelica (Ganeson *et al*, 2012) is used and a Python script is used to instantiate every chosen combination of the different powertrain system models with the driver input. Each combination is associated with a unique identifier and a .mat file containing the simulation results for each combination. The data is then analyzed in Matlab to check if the requirements are violated and/or verified.

When running the simulation with a clock period of 1.0 s, we miss some events that lead to the violation of `req3`. Therefore, we use a clock period of 0.5 s.

We start by looking at the 130 km/h variant of the Artemis Motorway Driving Cycle which represents motorway driving. In Figure 4 version 1 and 2 of the powertrain system model with requirements are plotted (`req1` at the top with the other requirements consecutively downwards). From the figure we can see that the requirement on the inputs (`req1`) is never violated. This is predictable since we have been in control of the measurement equipment and has thus made sure that correct inputs have been used. The second requirement is not verified which is also expected since a manual gearbox was used during each simulation and this requirement would only be applicable for an automatic gearbox. Looking at requirements three and four (`req3` and `req4`) we see that during the time 0 to 100 when the model error is the largest for both vehicle speed and engine RPM is also when the requirements are violated for the model v1. This is illustrated more in detail in Figure 5 where the first 55 seconds is shown.

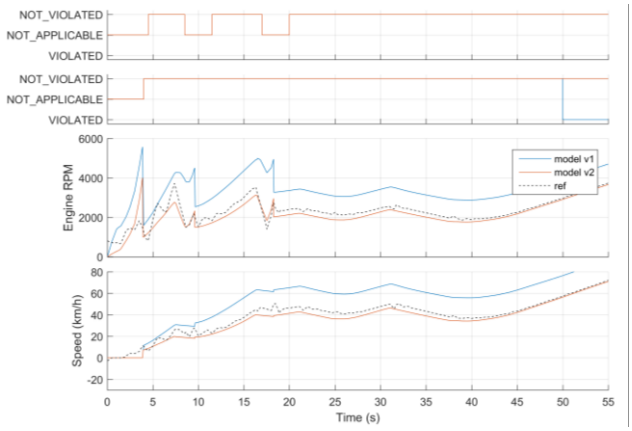

Figure 5. A close up of the first 55 seconds for models v1 and v2 during the 130 km/h variant of the Artemis Motorway Driving Cycle. Here **`req3`** and **`req4`** are shown.

In Figure 5 it can be seen that every time the clutch is pressed (at approximately the times 4, 9 and 17 seconds) an error in vehicle speed is introduced. For the system model v1 the error persists for more than 30 seconds after the third usage of the clutch and thus `req4` is

violated, see finite state machine in Figure 2. For the system model v2 it can be seen that a vehicle speed error is still present but significantly smaller. As the clutch error is below an acceptable limit `req4` is not violated. This is where model v2 is improved and thus the results are expected.

Running the simulation for our second driving cycle in the same way we get a matrix of requirements that we check for the values of `hasBeenVerified` and `hasBeenViolated`. The results are summed up in Table 2 where a green box means that the requirement has been verified and is not violated, a red box indicates that the requirement has been violated and a blue box means that the requirement has not been verified.

Table 2.

Table 2. Requirement status for the two powertrain models with two driving cycles.

		req1	req2	req3	req4
Model v1	*ArtemisMw130*				
	ArtemisRoad				
Model v2	*ArtemisMw130*				
	ArtemisRoad				

As can be seen in Table 2 the requirement set for the improved version of the model is not violated for the calmer motorway driving cycle while both models need improvement to meet the requirements for the rural road conditions. We can also see that `req1`, `req3` and `req4` have all been verified for all test cases and since automatic gearbox were never used `req2` were never verified. This result is predictable since model v2 is an improved version and should thus have better performance than model v1. From Figure 5 we can see that the difference between the models is in the region where lower gears are used. This was a region where model v1 was improved which further gives confidence that the results are correct. We also see that based on our requirement model none of the models are fit for rural road driving. This may mean that either the requirements are too strict or the model needs improvement.

Another aspect is that we do not distinguish between requirements violated early in the cycle and later on. Further inspection of the data shows that the requirements are typically violated when accelerating the vehicle from stand still to approximately 50 km/h. However, for a rural drive where the speed is above this limit most of the time this model could be used. This underlines the importance of context when looking at model assessment for a particular application. Thus a further repository with driving cycles would improve the accuracy of model verification.

Another advantage of this verification framework is that we could have several system models which doesn't violate any of the requirements for a driving task. In such a case where several models are good enough a model can be chosen based on a particular criteria not related to model accuracy. One example of such an application is models for real-time simulation where a developer can try out different numerical solvers for different system models to find the system model with good enough performance and lowest computation effort.

It should also be noted that we know that the clutch dynamics are not properly modeled, since the logged data from the CAN bus are 1 or 0. This has been taken into account in the requirements where e.g. the engine RPM is not checked when the clutch is pressed. This is also something that needs to be improved and if this is taken into account both models will violate req3. To improve the clutch model it would be good to run the vehicle while also logging intermediate clutch values to get the missing data.

6 Conclusions and Future Work

In this paper we have

- illustrated the use of requirement modeling using finite state machines in Modelica
- presented a setup for model validation of a physical system model using replaceable components to easily build different design and validation alternatives
- discussed the simulation results for different combinations of validation scenarios and used them to reason on the fitness of the chosen design alternatives for a particular purpose

In this paper we present the results at requirement level to see whether each requirement is fulfilled by a particular system design. For violated requirements however, it is not so obvious to trace the root of the problem back to the time when the requirement is violated, as the violations at time t can be due for instance to driver actions at time (t-n)., e.g. see `req4` where an error needs to persist for over 30 seconds to be `VIOLATED`. Therefore, once a violation is detected, a detailed analysis of the simulation results in the time frame surrounding the violation is still necessary by a human expert. However the semi-automation of the requirement verification process helps pinpoint the places where human intervention is necessary.

In the case-study presented in this paper, the overhead added by the simulation of the requirement model alongside the system model is considered to be negligible, however in larger-scale cases this will need to be taken into consideration. Therefore the next step is to take the overhead in account in the verification process.

Acknowledgments

This work is partially supported by the EU INTO-CPS project and the ITEA MODRIO and OPENCPS projects via the Swedish Government (Vinnova) and the German and French Government.

References

Anders Andersson, Sogol Kharrazi, Simon Lind, and Andreas Myklebust. Parameterization procedure of a powertrain model for a driving simulator. *Advances in Transportation Studies*, 2016, 1.

Michel Andre, The ARTEMIS European driving cycles for measuring car pollutant emissions. *Science of the Total Environment* 334– 335, pages 73–84, 2004.

Modelica Association. *Modelica 3.3 revision 1 specification*. 2014. URL www.modelica.org.

Peter Fritzson. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. 1250 pages. ISBN 9781-118-859124, Wiley IEEE Press, 2014.

Anand Ganeson, Peter Fritzson, Olena Rogovchenko, Adeel Asghar, Martin Sjölund, and Andreas Pfeiffer. An OpenModelica Python interface and its use in pysimulator. In Martin Otter and Dirk Zimmer, editors, *Proceedings of the 9th International Modelica Conference*. Linköping University Electronic Press, September 2012.

Per Öberg, Peter Nyberg, and Lars Nielsen. A new chassis dynamometer laboratory for vehicle research. *SAE International Journal of Passenger Cars- Electronic and Electrical Systems*, 2013, 6(1):152–161.

Martin Otter, Nguyen Thuy, Daniel Bouskela, Lena Buffoni, Hilding Elmqvist, Peter Fritzson, Alfredo Garro, Audrey Jardin, Hans Olsson, Maxime Payelleville, Wladimir Schamai, Eric Thomas, and Andrea Tundis. Formal requirements modeling for simulation-based verification. In Peter Fritzson and Hilding Elmqvist, editors, *Proceedings of the 11th International Modelica Conference*. Modelica Association and Linköping University Electronic Press, September 2015.

Wladimir Schamai, Lena Buffoni, and Peter Fritzson, An Approach to Automated Model Composition Illustrated in the Context of Design Verification. *Journal of Modeling, Identification and Control*, Volume 35- 2, pages 79—91, 2014.

Bernhard Thiele, Adrian Pop, and Peter Fritzson. Flattening of modelica state machines: A practical symbolic representation. In Peter Fritzson and Hilding Elmqvist, editors, *Proceedings of the 11th International Modelica Conference*. Modelica Association and Linköping University Electronic Press, September 2015.

Olof Zetterlund. Optimization of Vehicle Powertrain Model Complexity for Different Driving Tasks. *Master's thesis*, Linköping University, LiTH-ISY-EX–15/4897–SE, 2015.