Coordinated automotive libraries for vehicle system modelling

Mike Dempsey[1], Magnus Gäfvert[2], Peter Harman[3],
Christian Kral[4], Martin Otter[5], Peter Treffinger[6]

[1] Claytex Services Limited, Hatton, UK [4] arsenal research, Vienna, Austria
[2] Modelon AB, Lund, Sweden [5] DLR, Oberpfaffenhofen, Germany
[3] Ricardo UK Limited, Leamington Spa, UK [6] DLR, Stuttgart, Germany
vi@claytex.com

Abstract

A new free Modelica library, called VehicleInterfaces, has been developed to promote compatibility between Modelica automotive libraries. The library provides standard system interface definitions that enable the whole vehicle system to be conveniently modelled. Example vehicle models are also provided to illustrate the use of the interface definitions. Practical applications and examples of how these interface definitions can be used are presented.

1 Motivation

A number of Modelica library developers are working independently on automotive libraries focused on different vehicle systems such as PowerTrain [1], Transmission [2], VehicleDynamics [3] and SmartElectricDrives [4]. For many simulation activities it is desirable to be able to create whole system models [5] that combine elements from the different libraries and provide easy ways to integrate user-developed models.

This work is based on previous work carried out by the authors and also builds on the work done on the Modelica Vehicle Model Architecture published by Tiller et. al.[6].

The approach adopted in developing this library has been to focus on standardising the subsystem interfaces rather than developing a standard vehicle model architecture. Several different example architectures based on these interfaces are provided as examples in the VehicleInterfaces library. A typical example is shown in Figure 1 where all the main subsystems (driver, driverEnvironment, engine, transmission, driveline, chassis, brakes, accessories, road, atmosphere, etc.) are included at the top level of the model. In this example it is assumed that the controllers for the respective subsystems are part of the subsystem model. In other architectures, the controllers might be on the same level as the subsystems.

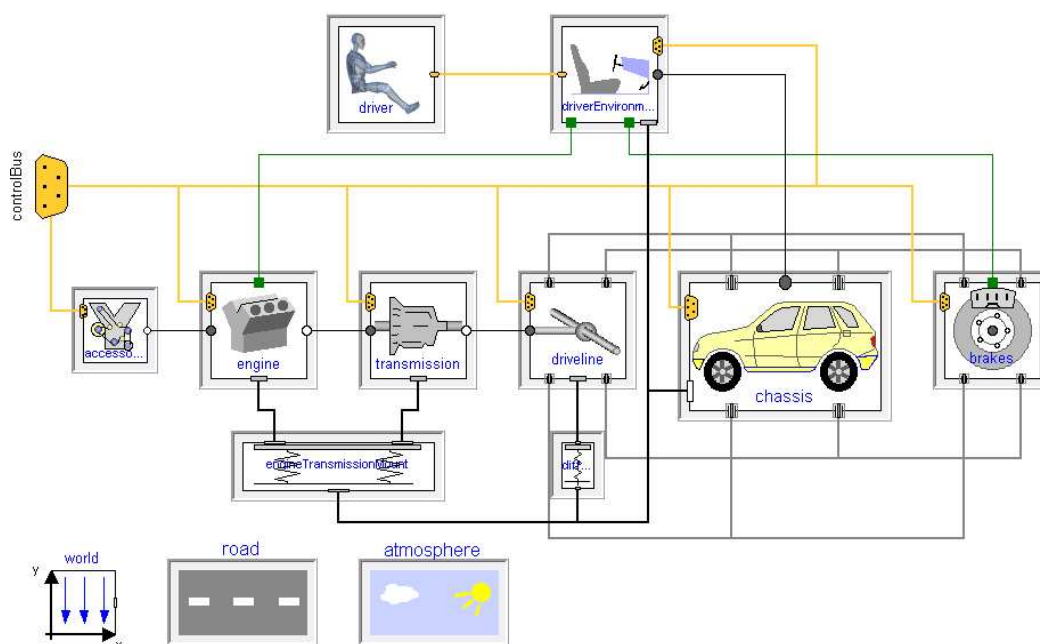All subsystem models in the architecture are de-

Figure 1: Example vehicle model using VehicleInterfaces shown in Dymola 6.0b

clared as "replaceable". When instanting an architecture model, the desired subsystem model can be selected via a redeclaration. In Dymola there are two ways to redeclare components using the graphical user interface as shown in Figure 2.

The dialog box shown at the top of Figure 2 is accessed through the model browser by right-clicking on the base class being extended. The scroll down menus show the available subsystem models that can be selected. As usual, these menus are automatically constructed via the annotation "choicesAllMatching = true" (all model components are shown that are loaded into the Modelica simulation environment and are derived by inheritance from the respective superclass).

In Dymola 6 redeclarations can be made by right clicking on a replaceable subsystem, the context menu then allows the user to select an option from the list of matching types. The list is determined

Figure 2: Two methods for the redeclaration of subsystem models in Dymola 6

using the same annotation as the dialog box method.

2 Interface Definitions

The subsystem decomposition used within the VehicleInterfaces package follows the same basic structure as used in the Modelica VMA developed by Tiller et al [6]. In the following sections we will discuss the subsystems and modelling methods introduced with the VehicleInterfaces package.

2.1 Conditional Connectors

The VehicleInterfaces package makes extensive use of conditional connectors so that a single subsystem interface definition can be used for a wide range of applications. A conditional component, such as a connector, is one that is only instantiated in the model if the corresponding Boolean condition is true. The Modelica code for an example conditional component is shown in Figure 3. In this example the component **shaft** is only instantiated in the model if the user sets the parameter **includeInertia**=true, if the parameter is false then the component and all related connections are completely removed from the model.

If we consider the engine subsystem, this includes conditional connectors for the engineMount and acceleratorPedal connections. The engineMount connector models the physical connection between the engine block and the engine mounting system as a MultiBody connection. This is not always required, for example when modelling the engine as a simple 1D system. Similarly the acceleratorPedal, which is a 1D translational connector provides a physical connection between the engine subsystem and the driver environment, which isn't required in a drive-by-wire vehicle.

By using conditional connectors these components are completely eliminated from a model when they are not required. If they were not eliminated it would be necessary to add additional components to ensure that all the flow variables within the connectors were being properly defined. This would add additional overhead to the simulation task and should
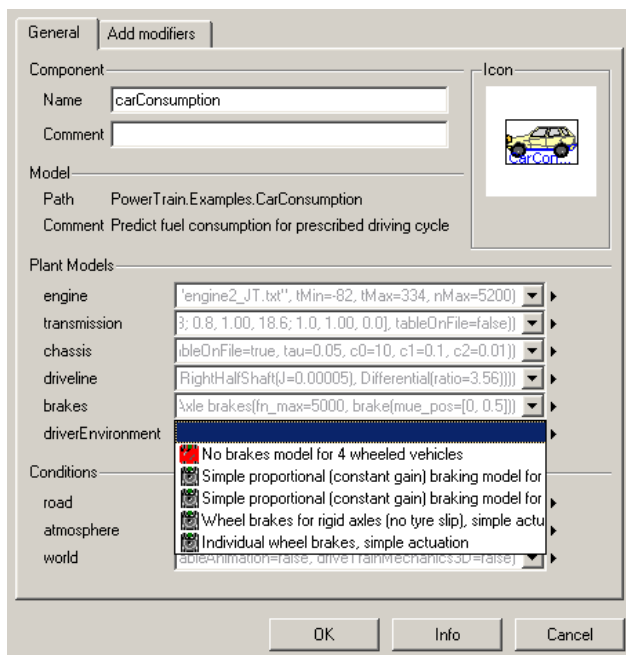
```
model Example
  parameter Boolean includeInertia=false;
  Rotational.Inertia shaft if includeInertia;
  ...
end Example;
```

Figure 3: Example of a conditionally instantiated Inertia component.

be avoided whenever possible.

Conditional connectors are used wherever a connector might not be needed in every application. These include all MultiBody connectors and all the physical connections between the driver environment and the subsystem models.

2.2 Modelling Rotating Components

Within VehicleInterfaces rather than limiting ourselves to modelling rotating components as a simple 1D rotation we have built in the flexibility to model rotating components as MultiBody systems. This has been possible through the development of a new connector called FlangeWithBearing. This is a hierarchical connector that contains a 1D rotational connector and a conditional MultiBody connector. The use of the MultiBody connector is controlled by a parameter in the connector. This enables the FlangeWithBearing connector to model rotational effects as a purely 1D system or it can correctly include MultiBody effects. The Modelica definition of this new connector is shown in Figure 4 and the connector is now available within the MultiBody.Interfaces package of the Modelica Standard Library 2.2.1.

Where this connector is used in an interface definition the parameter **includeBearingConnector** is linked to a parameter at the model level so that the model developer can easily activate this connector if required. An example of how this is used is shown in Figure 5 where the driveline interface definitions for a 2-axle vehicle are shown. In the Base class we can see that three Boolean parameters are declared as protected parameters. This means they are only available to the model developer as they create the subsystem model and cannot be changed when the model is used.

The use of the bearing connectors needs to carefully considered to avoid inadvertently creating mechanical loops in the model. The following example guidelines for the engine and transmission subsystems should help highlight the issues so that the

```
connector FlangeWithBearing
  parameter Boolean
    includeBearingConnector=false;
  Rotational.Interfaces.Flange_a flange;
  MultiBody.Interfaces.Frame bearingFrame
    if IncludeBearingConnector;
end FlangeWithBearing;
```
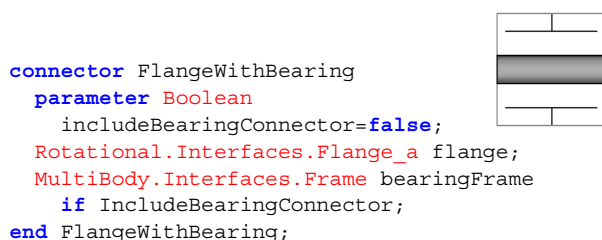
Figure 4: Modelica code and icon for
the FlangeWithBearing connector

```
model Base
  MultiBody.Interfaces.FlangeWithBearing
    TransmissionFlange(
      final includeBearingConnector=
      includeTransmissionBearing);
  VehicleInterfaces.Interfaces.ControlBus
    ControlBus
  MultiBody.Interfaces.Frame_a drivelineMount
    if includeMount;
protected
  parameter Boolean
          includeWheelBearings=false;
  parameter Boolean includeMount=false;
  parameter Boolean
          includeTransmissionBearing=false;
end Base;

model TwoAxleBase
  extends Base;
  MultiBody.Interfaces.FlangeWithBearing
    WheelHub_1(
      final includeBearingConnector=
      includeWheelBearings);
  MultiBody.Interfaces.FlangeWithBearing
    WheelHub_2(
      final includeBearingConnector=
      includeWheelBearings);
  MultiBody.Interfaces.FlangeWithBearing
    WheelHub_3(
      final includeBearingConnector=
      includeWheelBearings);
  MultiBody.Interfaces.FlangeWithBearing
    WheelHub_4(
      final includeBearingConnector=
      includeWheelBearings);
end TwoAxleBase;
```

Figure 5: Modelica code for the
driveline interface definition

model developer can determine when it is appropriate to use the bearing connectors. For the engine and transmission subsystem models:

1. When they are modelled as a pure 1D rotational system then no bearing connectors are required.

2. When they are modelled as a 1D rotational system with reactions on to a MultiBody system then no bearing connectors are required

3. When they are being modelled as a MultiBody system but they are rigidly connected together then the bearing frame between the engine and transmission should not be included. In this case the transmissionMount connector should support the MultiBody elements of the transmission. The rest of the model then needs to be considered before deciding whether to include the bearing between the transmission and driveline or between the engine and accessories subsystems.

4. When they are being modelled as a MultiBody system but they are **not** rigidly connected to-

gether then the bearing frame between the engine and transmission will be required to support the intermediate drive shaft.

2.3 Driver and Driver Environment Subsystems

The driver and the physical interaction between driver and vehicle can be modelled as separate subsystems or combined as a single subsystem within the VehicleInterfaces package. The decision is down to the model developers and will influence the extent to which models are reusable in other applications. The physical interactions, such as steering and throttle controls and feedback signals, are referred to as the driver environment within VehicleInterfaces.

When the driver and driver environment are modelled as separate subsystems the driver environment model is responsible for converting the normalised instructions passed from the driver model in to the correct values for this particular vehicle model. For example, a driver model should demand normalised steering wheel angles between –1 and 1 and this should be translated by the driver environment subsystem in to the appropriate steering wheel angle for the current vehicle. This enables the driver model to be defined in a generic way for use on many different vehicles.

The interaction between the driver and driver environment subsystem is modelled using an expandable connector known as the driver interaction bus. The connections passed across this bus are a combination of signal values and normalised physical connections. A normalised physical connection contains both the normalised position and the actual force or torque being applied across the connection. The naming and types for the signals that are exchanged between these two subsystems is defined within the VehicleInterfaces package to ensure compatibility between driver models and driver environment models from different libraries.

When only the driver environment subsystem is present this should also include the driver model and it is the responsibility of the individual model developers to provide a logical separation between the environment and driver models.

2.4 Powerplant Mounts Subsystem

Unlike in the Modelica VMA the Powertrain mounting systems are modelled as separate subsystems when they are required in a model. In Figure 1 we see that there are two separate mounting systems, one for supporting the engine and transmission and another that supports the differential. The modelling of the mounting systems in this way reflects the physical reality of a rear-wheel drive vehicle in which the engine and transmission are rigidly connected together and mounted as one system at the front of the vehicle and the differential in the rear axle is independently mounted. Vehicles with different driveline configurations would require a different arrangement for the mounting systems.

The mounting subsystems are all defined by extending a base class that includes a MultiBody connector that should be connected to the vehicle body. There are 3 mounting subsystem templates provided within the VehicleInterfaces package that can be used to support differing numbers of powertrain subsystems. The connections to the powertrain subsystems are modelled using MultiBody connectors.

When the mounting systems are not being modelled these subsystems can be removed from the model architecture to simplify the vehicle model.

2.5 Road Subsystem

The road subsystem is used to define the road surface and supports varying friction coefficients, curvature, gradients and banking. The road is defined as a series of replaceable functions that are used to determine the position along the road, the normal to the road surface, the current heading of the road centre line and the friction coefficient. By redeclaration of these functions a wide range of road models from a straight flat road through to a curved undulating road can be created.

When a road is used at the top level of a model it should be declared with the prefix inner so that it can be referenced from any subsystem or component within the model that needs to determine information about the road surface. When a subsystem needs to refer to the road subsystem it should contain an outer version of the road subsystem and this will then enable it to access the road definition from the top-level of the model.

2.6 Atmosphere Subsystem

The atmosphere subsystem defines the ambient conditions including temperature, pressure, humidity, wind speed and direction. The atmosphere is defined as a series of replaceable functions that determine these conditions at a specified point in space. This enables the ambient conditions to vary with the vehicle position so effects such as wind can be varied as the vehicle drives along a track.

2.7 Example Vehicle Architectures

Using these interface definitions we can create a variety of vehicle model architectures to suit different applications. Figure 1 shows an example architecture for a rear-wheel drive automatic transmission passenger car that includes a separate driver model and powertrain mounting systems. Figure 6 shows some other possible model architectures including (from top to bottom) a manual transmission vehicle; an alternative layout for an automatic transmission vehicle; a power-split hybrid vehicle model. In all these cases it is possible to re-use the same subsystem models because the interface definitions are consistent even though the top-level model appears very different.

3 Control Bus Structure

3.1 Overview

Within the VehicleInterfaces library every subsystem that forms part of the vehicle model has a connection to the control bus. The control bus is used to pass information between the subsystems that would normally be passed along the CAN bus or similar vehicle communication network. The VehicleInterfaces control bus does not model how the vehicle network communication actually works but instead provides a structure by which the same information can be exchanged between the various subsystems.

The control bus is modelled using a series of hierarchical expandable connectors, which means that the user can place any signal they need on to the control bus. As part of the VehicleInterfaces library a minimum set of signals and a structure for the control bus is recommended so that systems that follow these recommendations can easily be coupled together.

A hierarchical structure to the control bus is proposed where the subsystem name is used to help structure the signals on the bus. For example signals placed on to the control bus from the chassis subsystem should be placed within the chassisBus structure of the controlBus, see Figure 7 for an illustration of the current minimum set of signals for the control bus. A full naming convention is included with the VehicleInterfaces package.

3.2 Working with the Control Bus

Every subsystem within the VehicleInterfaces package contains a controlBus connector that will allow the subsystem access to the complete control bus
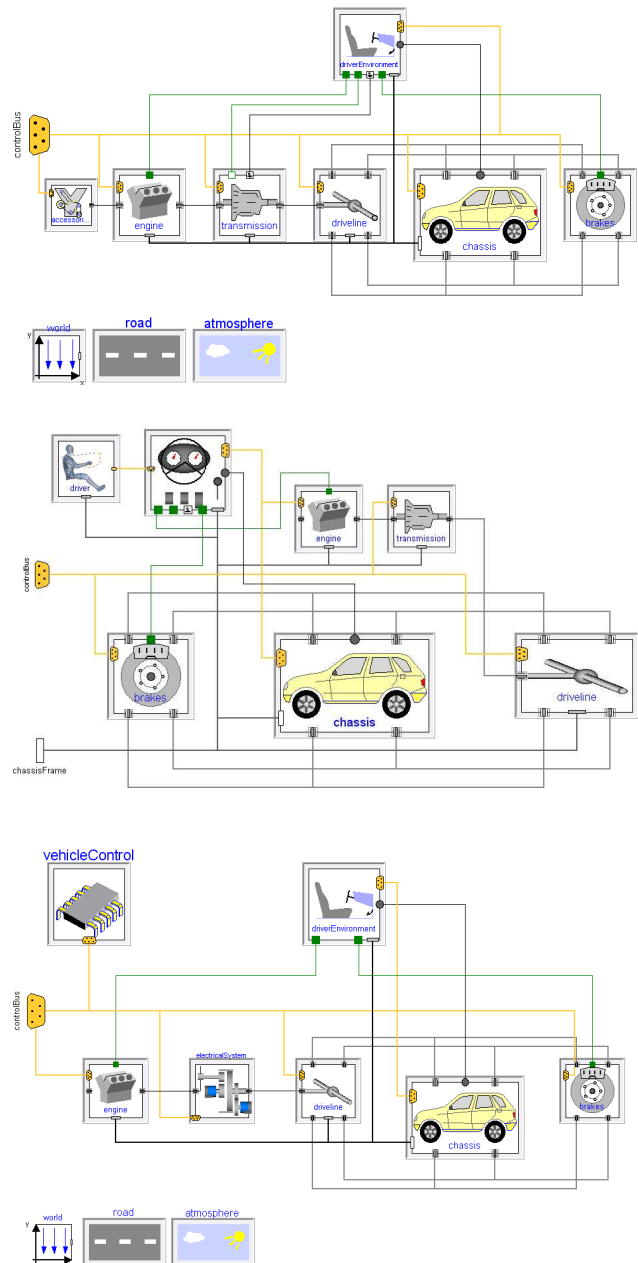

Figure 6: Example model architectures

structure. To access a signal within the control bus hierarchy it is first necessary to add the appropriate sub-bus connector to the model as a node, i.e. a protected connector. Signals within this part of the control hierarchy can then be accessed by connecting to the sub-bus connector. Figure 8 shows how the longitudinal velocity signal within the chassisBus sub-bus on the vehicle controlBus can be accessed. The Modelica code for this example is also shown. This methodology is necessary due to the way the Modelica language specification defines expandable connectors [7].
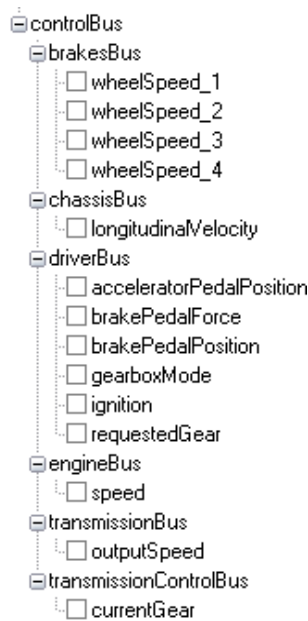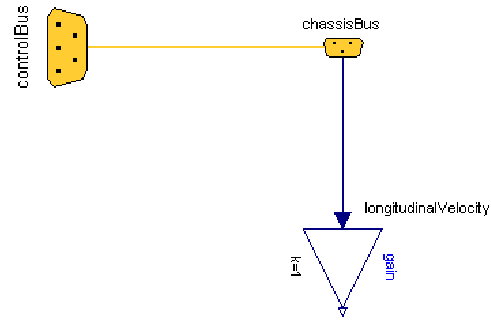
Figure 7: Current minimum set of
signals in the control bus


```
model controlBusDemo
  extends TwoAxleBase;
protected
  VehicleInterfaces.Interfaces.ChassisBus
    chassisBus;
public
  Modelica.Blocks.Math.Gain gain;
equation
  connect(controlBus.chassisBus,
          chassisBus);
  connect(gain.u,
          chassisBus.longitudinalVelocity);
end controlBusDemo;
```

Figure 8: Accessing signals on the control
bus (model diagram in Dymola)

3.3 The Sub-bus Connectors

The sub-bus connectors are defined within the VehicleInterfaces.Interfaces package and are all defined as expandable connectors. As such the connectors don't contain any signal names and yet we would like it to be possible to generate a list of pre-defined names to make it easier for the user to connect to the control bus and access the appropriate signal.

To enable this to happen, within the VehicleInterfaces package each expandable connector has been extended and the standard signals have been defined within these extended connectors. These connectors are not intended for use directly within a model but are necessary to enable a Modelica tool to generate a list of possible signal names. Within Dymola, when a connection to an expandable connector is made a dialog box is generated with a list of signal names. The list of signal names is now determined by searching through the open libraries to find connectors that extend from the type of expandable connector used in the current connection. All the signals that are defined within the connectors that extend from the base connector are then added to the list and the user can select the appropriate one.

This functionality means that the user can easily connect to one of the standard signal names but also means that it is not absolutely necessary for them to assign values to every signal that is defined as part of the standard VehicleInterfaces control bus. It also allows different library developers to extend the con-

trol signal bus in appropriate ways for their library and to have the names automatically appear in models.

4 Usage Examples

Three different usage examples are presented to illustrate different ways that the interface templates can be used to model vehicles with different levels of detail. The first two examples illustrate two different approaches to modelling a rear-wheel drive driveline and the third example illustrates how the different commercial model libraries that are adopting these interface definitions can be coupled together.

4.1 Rear-Wheel-Drive Vehicle as a 1D System

The simplest way to model a vehicle powertrain is as a 1D rotational system. This approach does have many uses, such as fuel economy studies, and this example illustrates how the interface templates can be used in this way. Figure 9 shows the driveline model diagram for a rear-wheel drive vehicle modelled as a purely 1D rotational system. Components in this model are taken from the Modelica Standard Library and the PowerTrain library.

In this example the parameters that control the conditional connectors for the driveline interface class are all left with their default values of false. This means
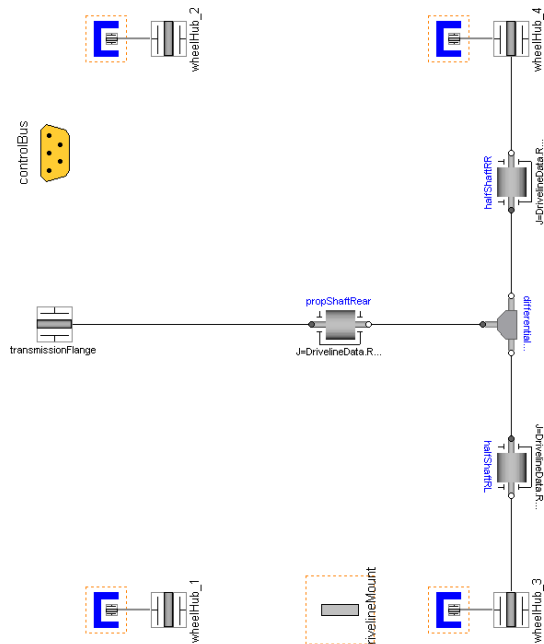
Figure 9: Simple 1D Rotational model
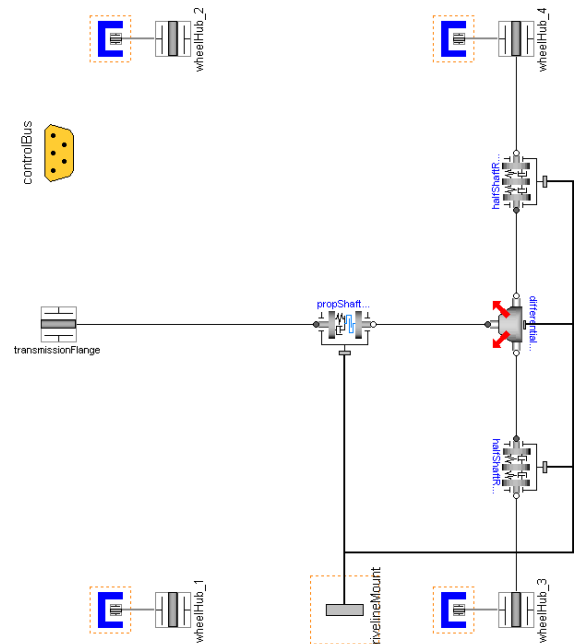of a rear-wheel drive driveline

Figure 10: 1D Rotational model of a rear-wheel
drive driveline with reactions on to a MultiBody
system

that the bearing frame connectors within the FlangeWithBearing connectors (transmissionFlange, wheelHub_1, etc.) are not instantiated in the model and neither is the drivelineMount connector. This leaves us with a simple model using just the 1D rotational connectors for the transmission and wheel hubs.

When modelling a 1D rotational system it is sometimes necessary to include the reactions of the 1D rotational system on a MultiBody system [8]. Adapting the rear-wheel-drive model to include MultiBody effects would lead to the diagram in Figure 10. To enable this model to be built the drivelineMount connector needs to be enabled so that the MultiBody reactions can be transmitted in to the vehicle body. The bearing frame connectors within the transmissionFlange and wheelHub connectors are not required in this model as the driveline is not being modelled as a MultiBody system.

4.2 Rear-Wheel-Drive Vehicle as a MultiBody System

The same driveline interface template can be used to model the complete driveline as a MultiBody system. In this case the use of the bearing connectors within the FlangeWithBearing connectors needs to be thought about carefully in order to make sure mechanical loops aren't inadvertently created. Consideration needs to be given to the way in which the MultiBody components are being supported both in the physical system and in the model itself.

In the case of a rear-wheel drive vehicle the propshaft is supported by the transmission and the differential. So in this case the bearing frame in the transmissionFlange needs to be included so that this end of the propshaft is correctly supported. The differential is also being modelled as a MultiBody system so this will support the other end of the propshaft. The differential itself is typically supported by an elastic mounting system that would be connected
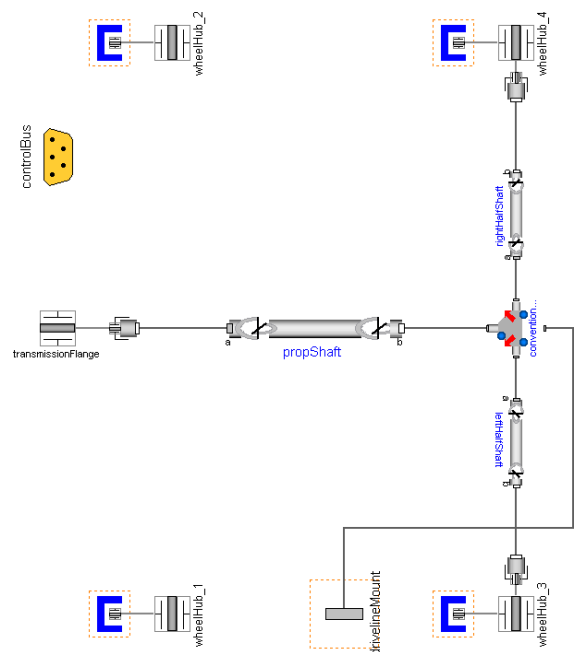

Figure 11: MultiBody model of
a rear-wheel drive driveline

to the driveline model via the drivelineMount connector.

Finally it needs to be considered how the halfshafts are supported, one end is attached to the differential and supported by the output bearings of the differential, the other end is attached to the wheel hub and supported by the wheel bearing. This means the bearing frames in the wheel hub connectors need to be included. An example of how this subsystem might look is shown in Figure 11.

4.3 Active 4WD Vehicle Model

By combining models from the PowerTrain and VehicleDynamics libraries it is possible to study the handling benefits of active four-wheel-drive systems and compare this to the handling of the same vehicle with a conventional, passive four-wheel drive system.

The vehicle model is created using various subsystem models from the PowerTrain library and the VehicleDynamics library. The PowerTrain library contains an active four-wheel drive system model shown in Figure 12. The driveline is modelled as a 1D rotational system and includes the reactions on to the vehicle body. To make this 1D driveline model compatible with a MultiBody chassis model from the VehicleDynamics library we need to activate the flag **usingMultiBodyChassis** in the "Advanced" menu of the driveline component parameter dialog. When this

parameter is set to true the bearing connectors within the wheelHub connectors are included and zero forces and torques are applied to these bearing connectors.

The driveline control system model within the PowerTrain library provides parameters to enable or disable the control of the active differentials. When disabled the driveline behaves as a conventional, passive four-wheel-drive system so this model can easily be used to assess the benefits of active versus passive four-wheel-drive.

The vehicles are tested by accelerating from rest to 100kmh and then negotiating a tight chicane at 100kmh whilst trying to maintain this speed. Figure 13 shows how the yaw rate, steering angle and longitudinal speed of the two cars varies during the test. As a chassis model from the VehicleDynamics library is being used the behaviour of the cars during the test can be animated, Figure 14 shows a comparison of how the two cars behave.

5 Outlook

The first version (1.0) of the VehicleInterfaces library has been presented. Future developments and refinements will be based on feedback from automotive library developers and users of the VehicleInter-
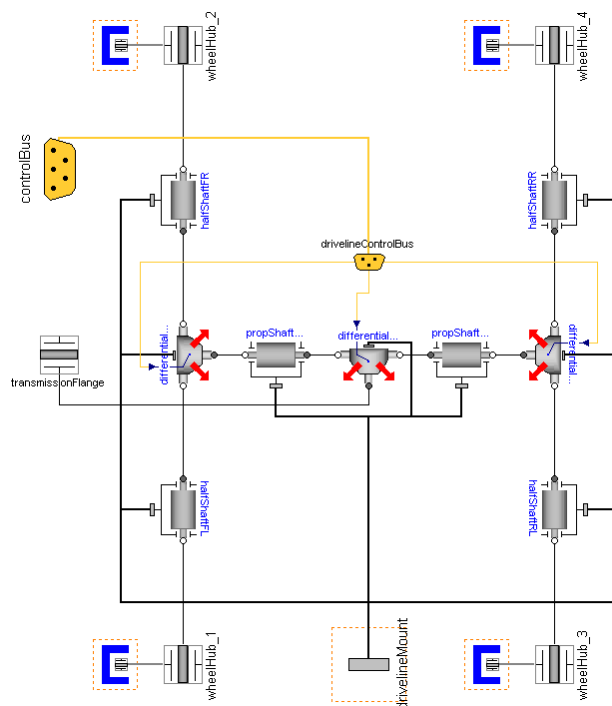

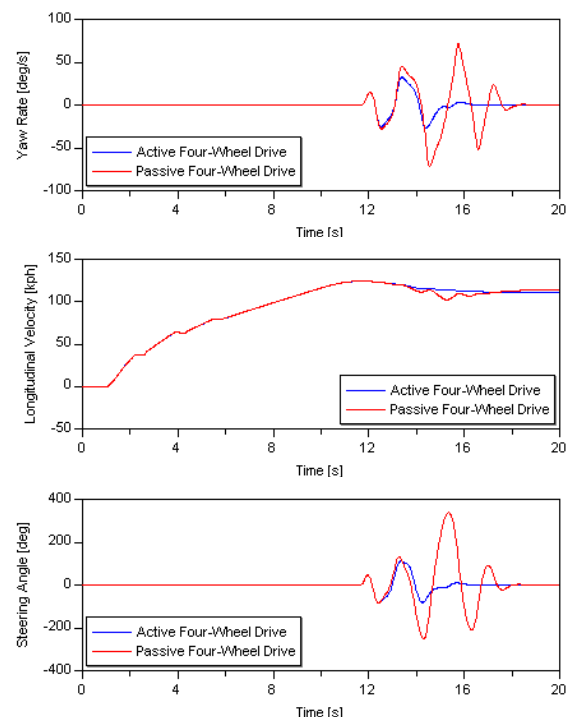Figure 12: Active four-wheel drive system from the PowerTrain library

Figure 13: Comparing Active and Passive Four-Wheel Drive. Yaw-rate (top), Longitudinal velocity (middle) and Steering angle (bottom).
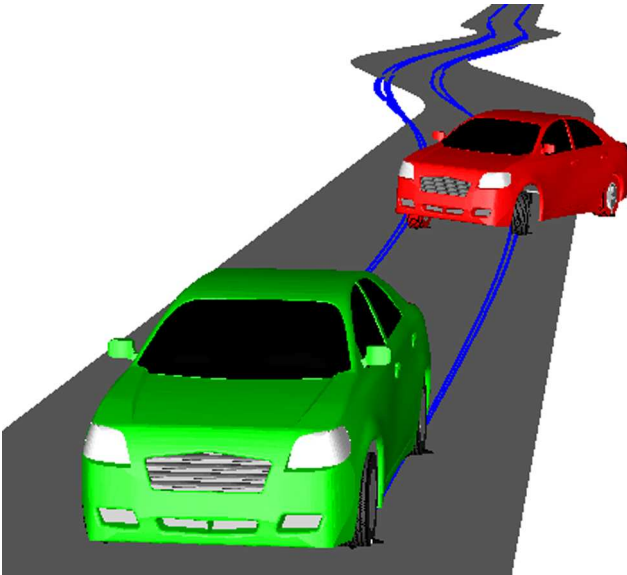
Figure 14: Visualising the behaviour of the two cars in Dymola. The green car has active four-wheel drive and the red car has passive four-wheel drive.

faces library. Currently only a small set of standardised signals have been defined on the control bus and it is likely that this will need to be extended significantly to meet the needs of users.

6 Acknowledgements

References

[1] Schweiger C., Dempsey M., Otter M.: *The Power-Train Library: New Concepts and New Fields of Applications*. Proceedings of Modelica 2005 Conference. http://www.modelica.org/events/Conference2005/online_proceedings/Session6/Session6a1.pdf

[2] Brandao F., Harman P.: *An Integrated Simulation Approach: Ricardo Transmission and Driveline Dynamic Simulation Library*. IMechE Integrated Powertrain and Driveline Systems 2006

[3] Andreasson J., Gäfvert M.: *Vehicle Dynamics Library*. Proceedings of Modelica 2006 Conference.

[4] Giuliani H., Kral C., Gragger J.V, Pirker F.;. *Modelica Simulation of Electric Drives for Vehicular Applications – The Smart Drives Library*. ASIM conference, 2005

[5] Alexander T., Liu C.S., Monkaba V.: *Multi-Body Dynamic Modeling Methods and Applications for Driveline Systems*. SAE 2002-01-1195

[6] Tiller M., Bowles P., Dempsey M.: *Development of a Vehicle Modeling Architecture in Modelica*. Proceedings of the Modelica 2003 Conference. http://www.modelica.org/events/Conference2003/papers/h32_vehicle_Tiller.pdf

[7] Modelica: Language Specification 2.2. Feb. 2005. Section 3.3.8, pp. 54 – 59 (expandable connectors). http://www.modelica.org/documents/ModelicaSpec22.pdf

[8] Schweiger C., Otter M.: *Modelling 3D Mechanical Effects of 1D Powertrains*. Proceedings of Modelica 2003 Conference,Nov. 2003. http://www.modelica.org/events/Conference2003/papers/h06_Schweiger_powertrains_v5.pdf