

Development of a Vehicle Model Architecture in Modelica

Michael Tiller[†]

Paul Bowles[†]

Mike Dempsey[‡]

[†]Ford Motor Company, Powertrain Research Department

[‡]Claytex Services Limited

ABSTRACT

The real power and flexibility that comes from using Modelica for physical modeling stems from the combination of the acausal approach to formulating physical connections combined with sets of standard connector definitions in various engineering domains. These features are important because they help avoid *a priori* causality assumptions (which promotes reuse of components) and ensure physical compatibility across connections. However, complex systems are generally made up of several complex, multi-domain subsystems with numerous connectors. Such systems also benefit from having standardized subsystem interface definitions. This paper will focus on an initial proposal for a vehicle model architecture for vehicle system applications. Ultimately, we hope that feedback on this proposal from other groups doing vehicle modeling will lead to a consensus on the appropriate subsystem interfaces such that we can achieve the same level of flexibility and reusability for vehicle subsystem models that we currently have with component level models.

1 Motivation

Vehicle system modeling is an important part of optimizing overall vehicle performance. To avoid building up complete vehicle models from scratch repeatedly, it is useful to develop a pre-wired vehicle model architecture. We had two goals in mind when formulating such a vehicle model architecture. First, it should allow the exchange of subsystem models between different organizations (e.g. part/subsystem vendors, design organizations, universities) without the need to "rework" the models to fit into existing vehicle system models. Second, it should greatly simplify the handling of alternative vehicle system configurations by allowing substitution of one particular subsystem or strategy implementation for another.

Ideally, we hope that this architecture will develop to the point that other groups, outside of Ford, will adopt it. Given the growing number of automotive related libraries in Modelica [1-4], both freely available and commercial, such a vehicle model architecture will be a practical necessity to allow subsystem models from these libraries to be easily assembled into complete vehicle models.

Previous efforts at Ford have focused on providing a vehicle model architecture for models developed in Simulink [5]. While not disputing the value of a corporate standard for vehicle subsystem models, groups working with Modelica were not willing to give up the acausal flexibility in Modelica for an approach that required *a priori* causality assumptions. Furthermore, most existing vehicle level modeling applications using Modelica at Ford involved details (e.g. modeling the motion of the powertrain mounts) that were not possible with the Simulink framework.

As a result of internal discussions, it was agreed that an acceptable compromise would be to develop a purely Modelica architecture using essentially the same subsystem decomposition, as was done in Simulink, but avoiding *a priori* causality assumptions. In cases where Modelica models would be useful to someone working in Simulink, we hope to develop a set of standard "wrappers" for each subsystem that will allow us to impose the required causality on an otherwise acausal subsystem model and then convert these into an S-function using Dymola [6].

2 Architecture Structure

A complete vehicle system model must take into account the response of the various physical subsystems, the function of the controller modules (both subsystem and vehicle level) as well as other "external" influences like the environment and the driver. The following sections will discuss the decomposition in each of these categories.

2.1 Physical Subsystems

The first category we will be discussing includes all the physical subsystems in the vehicle. This section will provide some discussion for each physical subsystem and some explanation of what is contained within each subsystem. The order of the subsystems corresponds, roughly, to the order that they appear (from left to right) in Figure 1.

Note that each physical subsystem is connected to a subsystem controller. We will defer the discussion of this connection until Section 2.2.3 and instead focus, for now, on the physical connections associated with each subsystem.

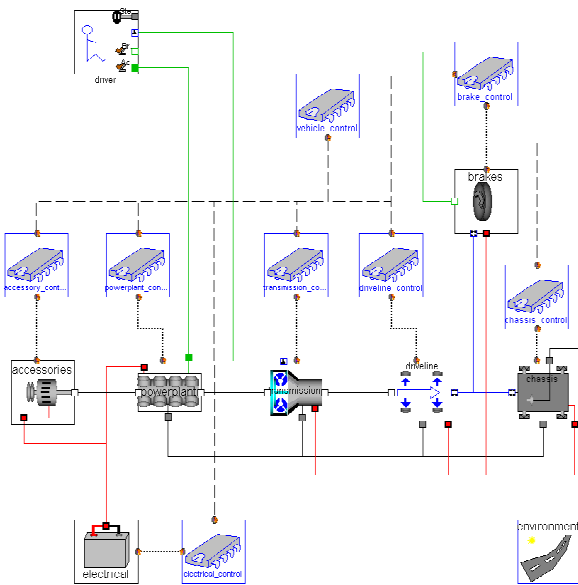


Figure 1: Vehicle Model Architecture

2.1.1 Accessories

The accessory subsystem is composed of those components typically connected to the front end accessory drive (FEAD) of an engine. Examples of such components would include an alternator or AC compressor. As shown in Figure 1, the accessories are connected to the front side of the powerplant. As a result, any torque required by these components will be taken from the powerplant. The accessories are also connected to the electrical subsystem and they typically represent a significant influence on the charging and discharging of the electrical system.

2.1.2 Electrical

The electrical subsystem is composed of the various purely electrical components in the vehicle. Typical examples would include the battery, radio and/or headlights. In addition to being the location

for all purely electrical components, the electrical system is also the source of electrical power for every other physical subsystem in the vehicle and, as such, is subject to "external" influences that may charge or deplete the battery (e.g. alternator, regenerative braking).

2.1.3 Powerplant

The powerplant subsystem represents the primary source of motive torque for the vehicle. Typically, this would be an internal combustion engine although it could also be, for example, an electric motor. Like the battery, the powerplant model provides power to the rest of the vehicle. As such, there are physical connections from the powerplant to the accessories and the transmission.

The powerplant is also connected to the electrical subsystem. Although the electrical influence of an internal combustion engine is normally quite small (e.g. spark plug energy, etc), if the powerplant were an electric motor, the connection to the electrical system would become quite important. In the case of hybrid electric vehicles, additional electrical components, such as electric motors, may be included in the powerplant or they may be lumped into the transmission (depending on the powertrain topology).

The physical connection between the driver and the powerplant includes a signal representing the physical position of the accelerator pedal. Typically, this signal is translated directly into a throttle position. However, in "drive by wire" applications, it is assumed that the pedal position sensor would be associated with the powerplant subsystem and that sensor information would be relayed to the powerplant subsystem controller and/or vehicle controller where, for example, the commanded throttle position (or motive torque, in the case of an electric vehicle) would be calculated and returned as an actuator command.

Finally, Figure 1 shows that the powerplant has a third mechanical connection. This connection is to the powertrain mounts and accounts for reaction torque to the powertrain mount system.

2.1.4 Transmission

The transmission subsystem represents any "gearing" done to deliver power from the powerplant to the wheels. One side of the transmission is connected to the powerplant while the other side is connected to the driveline. Any hydraulic function associated with the transmission is assumed to be encapsulated within the transmission subsystem.

Like the powerplant, the transmission is also connected to the powertrain mounts. This is an important aspect that differentiates this architecture from most vehicle level models because it accounts for the influence of reaction torques in the powerplant, transmission and driveline on the motion of the powertrain. This is particularly important for the transmission because it can be the source of large amplitude, low frequency disturbances not effectively isolated by the mounting system [11].

As with all the physical subsystems, the transmission subsystem is connected to the electrical subsystem. In addition, the transmission is also connected to the driver. The driver connection represents the shifting mechanism for either a manual or automatic transmission depending on the configuration options chosen for the vehicle (these will be discussed later in Section 3.3).

2.1.5 Driveline

The driveline subsystem is responsible for modeling the distribution of transmission output torque to each of the wheels. For many vehicles, this distribution is determined by simple mechanical connections (*e.g.* differentials in strictly front-wheel or rear-wheel drive vehicles). In other cases, this distribution is actively controlled (*e.g.* on-demand four wheel drive systems).

Physically, the driveline is connected to the output side of the transmission and generally has the potential to influence each of the wheels. In order to avoid a complex series of graphical connections, all wheels are lumped into a single connector which is also physically connected to both the brake and chassis subsystems. Note that the driveline subsystem is also connected to the mounting system and the electrical system.

2.1.6 Brakes

The brake subsystem represents not only the friction used to decelerate the vehicle but also, as with the transmission, any encapsulated hydraulic function. The brake subsystem is physically connected to each wheel (*via* the single connector described in Section 2.1.5), the electrical subsystem and the brake pedal (associated with the driver). As with the powerplant, the connection to the driver could represent either direct actuator control by the driver or a "brake by wire" configuration where the brake pedal position sensor would be contained in the brake subsystem with pedal position information communicated to the brake subsystem controller and/or vehicle controller.

2.1.7 Chassis

The chassis subsystem represents the vehicle body, frame, wheels and suspension system. One remaining issue with the decomposition described in [5] is the handling of the steering mechanism. It is still an open issue what the physical interface between the steering mechanism and the suspension system should be. For now, we have kept the steering components inside the chassis while we collect feedback from experts on the best way to separate these two systems.

While for many applications the chassis may be modeled as a simple unsprung mass constrained to move longitudinally, the goal of this architecture is to provide sufficient flexibility to accommodate complex vehicle dynamics models ([1, 9]). The chassis subsystem is physically connected to the wheels and also to the powerplant, transmission and driveline through the mounts. The modeling of the mounts is handled inside the chassis system. Furthermore, the actual physical type of the mounting connections is configurable (*e.g.* 1D, 3D, *etc.*). The modeling of the road-tire interface is also handled inside the chassis subsystem.

Physically, the chassis system is also connected to the electrical system and the steering wheel. As with the brake and powerplant models, the connection to the driver may represent a "by wire" connection.

2.2 Controllers

While analysis performed during the subsystem design process can sometimes be accomplished using simple open-loop control strategies for a single subsystem, it is much more important that vehicle level models include closed-loop control to capture communication between each subsystem plant and controller pair as well as physical interactions across the various physical subsystems.

The subsystem controllers are decomposed along similar lines as their physical counterparts. Rather than categorize the controllers by subsystem, we will focus on the controller hierarchy and how the controllers communicate both with each other and with the physical subsystems.

2.2.1 Vehicle System Controller

This vehicle architecture includes a hierarchy of controllers. At the top of this hierarchy is the vehicle system controller. The vehicle system controller exists to control vehicle level functions and deal with arbitration and apportioning of subsystem functions (*e.g.* balancing how much

motive torque is delivered from the internal combustion engine versus how much is delivered by electric motors in a hybrid electric vehicle).

In order to function, a vehicle system controller (if present, not all vehicles implement one) must communicate with each of the subsystem controllers on the vehicle. In an actual vehicle, this kind of communication would be done through a vehicle level communication bus (*e.g.* a Controller Area Network, or CAN, bus). Although the behavior of the bus itself can have a significant impact on overall vehicle performance, modeling of the bus is not currently within the scope of this architecture.

2.2.2 Subsystem Controllers

As shown in Figure 1, associated with each physical subsystem is a controller for that subsystem. These controllers are responsible for controlling the function of their particular subsystem. For example, for a vehicle with an internal combustion engine, the powerplant subsystem controller would be responsible for determining spark timing, injector timing and other specialized functions like cam phasing control.

Each subsystem controller must communicate with its associated physical subsystem to exchange sensor and actuator information. In addition, each subsystem may receive supervisory commands from a vehicle system controller. Finally, the architecture should accommodate any combination of continuous controllers (*e.g.* formulated using block diagrams) and/or discrete controllers (*e.g.* employing Petri-nets, z-domain blocks or embedded code).

2.2.3 Communication Buses

As mentioned previously, bus behavior can have a significant impact on vehicle performance. Although we would like to capture these effects, we feel it is important to focus initially on the interactions between the physical subsystems and controllers.

Even if we ignore the behavior of the communication bus, we still need to represent the information exchanged on the bus. This is complicated by the fact that each subsystem design can potentially have a wide variety of signals that must be communicated between the subsystem controller and its physical counterpart. For example, one powerplant may contain an internal combustion engine that has cam phasing while another one does not (while a third may have an electric motor as a powerplant and therefore an entirely different set of sensor and actuator signals).

For each case, the subsystem controller must have the appropriate architecture to deal with the varying sets of sensors and actuators in each case. As a result, the set of signals exchanged between the controller and its physical counterpart must be customizable on a per configuration basis.

In a similar way, the information exchanged between the vehicle system controller and each of the subsystem controllers will also depend on whether a vehicle system controller is present and, if so, what features are implemented at the system level.

2.3 External Influences

Apart from the physical subsystems and controllers, a vehicle system model must account for two important external influences. The first influence is the driver. While the driver is not strictly part of the vehicle, the driver obviously has a tremendous influence over the response of the vehicle. The other external influence is the environment. The environment could potentially influence things like air temperature and composition (used in predicting engine performance), road surface effects (*e.g.* changes in elevation, traction characteristics), obstacles or other vehicles (potentially necessary in evaluating intelligent cruise control and other active safety features).

In some sense, the driver is both a physical subsystem and a controller. Both of these functions are lumped into a single driver model. The environment is assumed to be purely autonomous typically based purely on time and vehicle position.

3 Modelica Features

3.1 Acausal Modeling

The rich set of physical modeling and configuration management features associated with the Modelica modeling language [10] provide great potential for vehicle system analysis [11].

Vehicle systems are typically modeled from either a "forward" [12] or "backward" [13] perspective. This limits the reusability of component models because they must be developed with these perspectives in mind. From a purely physical perspective, the ability to build components and subsystems without *a priori* causality assumptions allows these components and subsystems to be used in both "backward" and "forward" vehicle modeling applications. Beyond

the reusability of components that results from this acausal approach, the use of inheritance, subtype constraints and the ability to declare replaceable components and subsystems is often useful in practice for large scale modeling projects. In this section, we will discuss how these features allow us to satisfy important requirements for our vehicle model architecture.

3.2 Replaceable Subsystems and Controllers

The cornerstone of configuration management in Modelica is the ability to declare types and components as **replaceable**. In fact, all the physical subsystems, controllers and external influence components shown in Figure 1 are declared replaceable so that alternative configurations can be easily created. Furthermore, constraining types are also defined for each of these components to prevent inappropriate substitutions from being made.

One problem with making each component **replaceable** is that it leaves open the possibility that novice users will attempt to pair plant and controller models together that are not compatible with each other (*e.g.* the controller expects an automatic transmission but the actual transmission plant is a manual transmission). So, in addition to making each component in Figure 1 **replaceable**, the set of models associated with each subsystem (*i.e.* the plant, local controller bus signals, local controller and global bus signals) are grouped together (using replaceable packages) so that entire subsystem configurations can be changed in a single operation. This allows users to select from pre-packaged, consistent and compatible collections of these models that can be changed in a single operation.

Ultimately, vehicle level models will extend from the template shown in Figure 1 and then use redeclarations (as class modifications) to create each specific vehicle configuration. Furthermore, alternative vehicle configurations can then extend from each other *ad infinitum* to create many different variations on a baseline design. This approach allows users to easily control configuration options while at the same time maximizing reuse. In turn, this minimizes redundant code and/or configuration options across different configurations which greatly eases maintenance of the models.

3.3 Subsystem Configuration Options

As mentioned in Section 2.2.3, the set of signals communicated on each bus depends on the specific set of features implemented in each subsystem. To address this issue, our architecture contains a set of replaceable packages that are used to propagate specific definitions for connectors and/or records that are configuration specific.

For example, the powerplant configuration package includes a definition for the connector used to communicate information between the physical powerplant and the powerplant subsystem controller. That definition, in turn, can be customized (using replaceable type definitions) to specify what kind of information is required for each control feature. In this way, the fact that a particular powerplant has, for example, a dual independent cam phasing feature can be stated as a configuration option which then automatically adds the necessary signals to the connectors used on both the physical powerplant and the powerplant controller. In other words, for any given vehicle model there is a single top-level configuration option for each subsystem that ensures consistent bus definitions throughout the vehicle model.

This is essentially the same idiom, utilizing replaceable packages, that is sometimes used to model different media in fluid modeling applications [14].

3.4 Common Environment

The ambient environment in this architecture contains information that is potentially relevant to every subsystem. Since the environment is a model (potentially with its own equations and states), it isn't possible to propagate the environment component through the vehicle hierarchy. Instead, an **inner** qualifier is used to make the information available to other components in the hierarchy.

3.5 Documentation

The ability to embed documentation about a package, subsystem, connector, etc. into its definition has already been utilized in this package to provide model developers with a useful online reference for the various interface definitions as well as HTML versions of the same information which can be posted, for example, on a corporate intranet site for reference.

4 Sample Application

To demonstrate how this architecture can be used to build a specific vehicle, we started from the base vehicle configuration shown in Figure 1 and added specific engine, transmission, driveline, brakes and chassis models. Along with these physical subsystem models, controllers for the engine and transmission were included to handle spark timing and gear shifting. The accessory and electrical subsystems were neglected in our example. The purpose of the model is to evaluate performance characteristics such as 0-60 MPH times and 0-400 meter times.

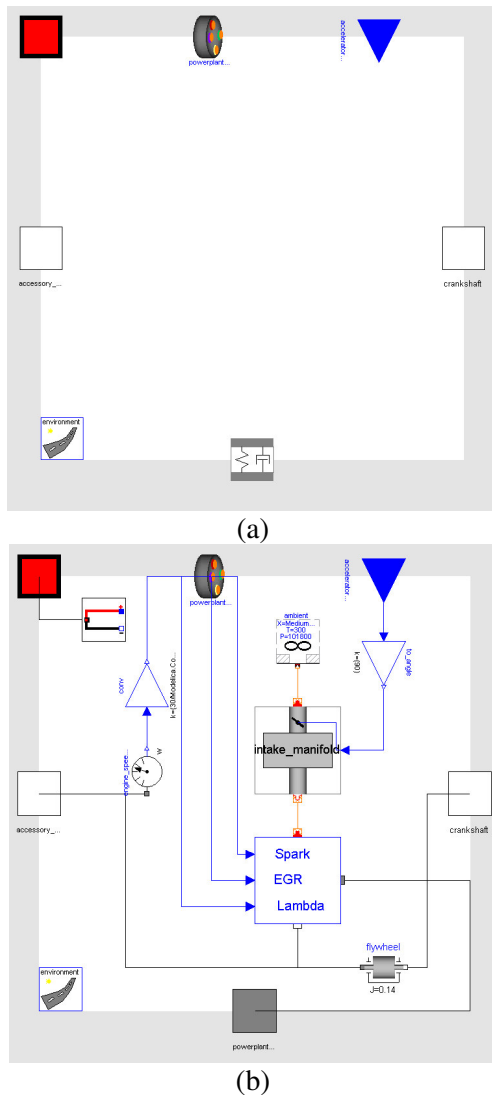


Figure 2: (a) Powerplant Interface; (b) Sample Engine

4.1 Engine

The engine model used in this example includes simple "filling and emptying" dynamics for the engine manifold and uses a table to lookup

engine torque as a function of spark timing, air fuel ratio and recirculated exhaust gas. Figure 2a shows the basic interface definition for a powerplant. Figure 2b shows our sample model which extends from the interface definitions so it can inherit all the physical and control system connectors required for compatibility with the overall architecture. Since, for this example, we are only interested in simple 1D rotational dynamics of the powertrain, the powertrain mount connection has been redeclared as a 1D rotational flange. Once this is done, the subsystem model is populated with component models which are connected to each other and to the interface connectors. Note that this particular subsystem translates driver accelerator pedal position directly into a throttle angle, reads the engine control parameters (*i.e.* spark, intended air-fuel ratio and command exhaust gas recirculation) from the subsystem control bus and writes the engine speed back onto the subsystem control bus.

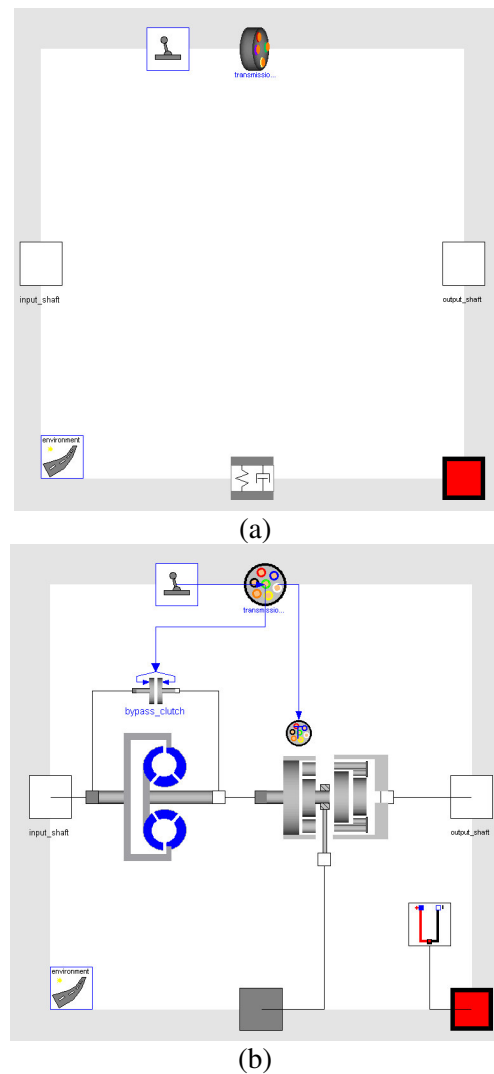


Figure 3: (a) Transmission Interface; (b) Sample Transmission

4.2 Transmission

The transmission model represents a six speed automatic transmission. The basic transmission interface is shown in Figure 3a. By extending from the interface, redeclaring connectors and adding components we eventually end up with a complete transmission model as shown in Figure 3b which includes the torque converter, bypass clutch and gearbox. The gearbox is further composed of a series of planetary gear sets, inertias and clutches (not shown). Note that in this model we assume that the gear selection information is propagated back to the transmission subsystem controller which, based on this information, command the engaging and disengaging of specific clutches inside the gearbox.

4.3 Remaining Subsystems

The remaining subsystems do not contain much detail. Rather than presenting the interface and implementation for each subsystem, we will just summarize the behavior represented in each:

- Accessories – No accessory loads are considered in this analysis.
- Electrical – The electrical system provides a constant 12V to the other components (although none of these simple models draw any current).
- Brakes – The brakes are modeled as simple friction elements (from the Modelica standard library).
- Driveline – The driveline provides power to the front axle of the vehicle through a final drive gearset and a simple differential element
- Chassis – The chassis response is purely longitudinal. The tire behavior uses the Pacejka characterization [7] and the vehicle mass is represented by a single lumped mass. No weight distribution effects are included.

4.4 Control

The only control functions required for this analysis are spark control (to maximize mean engine torque), shift scheduling and clutch control (*i.e.* engaging and disengaging clutches depending on the currently requested gear). In addition, the chassis subsystem provides vehicle speed to its local subsystem controller that transmits the information to the transmission subsystem controller via the vehicle level communication bus.

4.5 Results

The models used to demonstrate the capabilities of this vehicle model architecture are part of the training materials used within Ford to familiarize engineers and model developers with Dymola and Modelica. As such, it is important to point out that the subsystem specifications and system simulation results do not represent or reflect the performance of any particular Ford vehicles. In fact, the controller calibrations are intentionally made sub-optimal to allow students to further refine them.

The training exercise that these models were taken from focuses on vehicle acceleration performance. Figure 4 shows the vehicle acceleration plotted as a function of time. From this plot we can clearly see the "torque holes" that occur while the transmission is shifting. In addition, the upper limit on acceleration seen at the start of the simulation represents the limited longitudinal traction provided by the tires before they start to slip.

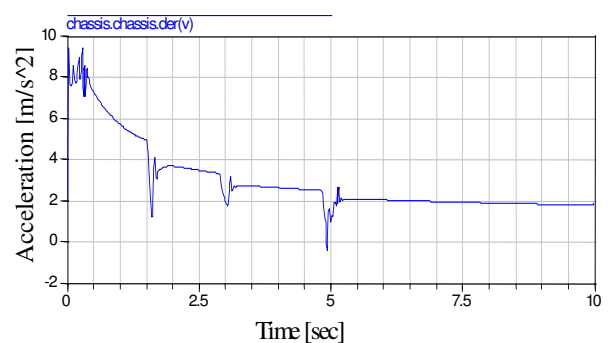


Figure 4: Vehicle Acceleration vs. Time

It is also interesting to examine the engine speed during the simulation as shown in Figure 5. Studying the RPM signal we can clearly see an "engine flare" at about 5 seconds into the simulation. Such flares occur when the shifting of the clutches in the transmission is not well controlled. As a result of poor control, the overall torque capacity of the transmission is less than the torque generated by the engine and the engine accelerates rapidly until the clutches engage.

In addition to examining the physical signals within the system (*e.g.* torques, speeds, *etc.*), it is also interesting to examine the communication between the controllers. Figure 6 shows the clutch and band engagement requests sent from the transmission controller to the physical transmission. These are actuator commands instructing the hydraulic controllers within the transmission to engage specific clutches and/or bands.

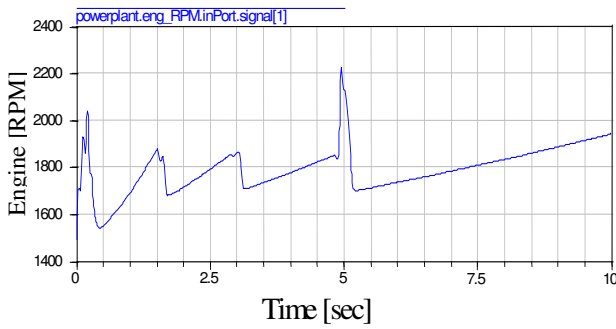


Figure 5: Engine Speed

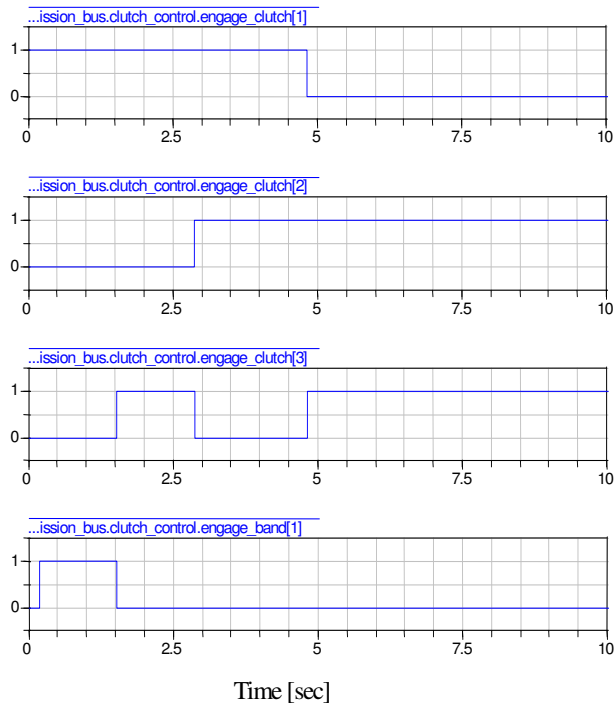


Figure 6: Clutch/Band Engagement

Similarly, in Figure 7 we can see the internal decision making process of the transmission subsystem controller by plotting its selection of gear during the simulation. This information is what ultimately dictates the detailed clutch/band engagements show in Figure 6.

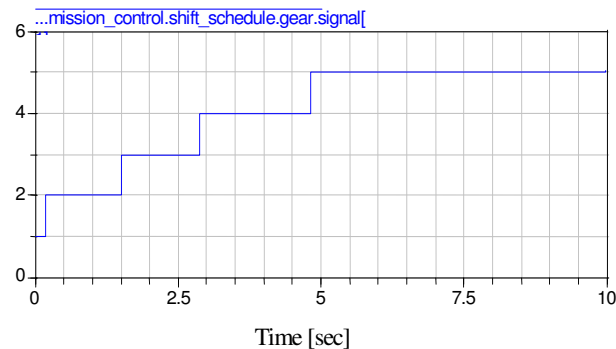


Figure 7: Gear Selection

Finally, many insights can be gained by plotting some of the simulation variables with respect to each other. For example, if an engineer knows at approximately what speed the peak in the engine power curve appears, he might plot the commanded gear selection as a function of engine speed, as shown in Figure 8 for this example, to make sure that the shift strategy appropriately straddles that peak.

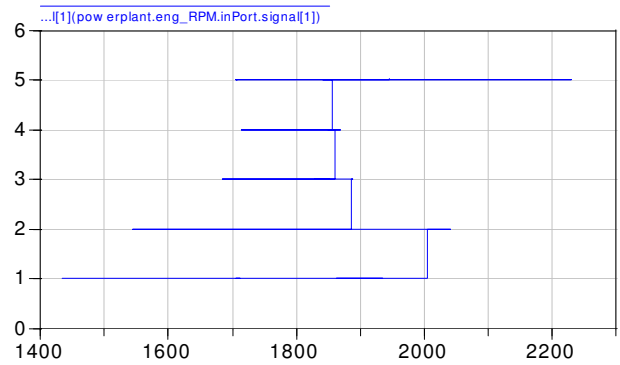


Figure 8: Gear Selection vs. Engine RPM

This section demonstrates just a few of the possible results that a vehicle level analysis can uncover. Having a standardized set of interfaces not only makes the exchange of models easier, it also assures, to some degree, that signals will have common names (at least those associated with the provided interfaces).

5 Usability Considerations

Some of the more advanced Modelica language features used in this architecture (*e.g.* replaceable packages, choice annotations, subtype definitions for classes, *etc*) are not necessarily accessible or intuitive for end users. In this section, we describe some ideas for representing the complex structure of the vehicle so that end users can easily configure and reconfigure vehicle models.

5.1 Handling User Choices

5.1.1 Link Choices to Component Icons

First, it should be possible to select a component in a vehicle model and browse a set of compatible alternative components. In other words, the set of alternatives should be easily accessible *from the graphical icon associated with that component* rather than requiring users to find components in, for example, the component browser (which requires knowledge of what classes the components were inherited from).

5.1.2 Consistent Handling of Choices

For complex "template" models (*i.e.* models that are designed so that end users can merely "fill in the blanks"), it is important that users be presented with a complete view of the model including all redeclarations/customizations they have made. Redeclarations can affect many different "visual" aspects of the model including its inheritance, its component hierarchy, the parameter dialogs, graphical appearance, results structure, associated scripts, etc. It is important for tools to make sure that all of these possibilities are always consistent with the choices made by the end user when customizing the models.

When interface definitions are influenced by top-level choices (*e.g.* the physical powerplant interface is altered by the choices made in the top level powerplant configuration package), this should influence the set of possibilities generated with the `choicesAllMatching` annotation in the models. For example, if the top-level configuration specifies a powerplant with dual independent cam phasing, the set of choices generated when redeclaring the powerplant should only include powerplant models that can satisfy that interface.

5.1.3 Carryover and Memory of Choices

While exploring alternatives, graphical tools should perpetuate user modifications for identical parameters and/or choices when possible and, when not possible, remember those modifications in case the same options reappear. For example, if a user configures a model to use one particular 5 speed transmission model and then switches to a different 5 speed transmission model, it should be possible to carryover any common parameters (*e.g.* gear ratios) or choices (*e.g.* torque converter model) between the two alternatives. In addition, if they explore the idea of a continuously variable transmission (CVT), the tool should remember the gear ratio settings if they decide to revert back to a 5 speed transmission.

5.2 Visualization

5.2.1 Decision Tree Visualization

With a template model as complicated as the one shown in Figure 1, the options and possibilities open to the end user can be quite disorienting. For these kinds of models, it would be very useful to have a compact representation of the tree of possible choices open to the user. Such a tree would need to be hierarchical and each decision that is made should be reflected in the tree (*i.e.* the tree should respond dynamically to user choices). Ideally, such

a tree should show, in a single comprehensive view, choices that influence topological changes (*e.g.* what transmission model is used) as well as parameters.

5.2.2 Visualizing Configurations

Another issue with template models is the proliferation of variations. It should be possible to visualize in a coherent way the modifications associated with a "tree" of configurations (in this case, a tree based on the inheritance hierarchy as opposed to the tree discussed in Section 5.2.1 which is based on the compositional hierarchy).

6 Limitations

While Modelica provides some powerful features to support the architecture described in this paper, there are still some areas where the existing features are still not sufficient. In this section, we will discuss some of the limitations we encountered and some ideas for overcoming those limitations.

As described in Section 3.3, we have chosen to propagate configuration information from the top down. In other words, decisions about connector definitions are made at the top level and then propagated to subsystems. This is awkward because it is often unnatural for this information to either appear or originate at the vehicle level. For example, information about signals exchanged between the powerplant and the powerplant controller is really determined by the set of sensors and actuators present on the powerplant itself but we were not able to find a way of expressing this in Modelica.

Along similar lines, the set of signals communicated on the vehicle control bus should be the union of all signals broadcast from each subsystem controller. From a user perspective, it would be best to simply choose the controller and physical subsystem and have the information about broadcast messages "propagate up" automatically to the vehicle level controller bus.

In the current design, the subsystem bus connector on the physical subsystems is always declared **inner**. This is done to allow the use of the `SignalBus` idiom [8] which allows sensors and actuators to reference only the specific signals they require (as opposed to all signals communicated in that subsystem). Unfortunately, the relationship between the bus connector and these sensors and actuators is not explicit because it relies on using **inner** and **outer** qualifiers. A better solution would be to allow direct connections.

Unfortunately, the current Modelica specification requires each connector to contain exactly the same signals. By relaxing this requirement and, for example, allowing one connector to be a subtype of the other, such connections would be possible and, as a result, clearer.

One of the biggest problems in developing such a framework is how to represent the fundamental engineering assumptions present. For example, the powertrain mounts might be represented as either 1D or 3D connections. Likewise, the electrical system may support multiple voltage levels. Several subsystem models can be impacted by these choices and there is no easy way of understanding what assumptions are made for particular models and how that affects the assembly and compatibility at the vehicle level. Rather than relying on complex nested replaceable type definitions and interfaces, the entire process might be more coherently represented with features (*e.g.* layers) that provide configuration based on a fixed set of possibilities.

7 Future Work

It is important to reiterate that the structure defined in this document is merely a proposal and that further discussion is required. Once a consensus is reached on the appropriate subsystem decomposition and interface definitions, there are several potential directions for this work. For example, it might be useful to extend the depth of the current hierarchy to define architectures for each of the various subsystems. For example, powerplant templates could be developed for internal combustion engines (*e.g.* I-4 or V-6 cylinder configurations) and transmission templates could be developed that decompose automatic transmissions into individual models for a torque converter, bypass clutch and gearbox (with interface definitions for each). Finally, other top-level architectures could be developed that reuse the subsystem interface definitions. These architectures may choose to use a subset of the subsystems shown in Figure 1 (*e.g.* an engine connected to a dynamometer) or they may choose to add additional subsystems for more exotic vehicle configurations (for towing applications, fuel cell vehicles, *etc.*).

8 Acknowledgments

The architecture presented in this paper is heavily based on a Ford Motor Company internal initiative, by Mark Jennings, Judy Che, Bradley Hieb, Tim Mortimer, Ken Butts, Chris Belton, Pete

Burchill, Peter Bennet, David Copp and Nick Darnton, to develop a vehicle model architecture for Simulink [5]. This work leverages a great deal from the system decomposition and thorough analysis that was done as part of that work. As a result, the authors would like to recognize the significant influence and impact that work had on the material in this paper.

The authors would also like to thank John Batteh, Chuck Newman, Erik Surewaard, Graham King, Johan Andreasson, Christian Schweiger, Martin Otter, Jonas Hellgren, Jonas Karlsson, Jonas Fredriksson, Bengt Jacobson and Lars Eriksson for their work in developing automotive component and subsystem models which we hope will, at some point, be compatible and freely exchangeable through this architecture.

9 References

1. J. Andreasson, A. Möller and M. Otter, "Modeling of a Racing Car with Modelica's Multi-Body Library", *Modelica Workshop 2000 Proceedings*, <http://www.modelica.org/workshop2000/proceedings/Andreasson.pdf>
2. M. Otter, M. Dempsey and C. Schlegel, "Package PowerTrain. A Modelica library for modeling and simulation of vehicle power trains", *Modelica Workshop 2000 Proceedings*, p. 23-32, <http://www.modelica.org/workshop2000/proceedings/Otter.pdf>
3. P. Treffinger and M. Goedecke, "Development of Fuel Cell Powered Drive Trains With Modelica", *Proceedings of the 2nd Modelica Conference*, p.125-131, http://www.modelica.org/Conference2002/papers/p16_Treffinger.pdf
4. J. Hellgren, "Modelling of Hybrid Electric Vehicles in Modelica for Virtual Prototyping", *Proceedings of the 2nd Modelica Conference*, p. 247-256, http://www.modelica.org/Conference2002/papers/p32_Hellgren.pdf
5. C. Belton, P. Bennet, P. Burchill, D. Copp, N. Darnton, K. Butts, J. Che, B. Hieb, M. Jennings and T. Mortimer, "A Vehicle Model Architecture for Vehicle System Control Design", *SAE Congress 2003*, SAE-2003-01-0092.
6. "Dymola 5.0 User's Manual", *Dynasim AB*, p. 206.

7. H. B. Pacejka and E. Bakker, "The magic formula tyre model.", *Proceedings of the 1st Tyre Colloquium, Delft*, October 1991.
8. M. Tiller, W. E. Tobler and M. Kuang, "Evaluating Engine Contributions to HEV Driveline Vibrations", *Proceedings of the 2nd Modelica Conference*, p. 19-24, http://www.modelica.org/Conference2002/papers/p03_Tiller.pdf
9. S. Drogies and M. Bauer, "Modeling Road Vehicle Dynamics with Modelica", *Modelica Workshop 2000 Proceedings*, p. 161-168, <http://www.modelica.org/workshop2000/proceedings/Drogies.pdf>
10. "Modelica Language Specification, Version 2.0", *Modelica Association*, 2002,
11. M. Tiller, "Introduction to Physical Modeling with Modelica", *Kluwer Academic Publishers*, 2001.
12. K. Wipke, M. Cuddy and S. Burch, "Advisor 2.1: A User-Friendly Advanced Powertrain Simulation Using a Combined Backward/Forward Approach", *IEEE Transactions on Vehicular Technology: Special Issue on Hybrid Electric Vehicles*, 1999, http://www.ctts.nrel.gov/analysis/pdfs/advisor_21.pdf
13. A. Rousseua, S. Pagerit, G. Monney and A. Feng, "The New PNGV System Analysis Toolkit V4.1- Evolution and Improvement", *SAE 2001 Future Transportation Technology Conference*, SAE 2001-01-2536.
14. C. Newman, J. Batteh and M. Tiller, "Spark-Ignited-Engine Cycle Simulation in Modelica", *Proceedings of the 2nd Modelica Conference*, p. 133-142, http://www.modelica.org/Conference2002/papers/p17_Newman.pdf

