VehicleSystems Library Tutorial

Dr. Michael Tiller
Ford Motor Company

Outline

- Background on **VehicleSystems** and VMA
- Why Modelica?
- Hands-on Exercise: Creating a new vehicle model
- Loading subsystems
- Hands-on Exercise: Simple fuel economy analysis
- Hands-on Exercise: Calibration study (race)
- Interfaces: In Depth
- Concluding remarks

Background

`VehicleSystems` History

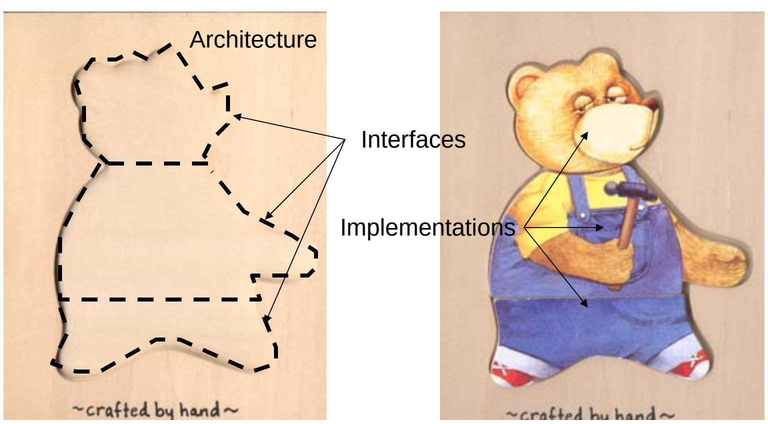
- Based on the “Vehicle Model Architecture” (VMA) developed at Ford Motor Company
- Vehicle decomposition with a standard interface for each subsystem
- Developed by vehicle modeling activities across Ford Motor Company (*e.g.* Jaguar, Ford, Volvo, *etc*)
- Includes plant and controllers

What is an "Architecture"?

- A *framework* of *interfaces* for combining compatible *implementations* of vehicle subsystems.

…a bit too technical to start with…

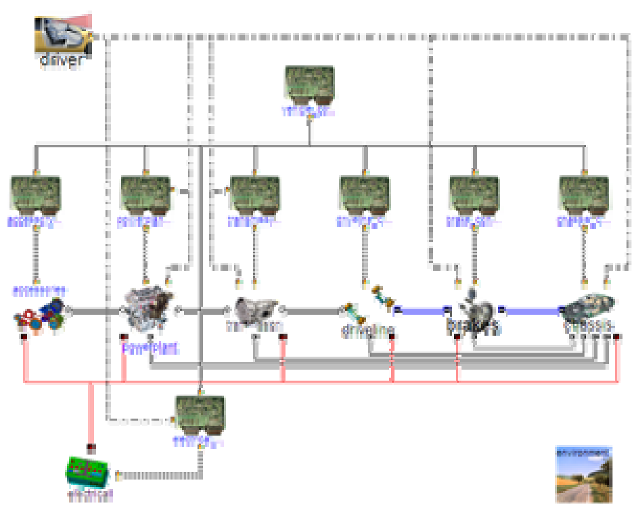
What is an "Architecture"?



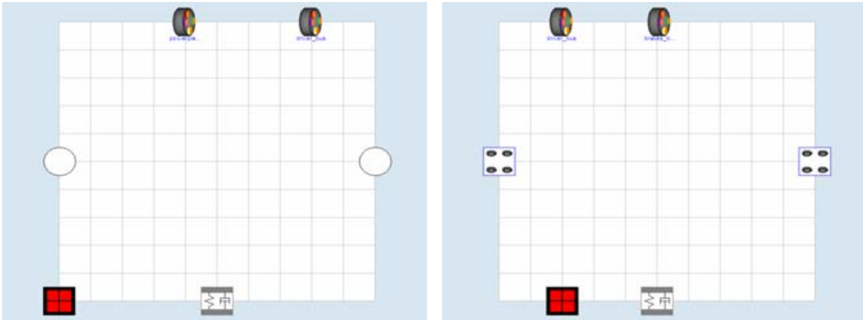
What is an "Architecture"?

- A *framework* of *interfaces* for combining compatible *implementations* of vehicle subsystems.
  - Framework – complete system description.
  - Interface – definition of the available connections (physical and signals).
  - Implementations – Actual working models of particular subsystems.

Conventional Vehicle Architecture



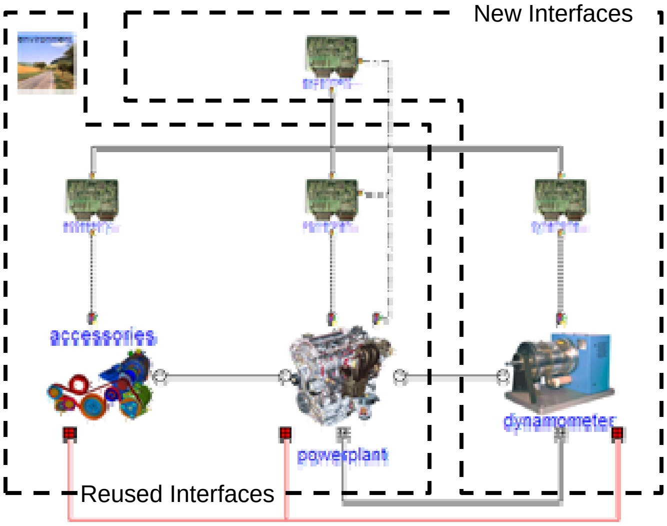
Interface Definitions



Powerplant Interface

Brake Interface

Alternative Architectures



Why Modelica?

- Object-oriented principles required:
  - Interfaces (`public`, `partial`)
  - Inheritance (`extends`)
  - Parameterized types and components (`replaceable` and `redeclare`)
  - Bus semantics (`expandable`)
- Multiple modeling formalisms
  - Signal flow/block diagrams
  - Acausal modeling
  - State transitions

The Supplier-OEM Conundrum

- OEMs want models from Suppliers
- OEMs want standard tools internally
- Suppliers deal with multiple OEMs using different tools
- Suppliers cannot support multiple representations of the same models
- **Need tool-neutral representation**

`VehicleSystems` Summary

- Provides a specification for OEMs, suppliers and even groups within the same company to target to promote compatibility
- Interfaces can be used independent of the architecture.
- Decomposition is currently only one layer deep.

Setup

Installing Dymola and Tutorials

- Insert CD
- Installation program for Dymola should come up automatically.
- Run “Modelica2005.exe” on the CD to install tutorial material.

Running Dymola

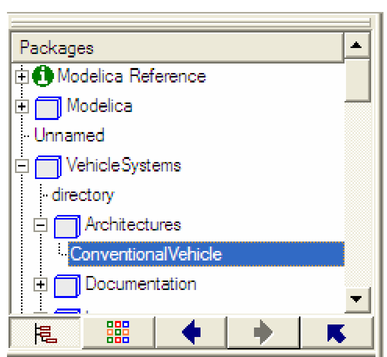
- Run Dymola
  - Either from desktop shortcut or from Start→ All Programs→Dymola
- Go to Simulation mode (tab in the lower right corner of Dymola)
- From main menu, Simulation→Run Script
  - File name: “setup.mos”
  - Default location: C:\Modelica2005\VehSys

Exercise 1

Creating a new vehicle model

Exercise 1: Steps

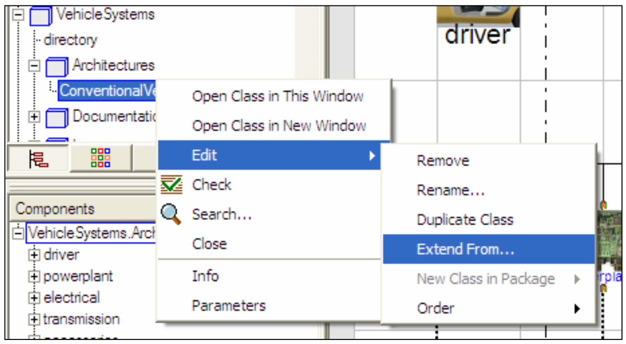
- Run Dymola
  - As described in Setup
- Find **`VehicleSystems`** in package browser



- In package browser, open
  - VehicleSystems→Architectures→ConventionalVehicle

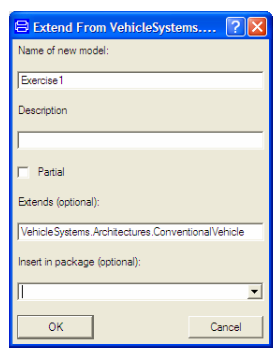
Exercise 1: New Vehicle

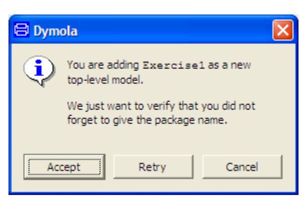
- Right click on
  - `ConventionalVehicle`
- Edit→Extend from...



Exercise 1: Dialogs

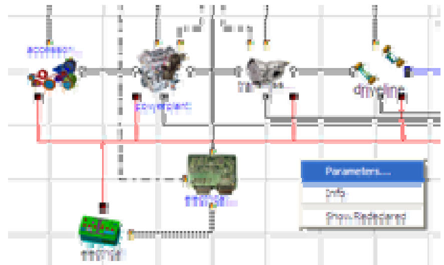
- New model dialog
  - Name of new model
    - Exercise1
  - Insert in package
    - Leave blank
- "You are adding `Exercise1` as a new top-level model…"
  - Accept





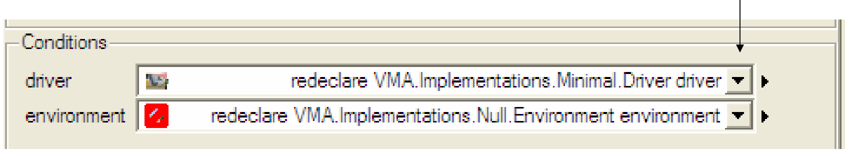
Exercise 1: Configuration

- Architecture configuration dialog
- In the diagram layer right click in an empty area, select "Parameters…"



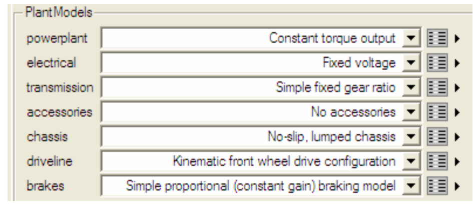
Exercise 1: Driver and Environment

- Select choice using
- Driver
  - "Max acceleration Driver"
- Environment
  - "No environment information"



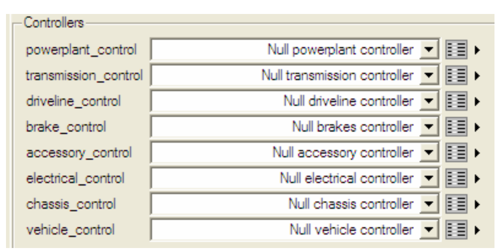
Exercise 1: Plant Models

- Only one option for each subsystem
- Fill in an implementation for each subsystem
- Should look like…



Exercise1: Controllers

- Again, only one option for each controller
- Should look like…



Exercise 1: Summary

- Demonstrates how to create a new vehicle model.
- Only used **`VehicleSystems`** library
  - Depends only on Modelica Standard Library (v2.1)
  - Currently includes only trivial implementations
- How do we build a real model?

Exercise 2

Fuel economy analysis

Disclaimer

- Exercises include sample models that have been developed to:
  - Provide functional implementations of subsystems
  - Demonstrate some basic characteristics and tradeoffs
- These models are not validated
- Basic, validated models will be available as part of the library when it is released.

Exercise 2: More Choices

- Exercise 1 had limited choices for each subsystem.
- No other compatible implementations had been loaded.
- File→Open
  - Open directory "`BasicVehicleSystems`"
  - Select "`package.mo`"

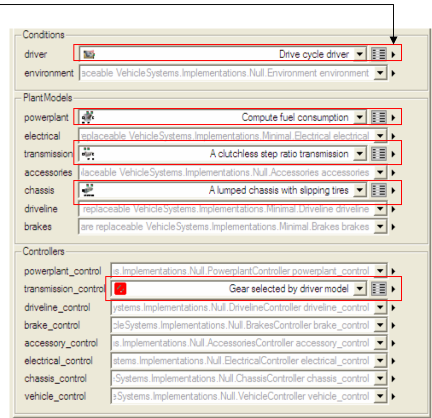
Exercise 2: New Subsystems

- Select “Exercise1” in package browser.
- Right click→Extend From…
  - Name: Exercise 1
- Right click on background of new model
- Select ‘Parameters…’
- Changes
  - Driver
  - Powerplant
  - Transmission
  - Transmission Controller
  - Chassis

Exercise 2: Config Dialog

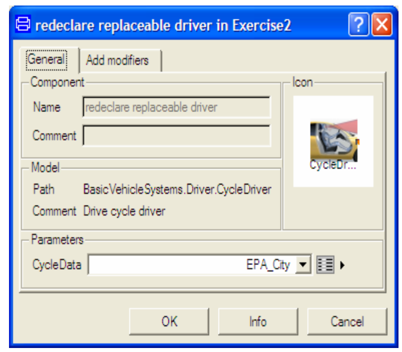
- Should look like…
- Configure subsystems by clicking here…

  …and select “Edit”



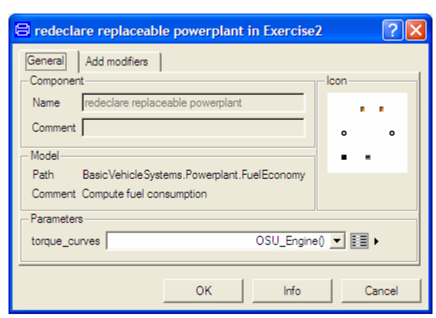
Exercise 2: Driver Configuration

- Choose:
  - “Drive cycle Driver”
- `CycleData`→ `EPA_City`



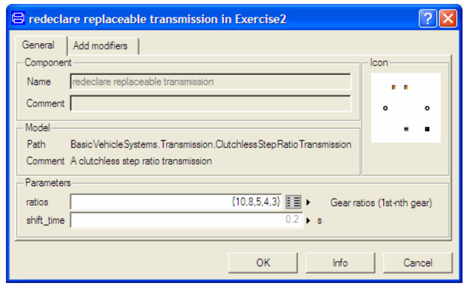
Exercise 2: Powerplant Configuration

- Choose:
  - “Compute fuel consumption”
- `torque_curves`→ `OSU_Engine()`



Exercise 2: Transmission Configuration

- Choose:
  - "a clutchless step-ratio transmission"
- `ratios` → `{10,8,5,4,3}`



- *Note: for simplicity these ratios include the final drive ratio.*

Exercise 2: Remaining Changes

- Transmission Controller:
  - "Gear selected by driver model"
  - Use default parameters
- Chassis:
  - "A lumped chassis with slipping tires"
  - Use default parameters

Exercise 2: Simulation

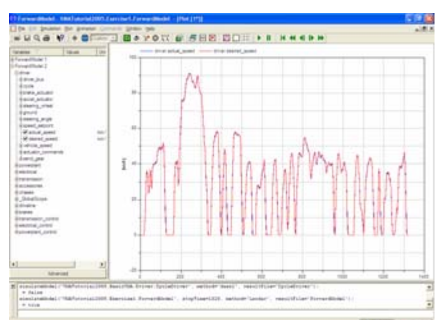
- Go into Simulation mode
- Setup simulation parameters
  - Simulation→Setup…
  - Stop Time: 1320
  - Algorithm: LSODAR
- Simulation→Simulate

Exercise 2: Results

- Plot actual vehicle speed vs. cycle speed profile:

  `driver.actual_speed`

  `driver.desired_speed`



- Fuel economy:
  - `powerplant.fuel_consumption`

Exercise 2: Summary

- High level assessment
- Basic data
- Simple plant models
  - No driveline dynamics
  - No suspension
- Simple control

Exercise 3

Performance assessment

Exercise 3: Overview

- Analyze performance (0-100 KPH time)
- Adjust controller calibration tables (shift schedule) to optimize performance
- Constrained by tire performance

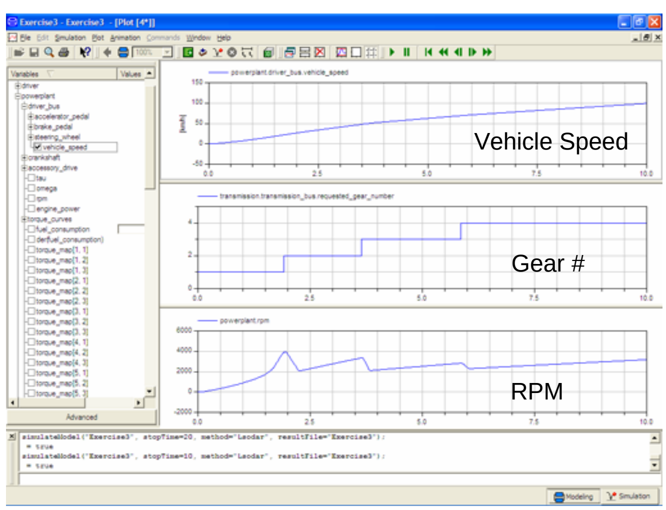
Exercise 3: New Subsystems

- Select Exercise2
  - Right click→Duplicate Class
  - Name: Exercise3
- Right click on background
- Select 'Parameters…'
- Changes
  - Driver: "Max acceleration"
  - Trans. Controller: "Gear selected by shift schedule"

Exercise 3: Simulation

- Go into Simulation mode
- Setup simulation parameters
  - Simulation→Setup…
  - Stop Time: **12**
  - Algorithm: LSODAR
- Simulation→Simulate

Exercise 3: Results

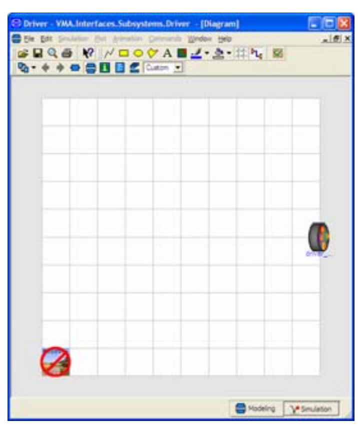


Interfaces

Quick introduction to subsystem interfaces

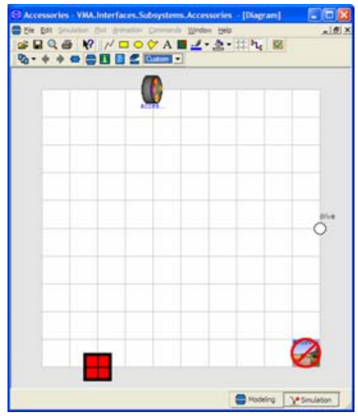
Driver – Physical Control

- Responsible for:
  - Commanded accelerator pedal position
  - Commanded brake pedal position
  - Commanded steering wheel angle
- Optional signals
  - Commanded gear
  - Electrical switches



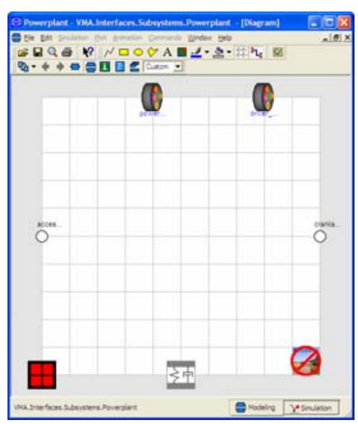
Accessories - Aux. Loads

- Connects to:
  - Powerplant
  - Electrical bus
  - Accessories controller
- Typical components:
  - A/C compressor
  - Starter/Alternator



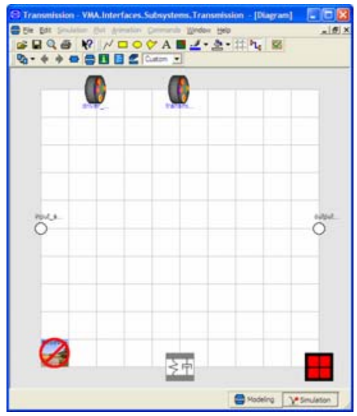
Powerplant – Mechanical Power

- Connects to:
  - Driver (throttle cable)
  - Powerplant controller
  - Accessories
  - Engine mounts
  - Electrical bus
  - Transmission



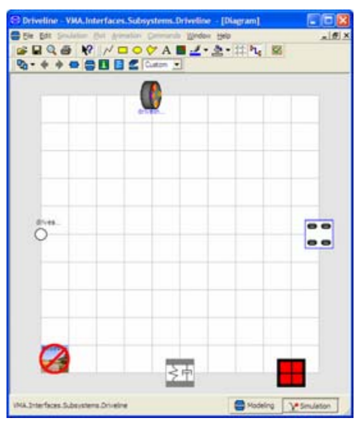
Transmission – Gear Reduction

- Connects to:
  - Driver (PRNDL)
  - Transmission
  - Powerplant
  - Engine mounts
  - Driveline
  - Transmission mounts
  - Electrical bus



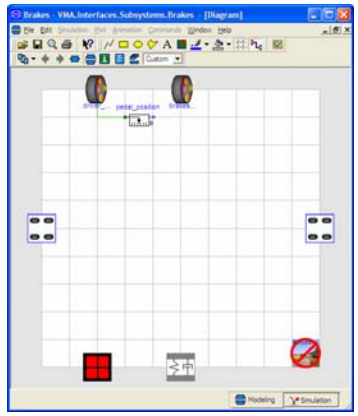
Driveline – Torque Distribution

- Connects to:
  - Transmission
  - Driveline controller
  - Wheels
  - Driveline mounts
  - Electrical Bus
- Typical components
  - FWD, RWD, 4WD, AWD



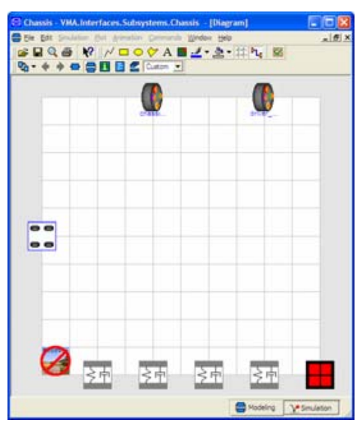
Brakes – Stopping the Vehicle

- Connects to:
  - Wheels
  - Driver (brake pedal)
  - Brake controller
  - Electrical bus
  - Brake mounts



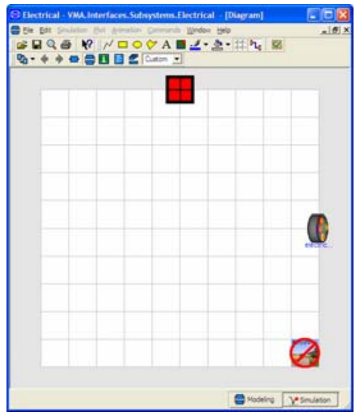
Chassis – Vehicle Body

- Connects to:
  - Wheels
  - Driver (feedback)
  - All Mounts
  - Electrical bus



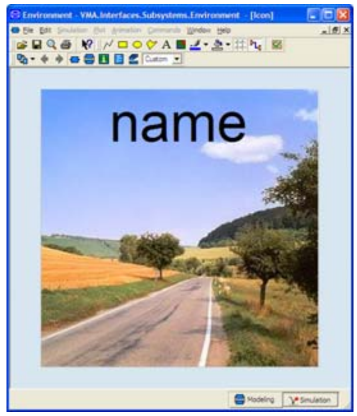
Electrical – Electrical Power

- Connects to:
  - All physical subsystems.
  - Electrical controller



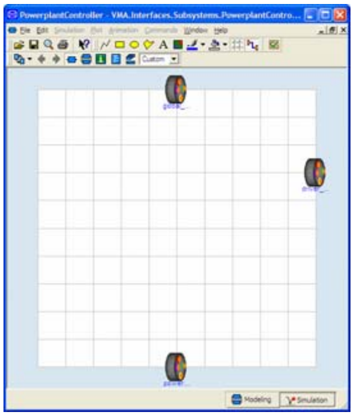
Environment – External

- Optional data:
  - Surface elevation
  - Ambient conditions
    - Pressure
    - Temperature
    - Humidity



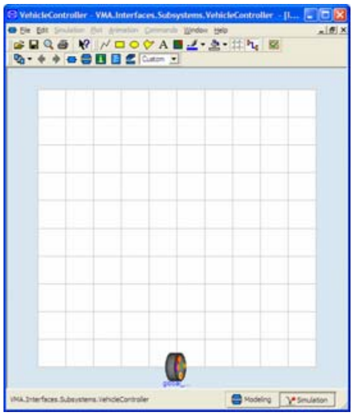
Subsystem Controllers

- Connect to:
  - Global vehicle bus (e.g. CAN bus)
  - Physical subsystem (e.g. high-speed bus)
  - Driver (in some cases)



Vehicle Controller

- Supervisor control:
  - Arbitration of subsystem requests
- Connects to:
  - Global bus (i.e. all subsystem controllers)



Conclusions

`VehicleSystems` Features

- Vendor neutral representation
- Developed with feedback from many groups
- Basis for collaboration when doing vehicle modeling
- Useful for performance, fuel economy, NVH applications
- Applicable to upfront assessments, design and calibration

Conclusions

- Ford Motor Company has released the original source to the public
- **`VehicleSystems`** version 1.0 should be released soon.
  - A few lingering issues
- Provides a framework for the exchange of subsystem models within and across various OEMs and suppliers.