# ISIS META Demo Script

## Table of Contents

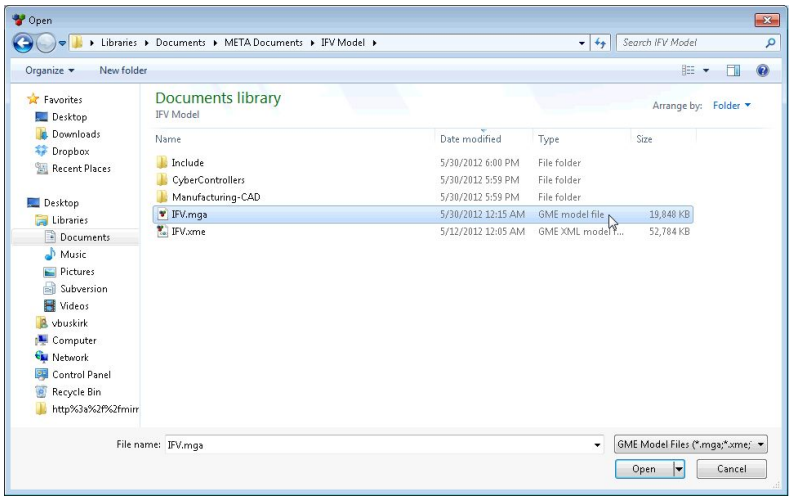
## Introduction to the GME Tool

This section will introduce some rudimentary concepts and give an overview of the user interface of the *Generic Modeling Environment* (*a.k.a.* GME) graphical modeling platform, which the META Toolsuite is built upon. The objective of this introductory section is to help first time users become accustomed to some generic aspects of GME such as toolbars, menus, and commonly used navigation widgets, in order to efficiently manipulate models[1].

We highlight some primary user interface widgets useful for working with AVM models. For demonstration purposes, the META tools come bundled with several example models. Throughout this tutorial, we will frequently work with the model named '*IFV*', as it is a good example of a non-trivial model encoded in the *CyPhy* graphical modeling language.

First, invoke the ***32—bit*** version of the GME application. Next, open the 'IFV' model in GME:

`File → Open Project… → C:\Users\Public\Public Documents\META Documents\IFV Model\IFV.mga`

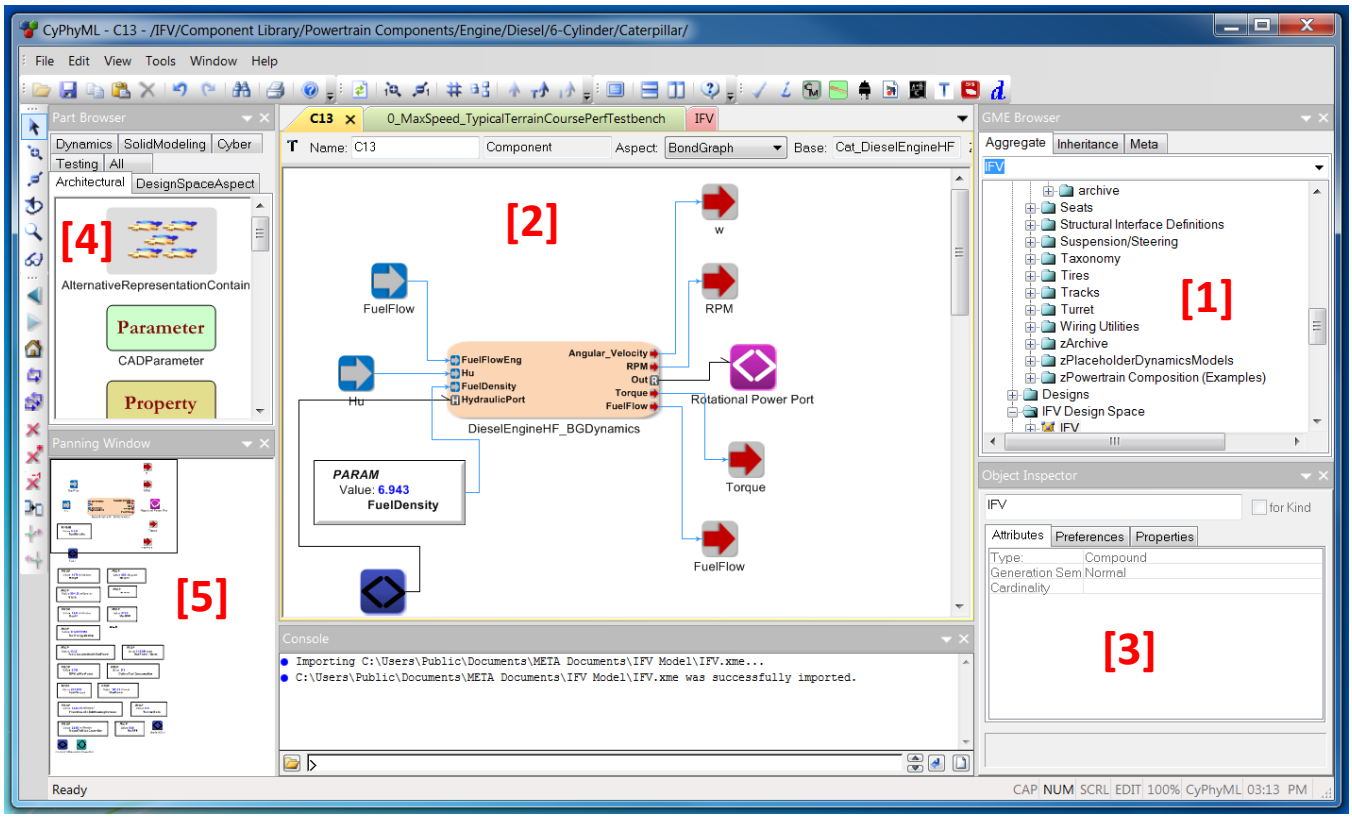



Alternatively, you may double-click on the desired '*GME model file*' (*i.e.* IFV.mga) from within a Windows Explorer application. Invoking GME this way will ensure that the 32-bit (versus the 64-bit) version of GME will be instantiated[2].

[1] A slightly lengthier discussion of the relationship between GME and the META Toolsuite is provided as Appendix B.

[2] While many of the model transformations rely on having a 64-bit version of GME installed, support for the 64-bit version of the graphical editor is not yet complete (as of June 2012). By running the 64-bit version of the GME installer, both 32 and




[1] **GME Browser** - Shows all available models comprising the IFV example in a tree view. Models reside inside folders, to open a model simply double click on it. Multiple models can be open simultaneously.

[2] **Active Model Window** – Opened models are arranged by tabs, clicking on a tab brings a model into focus. There are multiple items in the menu bar of these tabs. Some of the useful items are:

- Name of the displayed model.
- Aspect dropdown displays different views of the same model.
- Zoom Percentage

[3] **Object Inspector** – Displays attributes of the currently selected object. Objects are selected by single clicking on any element within the Active Model Window. Models may also be selected by left-clicking within the GME Browser widget.

[4] **Part Browser** – This tabbed interface shows the different *aspects* available in the displayed model, along with the valid types of modeling objects available in that aspect. One can switch between aspects by clicking on the tabs.

---

64-bit versions of GME.exe will be installed. When 64-bit executables or libraries are required by the META process, loading of the correct resources will be handled automatically by the tools. When invoking the model editor, however, please remember to use the 32-bit version of the tool.

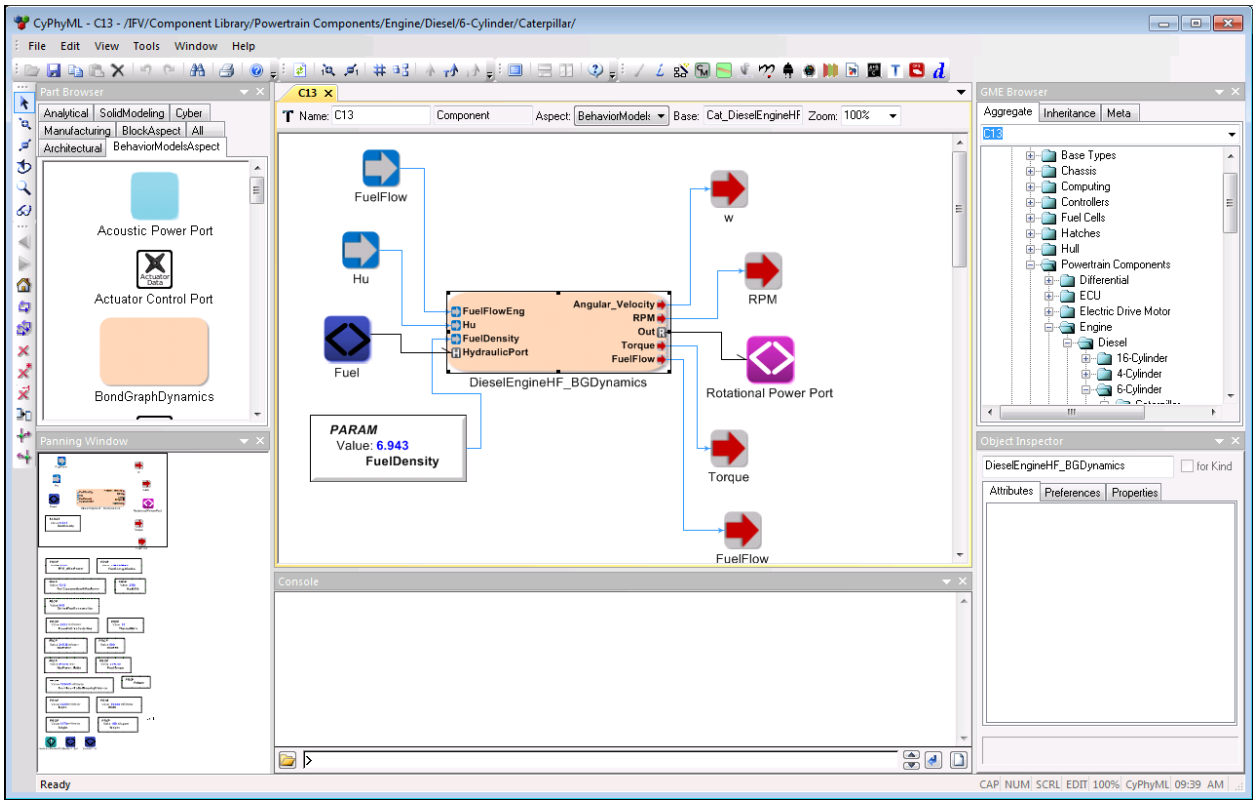
[5] **Panning Window** – Allows one to quickly navigate around the displayed model by moving the rectangular box in this window.

Each of these windows can be turned on and off in the View menu.

## An Overview of the IFV Example Model

This section introduces one of the elementary modeling concepts in CyPhy, namely 'Components'. Using the GME Browser widget, browse through the `Component Library` folder (*remember, the browser displays a tree structure of the model and is by default located on the upper-right window pane*). This demo will be working with a power train assembly modeled in the CyPhyML language. The component shown in the following screenshot is a **Caterpillar 6-Cyliner Diesel Engine**, which is highlighted in this power train demo:

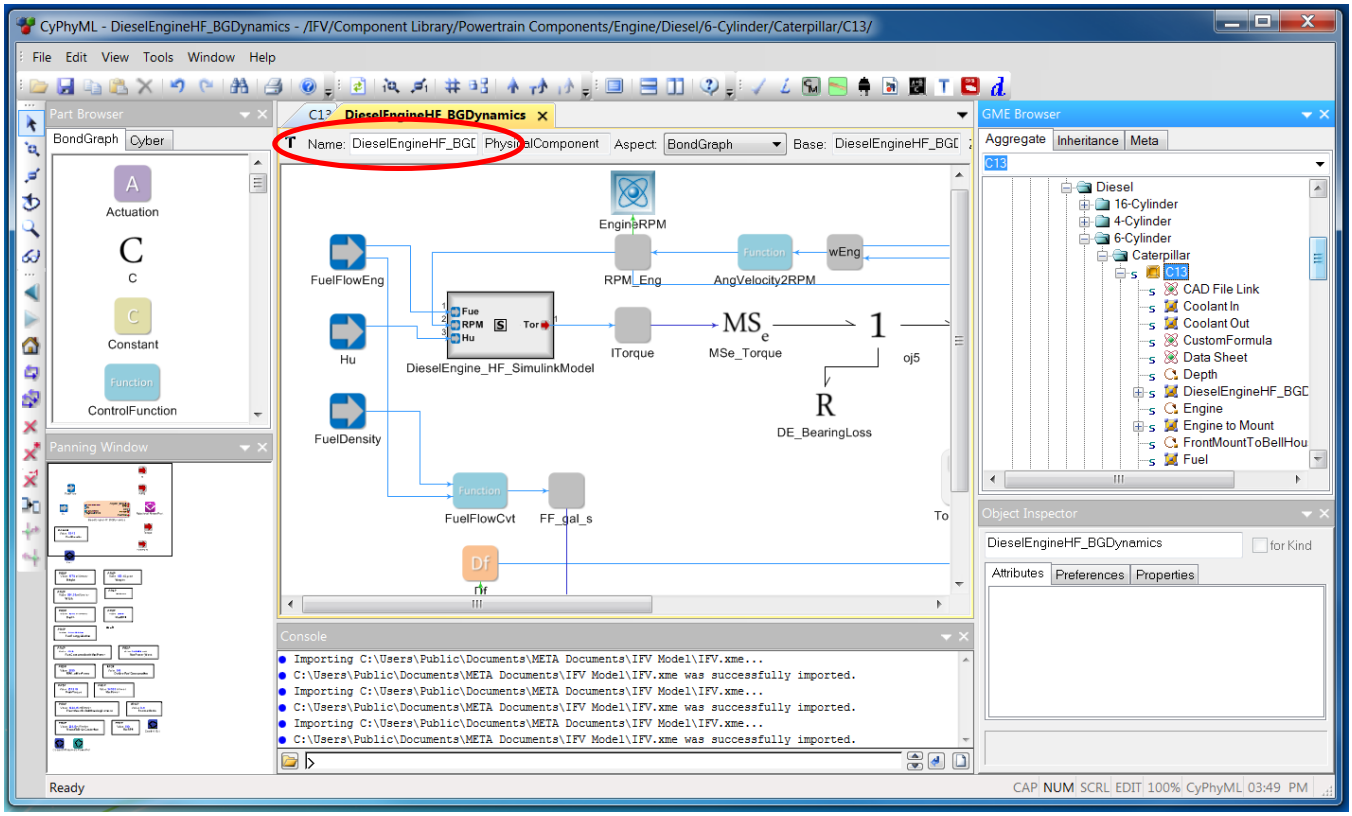
"`IFV/Component Library/Powertrain Components/Engine/Diesel/6-Cylinder/Caterpillar/C13`"



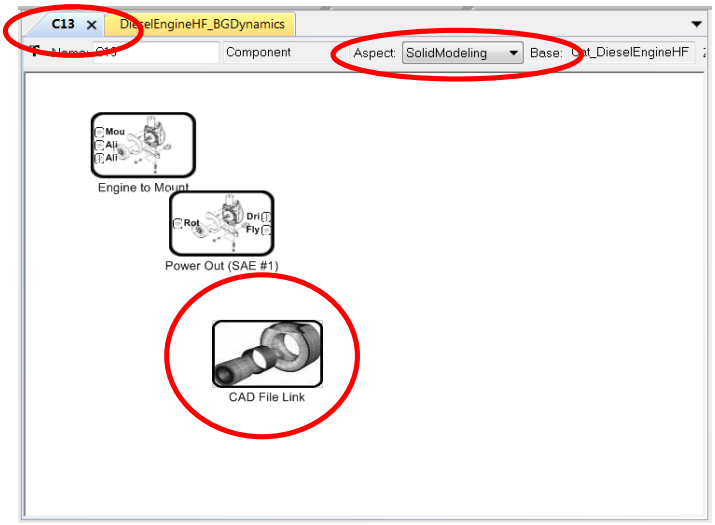

Take a tour of the `C13` diesel engine component. Use the `Aspect` drop-down in the center of the toolbar directly above the model window to select the model aspect you wish to view. The `ValueFlow` aspect reveals exposed properties[3] of the component, a reference to the data sheet(s) used in the creation of the model (*e.g.* double-click on the `Data Sheet` element). The `BehaviorModelsAspect` aspect contains the Bond Graph Dynamics Model of this diesel engine. Inputs to this `BondGraph` are `Fuel`, `FuelFlow`, `Hu` (*aka heating value*). The primary output is a `Rotational Power Port` to model the rotational energy produced by the engine. Explore the internals of the

[3] See also the 'Autolayout' section of Appendix A.

contained `DieselEngineHF_BGDynamics` model (by double-clicking) to view the details of the `BondGraph` representation of this physical component's behavior.



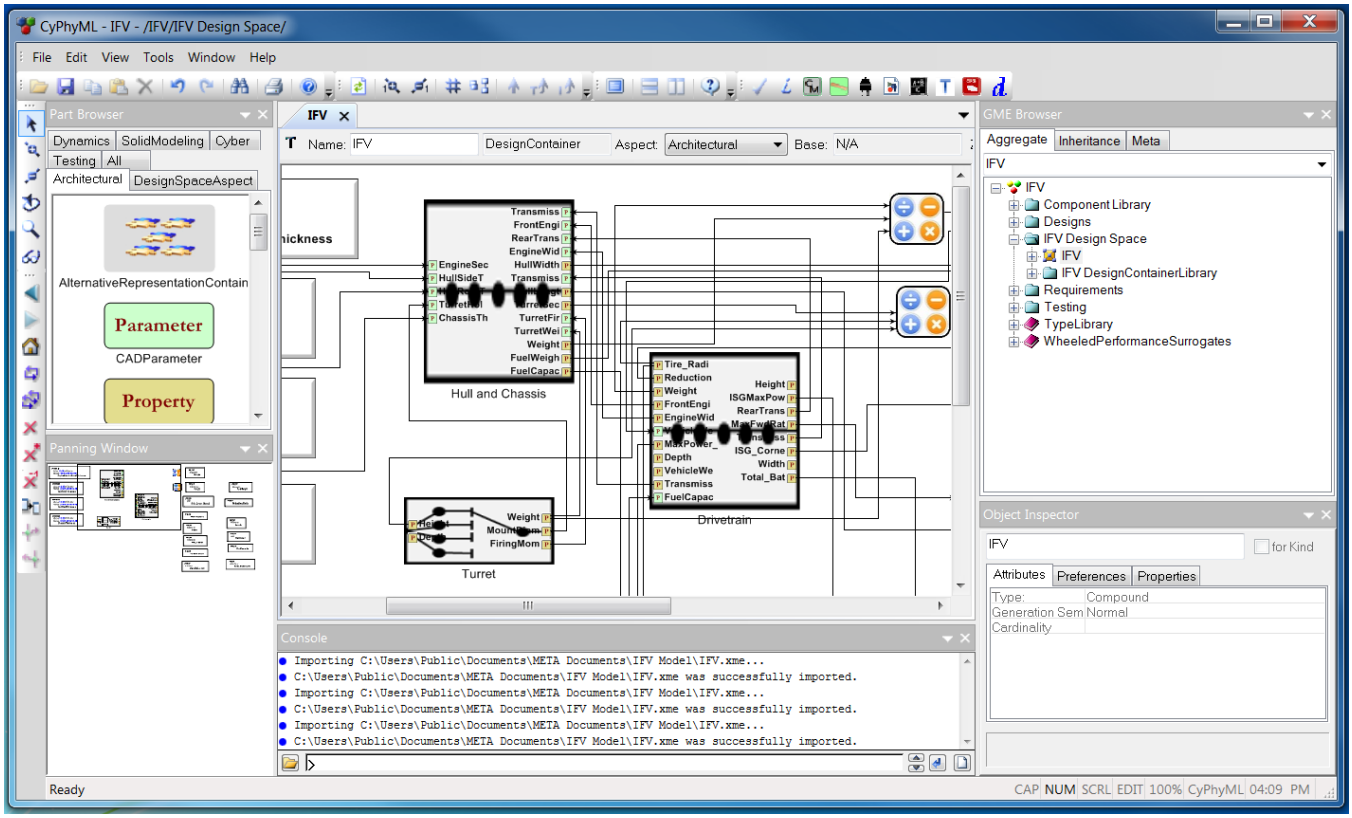

Return to `C13` and switch views by selecting its `SolidModeling` aspect. The `SolidModeling` aspect contains a link to the detailed geometry model (CAD), including references to the non-solid datum features, such as planes, axes, and coordinate systems that define this engine's structural composition rules.



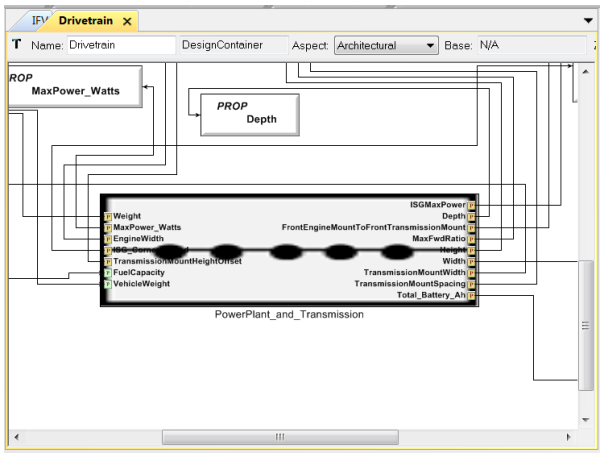
Many other domains are covered by the CyPhy language including: Electrical, Thermal, Software, and Hydraulic.

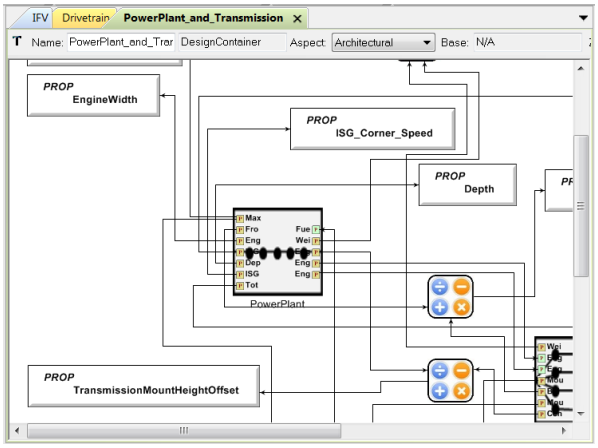
## Design Space Analysis / System Synthesis

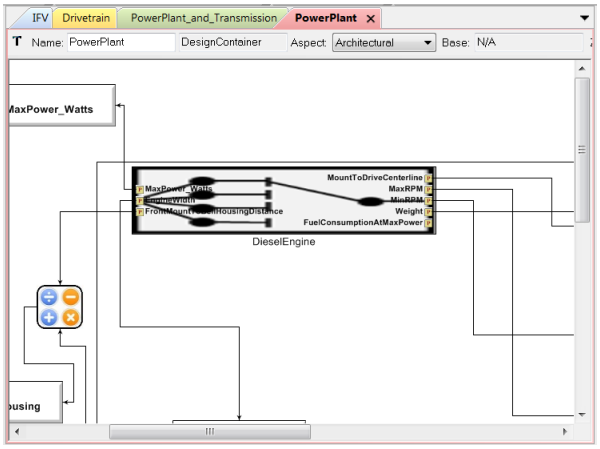
Next, in the `GME Browser`, open the `IFV/IFV Design Space/IFV` model. This model is a `DesignContainer`, showing the top-level `DesignSpace` or the conceptual, architectural decisions of the system under design. The IFV model's system design is divided into sections in the architectural aspect called `Hull and Chassis`, `Drivetrain`, and `Turret`.



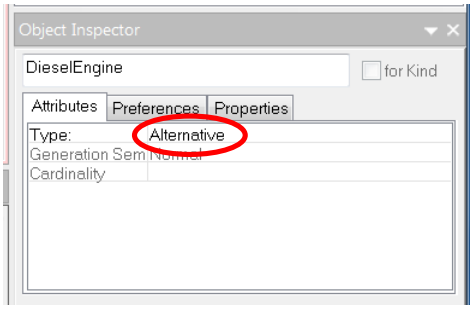

Explore the `Drivetrain` model (by double-clicking on the `Drivetrain` model element) to see the `Drivetrain's` decomposition: `PowerPlant_and_Transmission`, and `Driveline` models. Explore further into the `PowerPlant_and_Transmission` and its `PowerPlant` models. You will see a `DieselEngine` model and an `ISG_Power_Battery` model.

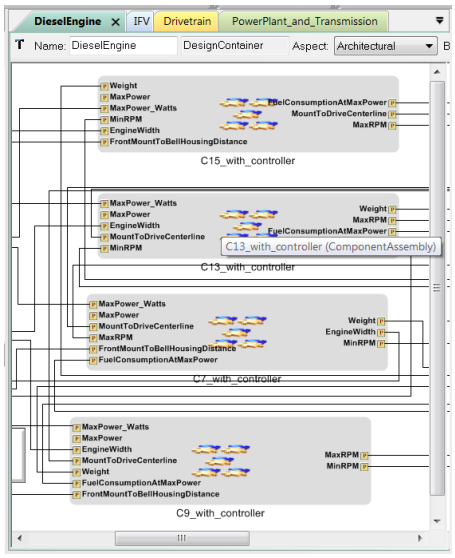


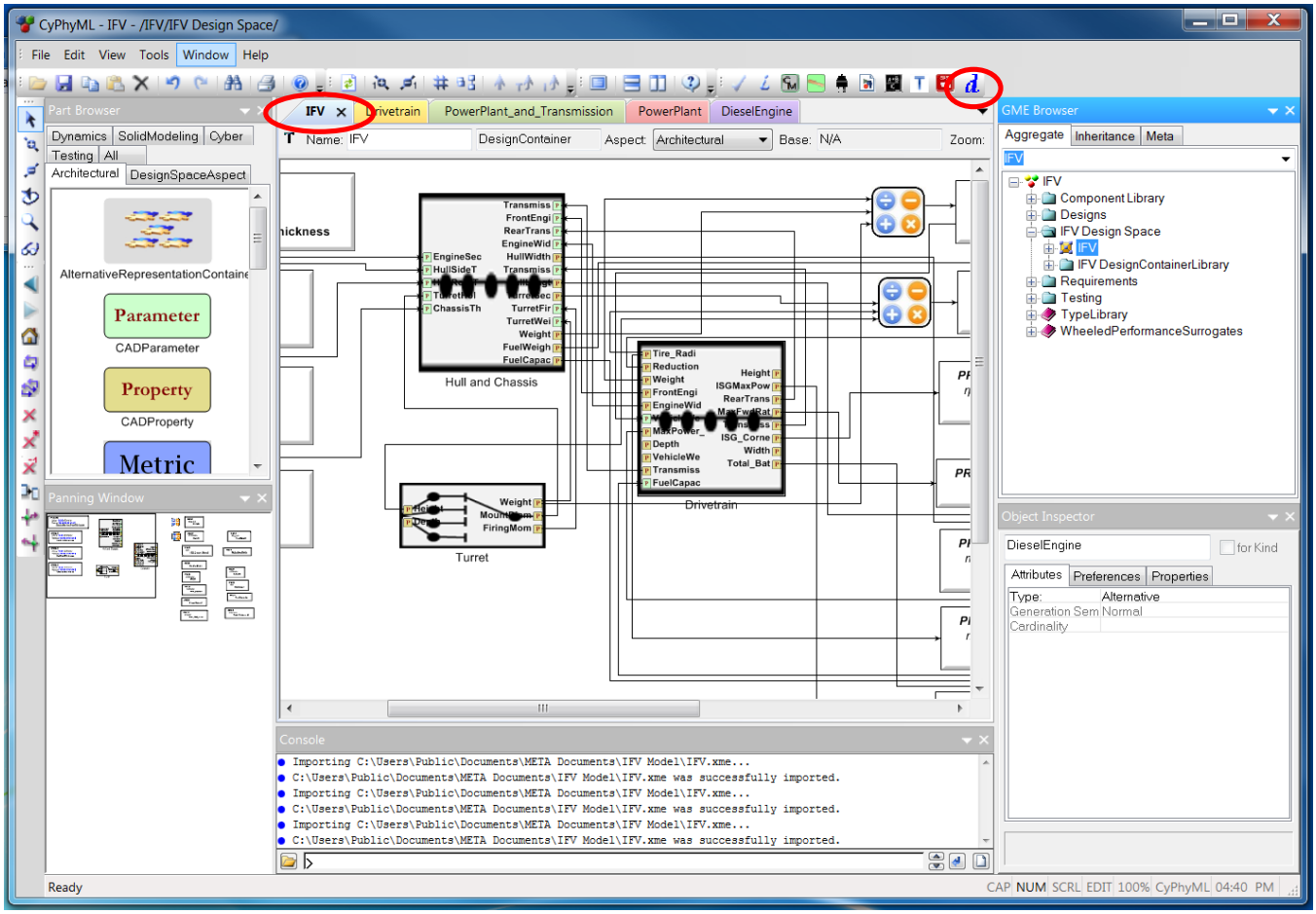




The `DieselEngine` model is an `Alternative` block. Select this model to view its attributes in the `Object Inspector` pane (usually located under the `GME Browser` pane). Explore further into the internal details of `DieselEngine` (*i.e.* double-click) to see the possible candidate components. The semantics of the `Alternative` block dictate that exactly one of the C7, C9, C13, or C15 engines must be selected in a concrete design.



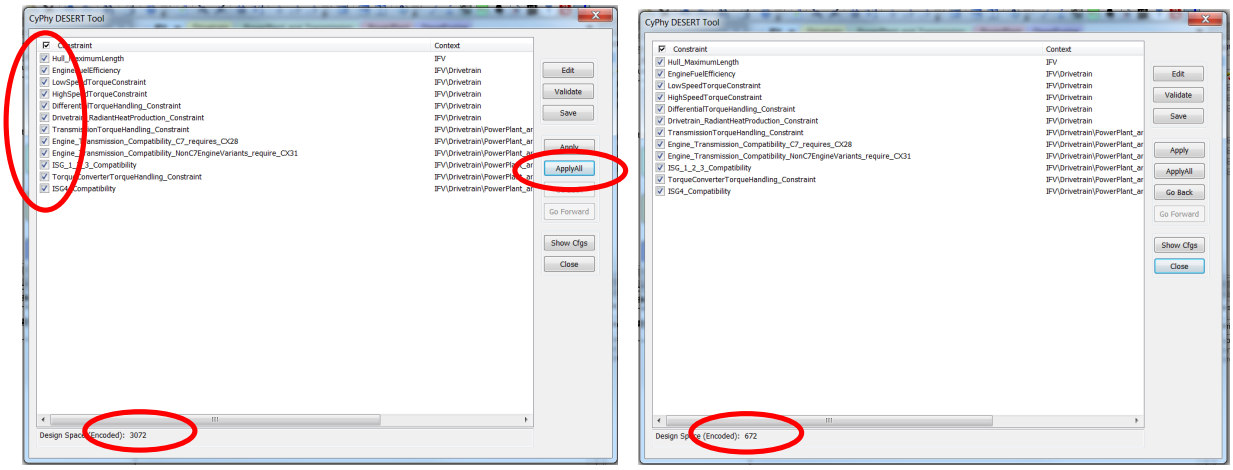



To begin the automated exploration of our family of alternative designs, first navigate back to the top-level `DesignContainer` (*viz.* `IFV/IFV DesignSpace/IFV`). Select the Design Space Exploration Tool[4] toolbar icon ( – upper right area of the application window). This will bring up the CyPhy Design Space Exploration Tool.

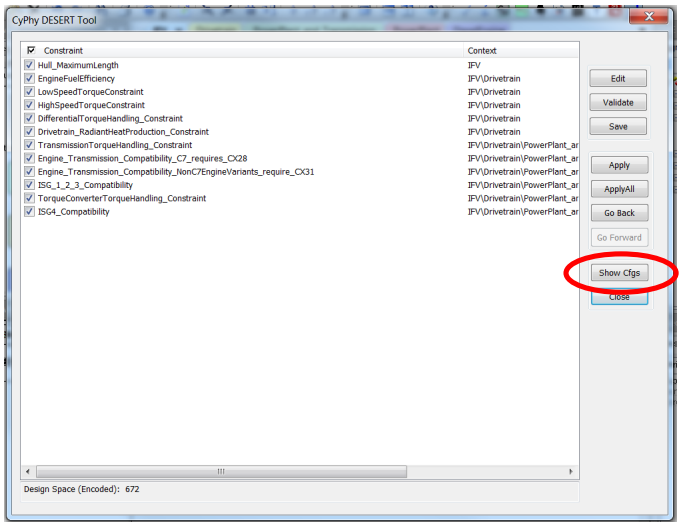



Apply all constraints by clicking `ApplyAll`. (Constraints can be applied individually by checking single constraints.) As constraints are applied, the number of `Design Space Configurations` will diminish (visible in the bottom left of the window).

[4] For more detailed information on the design space exploration tool see the supporting document: *DesignSpaceHelper.pdf.*



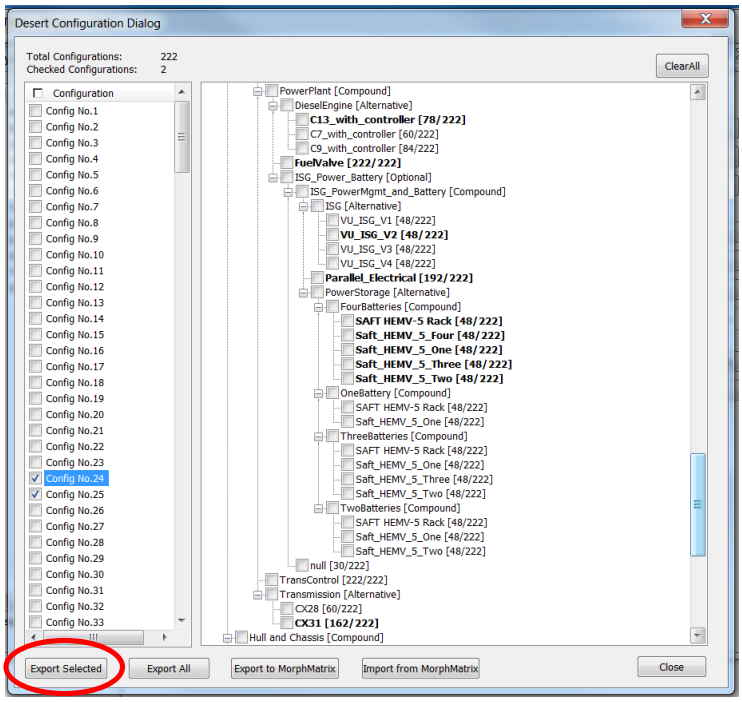

When finished applying constraints, click the `Show Cfgs` button to browse the computed set of valid design configurations (*i.e.* concrete 'point designs'). A number of these point designs (or all) can be exported for further analysis.



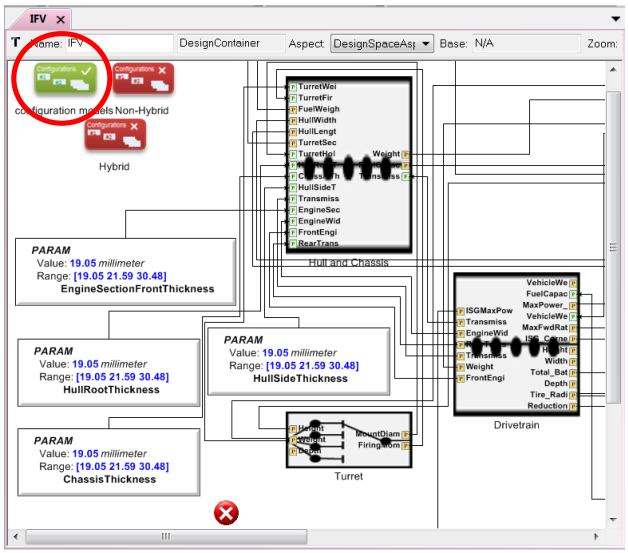

These point designs were computed by the component selection reasoner based on the constraints selected. For example, one constraint that we applied asserts the sum of the lengths of the engine compartment, the turret, and the troop section must not exceed 7.4m. Another example constraint specifies that the fuel consumption at max power for the engine must not exceed 1400 grams/KwH.

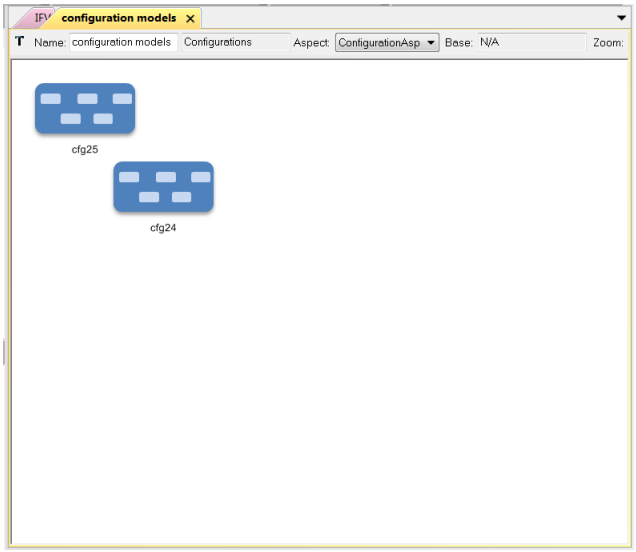
Selecting a configuration on the left (*i.e.* select the label/name of the config, not it's checkbox) will highlight included components on the right. Checking a component on the right will check configurations on the left which include it (*i.e.* this time, select the checkbox of the component rather than selecting it's label/name).

Check the configuration(s) you would like to elaborate and press `Export Selected`. Close this window ***and*** the Design Space Exploration Tool window using the two `Close` buttons, and click `Yes` when prompted to save your selected configurations. Note: Configurations 24 and 25 will be used throughout the examples in this document.

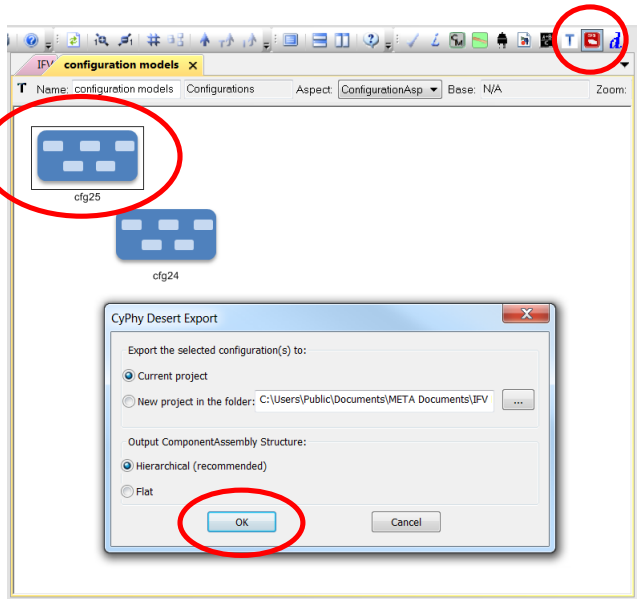



A new model called `configuration models` has been generated and placed in the `DesignSpace` aspect. Open it by double-clicking to view the configurations exported in the previous step.

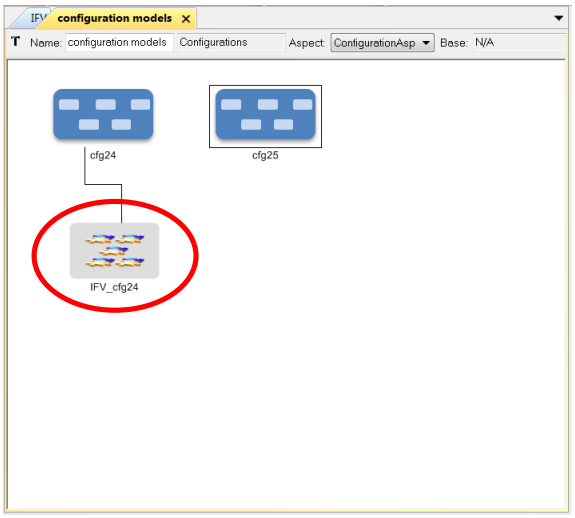


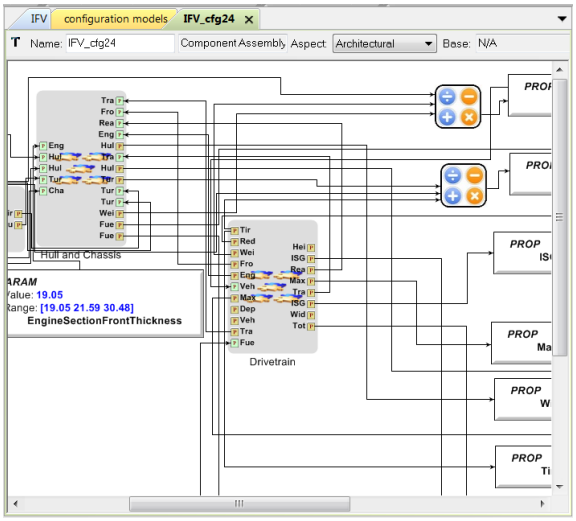



These configurations are simply a bag of components with no structural/assembly information contained within. You may elaborate a particular configuration's structural connections by selecting it and invoking the `ComponentAssembly` interpreter by pressing the  interpreter button. You may elaborate multiple configurations simultaneously by selecting multiple models before invoking this interpreter (*e.g.* CTRL-A selects *all* configs). Press `OK` on the resulting dialog. The default options are fine.




The assembled configuration will be attached to its unassembled configuration. Double click the newly assembled configuration to explore it. You can now browse the fully elaborated design, ready for analysis.



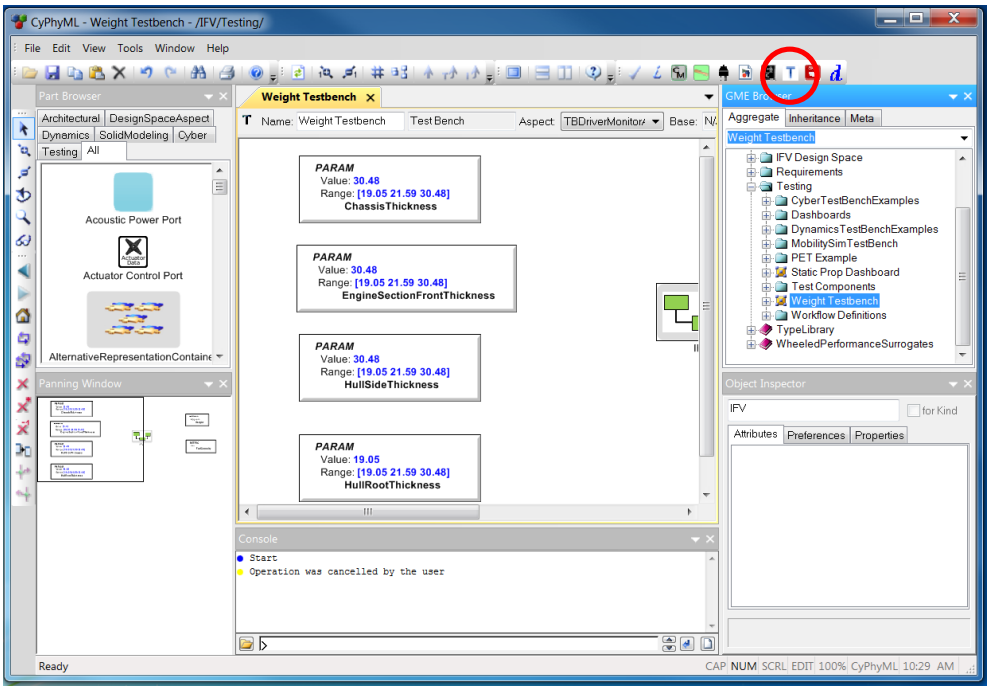



## Static Analysis

`Testbench` models are used to capture testing environments for individual components, component assemblies, and for aggregate composed systems. Components or Component Assemblies that have inputs, outputs, or parameters of interest to a Multidisciplinary Design Analysis and Optimization (MDAO)[5] process may be referenced within `Testbench` models. These references should point to the particular instances of those Components or Component Assemblies of interest. In CyPhy terminology, we call these references the `System Under Test`. Additionally, `Testbench` models may define `Test Drivers`, which are used to inject stimuli into the system (*e.g.* throttle position), as well as `Test Monitors` that transform outputs from the system into metrics that can be evaluated at analysis time (*e.g.* instantaneous speed). In the context of a `TestBench`, a `Parameter` may be varied by an MDAO process, and a `Property` is an "output" metric that would be examined by that process. This applies to parameters and properties of system components, and extends to test drivers (*i.e.* parameters of the drivers are altered and affect the stimuli applied) and to test monitors as well (stimulus received by the monitors from the system are distilled into Properties).

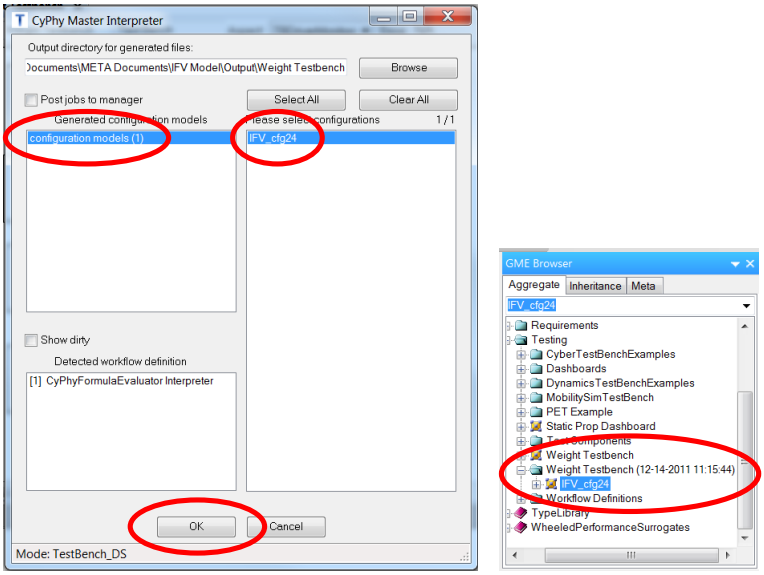
Open the test bench model: `IFV/Testing/Weight Testbench`.

Run the `Formula Evaluator` by selecting this icon:  to execute the `Weight Testbench`.

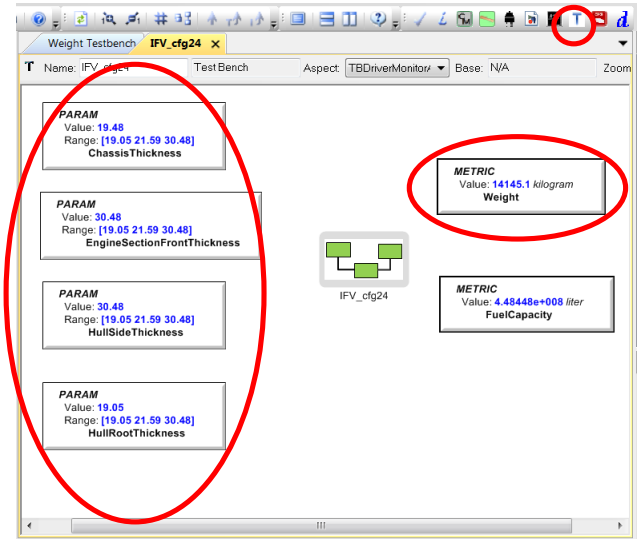



[5] See Also: C:\Users\Public\Documents\META Documents\Modelica and PET March 2012

First select an item under `Generated configuration models`. Then select one or more configurations. Click the `OK` button. The results are computed and stored in a Folder adjacent to the Weight Testbench.




Open the newly generated weight bench configuration by double clicking it in the `GME Browser`. Using the `Object Inspector`, alter the hull thickness values on the left (within the range of 19.05-30.48) and run the  again to see the weight change[6].

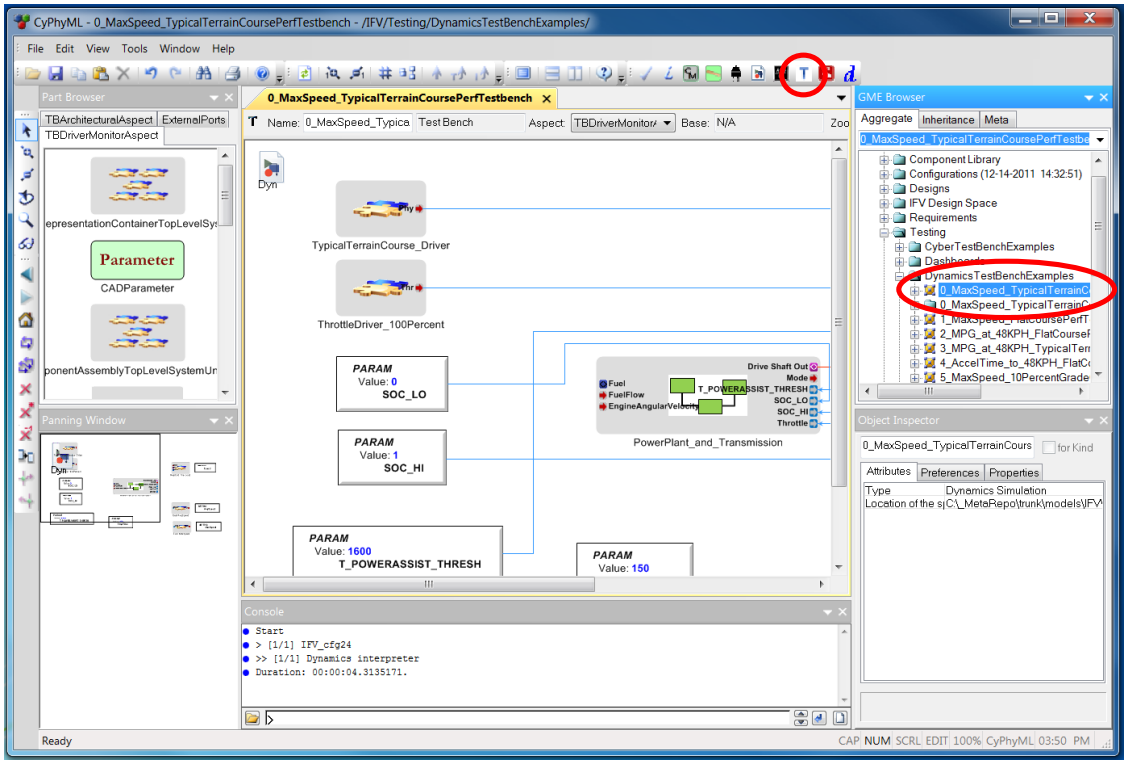



[6] See Appendix A for information about resetting the default model values.

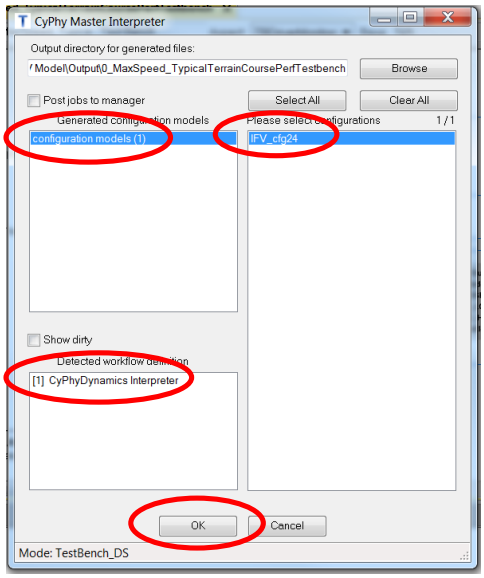
## Dynamics Analysis

Next, we will evaluate the performance of the composite powertrain assembly in the context of a test bench that will supply the stimulus under which we want to perform some analysis. In the following case, we have built a test bench scenario that drives throttle command and supplies a varying torque load on the powertrain to simulate operation over various terrains. The powertrain operation is monitored and recorded (engine torque, engine rpm, battery V/I, ISG voltage, etc.).

Open `Testing/DynamicsTestBenchExamples/0_MaxSpeed_TypicalTerrainCoursePerfTestbench` and run the `Master Interpreter`  .




The `Master Interpreter` shows that the `CyPhyDynamics Interpreter` will be run. Select an item under `Generated configuration models`, then select one or more configurations under `Please select configurations` and select `OK`.

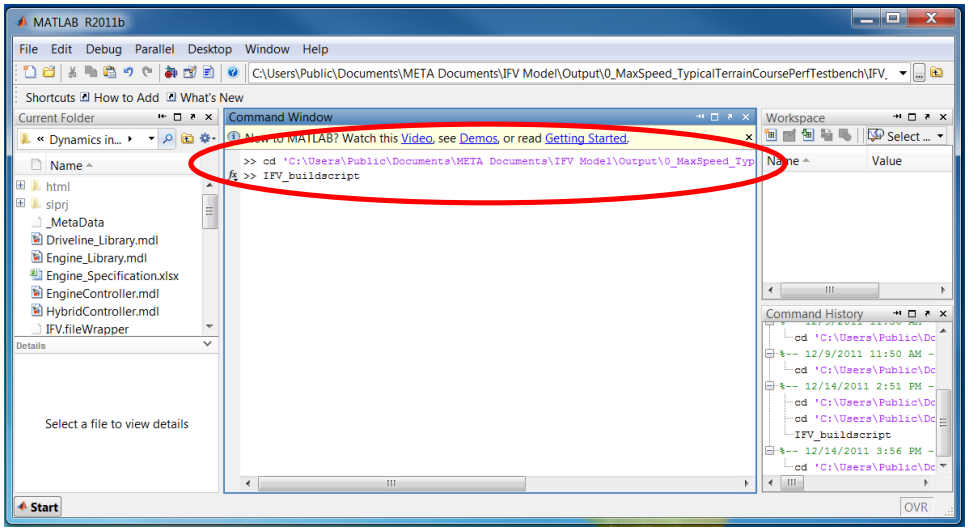



Open MATLAB[7].

Right-click Run the `IFV_buildscript.m` found in the following directory:

`C:\Users\Public\Public Documents\META Documents\IFV Model\Output\0_MaxSpeed_TypicalTerrainCoursePerfTestbench\IFV_cfg##*\Dynamics interpreter`

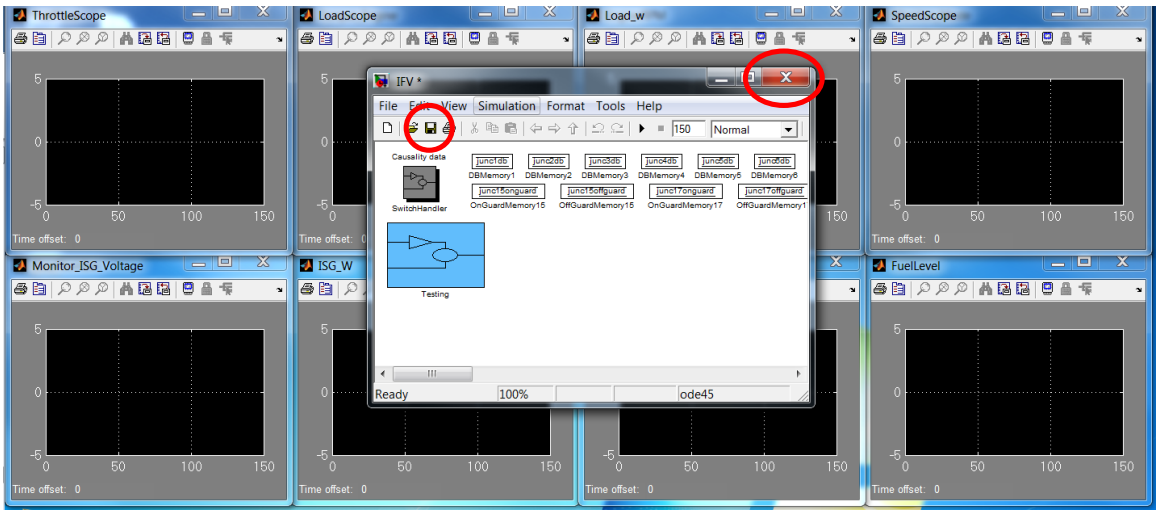
** obviously, changing IFV_cfg## to [one of] the configurations number you chose previously*

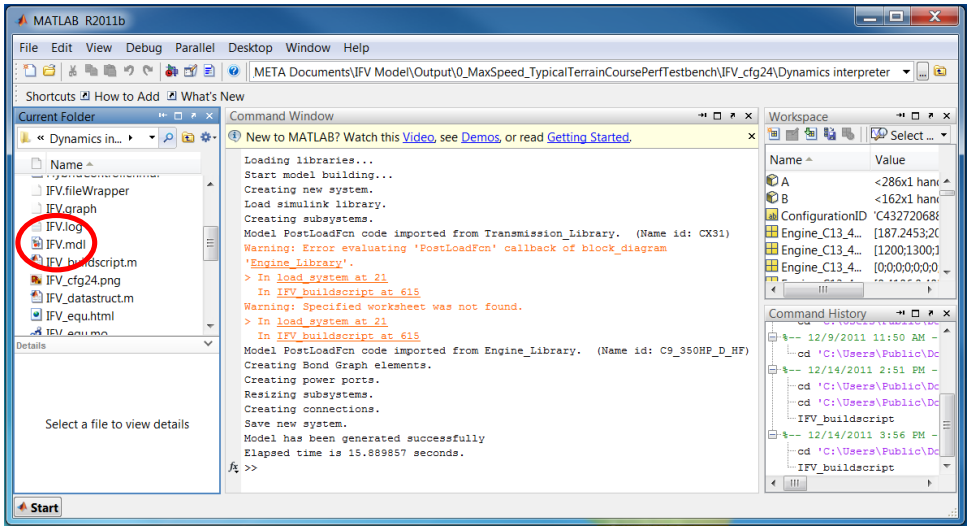



[7] Many of the behavioral models for physical components in the *IFV* example are represented as BondGraphs, which the tools ultimately transform to MATLAB/Simulink models for simulation. If you do not have a MATLAB license, there is an alternative Dynamics example that utilizes only OpenModelica as its underlying simulation engine.
`C:\Users\Public\Public Documents\META Documents\Modelica and PET March 2012\Demo Instructions.pdf`.

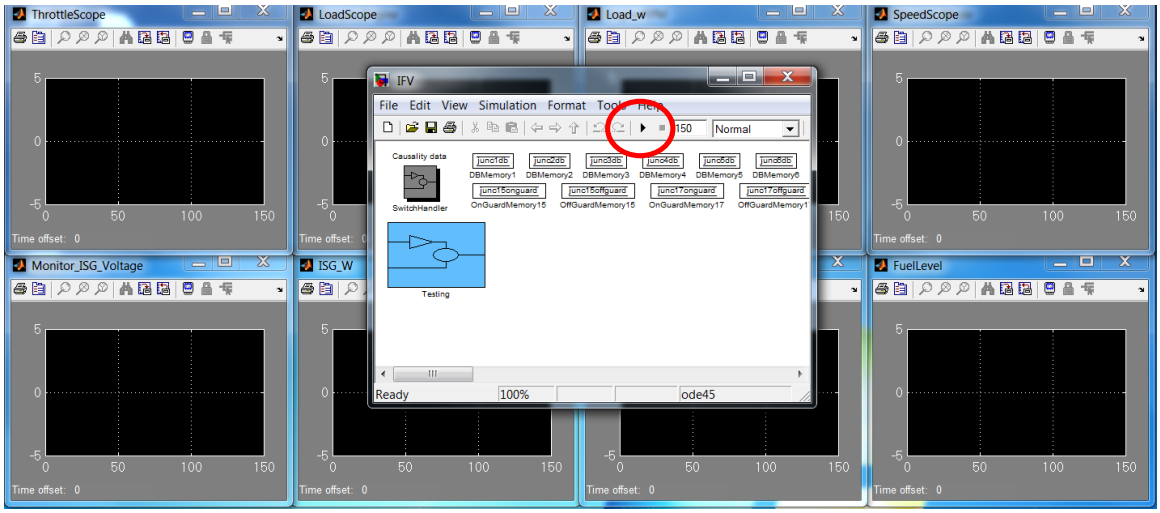
 

The script generates a MATLAB simulation of each particular point design's power train. Click the save button, and close the model.

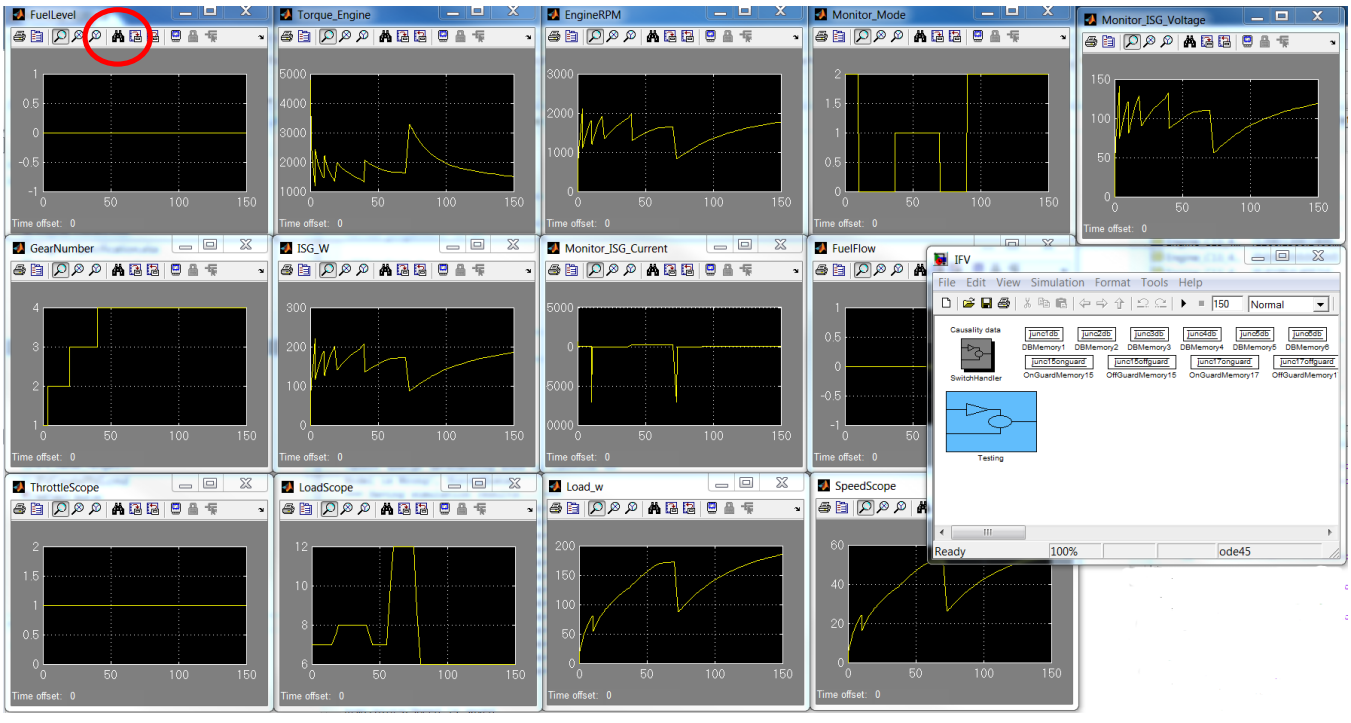



Use the ▶ button to run the simulation model that appears.



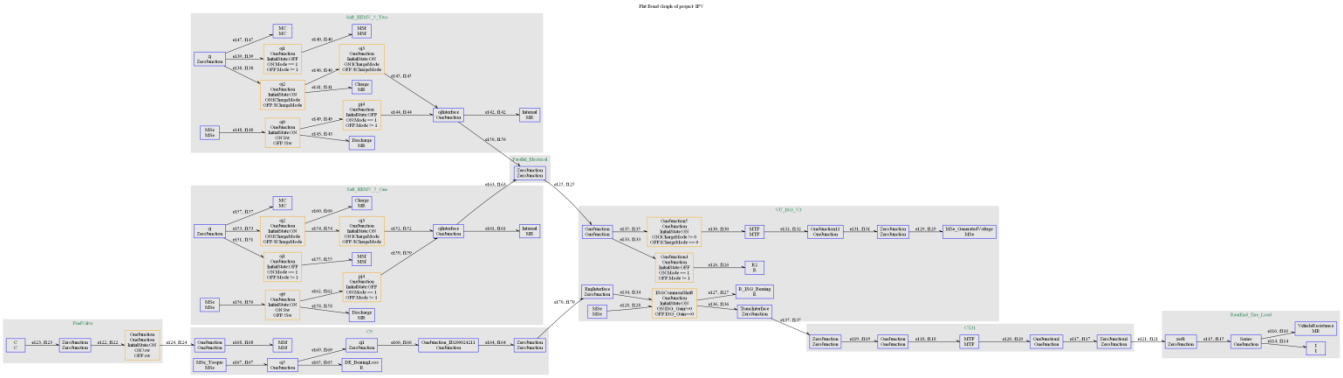

Click on the button to auto scale each scope. (Some graph windows may be hiding behind others.) This experiment shows different variables of the system during a specified scenario. The power demand of the diesel engine and the hybrid mode changes over time. The hybrid modes are as follows: charging the battery, discharging the battery, and the battery is disconnected.



Navigate to the following folder using Windows Explorer (changing IFV_cfg## to the configuration number you elaborated).

```
C:\Users\Public\Public Documents\META Documents\IFV
Model\Output\0_MaxSpeed_TypicalTerrainCoursePerfTestbench\IFV_cfg##\Dynamics interpreter
```

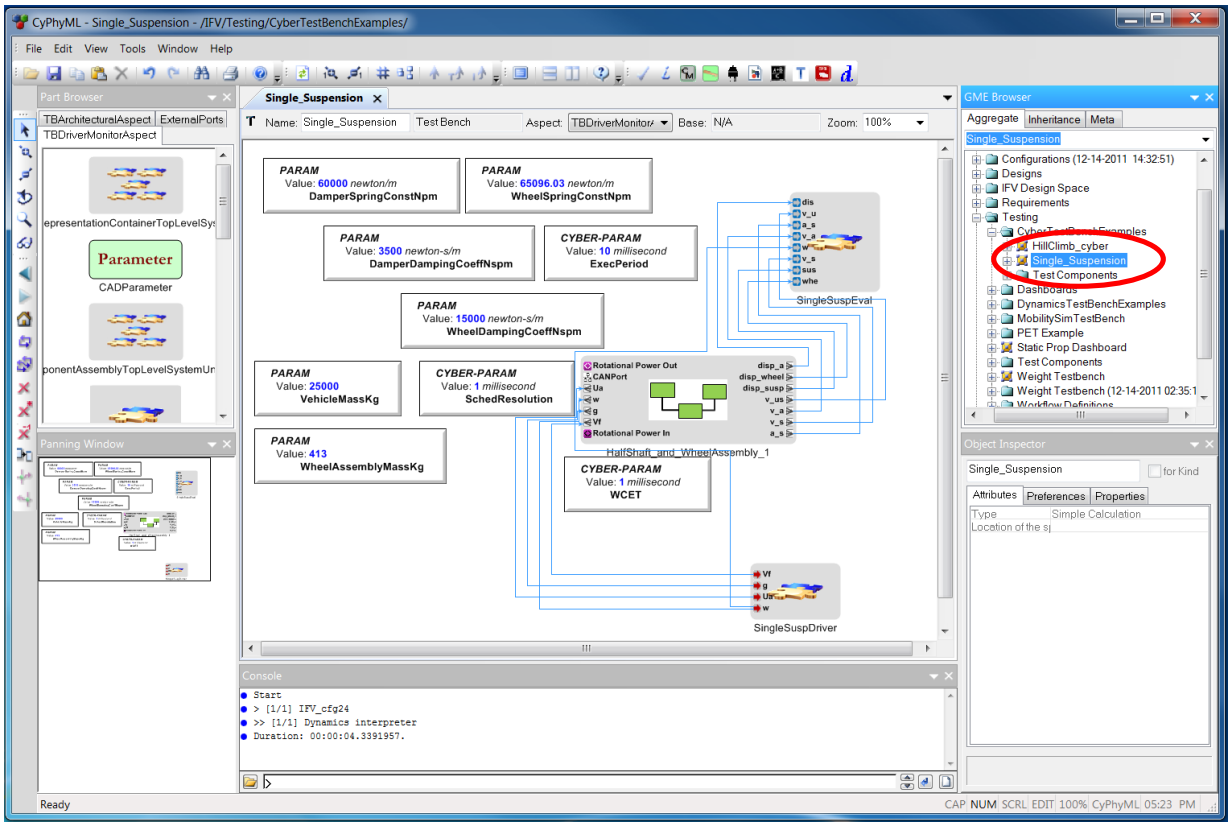
Run the script `IFVCreatePNG.cmd` by double clicking it to generate the composed bond graph. It will appear in the same directory named `IFV_cfg##.png`.



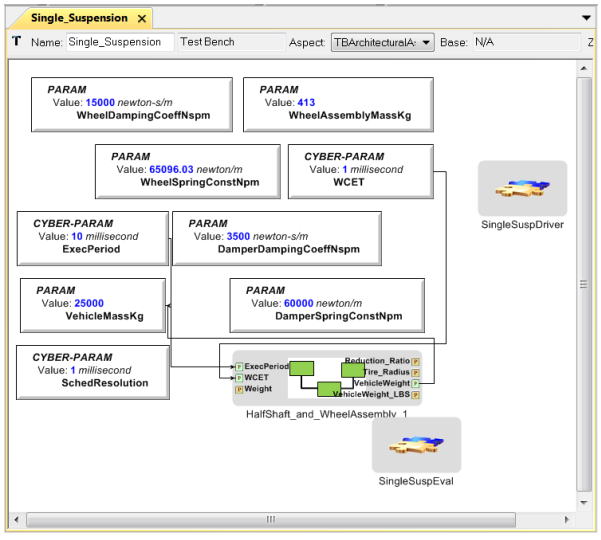
## Cyber Controller Evaluation

Cyber controllers can be evaluated in the context of a test bench. Execution parameters can be varied to determine their effects on the dynamics. Cyber assemblies are tested dynamically using Simulink together with an add-on called TrueTime, which extends Simulink with blocks representing processor scheduling and network communications to investigate the effects of a computational platform on the controller design. Just as in the Dynamics examples above, we use Simulink to explore model behavior. The generated Simulink model will also contain TrueTime blocks to represent the architecture and deployment of the computer implementation of the controller. For this example, we will use a single-wheel suspension controller that damps vibrations from the wheel to the vehicle body. The suspension test bench is defined with respect to a design container, which represents all possible valid configurations of the model.

Open `IFV/Testing/CyberTestBenchExamples/Single_Suspension`. Note the addition of the `Cyber-Param` objects in the model: `SchedResolution` is a global parameter that determines the size of the scheduling ticks that will be used to calculate the real-time schedule offsets (note: changing the global scheduling resolution may break the example). `ExecPeriod` specifies the sample rate of the real-time components, and `WCET` keeps the worst-case execution time for the suspension controller. `ExecPeriod` and `WCET` are specified per-component and then exposed as ports in the model hierarchy.




Changing aspects to `TBValueFlowAspect` shows the hierarchical parameter connections from the test bench into the wheel assembly components as well as the parameter connection from the wheel assembly to the total vehicle mass parameter, as shown below. The vehicle mass will have to be calculated automatically for a particular configuration in order to get the right suspension dynamics for that configuration.



The model interpreters relevant to the cyber example are as follows:

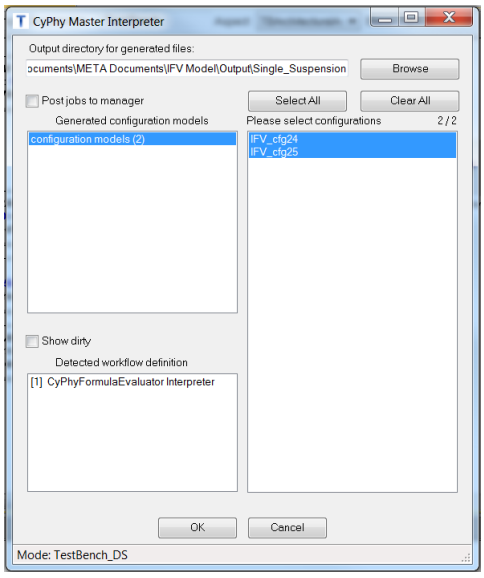
1.  `Design space exploration` – Apply constraints. Explore the resulting design space. Select specific configurations. Export and save them in the model.
2.  `Generate component assembly` – Create/assemble models from the exported configurations.
3.  `Formula Evaluator` – Compute Properties based on Parameters and Formulas.
4.  `Master Interpreter` – Create test benches for specific configurations.
5.  `CyPhy Dynamics Interpreter` – Check `Use ESMoL Implementation` to generate a `TrueTime` cyber model. Leave unchecked to build the controller model with the ideal Simulink-only controller (no `TrueTime`).

For the example, we assume that steps 1 & 2 have already been run as described in previous sections of this document. Also, be sure to run 3 on the component assemblies[8]. We will use the remaining steps to create test benches for specific configurations and generate a Simulink model for evaluation.

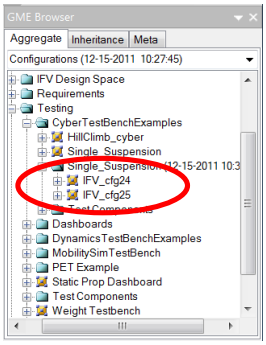
Step 3: Navigate to the `component assembly`  generated in step 2. Then run the `Formula Evaluator` by pressing the  button.

Step 4: From the `Single_Suspension` model, invoke the `Master Interpreter`  . Select the configurations that you wish to evaluate. Click `OK`.

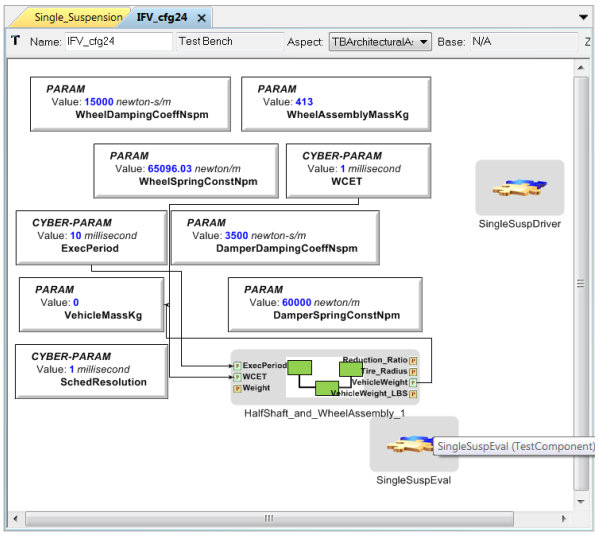
[8] Not always strictly necessary with more recent versions of the software. While useful for visualizing the structure of particular point designs, most downstream analysis tools will automatically elaborate point designs whenever necessary. In this case, we need to preprocess the point designs by computing the weight of the entire vehicle and not just the weight of the portion of system under test, however.



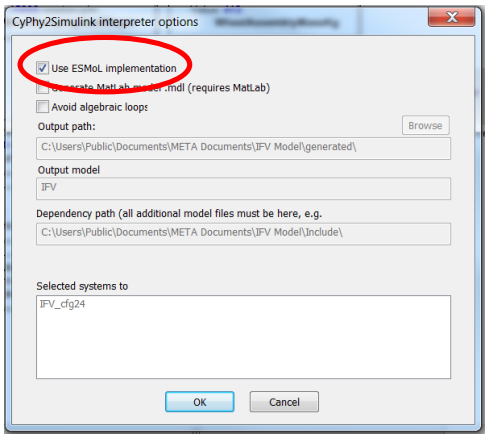

A new folder is created with a current time stamp. Within the folder are newly created test benches for the selected configurations.




Opening one of the test benches shows the same structure as the `Single_Suspension` test bench. The component in the center is no longer a design container. Instead, it is a reference to a wheel assembly in the specific point design chosen.




Step 4. Press the CyPhy Dynamics Interpreter button to create the simulation model  . Select `Use ESMoL Implementation` to generate the ESMoL cyber controller model (includes use of `TrueTime`).



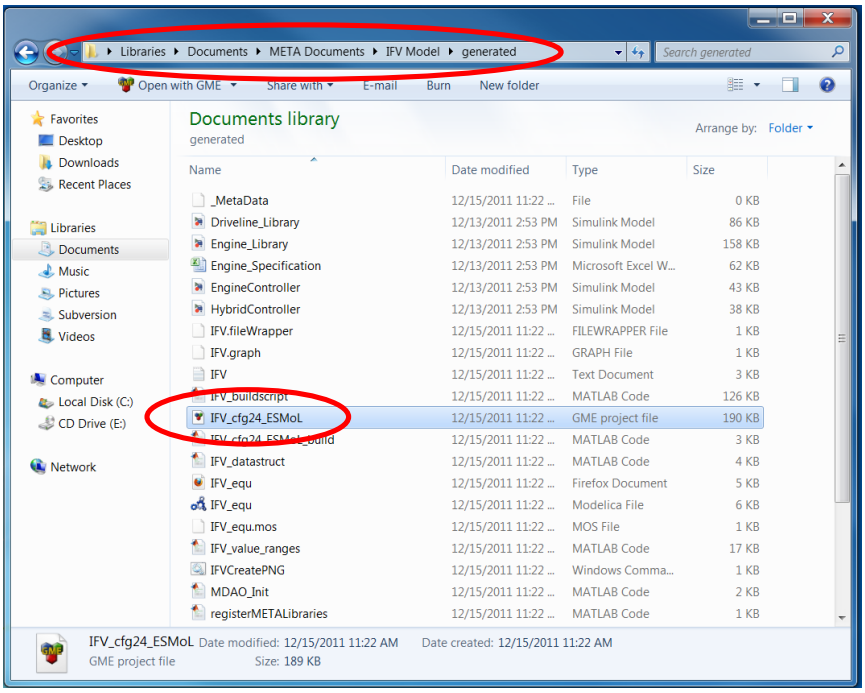

The interpreter will create a new directory called `generated` in the same location as the `IFV.mga` model file, which contains (among other things) the following artifacts:

- Library mdl-files: Simulink files from the Include directory are copied into the generated directory. The Simulink build script will take driver and evaluation subsystems from those library models to build the full model.
- MDAO_Init.m: script to populate the MATLAB workspace with the proper parameter values (taken from the model). This script is not called directly, as it is called by the model build script.
- Generated IFV_cfg##_ESMoL.mga file: The ESMoL translator creates a model representing the cyber controller and its hardware deployment.

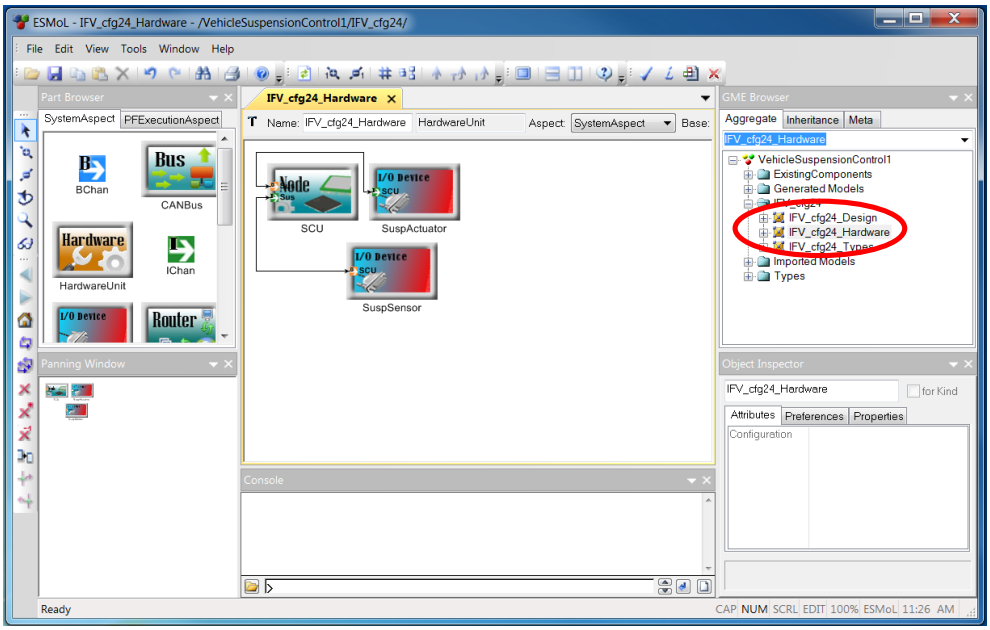
 

- IFV_buildscript.m: MATLAB script to build the final Simulink model.

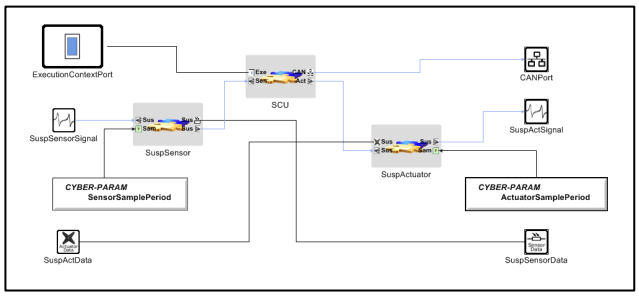
Open the generated ESMoL model (`IFV_cfg##_ESMoL.mga`).

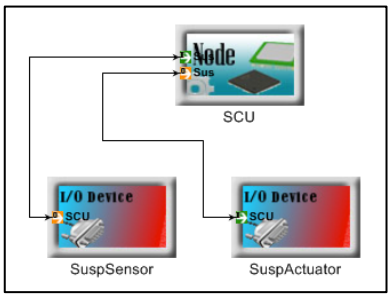



Within the model, navigate to `VehicleSuspensionControl1/IFV_cfg##/IFV_cfg##_Hardware` and open the model by double clicking it.

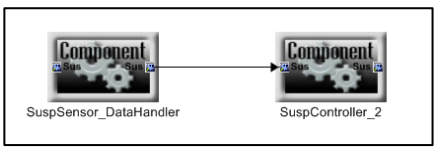


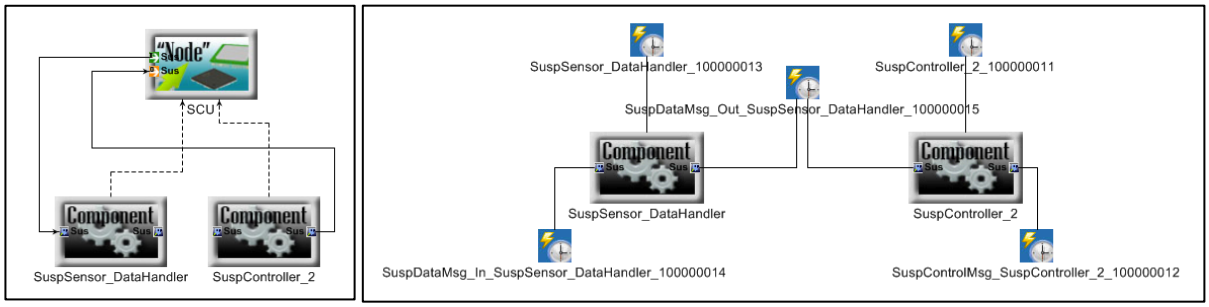

The hardware elements in the CyPhyML suspension component hardware model (left) are translated to the ESMoL platform model (right).




The remaining software component dataflow, deployment, and timing parameters are created in the ESMoL model from the CyPhyML model. The different aspects of the model at RootFolder/IFV_cfg7/IFV_cfg7_Design contain these details (aspects shown left to right below, `LogicalArchitectureView`, `DeploymentView`, and `ExecutionView`). The `AutoLayout plugin` can rearrange the models within any aspect.

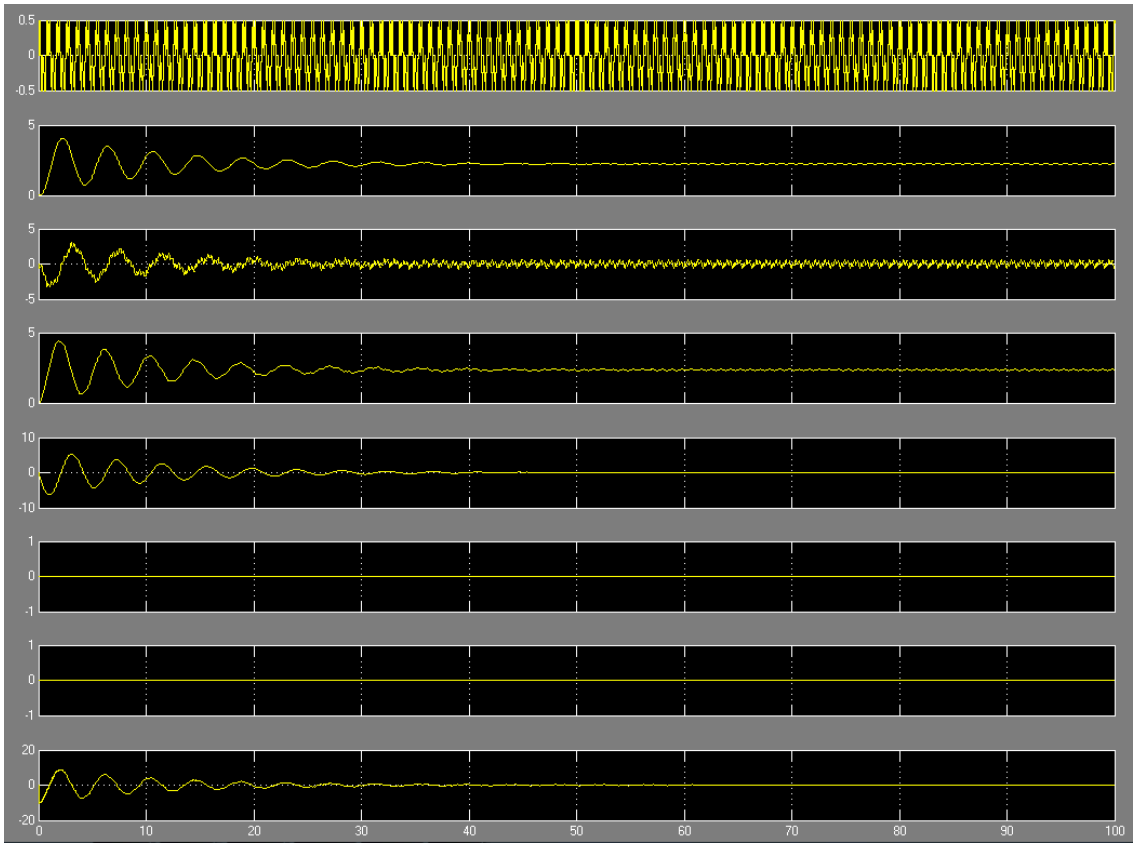





Run the `mex –setup` command in MATLAB and select a visual studio compiler. This step is only necessary once.

To create the TrueTime model set the current directory of MATLAB to 'generated' and run IFV_buildscript.m from the MATLAB command line (or right-click and run from within Windows Explorer). This will create the Simulink model and open it. After running the Simulink model, open the following path in Simulink to see the results:

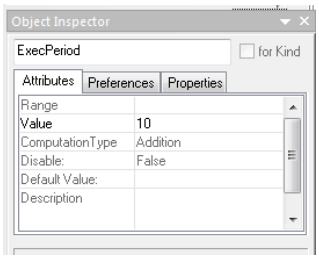
```
Testing/CyberTestBenchExamples/Suspension__<date stamp>/IFV_cfg7/SingleSuspEval/SuspEvalBG/
SuspEvaluator/Scope3
```

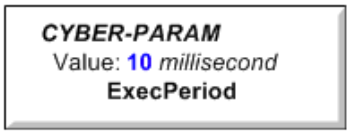


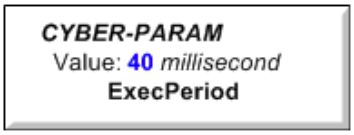

From top to bottom, the scope traces include

1. Road velocity input (input)
2. Wheel displacement
3. Wheel vertical velocity
4. Suspension displacement
5. Suspension velocity
6. Unused
7. Unused
8. Body acceleration (result)

Currently, exploration of the controller parameter space can only be performed manually. For example, from the test bench for configuration 7 changing the execution period will change the response. Change the Value attribute from 10 to another value (40 is good).



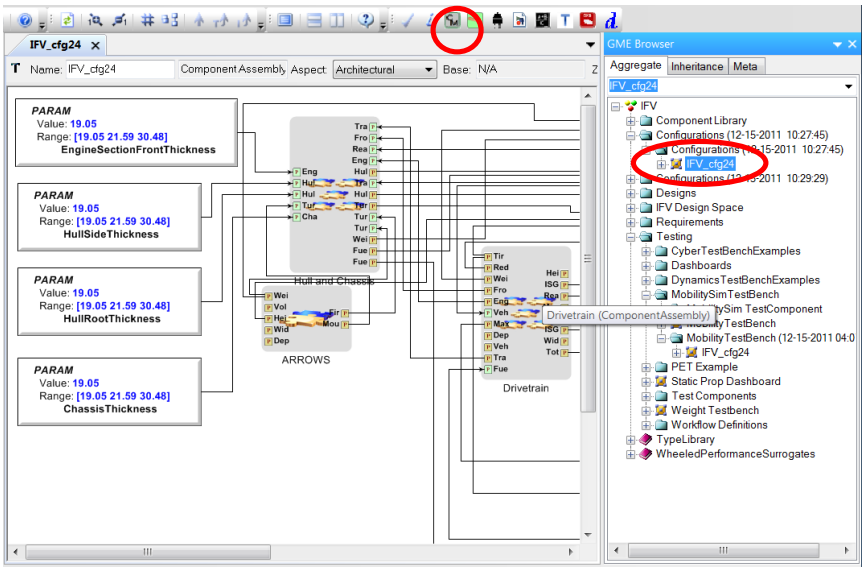

From this point, delete (or move) the 'generated' directory (exit MATLAB and close the ESMoL model first). Then re-run the CyPhyML2Simulink interpreter  and repeat the generation and evaluation process.

## CAD Synthesis and Analysis

CyPhy Design Configurations can be automatically translated into an integrated CAD assembly. Synthesize a CAD assembly from a Design Configuration by selecting the Design Configuration you want to synthesize from the following:

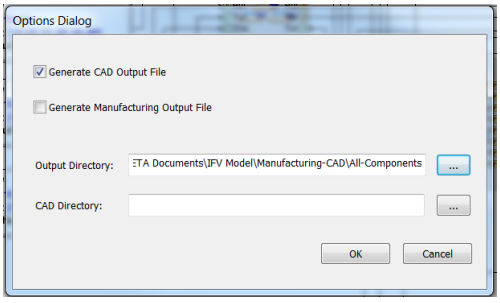
```
IFV/Configurations([timestamp])/Configurations([timestamp])/IFV_cfg##
```

Then invoke the CyPhy2CAD interpreter by clicking the CM interpreter icon  .



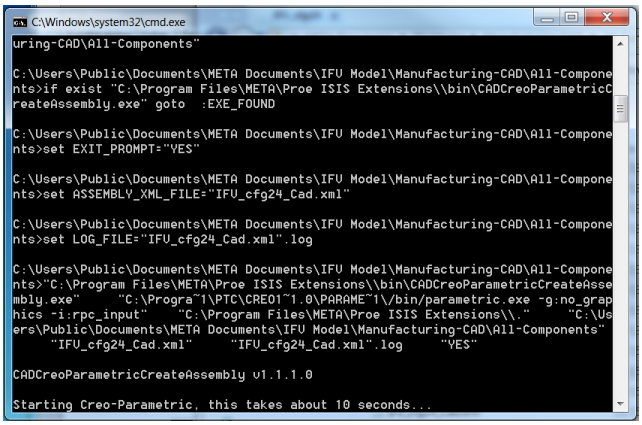

Select the `Generate CAD Output File`. Set the `Output Directory` generated CAD files to `C:\Users\Public\Documents\META Documents\IFV Model\Manufacturing-CAD\All-Components`.

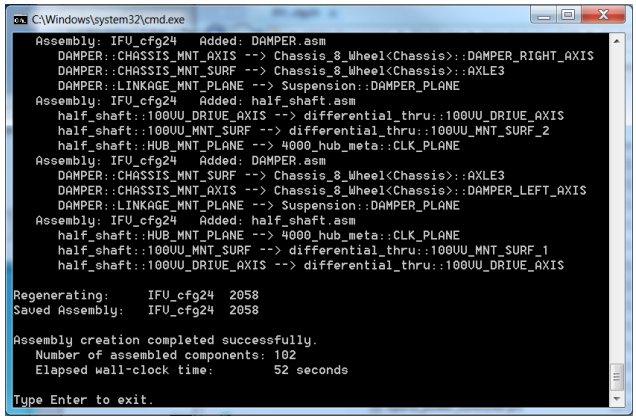
Click `OK`.




Open a Windows Explorer window and navigate to the directory selected above. Locate the generated `IFV_cfg##_Cad.bat` file.

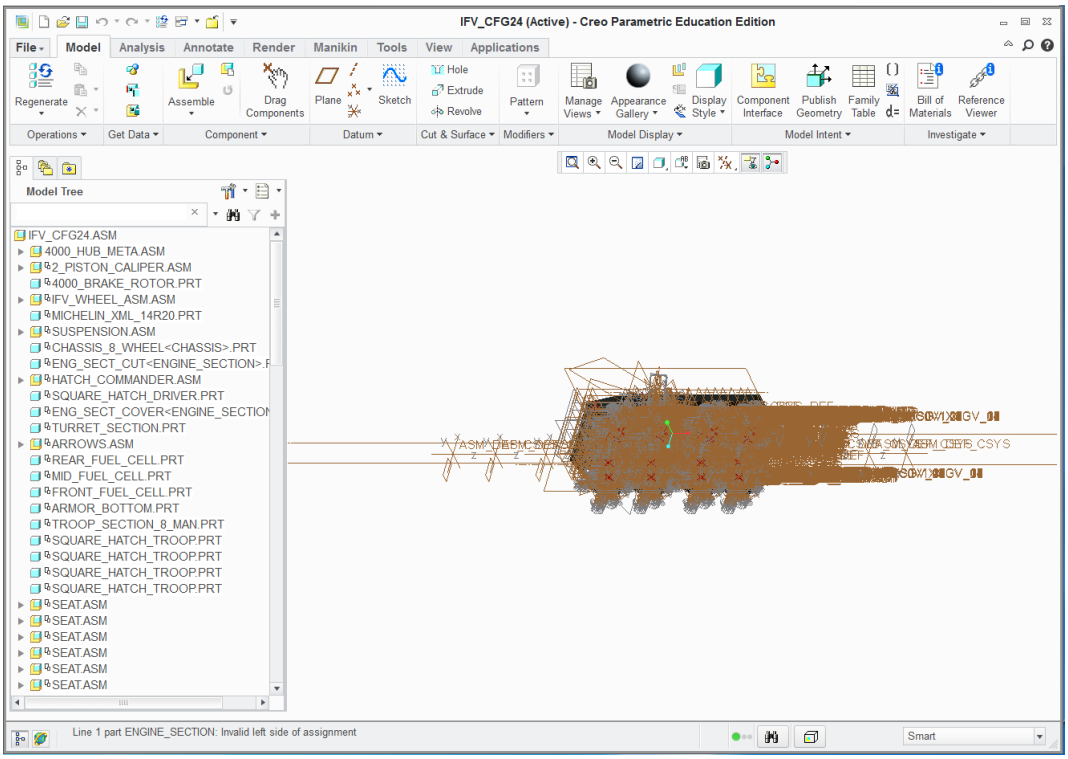
Double-click on this file to execute.



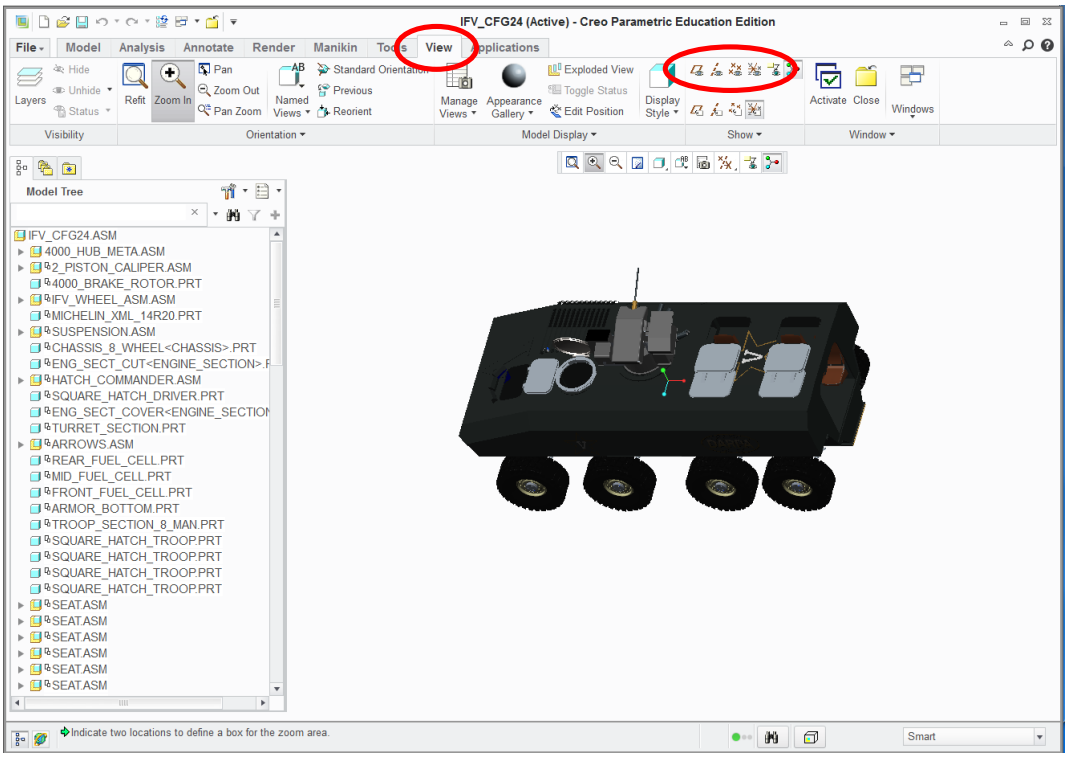

This process produces a vendor-agnostic XML-based description of the vehicle's geometry, which is then used to drive Creo to automatically build the CAD assembly model. (You may ignore any errors about missing material definitions for this demo.)

To view the generated assembled CAD model, start the Creo Parametric application, and open the generated CAD model: ( `File→Open C:\Users\Public\Public Documents\META Documents\IFV Model\Manufacturing-CAD\All-Components\ifv_cfg##.asm`).

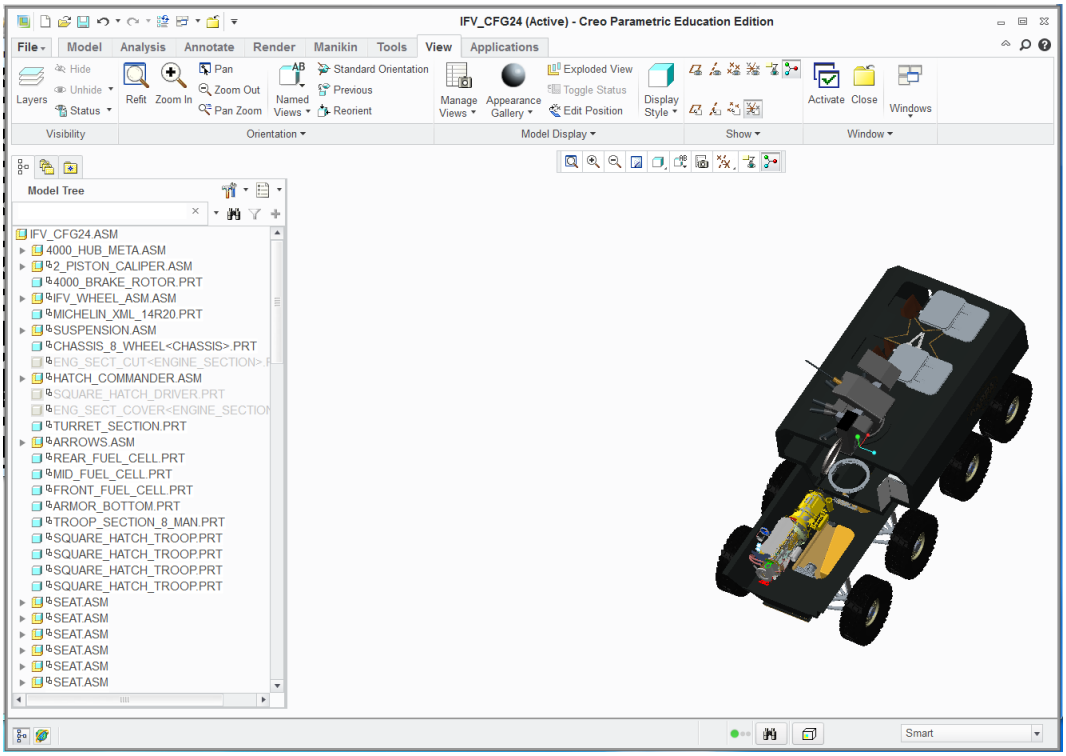



To better visualize, deselect the five `Display` toolbar buttons in the upper right on the `View` tab (see figure below). At this point, the 3D CAD model of the generated vehicle should be visible.

You can use the mouse scroll wheel to zoom; press and hold scroll wheel to change orientation.




Parts can be hidden to expose the internal parts for viewing. For example, right click on: `ENG_SECT_CUT_<ENGINE_SECTION>.PRT`, `SQUARE_HATCH_DRIVER.PRT`, and `ENG_SECT_COVER<ENGINE_SECTION.PRT` from the Model Tree and select `Hide` to hide the part.



## Finite Element Analysis (FEA)

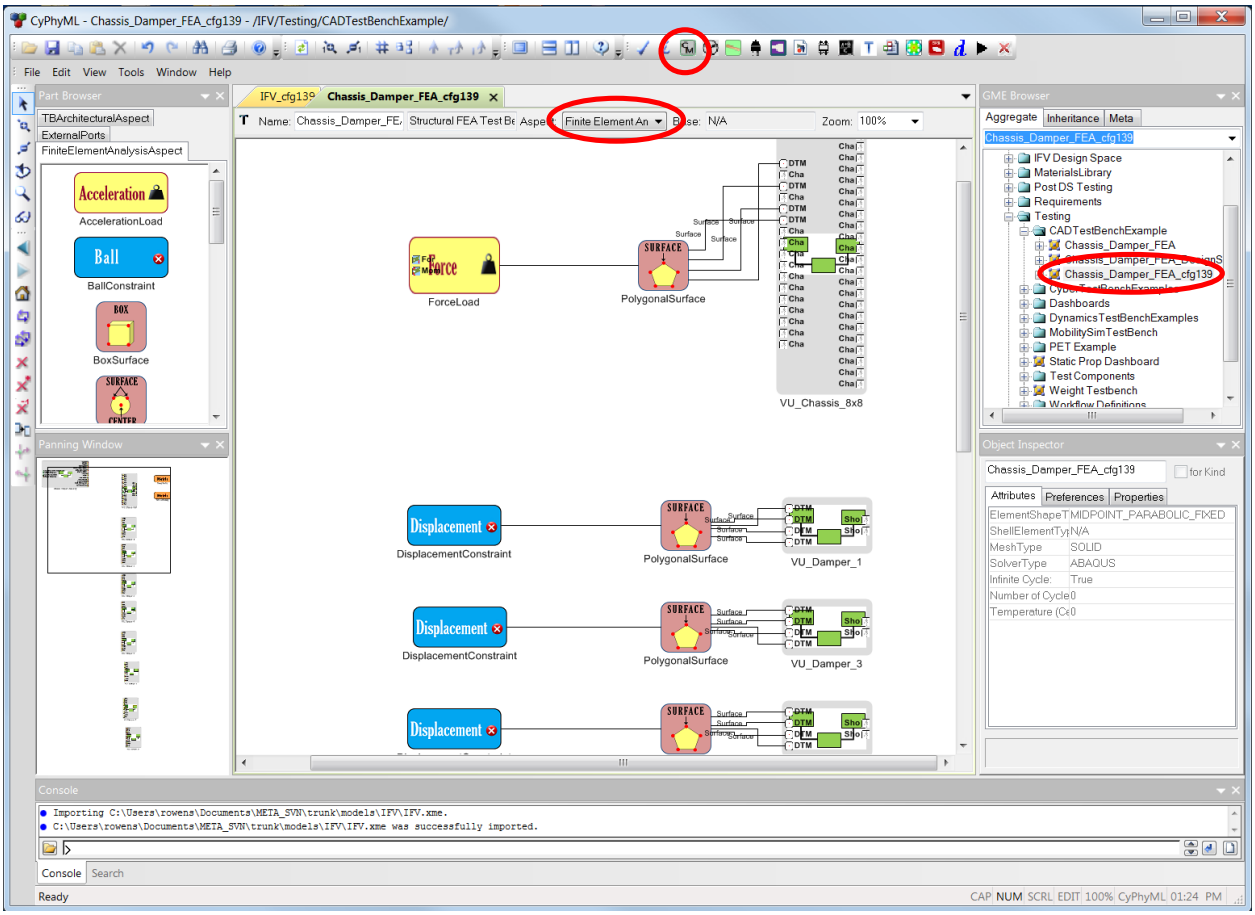
An FEA CyPhy test bench consists of a definition of a mechanical assembly as well as a definition of loads and constraints applied to the mechanical assembly. The FEA process entails exporting--via the CyPhy2CAD interpreter--the assembly definition, plus loads and constraints to an xml file, invoking a program that builds a Creo assembly and creates an FEA deck, and finally invoking a program that submits the deck to a FEA solver.

First select the test bench you want to synthesize. For this example, select the following:

```
IFV/Testing/CADTestBenchExample/Chassis_Damper_FEA_DesignSpace
```

Run the Master Interpreter, choosing the configurations you want to run. This step will create a time-stamped folder with the name of your selected configurations. Open the selected configuration and click the  icon.

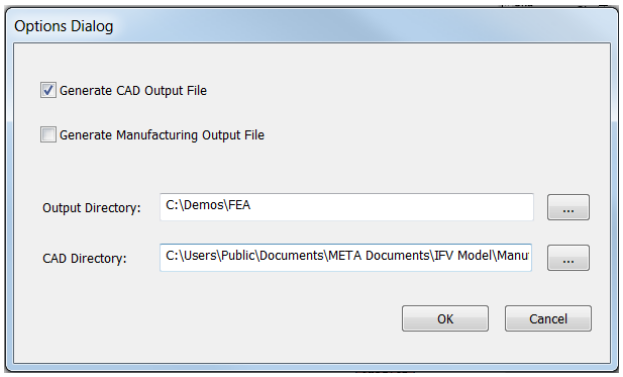
Then select the `Finite Element Analysis` aspect and again invoke the CyPhy2CAD interpreter by clicking the CM interpreter icon  .



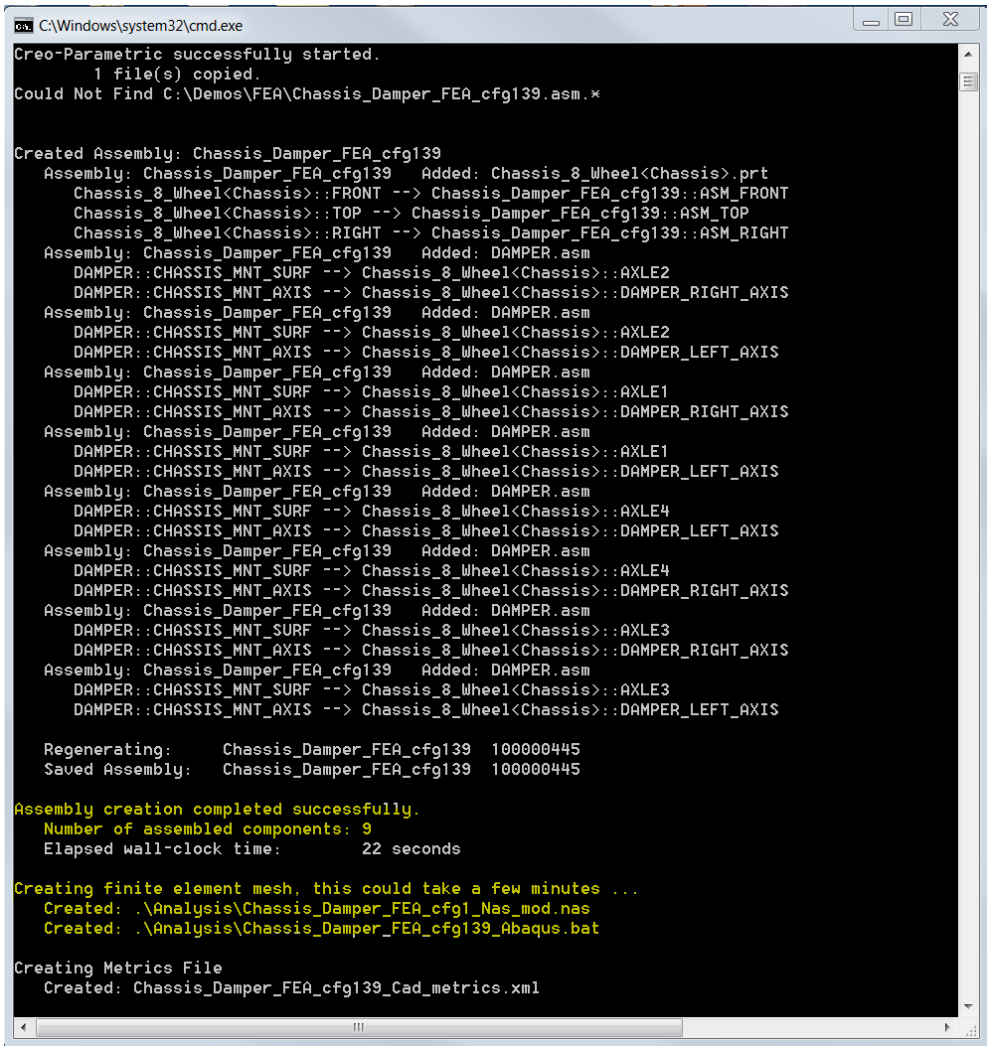
Per the following screen shot, select the `Generate CAD Output File`. Set the `Output Directory` to a directory of your choosing (*e.g.* `C:\Demos\FEA`).

Set the `CAD Directory` to `C:\Users\Public\Documents\META Documents\IFV Model\Manufacturing-CAD\FEAComponents`.

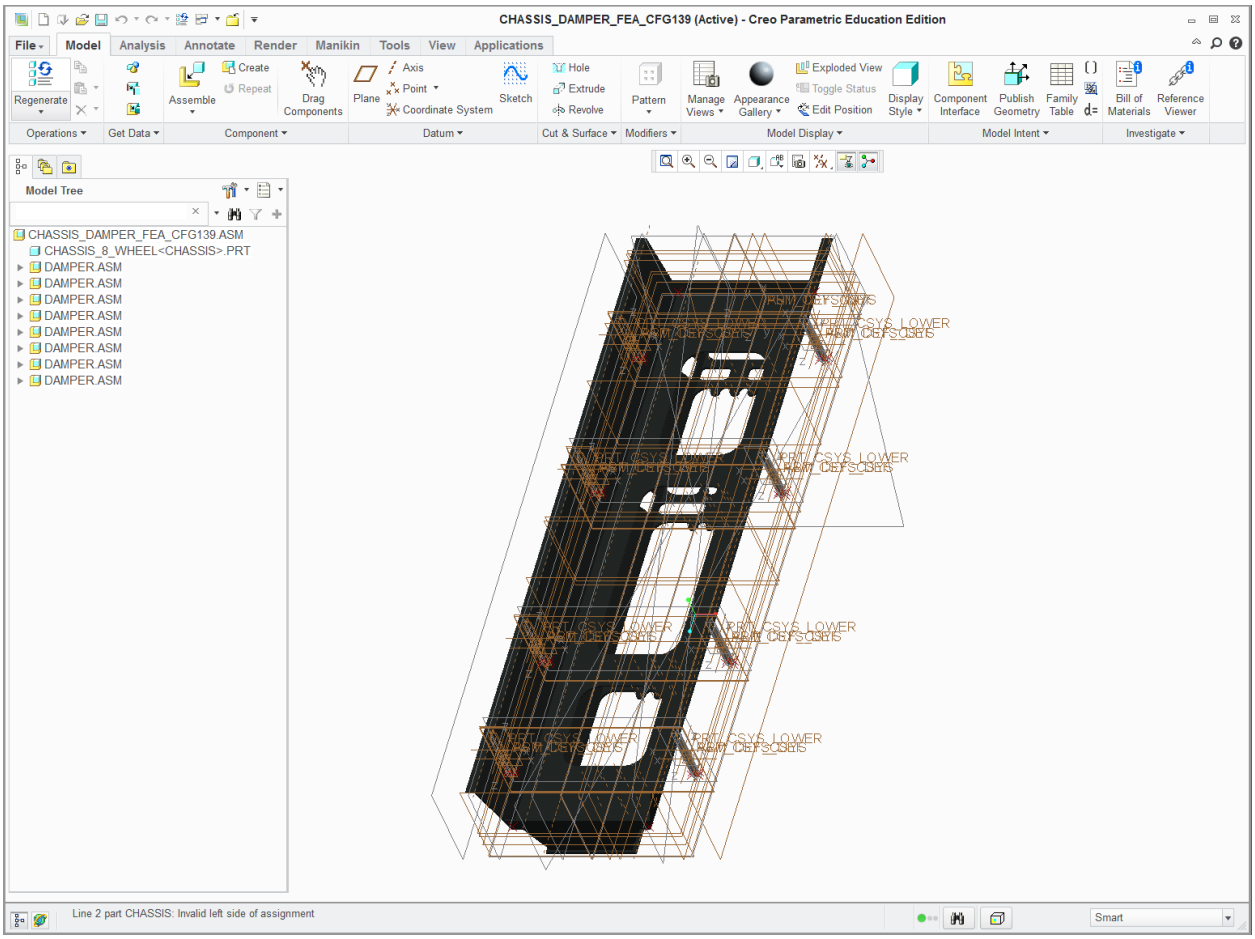
Press `OK`.



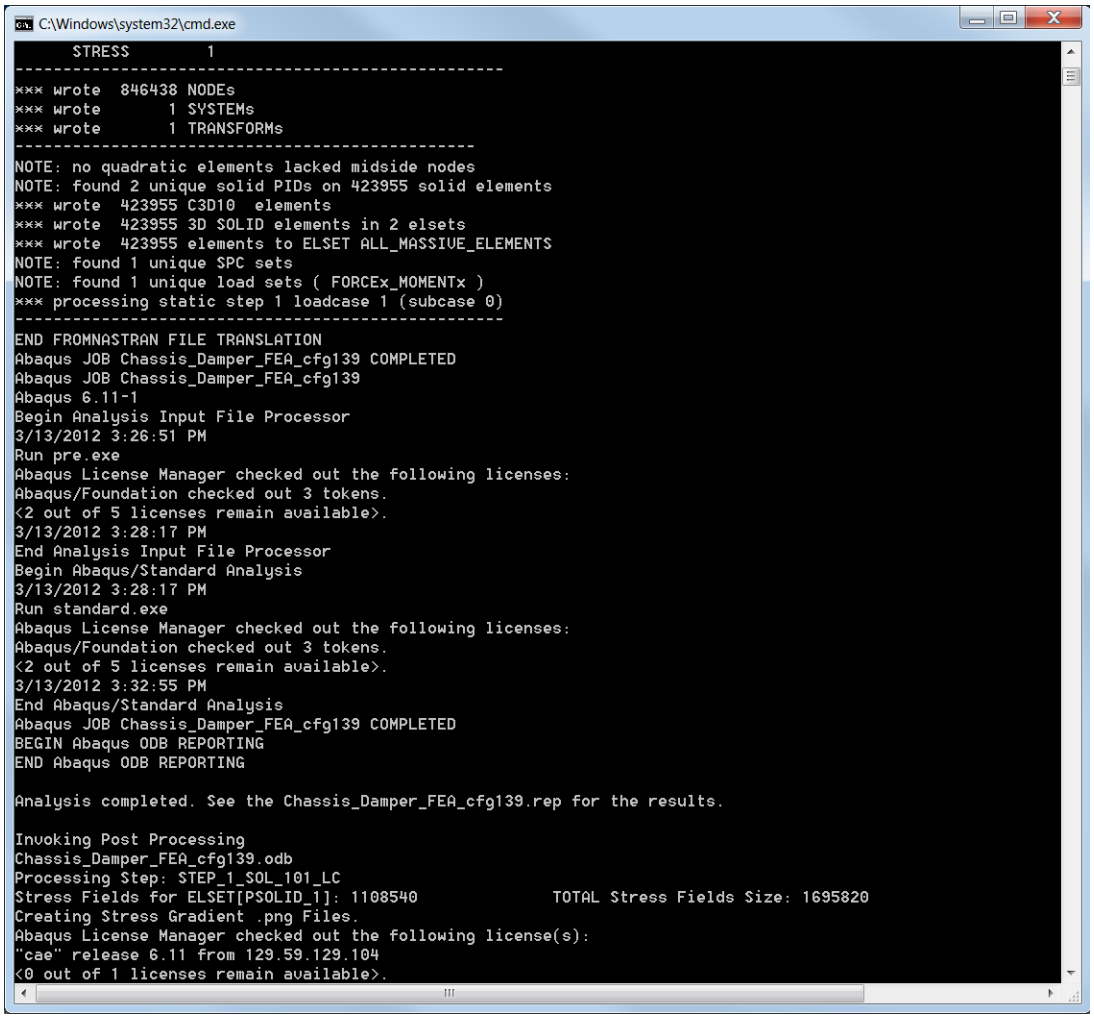

Open a Windows Explorer window and navigate to the `Output Directory` entered above. Locate the generated `CreateCADLinks.bat` file, right click on it and select `Run as administrator`. This will generate links to the Creo files in the `CAD Directory` specified above. Next, locate the file `IFV_cfg##_Cad.bat` and Double-click on it to execute the program that builds the Creo assembly and FEA deck. A Creo license is required for this step. A screen shot of the command window follows:



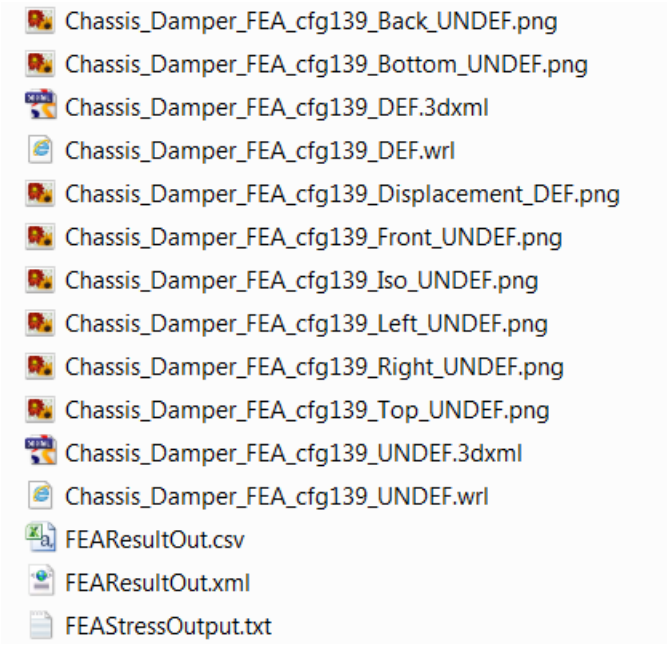

For the FEA process, it is not necessary that you view the created Creo assembly; however, if you wish to do so, start the Creo Parametric application, and open the generated CAD model ( `File→Open <Output Directory>\IFV_cfg##.asm`). A screen shot of the assembly follows:




Next, invoke the FEA solver. An Abaqus license is required for this step. With windows explorer, navigate to the `<Output Directory>\Analysis\Abaqus` directory and double click on the `IFV_cfg##_Abaqus.bat`. This takes approximately ten minutes to run. A screen shot of the command window follows:




The FEA results are contained in the `<Output Directory>\Analysis` directory as shown in the following screen shot. The `FEAStressOutput.txt` contains the maximum stress values and factors-of-safety. The `FEAResultOut.xml` contains information that was requested by the CyPhy test bench and that may be forwarded to the `Dashboard` analysis-results viewer. The .png, wrl, and 3dxml files show the stress gradient and deformed shape plots.



## Mobility (+Dynamics) Testbench

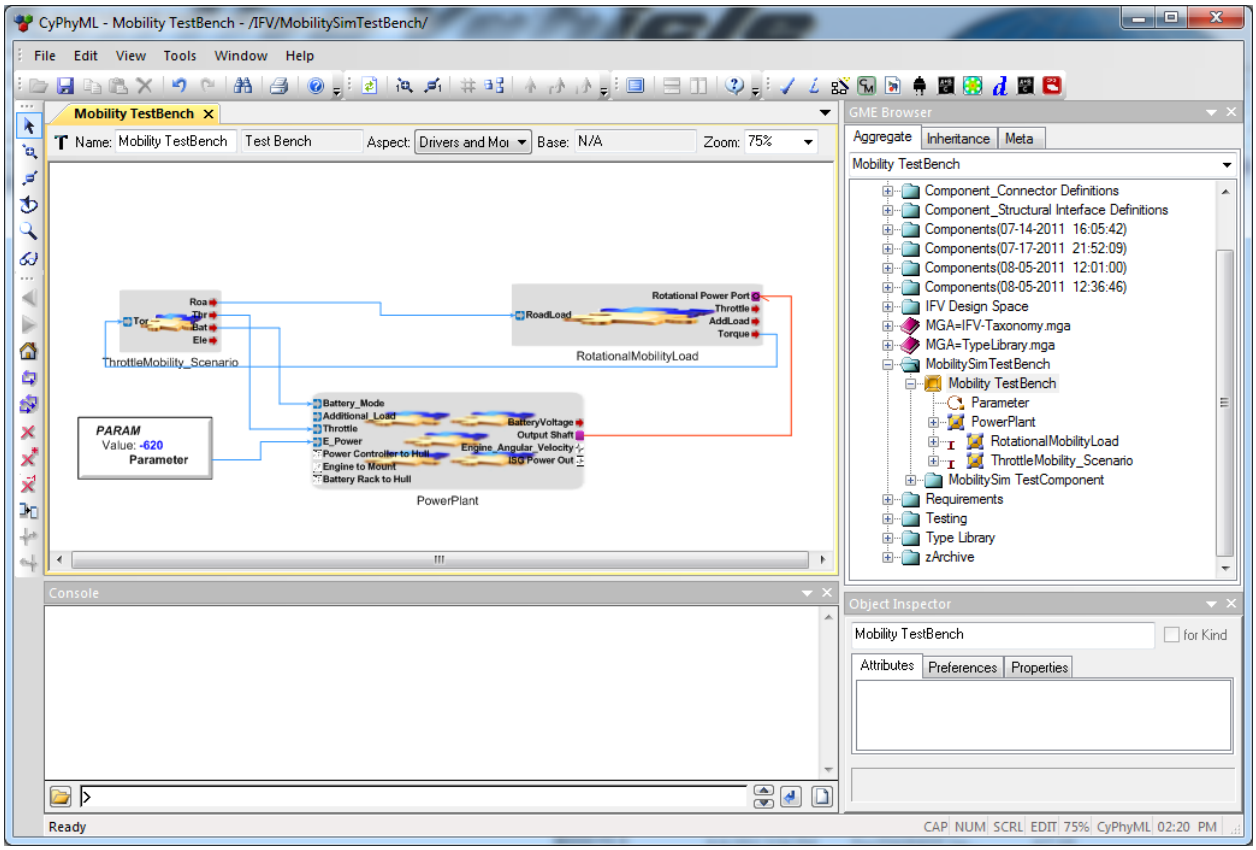
The `Mobility Testbench` allows for evaluation of the powertrain assembly in the context of a mobility scenario exercising the vehicle as a whole. A co-simulation environment is configured from the Testbench that executes a 3D physics-based simulation of the vehicle running alongside the dynamics model of the `PowerPlant_and_Transmission` assembly in MATLAB to give an integrated simulation of (a) the powertrain driving (b) the vehicle inside a virtual 3D world.

Mobility simulations can also be used to gather key operating parameters, via vehicle simulations in the context of specific scenarios (*e.g.* a modeled terrain). For example, a simulation of the IFV can be run on a course constructed for the purpose of measuring the maximum forces on suspension components experienced for that particular terrain. These forces can then be used as input to structural FEA analysis to evaluate the suspension components of the vehicle (to calculate stresses on suspension/IFV hull interface).

Open the Mobility Testbench: "IFV/ Testing/MobilitySimTestBench/ Mobility TestBench/"

Run the Master Interpreter  to generate a fully elaborated Testbench.

Open the elaborated Testbench.




Invoke the CyPhy2Simulink interpreter by clicking the  button. A dialog will appear. No options need to be selected in this dialog.

Press OK.

Open MATLAB.

Right-click Run the `IFV_buildscript.m` found in the following directory:

```
C:\Users\Public\Public Documents\META Documents\IFV Model\generated\ IFV_buildscript.m
```

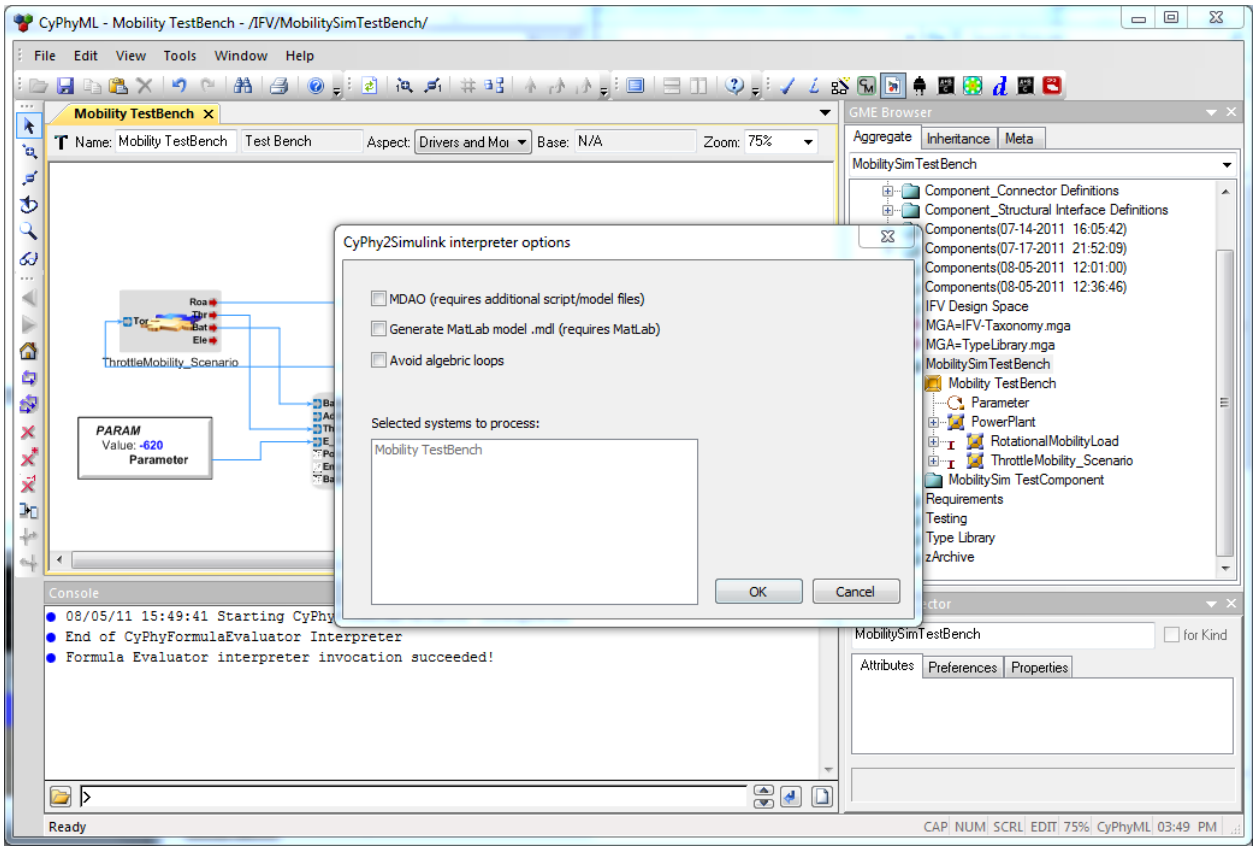
Press the "play" button in the "IFV" MATLAB window to start the simulation. Next, start the `MobilityTurretVehicle.exe` application found in the following directory:

```
C:\Program Files(x86)\META\bin\ParamMobilityDynamics
```

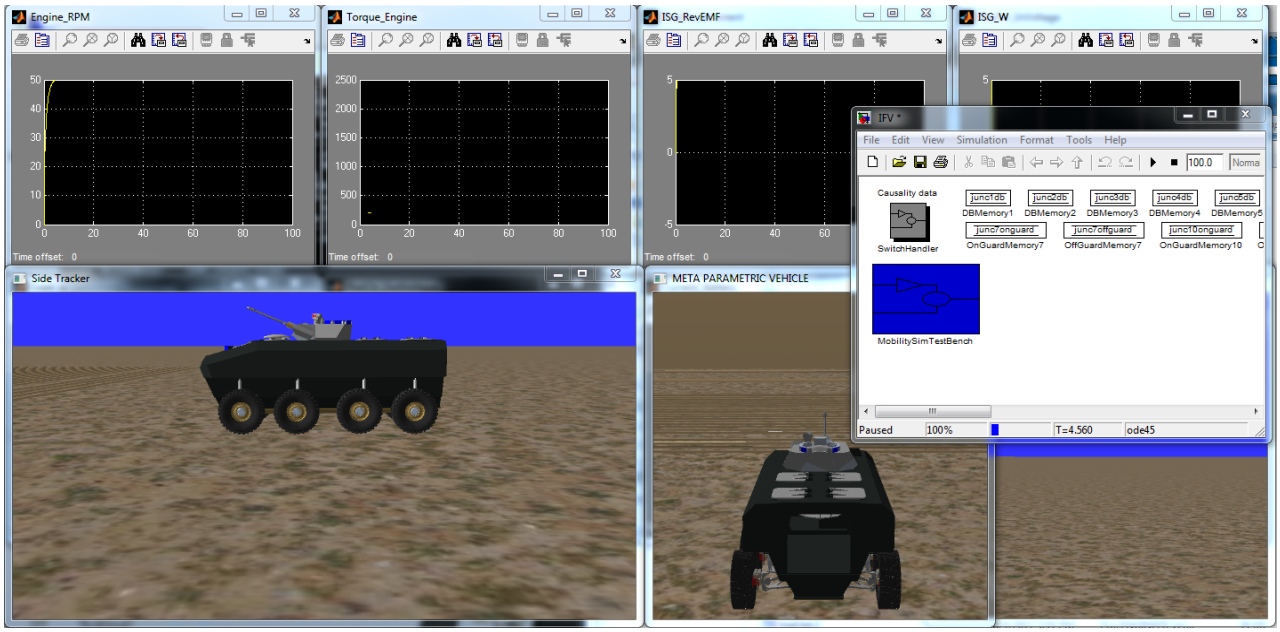
The simulations will run concurrently. MATLAB will simulate the powertrain operating in the context of the scenario, provided real-time by the mobility simulation. A simple scenario consisting of some "half-rounds" of various sizes and a "table-top" hill climb/descend is used to exercise the vehicle. You can observe the increased load placed on the powertrain as the vehicle begins the hill climb. RPM will drop and engine torque will increase on the MATLAB plot outputs.

**Vehicle Control Instructions:**

- The up arrow key raises the engine throttle.
- Down arrow operates the vehicle's brake.
- The left and right arrow keys steer the vehicle[9].
- Reverse is not currently supported. Running into a wall will require you to restart the Testbench.
- The tire friction model is idealized, meaning steering may feel a bit unnatural.
- The Mobility Testbench is timed, meaning that the simulation will eventually time out. Set the simulation time greater than your intended time of operation.




[9] Joystick control is also supported for the Mobility Testbench



# Appendix A: Power User Tips

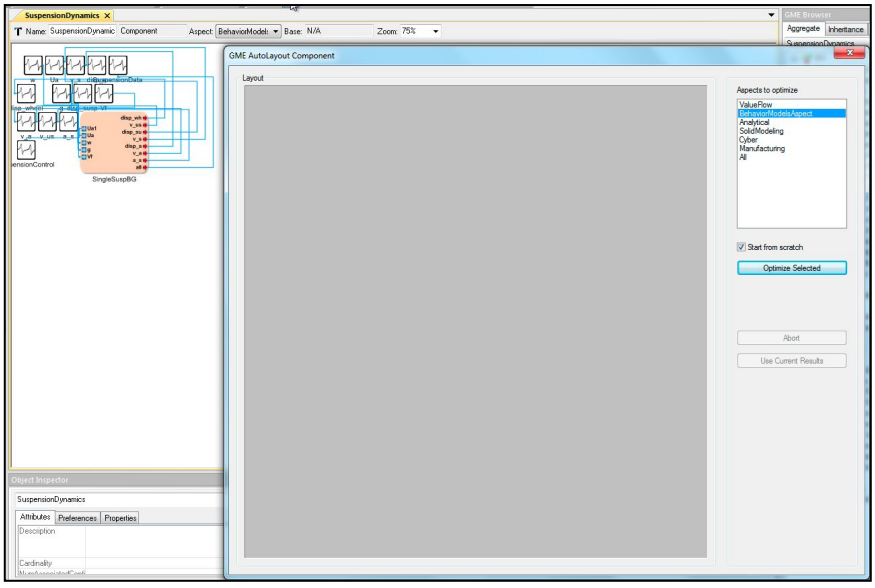
**Resetting the META demo –** You can restore the demo dataset to its default conditions by reloading the following file:

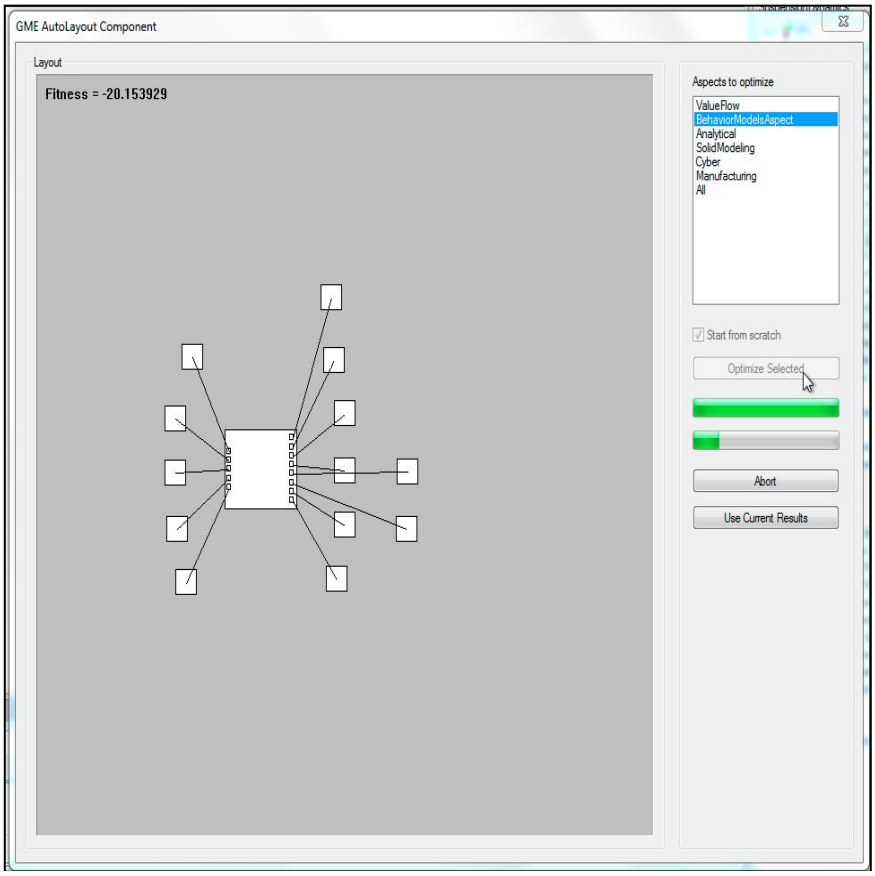
- `Documents/META Documents/IFV Model/IFV.xme`

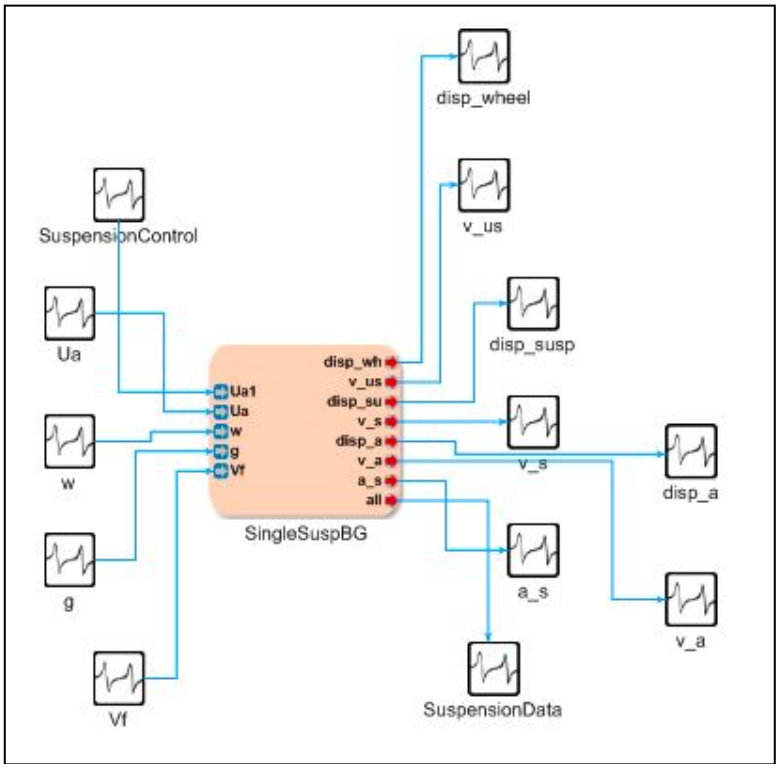
Double-clicking on this `GME XML Model file` will invoke the GME editor, allowing you to import from the selected `XME` archive to generate a new, corresponding `MGA` binary, project file. (Alternatively, you may use the `File` → `Import XML` menu item from within a GME application, **which currently has no project open**.) At the `Import to new project` dialog, select `Next` to create your re-initialized IFV.mga. You may safely ignore any warnings about paradigm mismatches; the tools should be able to resolve any paradigm inconsistencies automatically, within major releases of the software. Press `OK`, if prompted whether to upgrade to the current paradigm. After conversion, GME's console tab should contain an "IFV.xme was successfully imported" log message, and the timestamp on the corresponding MGA file should be recent.

**Auto Layout Plugin –** This is a useful tool for reconfiguring the position of your on-screen elements, efficiently facilitating model readability.

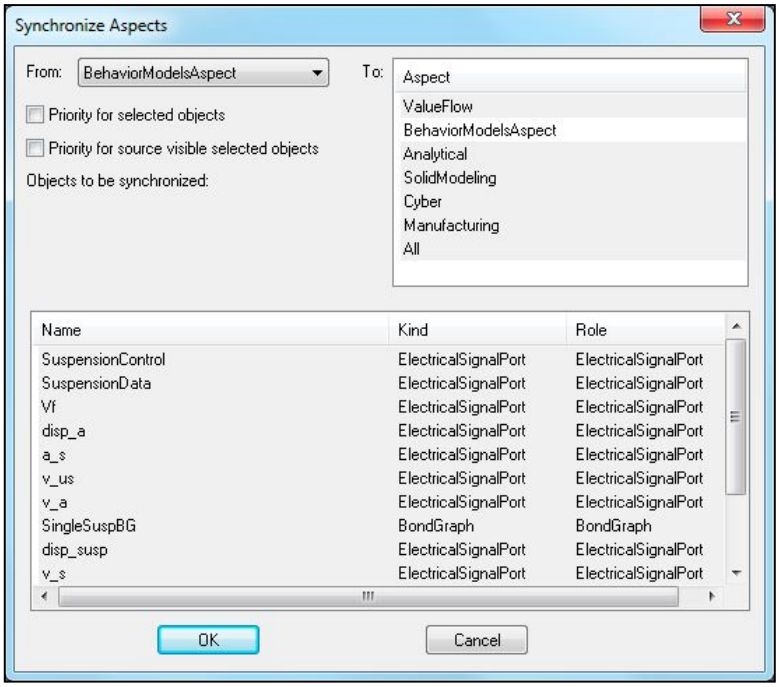
- Clicking the  icon on GME's toolbar will open a dialog that allows you to better optimize the visual layout of a model's various aspects.





**Synchronizing Aspects** – Similarly, the `Synchronize aspects`  toolbar item allows one to reposition the (x,y) coordinates of model elements in one or more aspects to be consistent with that of a source aspect. The default behavior is to synchronize all other aspects with that of the currently selected aspect, but you may override the default `From` and `To` fields before pressing `OK`.

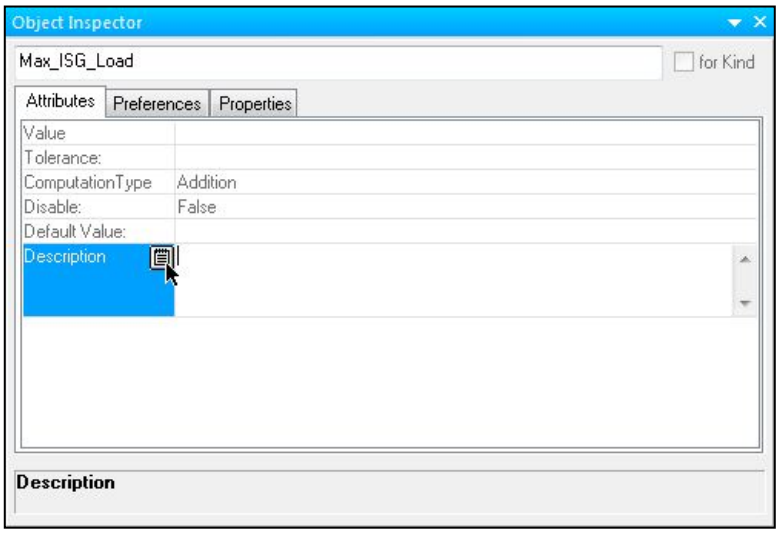



**Enabling an External Editor –** Editing attribute fields using the text editing widgets provided by the Object Inspector GUI is sometimes inconvenient for attributes intended to contain large amounts of multi-line text. Use the following instructions to configure an external editor that may be invoked for multi-line attributes.

- From the `Tools` → `Options` menu, select the `Enable External Text Editor` check box (for Multiline Attributes). Next, provide the path to the desired external text editor application (*e.g.* `C:\Windows\notepad.exe`).
- The `Object Inspector` pane will now contain a new icon near the attribute name for all multi-line attributes. Selecting this icon will invoke an external editor that will be initialized with that attribute's

current value. GME will monitor the lifecycle of this external process, and overwrite the attribute's value with the contents of the editor, upon detecting that the editor application has exited.




**Experimental Table Editor** – This tool allows users to examine model elements in a spreadsheet view. The table editor is currently under development, but available for experimentation.

**Table Editor Operation Instructions:**

- Once the IFV model is loaded, open the tools drop-down menu.
- From the tools menu select *Register Components…*
- At the bottom of the list you will find the Table Editor.
- Click the *both* radial button and enable the Table Editor.
- Click the  icon on the top toolbar to launch the table editor.
- Select the elements of the model that you wish to examine.

## Appendix B: The Architecture of the META Tools

What follows is a brief introduction to the philosophy of GME[10], with the intention of providing some surface level insight into the META tools' underlying architecture. At its core, GME is a highly configurable, visual language construction toolkit, used to engineer novel, domain-specific, graphical modeling and program synthesis environments, which are targeted toward specific problem domains. The domains GME has been successfully employed in span a wide space, including fields such as (a) building new visual languages, and (b) automotive engineering. Once instantiated for a particular problem domain, GME provides a graphical editor application for drawing and browsing models, but more importantly it also provides relevant model transformations for converting graphical models into useful artifacts. Examples of useful artifacts for the automotive engineering domain are (a) executable simulations of vehicle performance under certain conditions, and (b) a bill of materials used to support manufacture/assembly of the physical vehicle. An example of a useful artifact for the domain of language engineering would be a graphical model editor customized for the automotive domain.

The configuration and customization of GME itself in order to support a new, well-defined application domain is accomplished through *meta-models* specifying the modeling *paradigm* (*i.e.* the modeling languages to be provided to end users) that will encode the target application domain. A modeling paradigm defines all of the syntactic, semantic, and presentation information for the domain of interest. For example, which concepts will be used to construct models, what relationships may exist among these concepts, how the concepts may be organized and viewed by the modeler, and rules governing the construction of valid models. Any modeling paradigm formally defines the family of all models that can be created using the resultant, domain-specific tools. Once defined, modeling paradigms are then used to initialize and instantiate a customized version of the GME tools oriented toward the target application domain. In the case of META, we call this modeling paradigm *CyPhy* (short for the *Cyber Physical* modeling language, and pronounced as if it were an abbreviation for *Science Fiction*).

The details of the CyPhy modeling paradigm are of little interest to the average META end user[11]. By installing META after installing GME, you have therefore automatically enabled GME to support the AVM community. The specifics of how this works may be safely ignored; only be aware that there is a subtle distinction between what is referred to as the GME application—a family of generic, reprogrammable tools supporting visual language construction, model editing, and model transformations—and the META toolsuite, which is a specific instantiation of the GME tools provided to the AVM community.

[10] http://www.isis.vanderbilt.edu/Projects/gme/

[11] For the interested reader, GME is packaged with an extensive set of documentation and tutorials. See the 'Documentation' section of your MS-Windows 'GME' start menu.

## Appendix Z: Frequently Asked Questions

Q. I downloaded and installed GME, but I can't open the IFV model.

A. GME is only one of the pre-requisites for the META tools. Ultimately, the application necessary to use this IFV model is the 'META' toolsuite, not GME. If you have not run the META installer, you do not have a copy of the META tools. See Appendix B for an explanation.

Q. I have opened an MGA file as specified above, but there is no toolbar exposed within my GME application window.

A. See footnote 2 in the introductory section of this document.