

MARCH 2012 DEEP DIVE DEMO WALKTHROUGH	2
OVERVIEW	2
THE PARAMETRIC EXPLORATION TOOL (PET)	2
PREREQUISITES.....	3
FILES AND DIRECTORIES	3
AN EXAMPLE: MULTIBODYENGINE SIM	5
SIMULATING THE PET DEMO	5
HOW TO CREATE/CURATE A PET MODEL FROM SCRATCH	11
CREATING GME TEST BENCHES	23
PET TEST BENCHES	27

March 2012 Deep Dive Demo Walkthrough

Overview

This document serves as a walkthrough demonstration of the model curation process as outlined in the DARPA Adaptive Vehicle Make C2M2L project. The models presented here were developed and demonstrated during the March 2012 Deep Dive meeting at Vanderbilt University, and are used to drive a cohesive narrative built around a specific, concrete example. The goal of this document is to provide first-time users of the META Toolsuite with systematic instructions on how to transform pre-existing model artifacts (e.g. Modelica models), which exist in multiple domains and serve various purposes, into a form that is compatible with the philosophy and architecture of the AVM program. As the example use case is oriented toward using META's PET features, this tutorial also serves as an introduction to (a) simulating dynamics in Modelica, and (b) using the advanced *parametric exploration* features of the tools.

The Parametric Exploration Tool (PET)

PET allows a user to perform a Design of Experiments (DOE) in order to gain an understanding of the effects that varying inputs have on the output. In the case of this presentation, the DOE will systematically vary the dimensions of the engine's combustion cylinders, the number of cylinders, final drive ratio, and external load (applied to wheels of the vehicle), while measuring the simulated maximum speed of the vehicle. The goal of a such a DOE is to provide the user with a tool that allows them to identify the control parameters that are the most influential to changes in the output metrics; and to do so in as efficient a manner as possible. A flow-chart that will be the end product of this tutorial is provided in Figure 1 along with a short description of each component.

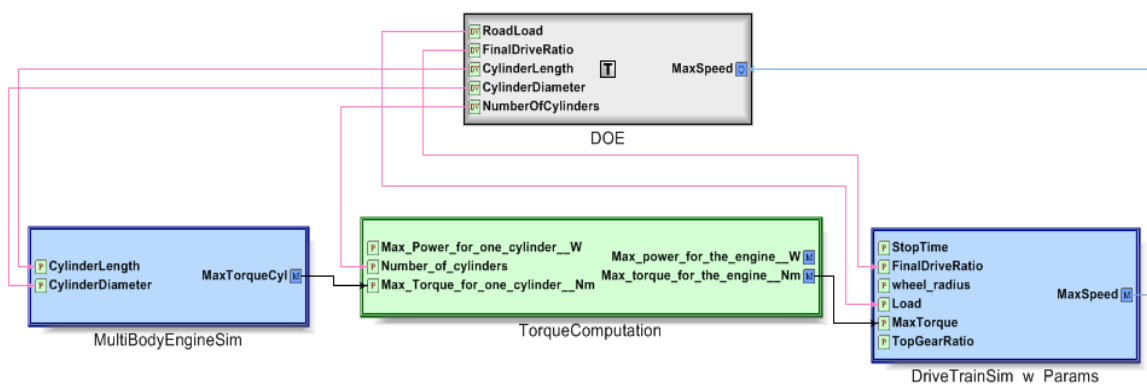


Figure 1. PET Flow Chart - March 2012 Deep Dive Demo

- **DOE:** This component contains all the information required for the DOE. It defines the type of DOE that is under consideration (e.g. Full Factorial, Latin Hypercube, *etc.*), number of levels, and the appropriate ranges for each variable.
- **MultiBodyEngineSim:** This component represents a curated Modelica model (multibodyenginesim.mo) that is responsible for estimating the maximum torque provided by a single combustion cylinder based on its geometry. The model itself is a simple equation with two inputs and a single output.
- **TorqueComputation:** This component represents a curated Excel component (TorqueComputationForNCyl.xlsx) that is responsible for estimating the maximum torque for an engine designed with N cylinders. The model receives number of cylinders, N , and maximum single-cylinder torque, as inputs.
- **DriveTrainSim_w_Params:** This component represents a curated system of Modelica dynamics models that define an entire powertrain. The model receives the final drive ratio and load (applied to the wheels of the vehicles) as inputs from the DOE, as well as the maximum engine torque calculated by the Excel component. This component then uses a system of dynamics models to estimate the maximum speed of the vehicle under consideration.

Prerequisites

- GME.msi (version 11.9.20 or newer)
- OpenModelica 1.8.0
- Python 2.7
- Numpy, Scipy, Matplotlib
- Microsoft Excel
- Win32com (Python for Windows extensions)
- OpenMDAO-0.2.2 (installed at c:\openmdao-0.2.2\)
- META.msi (version 2012 March or later)

Files and Directories

The META installer automatically creates a directory in C:\Users\Public\Public Documents\META Documents, called `Modelica and PET March 2012`. Within this folder are a series of supporting documentation and project files. Among them is a GME project file called `2012MarchDeepDive.xme`. The main focus of this document is to show the steps necessary to create this project. Other files found within this directory include:

- **IFV_Lib:** Structured library of Modelica dynamics models representing simplified engines, transmissions, drivers, etc.
- **Include:** The CyPhyDynamics interpreter requires this directory. If you put any Matlab models in this folder, they will be copied over to the `./generated/` directory.

- **MultiBodyEngineSim.mo**: Modelica model that estimates the torque of a single cylinder engine based on cylinder dimensions. The model exists as a simple equation with two inputs (cylinder diameter, cylinder length) and a single output (torque).
- **TorqueComputationForNCyl.xlsx**: Excel spreadsheet that estimates the maximum torque for an engine with N cylinders, given as input the number of cylinders and maximum torque for a single cylinder (uses simple regression).

An Example: MultiBodyEngineSim

This walkthrough will assume that the Modelica models have been created and delivered by an external entity and that the format of the component library has already been established. In addition, it is assumed that the inputs and outputs of each model have been provided by the entity that developed the model.

Simulating the PET Demo

Prior to running the PET Demo, the user must first create the appropriate supporting files for OpenMDAO. To do this, open the 2012MarchDeepDive.xme model and double-click on the DriveTrainSim_w_Params test bench and invoke the CyPhy2ModelicaInterpreter icon from the GME toolbar (see Figure 2).

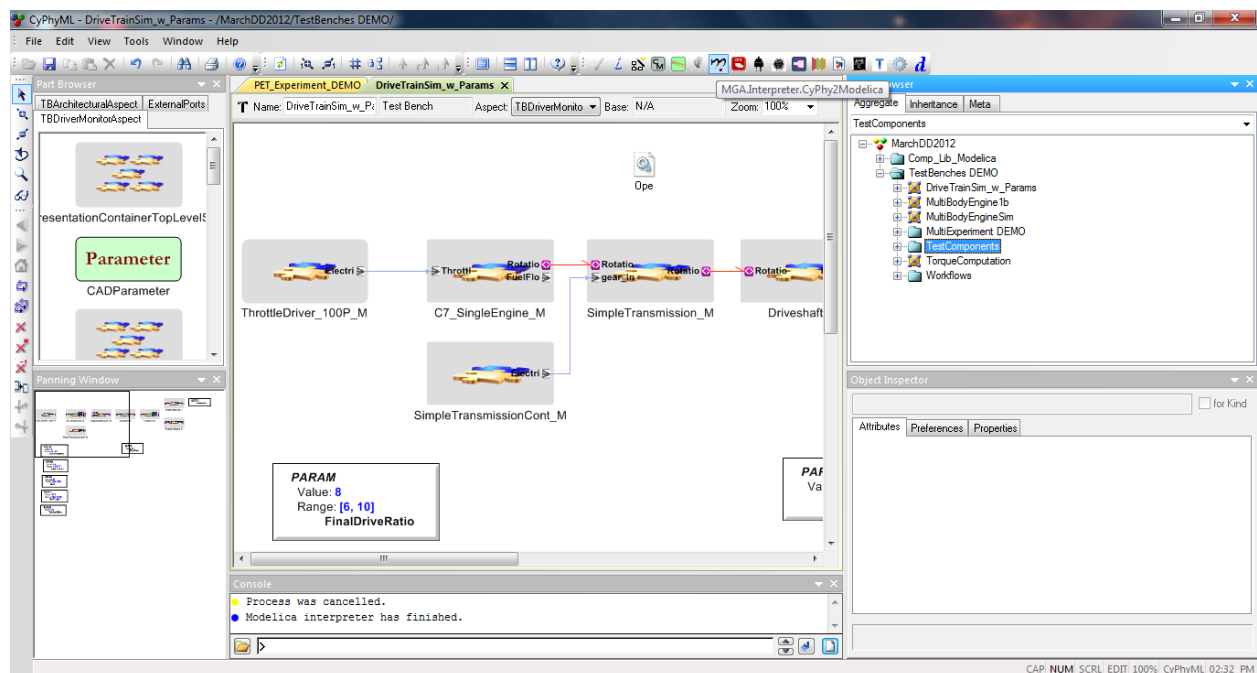


Figure 2. CyPhy2Modelica Interpreter

A pop-up window will appear asking the user to identify settings for the interpreter (see Figure 3).

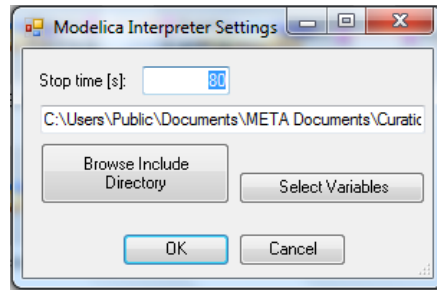


Figure 3. CyPhy2Modelica Interpreter Settings

Because many of the components associated with the PET Demo are located within a component library, the user must include this library in the interpreter settings. This is done by clicking on the `Browse for Library Directory` button in the `Modelica Interpreter Settings` dialog. This will prompt the user to identify where the library is stored (see Figure 4). All component models for this tutorial are stored in `IFV_Lib`.

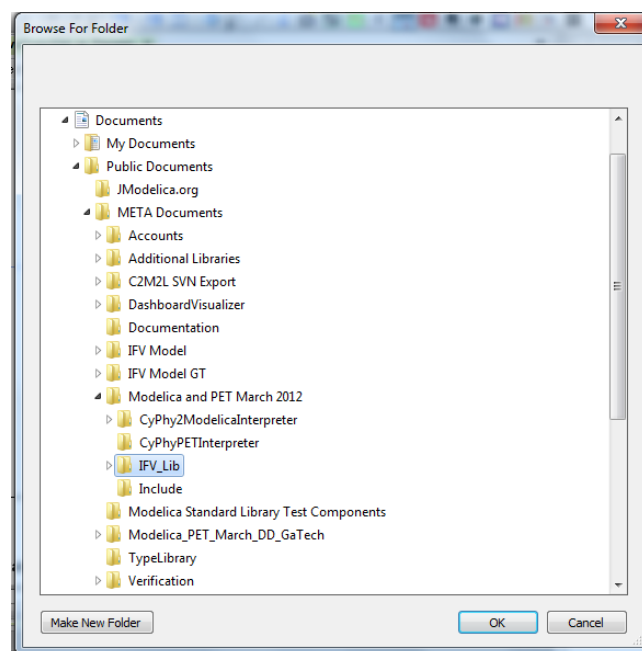


Figure 4. Include Component Library

Once the library folder has been located, click the `OK` button. This will bring the user back to the Interpreter Settings window where the user will have to push an `OK` button again to run the interpreter. The user should be prompted with the pop-up window shown in Figure 5.

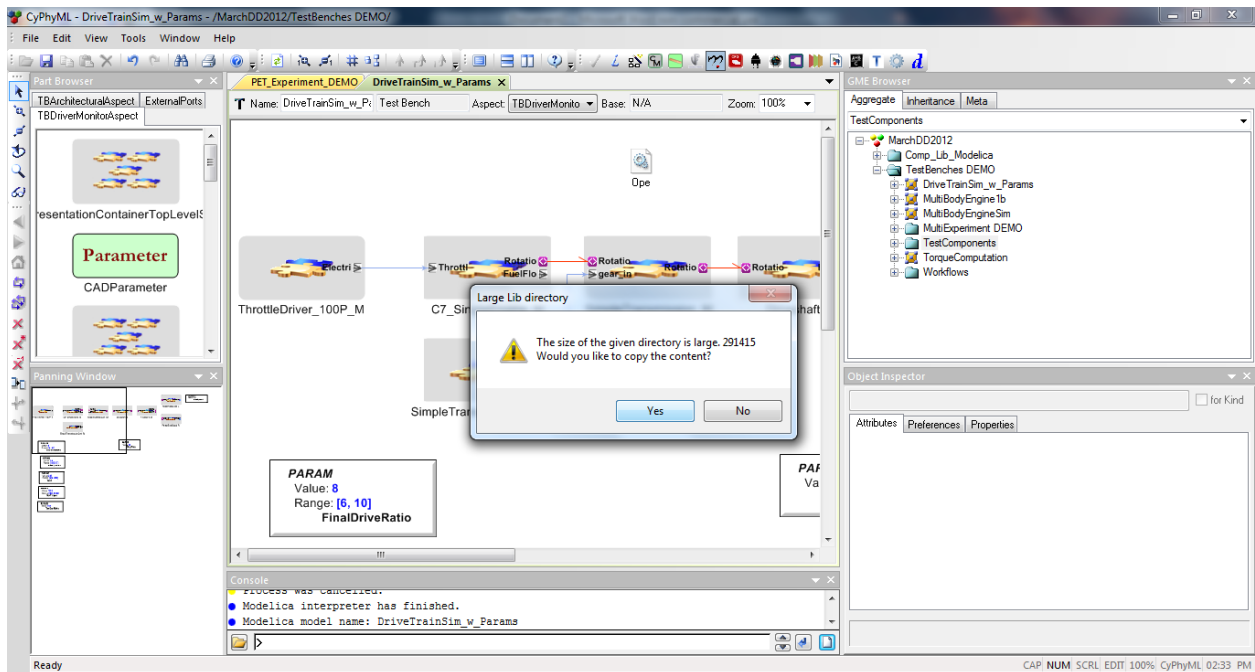


Figure 5. Copy Directory Content

Click **Yes** to copy the content. This will open up the command prompt where the user should see a few lines generated (see Figure 6). The command prompt should only be open for a few seconds.

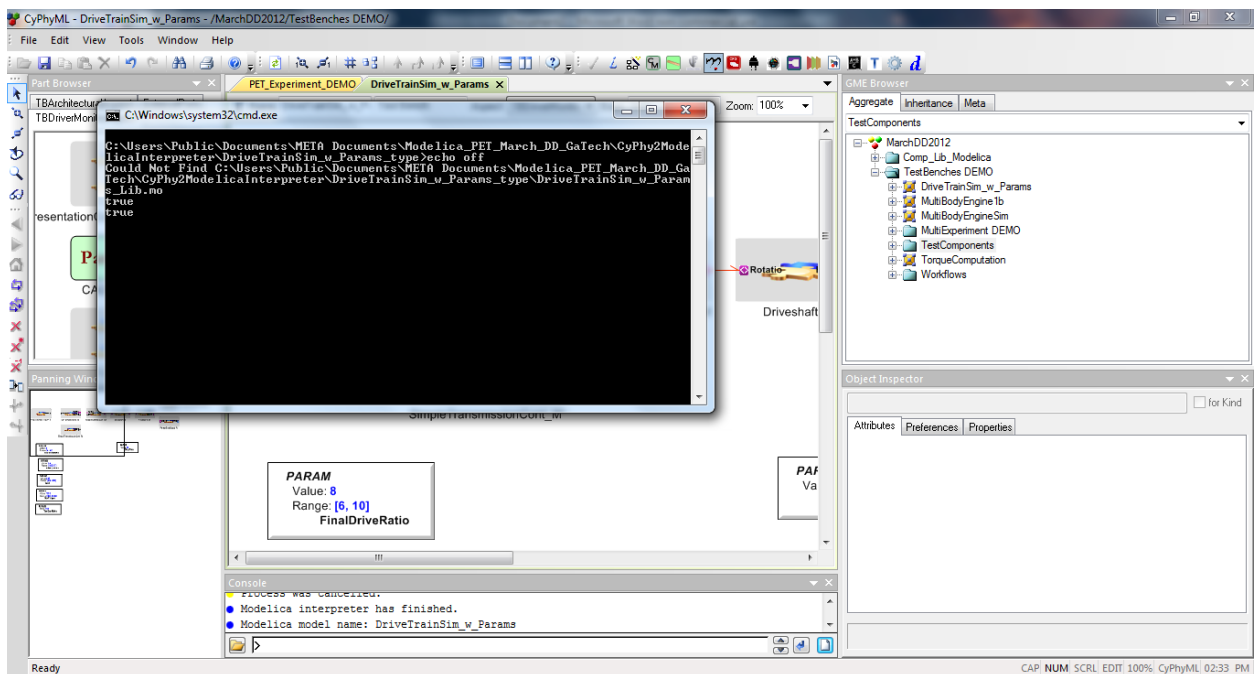


Figure 6. Interpreter Opening Command Prompt

Once the interpreter has finished, a series of messages will appear in the Console window located at the bottom of the GME application. One of the messages will have a link titled “here”. The user will need to right-click on the link and select the “Open in new window” option (see Figure 7).

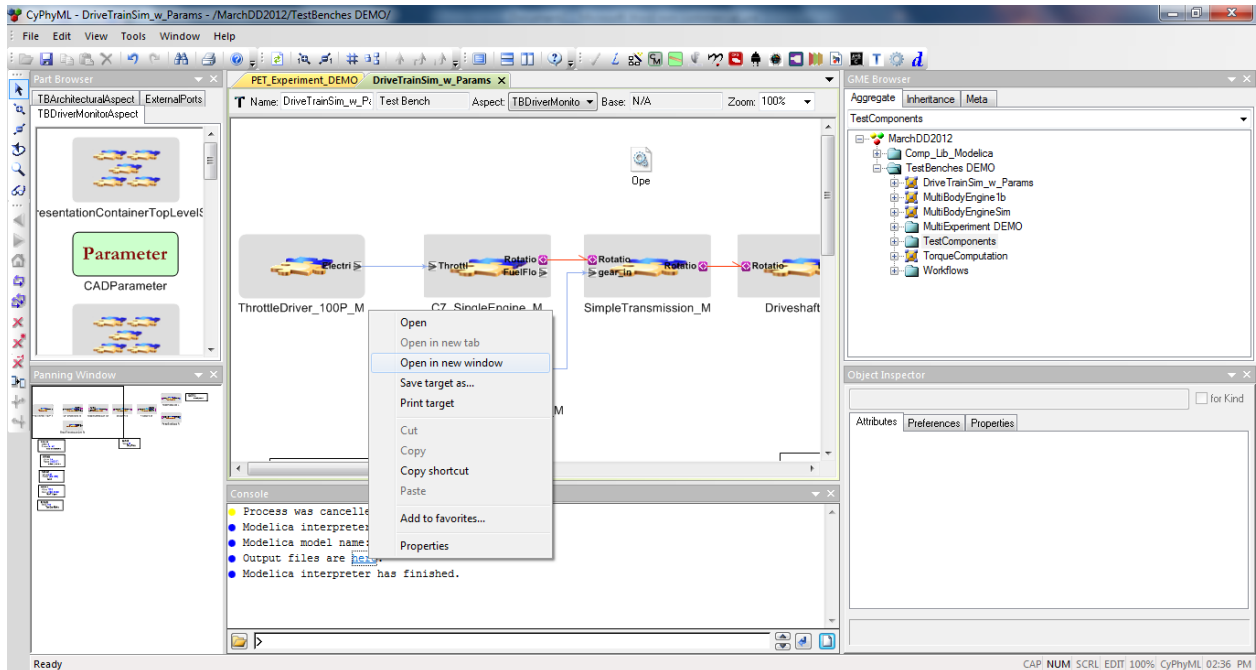


Figure 7. Output Files Link

This will open up the window shown in Figure 8.

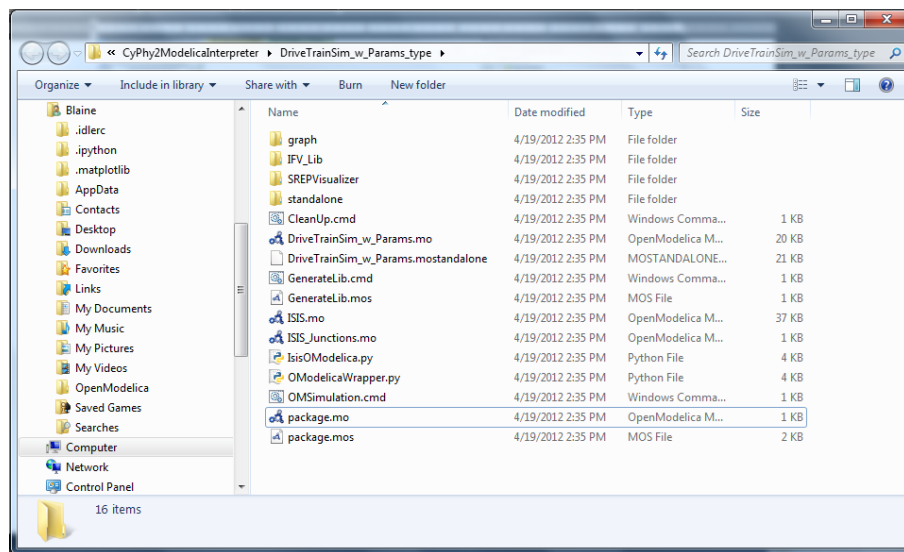


Figure 8. Interpreter Generated Files

Next, open the `package.mo` file in OpenModelica, expand the top-level container, & open and simulate the `DriveTrainSim_w_Params` model. Upon completion of the simulation, the user will be able to plot the maximum speed of the vehicle (variable: `MaxSpeed`). Figure 9 provides a screen shot of what the output from the user's simulation in OpenModelica should look like.

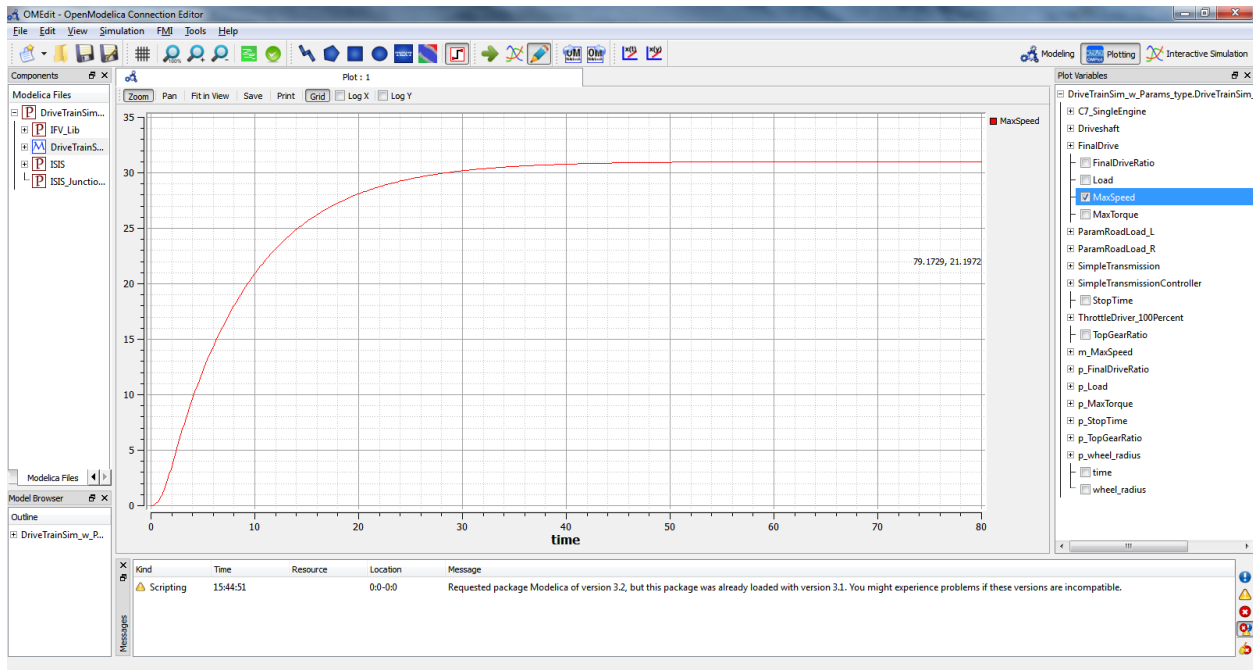



Figure 9. OpenModelica DriveTrainSim_w_Params Simulation Output: Max Speed Plotted

Close OpenModelica and return to the GME Main Editing Window. Double-click on the `MultiExperiment DEMO/PET_Experiment_Demo` model to open it. Once the screen is active, the user needs to run the CyPhyPET interpreter by clicking on the  icon along the top of the GME window. The interpreter generates a `CyPhyPETInterpreter` folder in the same directory as the GME project. This folder contains several files including seven (7) python scripts, one (1) html file, and one (1) cmd file. Double-click on the `PET_Experiment_DEMO.cmd` file to run the PET Demo. The simulation will compile the Modelica models into executables, run the DOE experiment, and save the results into files. Once the simulation ends, an `mdao_index.html` file will be generated. The user will need to open this file in a web browser (tested primarily with Firefox) to view the results of the DOE (see Figure 10).

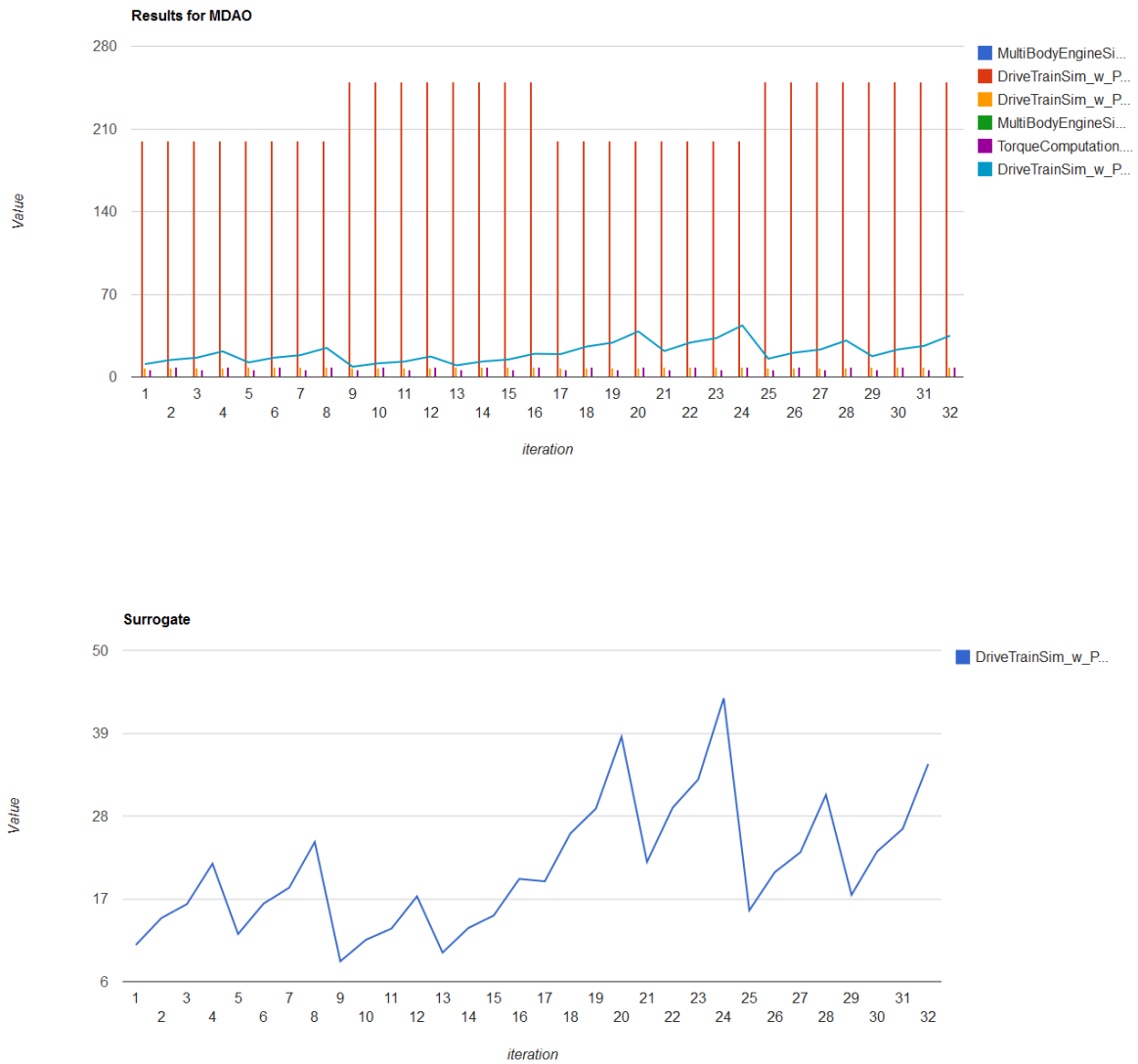


Figure 10. Results of DOE

The first plot provides the values of the DOE design variables (represented as vertical bars) and the output metric (represented as a blue line). The second plot represents a surrogate of the simulation. This concludes the PET Demo tutorial. Next, we will explain how to create all necessary elements in the GME modeling environment to simulate a simple system-level simulation and DOE.

How to Create/Curate A PET Model from Scratch

Navigate to the GME folder from the Start Menu and click on the GME icon. Upon opening the GME program, the user should be confronted with the screen shown in Figure 11.

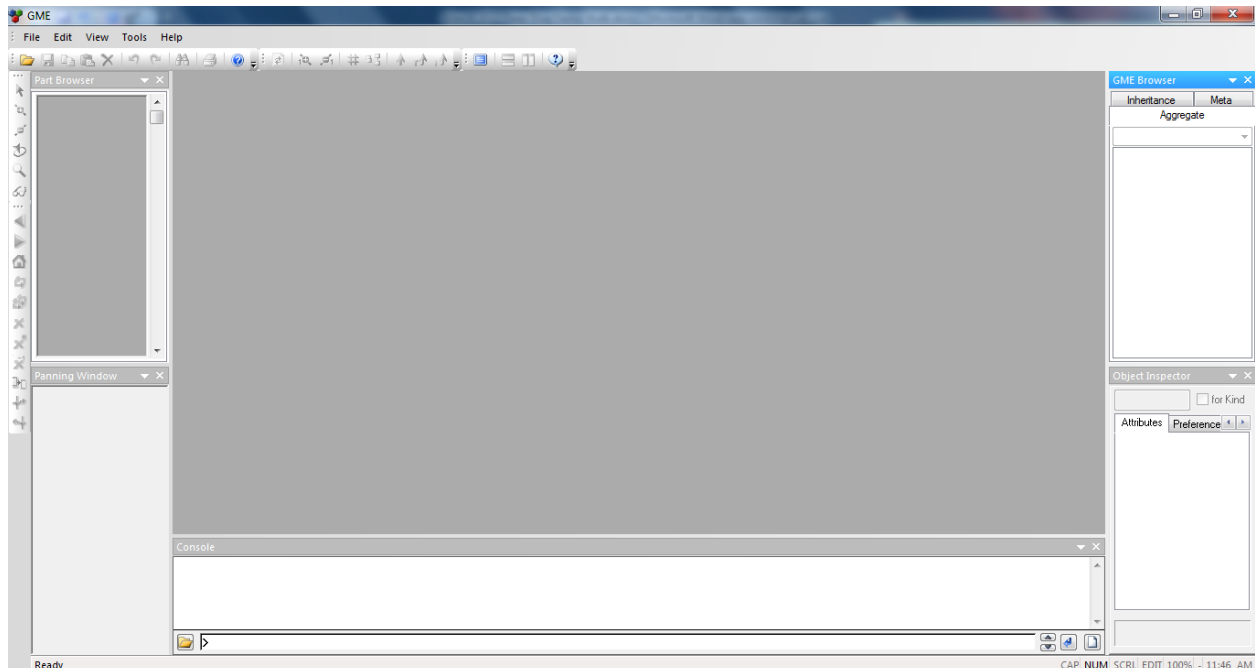


Figure 11. GME Main Editing Window

This interface is the GME Main Editing Window and will serve as the primary user interface for all model transformation. While excluded here, a detailed description of the Main Editing Window is provided in GME's help directory. To gain access to the help directory click on the help tab on the top of the screen and select contents. This will open the `GME Manual and User Guide` window. From this window, expand the main folder titled "GME Manual and User Guide". Inside "The Generic Modeling Environment" folder is a detailed description of the GME Main Editing Window.

New projects can be created by navigating to the "New Project" option in the `File` dropdown menu or through the shortcut key (Ctrl + N). This will open a screen that will prompt the user to select an appropriate paradigm for the project (see Figure 12).

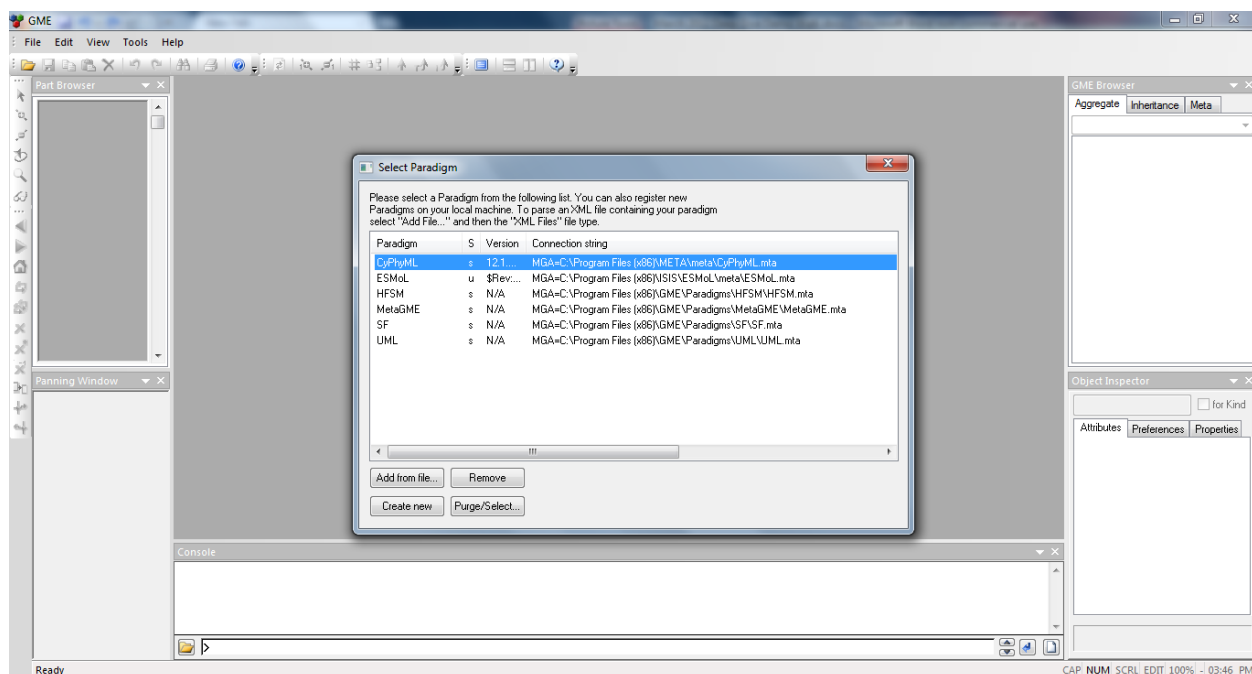


Figure 12. Paradigm Selection Window

For the purposes of META, we will be using the CyPhyML paradigm. Double-click on the CyPhyML paradigm.

On the following screen, make sure that the “Create project file” radio button is selected and press Next.

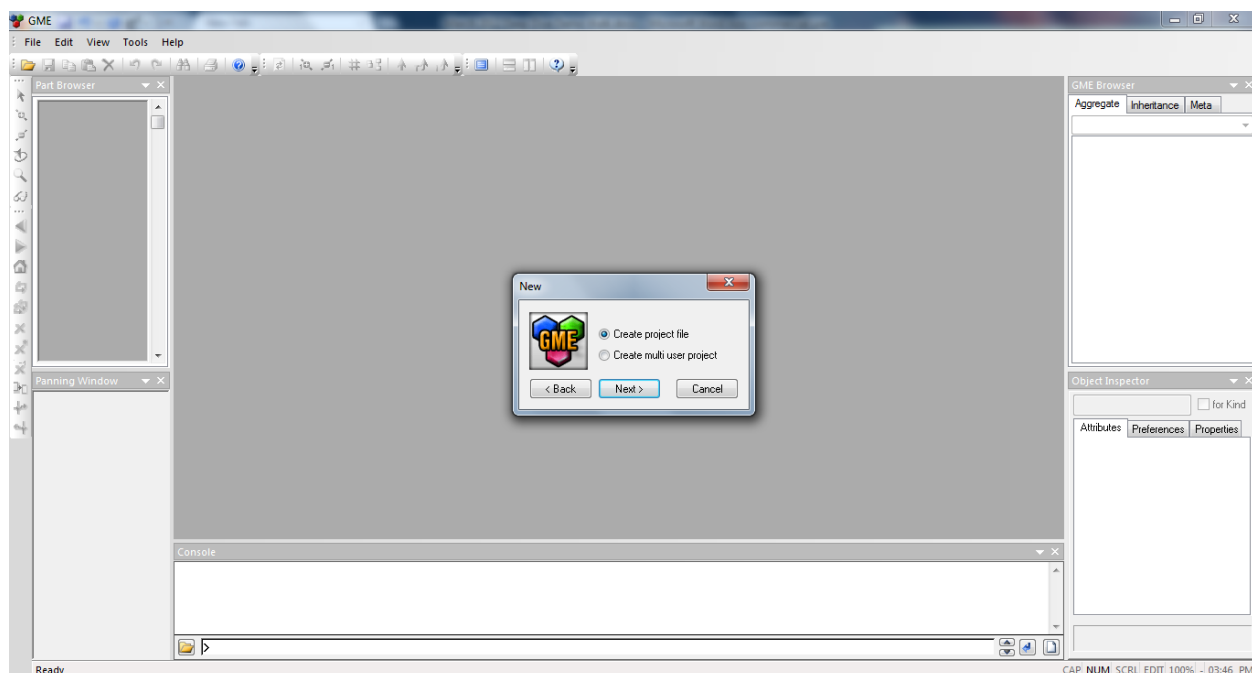


Figure 13. Create Project File Window

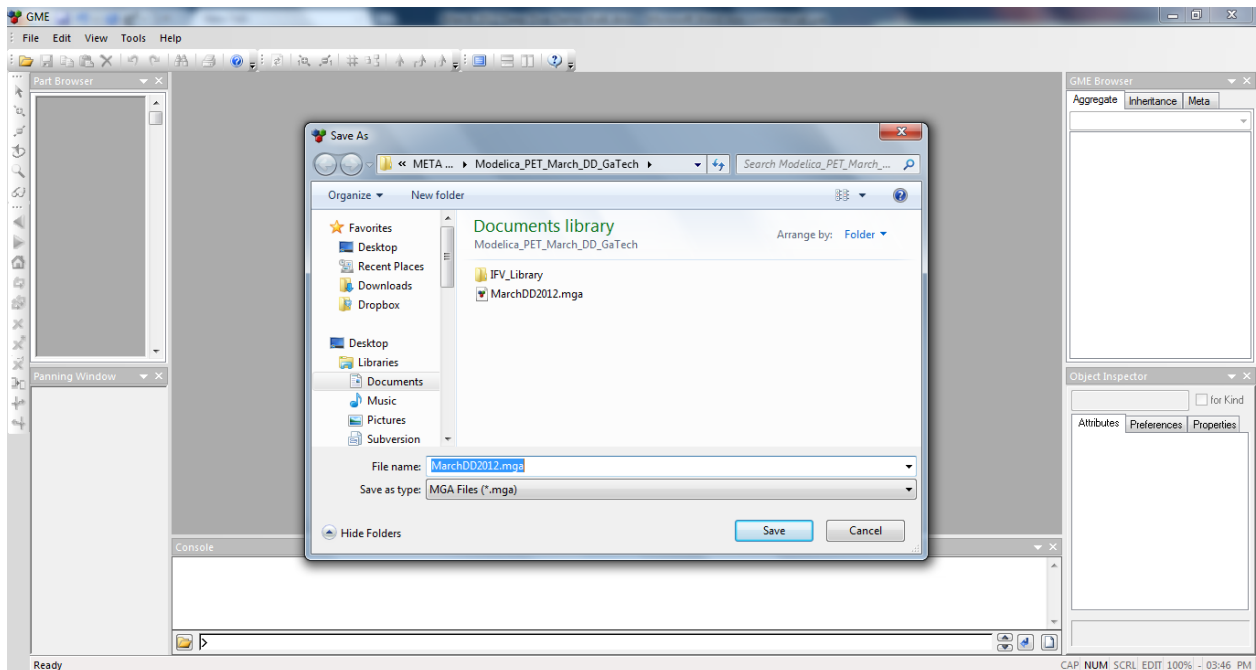


Figure 14. Naming Project File

The user will be prompted to save a project file (.mga). It is important to name this file something meaningful and to place it in a convenient directory. This directory will also need to contain the files discussed in the *files and directories* section in the intro section of this document.

After saving the project file, the user will be returned to the GME Main Edit Window with a new project created (see Figure 15). The GME Browser will contain the `Root Folder` of the project, which can be named to something more meaningful by user.

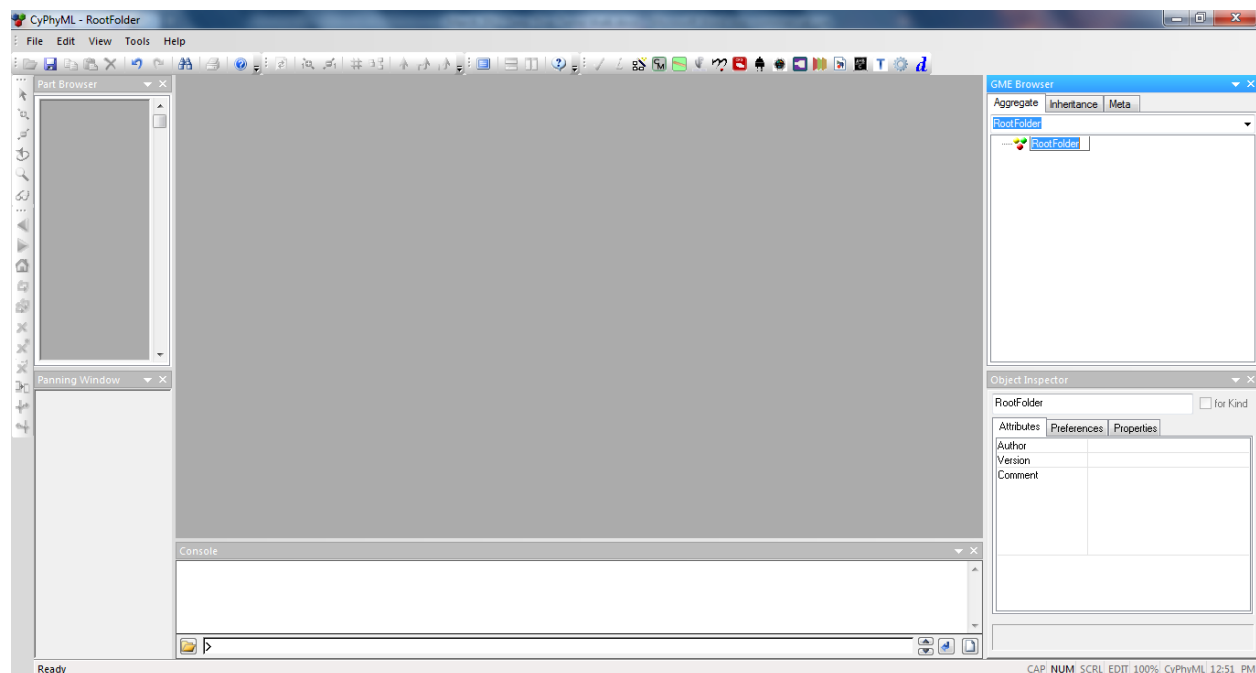


Figure 15. New GME Project

The Root Folder is the launch point from which the GME component library will be created. The library structure for the March Deep Dive Demo will consist of two main folder types: components and testing. The components folder will be responsible for organizing and storing all of the curated Modelica components. The testing folder will be used for organizing and storing all of the test benches and supporting files which are required to perform simulation.

To create a folder, right-click on the Root Folder and select the Insert Folder option. Multiple folder options will appear for the user to select. As mentioned above, we are interested in creating a folders for components and for testing (create these by selecting the appropriate option in the pop-up window).

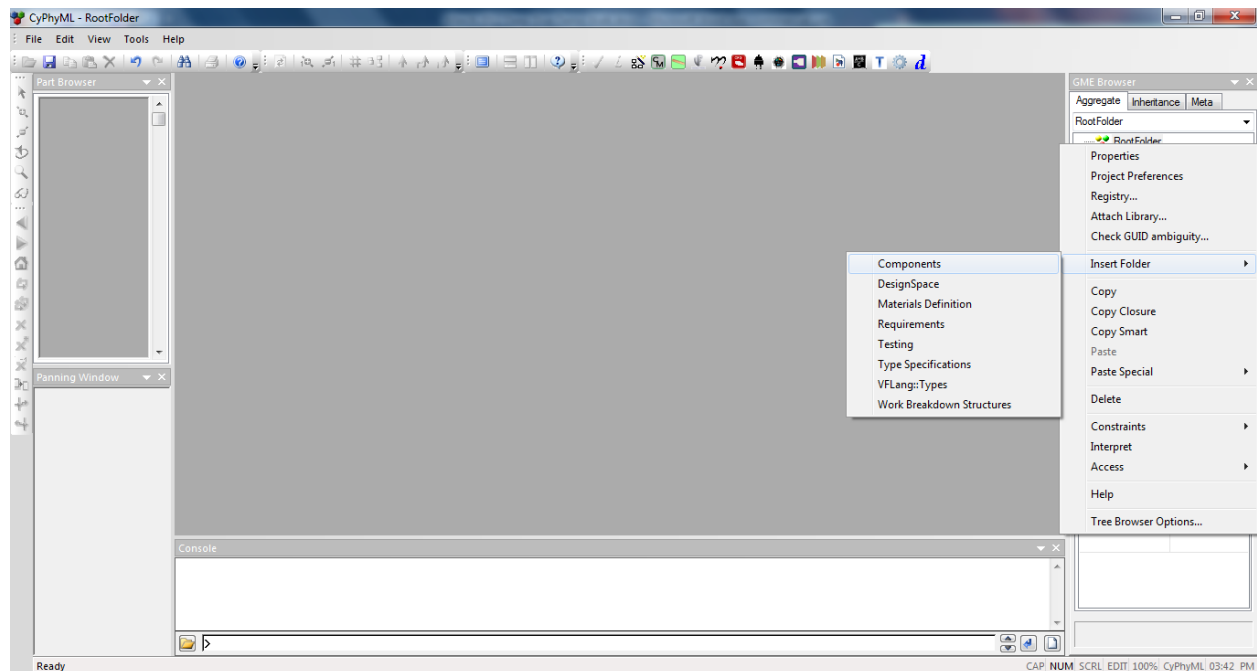


Figure 16. Insert New Components Folder

Once this step has been completed, the user may rename each of the folders by slowly left-clicking twice (but, not quickly *double-clicking*) causing an editable textfield widget to be enabled, or by typing in a new name within the Object Inspector. The user should be viewing a screen that looks similar to that in Figure 17. In this example, the component and testing folder have been named `Comp_Lib_Modelica` and `TestBenches` respectively.

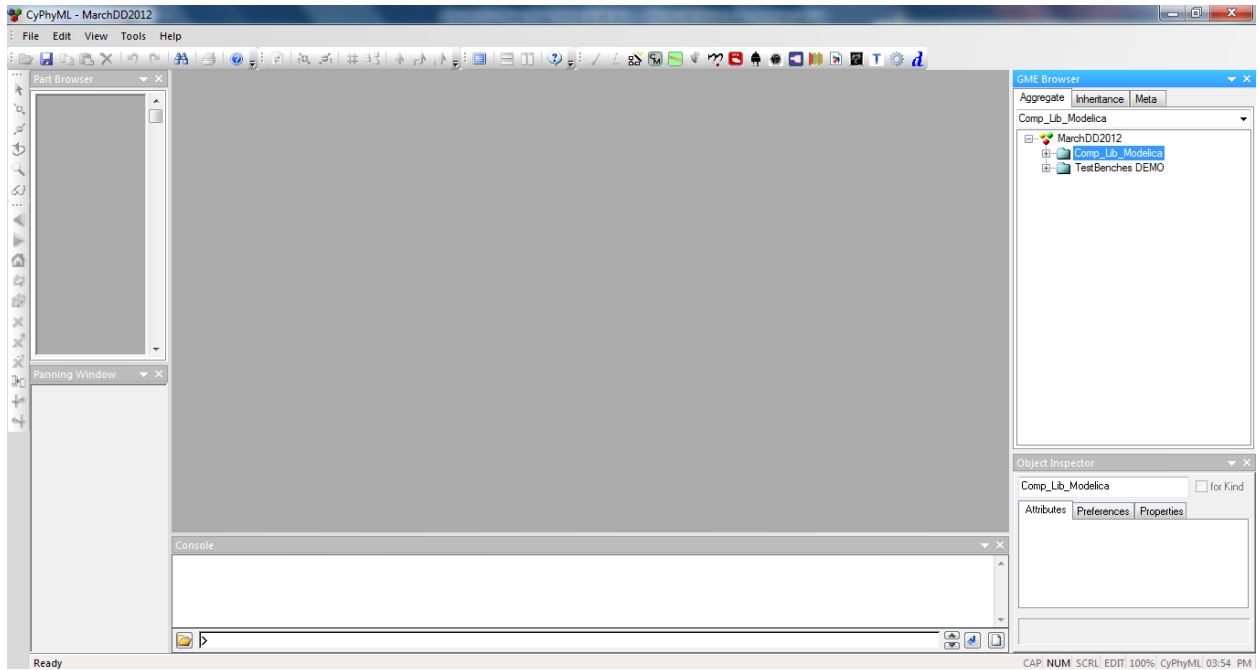


Figure 17. Renaming Folders

Now that the components and testing folders have been created, the user can begin creating the various components that will be utilized in this example. These models are listed in Table 1.

Table 1. Component Models List

Model	Software
C7 Engine	Modelica
Driveshaft	Modelica
Final Drive (Ratio)	Modelica
Transmission	Modelica
Transmission Controller	Modelica

The following walkthrough will demonstrate the creation of each component as it is presented in Table 1. However, it is important to note that the order holds no special purpose; a user may create the following components in any order of his or her choosing.

To create a component, right click on the components folder that was just created and scroll down to the Insert Model option. By highlighting this option, a pop-up window of potential model types appears as shown in Figure 18.

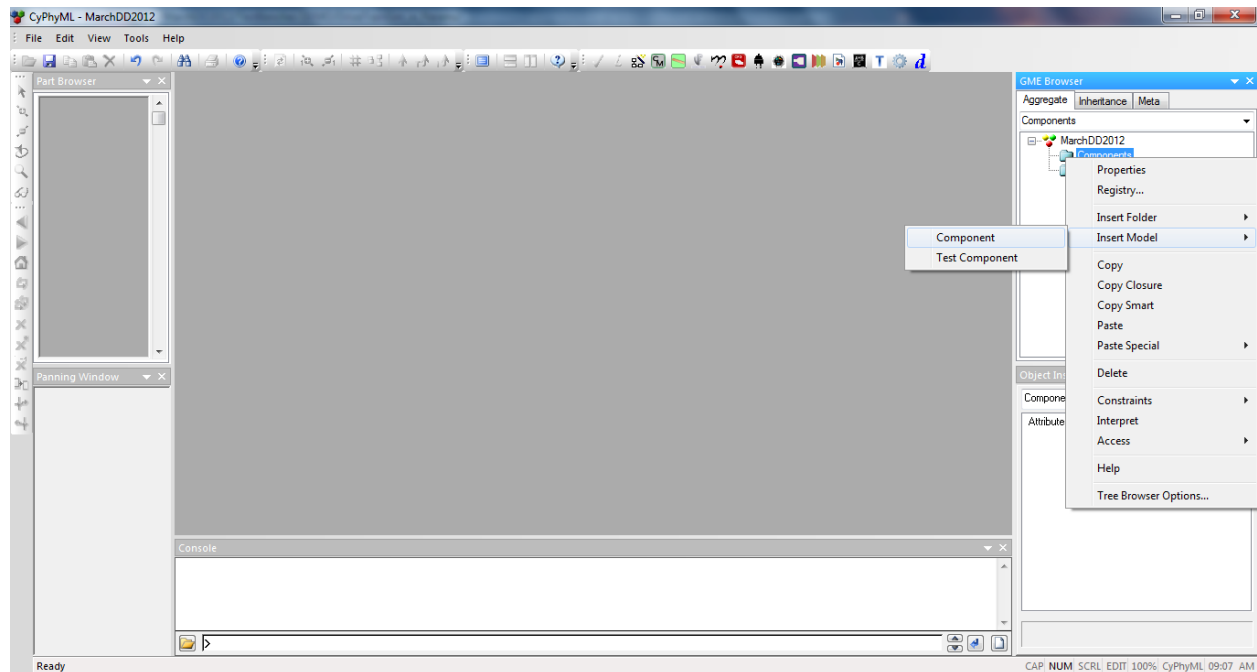


Figure 18. Insert New Component

The components created in this section will all utilize the Component type. Upon creation, the model should be renamed to something meaningful – the first model created will represent the C7 Diesel Engine. Avoid using spaces in the component name. Double-click on the component to reveal the screen shown below in Figure 19.

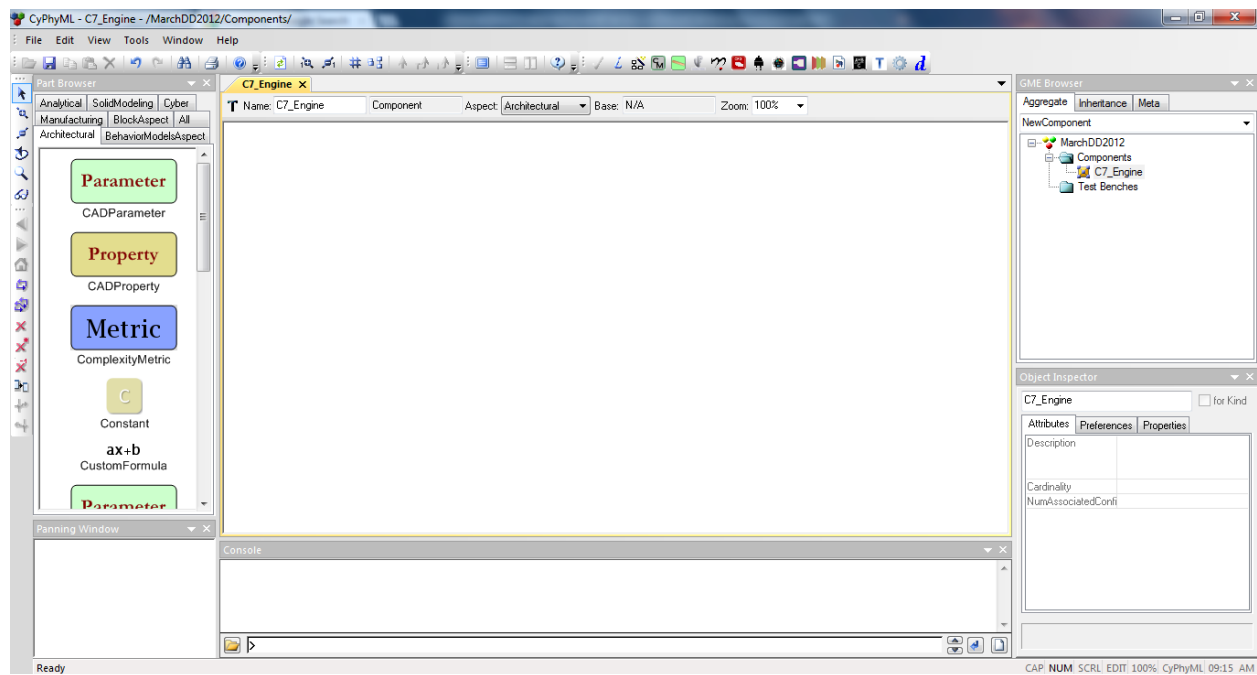


Figure 19. Component Editing Window

While a detailed description of the GME Main Edit Window is not presented here (it can be found in the GME User Manual), it may be helpful to provide the user with a brief description of the mechanisms at the user's disposal. The Part Browser (leftmost screen) contains an assortment of GME modeling elements that can be used for the selected aspect. The purpose of different aspects is to provide the user with control over what model components are visible at a given time. In large complex systems, the amount of information viewable by the user can quickly become overwhelming – using different aspects can greatly simplify the system such that the user can focus on the elements of the problem that are of interest. The user can control the aspect by selecting one of the tabs directly above the GME model elements or by activating the drop-down aspect menu located above the main editing window (Figure 20).

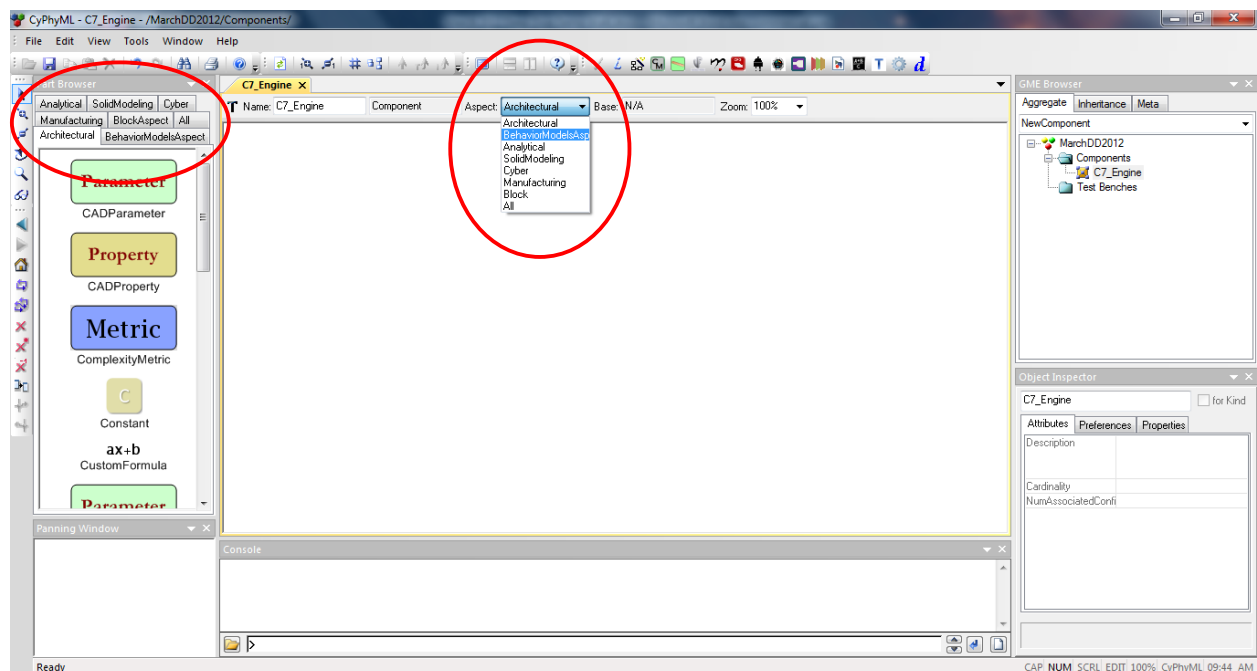


Figure 20. Aspect Control

While these are only two of the many options and controls available to the user, others will be presented at appropriate times throughout this document.

Our `C7_Engine` is based on a dynamics model created in OpenModelica. Therefore, the first step to creating a representative component in GME is to create a `ModelicaModel` component from within the `BehavioralModelAspect`. To do this, the user must first activate the `BehaviorModelAspect`. The user will notice that the GME elements in the Part Browser change for each aspect. One in the appropriate aspect, scroll down until a `ModelicaModel` element is found. The user must drag and drop this component onto the main editing window (see Figure 21).

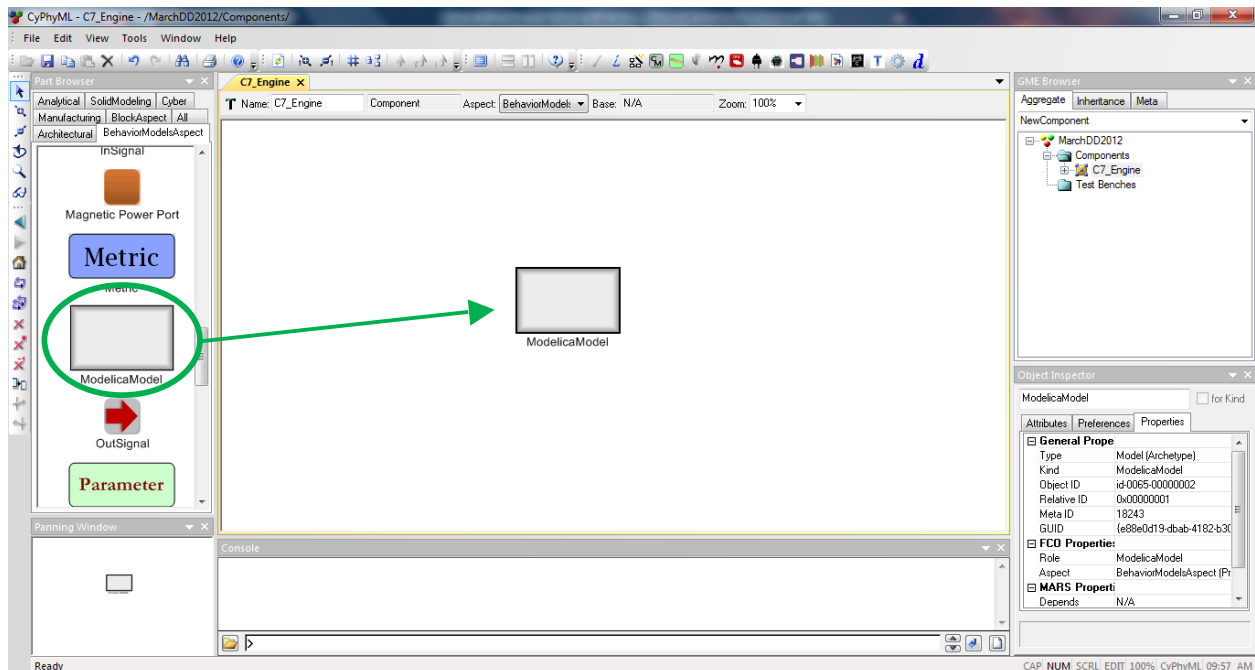


Figure 21. Modelica Model Component

This action has now established our C7 Engine component as a dynamics model that exists in OpenModelica. The user is free to rename the ModelicaModel component at this time by clicking and holding on the text underneath the component or by altering the text in the Object Inspector window (bottom right window). The next steps will involve creating variables within the ModelicaModel component that represent the actual inputs/outputs/interfaces associated with the model.

Double-click on the ModelicaModel. Those elements shown in the Part Browser for the BehavioralModelAspect will be used to establish the inputs, outputs, and interface components associated with this C7 Engine. The C7 Engine has four components in total that have been provided below in Table 2 along with the type of each.

Table 2. C7 Engine Component Names and Types

Component Name	Type
Throttle	Real Input
flange	Rotational Flange
fuelFlow	Real Output
max_torque_val	Parameter

In the Part Browser, locate the Parameter component and drag it onto the main editing window. Rename this component `max_torque_val`. The user's screen should now look like that in Figure 22.

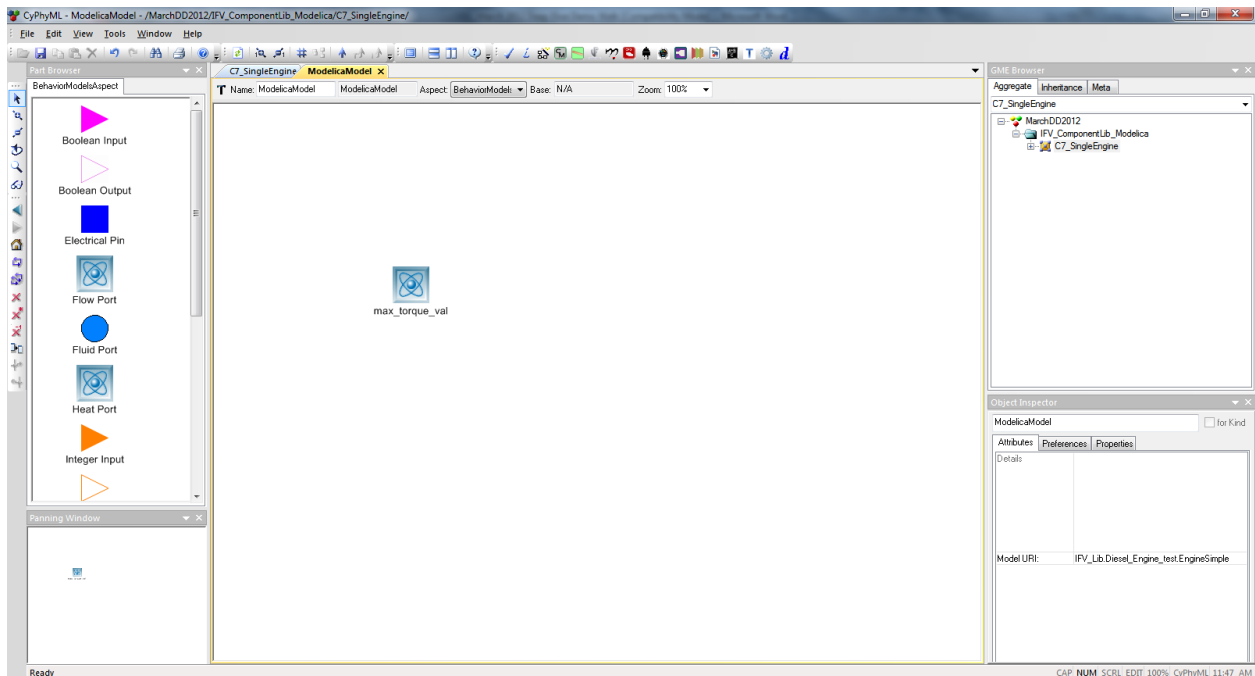


Figure 22. Modelica Model Parameter

Repeat this procedure for all of the elements provided in Table 2. Be sure to rename each element exactly as it appears in Table 2.

Upon completing this step, the user's window should look like that in Figure 23.

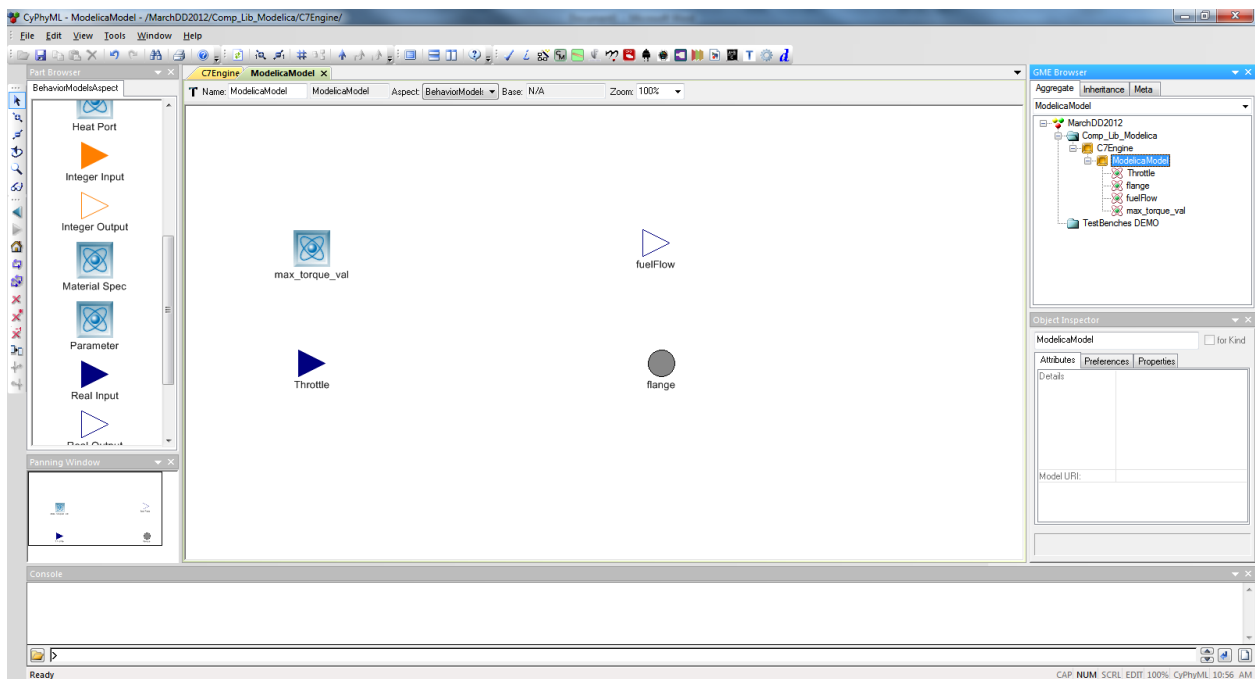


Figure 23. C7Engine Modelica Complete Parameters

Return to the C7Engine component such that you can see the ModelicaModel icon (now with four ports associated with the elements just created). The user must now create four parameterized components that will be used to pass information to the ModelicaModel and allow the C7Engine to interface with other components. To do this, again activate the BehavioralModelAspect tab in the Part Browser. Using the same procedure as the previous step, insert the four elements listed in Table 3.

Table 3. C7 Engine Component Types

Component Type
ElectricalSignalPort (2)
Parameter
RotationalPowerPort

The names of the components that exist outside of the Modelica model are not very important, but for the purpose of this tutorial, it may be easier for the user to copy the names of the components shown in Figure 24 below which shows how the user's window should look after completing this step.

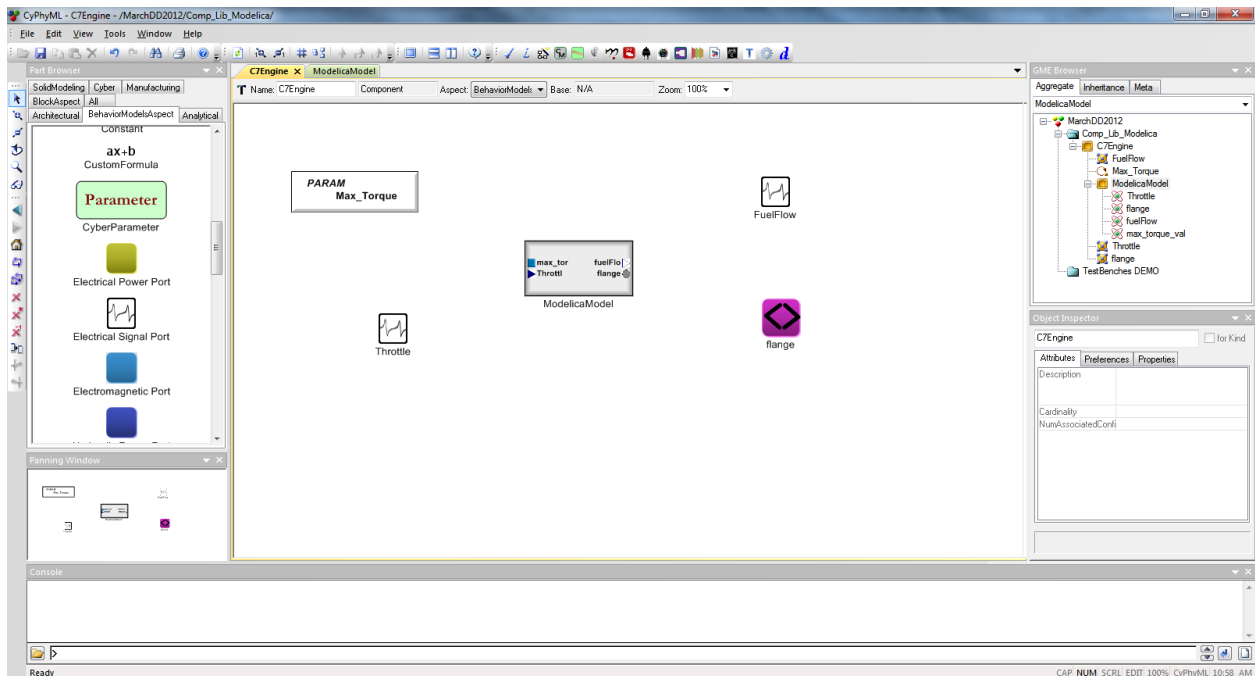


Figure 24. C7Engine Component (No Connections)

From within GME's *Connect Mode* (i.e. leftmost toolbar within the main application window), connect each of the elements to the appropriate ModelicaModel port (see Figure 25).

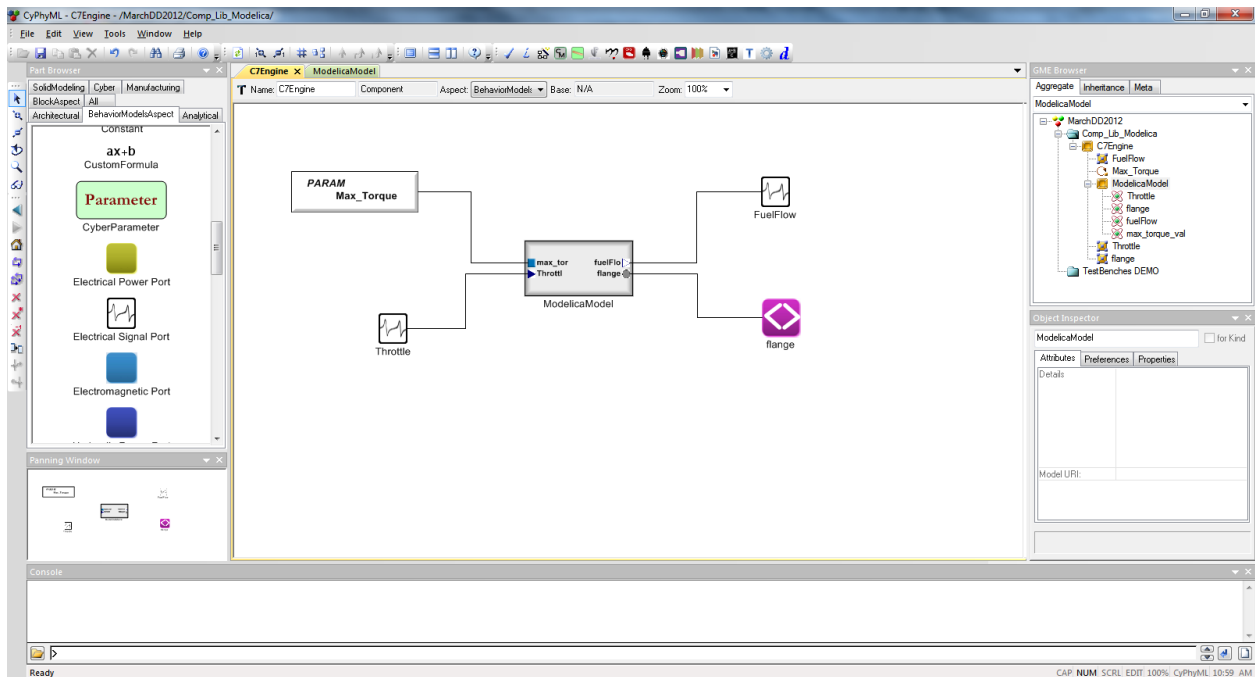


Figure 25. C7Engine Component with Connections

The last step needed to complete the development of the C7 Engine component model is to create a reference to the Modelica model within the Modelica component library (provided by outside source). This is accomplished by typing the Model URL in the Attributes tab located in the Object Inspector window. The format for this reference is much like an object-oriented programming language in that it uses the “dot” notation. The Model URL for the C7 Engine has been provided below along with a screen shot in Figure 26.

C7 Engine URL: IFV_Lib.Diesel_Engine_test.EngineSimple

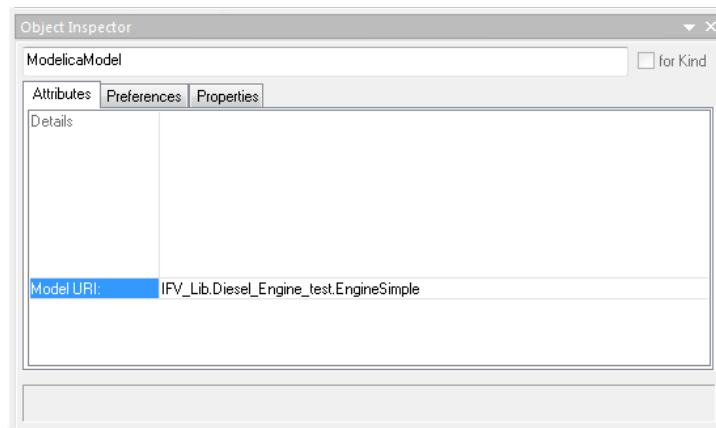


Figure 26. Object Inspector Window: Model URL

Repeat the above process for the components provided in Table 4. The input and output variables have been provided along with the appropriate name for each. Table 5 lists the model URL for each of the components.

Table 4. Component Model List with Inputs/Outputs Identified

Component Name	Inputs (name: type)	Outputs (name: type)
Driveshaft	flange_a: Rotational Flange	flange_b: Rotational Flange
FinalDrive	flange_a: Rotational Flange ratio: Parameter	flange_b_left: Rotational Flange flange_b_right: Rotational Flange
SimpleTransmission	flange_a: Rotational Flange u1: Real Input	flange_b: Rotational Flange
SimpleTransmissionController		y: Real Output

Table 5. Component Models List with Model URLs

Component Name	Model URL
Driveshaft	IFV_Lib.DriveShaft_test.Driveshaft
FinalDrive	IFV_Lib.IdealFinalDrive_test.FinalDrive
SimpleTransmission	IFV_Lib.SimpleTransmission_test.SimpleTransmission
SimpleTransmissionController	IFV_Lib.DriverFunctions.SimpleTransmissionController

Upon completion of these component models, the user should have a component library that has the same format as shown in Figure 27.

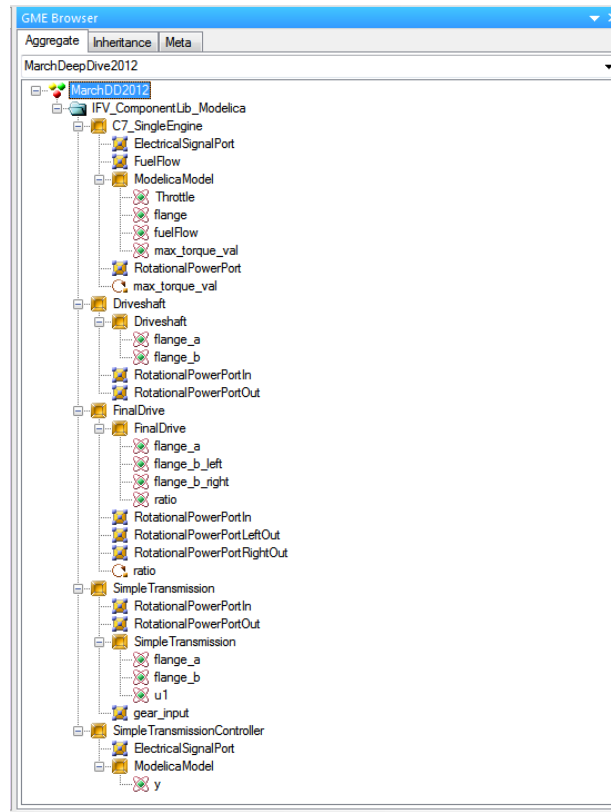


Figure 27. Complete GME Component Library

Creating GME Test Benches

A test bench in GME is a tool used to analyze both individual components and complex systems for verification and validation purposes. For example, a GME test bench for an engine model may involve a throttle profile as an input and record the estimated output torque and rpm. In this case, the simulation would represent a simplified dynamometer test conducted on real engines. In this walkthrough, three test benches will be created that will enable the PET Demo to be simulated. In addition to creating test benches, the user will need to create additional components that will support the successful simulation of the PET Demo. These additional elements will include the definition of two test components and a single workflow element.

Test component are created in the exact same manner as normal components. At this point, the user should already have experience developing component elements in GME so this description will remain brief, but is included for completeness.

Create a new Test Components folder inside of the Testing folder that was created in the beginning of this tutorial. Once created, right-click on the new folder and select the Insert Model option. A pop-up window will appear that should contain Test Component as the only option (select this). Figure 28 provides a screen shot of the window the user should now see.

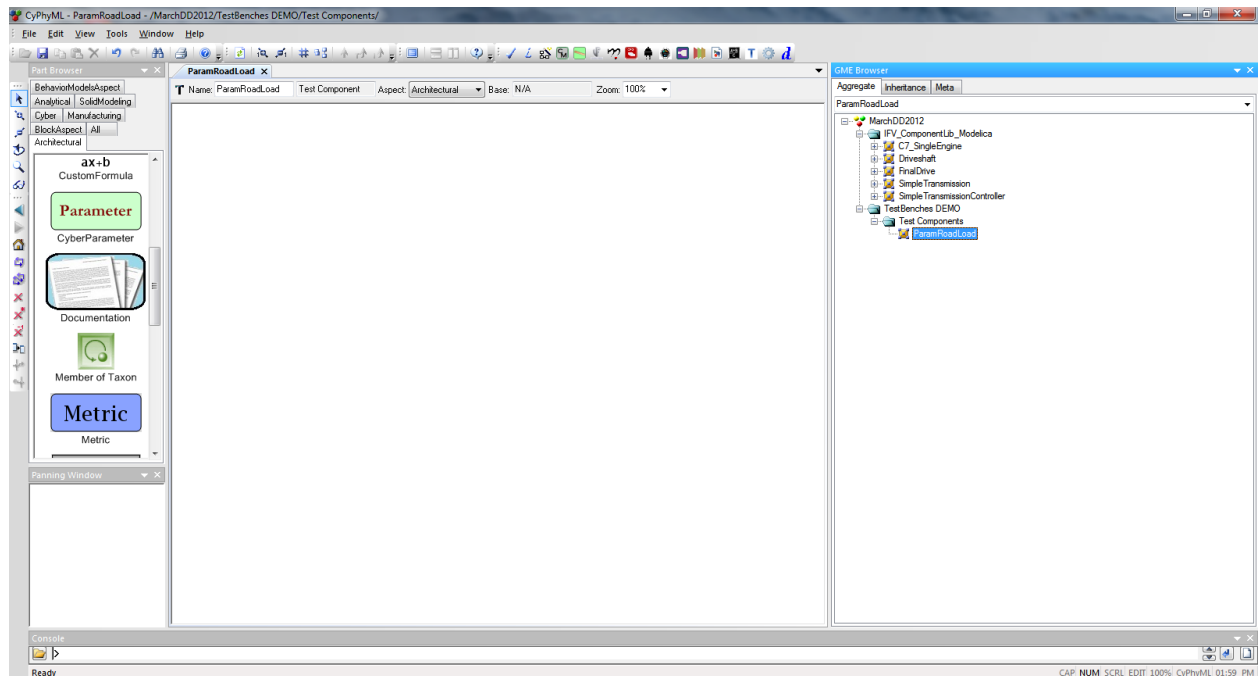


Figure 28. New Test Component

In Figure 28, the test component has already been named to `ParamRoadLoad`. This is the next element that will be created in this tutorial. This is a simple model of the load applied to the wheels of the vehicle during operation. The process of creating a test component is exactly the same as that of the normal components created earlier in this tutorial. The user simply needs to conduct the following steps:

- Identify inputs/outputs of the model in its original program (Modelica)
- Insert a `ModelicaModel` element from the `BehavioralModelAspect`
- Insert the necessary GME elements within the `ModelicaModel` element (with appropriate names)
- Connect the ports of the `ModelicaModel` element to parameterized elements.

Table 6 below provides a summary of the necessary inputs and outputs for the `ParamRoadLoad` model. The Modelica model for this component is `IFV_Lib.RoadLoad.ParamRoadLoad`. Figure 28 provides a screenshot of the completed GME model.

Table 6. `ParamRoadLoad` Component with Inputs/Outputs Identified

Component Name	Inputs (name: type)	Outputs (name: type)
ParamRoadLoad	flange: Rotational Flange wheel_radius: Parameter d: Parameter J: Parameter	Speed: Real Output

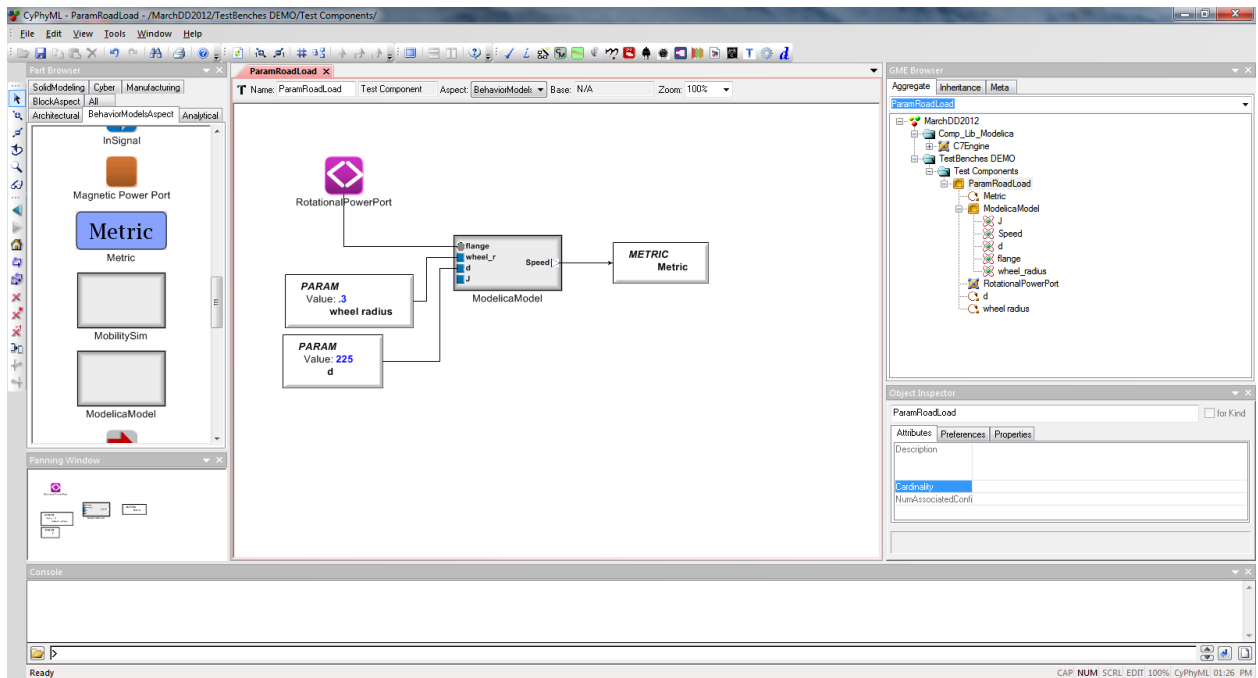


Figure 29. Complete ParamRoadLoad Component

This procedure must also be conducted for the second test component required for the PET Demo. This driver models the input throttle for the engine and is called `ThrottleDriver_100P` in this tutorial. Figure 30 provides a screen shot of this test component. This completes the test components element of this tutorial.

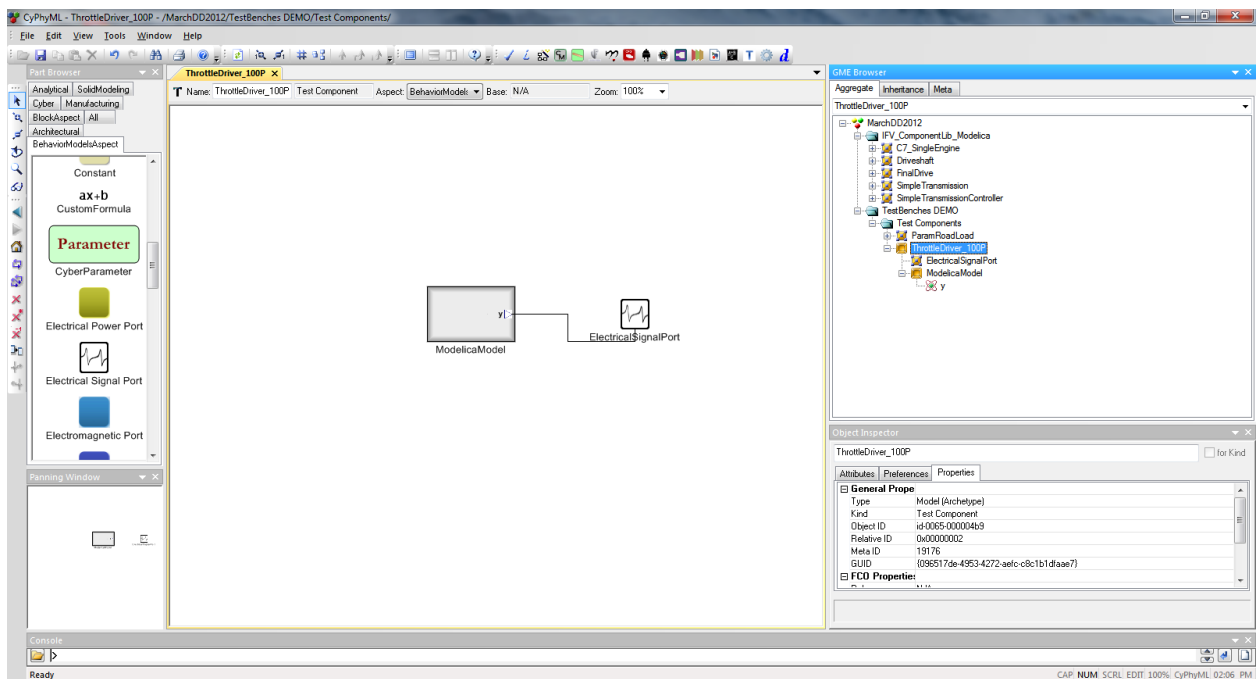


Figure 30. Complete ThrottleDriver_100P Component

The following section will discuss how to create a workflow element in GME. Workflows define a series of interpreters to be automatically invoked when using the Master Interpreter to orchestrate execution of a test bench. To create a workflow element, first right-click on the Testing folder, scroll down to the Insert Folder option and select the Workflow Definitions option from the pop-up window. Right-click on this new folder and insert a new Workflow model (Insert Model -> Workflow). This model will represent a Modelica workflow element so it may be helpful to rename this component. Double-clicking on this component will open up the main editing window which will reveal a single element in the Part Browser that can be added to this element (called a Task element). Upon adding this element to the main editing window, the user will be prompted to select from a list of interpreters built in to the GME program. Select the CyPhy2Modelica interpreter from the list and press OK (see Figure 31).

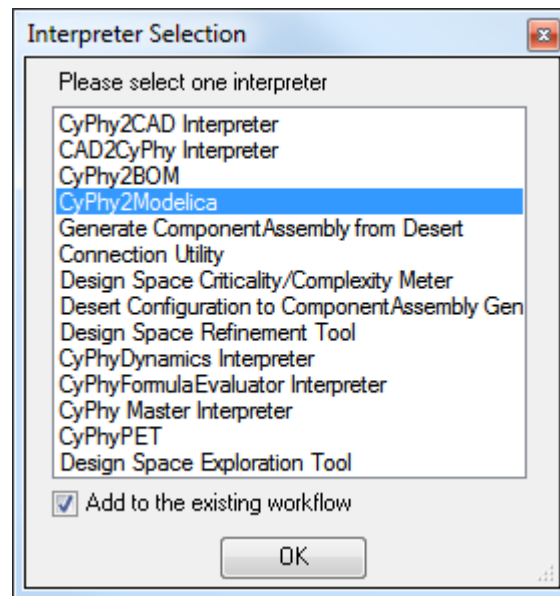


Figure 31. Interpreter Selection Window

The user must add a few lines of code to the Parameters section of the Task element in order for the PET Demo to work properly. To do this, select the task element and activate the Attributes tab in the Object Inspector. In the Parameters section, add the following lines of code (the following lines of code are unique to a component that exists in OpenModelica):

```
{
  "simulationTargetTool": "OpenModelica"
}
```

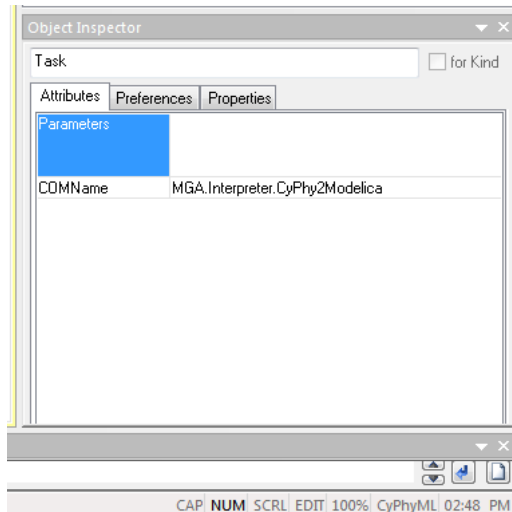
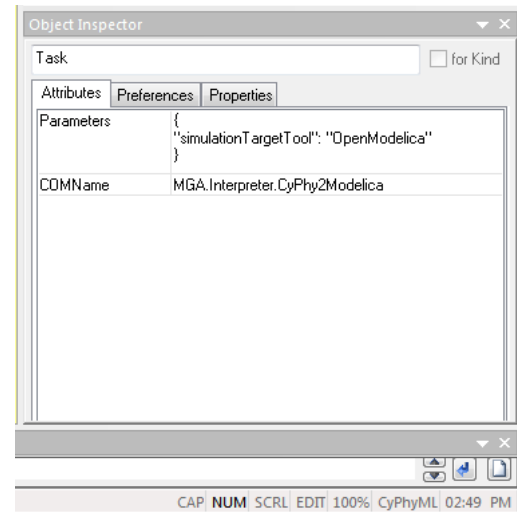


Figure 32. Modelica Workflow Element



This completes the workflow element section and will now enable the user to start creating test benches that will support the PET Demo.

PET Test Benches

This section will describe how to create the test benches that will be synchronized by the PET run. The three test benches that will be created have been listed below:

Test Benches:

- MultiBodyEngineSim
- TorqueComputation
- DriveTrainSim_w_Params

To create a test bench, right-click on the Testing folder and highlight the Insert Model option. A pop-up window should appear with one of the options inside that window being `Test Bench`; select this option and rename the new component `MultiBodyEngineSim`.

Double-click on this new test bench model to activate the main editing window. The Modelica model that represents the `MultiBodyEngineSim` element has two input parameters representing the combustion cylinder diameter and length. A regression then computes an estimated maximum torque delivered by a simple engine with a single cylinder. Therefore, the only GME elements that need to be added to this test bench are two parameters (`CylinderLength`, `CylinderDiameter`) and one metric (`MaxTorqueCyl`). To add these parameters, the user will have to activate the `TBValueFlowAspect` option above the GME elements in the Part Browser or by selecting the appropriate option from the Aspect

drop-down menu located above the main editing window (this is the same process used for adding GME elements to the component models). Upon completing this step, the user's window should look like that in Figure 33.

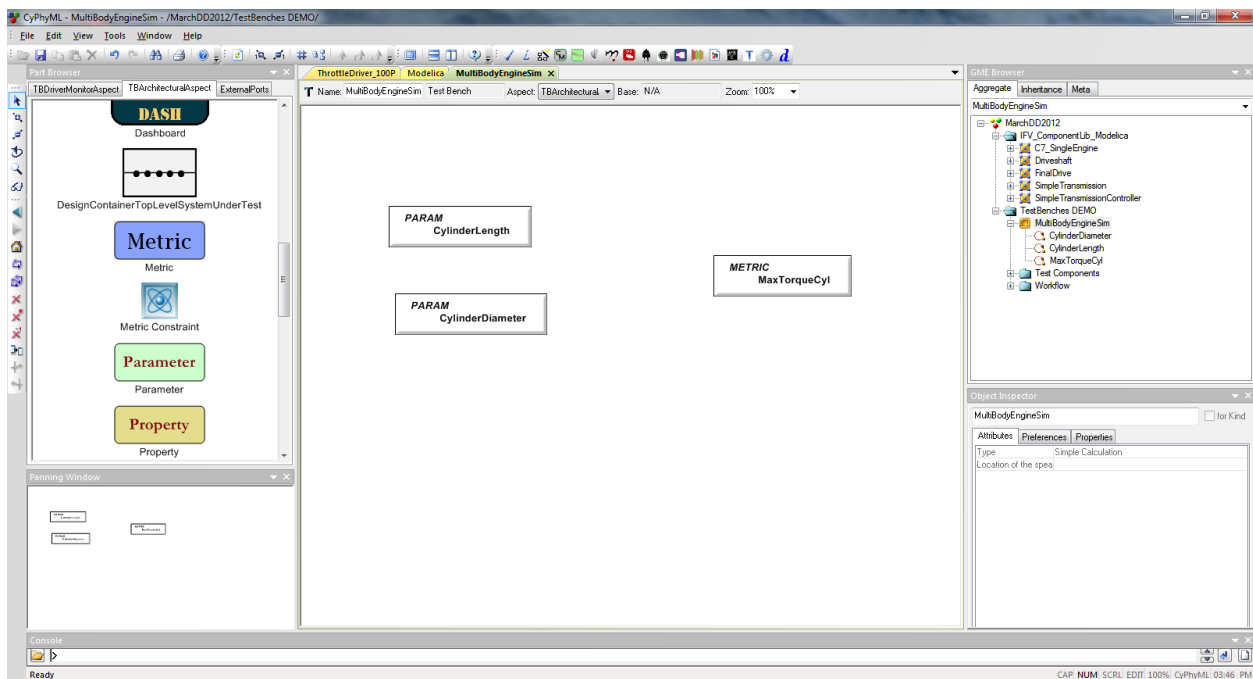


Figure 33. MultiBodyEngineSim Test Bench

One additional GME element must be added to this test bench in order for it to be utilized in the PET Demo: a workflow reference. Activate the `TBDriverMonitorAspect` from the Part Browser and scroll to the bottom of the GME element list. Drag and drop the `WorkflowRef` element onto the main editing window (see Figure 34).

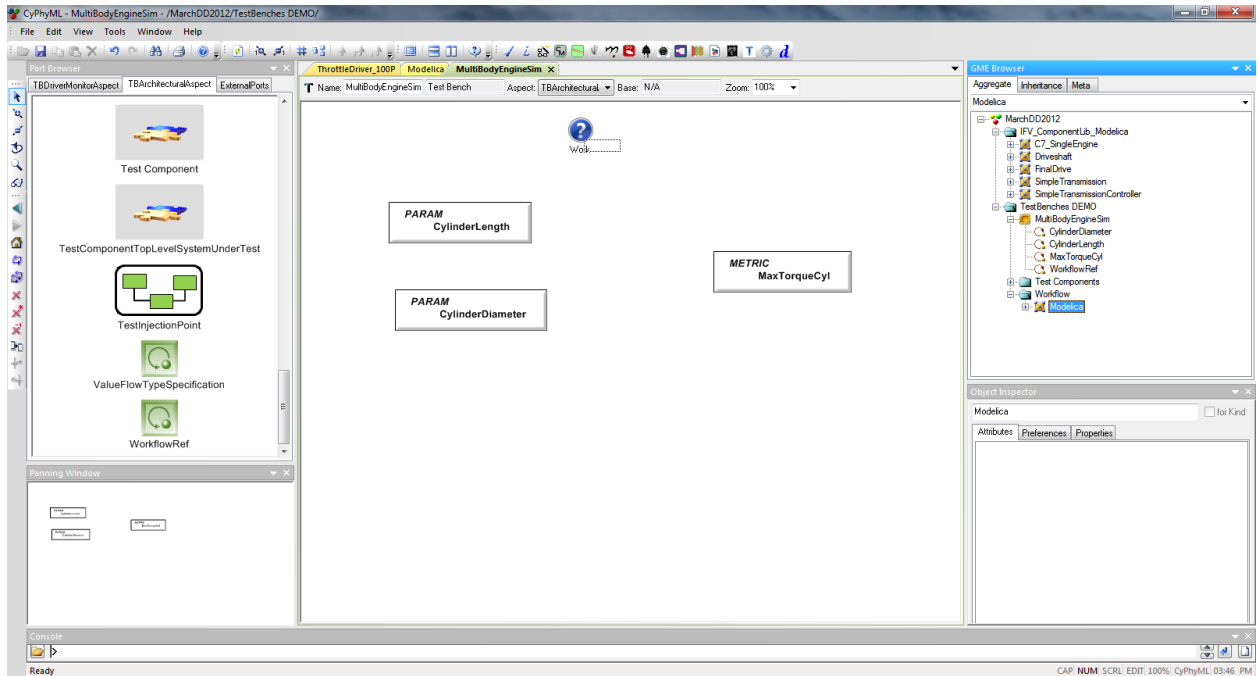


Figure 34. MultiBodyEngineSim Test Bench with Workflow Element

Locate the Modelica workflow reference within the Workflow folder. Drag this element onto the main editing window and drop it directly on top of the WorkflowRef element (your cursor must be directly over the WorkflowRef element for this to work). If done properly, the WorkflowRef element icon will change to a simple document-like image with two gears (see Figure 35).

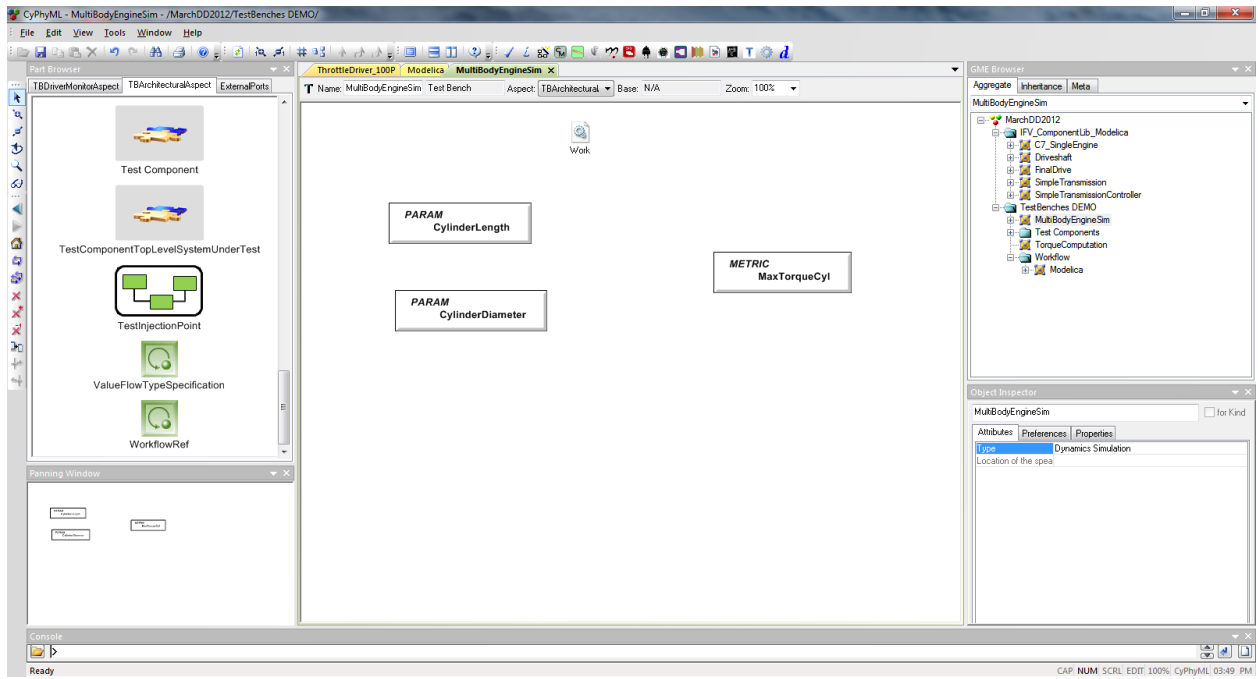


Figure 35. MultiBodyEngineSim Test Bench with Complete Workflow Element

To check that this has worked, double-click on the WorkflowRef element and make sure that the Task created previously for the Modelica workflow element has referenced correctly (see Figure 36). This completes the development of the MultiBodyEngineSim test bench.

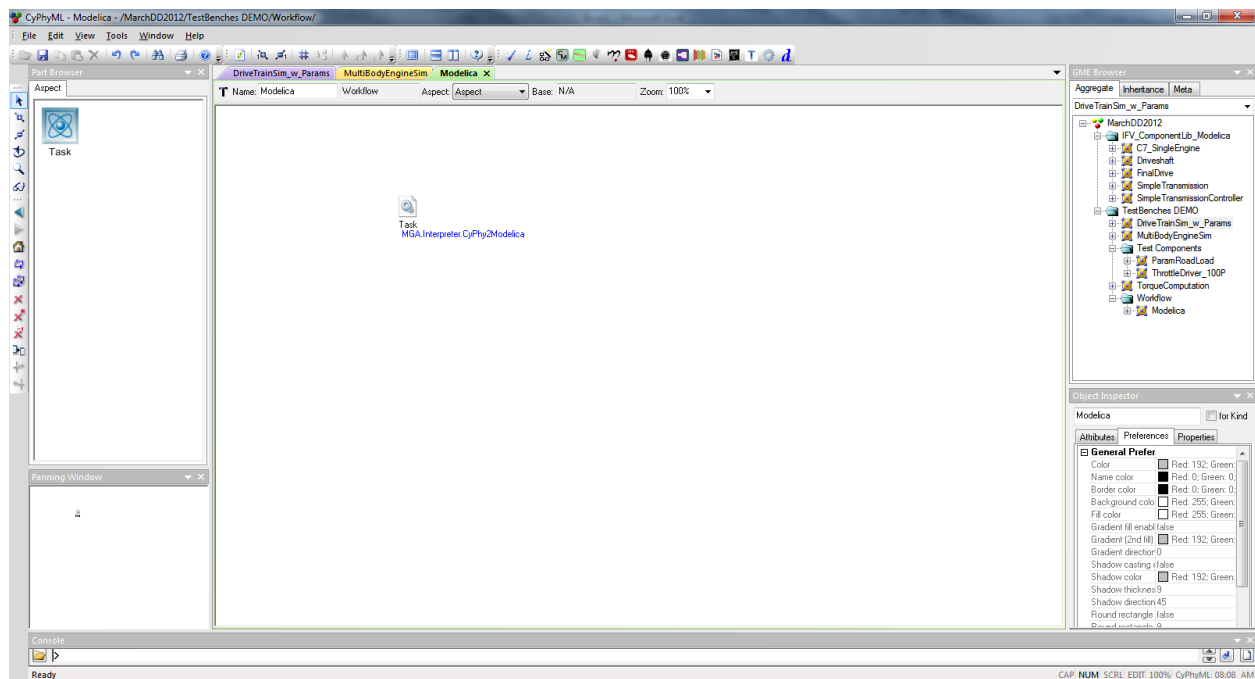


Figure 36. Workflow Element Check

The last step needed to complete the MultiBodyEngineSim test bench is to create a reference to the Modelica model by activating the Attributes tab in the Object Inspector and typing in the following: `MultiBodyEngineSim.mo` for the location attribute. Notice that this is different from the model URL used when creating the components; the reason for this is that the Modelica model for this component exists in the same directory as the GME project (i.e. is it not part of the pre-defined Modelica *component Library*).

The TorqueComputation test bench will now be created. This test bench represents the excel spreadsheet that takes in the maximum torque provided by a single cylinder engine (simulated by MultiBodyEngineSim in this tutorial) and uses simple regressions to estimate the torque delivered by an engine with N cylinders. Figure 37 provides a screen shot of the Excel spreadsheet.

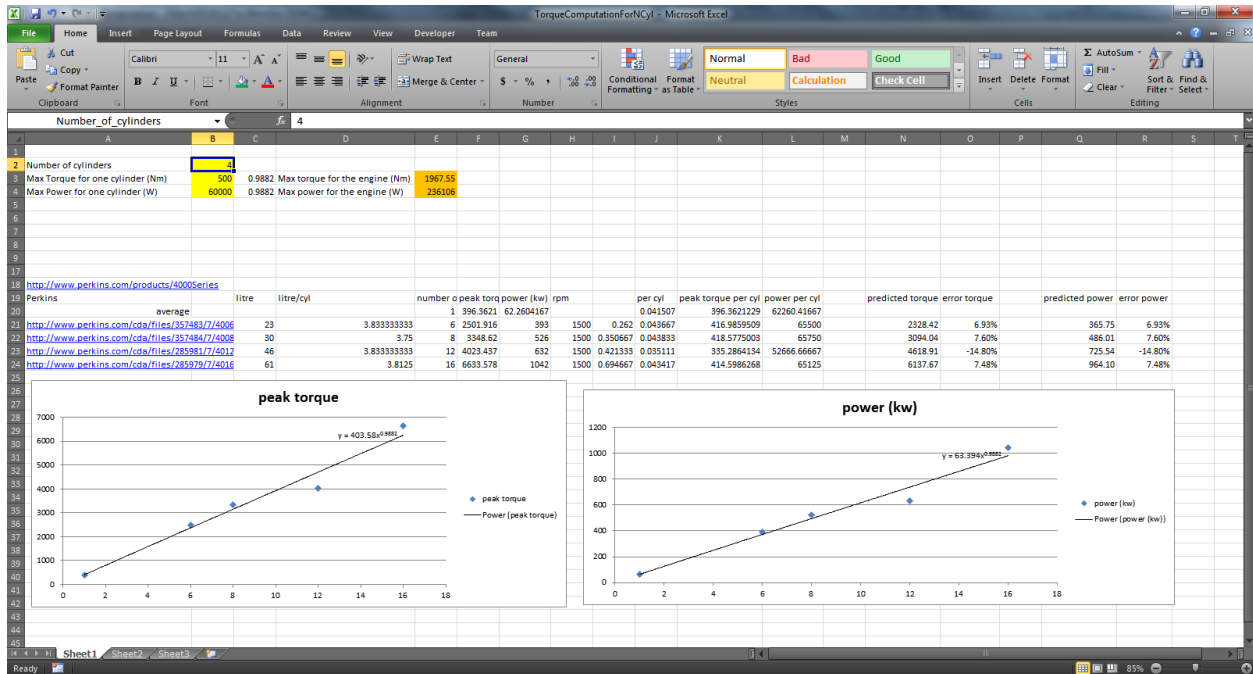


Figure 37. TorqueComputation Excel Spreadsheet

GME will input and extract information from the Excel spreadsheet using named ranges. This can be seen in Figure 37 - cell B2 has been given the name "Number_of_cylinders" which is self-explanatory but represents the variable N that has been discussed in previous sections. The use of these named ranges will become apparent in the following steps.

Create a new test bench called `TorqueComputation` by right-clicking on the Testing folder, scrolling down to Insert Model, and clicking on Test Bench. Double-click on the test bench to activate the main editing window. Now, instead of dragging GME elements from the Part browser, activate the Attributes tab in the Object Inspector (lower right screen). In this tab, identify the Type option and select it (it should highlight blue). Please see Figure 38 for reference.

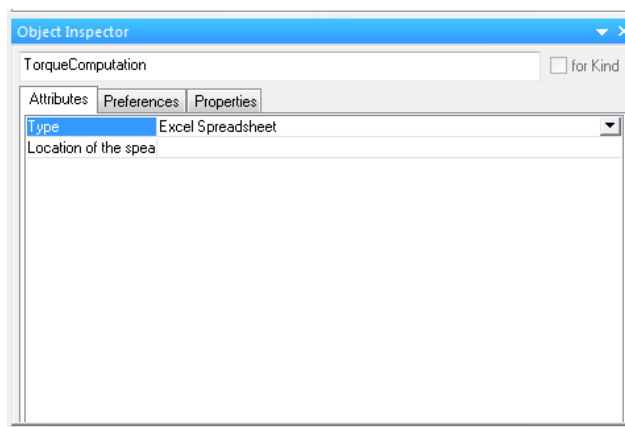


Figure 38. Object Inspector: Excel Spreadsheet Type Definition

Click on the drop-down arrow on the far right and select the Excel Spreadsheet option from the menu that appears. Upon selecting this option, the user will be prompted to provide the directory of the Excel spreadsheet. The spreadsheet was created as in the Microsoft Office 2010 suite so the user will have to change the file type in the pop-up window to .xlsx (see Figure 39).

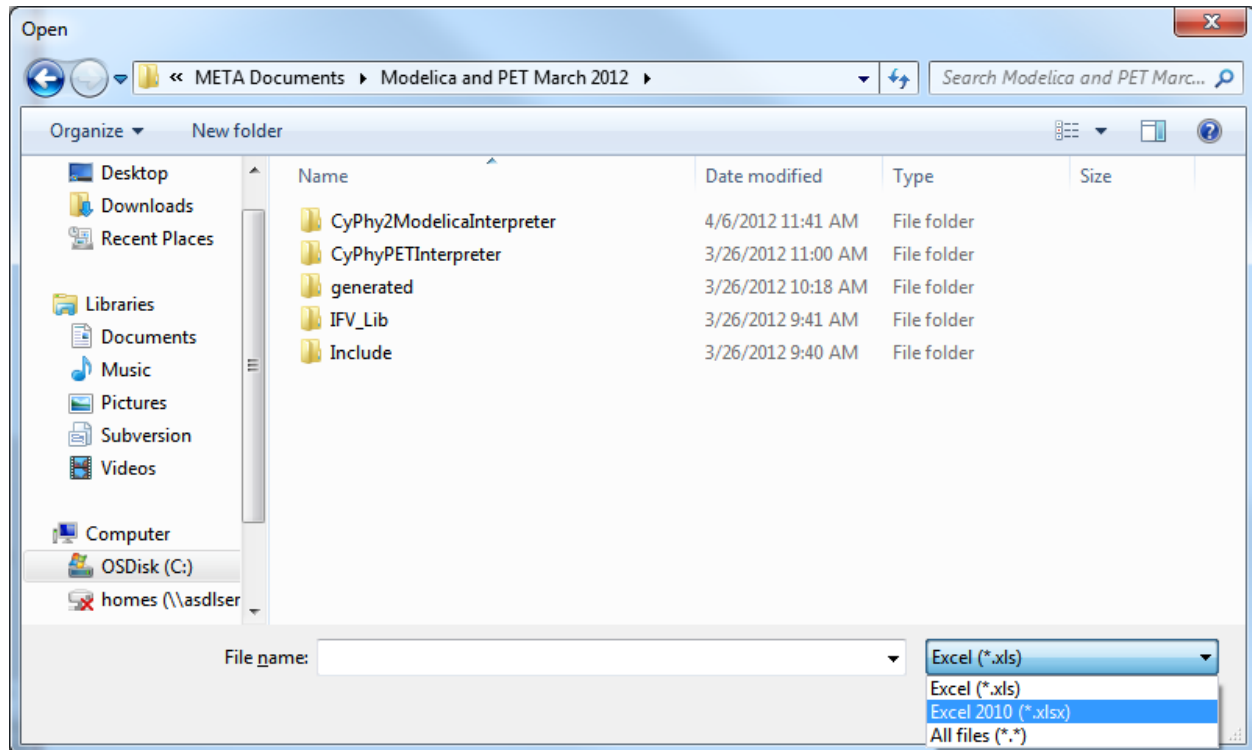


Figure 39. Excel Spreadsheet Directory Identification

The TorqueComputationForNCyl spreadsheet should now be visible in the pop-up window (see Figure 40).

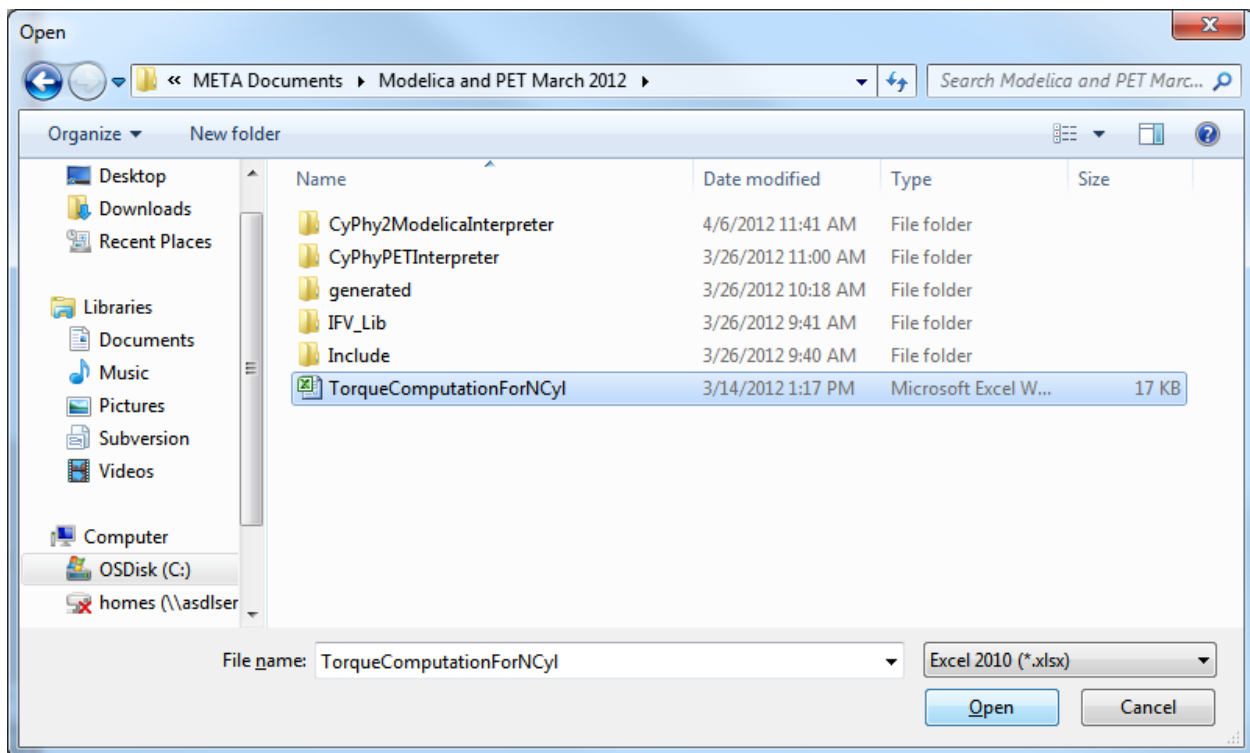


Figure 40. Excel Spreadsheet Selection

Highlight the spreadsheet and click ok. This will activate the `Excel Import Form`, which will allow the user to identify the parameters (inputs) and metrics (outputs) that are represented in the spreadsheet. The user should see five (5) options in the middle window of the form under `Detected named cells`. These are the named ranges that were identified in the spreadsheet (previously, we discussed cell B2 being named `Number_of_cylinders` which can now be among the list of named cells). To move the identified ranges to the `Parameters` and `Metrics` windows, the user must highlight the appropriate names and select the directional arrows found on either side of the middle window. The parameters and metrics have been identified in Table 7 and a series of screen shots have been provided in Figure 41 to provide visual aid.

Table 7. `TorqueComputation ExcelSpreadsheet Variables`

Name	Type
<code>Max_Power_for_one_cylinder_W</code>	Parameter
<code>Max_power_for_the_engine_W</code>	Metric
<code>Max_torque_for_one_cylinder_Nm</code>	Parameter
<code>Max_torque_for_the_engine_Nm</code>	Metric
<code>Number_of_cylinders</code>	Parameter

ExcelImportForm

Parameters

Detected named cells

Max_Power_for_one_cylinder__W
Max_power_for_the_engine__W
Max_Torque_for_one_cylinder__Nr
Max_torque_for_the_engine__Nm
Number_of_cylinders

Metrics

< >

> <

OK

ExcelImportForm

Parameters

Detected named cells

Max Power for one cylinder __W
Max_power_for_the_engine__W
Max_Torque_for_one_cylinder__Nr
Max_torque_for_the_engine__Nm
Number_of_cylinders

Metrics

< >

> <

OK

ExcelImportForm

Parameters

Max_Power_for_one_cylinder__W
Max_Torque_for_one_cylinder__Nr
Number_of_cylinders

Detected named cells

Max_power_for_the_engine__W
Max_torque_for_the_engine__Nm

Metrics

< >

> <

OK

ExcelImportForm

Parameters

Max_Power_for_one_cylinder_W
Max_Torque_for_one_cylinder_Nm
Number_of_cylinders

Detected named cells

Max_power_for_the_engine_W
Max torque for the engine Nm

Metrics

< >

OK

ExcelImportForm

Parameters

Max_Power_for_one_cylinder_W
Max_Torque_for_one_cylinder_Nm
Number_of_cylinders

Detected named cells

Metrics

Max_power_for_the_engine_W
Max torque for the engine Nm

< >

OK

Figure 41. Excel Import Form

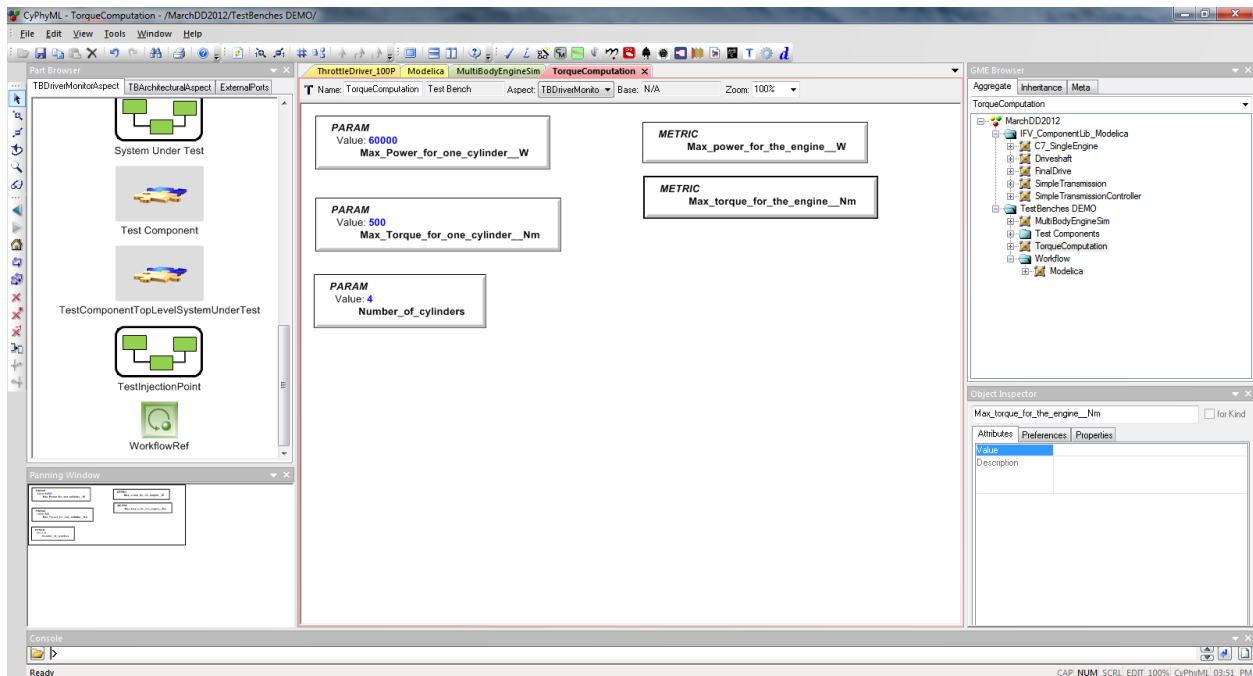


Figure 42. Complete TorqueComputation Component

When the parameters and metrics have been identified and allocated to the appropriate windows, select the ok button at the bottom of the form. This action will automatically generate the parameter and metric components in the main editing window (see Figure 42). This completes the development of the TorqueComputation test bench.

The `DriveTrainSim_w_Params` test bench will now be created. Create a new test bench in the same manner as the first two test benches and rename it `DriveTrainSim_w_Params`. Double-click on the new test bench to activate the main editing window. This test bench will utilize the components and test components that were created earlier in this tutorial. To begin, locate the `ThrottleDrive_100P` component from the Test Components folder. Right-click on this component and select the `Copy` option. Move your cursor into the main editing window and right-click to activate the pop-up window. From this window, highlight the `Paste Special` option and select `As Instance`. This will activate a pop-up window titled `Reserve Role Type` (see Figure 43). From this window, select the `TestComponent` option and select `OK`.

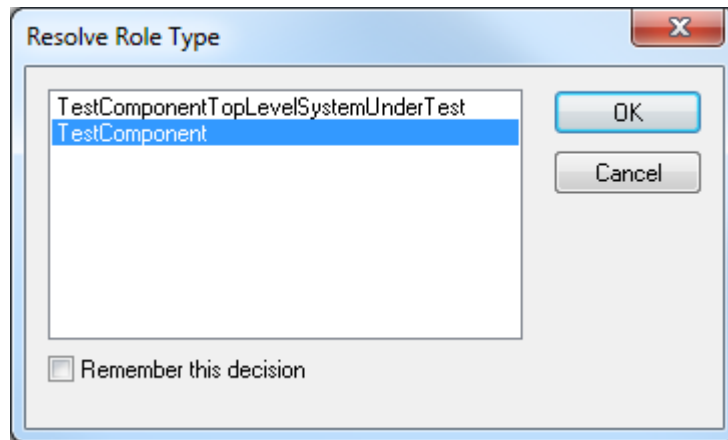


Figure 43. TestComponent Role Type Window

This will create an instance of the ThrottleDrive_100P test component in the main editing window. The single output, *y*, that was associated with this component's Modelica model is now visible as an electrical signal port (see Figure 44).

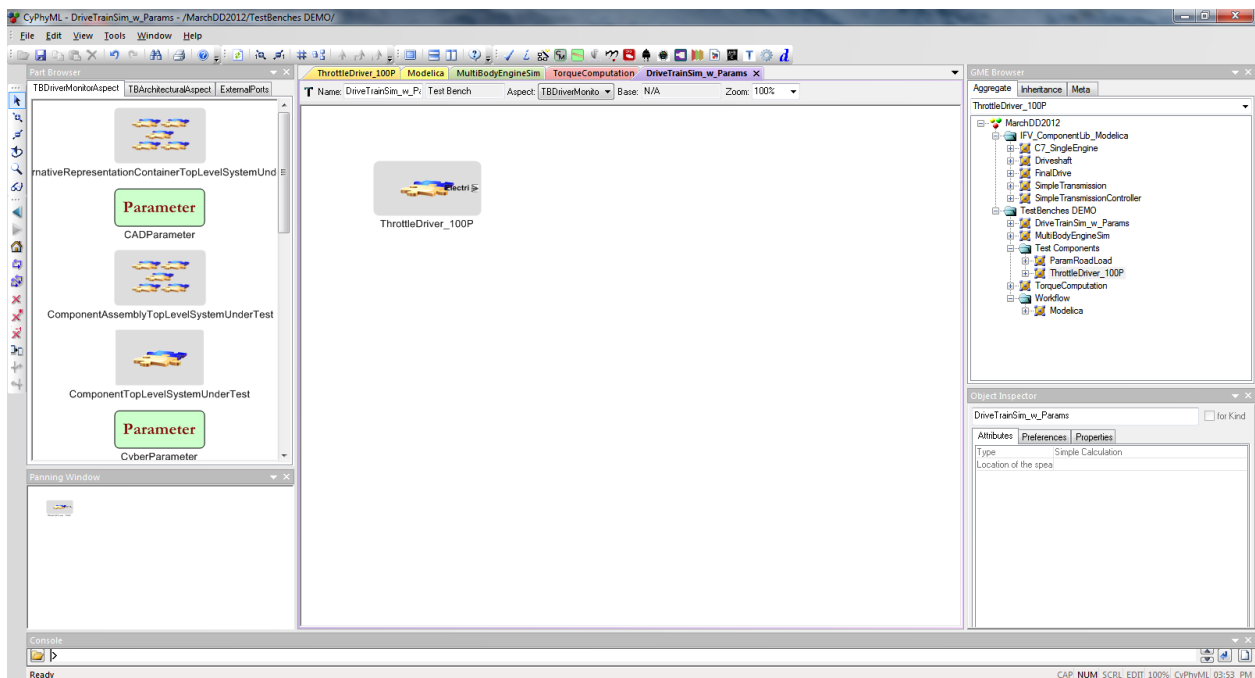


Figure 44. ThrottleDrive_100P Instance in DriveTrainSim_w_Params Test Bench

Repeat the above process for all of the components located in the Components folder in the following order shown below. It is important to note that when pasting these components as instances in the main editing window, the user will not be prompted with the Reserve Role Type window (this only occurs for test components).

- 1) C7_SingleEngine
- 2) SimpleTransmission
- 3) Driveshaft
- 4) FinalDrive
- 5) SimpleTransmissionController

Upon completion of these steps, the user's main editing window should look like that in Figure 45.

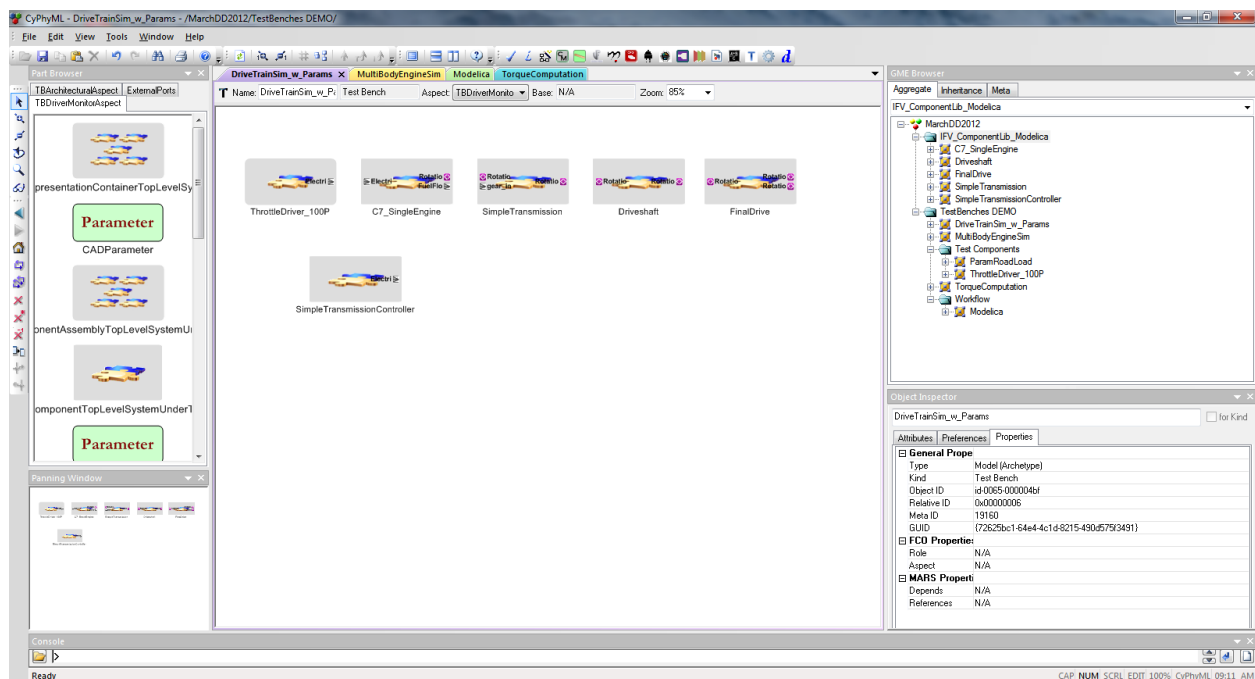


Figure 45. DriveTrainSim_w_Params Test Bench with Components

It will be helpful to ensure that the arrangement of components in all aspects remains the same when designing complex test benches such as DriveTrainSim_w_Params. To accomplish this, select the Synchronize Aspects icon from the top GME toolbar (shown in Figure 46) while the DriveTrainSim_w_Params test bench is active in the main editing window.

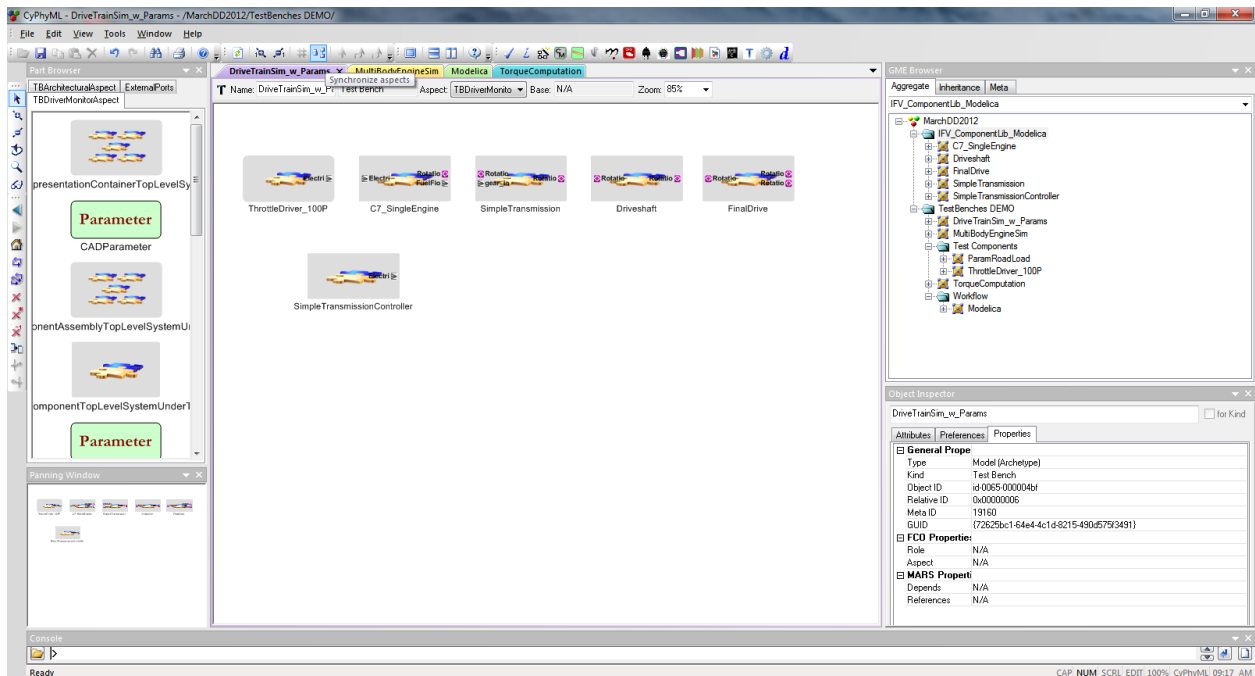


Figure 46. Synchronize Aspects Icon

Selecting this icon will activate a pop-up window. The three aspects associated with the active test bench will be shown in the upper right window. Highlight all three aspects and select the ok button (shown in Figure 47).

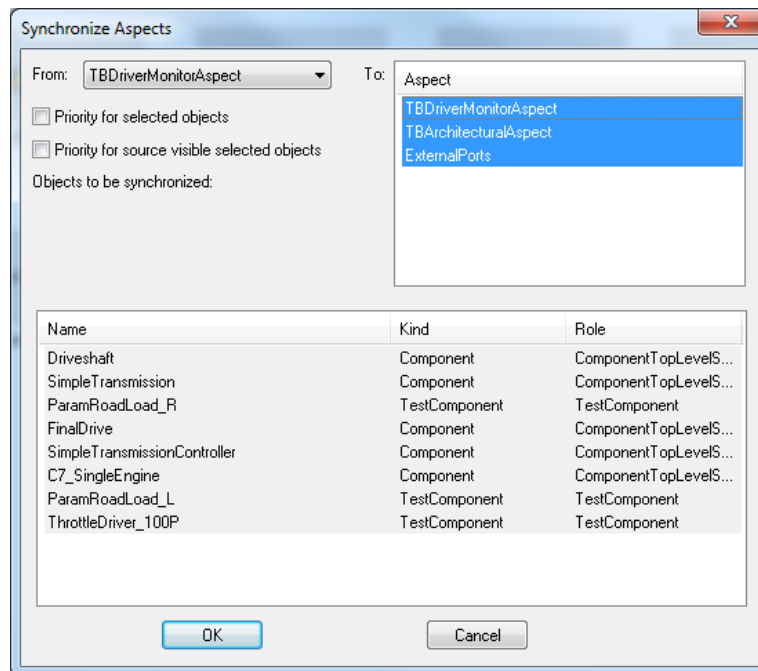


Figure 47. Synchronize Aspects Window

This will ensure that the organization of icons and parameters within the test bench do not shift when changing aspects.

The final two components that need to be added to the main editing window are instances of the test component `ParamRoadLoad` (located in the Test Components folder). Follow the procedure outlined for the `ThrottleDrive_100P` component and be sure to add two instances to the main editing window. Once completed, rename each component by adding a left and right identifier (e.g. add an “_L” and “_R” to the end of each component’s name). The main editing window should now look like that in Figure 48.

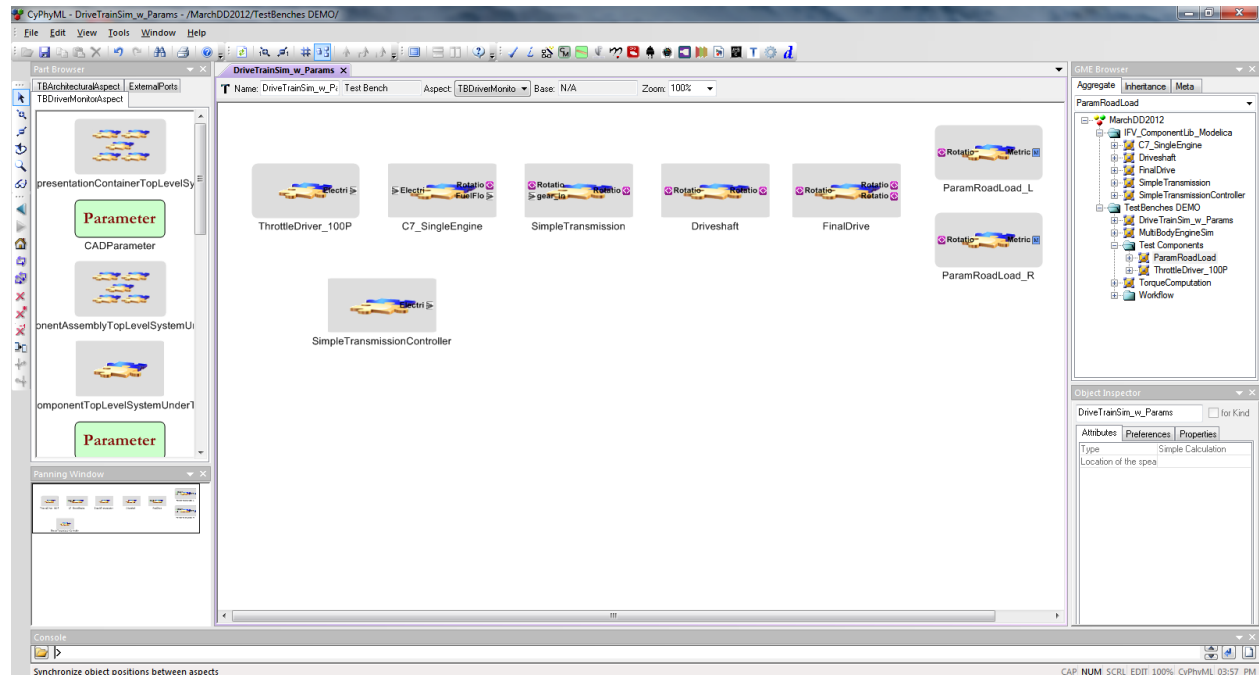


Figure 48. DriveTrainSim_w_Params Test Bench with All Components

Finally, the user must add parameter and metric elements that will pass values to the various input ports associated with the test bench components. To do this, activate the `TBDriverMonitorAspect` window in the Part Browser and insert the five (5) parameters and one (1) metric provided below in Table 8. In addition, Table 8 provides the user with the values for each parameter and the component ports they must connect to. Figure 49 provides a screen shot of how the user’s main editing window should look upon completing this step. It is important to note that connections between the parameters and component ports were conducted in the `TBValueFlowAspect` aspect (these ports will not be visible in other aspects).

Table 8. Parameters Associated with DriveTrainSim_w_Params Test Bench

Parameter	Value	Connection Port
FinalDriveRatio	8	ratio (Final Drive component)
Wheel_radius	0.6	wheel_radius (ParamRoadLoad_L and _R)
Load	225	d (ParamRoadLoad_L and _R)
MaxTorque	1000	Max_torque_val (C7_SingleEngine)
StopTime	80	No Connection

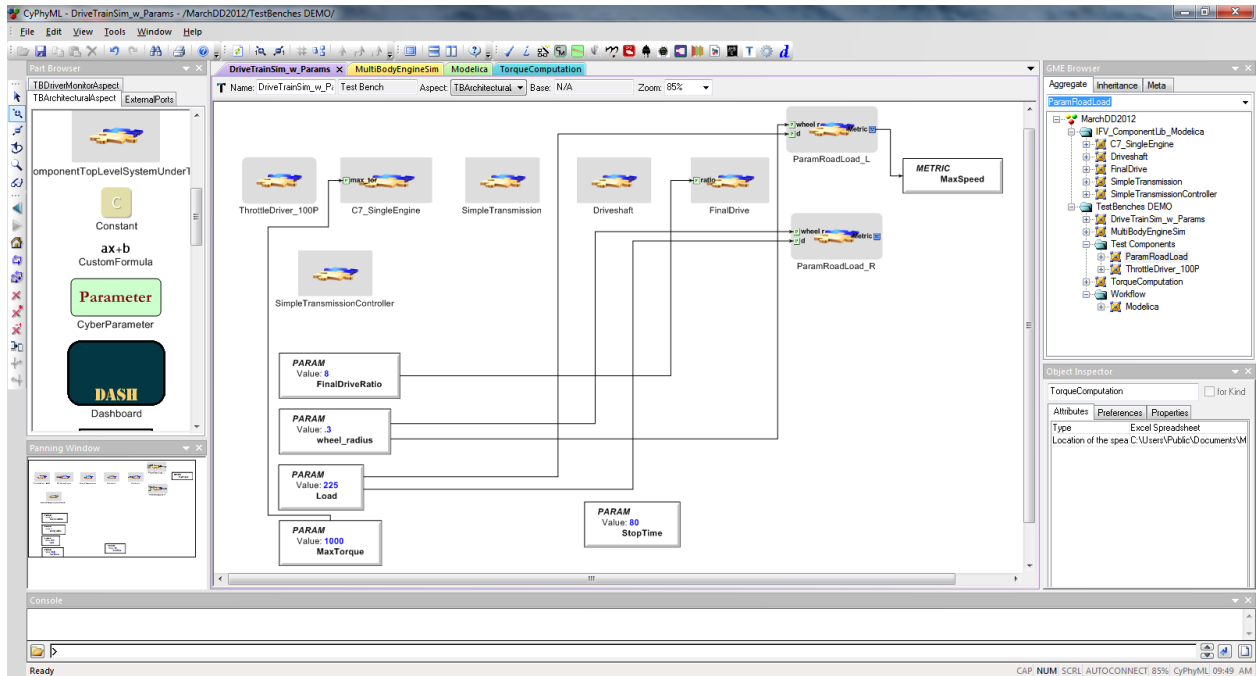


Figure 49. Complete DriveTrainSim_w_Params Test Bench

Now activate the TBDriverMonitorAspect and connect the ports that will allow information to “flow” between components. Figure 50 provides a screen shot of how the user’s window should look after completing this step.

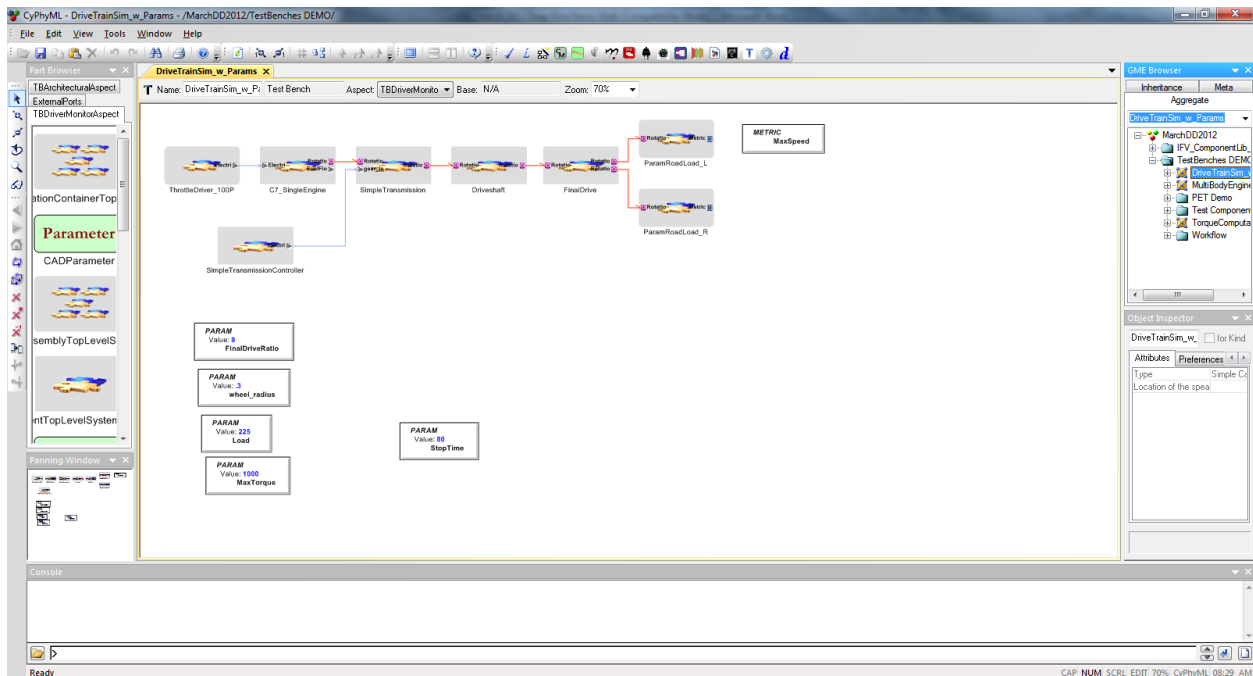


Figure 50. DriveTrainSim_w_Params Component: TBDriverMonitor Aspect

Notice that the `FuelFlow` port associated with the `C7_SingleEngine` component is not linked to anything. This is an important feature of GME that not all ports or parameters must be linked. A component may have dozens of input and output ports and parameters associated with it but not all are necessarily required for some particular analysis scenario. Only those that are of interest to the given test bench must be used. The final step in creating the `DriveTrainSim_w_Params` test bench is to insert a Modelica workflow reference (exact same procedure as was used for the `MultiBodyEngineSim` test bench). Adding this element completes the development of the `DriveTranSim_w_Params` test bench.

It is now time to create the PET Demo. Create a new `Parametric Exploration` folder by right-clicking on the `Testing Components` folder, highlighting the `Insert Folder` option, and selecting `Parametric Exploration`. Rename this folder to `PET Demo`. Insert a new `Parametric Exploration` model within this folder and rename it `PET_Experiment_Demo`. Double-click on this new component to activate the main editing window. The user should only have one aspect available to them and a limited number of GME elements should be visible in the Part Browser (see Figure 51).

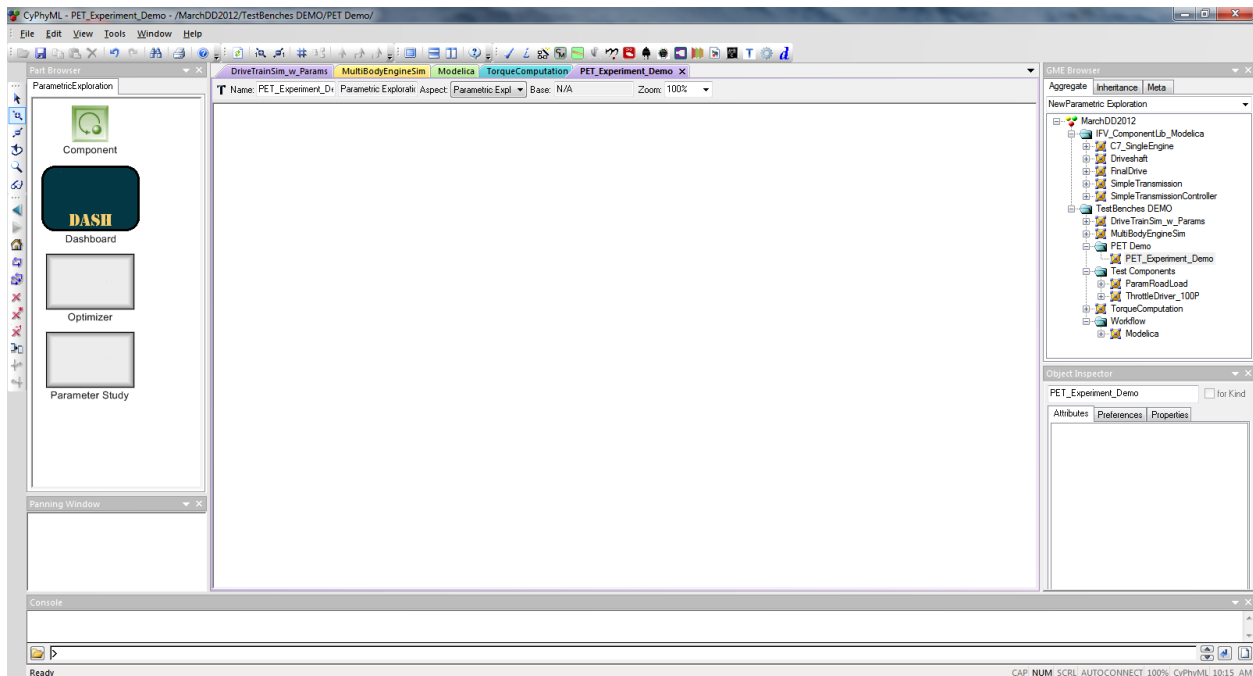


Figure 51. Parameter Exploration Window

Drag a Parameter Study element from the Part Browser onto the main editing window and rename it DOE. This component of the PET Demo will represent the type of DOE to be simulated, the number of levels, and all design variables. Click once on the DOE component in order to activate it. In the Object Inspector select the attributes tab and enter in the following information:

Type: FullFactorial
 Code: num_levels = 2
 IsPrimary: False

This established that the PET Demo will consist of a full factorial DOE with two levels for each design variable. Figure 52 shows how the user's screen should look.

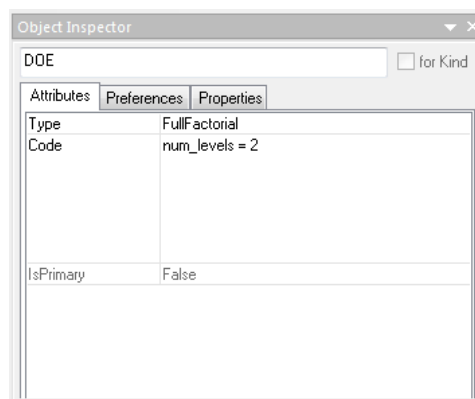


Figure 52. DOE Type and Required Code

Double-click on the DOE component and add a single `Design Variable` element from the Part Browser. Rename this element to `RoadLoad` and enter the following data into the Range variable located in the Attributes tab in the Object Inspector:

Range: 200, 250

It is important to ensure that there is a space between the comma and the second entry (see Figure 53 for reference).

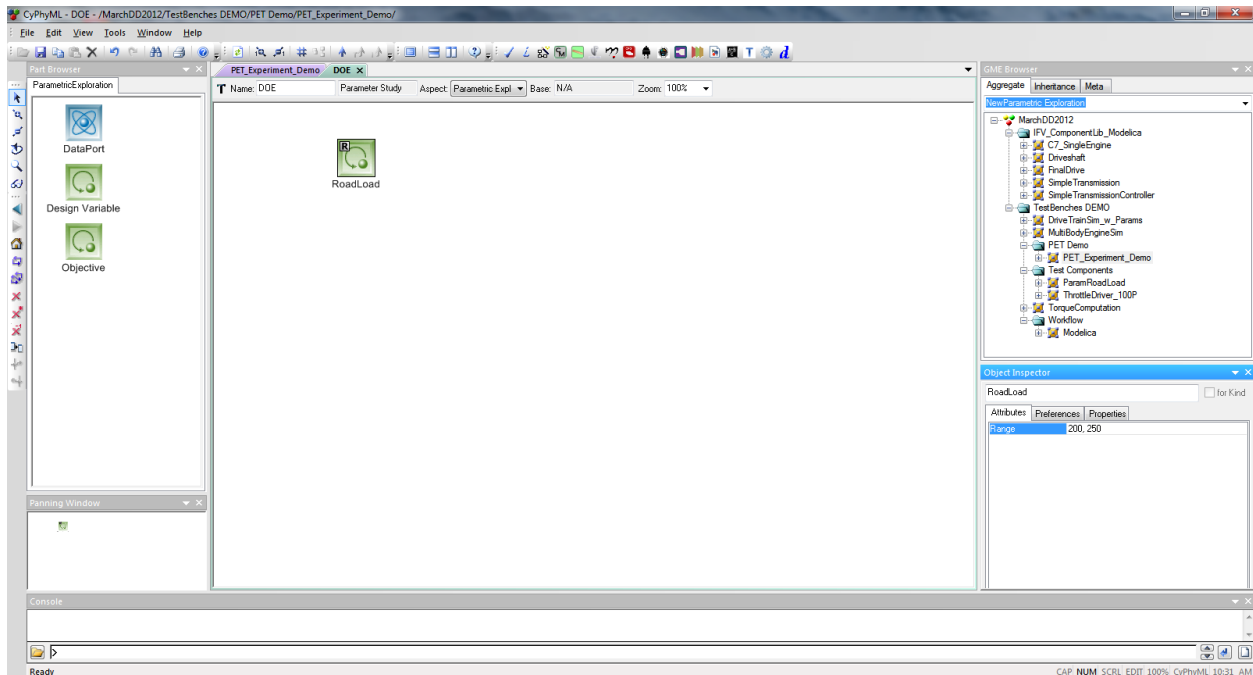


Figure 53. DOE Design Variables

Repeat this procedure for all design variables and ranges provided in Table 9.

Table 9. List of DOE Design Variable and Appropriate Ranges

Design Variable	Range
FinalDriveRatio	7.5, 8.5
CylinderLength	0.2, 0.3
CylinderDiameter	0.12, 0.16
NumberOfCylinders	6, 8

In addition, add a single `Design Variable` element and rename it `MaxSpeed`. The user's window should now look like that in Figure 54.

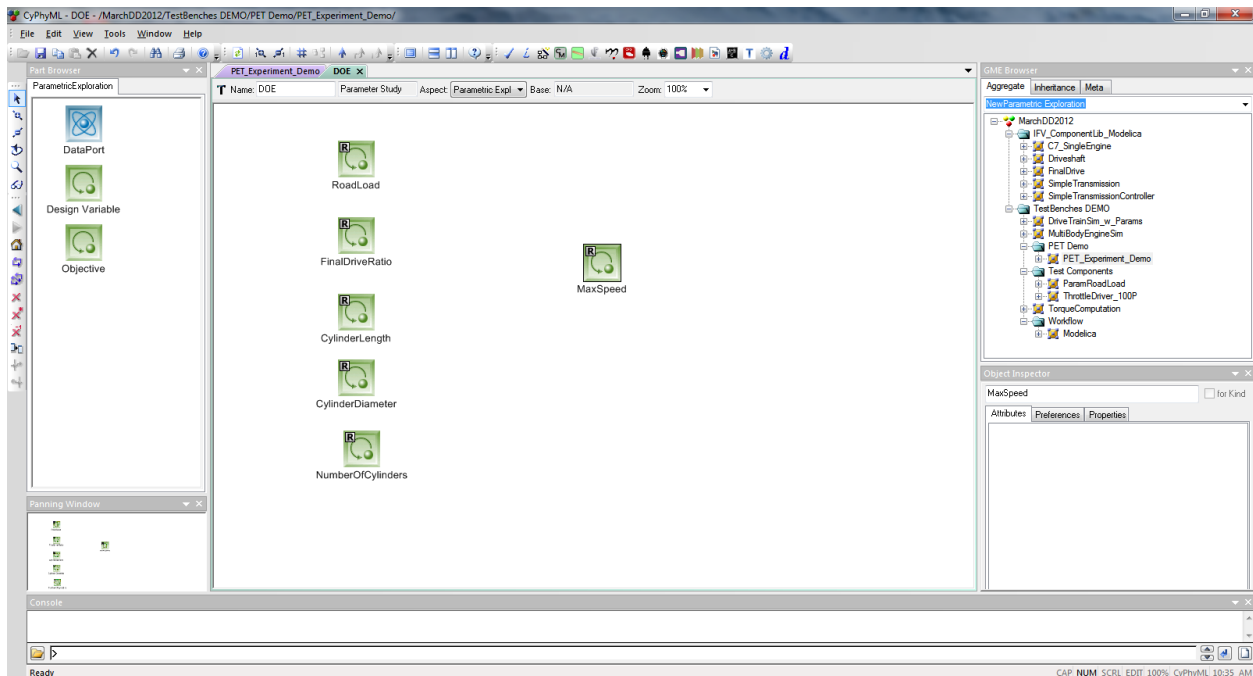


Figure 54. Complete DOE Design Variables and Objective

This completes the development of the DOE element. The user should now return to the PET_Experiment_Demo element main window.

It is now time to create test bench references for the PET Demo. Drag a Component element from the Part Browser onto the main editing window. Locate the MultiBodyEngineSim test bench within the Testing Folder and drag it onto the TestBenchRef element just created in the main editing window (see Figure 55). Make sure that your cursor is directly over this TestBenchRef element prior to releasing the test bench component. Rename this component to MultiBodyEngineSim.

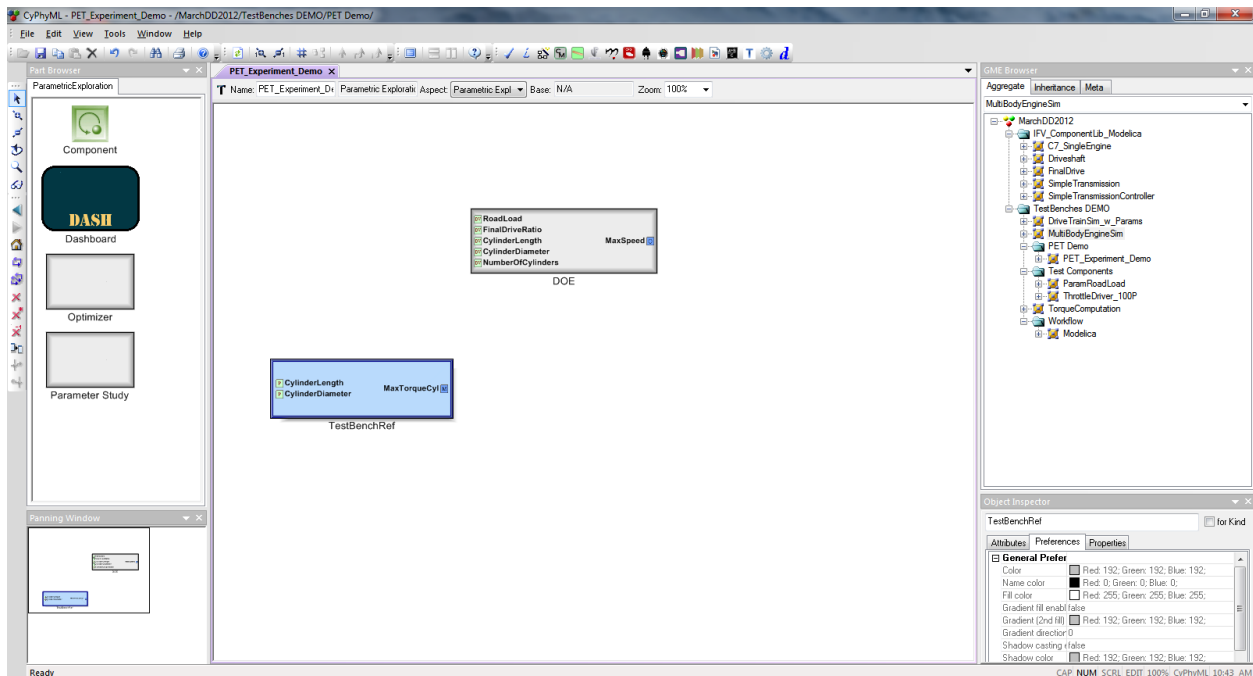


Figure 55. Test Bench Reference

As seen in Figure 55, this drag-and-drop procedure has automatically created the ports that represent the inputs and outputs of the MultiBodyEngineSim model. Repeat this process for the TorqueComputation and DriveTrainSim_w_Params test benches. Upon completing this, the user's main editing window should look like that in Figure 56.

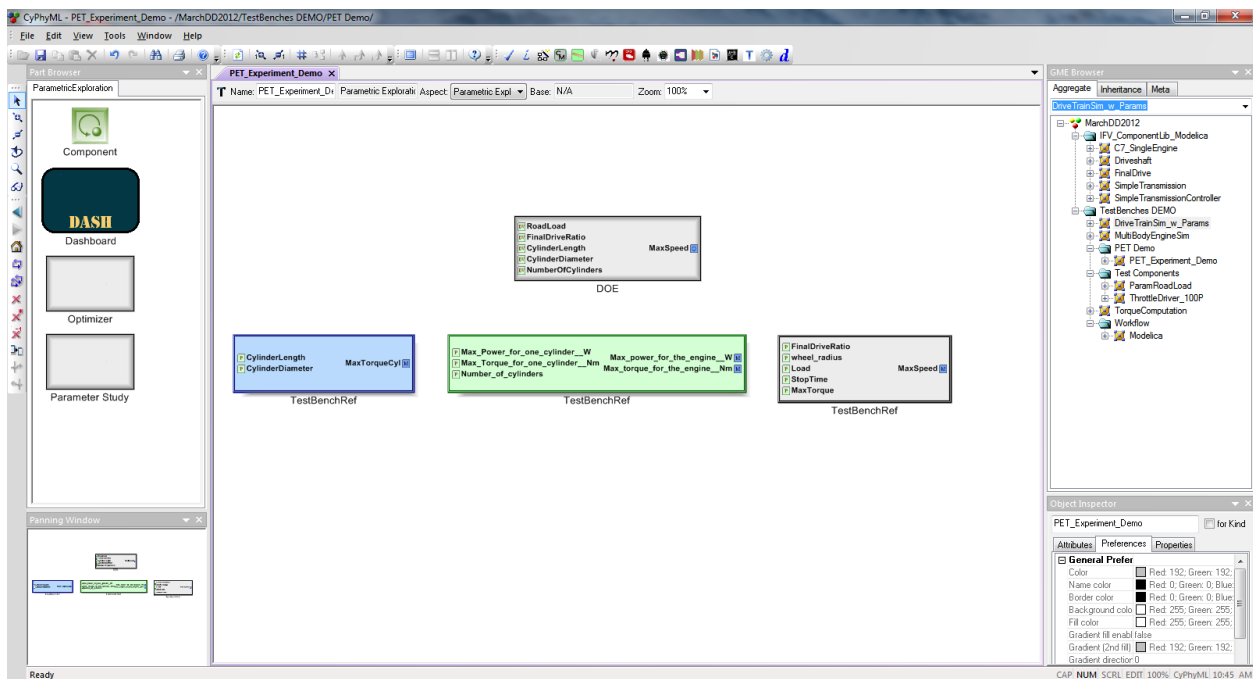


Figure 56. Complete PET Demo with No Connections

The user must now connect the design variable from the DOE component to the appropriate variable ports within each of the test bench reference components. Note: not all parameter ports will have a

design variable connected to it (this means that the parameter will simply remain constant throughout the DOE and equal to the value established inside the component model). Please consult Figure 57 as a reference to this connection task.

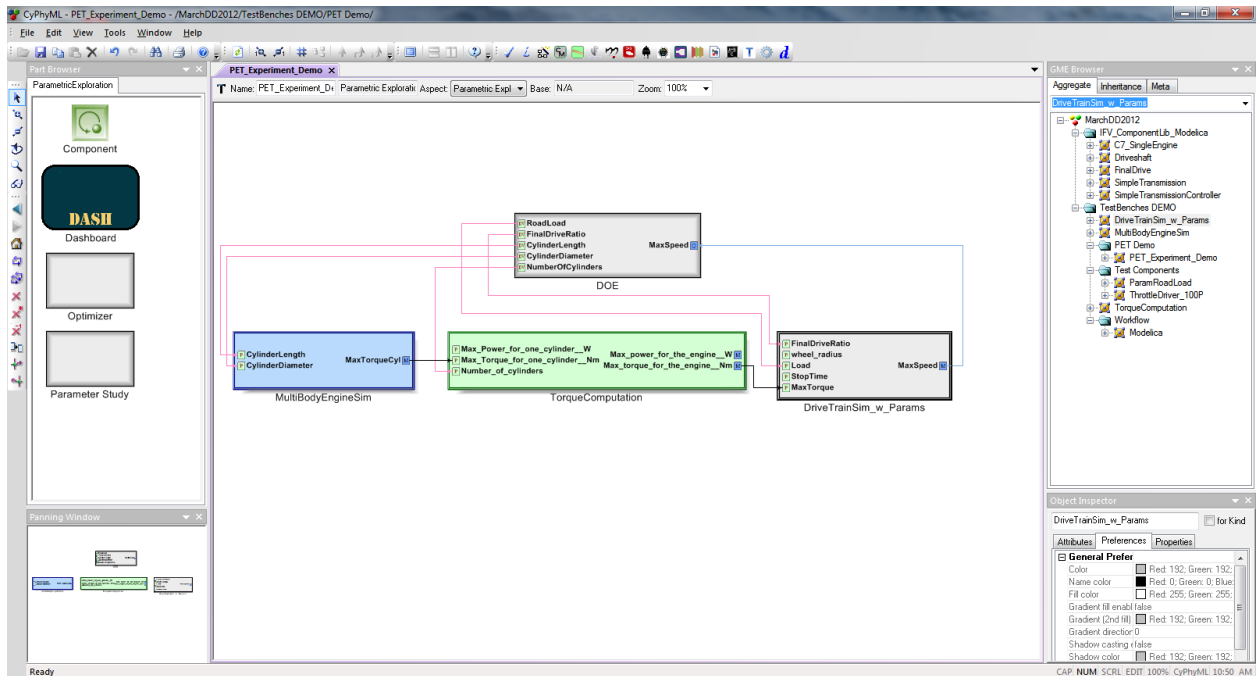


Figure 57. Complete PET Demo

Note that we have also connected the output metrics of each test bench reference component to the appropriate inputs in subsequent test bench reference components. This completes the development of the PET Demo component.