Tutorials and Examples of Software Integration Techniques for Aircraft Design using ModelCenter™

By

Mark Bigley, Candy Nelson, Peter Ryan

and W.H. Mason

MAD 99-06-02

June 1999

Supported by Virginia's Center for Innovative Technology
under grant INF-98-009
and
Phoenix Integration, Inc.
1750 Kraft Drive, Suite 2200
Blacksburg, VA 24060
540-961-7215

An electronic version of this report is available at
http://www.aoe.vt.edu/aoe/faculty/Mason_f/MRNR99.html

Multidisciplinary Analysis and Design Center for Advanced Vehicles
Department of Aerospace and Ocean Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0203

Executive Summary

In the summer and fall of 1998, two Virginia Tech undergraduate students and one graduate student developed tutorials to illustrate the use of the new Phoenix Integration software product ModelCenter™. ModelCenter™ provides users with a way to integrate existing codes, originally conceived to be operated in a “stand alone” manner, into a system where the codes can communicate with each other directly. This process is generally termed “wrapping”. ModelCenter™ is such a unique product that it requires examples to illustrate its capabilities to potential customers.

The students worked at Phoenix Integration Inc., which is located at the Corporate Research Center of Virginia Tech in Blacksburg, Virginia. This allowed the students to work closely with the developers of ModelCenter™. The resulting tutorials have been used to illustrate the product to new users and potential customers. Feedback from Phoenix has been positive. The outcome demonstrates the value of Virginia’s CIT in helping small companies compete in the marketplace. Finally, the students got valuable experience, seeing the requirements for successful product development.

Table of Contents

Chapter 1. Introduction

Phoenix Integration and Virginia Tech conducted a six-month effort using ModelCenter™, a newly developed engineering software package, in the summer and fall of 1998. Virginia's CIT supported a majority of the effort. ModelCenter™ is a systems integration and design tool for engineering, science and other technical applications that rely heavily on computer modeling. In this project software developers at Phoenix Integration collaborated with faculty and students to develop a series of test and sample airplane design problems and a Web-based tutorial for ModelCenter™. Students developed a number of airplane designs using the ModelCenter™ architecture, connecting various existing codes, including portions of ACSYNT (Ref. 1), to demonstrate the effectiveness of the tool for product development.

ModelCenter™ provides users with a way to integrate existing codes, originally conceived to be operated in a "stand alone" manner, into a system where the codes can communicate with each other directly. This process is generally termed "wrapping". It is one aspect of ModelCenter™. ModelCenter™ is such a unique product that it requires examples to illustrate its capabilities to potential customers. More details on ModelCenter™ are presented in Chapter 2.

The students had not had any prior exposure to the products, and the work provided the developers with an immediate source of feedback valuable to the final product, both in terms of any problems with the package and with the interface design. The students gained valuable experience with product development and commercialization.

This report presents paper-based documentation of the effort. However, as with many innovative software products, it is best understood working interactively directly with ModelCenter™. A demonstration version of the software is available at the Phoenix Integration website: http://www.phoenix-int.com/products/ModelCenter.html.

Chapter 2. ModelCenter™

ModelCenter™ is an innovative new software product from Phoenix Integration. The description in this section is taken from Phoenix Integration's website.* ModelCenter™ provides design and manufacturing engineers with the ability to perform distributed modeling and analysis. It uses a unique integration architecture to wrap and integrate legacy programs, data, and geometry features. Using Phoenix Integration's Analysis Server™, designers can access multiple design programs, databases, and CAD APIs from remote computers. ModelCenter™ and the Analysis Server™ provide a client/server environment. Components can be created and distributed using the Analysis Server and then interfaced using ModelCenter™. ModelCenter™ provides tools such as optimization and probabilistic drivers and is easily integrated with standard object protocols such as Microsoft COM

ModelCenter's Features

Screen Features:	ModelCenter™ has four distinct perspectives from which to view the model.
Visual Integration:	ModelCenter's power is in visually linking data.
Geometric Modeling:	Link analysis tools to geometric parameters and watch the changes take effect.
Parametric Trade Studies:	Measure the performance of the model with performance analyses tools.
Integration with Microsoft Excel:	Link Excel spreadsheets with component-based analysis.

Using ModelCenter™

ModelCenter™ provides the design engineer with the ability to access remote design applications and bring together multiple analysis tools to form an integrated design environment. Engineers using ModelCenter™ have been able to wrap and provide consistent interfaces to legacy applications such as FORTRAN codes and CAD applications.

* http://www.phoenix-int.com/products/ModelCenter.html.

ModelCenter™ can save weeks worth of work by coordinating multiple engineering programs to work together. In addition, costly rework is avoided by using coordinated models that have built-in links to previous design data.

Illustrations of how ModelCenter™ can be used are presented in the next four chapters, which provide a general tutorial and three examples.

Chapter 3. ModelCenter™ Tutorial*

This tutorial shows you how to use ModelCenter™ to link together two components to create a model. A component is an external analysis or design program that is controlled by ModelCenter. The tutorial also describes the ModelCenter window and the workspace concept.

After completing this tutorial, you will be able to:

1. Create and save a model.
2. Connect to an Analysis Server™ and instantiate components to be linked.
3. Modify the value of variables in components.
4. Run the model.
5. Link components.

Getting Started

After you have installed ModelCenter and have setup the Analysis Server™, you are ready to use ModelCenter.

Double click the ModelCenter icon. The following screen will appear, as shown in Fig. 1:

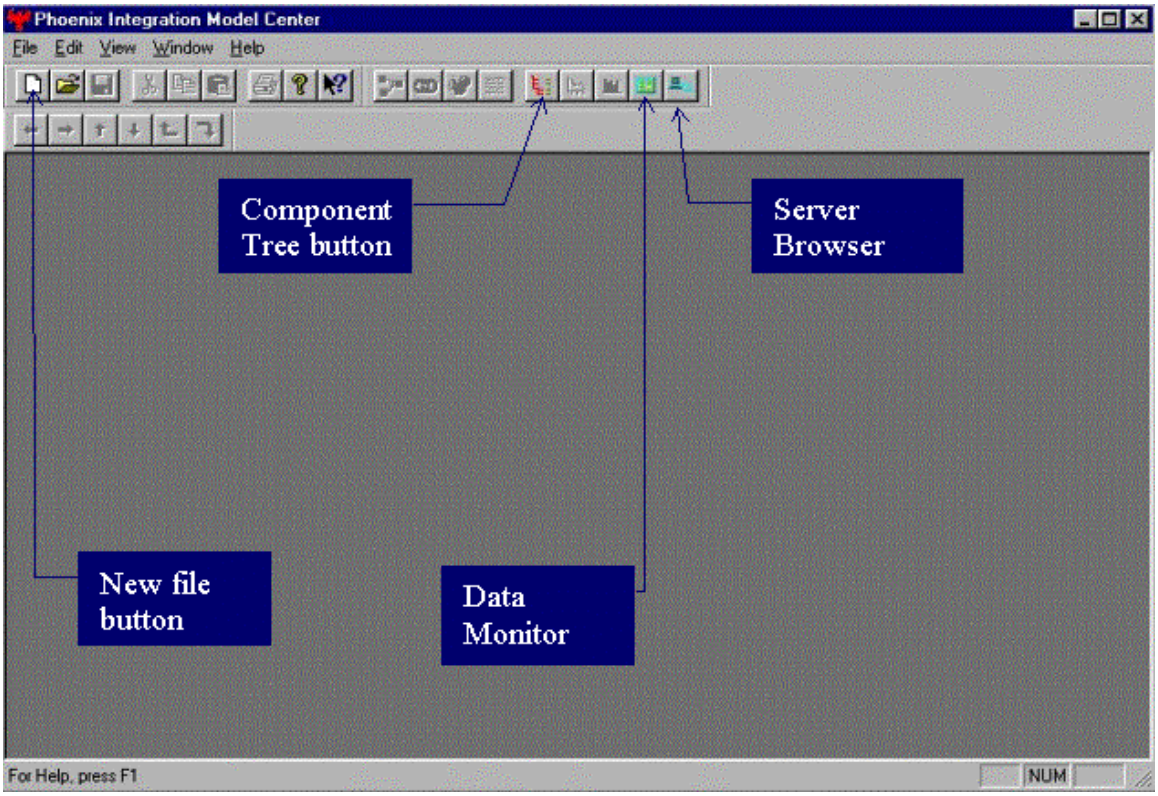

Figure 1. The ModelCenter ™ main window

* This chapter was primarily the contribution of Phoenix Integration staff.

On the toolbar, click the **New File**, **Component Tree**, **Data Monitor**, and **Server Browser** buttons.

The **Configuration Window**, **Component Tree**, **Data Monitor**, and **Server Browser** will appear in the window. The next section shows and describes the ModelCenter window.

Note: When you open ModelCenter, the main window will be configured the way it was configured the last time you exited ModelCenter.

ModelCenter Window

Figure 2 shows the ModelCenter window after the windows described in the previous section have been opened.

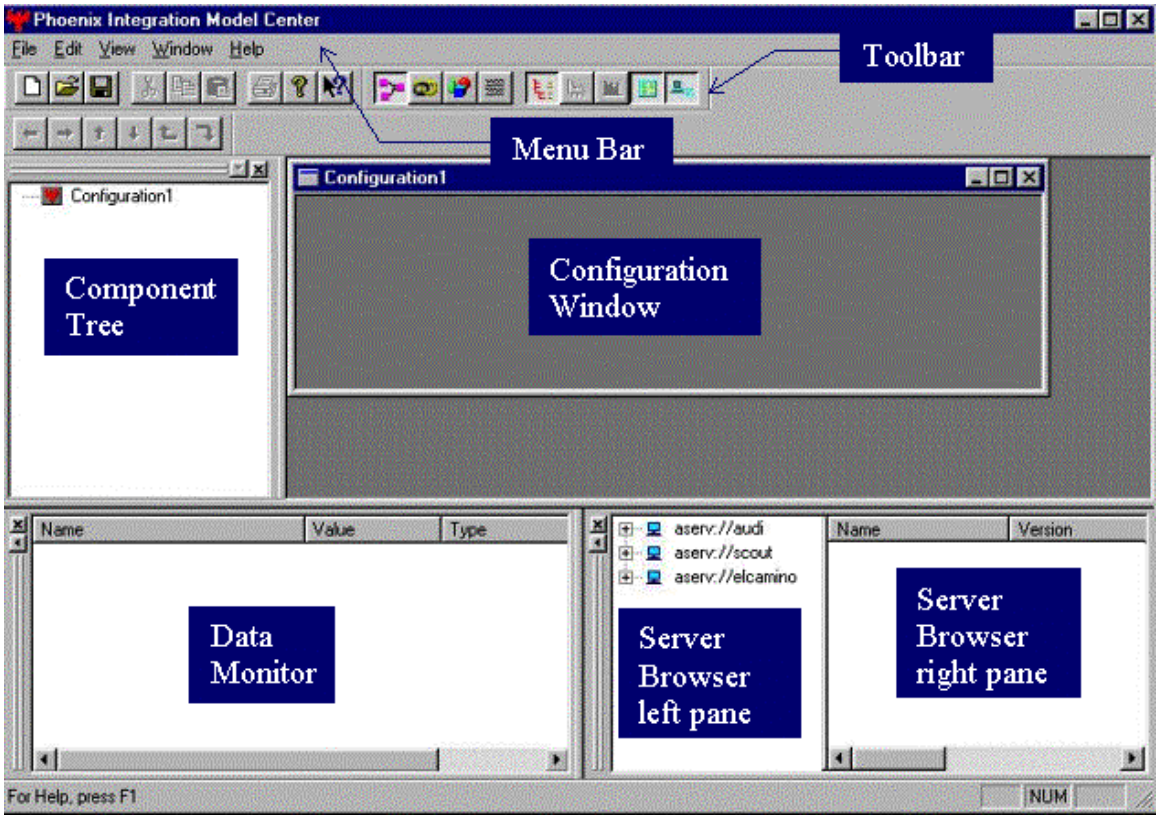

Figure 2. The ModelCenter window with the Component Tree, Data Monitor, Server Browser, and Configuration Window open

The main window has six components: **Component Tree, Configuration Window, Data Monitor**, **Menu Bar**, **Server Browser**, and **Toolbar**.

- The **Configuration Window** is the main work area of ModelCenter; it allows you to view components in three ways. The **Assembly View** allows you to view the model schematically. The **Geometry View** allows you to view the geometric aspects of the model. The **Link Editor View** allows you to view the data-sharing relationships inside the model.

Note: The **Assembly View**, **Geometry View**, and **Link Editor View** are described in greater detail in The Workspace Concept.

- **The Component Tree**, when expanded, lists all data items of the components that are in the current model. These items are organized hierarchically.
- In the **Data Monitor**, you can change the values of selected data. This space identifies the name, value, type, and units of the data item.
- The **Menu bar** is a standard menu bar that helps you use ModelCenter.
- The **Server Browser** is a network-based browser that helps you locate servers and design objects; it allows you to search through directories and entire networks. In the left pane of the Server Browser, you choose which analysis server to use. In the right pane, you choose the components to configure.
- The **Toolbar** has buttons that allow you to toggle between views of the current configuration window and to toggle the **Data Monitor**, **Server Browser**, and **Component Tree**.

Note: You can undock any of the windows by right clicking on the window and deselecting **Allow Docking**. Then, drag the window to a new position. **Figure 3** shows the Data Monitor undocked.

To dock a window, right click on the window and select **Allow Docking**. Then, drag the window to the position you want.

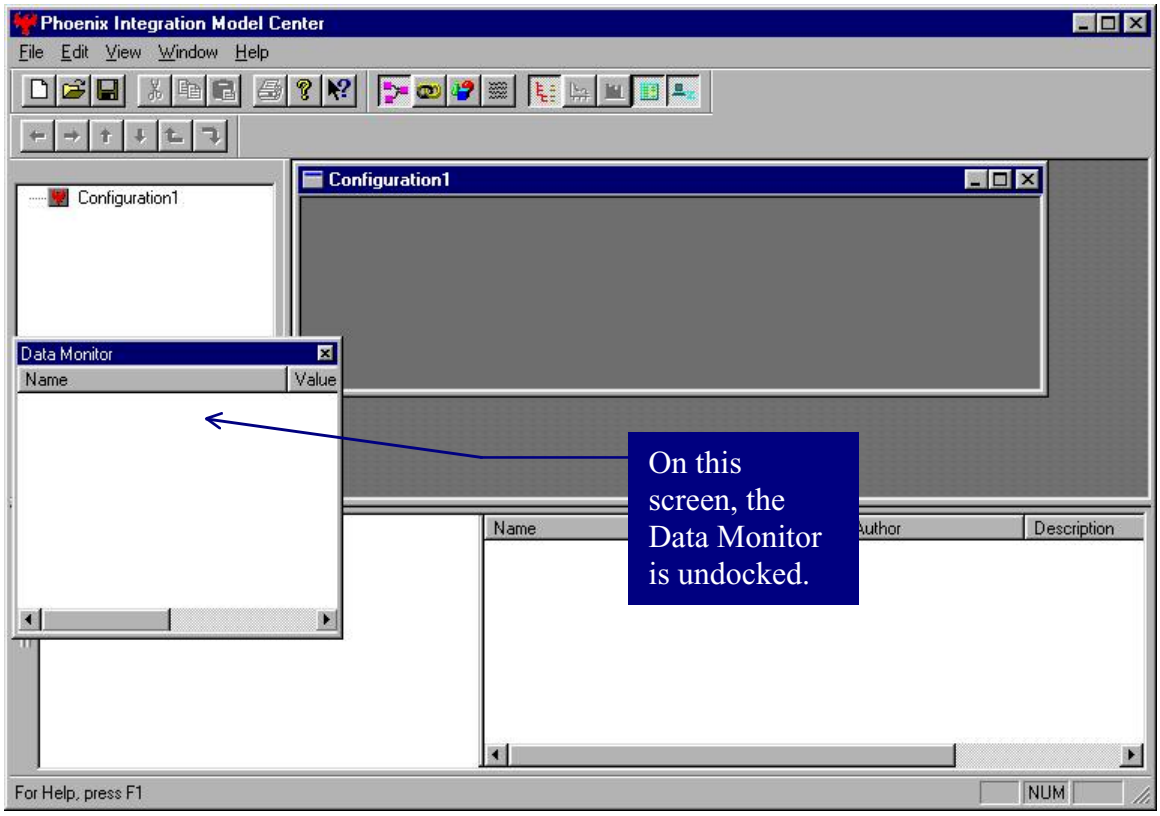

Figure 3. The ModelCenter window with the Data Monitor undocked

The Workspace Concept

ModelCenter™ uses a workspace concept that provides different ways of considering an engineering model. Each ModelCenter™ configuration window holds a single hierarchical model. You may have several configurations on the screen at the same time. Figure 4 shows ModelCenter™ with three open configurations; each configuration has a different view.

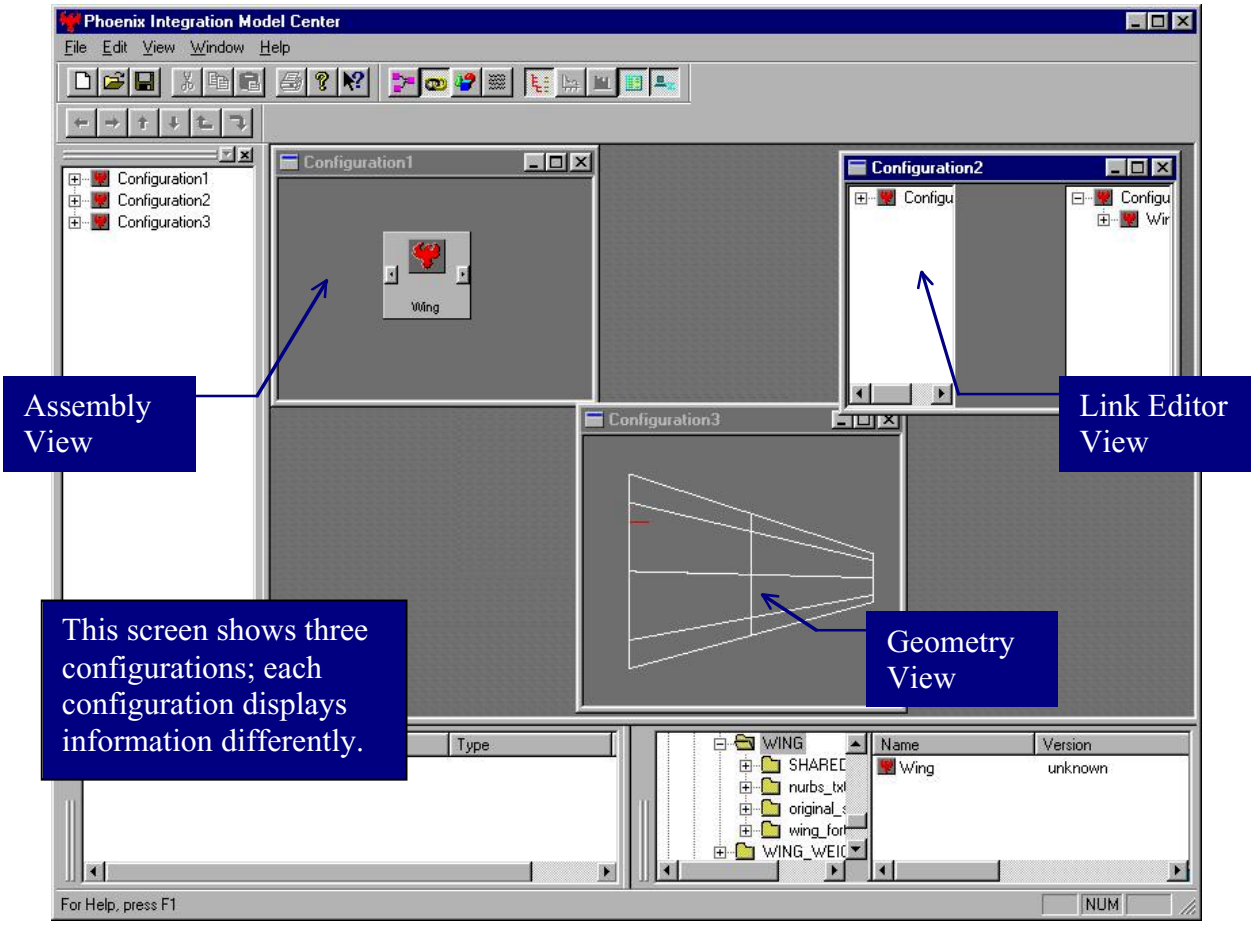

Figure 4. The ModelCenter window with three open configuration windows

The **Assembly View**, a schematic view of the model, shows all components as icons. Links between components are shown as arrow connectors between the component icons. This view allows you to hierarchically construct a model. Figure 5 shows an Assembly View window that contains two components.

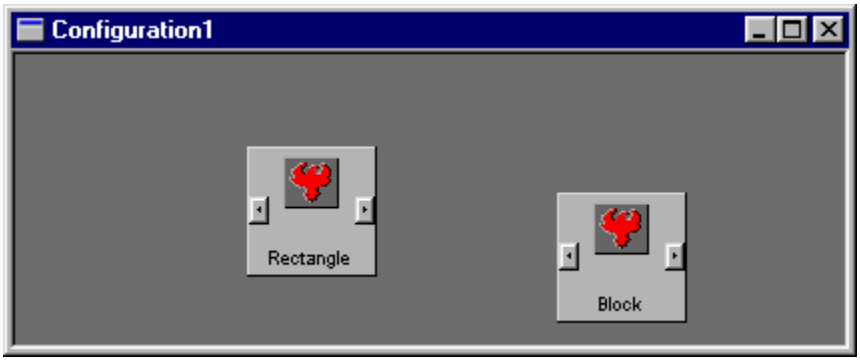

Figure 5. An Assembly View window that contains two components

The **Geometry View** provides a three-dimensional representation of geometric components. Non-geometric components will not appear in the Geometry View. Figure 6 shows a geometric rendering of a component.

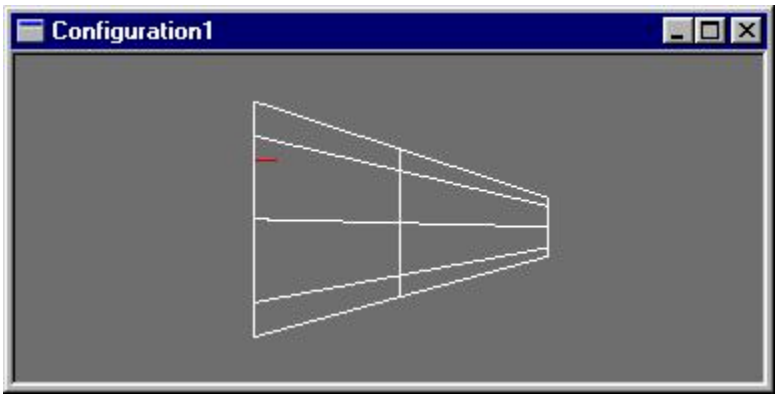

Figure 6. A Geometry View that displays the physical properties of a component

The **Link Editor** displays each component as a hierarchical tree of data. You can create links between parameters from several components by dragging a line from the desired parameters on the left side of the workspace to one on the right side of the workspace. In this view, certain links are not allowed because they may change existing relationships between data. Figure **7** shows a model with two linked components.

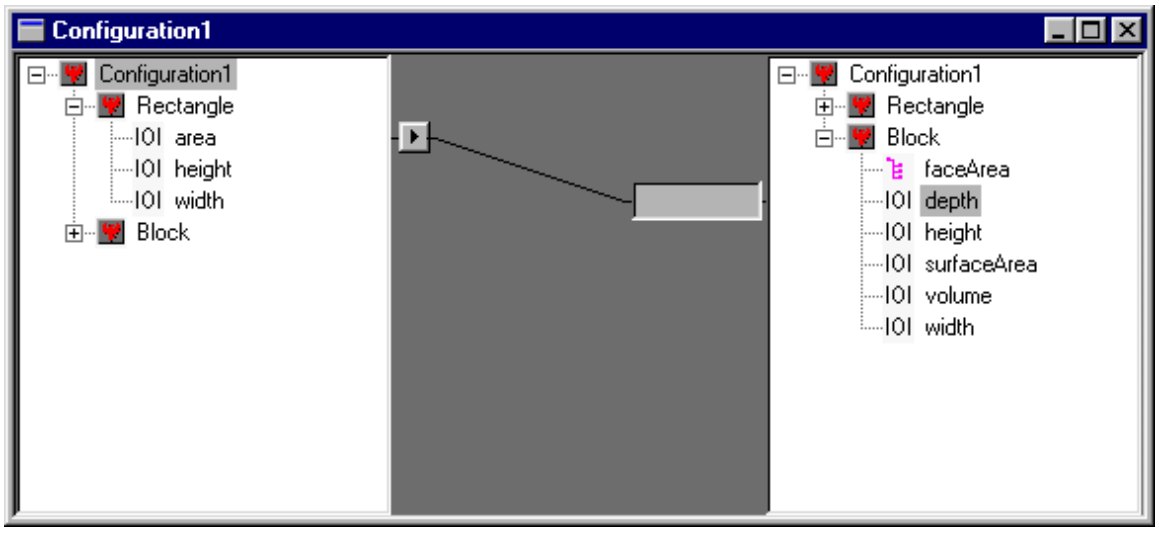

Figure 7. The Link Editor displaying two linked components

You can use the configuration windows to view your data in three different ways. For example, if you were working on a model that contained components of an aircraft, you could display the physical representation of the wing while linking the wing to the wing weight in the Link Editor view.

Working with Components

Once you are familiar with the main window and the workspace concept, you are ready to work with and link components. In this section, you will

1. Select the Analysis Server™ and select the components you are going to link.
2. Modify the values of the data items of the components.
3. Link the components so that the value of one component is dependent upon the value of the other component.
4. Update the values of the components after they have been linked.

Step1: Selecting the components

In this step, you will select the Analysis Server™ and the components with which to work.

1. Select **File: New** to open a new configuration if you haven't already opened a configuration window.
2. Select the Analysis Server™ that contains the tutorial from the left pane of the Server Browser. The components that you will be using, **Block** and **Rectangle**, will appear in the right pane.

   Your screen should look similar to the one below in Fig. 8.

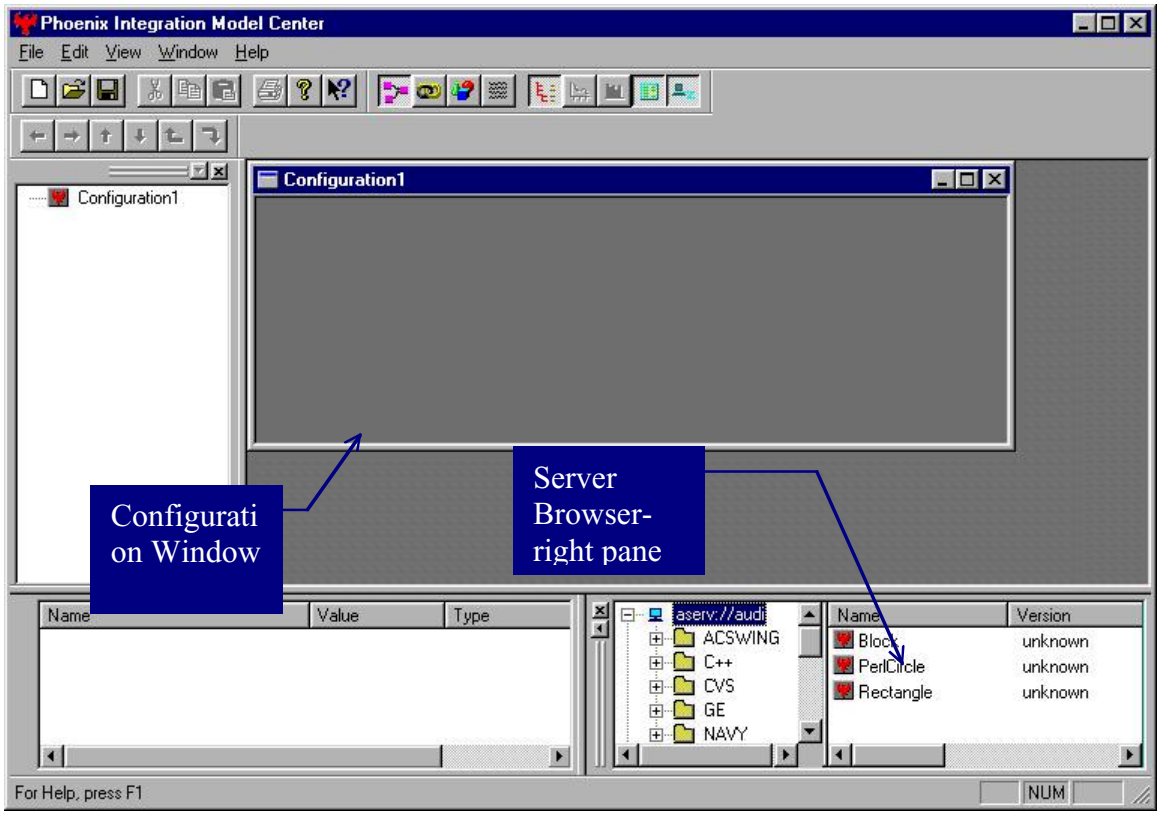

Figure 8. The ModelCenter window displayed with one open configuration window and components in the right pane of the Server Browser

3. Select **Rectangle** and drag it to the **Configuration Window**. You will be asked to specify the component name. For this example, accept the dialog box defaults as shown in Fig. 9, and click **OK**.

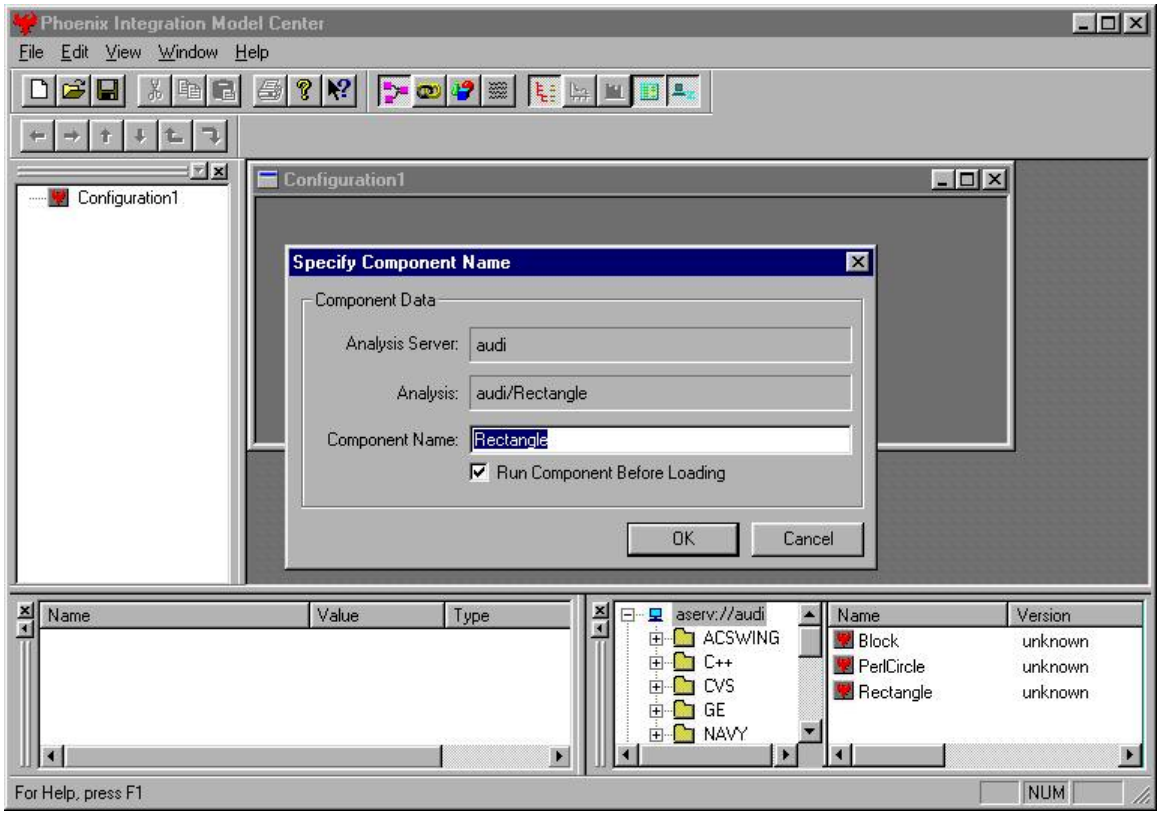

Figure 9. When you drag components into the configuration window, you must specify a component name

4. Next, select **Block** from the **Server Browser** and drag it to the **Configuration Window**. When prompted, accept the dialog box defaults and press **OK**. Your screen should look similar to the one below in Fig. 10.

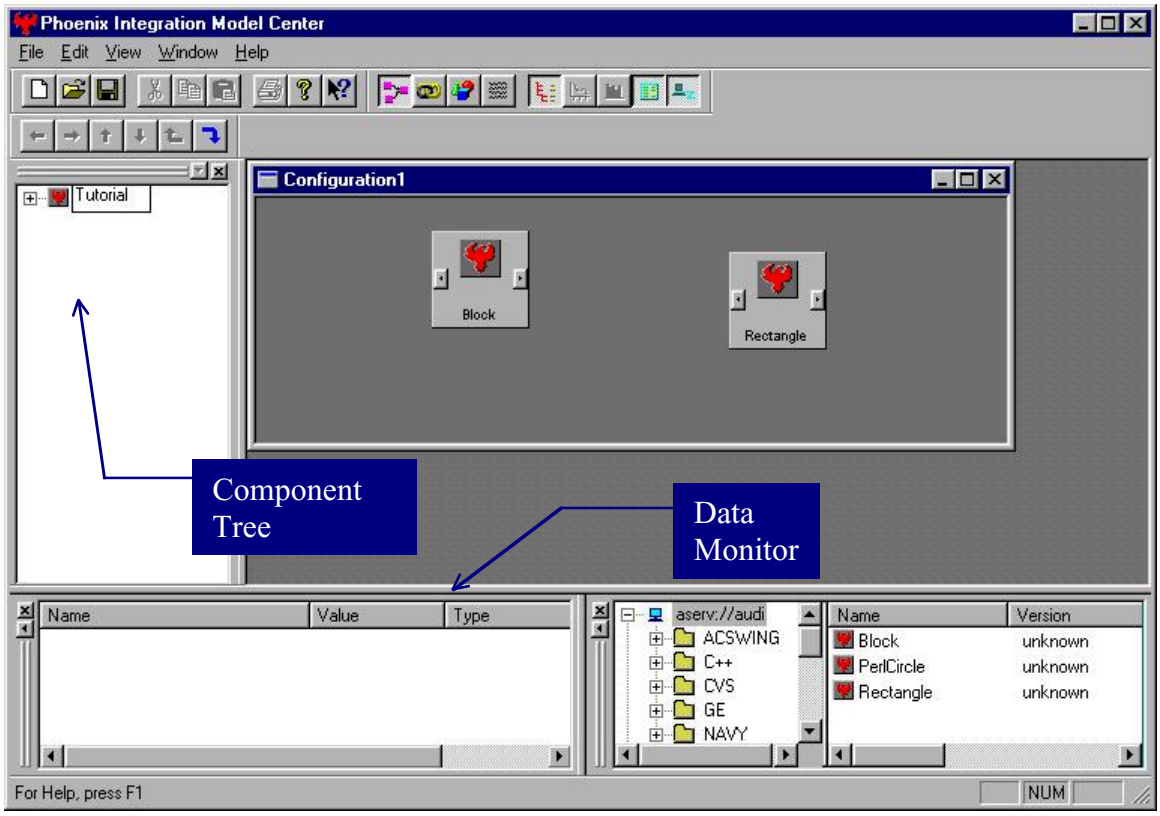

Figure 10. ModelCenter with two components in the configuration window

5. In the **Component Tree**, select Configuration 1 and press F2; this will allow you to rename the configuration. Name it Tutorial.
6. Select **Tutorial** and drag it to the Data Monitor.
7. In the **Data Monitor**, select **Tutorial**; it will expand, as shown below in Fig. 11.

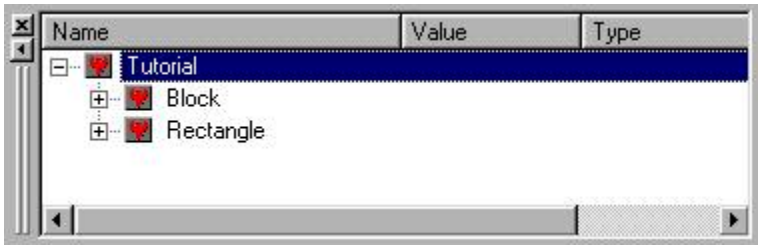

Figure 11. The Data Monitor with Configuration 1 expanded

Your screen should look similar to Fig. 12.

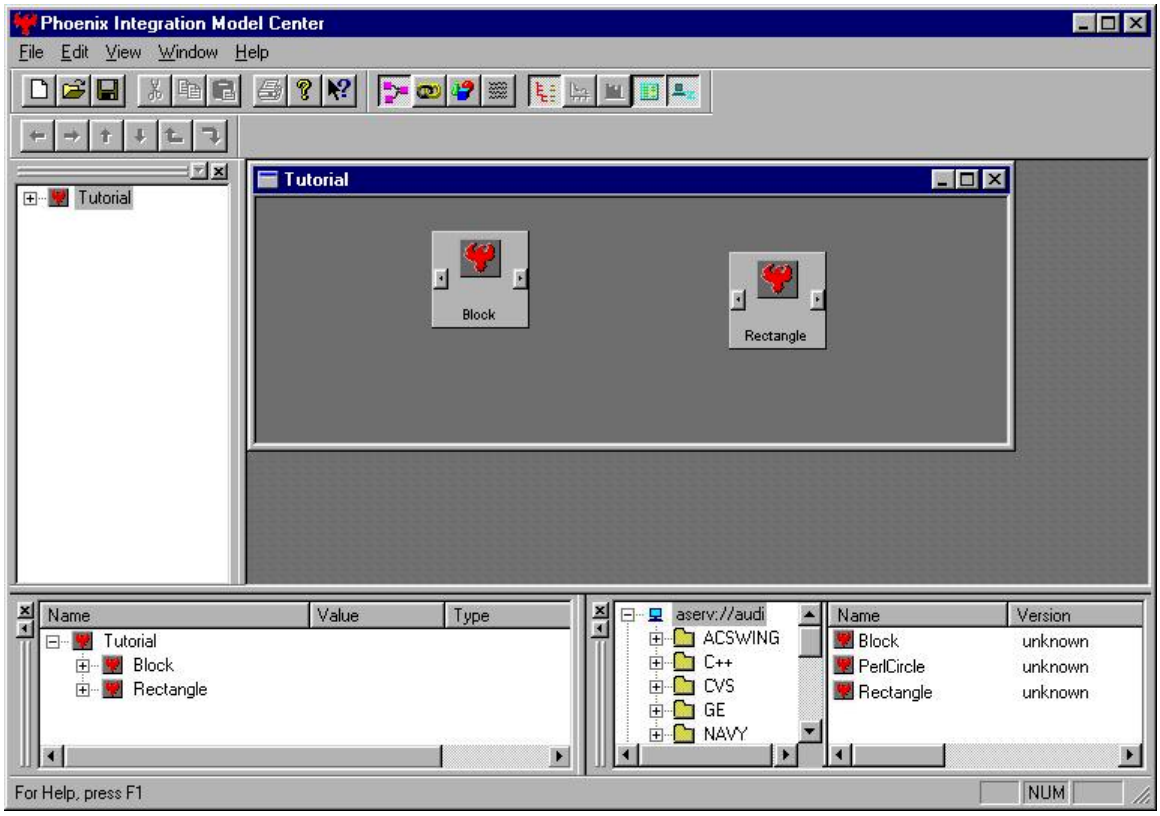

Figure 12. The ModelCenter window after the components have been selected

Step2: Modifying the Initial Settings

After you have completed Step 1, you are ready to set the values for **Rectangle** and **Block**.

To modify the values of Rectangle:

1. In the Data Monitor, click on **Rectangle** to expand it.

   In the Data Monitor, each data item has a small rectangle to the left of the name of the variable. Input variables have a green arrow to the left of the rectangle; output variables have a blue arrow to the right of the rectangle. **Area**, because it is calculated by multiplying **height** and **width** (input values), is an output value, as shown in Fig. 13, below.

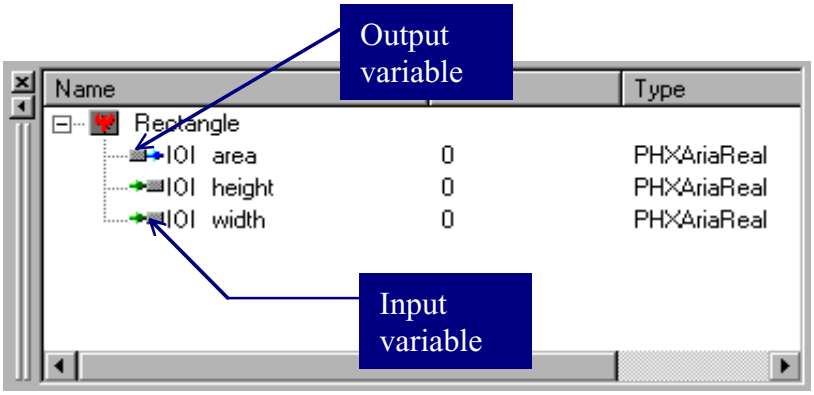

Figure 13. The Data Monitor with the component expanded

2. Select **height** from the Data Monitor and click on the number in the **Value** field. This will allow you to modify the value for height. Enter **2** as shown in Fig. 14.

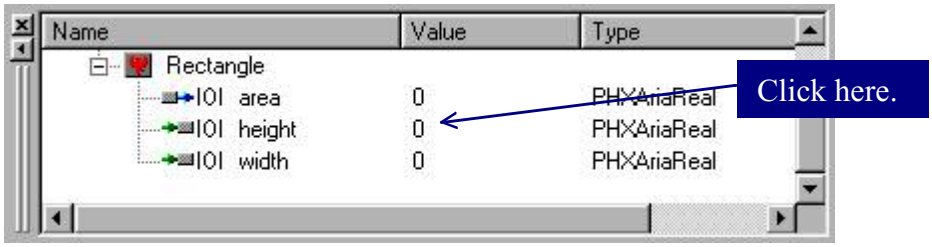

Figure 14. The Data Monitor with the component expanded and ready for modification

3. Next, select **width** and click on the number in the **Value** field. Enter **4**. The Data Monitor should look similar to the one below in Fig. 15.

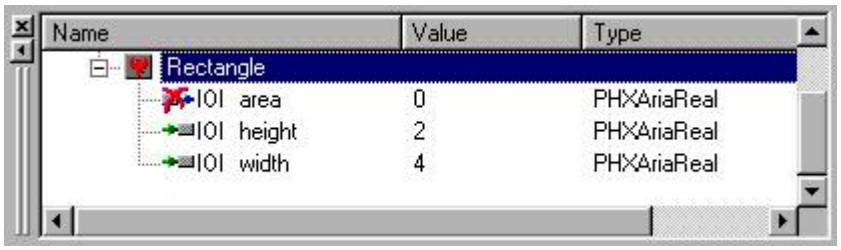

Figure 15. The Data Monitor with the output variable ready to be updated

Note: After an input value is modified, a red X will appear beside the output values that are dependent upon that input value. The red X will disappear when the output values are updated.

4. Click on the red X to the left of **area**. The Analysis Server will run the Rectangle component; after the component has been run the value of area will be updated and the red X will disappear. The value of area is now 8.

Steps 3 and 4 above demonstrate accessing a simple, remote program that is wrapped on a server.

To configure Block:

1. In the **Data Monitor**, click on **Block** to expand it. Your screen should look similar to the one below in Fig. 16.

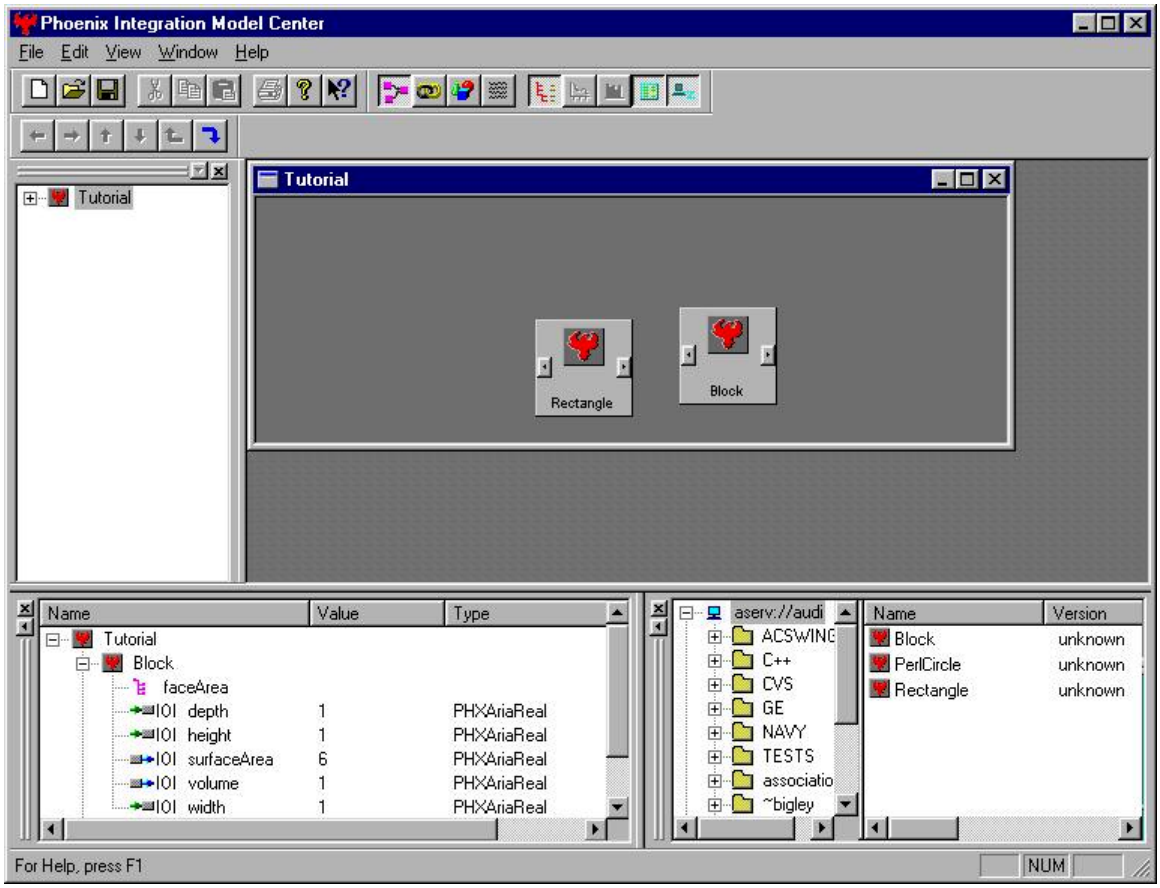

Figure 16. The ModelCenter window with a component expanded in the Data Monitor

2. Select **depth** from the list of the data items under **Block**. Click on the number in the **Value** field; this will allow you to modify the value of depth. Enter **4**.
3. Next, select **height** and click on the number in the **Value** field. Enter **3**.

Note: Make sure you are modifying the **height** and **width** of **Block** in steps 2 and 3, not the **height** and **width** of **Rectangle**.

4. Select **width** and click on the number in the **Value** field. Enter **5** for the value of **width**.

   The **Data Monitor** should look similar to the one below in Fig. 17.

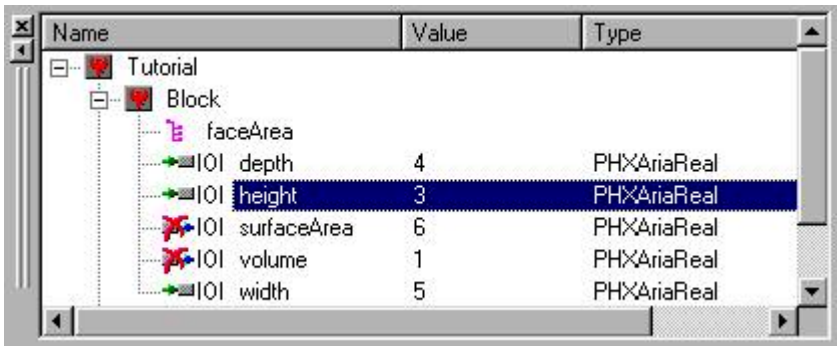

Figure 17. The Data Monitor with the output values of the component ready to be updated

Step 3: Linking the components

In the next part of the tutorial, you will link the two components so they can share data. After linking, the values of one component will feed into the other component. The codes will be chained together. In this example, you will link the **area** of **Rectangle** to the **depth** of **Block**; therefore, after the components are linked, the **depth** of Block will change only when the **area** of Rectangle changes.

1. Choose the Link Editor Button which will bring you to the Link Editor view.

   Your screen should look similar to the one below in Fig. 18.

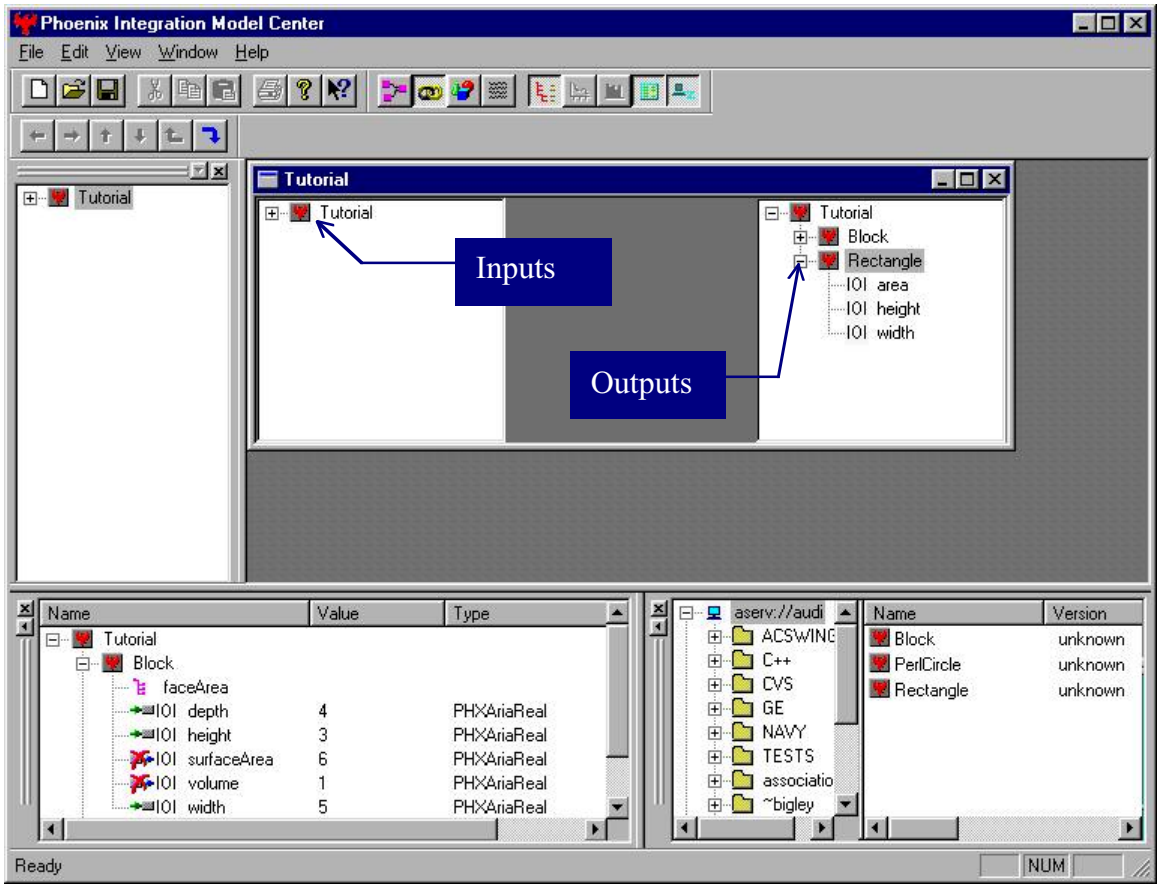

Figure 18. ModelCenter with the Link Editor open

The Link Editor has a left pane and a right pane. Each pane contains the configurations and components of the model. In the Link Editor, links flow from the left pane to the right pane. After linking, the values of variables in the right pane will only change when the variable in the left pane that it is linked to changes.

3. In the left pane of the Link Editor, click on **Tutorial** to expand it. Then, click on **Rectangle** to expand it.
4. In the right pane of the Link Editor, click on **Tutorial** to expand it. Then, click on **Block** to expand it.

   Your screen should look similar to the one below in Fig. 19.

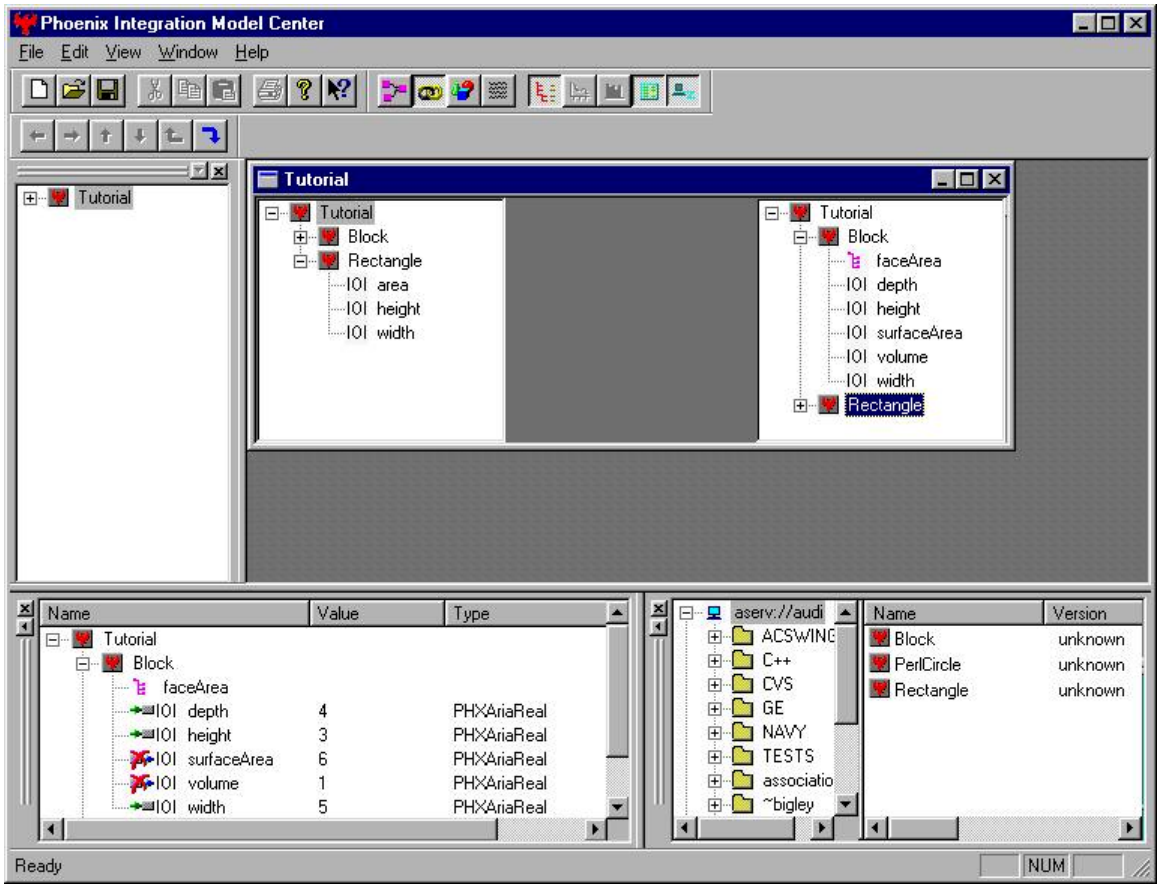

Figure 19. The Link Editor with the components expanded

5. In the left pane of the **Link Editor**, select **height** (under Rectangle) and drag it to **depth** (under Block) in the right pane of the Link Editor. Figure 20 indicates the variables to link.

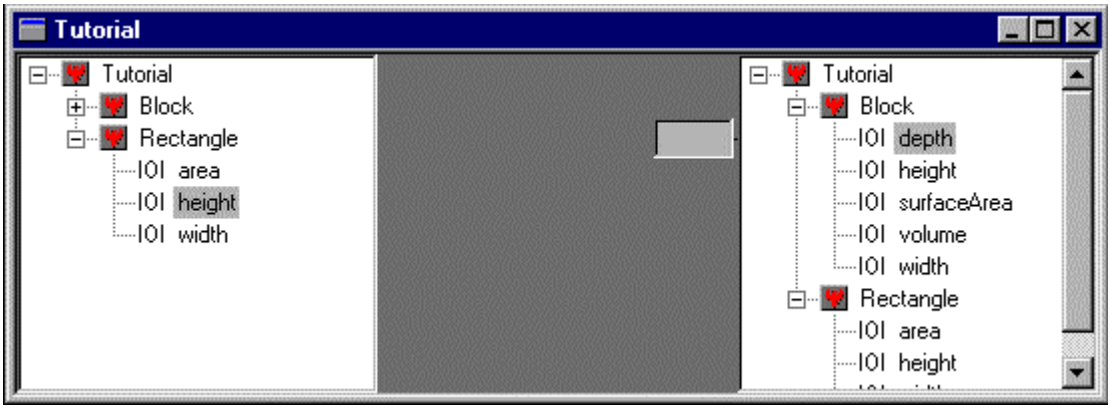

Figure 20. The Link Editor displayed with the components ready to be linked

Note: Height will not actually move to the right pane; it will only connect to **depth**.

Your screen should look similar to Fig. 21.

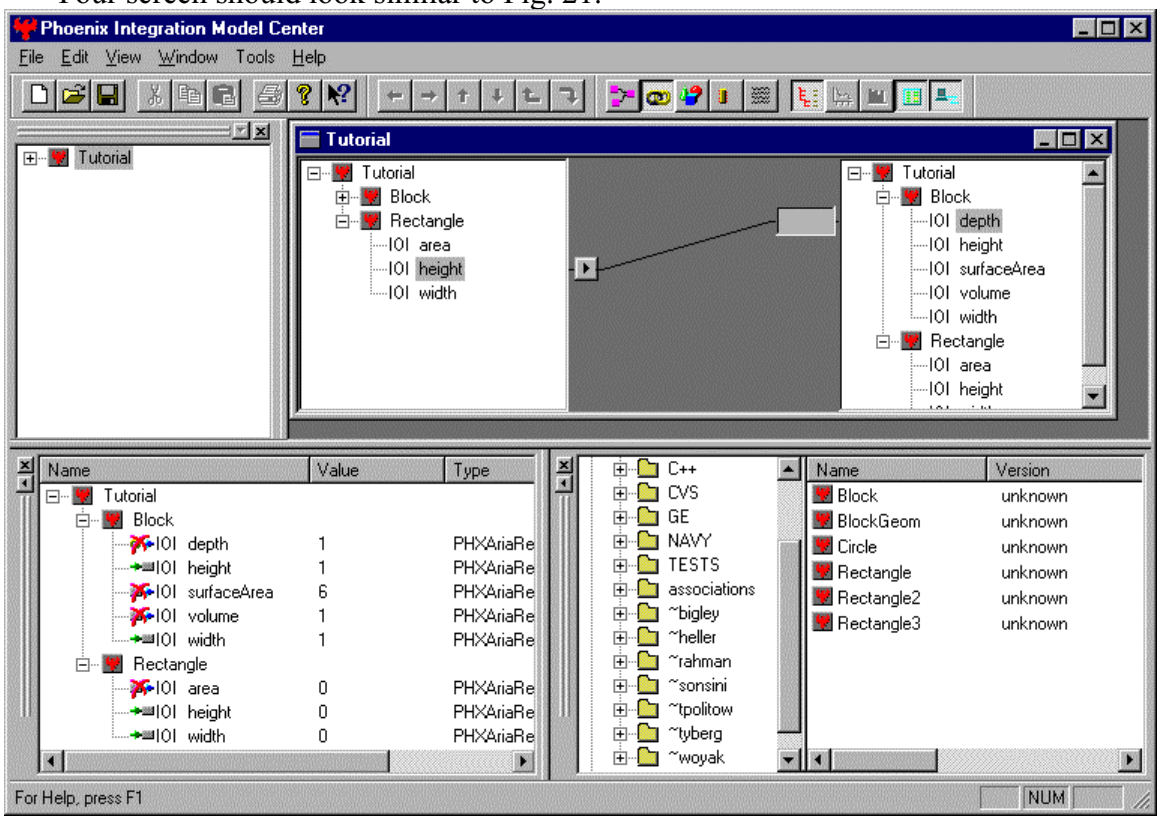

Figure 21. Two components linked in ModelCenter

Block and **Rectangle** are now linked. Because these components are linked, the value of **Block**'s depth will be set equal to the value of the **Rectangle**'s height.

In the **Data Monitor**, all output values (**depth**, **surfaceArea**, and **volume**) for Block will now have red Xs beside them.

After an output variable is linked to another variable its icon in the Data Monitor will change from a small rectangle to a small circle. This icon change helps you quickly identify which output variables are linked.

6. Switch to the **Assembly View**; this view will show the components' icons as linked. The view is shown in Fig. 22.

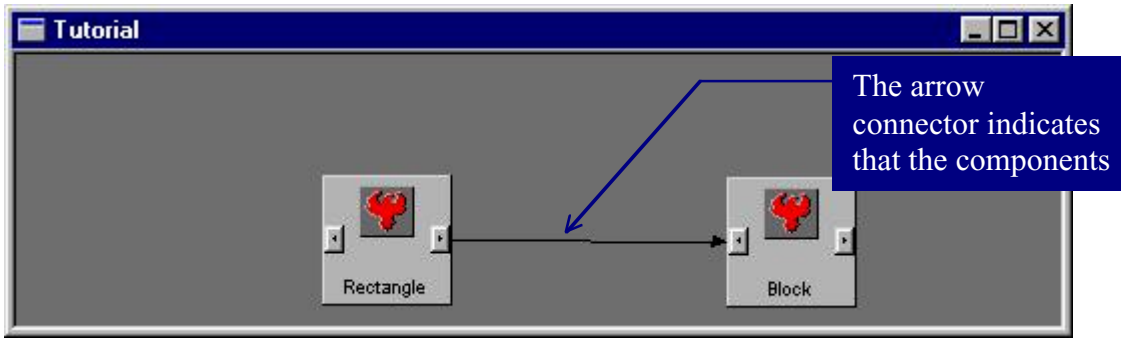

Figure 22. The Assembly View displaying two linked components

The arrow is pointing to Block because Block is now dependent upon Rectangle for a value.

Step 4: Updating all output values

The final step in this tutorial is updating all output values of Block and Rectangle and saving your work.

1. In the Data Monitor, click on the X beside **volume**. Clicking on this X will update the value of volume and will simultaneously update the value of **surfaceArea**. When you are in the Assembly View, the icon being updated will be highlighted until the analysis has finished running. In this tutorial, Block will be highlighted, but it will only briefly flash because the analysis is quickly finished. Some analysis may take an hour or longer to run.

   After updating, the value of **depth** will be 8 (the same as the value of Rectangle's area), the value of **surfaceArea** will be 158, and the value of **volume** will be 120, as shown in Fig. 23.

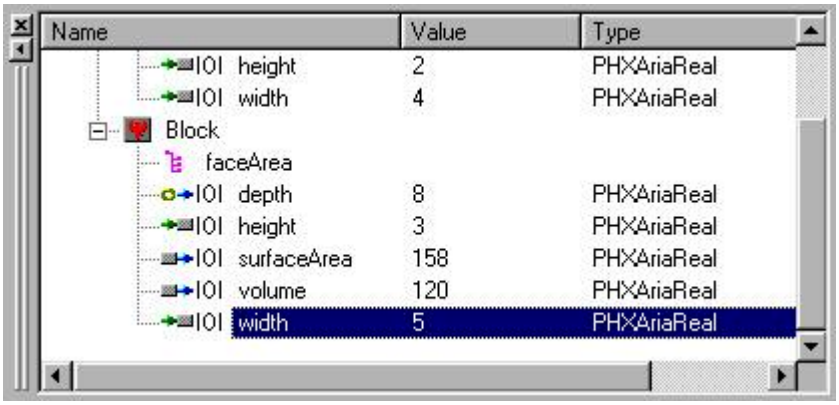

Figure 23. The Data Monitor with the values of the linked components updated

Because the components are linked, each time the value of the height changes, the value of depth will change. The value of the Block's depth is now dependent upon the value of the height of the Rectangle, so the value of depth will change each time height. For example, if you change the value of height from 2 to 6, the value of depth, after being updated, will change from 2 to 6. The values of **surfaceArea** (**Block**) and **volume** (**Block**) will also change because they are dependent upon the value of depth. ModelCenter automatically knows which component to run and where to run it. The screen below shows how the above values will change if the value of **height** is changed to 6. This is shown in Fig. 24.

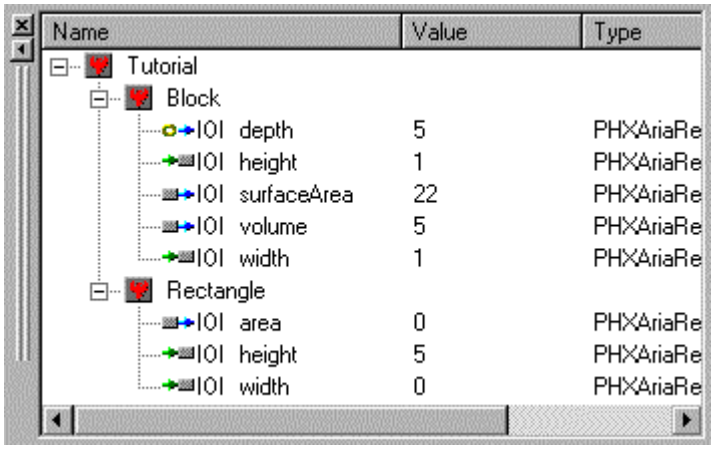

Figure 24. The Data Monitor with the value of height changed to 5

2. Choose **File: Save** to save your work. A standard dialog box will appear. Name your work and then click **Save**.

3. Choose **File: Exit** to exit ModelCenter.

To summarize, in this tutorial you have used ModelCenter™ to access and run remote, stand-alone programs. ModelCenter™ can make a model consisting of multiple programs behave as if the programs were a single, custom program by dynamically sharing data between them.

To get more practice linking components and to learn some advanced concepts, continue to Example 1 in the next chapter.

Chapter 4. Example 1-Building an Aircraft Model

This example covers advanced concepts that were introduced in Tutorial. For a brief overview of ModelCenter's capabilities, please see Tutorial.
After you finish this example, you will be able to:

1. Modify the geometric dimensions of components.
2. Change between the different views in ModelCenter.
3. Link Components
4. Change the orientation of components in the Geometry View

Before You Begin…

Before using this tutorial, you should perform The Tutorial and you should be familiar with the toolbars and windows of ModelCenter.

Getting Started

1. Double click on the ModelCenter icon to open the ModelCenter main window.
2. On the toolbar open a new configuration by clicking on the **New File** button. If they are not already open, open the **Data Monitor**, **Component Tree**, and **Server Browser**. After opening the above windows, the ModelCenter window should look like the one below in Fig. 25.

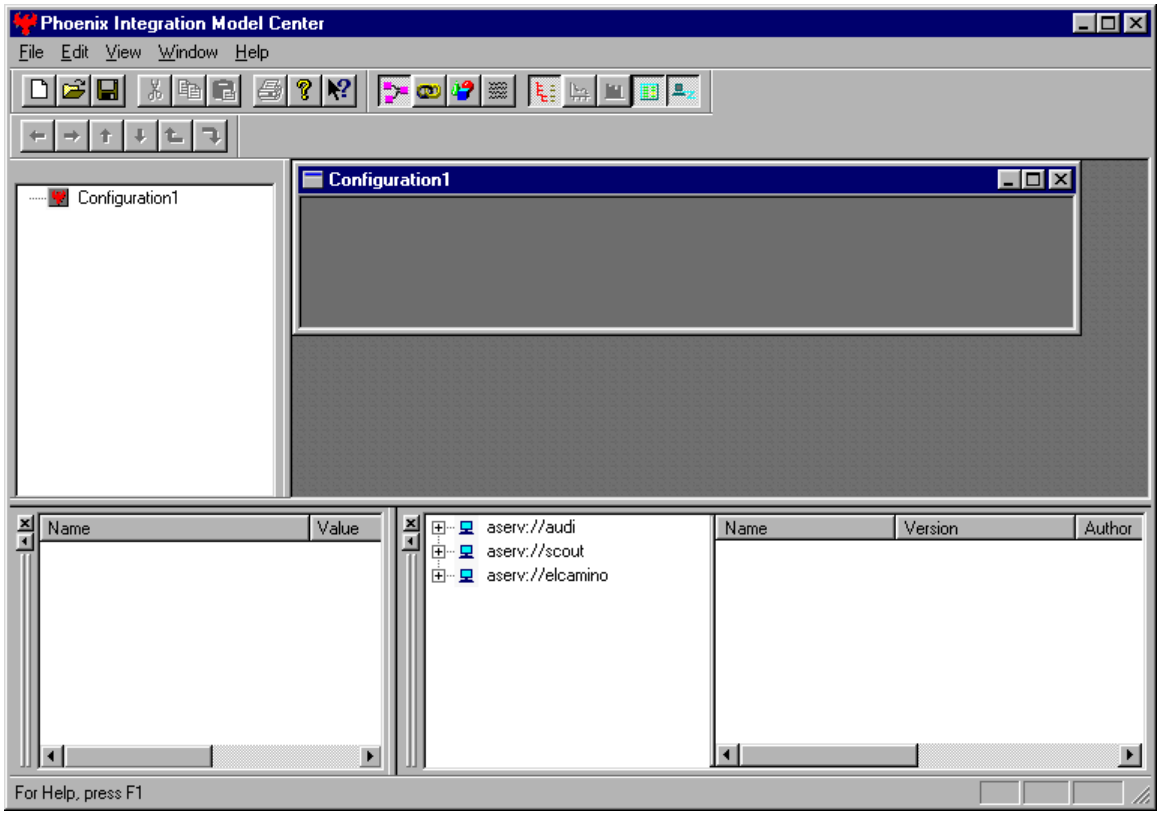

Figure 25. ModelCenter with Configuration window open.

3. If the configuration window is not in the **Assembly View** switch to it now.
4. In the Configuration Tree, long click on **Configuration 1**; this will allow you to rename the configuration. Change the name to Aircraft. This is shown in Fig. 26.

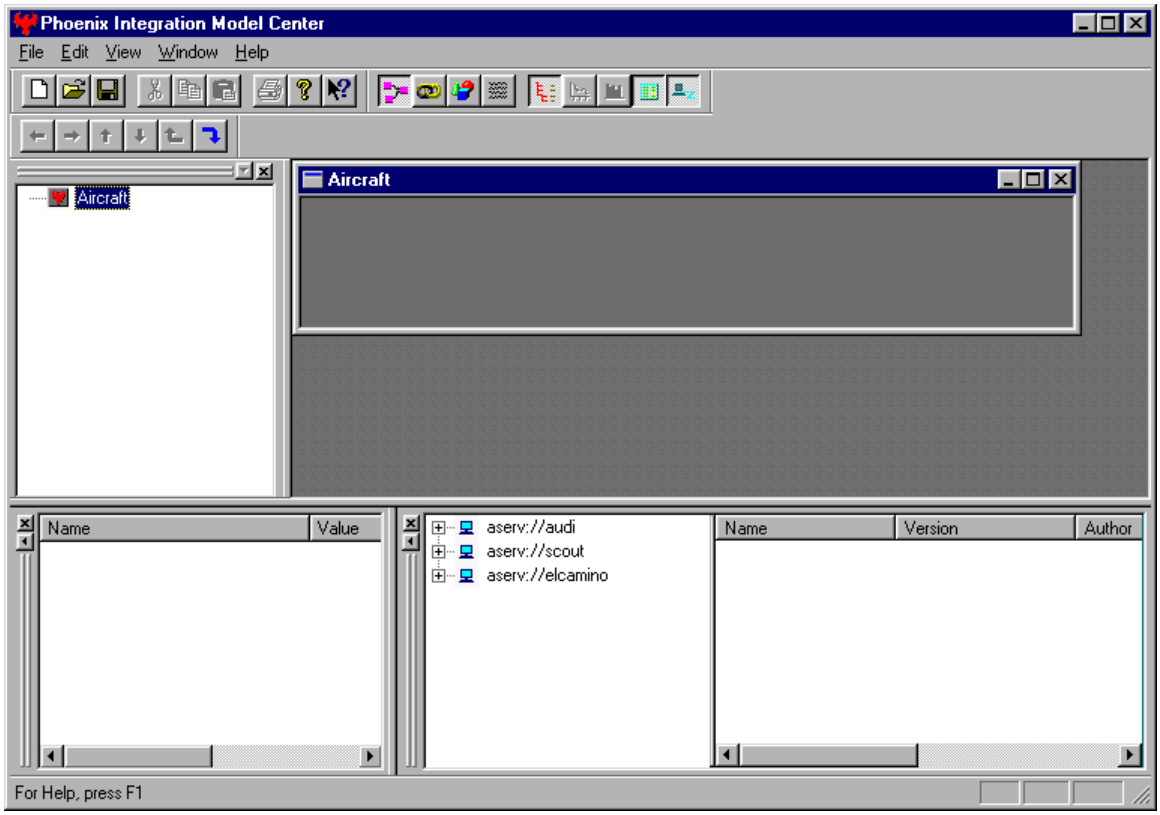

Figure 26. ModelCenter's Configuration window showing the new name entered.

Working with Components

Step 1: Selecting the components

1. From the Server browser, select your analysis server.

Note:	For more information on connecting to and selecting an Analysis Server, please see the Server Browser section in the Appendix.

2. Choose the folder titled **~tyberg**. [this folder may change based on the local tutorial installation]
3. From the **Server Browser**, choose **Wing** and drag it into the **Assembly View**. When prompted for a name, enter **Right_Wing** and click **Ok**.
4. Select another **Wing** from the Server Browser and drag it to the Assembly View. Name this wing **Left_Wing.**

5. Drag two more wings into the Assembly View. Name one of the wings **Right_Tail** and name the other **Left_Tail**
6. Using the process described in steps 3 and 4, select one **Midsection**, **Aftbody**, and **Nose** from the Server Browser and drag each to the Assembly View. For these components, accept the dialog box defaults and click **Ok**.

After completing the above steps the ModelCenter main window should look like the one below in Fig. 27.

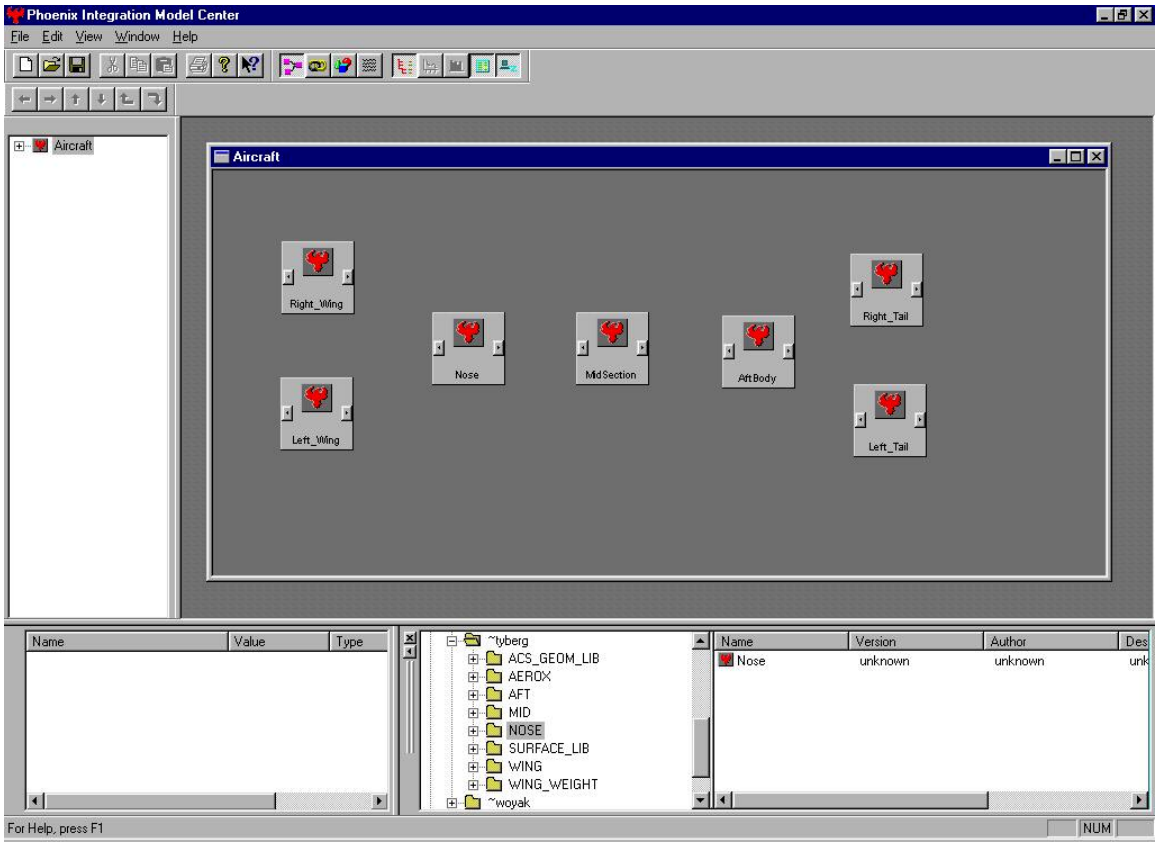

Figure 27. Assembly View of the model.

Note: In the Assembly View, you can move the components around so they are not stacked on top of each other.

Step 2: Modifying the initial settings

After choosing the components, you are ready to modify the initial settings. Each project is different and when you are using the ModelCenter for your own projects, you will need to determine the values and variables for each component.

1. Select **Aircraft** from the component tree and drag it into the **Data Monitor**.
2. Click on the plus sign next to **Aircraft** to display the components that were added to it. The **Data Monitor** should look like the one shown below in Fig. 28.

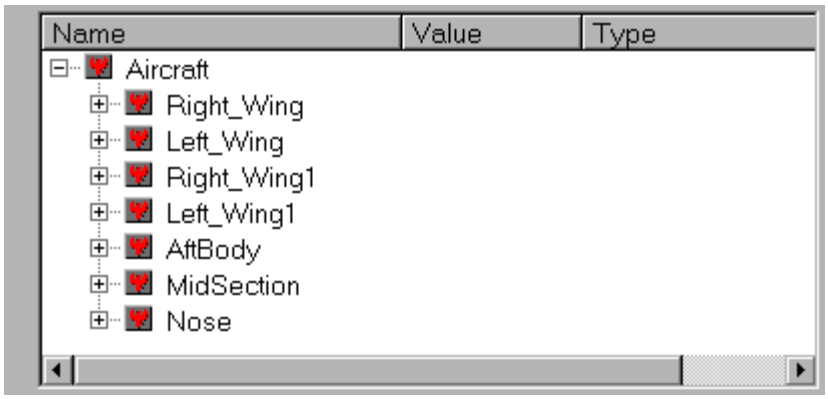

Figure 28. Data Monitor showing the components of the Aircraft example.

3. Click on the plus sign next to **Right_Wing** to display the input and output variables. The Data Monitor shows the value, type, and units of the variable. When **Right_Wing** is expanded, the Data Monitor should look like the one below in Fig. 29.

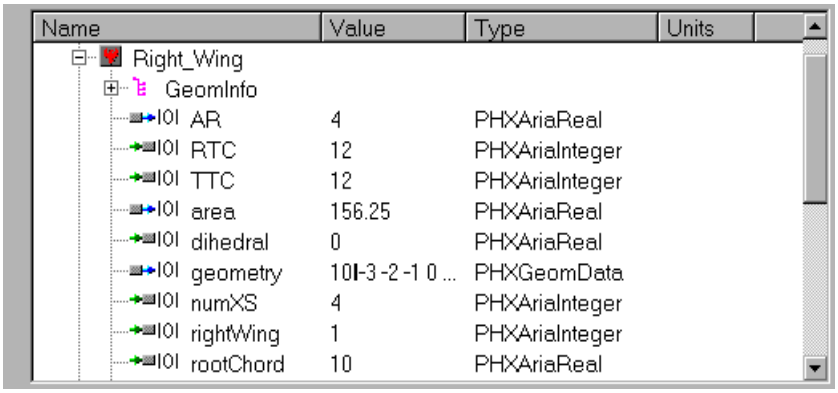

Figure 29. Right_Wing variables displayed in the Data Monitor

4. In the Data Monitor, select **rootChord** from the **Right_Wing** and then click on the number in the Value field. This will allow you to modify the value of rootChord.
5. Enter **2** into the Value field. The output values that are dependent upon root chord will be automatically updated.

6. Select **Span** from underneath **Right_Wing** in the Data Monitor. Change the value to 36
7. Select **Sweep** from the Data Monitor. Change the value to 0.
8. Select **TaperRatio**. Change the value to .6.
9. Using the procedure described in steps 3 through 6, modify the values of the rest of the components.

Left_Wing

Variable	Value
RightWing	0

Right_Tail

Variable	Value
RootChord	2
Span	15
Sweep	0
TaperRatio	0.8

Left_Tail

Variable	Value
RightWing	0

MidSection

Variable	Value
Length	2

AftBody

Variable	Value
Length	10
Rad3	1
Rad4	1

Nose

Variable	Value
Length	6
Rad1	1.5
Rad2	1.5
Tipangle	60

Note: In this tutorial, you are only modifying certain variables. You may need to modify all the variables while configuring your own models.

Step 3: Linking Geometry Modules Together

After you have changed the values of the variables as directed above, you will need to link some of the components together. To link the components:

1. Switch to the Link Editor view. On the right side of the Link Editor open up the Left_Wing and on the left side of the Link Editor open up the Right_Wing.
2. Now, as shown in the tutorial, click on rootchord underneath Left_Wing, then drag the rootchord over from the Right_Wing over to the box next to rootchord of the Left_Wing. After this is done the screen should look like the screenshot shown in Fig. 30:

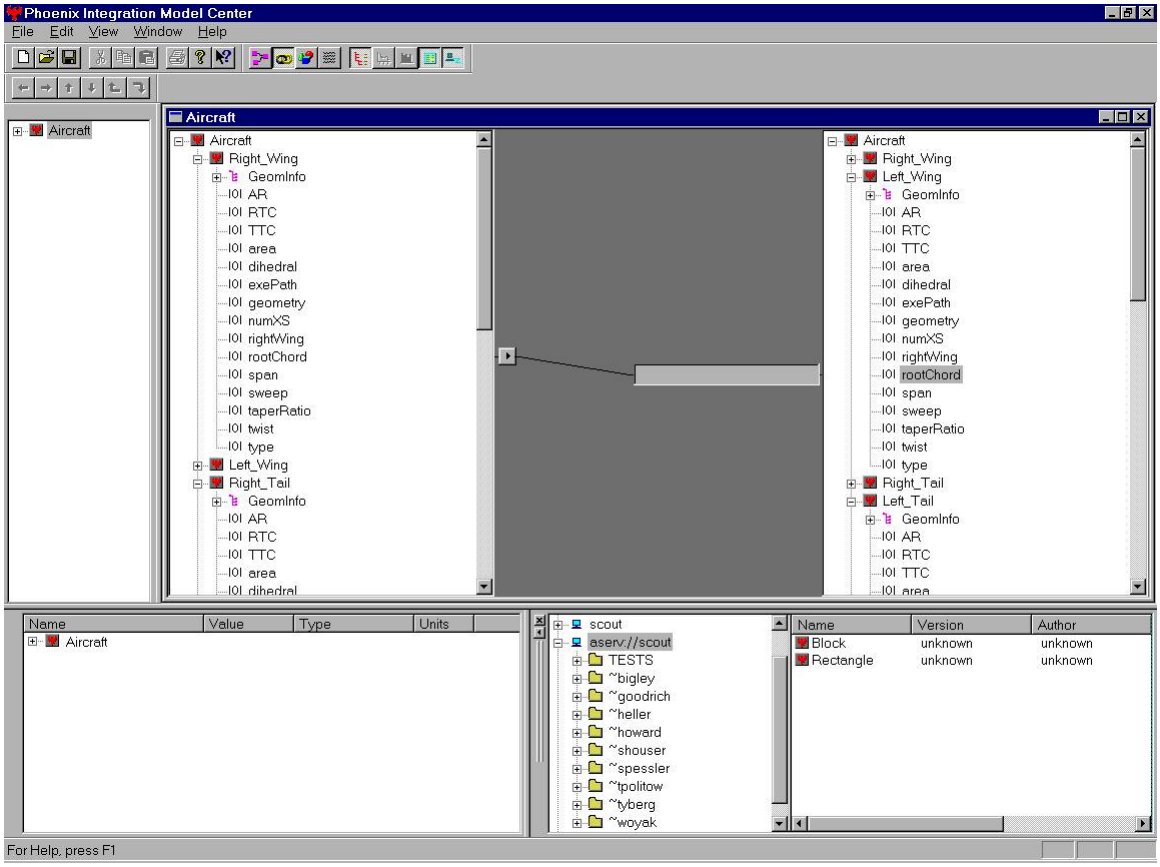

Figure 30. Link Editor showing the rootchord of the Right_Wing linked to the Left_Wing.

3. You want to repeat steps 1 and 2 for the following components:

Right_Wing	Left_Wing
Span	Span
Sweep	Sweep
TaperRatio	taperRatio

Right_Tail	Left_Tail
Rootchord	Rootchord
Span	Span
Sweep	Sweep
TaperRatio	taperRatio

Nose	MidSection
Rad 1	Rad 1
Rad 2	Rad 2
Rad 1	Rad 3
Rad 2	Rad 4

MidSection	Aftbody
Rad 3	Rad1
Rad 4	Rad 2

After these links have been made the screen in the assembly view should look as shown in Fig. 31.

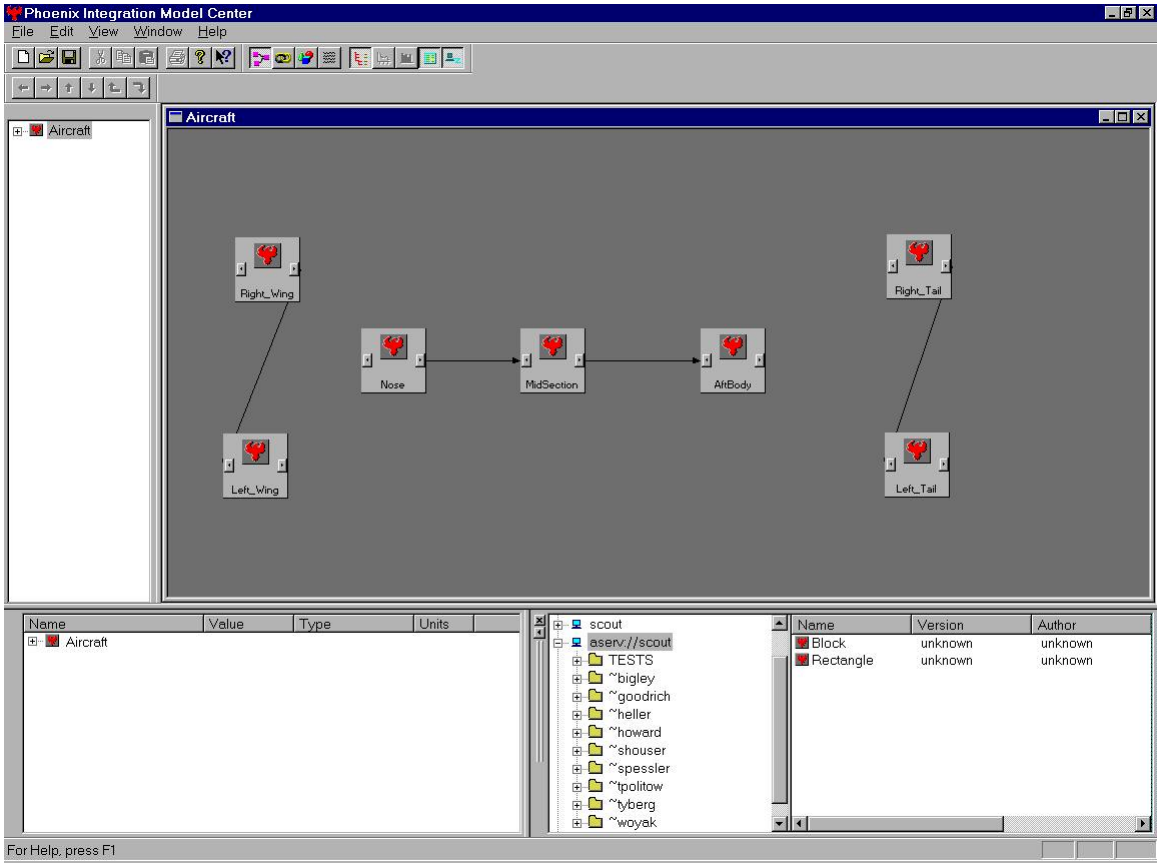

Figure 31. Assembly View showing the model components linked together.

Step 4: Fixing the Orientation of the Components

After you have modified the default values of the components, you will need to position the components. To position the components:

1. Switch to the Geometry view. The Geometry View allows you to see a geometric rendering of the components. Because the components have not yet been positioned, they may appear to be on top of each other, as shown in Fig. 32.

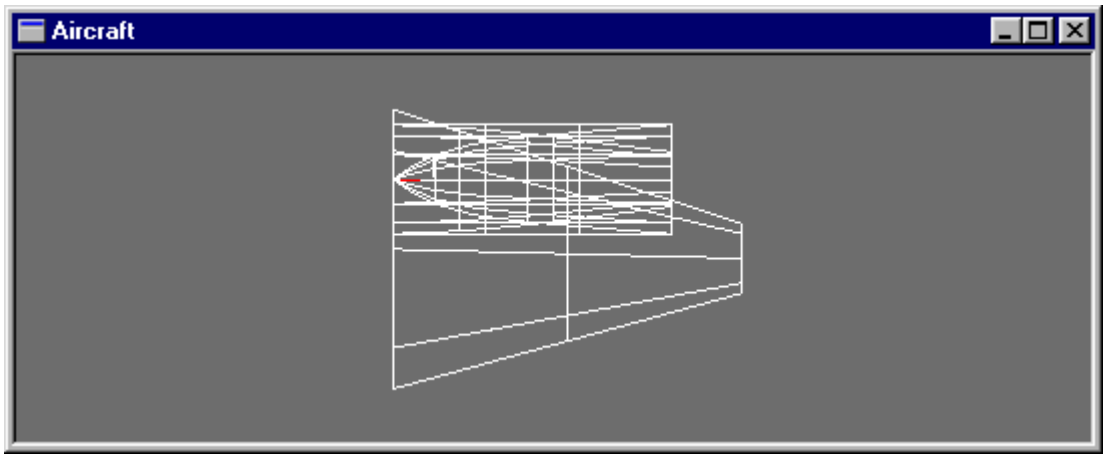

Figure 32. Geometry View of components.

2. Click on **Right_Wing** in the Data Monitor. The component will expand and list the variables that comprise Right_Wing.
3. Open **Geominfo**.
4. Open **Orientation** listing. This gives a listing of XYZ rotations in degrees and translations in scaled units, as shown in Fig 33.

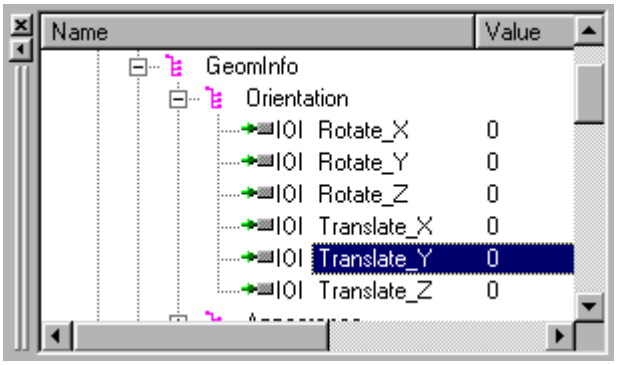

Figure 33. Orientation variables as seen in the Data Monitor.

5. Set **Rotation Y axis** of the **Right_Wing** to –5.
6. Set **Translation Z axis** to –0.75. The **Right_Wing** will move to the specified position.
7. Using the steps described in Steps 2 through 5 set the geometric variables in the remaining components to the values listed below.

Left_Wing

Variable	Value
Rotation Y axis	5
Translation X axis	-3
Translation Z axis	-0.75

Right_Tail

Variable	Value
Rotation Y axis	25
Translation X axis	-0.5
Translation Y axis	-9

Left_Tail

Variable	Value
Rotation Y axis	-25
Translation X axis	-2.5
Translation Y axis	-9
RightWing	0

MidSection

Variable	Value
Rotation Z axis	270
Translation X axis	-1.5
Translation Y axis	0.5

AftBody

Variable	Value
Rotation Z axis	270
Translation X axis	-1.5
Translation Y axis	-1.5

Nose

Variable	Value
Rotation Z axis	270
Translation X axis	-1.5
Translation Y axis	6.5

The aircraft should now look like Fig. 34.

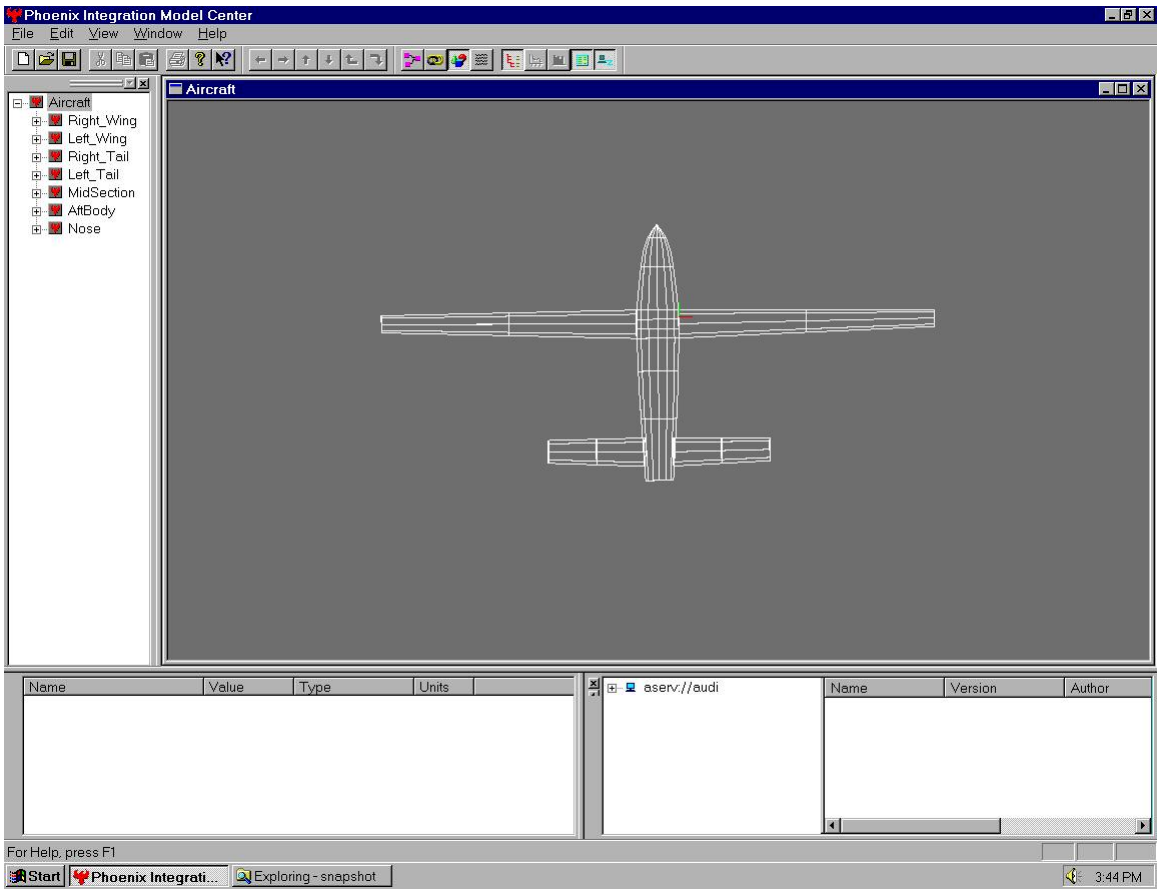

Figure 34. Geometry View displaying the completed aircraft model.

Note: Move the mouse while holding down the left mouse button in the Geometry window to view the visual representation from different perspectives.

Note: By default, the Geometry View shows geometric components as wireframe objects. You can right click on the configuration window to **View: Shade All** to view the components as solid objects.

This completes the "building an aircraft model example". For further analysis go to the next example.

Chapter 5. Example 2 - Mission Weights Example

This tutorial expands on advanced concepts that were introduced earlier. For a brief overview of ModelCenter's capabilities, please refer to the Tutorial in Chapter 3.

In this example we will be estimating the gross weight of the aircraft designed in Example 1. To do this we will employ a component, which calculates fuel weights and estimates the empty weight for the aircraft. The component requires aerodynamic and propulsive information about the aircraft as it flies along its mission. This necessitates the use of additional modules and the **Link Editor**. The goal of this example is to impart valuable experience using the integration capabilities of ModelCenter™.

About the Example

A specific mission or goal needs to be specified to size the aircraft. For this lesson's purposes we will assume a simple mission profile with no weapons drop. We are informed the aircraft must carry a payload of surveillance equipment weighing 200 pounds. The **Mission_Weights** component will calculate the aircraft's gross takeoff weight. It estimates the fuel burned in each segment of the mission to assess the necessary amount of fuel. A statistical method is employed to calculate the empty weight of the aircraft. Since there is no flight crew the take off gross weight is assumed to be the sum of the empty weight, the fuel weight, and the payload weight. For more information on this component see Appendix A2, Description of Components.

From this fictional RFP (Request For Proposal) we assume the following mission:

1. Taxi and Take Off
2. Climb to a cruise altitude of 5000 meters and a cruise speed of mach 0.3
3. Cruise at cruise conditions to a target area 463,000 meters (250 nautical miles) out
4. Loiter on site and collect data for an hour
5. Cruise home
6. Descend
7. Loiter at base awaiting landing permission (maximum time 30 minutes)
8. Land and taxi to hanger

Getting Started

1. If ModelCenter is not currently opened double click on the ModelCenter icon.
2. Open the saved UAV model created in Example 1. If you did not complete this lesson you can open the file, example1.pcx, which accompanied the tutorial. It can be found in the **tutorial\saved** directory.* The main window should look like the view shown in Fig. 35:

* The actual directory for these files may change depending on the installation of the tutorial. This is true for all the file and directory names cited in this chapter.

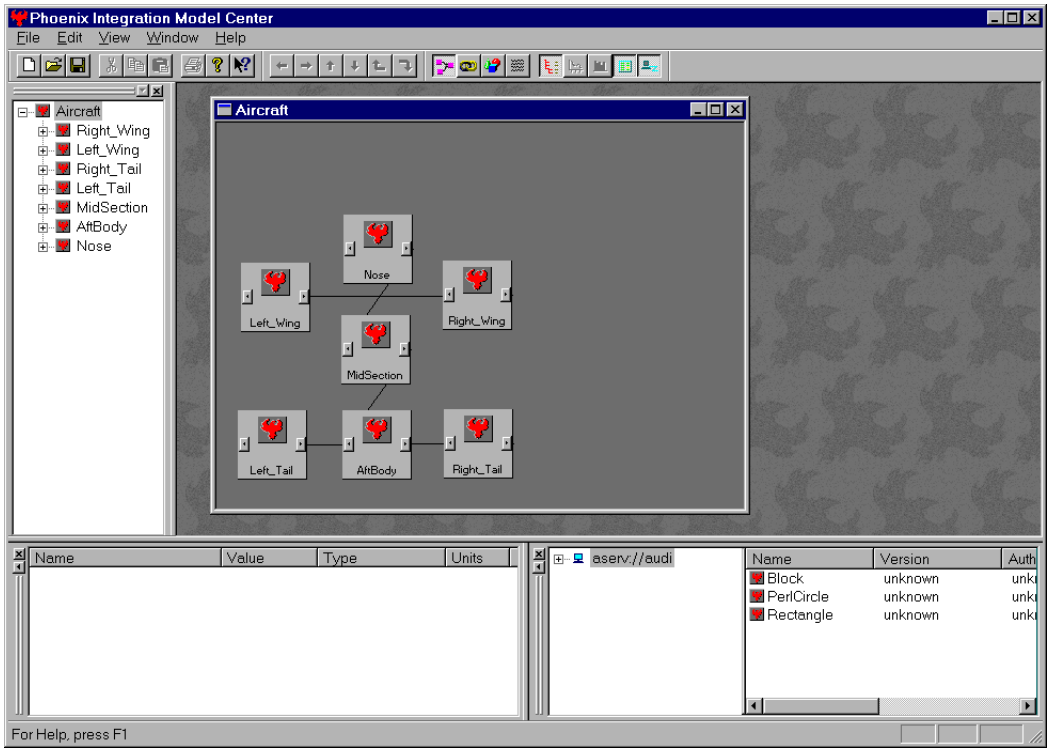

Figure 35. ModelCenter with a new configuration window opened

Setting up the weights analysis

After opening the main window, you will retrieve the components necessary for the analysis and modify their input variables.

Step 1: Getting the necessary modules

1. From the Server browser, choose **aserv://audi**.
2. Pull up the following modules and change their names as follows.

Components

Module	Folder	Name
Polar	**~bigley\AEROeqns**	**Cruise_Segment**
Polar	**~bigley\AEROeqns**	**Loiter_Segment**
BADAPropulsion	**~bigley\BADA**	**Cruise_Propulsion**
BADAPropulsion	**~bigley\BADA**	**Loiter_Propulsion**
Mission_Weights	**Tutorial\aero**	**Mission_Weights**
Conversion	**Tutorial\aero**	**Conversion**

Note: for a description of the components and an explanation of the calculations they perform see the components' section in the Appendix (A2).

3. Arrange the boxes in the assembly view so that the ModelCenter main window looks like Fig. 36.

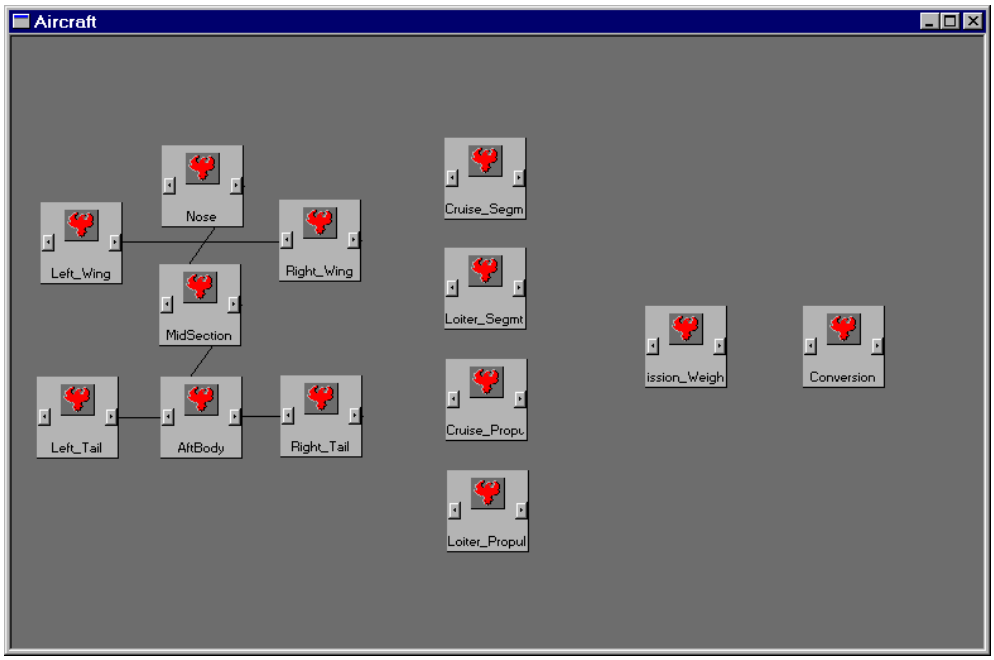

Figure 36. ModelCenter with the components arranged in the assembly view

Step 2: Inputting the proper data

1. Select **Aircraft** from the component tree and drag it into the **Data Monitor**. In this manner more then one component can be brought into the **Data Monitor**.
2. Using the F2 key or by directly clicking on the values, modify the component values.

Note: the values are now in SI units as required by the **Mission_Weights** module.

Component 1- Wing

Variable	**Value**
RootChord	0.6096
Span	12.192
Sweep	0
TaperRatio	0.5

Component 2- Cruise_Segment

Variable	**Value**
Altitude	5000
DeltaT	0
Cdo	0.02
Eff	0.8
MachNumber	0.3
Weight	8899.44

Component 3- Loiter_Segment

Variable	Value
Altitude	4000
DeltaT	0
Cdo	0.02
Eff	0.8
MachNumber	0.3
Weight	8899.44

Component 4- Cruise_Propulsion

Variable	Value
Actype	Uav1
Altitude	5000
DeltaT	0
Flag	2
MachNumber	0.3
Weight	8899.44

Component 5- Loiter_Propulsion

Variable	Value
Actype	Uav1
Altitude	4000
DeltaT	0
Flag	2
MachNumber	0.3
Weight	8899.44

Component 6- Mission_Weights

Variable	Value
A	0.91
C	-0.05
CrewWeight	0
CruiseRange	463000
LoiterEndurance	3600
PayloadWeight	889.94

Step 3: Linking components

Now we can link the necessary data to the **Mission_Weights** module. Click on the **Link Editor** button to open the link editor window.

1. Using the skills learned in the Tutorial link the following data between modules.

From: Right_Wing	**To: Cruise_Segment**
Ar	AspectRatio
Area	WingArea

From: Right_Wing	**To: Loiter_Segment**
Ar	AspectRatio
Area	WingArea

From: Cruise_Segment	**To: Mission_Weights**
LoverD	CruiseLD
Velocity	CruiseVel
MachNumber	Mach

From: Loiter_Segment	**To: Mission_Weights**
LoverD	CruiseLD

From: Cruise_Propulsion	**To: Mission_Weights**
Tsfc	CruiseSfc

From: Loiter_ Propulsion	**To: Mission_Weights**
Tsfc	LoiterSfc

From: Mission_Weights	**To: Conversion**
EmptyWeight	EmptyWeight
FuelWeight	FuelWeight
Weight	Weight

Now ModelCenter should appear as shown in Fig 37.

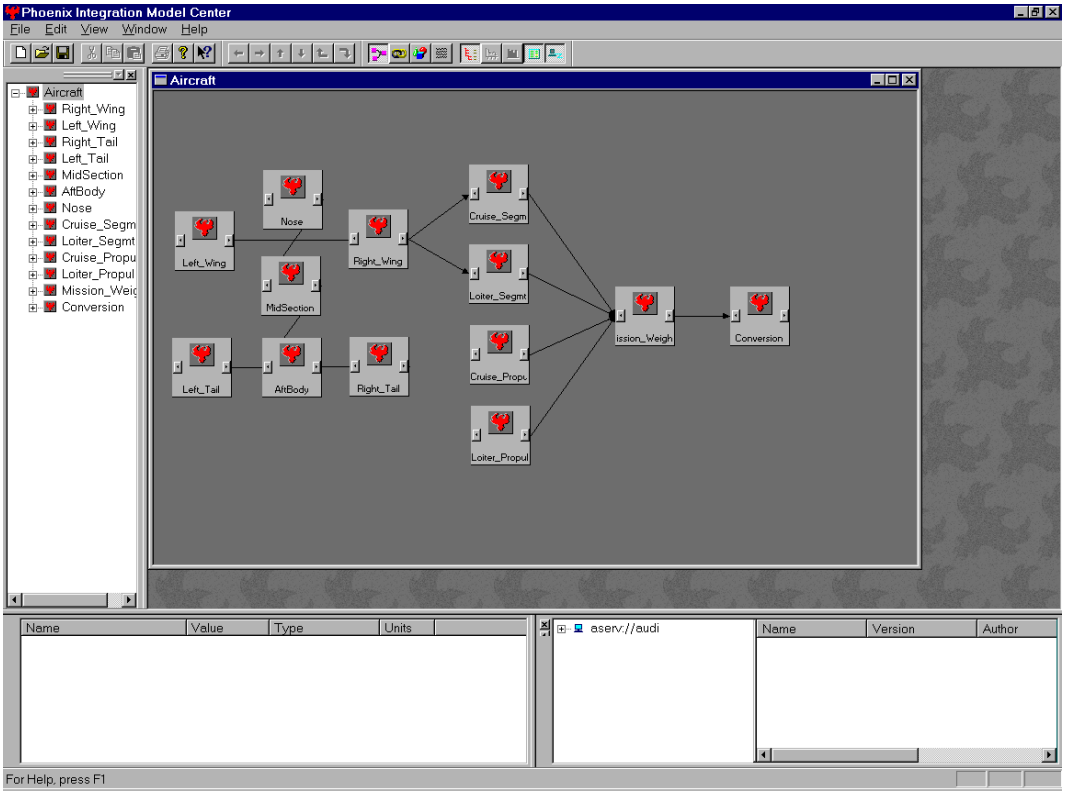

Figure 37. ModelCenter with linked components

Running the Analysis

Now that everything is properly set up, running the analysis is effortless. Simply click on one of the variables with the red X's in the Conversion module. ModelCenter will update every module that the variable depends on. The estimated weight should be about 2680 lbs.

Note: When validating a variable ModelCenter will only update the modules that the variable depends on. This means that modules linked to the variable will update, but modules that the variable links to will not. For instance, clicking on the emptyWeight in **Mission_Weights** will not cause the variables in **Conversion** to update.

With the components properly linked ModelCenter becomes a powerful tool. Running sensitivity tests on the aircraft model are now as easy as changing a value anywhere in the system and updating the components.

Chapter 6. Example 3 – Driving ModelCenter™ from Other Platforms

This example introduces the concept of running ModelCenter™ from MS Excel using Visual Basic commands. We will be revisiting Example 2, "Mission Weights Example". This time the mission performance and sizing will be driven from an **Excel Macro** rather then by a component in ModelCenter. This macro will still require aerodynamic and propulsive data about the model during different stages of flight. ModelCenter™ will provide this data by employing polar and propulsion components in conjunction with the model.

About the Example

For this lesson's purposes we will assume the same simple mission profile used in Example 2. The Excel macro will calculate the amount of fuel burned in each segment and estimate the empty weight of the aircraft. Raymer's improved statistical equation is used for the empty weight estimation(Ref. 2). The ModelCenter file will contain the model designed in Example 1 and polar and propulsion components to calculate aerodynamic and propulsive data for given flight conditions. To lower program execution time, simplified versions of **Polar** and **BADA_Propulsion** will be used.

Getting Started

1. If ModelCenter is not currently opened double click on the ModelCenter icon.
2. Open the saved UAV model created in Example 1. If you did not complete this lesson you can open the file, example1.pcx, which accompanied the tutorial. It can be found in the **tutorial\saved** directory. The main window should look like the view shown in Fig. 38.

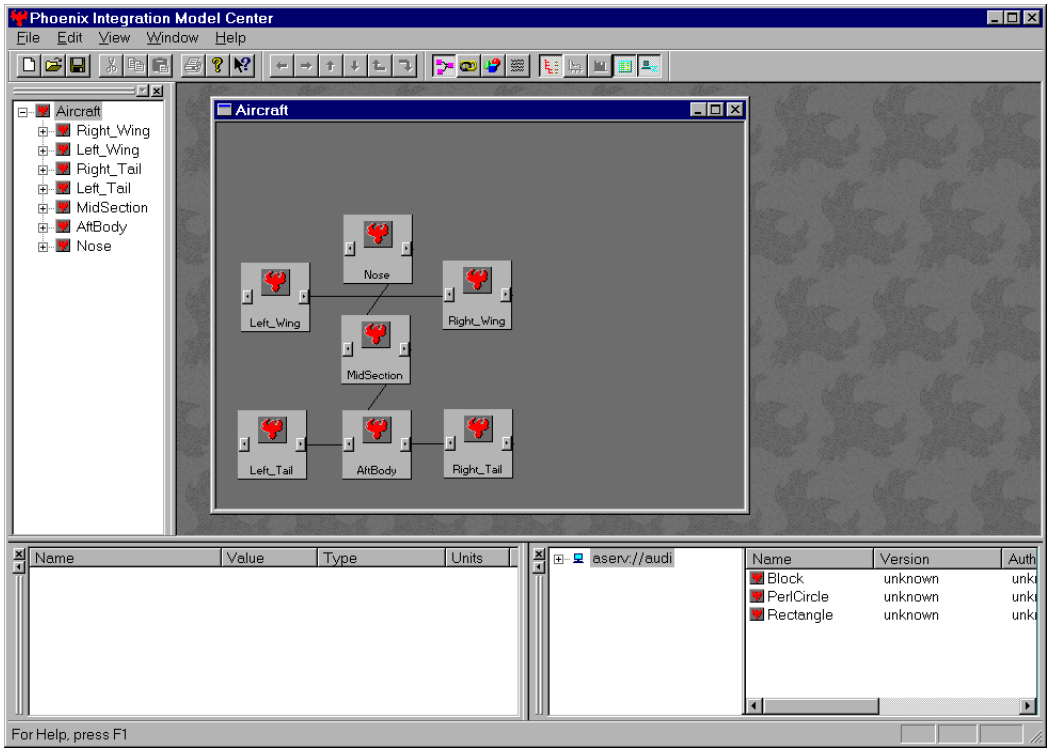

Figure 38. ModelCenter with a new configuration window opened

3. Rename the configuration window as **MissionAnalysis**.

Setting up the mission performance analysis

After opening the main window, you will retrieve the components necessary for the analysis and set up the appropriate links.

Step 1: Getting the necessary modules

1. From the Server browser, choose **aserv://audi**.
2. Pull up the following modules and change their names as follows.

Components

Module	Folder	Name
PolarTwo	**Tutorial\AEROeqns**	**Segment_Polar**
PropulsionTwo	**Tutorial\AEROeqns**	**Segment_Propulsion**
Converter	**Tutorial\AEROeqns**	**Converter**

Note: for a description of the components and an explanation of the calculations they perform visit the components' section of the appendix.

3. Arrange the boxes in the assembly view so that the ModelCenter main window looks like that shown in Fig. 39 below.

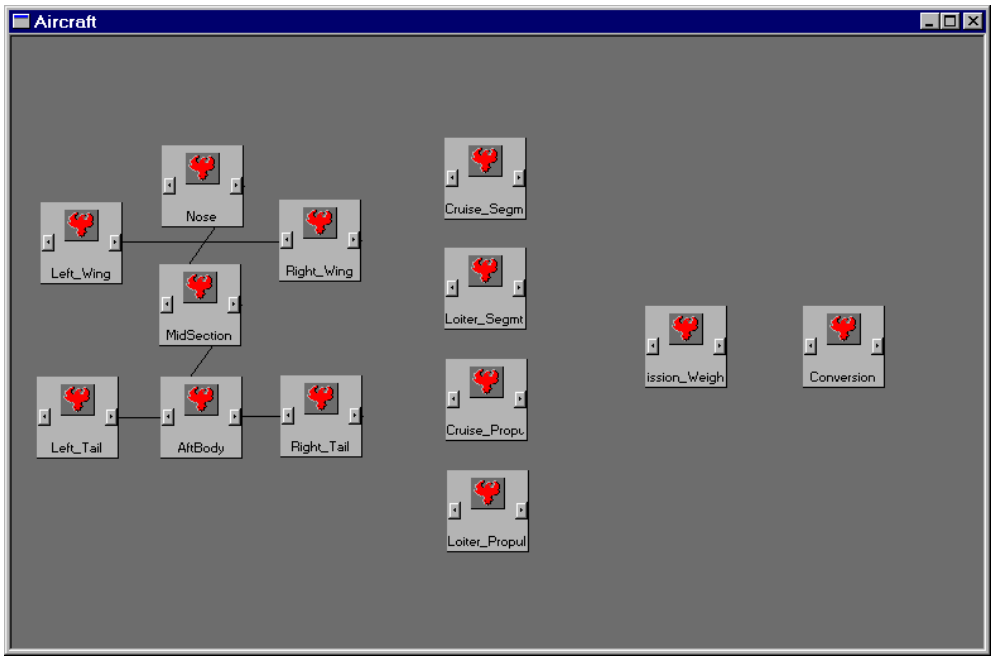

Figure 39. ModelCenter with the components arranged in the assembly view

Step 2: Linking components

Now we can link the necessary data to the **Mission_Weights** module. Click on the **Link Editor** button to open the link editor window.

1. Using the skills learned in the Tutorial link the following data between modules.

From: Right_Wing	**To: PolarTwo**
Ar	AspectRatio

From: Right_Wing	**To: Converter**
area	ftsq

From: Right_Wing	**To: PolarTwo**
output	Area

Now ModelCenter should appear like Fig. 40.

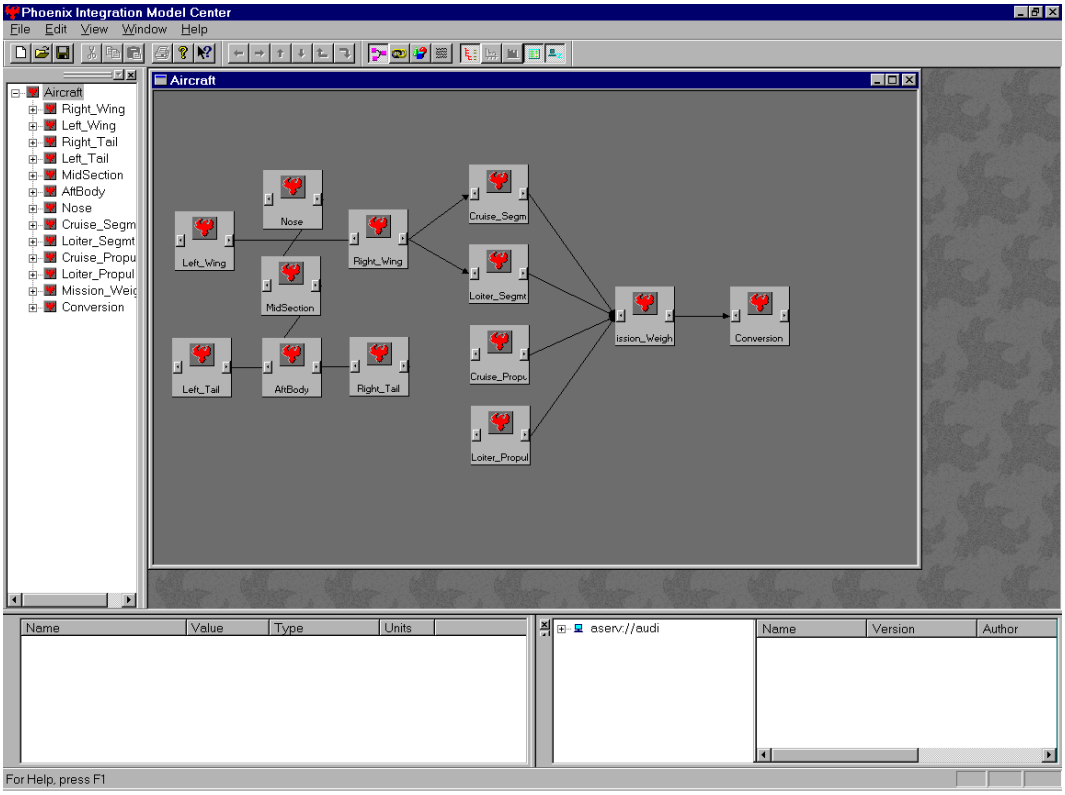

Figure 40. ModelCenter with linked components

2. Save the file as analysis.pcx in the **tutorial/saved** directory.

Using the Excel Spreadsheet

The Excel spreadsheet has already been created. Open the Excel file, Example3.xls, in the **tutorial\saved** directory. The main window should look like the one in Fig. 41. Note that the concept for this tutorial was to allow the user to download a compressed file containing all of the analysis and geometry files. When expanded, the files would be arranged in a set of directories that would correspond to the directory names referred to in this report.

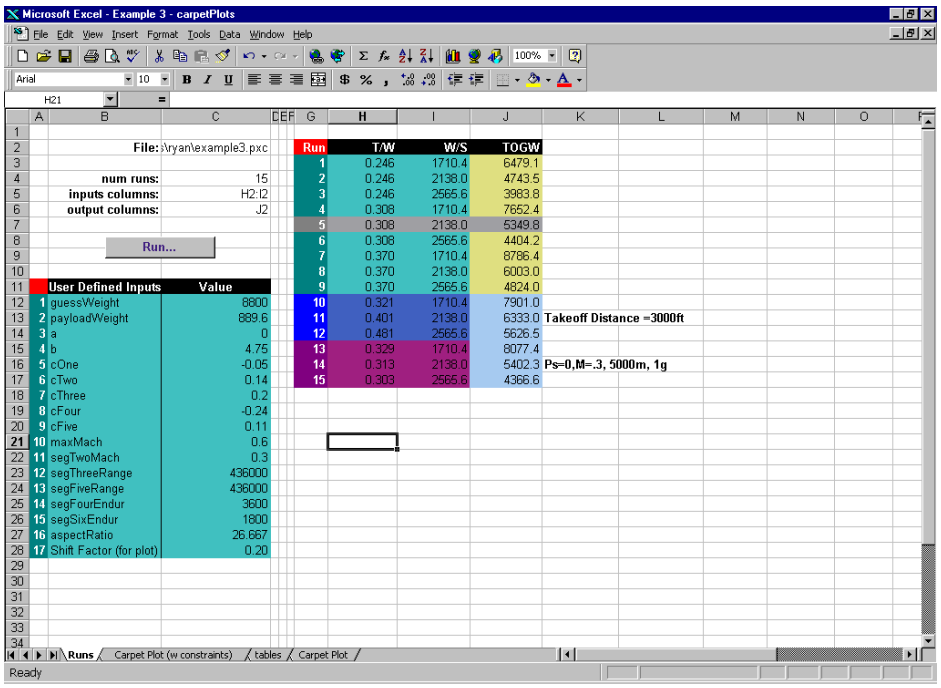

Figure 41. Mission Analysis Excel Spreadsheet

The first sheet contains cells where the user inputs data on the aircraft, including a set of base line thrust to weight and wing loading ratios. The rest of the T/W and W/S for the sizing matrix are automatically calculated. For more information on the inputs see the example3.xls entry in the Appendix.

The mission macro is assigned to the run button. When started, the macro will clear the TOGW output column. Following an iterative method the macro begins with an estimated weight and calculates the fuel burned during each segment of the mission. A total fuel weight is calculated accounting for reserve and trapped fuel. As the weight decreases during each segment the macro must have updated aerodynamic and propulsive data. When needed, the macro will send current weight, altitude, and mach data to ModelCenter, which will then calculate updated lift to drag and thrust specific fuel consumption ratios. A statistical method is used to estimate the empty weight of the aircraft. This weight added to the fuel and payload weights gives an estimate of the takeoff gross weight of the aircraft. A new guess weight is chosen between the original guess and the calculated weight. The macro iterates this process until it finds that the TOGW and the estimated weight are within a tolerance of each other. This final value is outputted to **Runs** sheet.

Taking a look at the second sheet, the user should see a carpet plot like that shown in Fig 42. If portions of the plot appear off the chart then the axis' scales need to be adjusted.

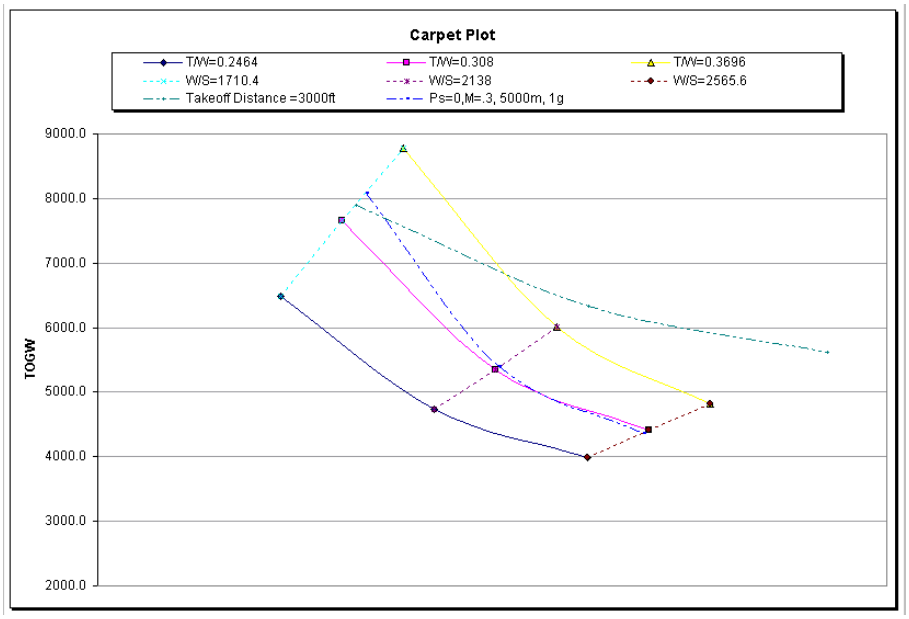

Figure 42. Carpet Plot

The visual Basic commands for using ModelCenter are as follows. Note that there are no commands to run or update the ModelCenter file. ModelCenter will automatically update when the **getValue** function is used. It will not update if no inputs have been changed.

Connecting to the analysis server and creating a ModelCenter object:

```
Dim ModelCenter As Object
Set ModelCenter = CreateObject("Phoenix.ModelCenter")
```

Opening the ModelCenter file:

```
ModelCenter.loadFile filename
```

Setting the value of a variable in the ModelCenter file:

```
ModelCenter.setValue "Configuration name. Component name. Variable name", value
```

Retriving a value from ModelCenter:

```
Variable = ModelCenter.getValue("Configuration name. Component name. Variable name ")
```

To view the full Macro used in this excel spreadsheet go to the macro menu under the Tools menu, or press Alt + F8.

ModelCenter's capability to be driven from other applications makes it a useful tool even for companies with many existing analysis packages. ModelCenter gives the user the ability to make rapid changes to the model and rerun analysis software to validate their design.

Chapter 7. Conclusions

In the summer and fall of 1998, three Virginia Tech undergraduate students developed several tutorials to illustrate the use of the new Phoenix Integration software product ModelCenter™. The students were located at Phoenix Integration, working closely with the developers of ModelCenter™. These tutorials were then used to illustrate the product to new users and potential customers. Because of the unique characteristics of ModelCenter™, these demonstrations and tutorials are essential to illustrate the impact ModelCenter™ can make on engineering productivity. Feedback from Phoenix has been positive. The outcome demonstrates the value of Virginia's CIT in helping small companies compete in the marketplace. Finally, the students got valuable experience, seeing the requirements for successful product development.

References

1. Arvid Myklebust and Paul Gelhausen, “Putting the ACSYNT on Aircraft Design,” *Aerospace America*, Vol. 32, No. 9, Sept. 1994, pp. 26-30.

2. Raymer, Daniel P., *Aircraft Design: A Conceptual Approach*, 2nd Ed., AIAA, Washington, 1992.

APPENDIX

A1. ModelCenter's Application Windows

ASSEMBLY VIEW

In Fig. A1, the assembly view shows the module blocks in the configuration window.

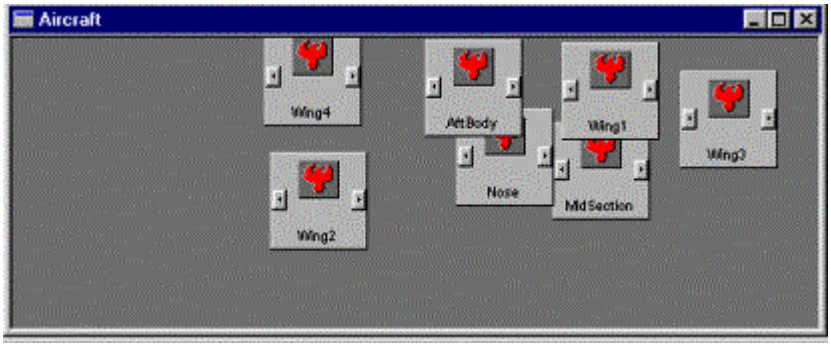

Figure A1. Configuration Windo shown in assembly view

COMPONENT TREE

The Component Tree, Fig. A2, is the window that allows the user to view the modules that he/she has pulled into the configuration window. This will show the variables that are contained in each module but one can not modify the values in this window. The Data Monitor exists for this purpose.

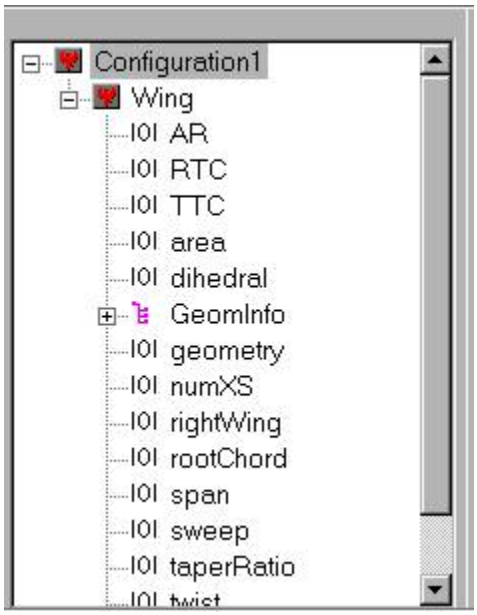

Figure A2. Component Tress

CONFIGURATION WINDOW

The Configuration Window, Fig. A3, is the area that the user will pull modules to from the server browser. It also allows the user to view the geometry of what he/she has designed or simply the modules boxes that have been pulled up to be worked on if no geometry is involved. Figure A4 shows a geometry view in the configuration window.

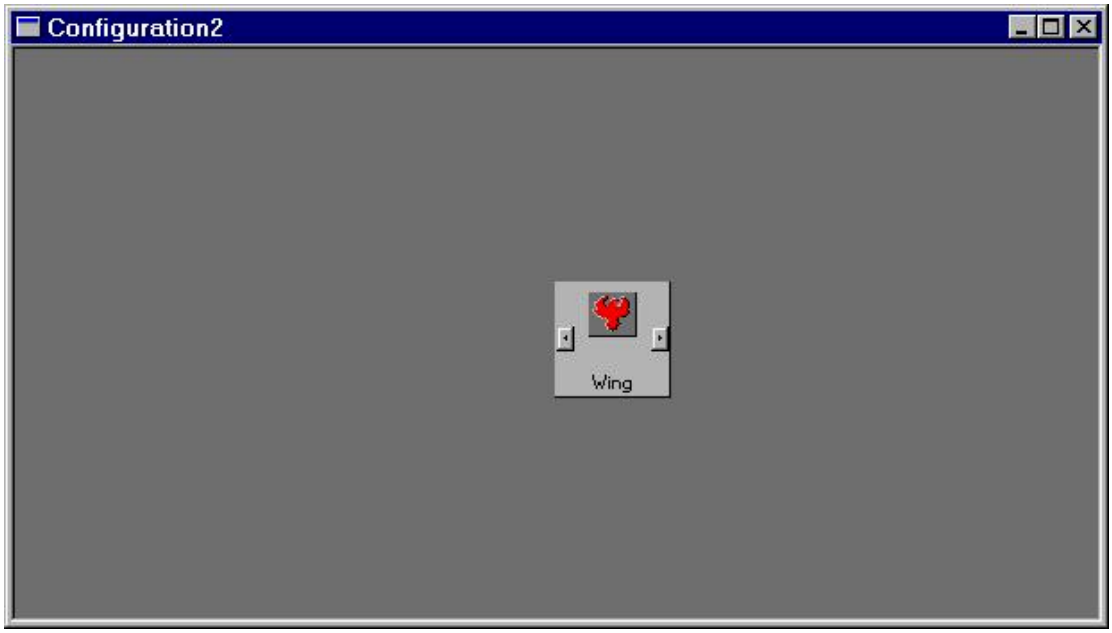

Figure A3. Configuration Window

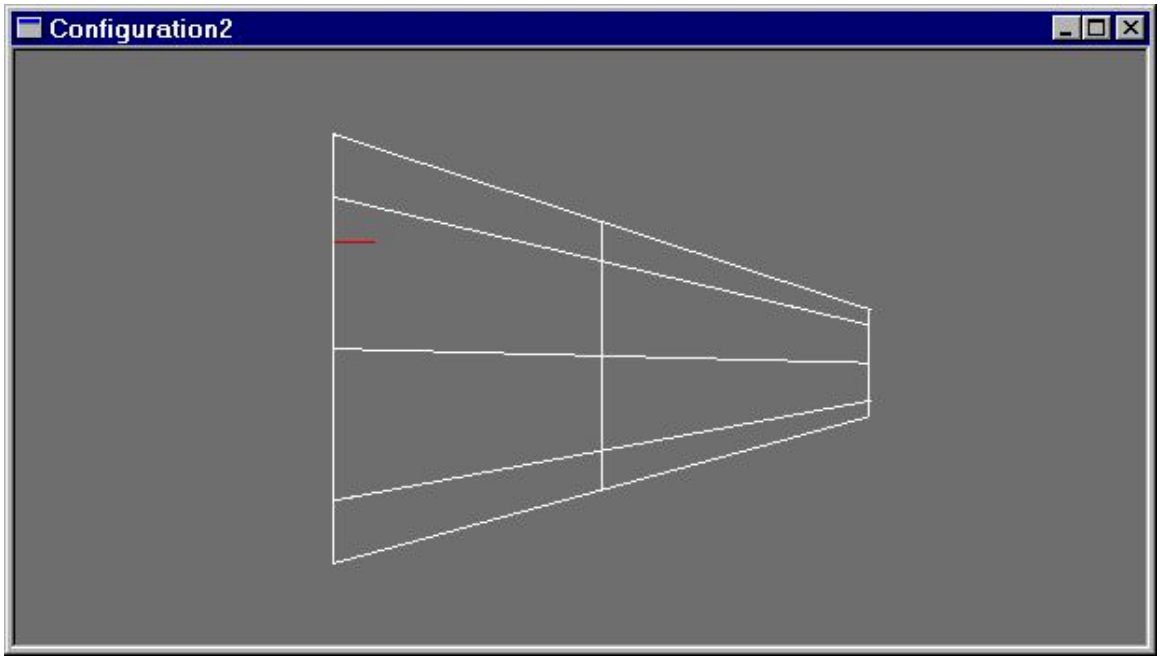

Figure A3. Configuration Window in Geometry View Mode

DATA MONITOR

The Data Monitor shown in Fig. A5 is a window that allows the user to view and edit all of the variables within the chosen module. For example: when Wing is chosen and dragged down into the Data Monitor, then opened up it will look like the following picture.

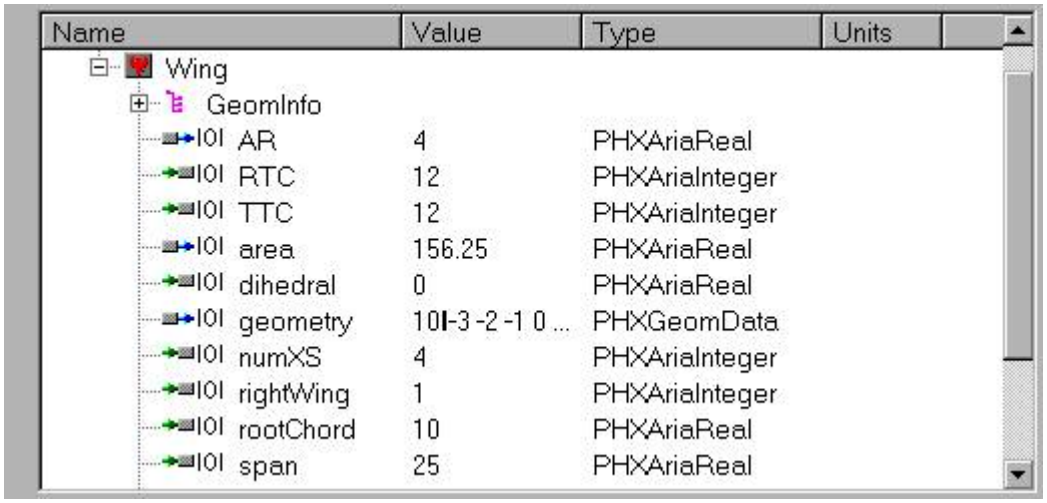

Figure A4. Data Monitor Window

One can then modify default settings by clicking on the value of the variable of which the change is desired.

GEOMETRY VIEW

The geometry view allows the user to see the built model in the configuration window in either a rendered or wireframe view. An example of what the window would look like when the geometry view button is sen in Fig. A6.

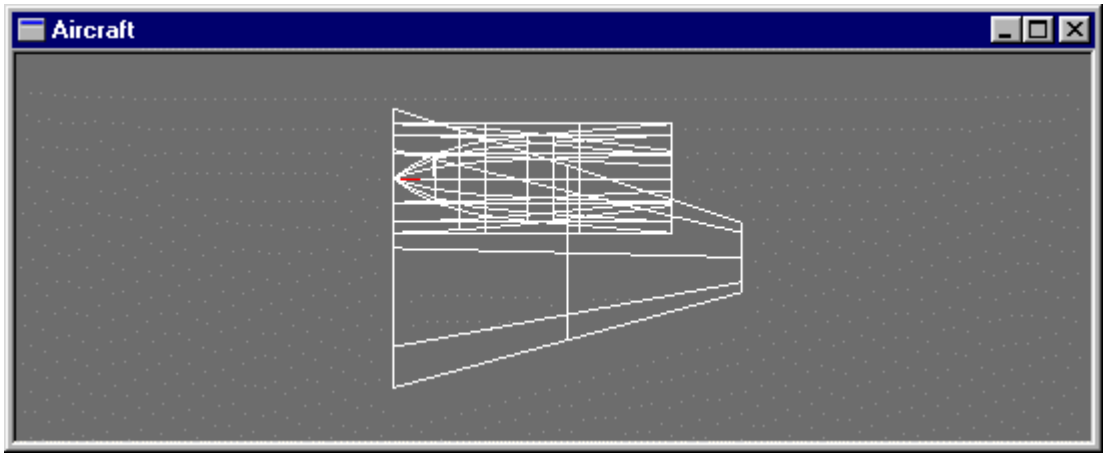

Figure A6. Configuration Window in Geometry View Mode

LINK EDITOR

The Link Editor, Fig. A7, is a tool that allows different modules to be linked together. For example: One can link the aspect ratio of a wings geometry to the aspect ratio in the drag polar module. At this time all variables dependent on the aspect ratio in the Polar module will be updated.

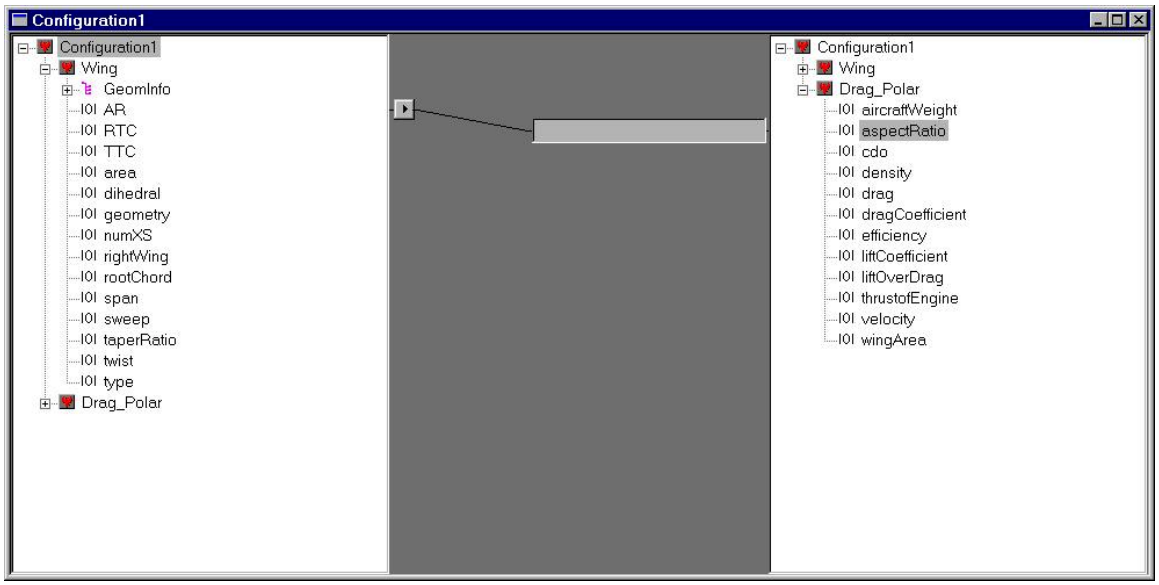

Figure A7. Link Editor Window

SERVER BROWSER

The Server Browser, shown in Fig. A8, is the window that controls the destination of the desired server for a particular function. For example: For aircraft geometry :

- Open aserv://audi
- Open ~tyberg
- Click on desired module.

The module is then ready to be dragged into the configuration window.

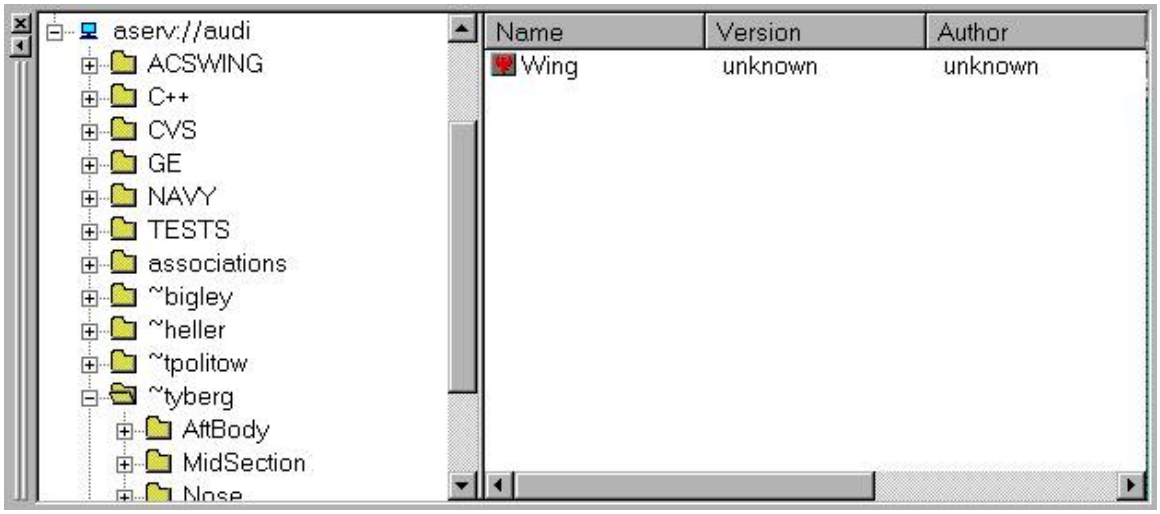

Figure A8. Server Browser Window

A2. DESCRIPTION OF COMPONENTS

ABOUT THE COMPONENTS

The following shows a description of the variables and assumptions for the components used in this tutorial.

SEGMENT POLAR MODULE

This module calculates the aerodynamic lift and drag forces for the segment's flight conditions. It takes geometric information from the aircraft's lifting surface, the wing module, and passes aerodynamic information to the mission weight analysis module.

Variables

altitude:	the altitude at which the aircraft begins its segment. (Meters)
deltaT:	the difference between standard atmospheric sea level temperature and the operating sea level temperature during the segment. (Kelvin)
cdo:	drag coefficient at zero lift or parasitic drag. (Non-dimensional)
cd2:	the drag due to lift factor. (Non-dimensional)
eff:	Oswald span efficiency factor of the wing. (Non-dimensional)
machNumber:	the mach at which the aircraft is flying during the cruise segment. (Non-dimensional)
weight:	the weight of the aircraft during the cruise segment. (Newtons)
aspectRatio:	the aspect ratio of the wing. (Non-dimensional)
wingArea:	the reference area of the wing. ($Meters^2$)
drag:	the drag force exerted on the aircraft. (Newtons)
dragCoefficient:	the drag coefficient. (Non-dimensional)
liftCoefficient:	the lift coefficient. (Non-dimensional)
loverD:	lift to drag ratio. (Non-dimensional)
velocity:	true air speed of the aircraft. (Meters / seconds)

Assumptions

The calculations are based on straight and level flight assumptions. For instance lift is assumed to be equal to weight.
Atmospheric conditions (temperature, density and speed of sound) are calculated using equations from BADA version 2.4
Other equations used :
Drag Coefficient: $C_D = (C_{DO} + C_{D2}C_L^2) (1 + C_{M16}M^{16})$
Drag: $D = C_D \rho V_{TAS} S / 2$
Lift Coefficient: $C_L = 2mg/\rho V_{TAS}^2 S$

SEGMENT BADAPROPULSION MODULE

This module estimates the thrust and fuel consumption rate of the aircraft's engines for the segment's flight conditions. The equations and variables used are based on the Base of Aircraft Data version 2.4 methods developed by the Eurocontrol Experimental Centre. It passes the thrust and fuel consumption data to the mission weights module.

Variables

actype: the aircraft type's BADA designation.
altitude: the altitude at which the aircraft begins its segment. (Meters)
deltaT: the difference between standard atmospheric sea level temperature and the operating sea level temperature during the segment. (Kelvin)
machNumber: the mach at which the aircraft is flying during the cruise segment. (Non-dimensional)
weight: the weight of the aircraft during the cruise segment. (Newtons)
flag: a segment type designator. (0 – Take-Off, 1 – Climb, 2 – Cruise, 3 – Descent)
thrust: the maximum installed thrust under the segment's flight conditions. (Newtons)
tscf: thrust specific fuel consumption of the engine due to the segment's flight conditions. (1 / seconds)

BADA Variables

engineType: the type of engine used in the aircraft's propulsion system. ("Jet", "Turboprop", or "Piston")
massReference: aircraft's reference mass. (Tonnes)
Ctcone: C_{TC1}. (Jet / Piston – Newtons, Turboprop – knot x Newtons)
Ctctwo: C_{TC2}. (Feet)
Ctcthree C_{TC3}. (Jet – 1/feet^2, Turboprop – Newton, Piston – knot x Newton)
Ctcfour: C_{TC4}. (^{o} Celsius)
Ctcfive: C_{TC5}. (Non-dimensional)
Ctcr: C_{TCR}. (Non-dimensional)
Ctdesl: $C_{TDES,low}$. (Non-dimensional)
Ctdesh: $C_{TDES,high}$. (Non-dimensional)
atDescent: reference descent altitude. (Feet)
velDescent: reference descent velocity. (knots)
machDescent: reference descent mach number. (Non-dimensional)
Cfone: C_{F1}. (Jet – kg/minute/kN, Turboprop – kg/minute/kN/knot, Piston – kg/minute)
Cftwo: C_{F2}. (knots)
Cfthree: C_{F3}. (kg/minute)
Cffour: C_{F4}. (Feet)

Assumptions

All equations used are from the BADA version 2.4 manual. They are as follows:
Max Climb Thrust
Jet Engine: $(T_{maxclimb})_{ISA} = C_{Tc,1}(1 - h/C_{Tc,2} + C_{Tc,3}h^2)$
TurboProp Engine: $(T_{maxclimb})_{ISA} = C_{Tc,1}(1 - h/C_{Tc,2}) / V_{TAS} + C_{Tc,3}$
Piston Engine: $(T_{maxclimb})_{ISA} = C_{Tc,1}(1 - h/C_{Tc,2}) + C_{Tc,3}/V_{TAS}$
Max Takeoff Thrust
All engine types: $(T_{takeoff})_{max} = 1.2\ T_{maxclimb}$

Thrust at Cruise
All engine types: $(T_{cruise}) = C_{Tcr} * T_{maxclimb}$

Descent Thrust
All engine types: $(T_{des,low})_{nom} = C_{Tdes,low} (m / m_{ref})^{.5} (T_{maxclimb})_{nom}$

The values calculated assume a full or near full thrust setting.

MISSION PERFORMANCE MODULE

This program analyzes an aircraft for a set mission in order to estimate the take off gross weight. The mission consists of eight segments.

Taxi and Take Off.
Climb and accelerate to the cruise altitude and speed.
Cruise out.
Loiter at target.
Cruise back.
Loiter at base.
Descend.
Land and taxi.

Variables

CruiseLD:	the lift to drag ratio during the cruise segments. (Non-dimensional)
LoiterLD:	the lift to drag ratio during the loiter at target segment. (Non-dimensional)
Mach:	the mach number during the cruise segments. (Non-dimensional)
CruiseRange:	the range covered during the cruise segments. (Meters)
CruiseVel:	the aircraft's velocity during the cruise segments. (Meters / second)
CruiseSfc:	the thrust specific fuel consumption of the aircraft's engines during the cruise segments. (1 / second)
LoiterEndure:	loiter endurance, the time spent in the loiter at target segment. (Seconds)
LoiterSfc:	the thrust specific fuel consumption of the aircraft's engines during the loiter segments. (1 / second)
A:	multiplier in estimation of empty weight fraction. (see assumptions)
C:	exponent in estimation of empty weight fraction. (see assumptions)
SegOne:	weight fraction for segment one. (Non-dimensional)
SegTwo:	weight fraction for segment two. (Non-dimensional)
SegThree:	weight fraction for segment three. (Non-dimensional)
SegFour:	weight fraction for segment four. (Non-dimensional)
SegFive:	weight fraction for segment five. (Non-dimensional)
SegSix:	weight fraction for segment six. (Non-dimensional)
SegSeven:	weight fraction for segment seven. (Non-dimensional)
SegEight:	weight fraction for segment eight. (Non-dimensional)

Wx: weight fraction for entire mission, W_8/W_0. (Non-dimensional)
We: empty weight fraction of aircraft. (Non-dimensional)
Wf: fuel weight fraction needed for mission including trapped and reserve fuel. (Non-dimensional)
Wo: intial guess of take off gross weight. (Newtons)
CrewWeight: weight of the aircraft's crew. (Newtons)
PayloadWeight: weight of the aircraft's payload. (Newtons)
FuelWeight: weight of fuel including trapped and reserve fuel. (Newtons)
EmptyWeight: empty weight of aircraft. (Newtons)
Weight: final estimate of take off gross weight. (Newtons)

Assumptions

The range equations assume that the aircraft is flying straight and level and that angle of attack and velocity are constant.
Briguet Range Equation:

$$R = \frac{-V}{C} * \frac{Cl}{Cd} * \ln(\frac{Wi}{Wi+1})$$

The endurance equations assume that the aircraft is flying with a constant angle of attack. The equation is as follows:

$$E = \frac{1}{c} * \frac{Cl}{Cd} * \ln(\frac{Wi}{Wi+1})$$

The fuel fractions for the segments one, seven, and eight are assumed constant and were obtained from Nicolai's Fundamentals of Aircraft Design.
The fuel fraction for segment two is calculated using an equation from Nicolai.
The estimation of the empty weight fraction is obtained from a statistical fit to historical data found in Raymer's Aircraft Design.
TOGW is found using an iterative method based on Raymer's initial takeoff gross weight sizing technique.

Universal Conversion Module

This module performs single conversions based upon user input. The advantage to using this converter is that a greater variety of conversions can be performed.

Conversion takes place when a single variable is entered into one of the input variables. The module will determine the needed conversion based upon where the value has been entered. The converted value is displayed as the 'output' variable.

Variables

fps: entered as feet per second; converted to meters per second (feet)
ft: entered as feet; converted to meters
ftcu: entered as feet cubed; converted to meters cubed
ftsq: entered as feet squared; converted to meters squared
knots: entered as knots; converted to meters per second (knots)
lbf: entered as pounds force; converted to newtons

m: entered as meters; converted to feet
mcu: entered as meters cubed; converted to feet cubed
mpsf: entered as meters per second (feet); converted to feet per second
mpsk: entered as meters per second (knots); converted to knots
msq: entered as meters squared; converted to feet squared
n: entered as newtons; converted to pounds force
output: output of converted value

WEIGHTS CONVERSION MODULE

This module converts the three weights obtained from the weights module into pounds force.

Variables

cemptyWeight: empty weight of aircraft. (Pounds)
cfuelWeight: fuel weight needed to complete mission. (Pounds)
cWeight: take off gross weight. (Pounds)
emptyWeight: empty weight of aircraft. (Newtons)
fuelWeight: fuel weight needed to complete mission. (Newtons)
weight: take off gross weight. (Newtons)

The conversion used was $1lb_f = 4.48$ N.

AFT-BODY

The Aft-body of the aircraft is the rear section of the fuselage. It allows for control of the shoulder angles and radii of the aft section of the fuselage.

Variables

length: length of the aft section of the fuselage
rad1: radius 1 of the first cross section
rad2: radius 2 of the first cross section
rad3: radius 1 of the last cross section
rad4: radius 2 of the last cross section
a1: shoulder angle at first cross section
a2: shoulder angle at last cross section
num_xs: number of cross sections that are used to define the surface
geometry: shows the geometry of the aft section

Geometry Variables

Each component has a set of geometry variables.

Orientation

Rotation_x: rotates component around the x-axis
Rotation_y: rotates component around the y-axis
Rotation_z: rotates component around the z-axis
Translation_x: translates component along the x-axis
Translation_y: translates component along the y-axis
Translation_z: translates component along the z-axis

Appearance

red: shows the degree of red desired from 0-?
blue: shows the degree of blue desired from 0-?
green: shows the degree of green desired from 0-?
alpha:
show_hide:
fill_mode: shows how component is filled, ("solid"; "wire")

MID-SECTION

The mid-section is the middle section of the fuselage. It allows for the modifications of the radii of the mid-section.

Variables

length: the length of the mid section of the fuselage
rad1: radius 1 of the first cross section
rad2: radius 2 of the first cross section
rad3: radius 1 of the last cross section
rad4: radius 2 of the last cross section
num_xs: number of cross sections that are used to define the surface
geometry: shows geometry of the mid section

NOSE

The nose represents the front section of the fuselage. It allows manipulation of the dive angle, tip angle, shoulder angle and radii of the nose.

Variables

length: length of the nose of the aircraft
rad1: radius 1 of the last cross section
rad2: radius 2 of the last cross section
da: dive angle
ta: tip angle
sa: shoulder angle
num_xs: number of cross sections that are used to define the surface

WING

The wing represents the full wing of the aircraft. Several input variables allow for manipulation of the geometry of the wing.

Variables

span: distance from tip to tip of the wings
sweep: angle of the leading edge sweep
dihedral: dihedral angle of the wing
taper_ratio: ratio of the tip chord to the root chord
root_chord: length of the wing chord at the fuselage

twist: wing twist angle
AR: aspect ratio; span squared over area
rtc: airfoil thickness to chord ratio at the wing root
ttc: airfoil thickness to chord ratio at the wing tip
type: sets type of wing, (4 – wing; 5 – htail; 6 – vtail; 7 – canard)
num_xs: number of cross sections used to define the surface
right_wing: sets which half of the wing to be displayed, (0 – right wing; 1 – left wing)

Carpet Plot Generator

For description of all variables and assumptions see Mission Perfomance Module. The TOGWs for the carpet plot are being generated from this module although it is not seen in the excel spreadsheet.

Variables

T/W: Thrust to Weight ratio; (the thrust of the engine divided by the total weight of the aircraft)
W/S: Wing Loading; (the weight of the aircraft divided by the total wing area).
C_{LMAX}: Maximum lift coefficient.
S_{TOFL}: The takeoff field length distance.
P_S: Specific excess power.
ρ: Density
V: Aircraft velocity
q: Dynamic pressure: ($q = .5*\rho*V^2$)
C_{DO}: Drag coefficient at zero lift
n: Number of g-forces pulled.
k: $k = 1/(\Pi*AR*e)$;
V_{APP}: Approach velocity of the aircraft.

Assumptions

Constraint equations used for Carpet Plot

Takeoff Equation: $T/W = (37.7 * (W/S)) / (C_{LMAX} * S_{TOFL})$
Specific Excess Power: $PS = V [(T/W) – ((q * C_{DO}) / (W/S)) – (n^2 * (k/q) * (W/S))]$
Landing: $W/S = .5 * C_{LMAX} * \rho * V_{APP}^2$