```
This is a draft chapter for the book
System Design, Modeling, and Simulation
Claudius Ptolemaeus, Editor,
Draft Version 0.05, January 7, 2012
```

C Creating Web Pages

Christopher X. Brooks and Edward A. Lee

Contents

Ptolemy II includes a flexible mechanism for creating web pages from models. These pages provide easy access and documentation for models that archives both the structure of the models and the results of executing the models.

More interestingly, the mechanism is extensible and customizable. You can associate hyperlinks or actions defined in JavaScript[*] with icons in a model. The customization

[*]The export to web facility uses JavaScript to display the parameters of actors.

can be done for individual icons in a model, for sets of icons in a model, or for an entire executable modeling tool (using the configuration mechanism in Ptolemy II, which allows creation of customized modeling tools).

In Vergil, assuming you are running the default full configuration, when you export a model to a web page, you get a page that displays parameters of actors when the mouse moves over the icon of the actor, hyperlinks that display the contents of composite actors, and hyperlinks that display plots that result from a run of the model. These default behaviors can all be changed, as explained in this chapter.

C.1 Export to Web

To export a model to the web, select [`File->Export->Export to Web`], as shown in Figure C.1. This will open a dialog that enables you to select a directory (or create new directory). That directory will be populated with a file called index.html, some image files, and some subdirectories. One image file shows whatever portion of the model is visible when you perform the export. In addition, there will be an image file for each open plot window. Moreover, there will be one subdirectory for each composite actor that is open at the time of export.

The export dialog offers a number of options, as follows.

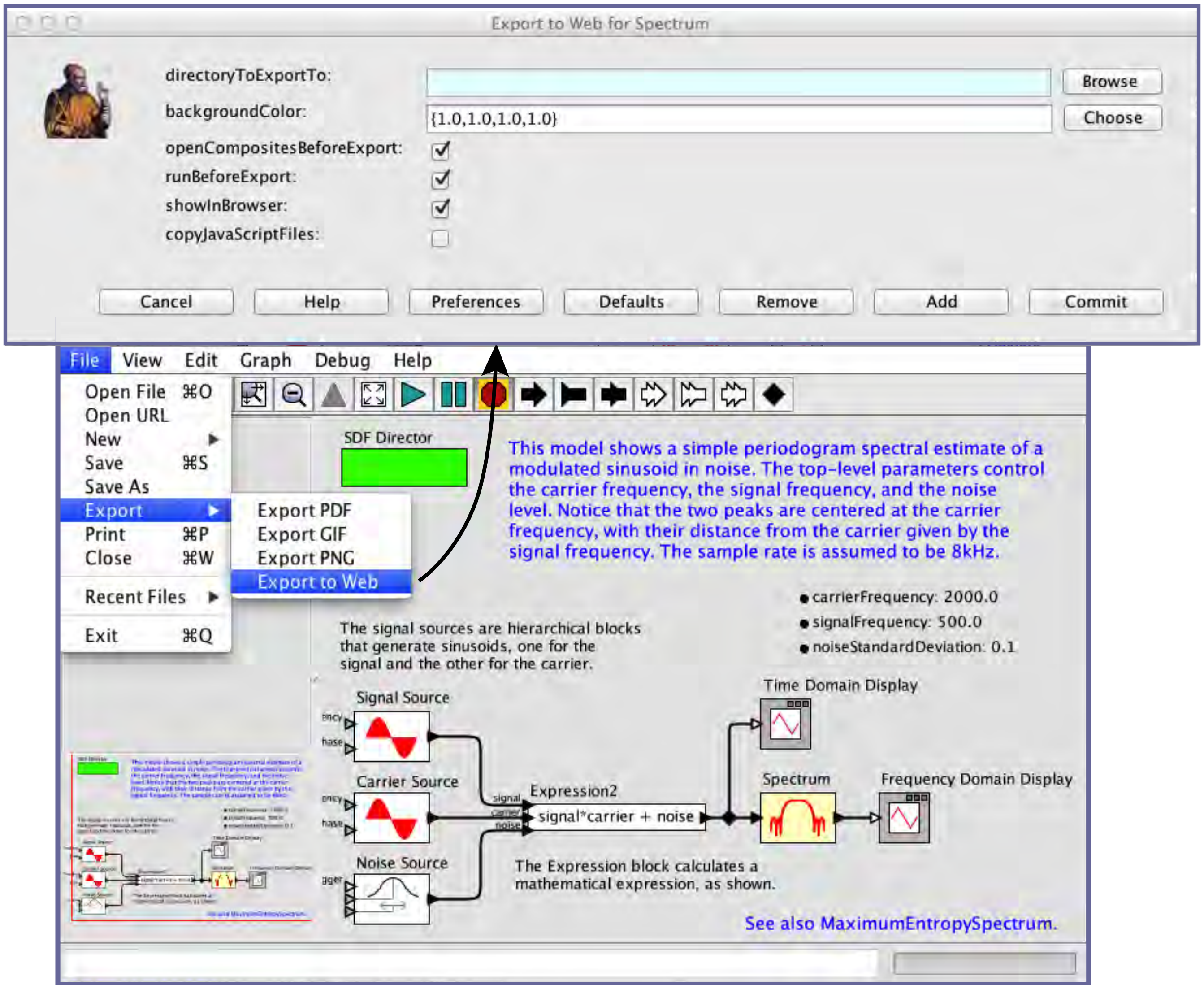
- *directoryToExportTo*: The directory into which to put the web files. If no directory is given, then a new directory is created in the same directory that stores the MoML file for the model. The new directory will have the same name as the model, with any special characters replaced so that the name is a legal file name.
- *backgroundColor*: The background color to use for the image model. By default, this is blank, which means that the image will use whatever background color the model has (typically gray). But white is a good option for web pages, as shown in Figure C.1.
- *openCompositesBeforeExport*: If this is true, then composite actors in the model are opened before exporting. Each composite actor will also be exported into its own web page, and hyperlinks will be created in the top-level image to allow navigation to those web pages in the browser. If you want only some of the composite actors to be included

By default, JavaScript may be disabled in your web browser. If JavaScript is disabled in your browser, then the web page will include the text "Your browser does not support Javascript...".

The solution is to enable JavaScript. See http://support.microsoft.com/gp/howtoscript for details.

in the export, then you can manually open the ones you want. Only open windows will be included in the export.

- *runBeforeExport*: If this is true, then the model is run before exporting. This has the side effect of opening plot windows, which will therefore be included in the export. If you want only some of the plot windows to be included in the export, then you can run the model and close the ones you don't want. Only open plot windows will be included in the export.
- *showInBrowser*: If this is true, then once the export is complete, the resulting web page will be displayed in your default browser.



directoryToExportTo:
backgroundColor: {1.0,1.0,1.0,1.0}
openCompositesBeforeExport: ✓
runBeforeExport: ✓
showInBrowser: ✓
copyJavaScriptFiles:

This model shows a simple periodogram spectral estimate of a modulated sinusoid in noise. The top-level parameters control the carrier frequency, the signal frequency, and the noise level. Notice that the two peaks are centered at the carrier frequency, with their distance from the carrier given by the signal frequency. The sample rate is assumed to be 8kHz.

- carrierFrequency: 2000.0
- signalFrequency: 500.0
- noiseStandardDeviation: 0.1

The signal sources are hierarchical blocks that generate sinusoids, one for the signal and the other for the carrier.

The Expression block calculates a mathematical expression, as shown.

See also MaximumEntropySpectrum.

Figure C.1: Menu command to export a model to the web.

- *copyJavaScriptFiles*: If this is true, then additional files will be included in the exported page so that the page does not depend on any files from the internet. The files include JavaScript code and image files that affect the interactivity and look-and-feel of the web page. By default, these files are not included and are instead retrieved by the web page from `http://ptolemy.org`.

For the example shown in Figure C.1, the resulting web page is displayed by the Safari web browser as shown in Figure C.2. This page exhibits some of the default behavior of export to web. A title for the page is shown at the top; this is, by default, the name of the model. Moreover, in the image shown in Figure C.2, the mouse is hovering over the Signal Source actor, which is outlined; when the mouse hovers over an actor, then by default, a table with the parameter values of the actor is displayed at the bottom of the page, as shown in Figure C.2.

In Figure C.1, *openCompositesBeforeExport* and *runBeforeExport* are both set to true (the default is false). Hence, the model is executed before the export, opening plot windows. Hyperlinks to the plot windows are created, and clicking on a plot actor on the web page image will display the plot, as shown in Figure C.3. In addition, the composite actors in the model, Signal Source, Carrier Source, and Spectrum, all have hyperlinks to a page showing the inner structure of the composite.

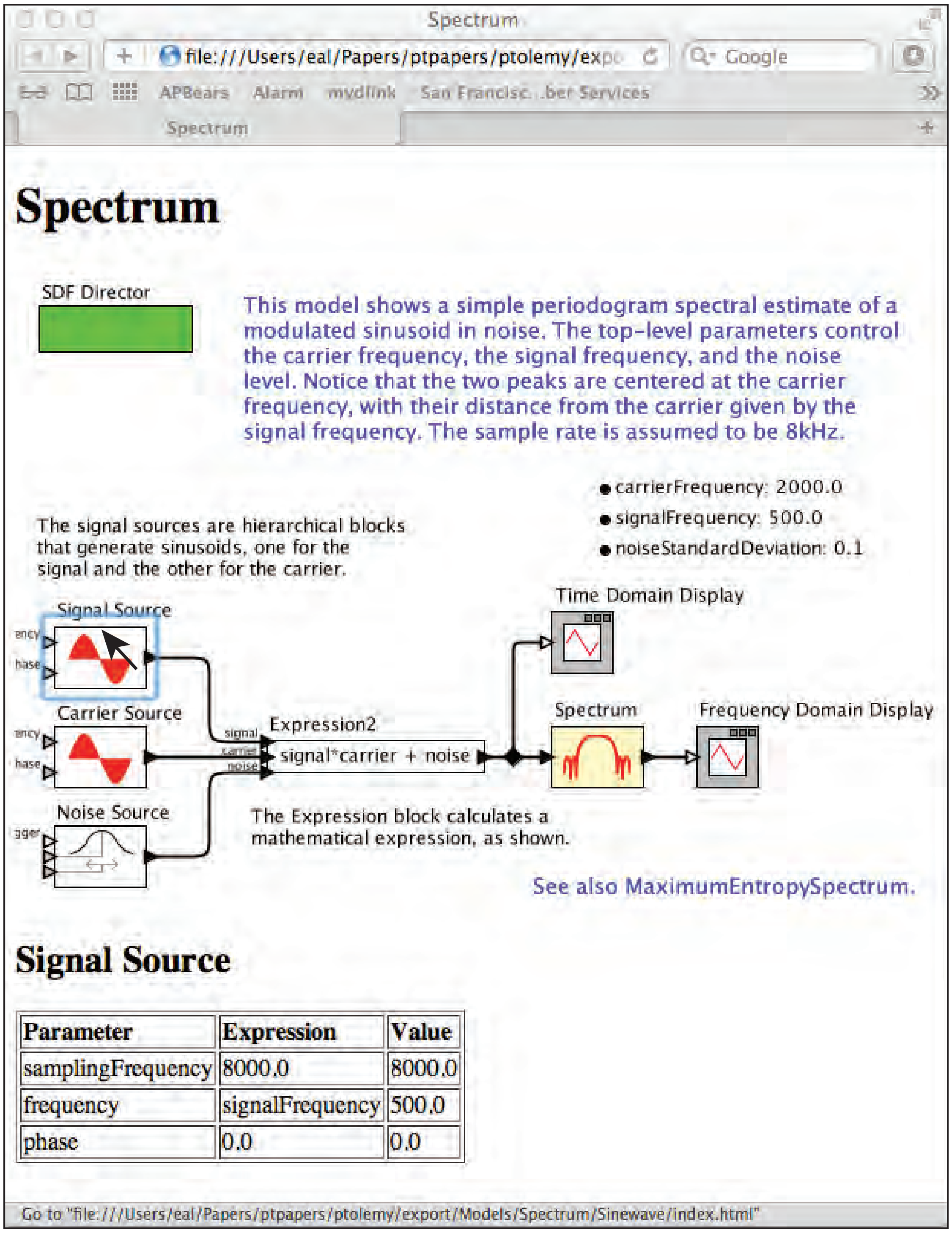
All these functions can be customized, as we will explain next.

C.2 Customizing the Export

As shown in Figure C.4, the `Utilities->WebExport` library provides attributes that, when dragged into a model, customize the exported web page. This section explains each of the items in this library, shown on the left in the figure. In each case, you can right click (or control click on a Mac) and select `Get Documentation` to view documentation about the attribute. The attributes are related to one another as shown in the UML class diagram in Figure C.5.

C.2.1 HTMLText: Adding Text to Web Pages

The *HTMLText* attribute inserts HTML text into the page exported by Export to Web. Drag the attribute onto the background of a model, as shown in Figure C.4, and double



Spectrum

This model shows a simple periodogram spectral estimate of a modulated sinusoid in noise. The top-level parameters control the carrier frequency, the signal frequency, and the noise level. Notice that the two peaks are centered at the carrier frequency, with their distance from the carrier given by the signal frequency. The sample rate is assumed to be 8kHz.

- carrierFrequency: 2000.0
- signalFrequency: 500.0
- noiseStandardDeviation: 0.1

The signal sources are hierarchical blocks that generate sinusoids, one for the signal and the other for the carrier.

The Expression block calculates a mathematical expression, as shown.

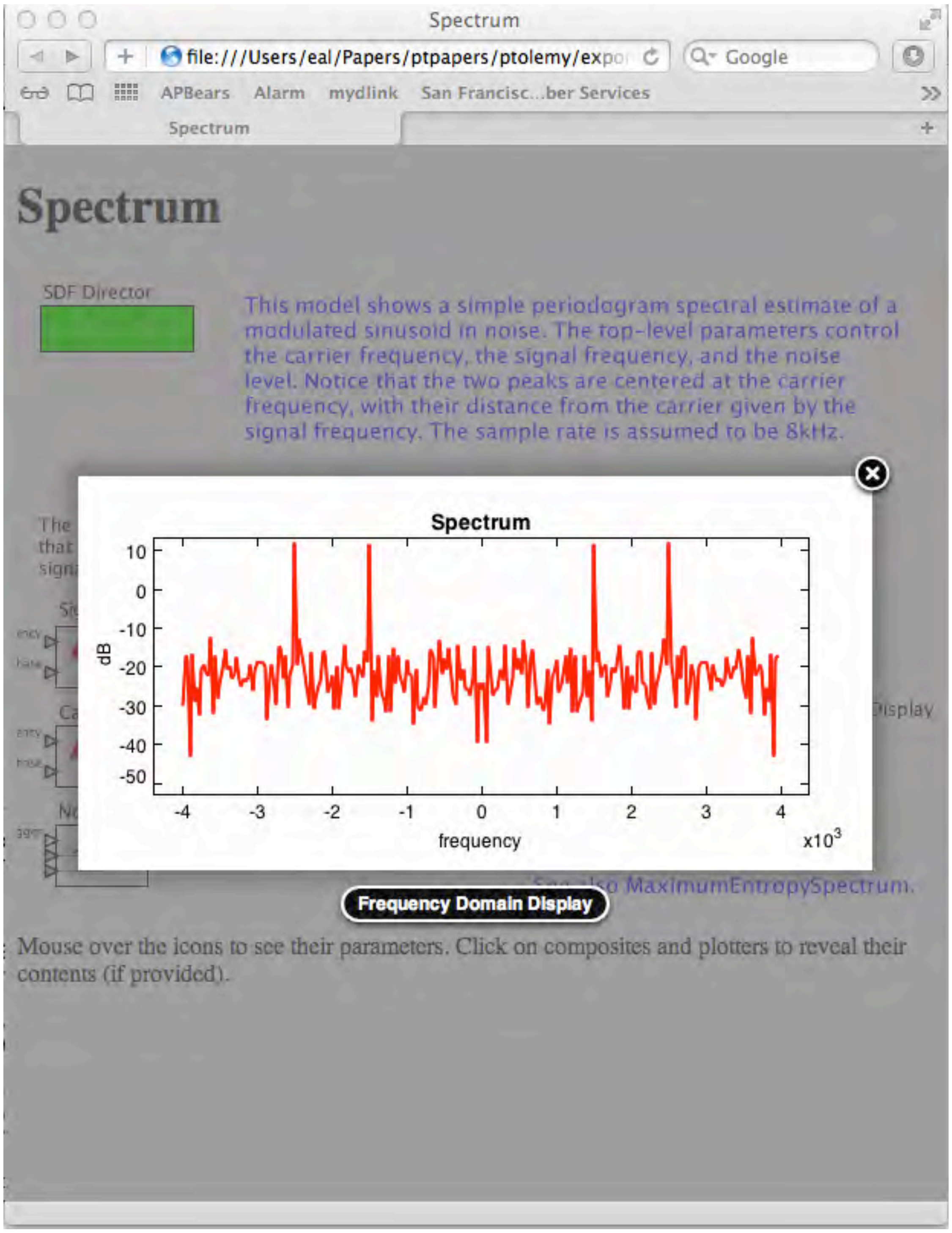
See also MaximumEntropySpectrum.

Signal Source

Parameter	Expression	Value
samplingFrequency	8000.0	8000.0
frequency	signalFrequency	500.0
phase	0.0	0.0

Go to "file:///Users/eal/Papers/ptpapers/ptolemy/export/Models/Spectrum/Sinewave/index.html"

Figure C.2: Web page exported from the model shown in Figure C.1.



This model shows a simple periodogram spectral estimate of a modulated sinusoid in noise. The top-level parameters control the carrier frequency, the signal frequency, and the noise level. Notice that the two peaks are centered at the carrier frequency, with their distance from the carrier given by the signal frequency. The sample rate is assumed to be 8kHz.

See also MaximumEntropySpectrum.

Mouse over the icons to see their parameters. Click on composites and plotters to reveal their contents (if provided).

Figure C.3: Clicking on the Frequency Domain Display actor in Figure C.2 displays the plot generated by running the model.

click on its icon to specify the HTML text to export. To specify the text to include in the HTML page, double click on the icon for the *HTMLText* attribute (which by default is a textual icon reading “Content for Export to Web”), as shown in Figure C.6. You can type in the text to export, including any HTML content you like such as hyperlinks and formatting directives. The web page including the text specified in Figure C.6 is shown in Figure C.7.

By default, this text will be placed before the image for the model, but you can change the position by setting the *textPosition* parameter, as shown in Figure C.6. In that figure, you

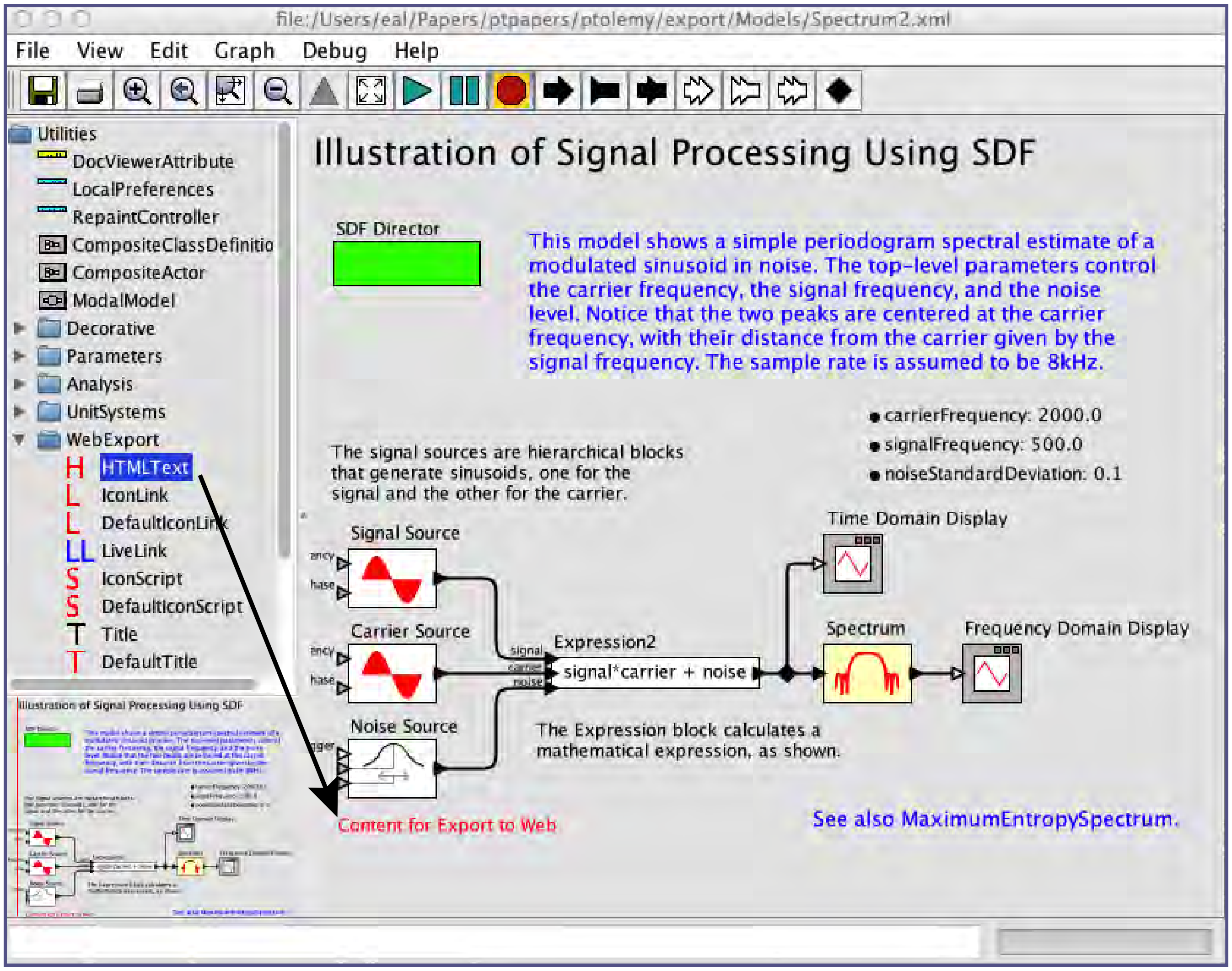


Illustration of Signal Processing Using SDF

This model shows a simple periodogram spectral estimate of a modulated sinusoid in noise. The top-level parameters control the carrier frequency, the signal frequency, and the noise level. Notice that the two peaks are centered at the carrier frequency, with their distance from the carrier given by the signal frequency. The sample rate is assumed to be 8kHz.

- carrierFrequency: 2000.0
- signalFrequency: 500.0
- noiseStandardDeviation: 0.1

The signal sources are hierarchical blocks that generate sinusoids, one for the signal and the other for the carrier.

The Expression block calculates a mathematical expression, as shown.

Content for Export to Web

See also MaximumEntropySpectrum.

Figure C.4: The `Utilities->Web Export` library provides attributes that, when dragged into a model, customize the exported web page.

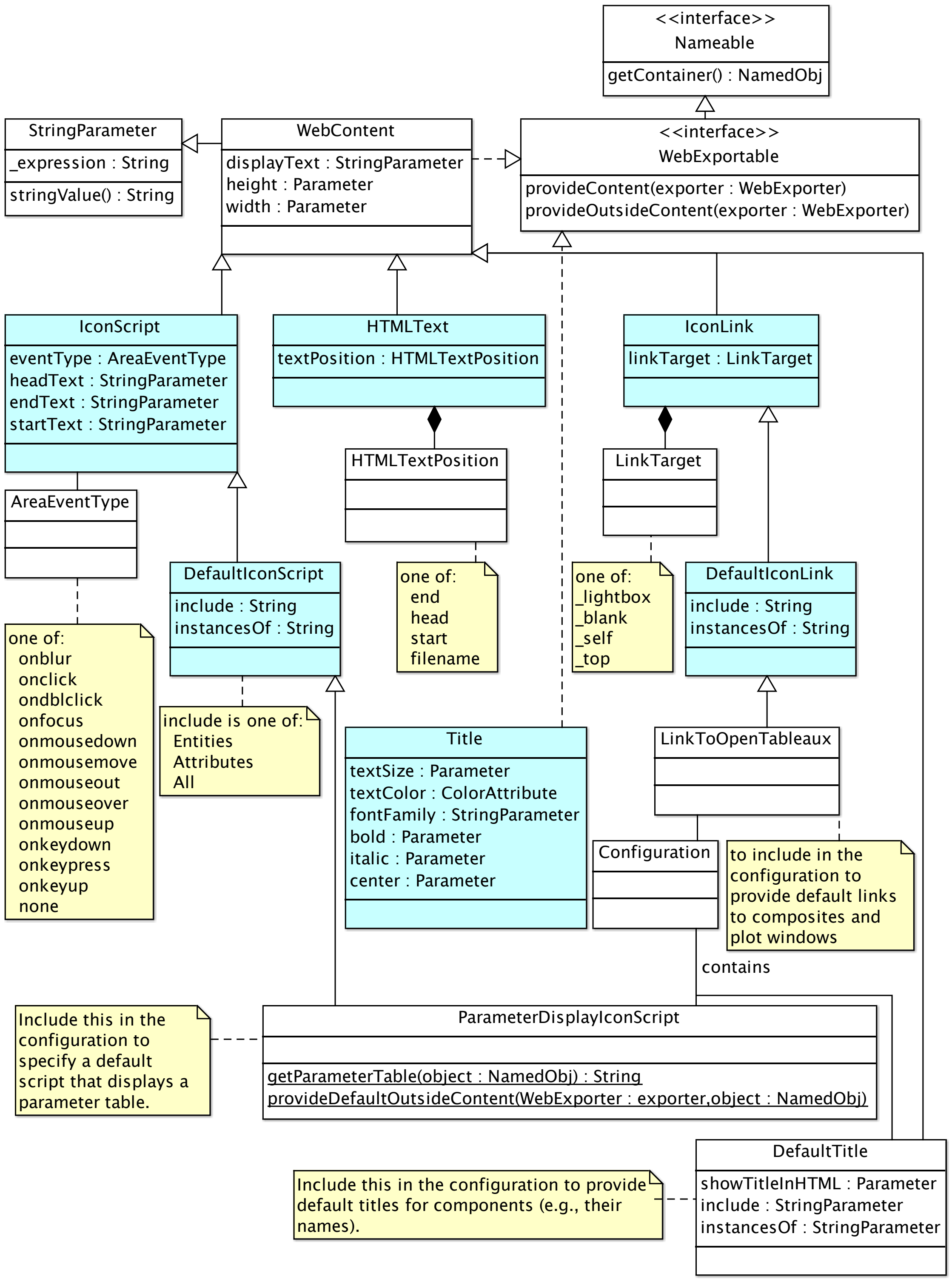
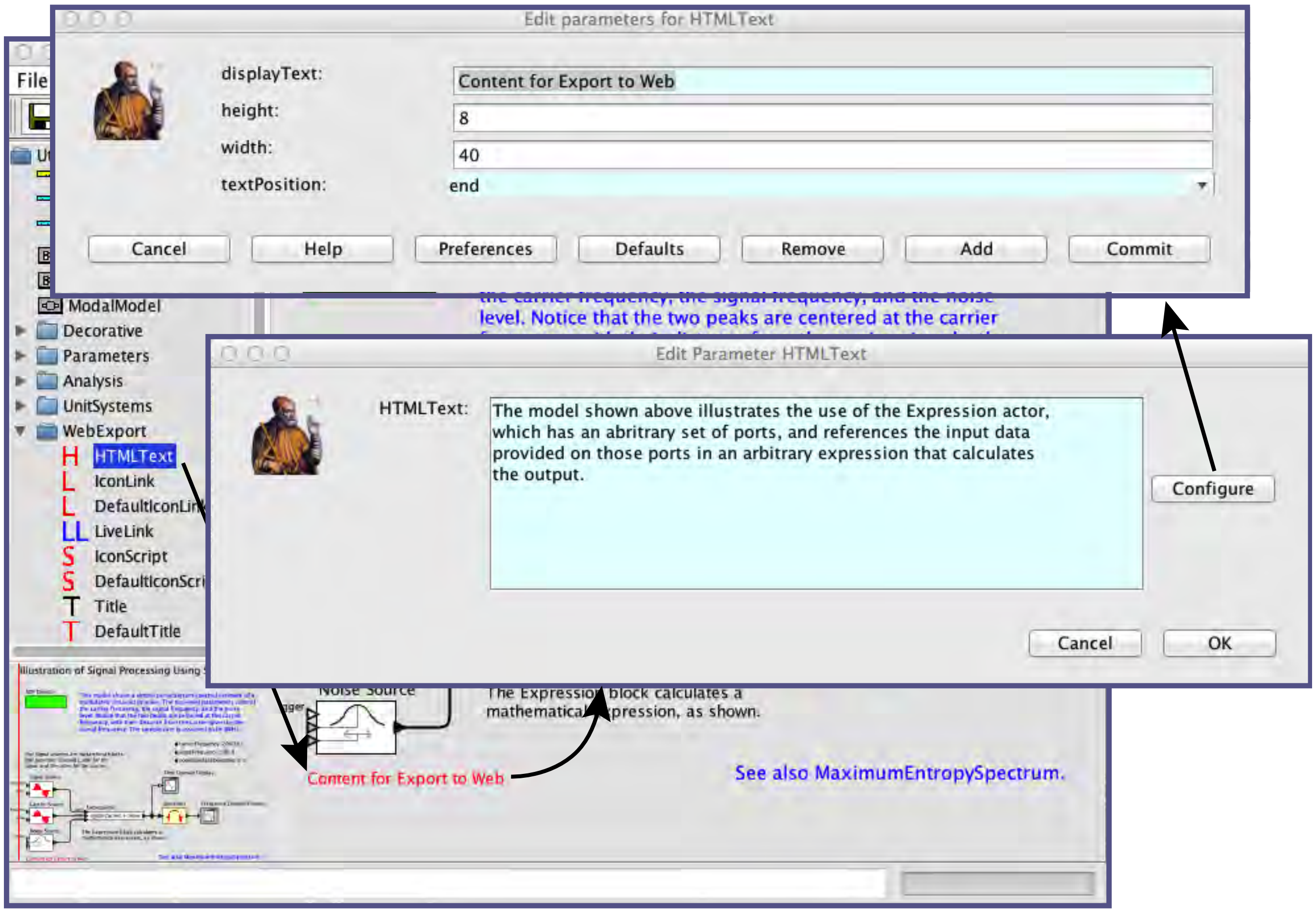

<<interface>>
Nameable
getContainer() : NamedObj
StringParameter
_expression : String
stringValue() : String
WebContent
displayText : StringParameter
height : Parameter
width : Parameter
<<interface>>
WebExportable
provideContent(exporter : WebExporter)
provideOutsideContent(exporter : WebExporter)
IconScript
eventType : AreaEventType
headText : StringParameter
endText : StringParameter
startText : StringParameter
HTMLText
textPosition : HTMLTextPosition
IconLink
linkTarget : LinkTarget
HTMLTextPosition
LinkTarget
AreaEventType
DefaultIconScript
include : String
instancesOf : String
one of:
end
head
start
filename
one of:
_lightbox
_blank
_self
_top
DefaultIconLink
include : String
instancesOf : String
one of:
onblur
onclick
ondblclick
onfocus
onmousedown
onmousemove
onmouseout
onmouseover
onmouseup
onkeydown
onkeypress
onkeyup
none
include is one of:
Entities
Attributes
All
Title
textSize : Parameter
textColor : ColorAttribute
fontFamily : StringParameter
bold : Parameter
italic : Parameter
center : Parameter
LinkToOpenTableaux
Configuration
to include in the configuration to provide default links to composites and plot windows
contains
Include this in the configuration to specify a default script that displays a parameter table.
ParameterDisplayIconScript
getParameterTable(object : NamedObj) : String
provideDefaultOutsideContent(WebExporter : exporter,object : NamedObj)
Include this in the configuration to provide default titles for components (e.g., their names).
DefaultTitle
showTitleInHTML : Parameter
include : StringParameter
instancesOf : StringParameter

Figure C.5: UML class diagram for the attributes for customization of exported web pages. The shaded attributes are the most commonly used in models.



displayText: Content for Export to Web
height: 8
width: 40
textPosition: end

level. Notice that the two peaks are centered at the carrier

HTMLText: The model shown above illustrates the use of the Expression actor, which has an abritrary set of ports, and references the input data provided on those ports in an arbitrary expression that calculates the output.

The Expression block calculates a mathematical expression, as shown.

Figure C.6: Dialog for customizing HTML text to include in an exported web page.

can see that the *HTMLText* attribute is configured to put the text at the end of the HTML file, which explains why that text appears at the bottom of the page in Figure C.7.

The *HTMLText* attribute has several options for customizing it:

- *displayText*: This parameter determines what shows up in the model itself. By default, this is the text "Content for Export to Web." Notice that this text also appears in the exported web page in Figure C.7, which is a bit odd. This text is not an interesting part of the model; it is simply a placeholder for an attribute that customizes the exported web page. If you do not want this attribute to show up in an exported web page, you can simply move the attribute out of the field of view before doing the export. Alternatively, you can set *displayText* to an empty string, but this technique has the disadvantage of making it slightly more difficult to find the attribute to edit or customize the exported text. In Figure C.8, the *displayText* has been set to the empty string. The *HTMLText* parameter is still present and can be selected (the small yellow box that is barely visible at the lower left in the figure is the *HTMLText* parameter), but since there is no visible icon, it is hard to find. An easier way to edit the *HTMLText* parameter is to right click on the background of the model, as shown in Figure C.8. The *HTMLText* parameter appears as a parameter of the model, along with whatever other parameters have been defined in the model.
- *height*: The height of the editing box for specifying the text to export. If you change this value, close and re-open the dialog for the change to take effect.
- *width*: The width of the editing box for specifying the text to export. If you change this value, close and re-open the dialog for the change to take effect.
- *textPosition*: As mentioned above, this parameter determines the position of the exported text. The built-in options are end, start, and head. Choosing "end" puts the text after the exported model image. Choosing "start" puts the text before the exported model image. Choosing "head" puts the text in the header section of the HTML page. If you specify any other value for *textPosition*, then that value is assumed to be the name of a file, and a file with that name is created in the same directory as the export. The specified text is then exported to that file.

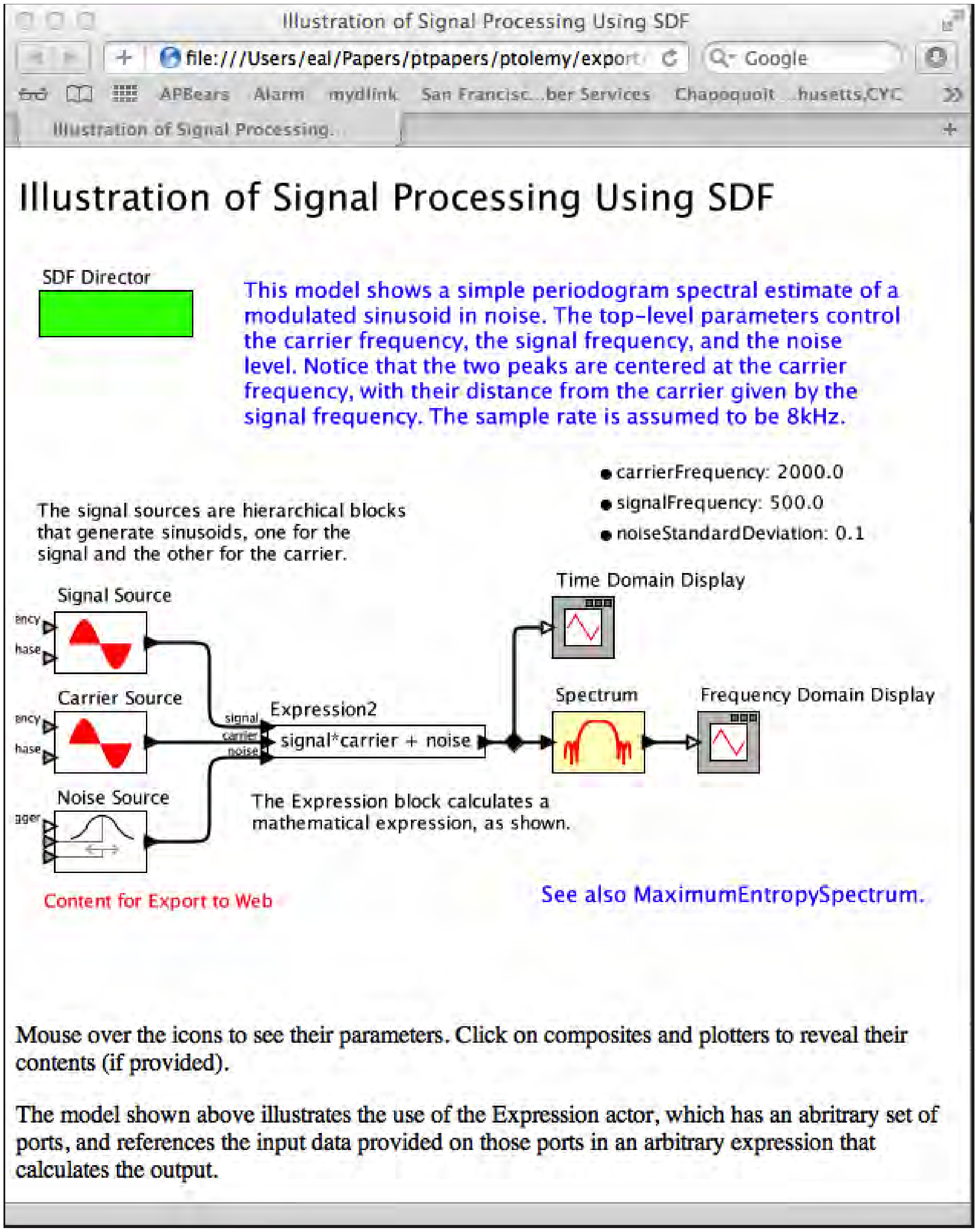


Illustration of Signal Processing Using SDF

This model shows a simple periodogram spectral estimate of a modulated sinusoid in noise. The top-level parameters control the carrier frequency, the signal frequency, and the noise level. Notice that the two peaks are centered at the carrier frequency, with their distance from the carrier given by the signal frequency. The sample rate is assumed to be 8kHz.

- carrierFrequency: 2000.0
- signalFrequency: 500.0
- noiseStandardDeviation: 0.1

The signal sources are hierarchical blocks that generate sinusoids, one for the signal and the other for the carrier.

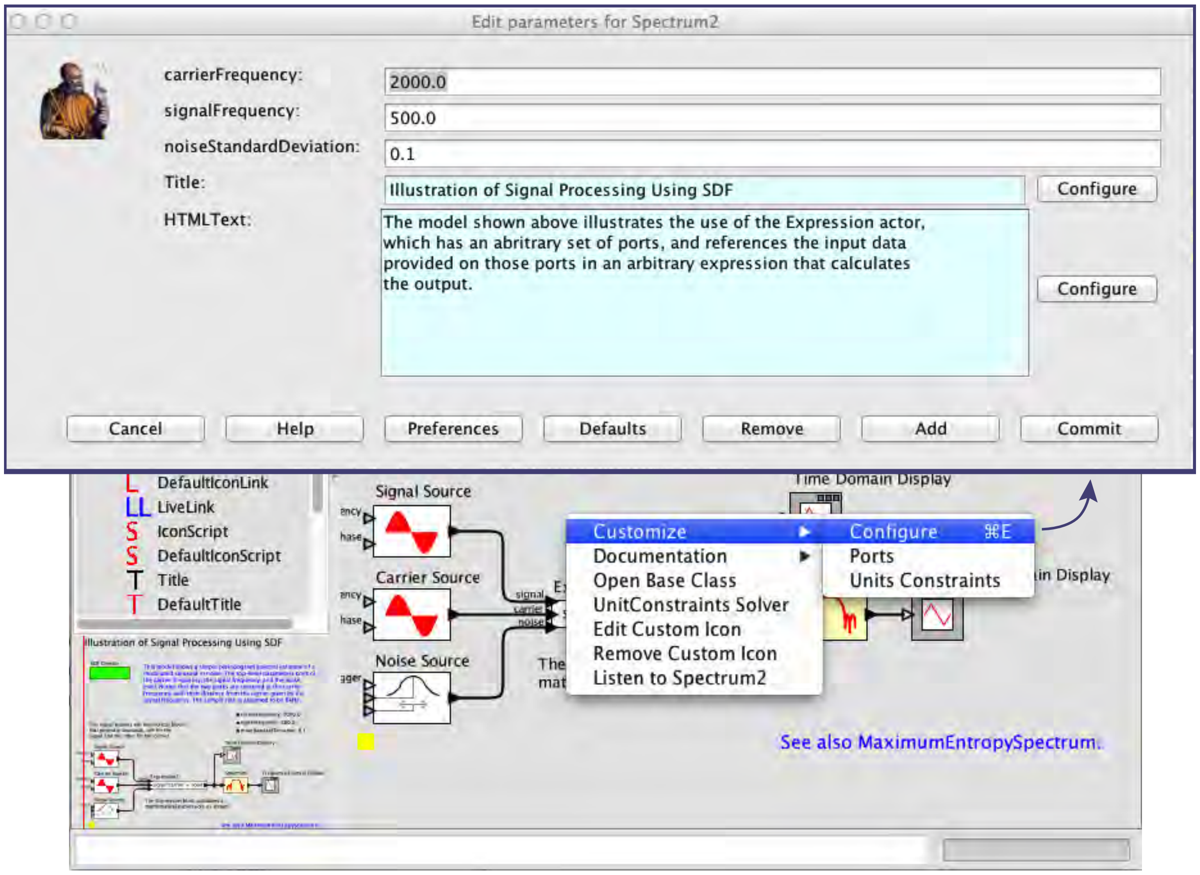
The Expression block calculates a mathematical expression, as shown.

See also MaximumEntropySpectrum.

Mouse over the icons to see their parameters. Click on composites and plotters to reveal their contents (if provided).

The model shown above illustrates the use of the Expression actor, which has an abritrary set of ports, and references the input data provided on those ports in an arbitrary expression that calculates the output.

Figure C.7: Page resulting from inserting an HTMLText attribute into the example of Figure C.4 and configuring it as shown in Figure C.6.

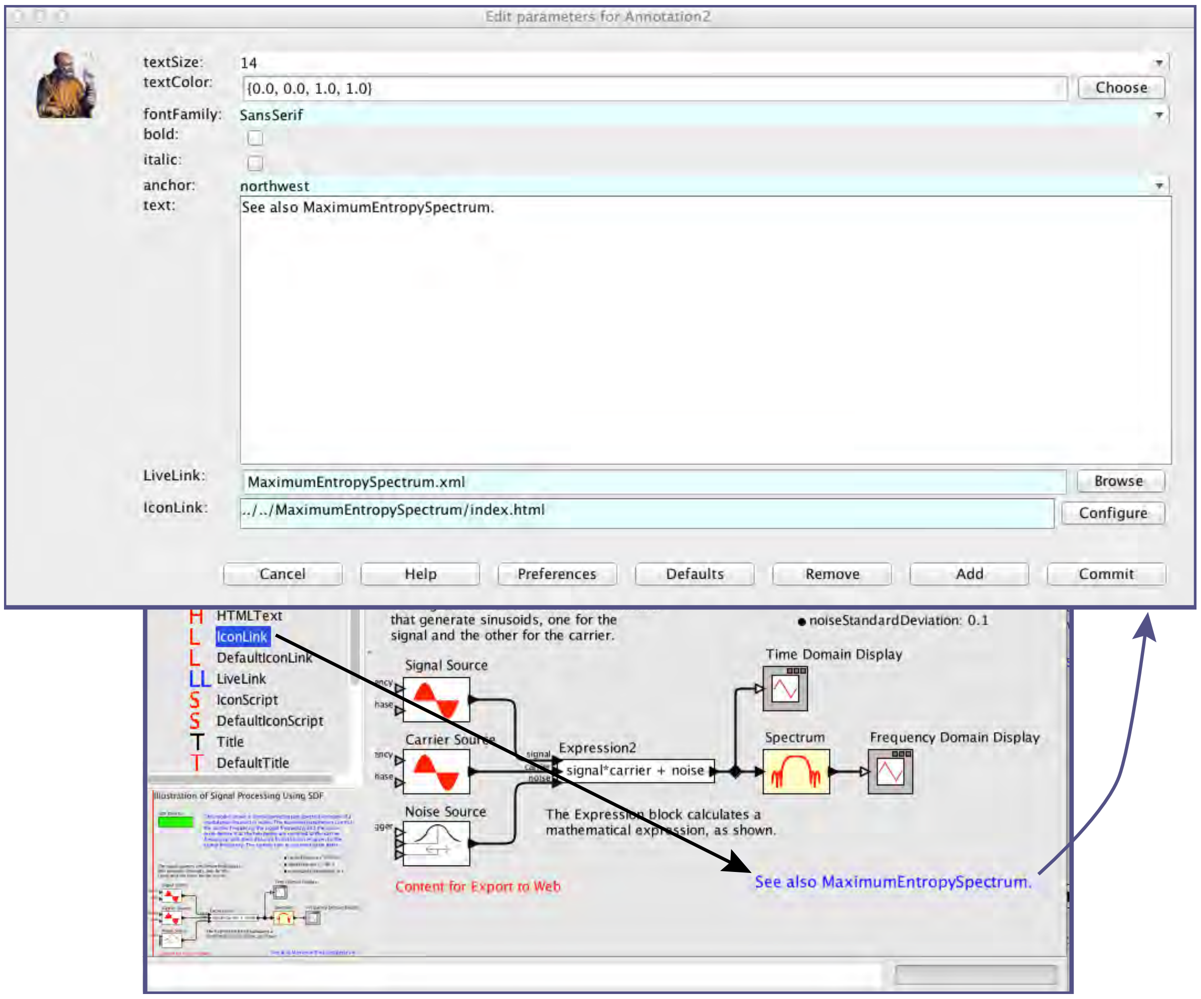

Edit parameters for Spectrum2

carrierFrequency: 2000.0
signalFrequency: 500.0
noiseStandardDeviation: 0.1
Title: Illustration of Signal Processing Using SDF
HTMLText: The model shown above illustrates the use of the Expression actor, which has an abritrary set of ports, and references the input data provided on those ports in an arbitrary expression that calculates the output.

Figure C.8: The *HTMLText* attribute can be hidden by setting its *displayText* parameter to the empty string. It can still be edited by right clicking on the background of the model. Notice that *HTMLText* appears in the list of model parameters.

C.2.2 IconLink: Specifying Hyperlinks for Icons

The *IconLink* parameter shown in the `Utilities->WebExport` library can be used to specify a hyperlink for an icon in the model. To use it, drag it from the library onto the icon that you would like to have the link. In the example of Figure C.9, we have done such a drag onto the text annotation shown at the lower right that reads "See also MaximumEntropySpectrum."



Edit parameters for Annotation2

textSize: 14
textColor: {0.0, 0.0, 1.0, 1.0}
fontFamily: SansSerif
bold: ☐
italic: ☐
anchor: northwest
text: See also MaximumEntropySpectrum.
LiveLink: MaximumEntropySpectrum.xml
IconLink: ../../MaximumEntropySpectrum/index.html

that generate sinusoids, one for the signal and the other for the carrier.

The Expression block calculates a mathematical expression, as shown.

Figure C.9: The *IconLink* attribute can be dragged onto an icon. The object onto which it is dragged acquires a parameter that can be used to specify a web page to link to from that icon when the model is exported to the web. Here, the exported web page will have a link on this icon to "../../MaximumEntropySpectrum/index.html".

Double clicking on the text annotation reveals an *IconLink* parameter that can be set to a URL. The exported web page will include a hyperlink from the text annotation to that specified page.

The *IconLink* parameter can be customized (click on the Configure button at the lower right of the dialog in Figure C.9). The parameters *displayText*, *width*, and *height* are the same as those for *HTMLText*, described above. A new parameter is *linkTarget*. This has four allowed values:

- `_lightbox`: Display the link in a pop-up lightbox.
- `_blank` (the default): Display the link in new blank window of the browser.
- `_self`: Display the link in the same window, replacing the current page or frame.
- `_top`: Display the link in the same window, replacing the current page.

An example of the lightbox display is the plot shown in Figure C.3.

In addition, if the *linkTarget* parameter is given any other value, then that value is assumed to be the name of a frame in the web page, and that frame becomes the target.

C.2.3 DefaultIconLink: Default Hyperlinks for Icons

The *DefaultIconLink* parameter shown in the `Utilities->WebExport` library on the left in Figure C.4 can be used to specify a default hyperlink for any icon in a model that does not contain an *IconLink*. In addition to the parameters of *IconLink*, *DefaultIconLink* has two additional parameters:

- *include*: This parameter can be used to restrict icons to which the default applies. Specifically, the defaults may be specified for icons for attributes, entities, or both.
- *instancesOf*: If non-empty, this attribute specifies a class name. Only entities or attributes (depending on the *include* parameter) implementing the specified class will be assigned the default link.

C.2.4 LiveLink: Hyperlinks in Vergil

Although not directly related to web page exporting, the *LiveLink* parameter is included in the library because it works particularly well with *IconLink*. In particular, if you drop an instance of *LiveLink* onto an icon, then you can specify a file or URL to be opened when a user double clicks on the icon in Vergil (vs. clicking on an icon in a browser showing the exported web page). This does not automatically result in a hyperlink in an exported web page because typically a model will want to specify a different file or URL to be opened by Vergil than what would be opened by a browser. Vergil can open and display MoML files, for example, whereas a browser will simply display the XML content.

Example C.1: Notice that in Figure C.9, the annotation that reads "See also MaximumEntropySpectrum" contains both an instance of *IconLink* and an instance of *LiveLink*. The *LiveLink* references a MoML file, MaximumEntropySpectrum.xml, assumed to be stored in the same directory as the Spectrum model. The *IconLink* parameter, however, references an HTML file. That reference assumes that both Spectrum and MaximumEntropySpectrum will have exported web pages, and that the relative locations of these pages on a server are such that the specified path will provide a link to the HTML file for the MaximumEntropySpectrum.

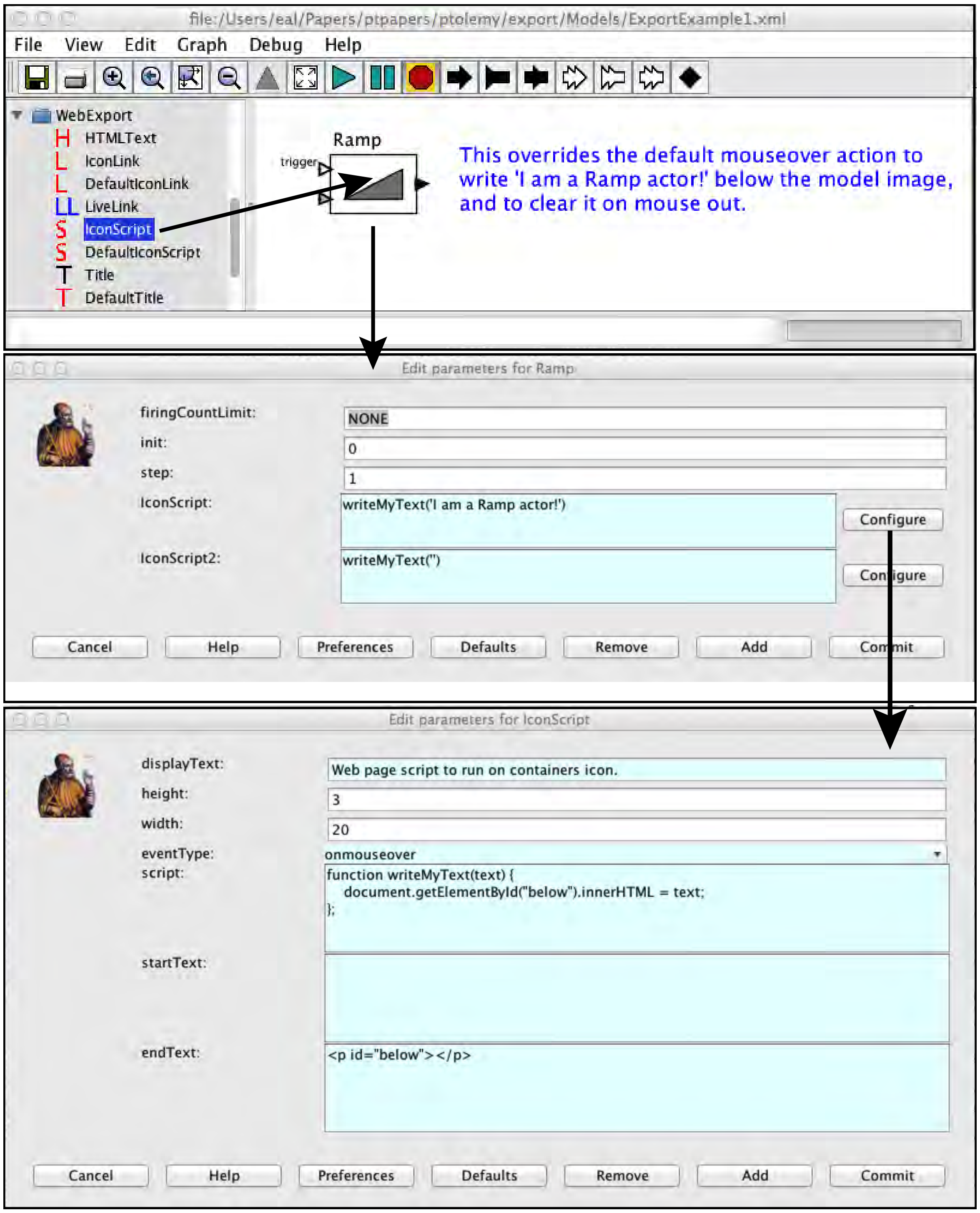
Assuming all files are arranged appropriately in the file system, the Vergil hyperlink and the web page hyperlink will do essentially the same thing. They will each open the referenced model, MaximumEntropySpectrum. But Vergil will open it in Vergil, whereas a browser will open its exported web page in the browser.

C.2.5 IconScript: Scripted Actions for Icons

The *IconScript* parameter is used to provide a scripted action associated with an icon in a model. Specifically, an action can be associated with mouse movement over the icon, mouse clicks, or keyboard actions. The action is specified as a JavaScript script.

Example C.2: An example using *IconScript* is shown in Figures C.10 and C.11. In this example, two instances of the *IconScript* parameter have been dragged onto the icon for a Ramp actor. These parameters have been customized to display "I am a Ramp actor!" when the mouse enters the icon on the exported web page, and to clear the display when the mouse leaves the icon, as shown in Figure C.11.

The way that this works is that the value of the first *IconScript* parameter is the JavaScript code:



This overrides the default mouseover action to write 'I am a Ramp actor!' below the model image, and to clear it on mouse out.

firingCountLimit: NONE
init: 0
step: 1
IconScript: writeMyText('I am a Ramp actor!')
IconScript2: writeMyText('')

displayText: Web page script to run on containers icon.
height: 3
width: 20
eventType: onmouseover
script:
```
function writeMyText(text) {
    document.getElementById("below").innerHTML = text;
};
```
startText:
endText: `<p id="below"></p>`

Figure C.10: Here, two instances of the *IconScript* parameter have been dragged onto the icon for a Ramp actor. These parameters have been customized to display "I am a Ramp actor!" when the mouse enters the icon on the exported web page, and to clear the display when the mouse leaves the icon, as shown in Figure C.11.

```
writeMyText('I am a Ramp actor!')
```

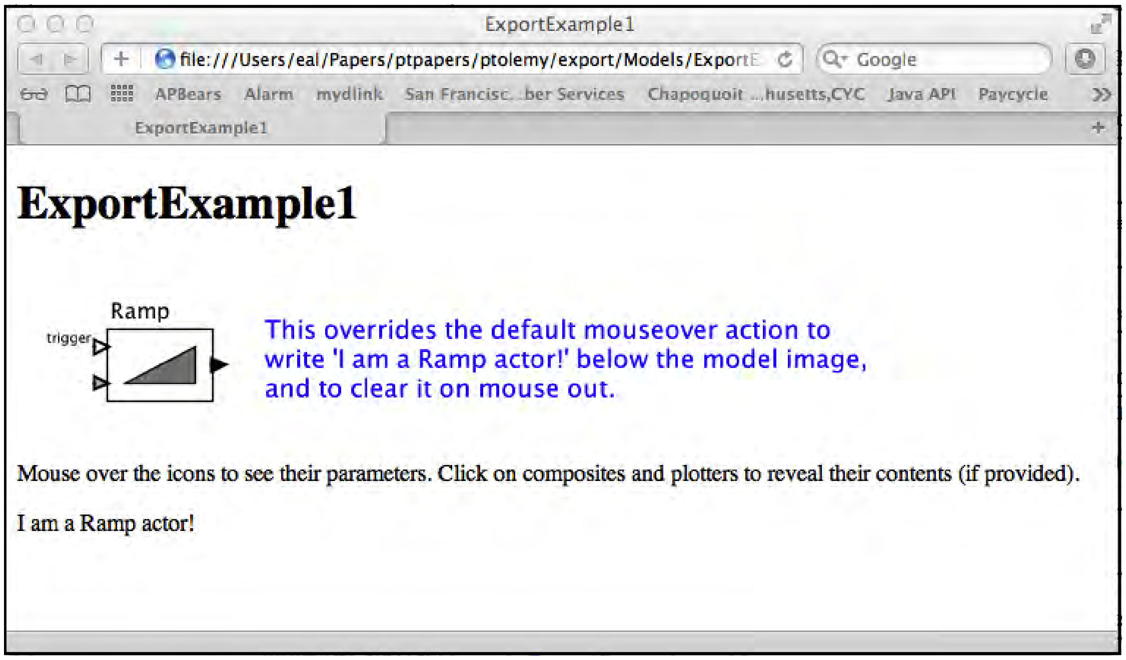
This invokes a JavaScript procedure `writeMyText`, which is defined in the *script* parameter of the *IconScript* parameter to be:

```
function writeMyText(text) {
    document.getElementById("below").innerHTML = text;
};
```

This procedure takes one argument, `text`, and writes the value of this argument into the `innerHTML` field of the element with ID `below`. That element is defined in the *endText* parameter of the *IconScript* parameter as follows:

```
<p id="below"></p>
```

This is an HTML paragraph with ID `below`. This paragraph will be inserted into the exported web page below the model image. Finally, the *eventType* parameter of the *IconScript* is set to `onmouseover`, which results in the script being invoked when the mouse enters the area of the web page displaying the Ramp icon, as shown in Figure C.11.


ExportExample1
file:///Users/eal/Papers/ptpapers/ptolemy/export/Models/ExportE
Google
APBears
Alarm
mydlink
San Francisc...ber Services
Chapoquoit ...husetts,CYC
Java API
Paycycle
ExportExample1
Ramp
trigger

ExportExample1

This overrides the default mouseover action to write 'I am a Ramp actor!' below the model image, and to clear it on mouse out.

Mouse over the icons to see their parameters. Click on composites and plotters to reveal their contents (if provided).

I am a Ramp actor!

Figure C.11: Web page exported by the model in Figure C.11, shown with the mouse lingering over the Ramp icon.

The second instance of *IconScript*, named IconScript2, specifies the following script:

```
writeMyText('')
```

This uses the same JavaScript procedure to clear the display when mouse exits the Ramp icon. The *eventType* parameter of this second *IconScript* is set to `onmouseout`.

If multiple instances of *IconScript* have exactly the same *script* parameter, then the value of that parameter will be included only once in the head section of the exported HTML page. Hence, the value of the *script* parameter is required JavaScript definitions. The web page exporter is smart enough to include those definitions only once if they are required at least once in the model.

C.2.6 DefaultIconScript: Default Scripted Actions for Icons

DefaultIconScript is similar to *IconScript*, except that it gets dragged onto the background of a model rather than onto an icon, and it specifies actions for many icons instead of just one. It has the same parameters as *IconScript*, but like *DefaultIconLink* described above, it also has *include* and *instancesOf* parameters, which have the same meaning described above in Section C.2.3.

DefaultIconScript can be used, for example, to override the default behavior that causes parameters to be displayed on mouse over, as shown in Figure C.2.

C.2.7 Title: Title for Icons

The *Title* parameter is used to customize the title displayed in a web page. This parameter also appears as a title in the Vergil window. The title in Figure C.7 is actually given by an instance of *Title* inserted into the model, with the default title changed to read "Illustration of Signal Processing Using SDF." This replaces the default title provided by the web export, which is the name of the model. This title also becomes the title defined in the header of the exported HTML file.

The default value of the *Title* parameter is the expression

```
    $(this.getName())
```

which is an expression in the Ptolemy II expression language for string parameters (see Appendix A). This expression invokes the `getName()` method on the container object, so the default title that is displayed is the name of the model.

C.2.8 DefaultTitle: DefaultTitle for Icons

The *DefaultTitle* parameter is used to customize the title associated with each icon in a model. This title is what shows up on the exported web page as a tooltip when the mouse lingers over an icon. Like *DefaultIconLink* described above, it also has *include* and *instancesOf* parameters, which have the same meaning described above in Section C.2.3. These can be used to specify default titles for subsets of icons.

Sidebar: Command-Line Export

Given a MoML file for a model, you can generate a web page using a command-line program called `ptweb`. The command should have the following form:

```
    ptweb [options] model [targetDirectory]
```

The "model" argument should be a MoML file. If no target directory is specified, then the name of the model becomes the name of the target directory (after any special characters have been replaced by characters that are allowed in file names). The options include:

- `-help`: Print a help message.
- `-run`: Run the model before the web page is exported, so that plot windows are included the export.