Model Comparison Utility

September 2019

McMaster Centre for Software Certification (McSCert)

Contents

1 Introduction

Differencing between two models is natively supported in Simulink via the Simulink Comparison Tool. This tool relies on XML comparison techniques to generate a Word or HTML report displaying the changes that occur between models. Unfortunately, for large industrial models, these generated reports are not readable. As an alternative, the tool can output the comparison results to the MATLAB base workspace as an `xmlcomp.Edits` object that is structured as a tree. An example of this object's structure can be seen in Figure 2.

Unfortunately, MathWorks provides no built-in commands to be able to easily and programmatically query or parse this tree from the command line or a script. Manually doing so for industrial models is simply not possible. Moreover, extracting information from the tree requires thorough knowledge of the tree structure and the object parameters, and thus is not trivial without much effort. The Model Comparison Utility was created to facilitate such operations via a collection of commands. Some useful commands provided by this tool are:

- `find_node` – Search the comparison tree for nodes with specific block types, changes, names, etc. (Section 2.3.1).
- `getHandle` – Get the handle of the model element associated with the node from the comparison tree (Section 2.3.2).
- `getPath` – Get the pathname of the model element associated with the node from the comparison tree (Section 2.3.3).
- `plotTree` – Plot the digraph of the comparison tree (Section 2.3.5).
- `summaryOfChanges` – Print a summary report of the changes in the comparison tree to the Command Window or a `.txt` file (Section 2.3.6).

Many other commands are included and are free to be used, but are not listed here. Please explore the source files for this utility to see all the of the various functions.

1.1 About the Comparison Tree

This section gives a brief overview of how a comparison tree is structured and explains some of the unintuitive aspects of the tree. A graphical representation of a comparison tree is shown in Figure 2. Note that there are slight differences to this tree between some Simulink versions (e.g., between R2016b, R2017b, and R2019a).

In general, there are two kinds of objects that comprise the tree: `xmlcomp.Edits` and `xmlcomp.Node`. The root node of a comparison tree is an `xmlcomp.Edits` object. It is shown in black in Figure 2. This object contains information about the comparison, including file names of the files being compared, filters applied during comparison, and most importantly, the hierarchical nodes that differ between the two models. The `xmlcomp.Edits` properties are described in

Figure 1. The LeftRoot and RightRoot link to the `xmlcomp.Node` objects that make up each sub-tree representing each model.

Property of `xmlcomp.Edits`	Description
`Filters`	Array of filter structure arrays. Each structure has two fields, Name and Value.
`LeftFileName`	File name of left file exported to XML.
`LeftRoot`	`xmlcomp.Node` object that references the root of the left tree.
`RightFileName`	File name of right file exported to XML.
`RightRoot`	`xmlcomp.Node` object that references the root of the right tree.
`TimeSaved`	Time when results exported to the workspace.
`Version`	MathWorks® release-specific version number of `xmlcomp.Edits` object.

Property of `xmlcomp.Node`	Description
`Children`	Array of `xmlcomp.Node` references to child nodes, if any.
`Edited`	Boolean — If `Edited = true` then the node is either inserted (green) or part of a modified matched pair (pink).
`Name`	Name of node.
`Parameters`	Array of parameter structure arrays. Each structure has two fields, Name and Value.
`Parent`	`xmlcomp.Node` reference to parent node, if any.
`Partner`	If matched, `Partner` is an `xmlcomp.Node` reference to the matched partner node in the other tree. Otherwise empty `[]`.

Figure 1: Properties of the comparison objects [1].

Each element that is an `xmlcomp.Node` usually represents a block, line, annotation, port, mask, or block diagram from the Simulink model that has been changed. Simulink model elements which have *not* been changed are not included in the tree, unless they are a componentization block, such as a Subsystem, that is needed to preserve the hierarchy (e.g., Subsystem1 and Subsystem2 in Figure 2. It is important to remember that the tree is only representative of the parts of the model that have changed, and unchanged parts of the model may not be represented by the tree.

An `xmlcomp.Node` object's properties are described in Figure 1. Note that the Edited field is only set to *true* when the node itself is different in the tree hierarchy (deleted, added, moved). It is not set when a node property (block parameter) is changed. The Model Comparison Utility does not rely on this field as an accurate indicator of change in the model. To determine the types of changes that occur, the Model Comparison Utility takes into account changes to the node's Name, Partner, Parent, and Parameters. Moreover, the placement in the tree is important for understanding how the element has changed, as summarized in Table 1.

It is important to look at the whole tree to understand the changes. By only looking at the left sub-tree, a node that exists in the left sub-tree can be either deleted or renamed (but definitely not added or modified). By only looking at the right-subtree, a node that exists in the right-subtree can be added or

Change Type	`xmlcomp.Node` Placement in the Comparison Tree
Added	Node exists in the right sub-tree, but not the left
Deleted	Node exists in the left sub-tree, but not the right
Modified	Node exists in the left and right sub-trees, and it is partnered
Renamed	Node exists in the left and right sub-trees, and it is not partnered

Table 1: Effect of placement on an `xmlcomp.Node`

modified (but definitely not deleted). Even after determining the placement of the node, it is important to check its other properties (and that of its partner) to further understand what type of change it has experienced. The Model Comparison Utility performs all of these checks automatically, so the user does not have to.

1.2 More Information

For more information about model comparison with MATLAB/Simulink, please see the MathWorks documentation:
https://www.mathworks.com/help/simulink/model-comparison.html

2 How to Use the Tool

This section describes what must be done to setup the tool, as well as how to use the tool.

2.1 Prerequisites and Installation

1. Use MATLAB/Simulink R2016b or newer.
   - Note: R2016b has many bugs in the model comparison algorithm. For best results, please use R2019a+.
2. To install the tool,
   (a) from a `.zip` file — unzip the contents into your desired location. Ensure the unzipped folder and subfolders are present in your MATLAB search path, or add them if they are not present.
   (b) from a `.mltbx` file — simply open MATLAB and double-click on the file. Your MATLAB search path should be automatically configured.
   (c) from the files only — add the folders and subfolders to your MATLAB search path.
   - Note: If running "`which ModelComparisonUtility_UserGuide.pdf`" displays "'ModelComparisonUtility_UserGuide.pdf' not found.", then the tool needs to be added to the MATLAB search path. For information on adding files to the MATLAB search path, see the documentation for it online.
3. Ensure your models are open or loaded.

2.2 Getting Started

The utility commands are used via the MATLAB Command Window. To compare models and create a comparison tree, use the `slxmlcomp.compare` function by entering:

```
1 Edits = slxmlcomp.compare(model1, model2)
```

where `model1` is the model before changes, and `model2` is the model after changes. Two example models, `demo_before.mdl` and `demo_after.mdl`, are provided.

```
1 Edits = slxmlcomp.compare('demo_before', 'demo_after')
```

The `xmlcomp.Edits` object is a root node that links to two n-ary sub-trees of differences between two models. The comparison tree for the example is shown in Figure 2. The nodes in blue correspond to actual elements in the Simulink models.

2.3 Functionality

This section describes a few of the useful functions that are provided by the Model Comparison Utility. Full instructions on function parameters and output are given in the scripts' header comments. Feel free to explore all the scripts that are included! An example of using these functions is given in Section 3.

2.3.1 Finding Nodes

A comparison tree can be comprised of numerous nodes. The `find_node` function lets you search the tree for a specific node, in a similar way that the `find_system` function searches for elements in a Simulink model. The user can provide a list of constraints, and the function will return all nodes that fit them. It is possible to search for a node based on its change type (e.g., added, deleted, modified, renamed), block type (Subsystem, Inport, Constant, etc.), block name, or node name.

2.3.2 Getting a Node's Handle in the Model

One of the issues with the comparison tree is that although `xmlcomp.Node` objects abstractly represent elements from the Simulink models from which it was generated, there is no built-in way of getting a node's handle. The `getHandle` function will return the node's handle, if one exists. Note that some objects do not have an associated handle in the model (e.g., Mask, Comparison Root), while other objects have two handles if they exist in both sub-trees (e.g., renamed block).

2.3.3 Getting a Node's Path in the Model

The `getPath` function returns the node's full pathname in the model. Be aware that some model elements do not have a path (e.g., lines, annotations). Note, that if the path of the node in the tree is desired, please use the `getPathTree` function.

2.3.4 Getting a Node's Path in the Comparison Tree

The `getPathTree` function returns the node's full path in the comparison tree.

2.3.5 Plotting the Comparison Tree

The `plotTree` function provides a way of visually viewing the structure of a comparison tree. It plots a directed graph. Note that the full names of each node are used because unique node names are required when plotting a directed graph. The plot for the example is shown in Figure 3.

2.3.6 Printing a Summary of Changes

The `summaryOfChanges` function prints a text summary of the changes to a file or to the command line. Feel free to make modifications and write your own queries to include in the report.

2.3.7 Highlight Nodes in the Model

The `highlightNodes` function highlights model elements corresponding to nodes.

3 Example

Two models are provided in the `example` folder for demonstration purposes. They are shown in Figure 4. These models have several differences as a result of the following changes:

- 1 added element:
  - Data Store Memory block was added.
- 3 deleted elements:
  - Integrator block was deleted and replaced by Gain block.
  - The lines going into and out of Integrator are implicitly considered deleted when Integrator is deleted, and then added when Gain is connected.
- 2 modifications:
  - Add block's List of signs property was changed from ++ to – –.
  - Constant block's Constant value property was changed from 1 to 2.
- 1 renamed element:
  - Outport block named Out1 was renamed to NewName.

To create the comparison tree, use the commands in the Command Window:

```
model1 = 'demo_before';
model2 = 'demo_after';
load_system(model1);
load_system(model2);
Edits = slxmlcomp.compare(model1, model2)
```

After the `xmlcomp.Edits` object is created, the Model Comparison Utility can be used.

3.1 Finding Nodes

To find a specific set of nodes, use the `find_node` function. For example, to find only blocks that have been added, use the following command:

```
added = find_node(Edits, 'ChangeType', 'added', 'NodeType', 'block')

added =

  2x1 Node array with properties:

    Children
    Edited
    Name
    Parameters
    Parent
    Partner
```

3.2 Getting a Node's Handle in the Model

To determine the handle of one of the added nodes, use the following command. Note that because the node is added, that means that it only exists in the right sub-tree (see Table 1), so we find the element in `model2`.

```
h = getHandle(added(1), model2)

h =

   13.0004
```

3.3 Getting a Node's Path in the Model

To determine the path of the added node in the model, use the following command. Again, because the node is added, that means that it only exists in the right sub-tree (see Table 1), so we find the element in `model2`.

```
p = getPath(added(1), model2)

p =

    'demo_after/Subsystem/Subsystem/Gain'
```

3.4 Getting a Node's Path in the Tree

To determine the path of the node in the comparison tree, use the following command:

```
pt = getPathTree(added(1))

pt =

    'Comparison Root/Simulink/Subsystem/Subsystem/Gain'
```

3.5 Plotting the Comparison Tree

To visualize the comparison tree, use the following command to plot it. This will display Figure 3.

```
plotTree(Edits);
```

3.6 Printing a Summary of Changes

To print a textual report of the changes to the Command Window, use the following command. The format of the report is first the query information: <Query constraints> -- TOTAL <#>, following by the tree paths of the nodes. It is possible to print the summary to a `.txt` file, or omit including the paths. Please see the function's header comment for more information. Feel free to modify this report with the queries you require.

```
>> summaryOfChanges(Edits, 1)

ChangeType, added -- TOTAL 4
   Comparison Root/Simulink/Subsystem/Subsystem/Gain
   Comparison Root/Simulink/Subsystem/Subsystem/In2:1 -> Gain:1
   Comparison Root/Simulink/Subsystem/Subsystem/Gain:1 -> Out2:1
   Comparison Root/Simulink/Subsystem/Data Store Memory
ChangeType, deleted -- TOTAL 3
   Comparison Root/Simulink/Subsystem/Subsystem/Integrator
   Comparison Root/Simulink/Subsystem/Subsystem/In2:1 -> Integrator:1
   Comparison Root/Simulink/Subsystem/Subsystem/Integrator:1 -> Out2:1
ChangeType, renamed -- TOTAL 2
   Comparison Root/Simulink/Subsystem/Subsystem/Out1
   Comparison Root/Simulink/Subsystem/Subsystem/NewName
ChangeType, modified -- TOTAL 4
   Comparison Root/Simulink/Subsystem/Subsystem/Add
   Comparison Root/Simulink/Subsystem/Subsystem/Constant
   Comparison Root/Simulink/Subsystem/Subsystem/Add
```

```
   Comparison Root/Simulink/Subsystem/Subsystem/Constant

NodeType, block, ChangeType, added -- TOTAL 2
   Comparison Root/Simulink/Subsystem/Subsystem/Gain
   Comparison Root/Simulink/Subsystem/Data Store Memory
NodeType, block, ChangeType, deleted -- TOTAL 1
   Comparison Root/Simulink/Subsystem/Subsystem/Integrator
NodeType, block, ChangeType, renamed -- TOTAL 2
   Comparison Root/Simulink/Subsystem/Subsystem/Out1
   Comparison Root/Simulink/Subsystem/Subsystem/NewName
NodeType, block, ChangeType, modified -- TOTAL 4
   Comparison Root/Simulink/Subsystem/Subsystem/Add
   Comparison Root/Simulink/Subsystem/Subsystem/Constant
   Comparison Root/Simulink/Subsystem/Subsystem/Add
   Comparison Root/Simulink/Subsystem/Subsystem/Constant

NodeType, block, ChangeType, added, BlockType, inport -- TOTAL 0
NodeType, block, ChangeType, deleted, BlockType, inport -- TOTAL 0
NodeType, block, ChangeType, renamed, BlockType, inport -- TOTAL 0
NodeType, block, ChangeType, modified, BlockType, inport -- TOTAL 0

NodeType, block, ChangeType, added, BlockType, outport -- TOTAL 0
NodeType, block, ChangeType, deleted, BlockType, outport -- TOTAL 0
NodeType, block, ChangeType, renamed, BlockType, outport -- TOTAL 2
   Comparison Root/Simulink/Subsystem/Subsystem/Out1
   Comparison Root/Simulink/Subsystem/Subsystem/NewName
NodeType, block, ChangeType, modified, BlockType, outport -- TOTAL 0

NodeType, line, ChangeType, added -- TOTAL 2
   Comparison Root/Simulink/Subsystem/Subsystem/In2:1 -> Gain:1
   Comparison Root/Simulink/Subsystem/Subsystem/Gain:1 -> Out2:1
NodeType, line, ChangeType, deleted -- TOTAL 2
   Comparison Root/Simulink/Subsystem/Subsystem/In2:1 -> Integrator:1
   Comparison Root/Simulink/Subsystem/Subsystem/Integrator:1 -> Out2:1
NodeType, line, ChangeType, renamed -- TOTAL 0
NodeType, line, ChangeType, modified -- TOTAL 0
```

3.7 Highlighting Nodes in the Model

To highlight nodes in the model, use the following command. Note that you must specify which model to use, and that some nodes may not exist in that model (e.g., deleted nodes, added nodes).

```
>> open_system(mdl2);
>> highlightNodes(added, mdl2);
```

4 Limitations

Support for Stateflow comparison will be added in a future release of this tool.

References

[1] The MathWorks. "Compare XML Files".
`https://www.mathworks.com/help/matlab/matlab_env/compare-xml-files.html`
[Online; Accessed June 2019]

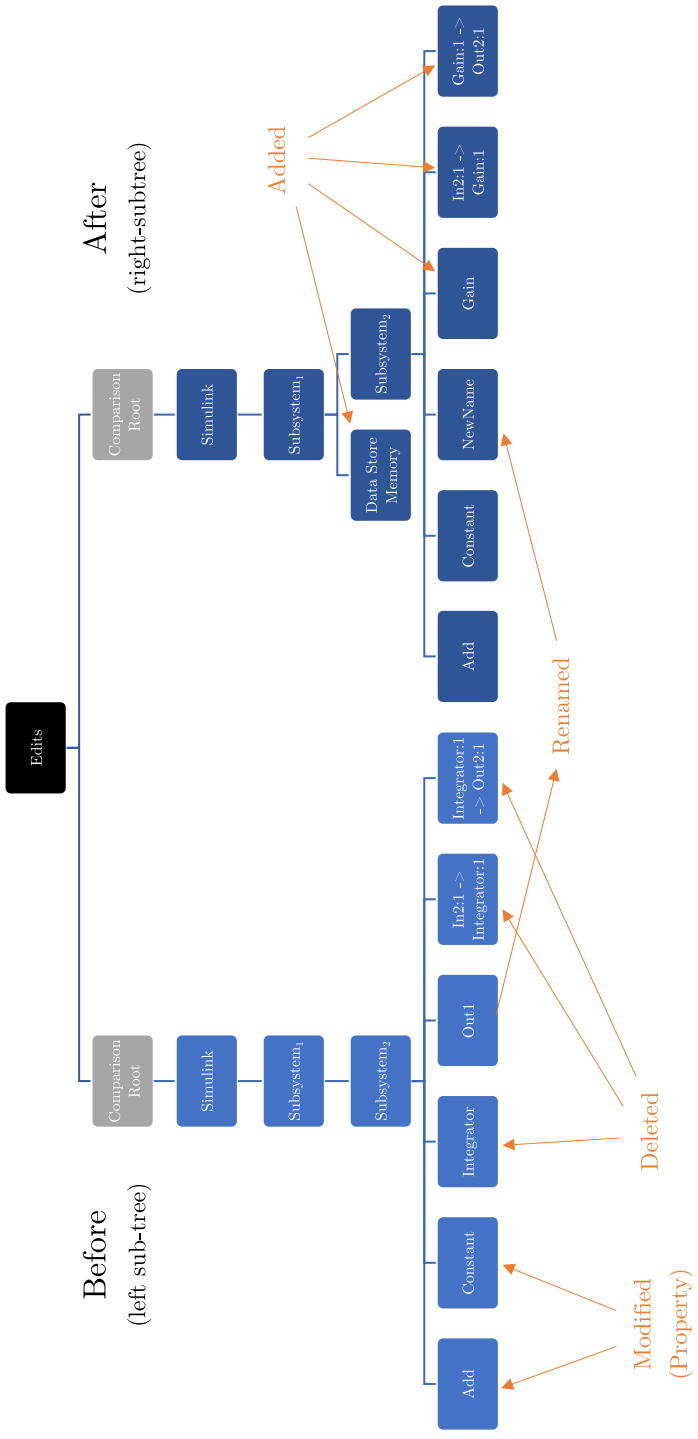


Figure 2: The `xmlcomp.Edits` object as created by the Simulink Comparison Tool.

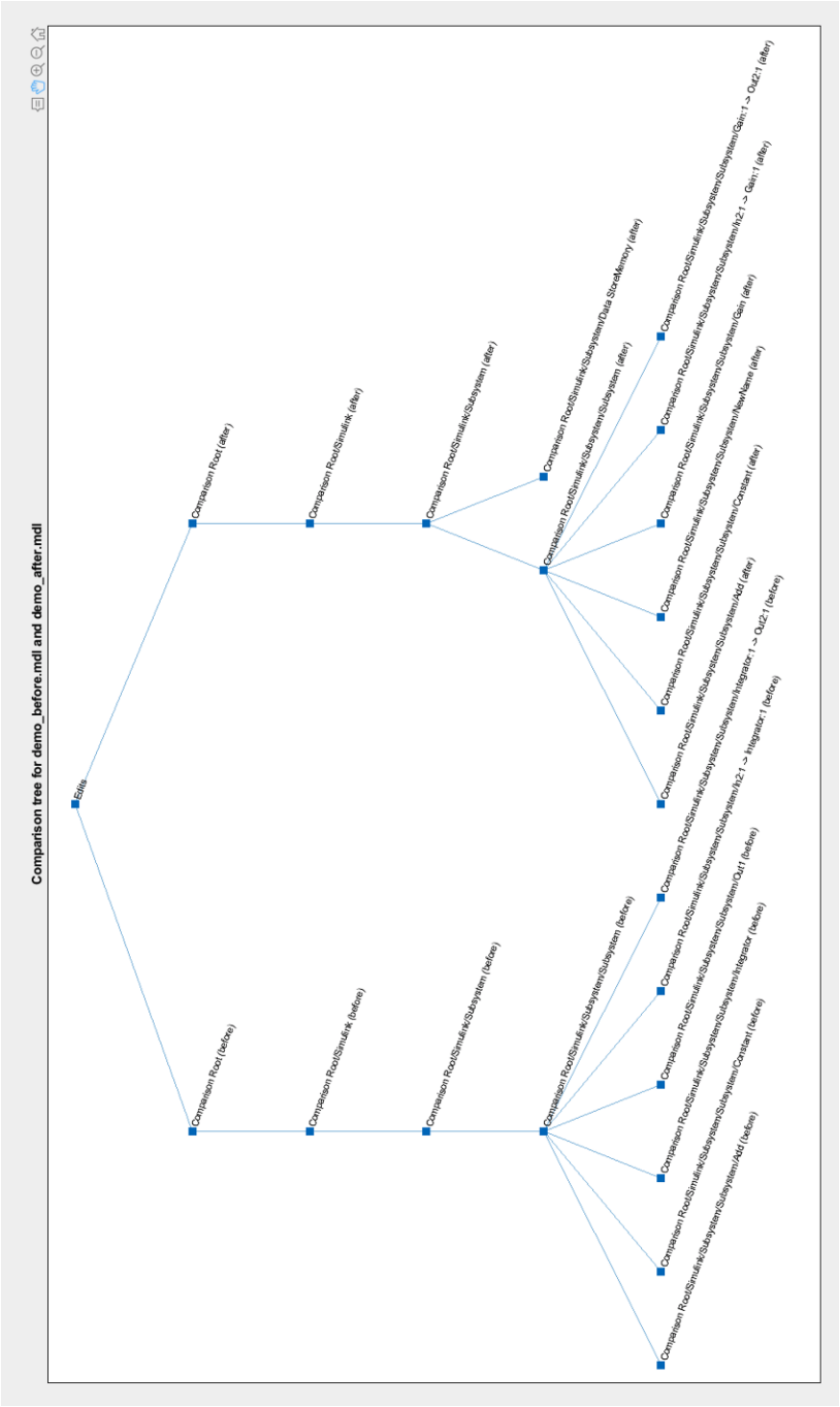
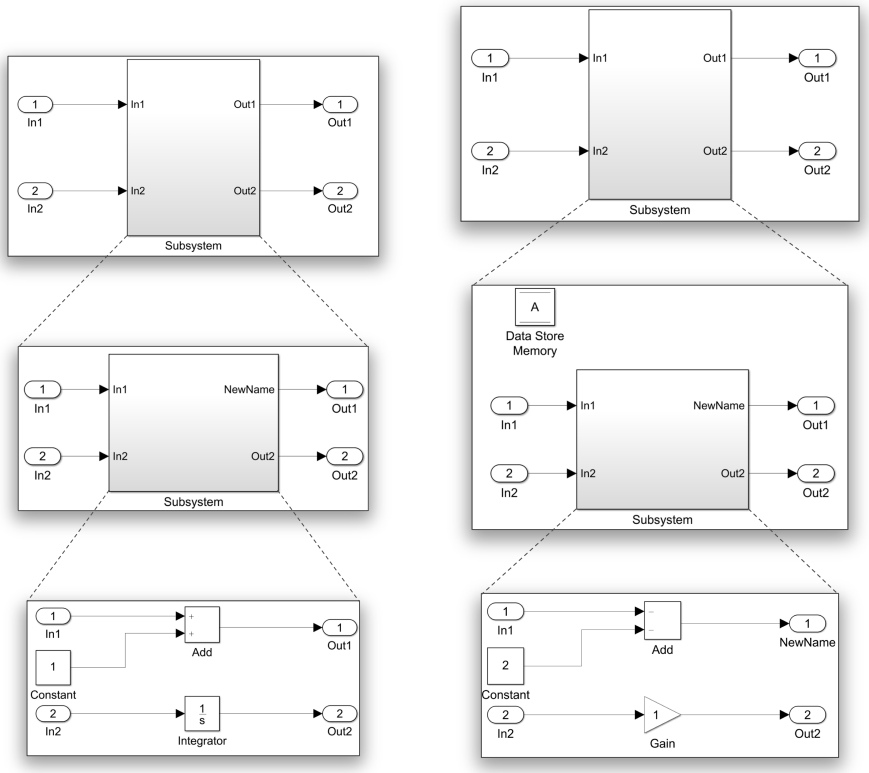


Figure 3: Plot of the comparison tree.



(a) Model Before Changes. (b) Model After Changes.

Figure 4: Two Versions of a Single Model.