



European Organisation  
for Astronomical Research in  
the Southern Hemisphere

Organisation Européenne  
pour des Recherches  
Astronomiques dans  
l'Hémisphère Austral

Europäische Organisation für  
astronomische Forschung in  
der südlichen Hemisphäre

---

## Systems Reasoner User Manual

1 ISSUE 1

1

Name

Date

Signature



## Authors

Name	Affiliation
Chakajkla Jesdabodi	TUM

## Change Record

Issue	Date	Section / Paragraph affected	Reason / Initiation Documents / Remarks

---

## Table of Contents

1. Introduction .....	5
1.1. Purpose .....	5
1.2. Scope .....	5
1.3. Abbreviations and Acronyms .....	5
1.4. Copyright .....	5
2. Overview .....	7
3. Instruction .....	8
3.1. Apply reasoning pattern .....	9
3.2. Remove reasoning pattern .....	10
3.3. Example Usage .....	11
3.3.1. Create product model .....	11
3.3.2. Applying the patterns .....	13
3.3.3. Creating instance model and simulator .....	16
3.3.4. Applying multiple patterns .....	19

---

## Chapter 1. Introduction

Reasoner component is a subpart of the MBSE plugin as described in the MBDG Documentation. Reasoner component is used for applying a reasoner pattern for getting the total cost, mass or power consumption of a product structure tree etc. Different type of pattern blocks can be created to suit the calculation needs, the reasoner component then does the work of automatically applying these pattern blocks (generalization) to a single product block or multiple blocks recursively.

### 1.1. Purpose

The reasoner module is used to help modeling easier, faster and less prone to error by providing an automatic wizard for applying a pattern and creating property values.

### 1.2. Scope

This document is the user manual for a reasoner module of MBSE Plugin. It also briefly addresses the steps of applying a pattern to a product tree and how the instance of this tree can be simulated to get a certain values like total cost, mass or power consumption.

### 1.3. Abbreviations and Acronyms

**MBDG**     **Model Based Document Generation**

### 1.4. Copyright

The software described in this document is subject to the following copyright statement:

\*

\*

\* (c) INCOSE SE2 Challenge Team for Telescope Modeling 2011

\* Copyright by ESO, HOOD, TUM, oose, GfSE

\* All rights reserved

\*

\* This library is free software; you can redistribute it and/or

\* modify it under the terms of the GNU Lesser General Public

\* License as published by the Free Software Foundation; either

\* version 2.1 of the License, or (at your option) any later version.

\*

\* This library is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

\* Lesser General Public License for more details.

\*

\* You should have received a copy of the GNU Lesser General Public

\* License along with this library; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

\*

\*

\*

---

## Chapter 2. Overview

The general usage of the reasoner component is to aid systems engineers in creating pattern based reasoning. With this component, automatic generalization and creation of property values can be performed to reduce modeling time and error.

---

## Chapter 3. Instruction

The general steps for using a reasoning pattern are:

1. Create system model (product tree)
2. Create a pattern blocks (Cost, Mass, Power Consumption blocks etc)
3. Apply the pattern blocks to the system model
4. Generate a system model instance and modify as needed
5. Run the simulator to get the results.

The reasoning component is created to aid the user in step 3 of the process, to automatically apply the pattern blocks recursively,

There are two options that can be used in the reasoner component, namely:

1. Apply reasoning pattern
2. Remove reasoning pattern

A screenshot below shows how the two actions can be invoked.

However, there are two other actions "SE2: getCost" and "SE2: get Power" menu option, this two options are customized to calculate the cost and power of the product tree directly (by working directly on the default value of value property).

They expect a value property called **cost** resp. **power**, and create a bill of materials (Figure 3.2, "Cost Rollup" ).The user can apply the pattern and then invoke these actions.

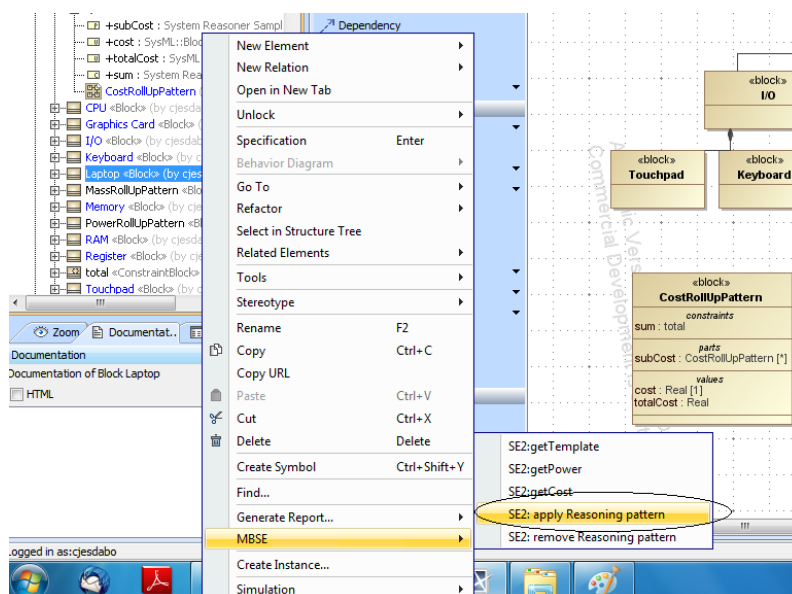
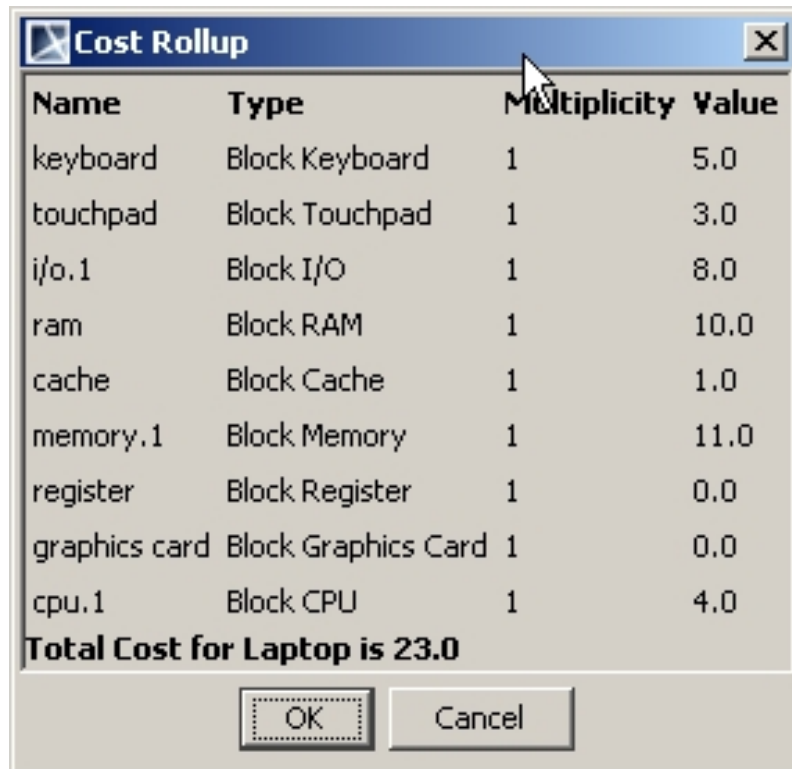


Figure 3.1. Invoking Actions





The image shows a 'Cost Rollup' dialog box with a table of components. The table has four columns: Name, Type, Multiplicity, and Value. The components listed are keyboard, touchpad, i/o.1, ram, cache, memory.1, register, graphics card, and cpu.1. The total cost for the laptop is 23.0. The dialog box has an OK button and a Cancel button at the bottom.

Name	Type	Multiplicity	Value
keyboard	Block Keyboard	1	5.0
touchpad	Block Touchpad	1	3.0
i/o.1	Block I/O	1	8.0
ram	Block RAM	1	10.0
cache	Block Cache	1	1.0
memory.1	Block Memory	1	11.0
register	Block Register	1	0.0
graphics card	Block Graphics Card	1	0.0
cpu.1	Block CPU	1	4.0

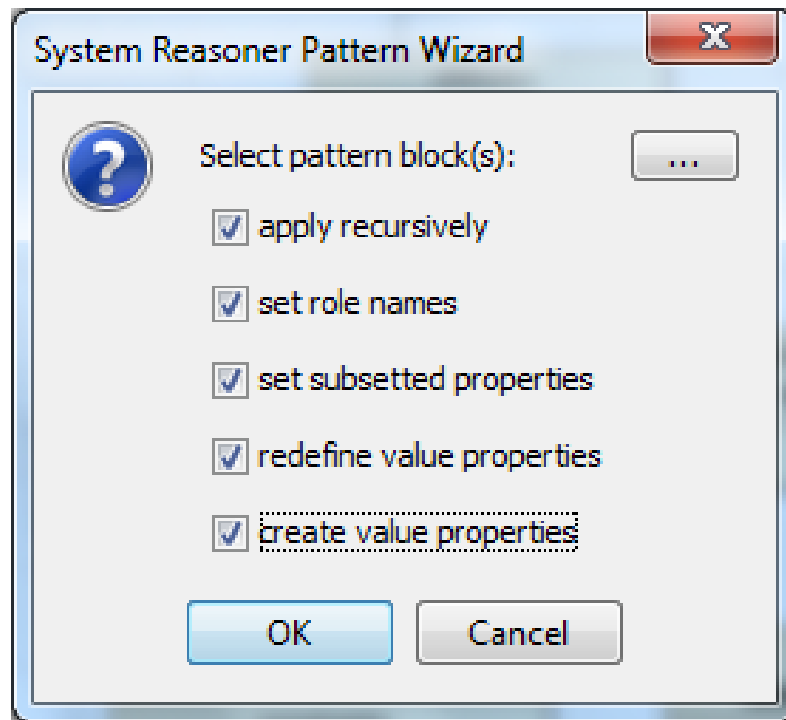
**Total Cost for Laptop is 23.0**

OK Cancel

Figure 3.2. Cost Rollup

### 3.1. Apply reasoning pattern

Applying reasoning pattern can be invoked through "SE2: apply reasoning pattern" command under a block element. Selecting the command will open the systems reasoning wizard as shown the figure below.



**Figure 3.3. Apply Reasoning pattern wizard**

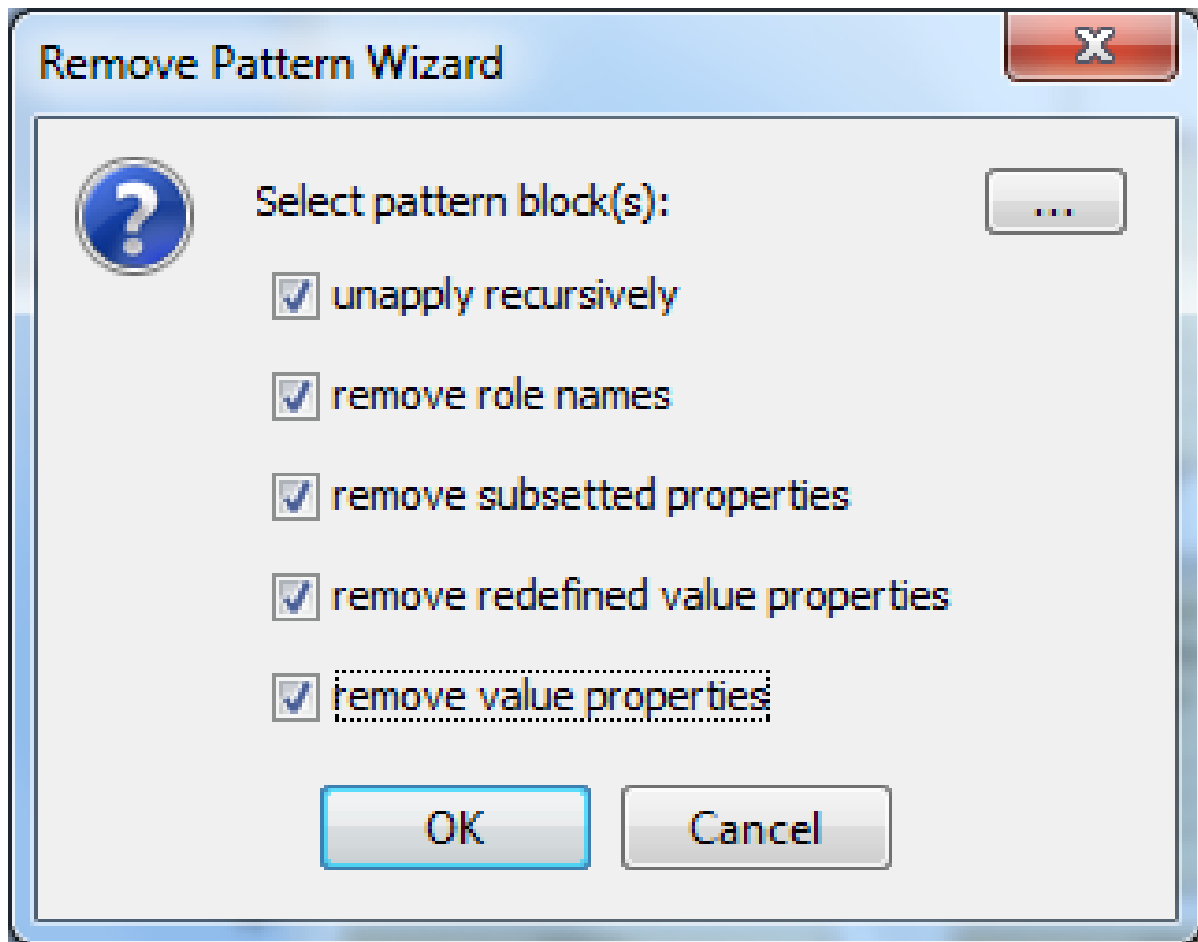
There are 5 options that can be selected:

1. Apply the pattern block recursively
2. setting the role names of part properties (lowercase of the blockname)
3. setting the subsetted properties of part properties
4. redefine value properties for the pattern blocks
5. create value properties of the pattern blocks

Multiple pattern blocks can be selected in dialog, this allows for multiple pattern blocks to be generalized (apply) automatically.

### 3.2. Remove reasoning pattern

Remove reasoning pattern is a wizard for removing the applied patterns, as well as removing all the created values, subsetted properties, redefined properties etc. An example wizard is shown in the figure below.



**Figure 3.4. Remove reasoner wizard**

Similarly to apply pattern wizard, there are 5 options the user can select either to remove the pattern recursively, with an option to remove the value properties and subsetted properties etc.

### 3.3. Example Usage

#### 3.3.1. Create product model

An example of pattern blocks is created in the package System Reasoner Sample Model. There are 3 pattern blocks created:

1. MassRollupPattern,
2. CostRollupPattern,
3. and a PowerRollupPattern.

MassRollUpPattern is created for calculating the total mass of the product tree components, this pattern can be applied to a single block to get a single block mass, or applied to the blocks under it recursively to get the entire mass of the product structure subtree. Similarly for CostRollUpPattern and PowerRollUpPattern, which are used to calculate the cost and power respectively. A product tree structure of Laptop is created and shown below.

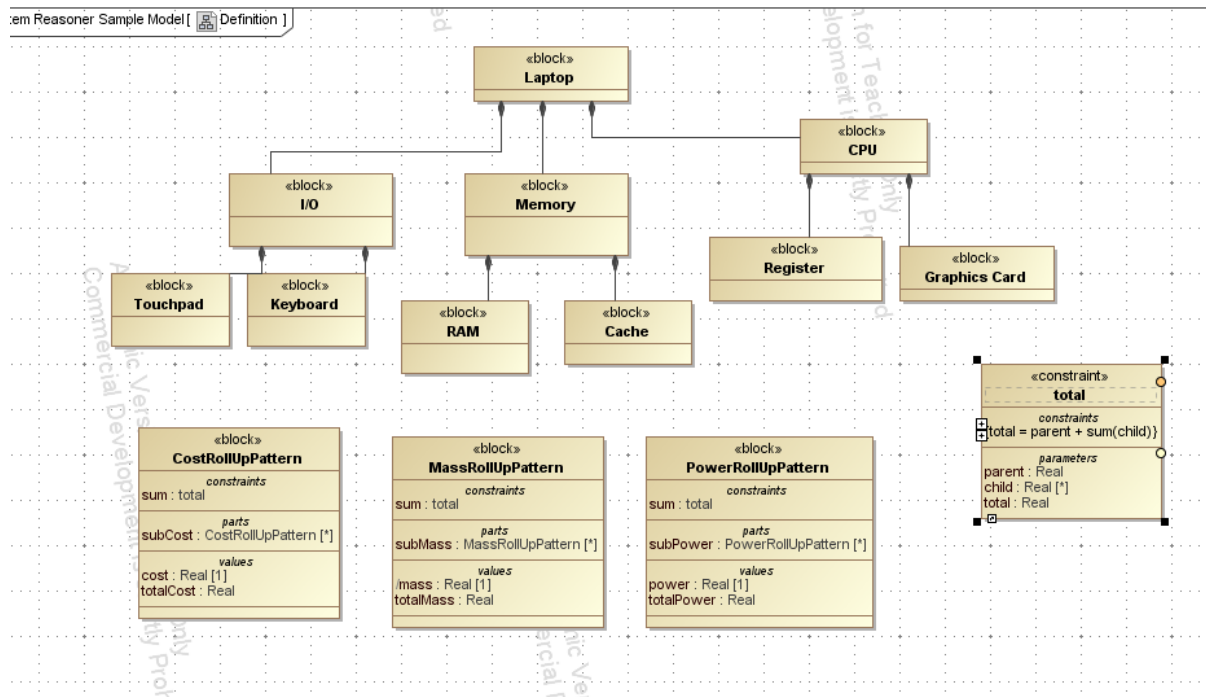


Figure 3.5. Initial Diagram

The 3 patterns must be created in association with a constraint block which defines the logic of the constraints. As shown below the constraints property "sum" is generalized as total constraint block.

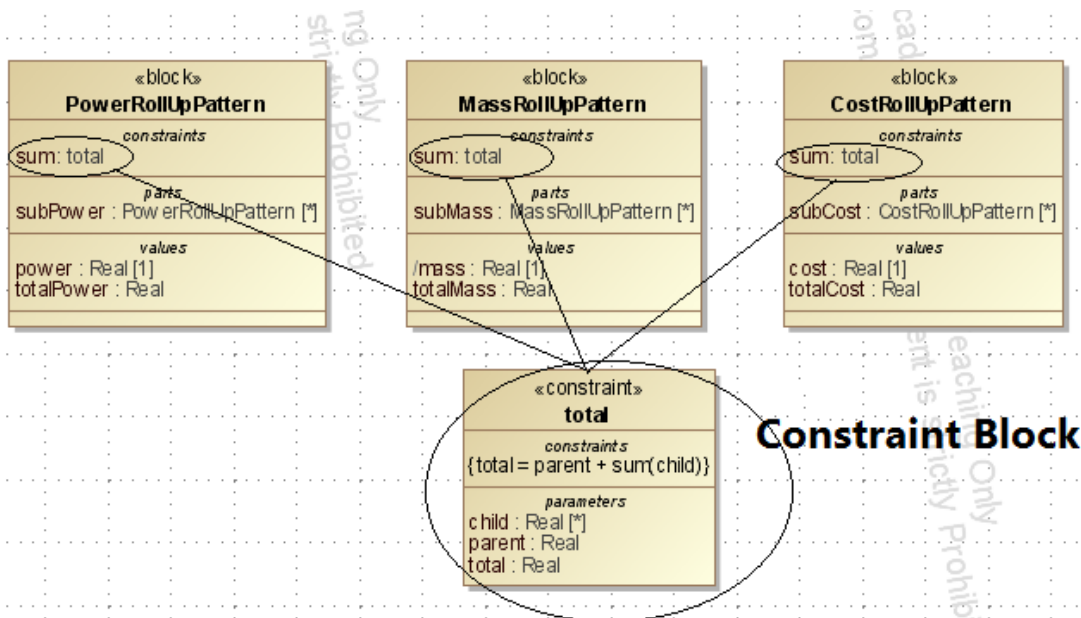


Figure 3.6. Constraints block

Also each pattern for e.g. CostRollUpPattern, must also contain a SysML parametric diagram which defines the reasoning pattern of the model. This parametric diagram allows for a more flexible way to define any kind of reasoning pattern to the product tree.



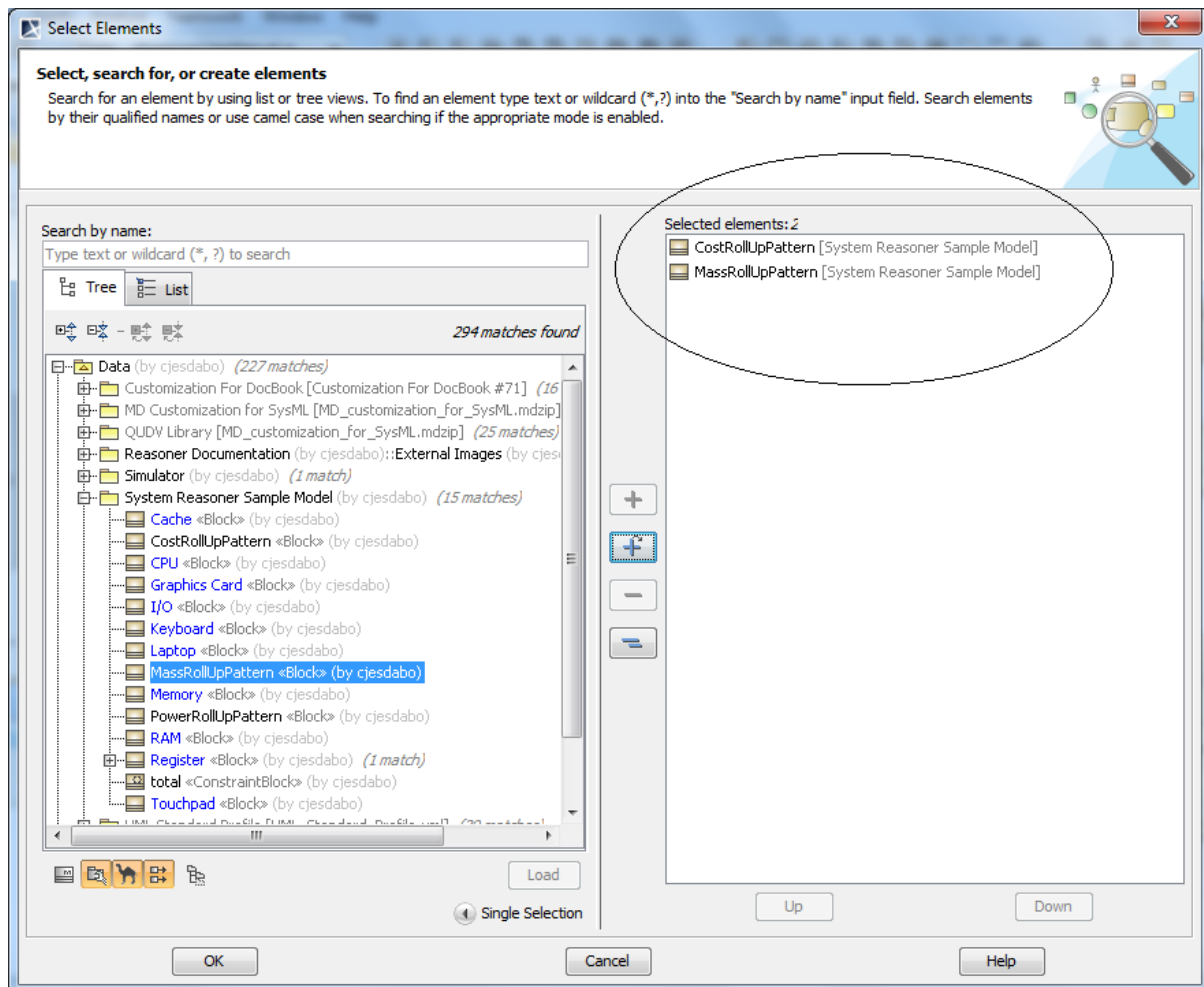
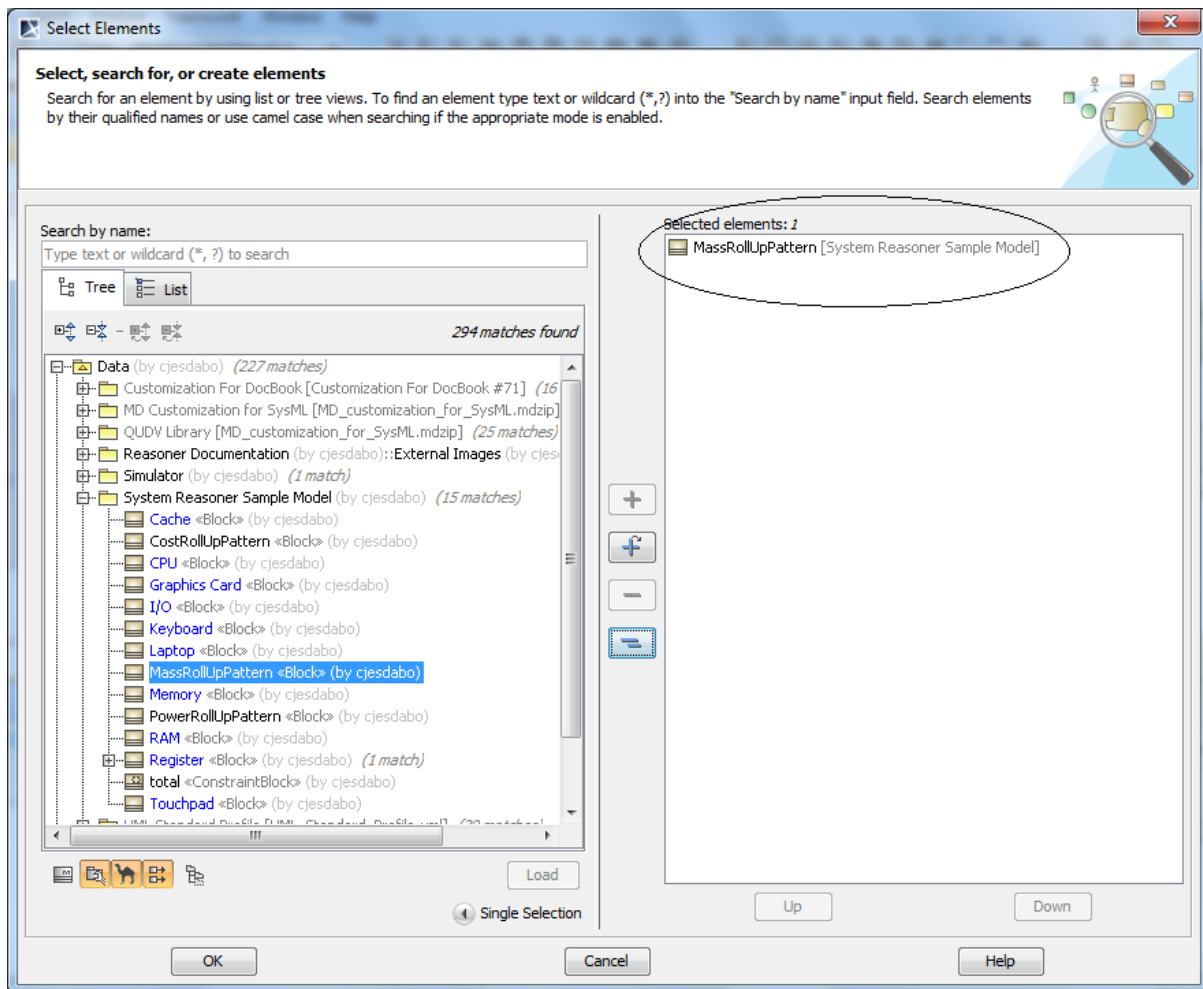


Figure 3.8. Multiple Pattern



**Figure 3.9. Singel apply**

After having configured the wizard to apply the pattern block, the resulting product tree will be updated and as shown below. Note that in this example we have selected all the 5 options (apply recursively, redefine roles, subsetting property etc.) and selected only MassRollUpPattern block. It is noticeable on the model that new property values are created, as well as roles names and subsetting properties are added. Most importantly all the blocks under the Laptop block is now a generalization of MassRollUpPattern block.

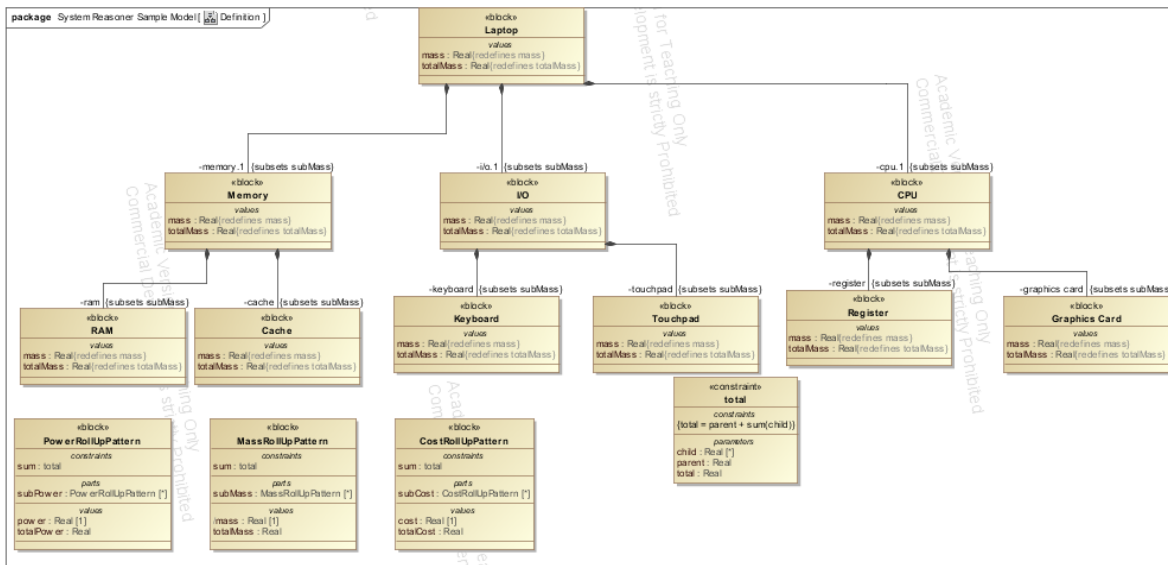


Figure 3.10. Model After apply

### 3.3.3. Creating instance model and simulator

The next step is to create an instance of this model. We have created the instance in a package "Instance". This package will contain all the instance blocks that will be used for the simulation. It is to be noted that currently in MD 17.0 redefines are not properly handled when using the Instance wizard but they are supported in 17.0.1. And, that subsets will be supported in future versions of MD. Therefore, the user will be able to generate the instance model directly using the wizard.

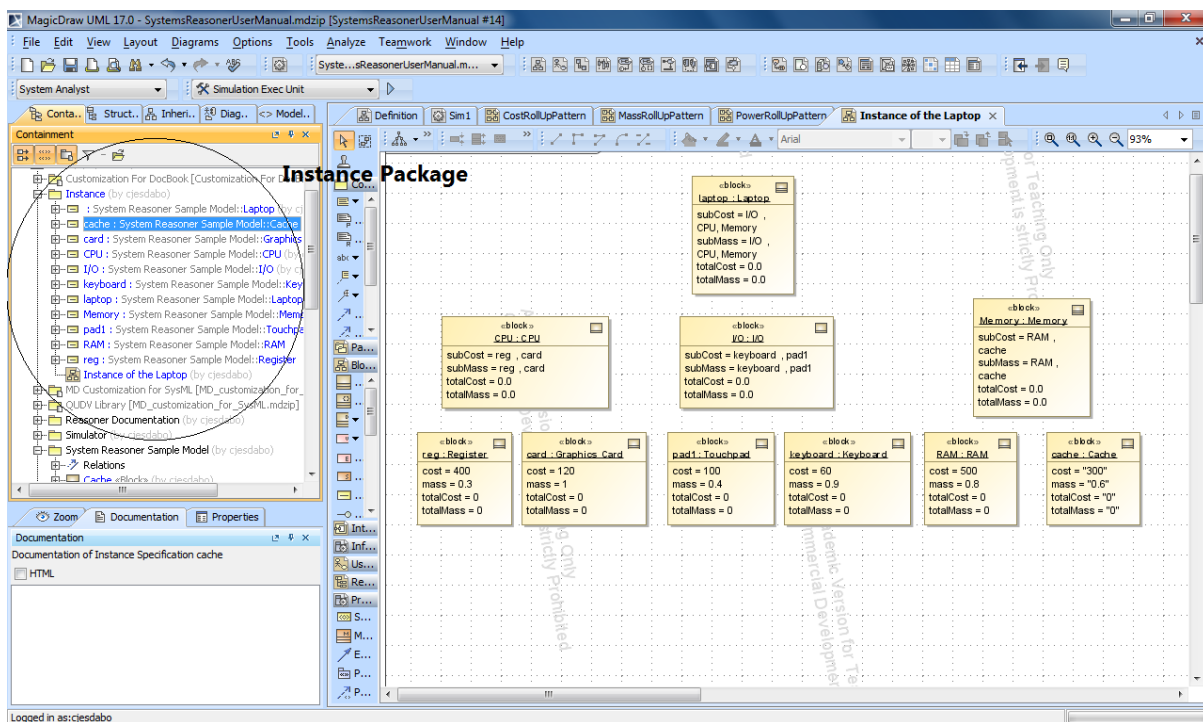


Figure 3.11. Instance Package

Having completed the instance model creation, we must now create a package that contains a simulator that contains



## 1. Simulation configuration diagram

## 2. Execution Configurator

With the execution configurator, the user can select the target instance which he wants to run the simulation against and also select the resulting instance block where he wants to store the simulation results. In this example, an Execution Configurator "Simulation Exec Unit" is created and set target instance and resulting instance to "Laptop" instance we have created.

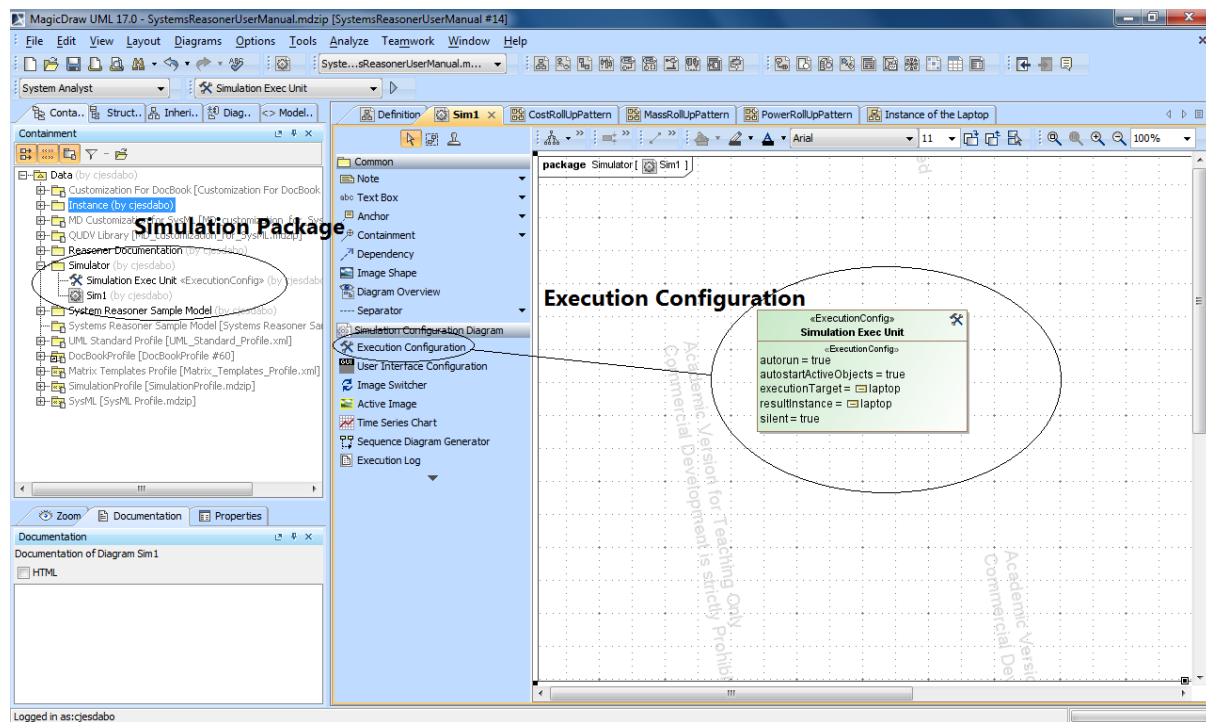


Figure 3.12. Simulator

Next we can run the execution by selecting run menu and set the Execution Configurator to Simulation Exec Unit. Note that multiple Execution Configurator can be created in the diagram.

The figureImage below shows the instance model before the execution, note that the totalCost of Laptop instance is 0. After the execution the totalCost value should reflect the true total cost of the entire instance product structure.

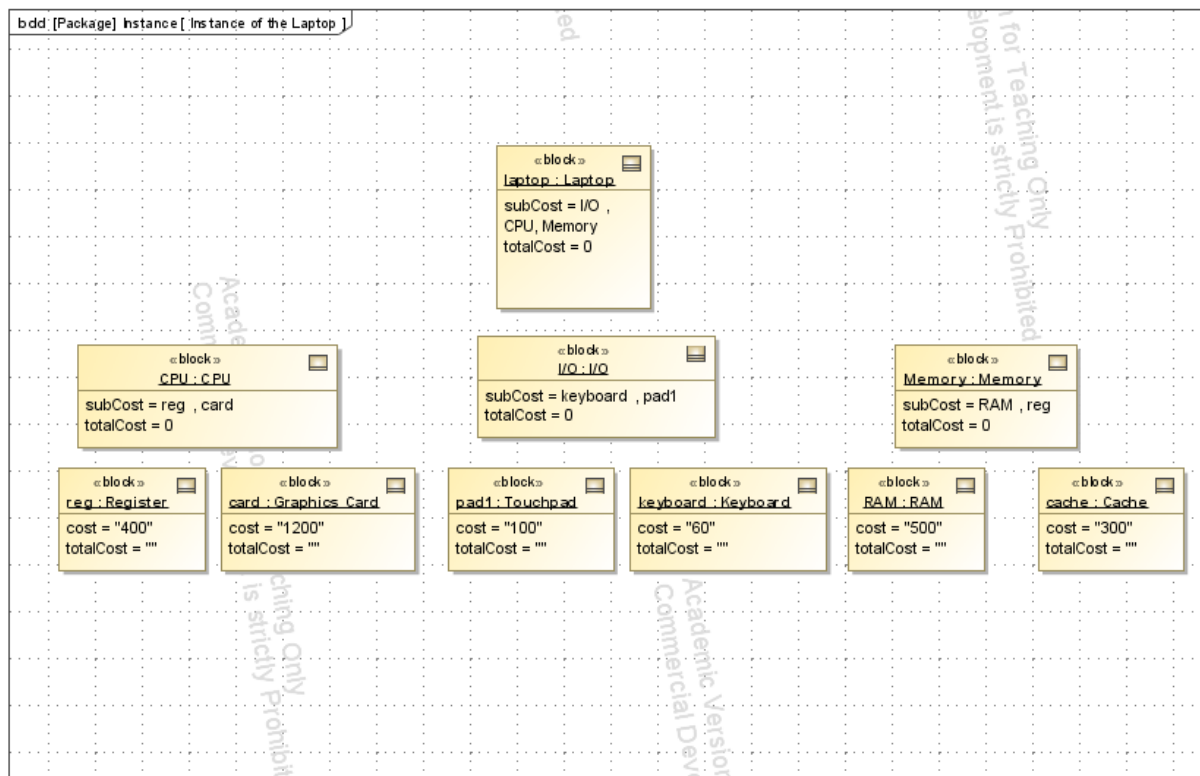


Figure 3.13. Instance before

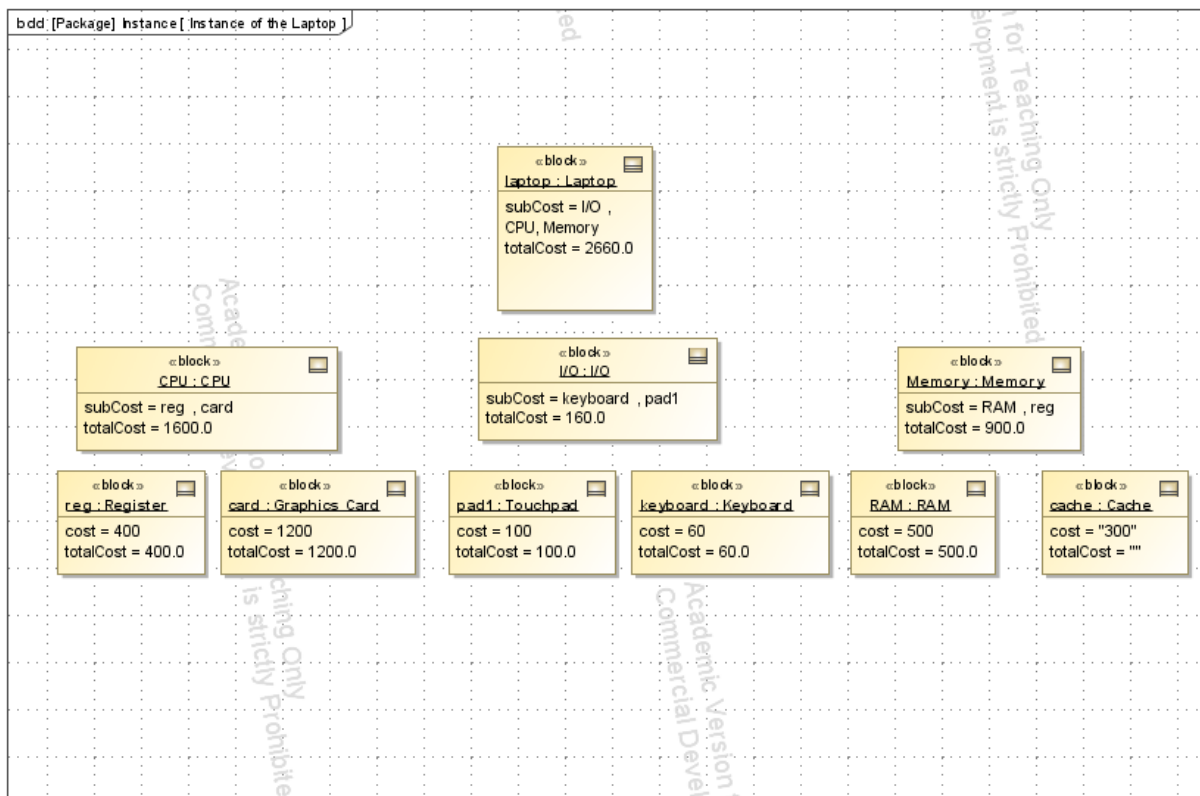


Figure 3.14. After

### 3.3.4. Applying multiple patterns

Multiple pattern blocks can be applied simultaneously through multiple selection dialog. An example of the product model after multiple blocks are applied is shown below. In this example both CostRollUpPattern and MassRollUpPattern is applied to Laptop block recursively.

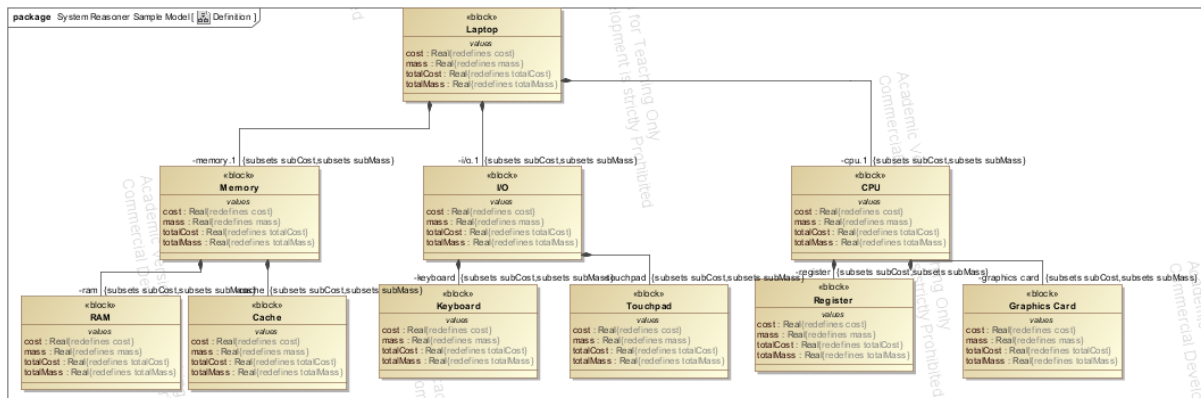


Figure 3.15. Multiple apply

Similarly, the instance models must then be created to reflect the apply changes.

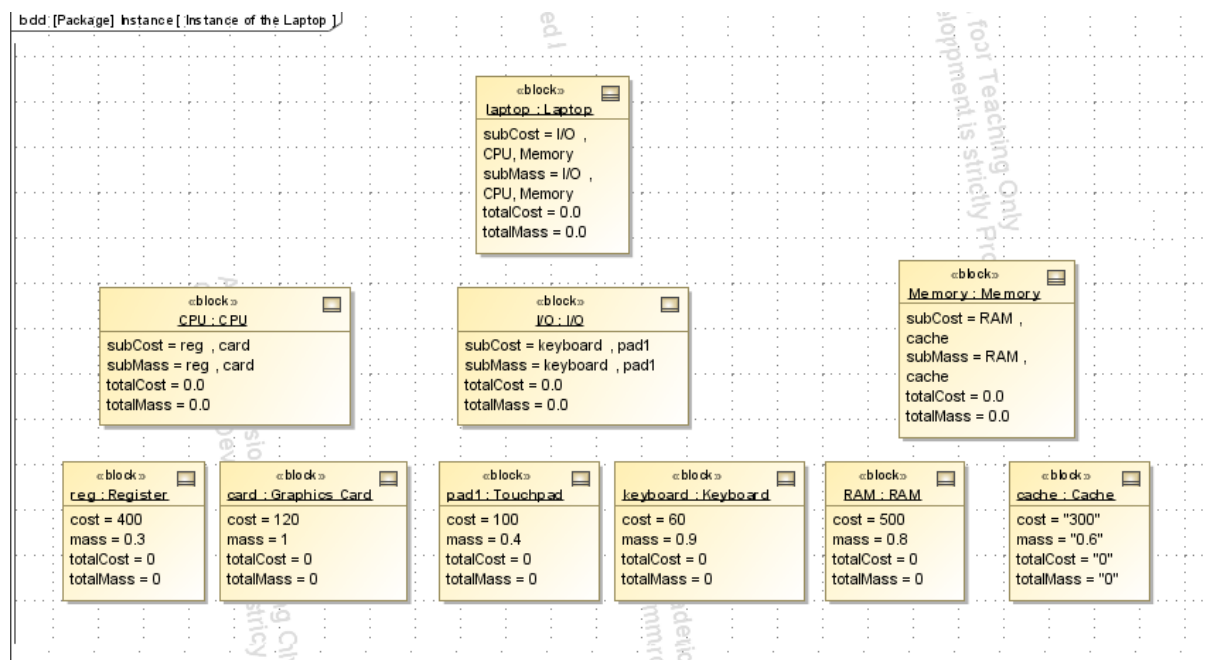


Figure 3.16. Multiple instance

Running the same Execution Configurator this time, will produce two results' total mass and cost of the laptop.

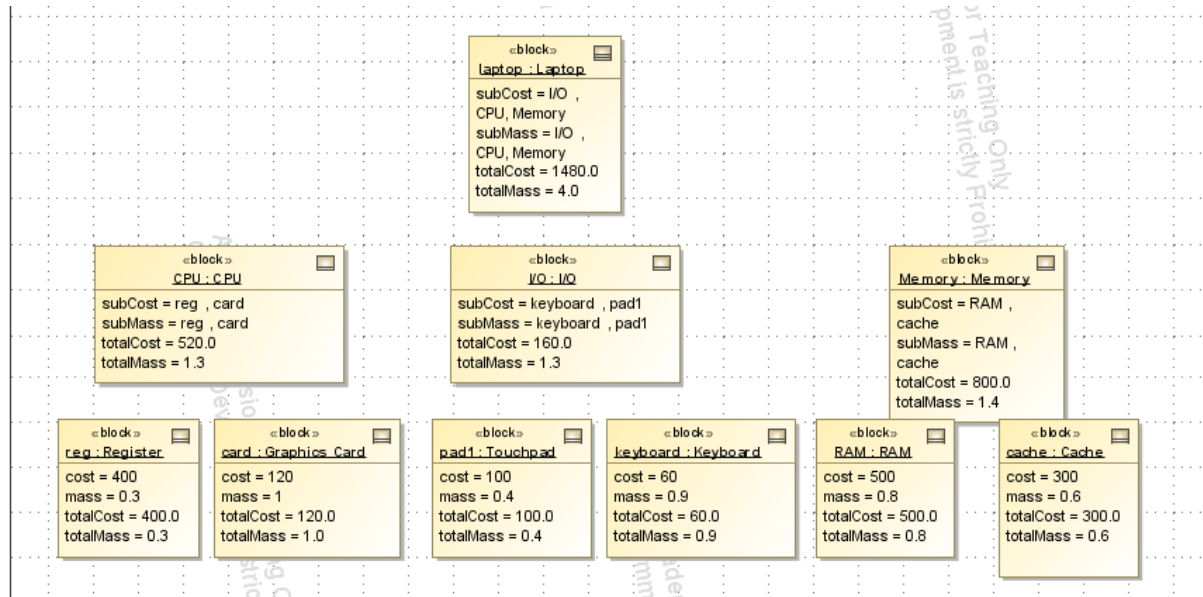


Figure 3.17. After multiple