

Requirements verification

On this page

- [Getting ready for automated Requirements verification](#)
- [Performing automated Requirements verification in Requirement Table](#)
- [Performing automated Requirements verification in Instance Table](#)
- [Performing automated Requirements verification using simulation](#)
- [Webinar: Automated Requirements Verification](#)

Systems Modeling Language (SysML) is used to capture systems design as descriptive and analytical system models, which relate text requirements to the design and provide a baseline to support analysis and verification. With the system parameter calculated, you can verify the system requirement and decide whether it is satisfied or not. The modeling tool enables you to perform this verification automatically.

Getting ready for automated Requirements verification

Before performing the automatic Requirements verification, you need to get ready.

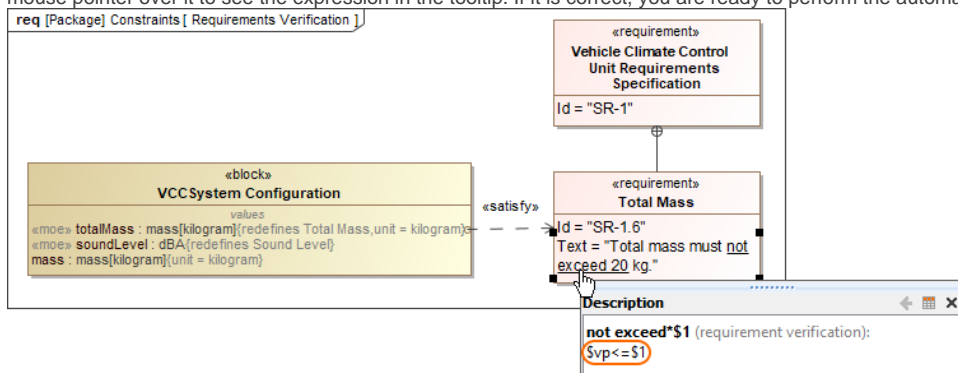
To get ready for automated Requirements verification

1. Define the constraint in the Requirement text.
2. Create a Satisfy relationship from the Value Property to the Requirement.



The Value Property captures the system parameter whose value determines whether the system requirement is satisfied or not.

3. Do one of the following:
 - If the **Underline Patterns in Requirement Text** option is enabled, the condition pattern in the Requirement text is underlined. Move the mouse pointer over it to see the expression in the tooltip. If it is correct, you are ready to perform the automatic Requirement verification.

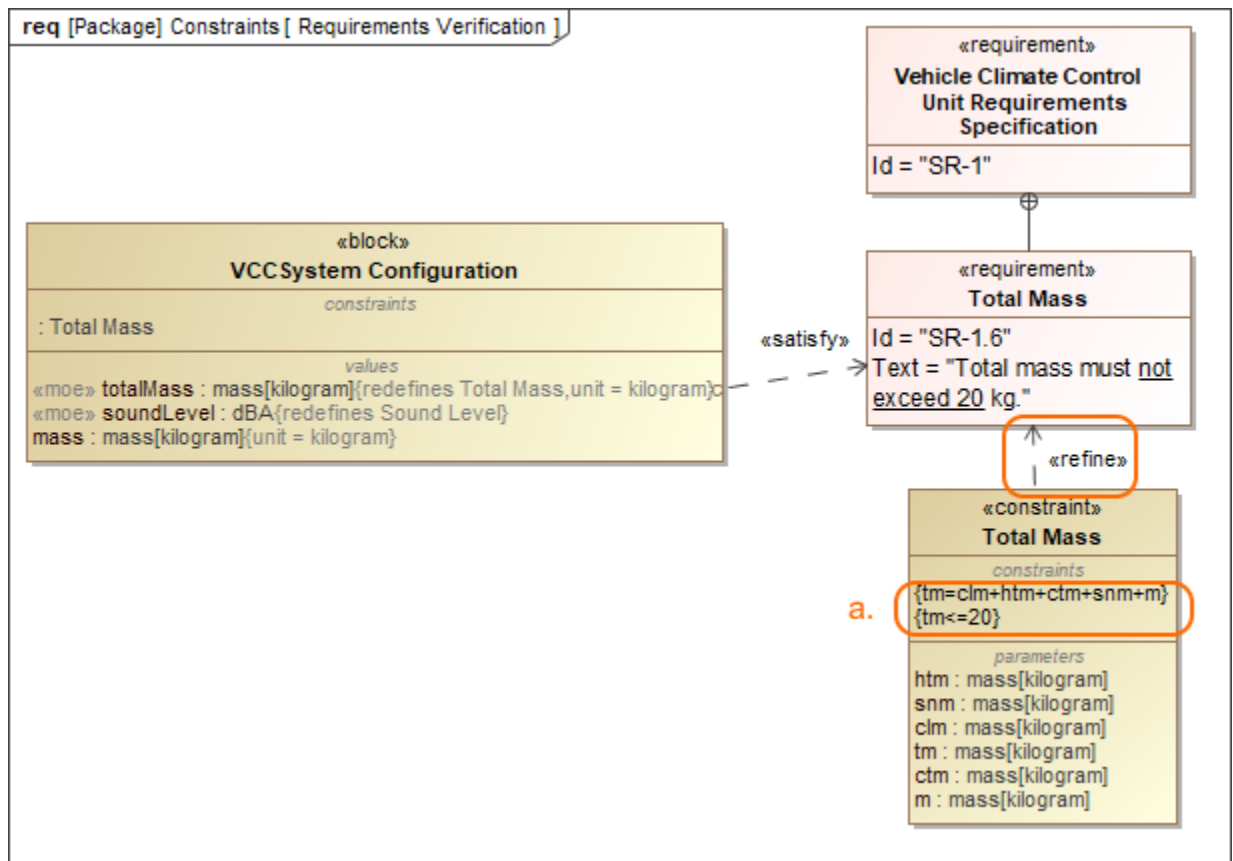


Learn more:

- [How to enable/disable the Underline Patterns in Requirement Text option.](#)

- If the expression in the tooltip is not correct, do the following:
 - [How to define custom condition patterns.](#)
 - a. Perform either of the two options:
 - Right-click a value property in the compartment area of the element shape.
 - [How to extract the Constraint from Requirement text.](#)
 - b. Select **Tools > Extract Constraint Block From Requirement** to automatically create a Constraint Block.
 - c. You can modify the constraint expression and the parameters as needed (see Figure A below).

You are now ready to perform the automatic Requirement verification.



In the example above, a Constraint Block with the constraint $\{totalMass \leq 20\}$ and the constraint parameters is automatically created. The constraint expression is then modified to $\{tm=clm+htm+ctm+snm+m\}\{tm \leq 20\}$.

Performing automated Requirements verification in Requirement Table

The automated Requirements verification analysis can be done directly in the Requirement Table. The analysis is performed by evaluating whether the value of the property that satisfies the Requirement falls within a range of upper and lower bounds that are extracted from the Requirement text. Additionally, the automatically calculated margin value helps to determine how close the system model is to fulfilling Requirements.

To perform automated Requirements verification

1. [Create](#) a Requirement Table. You can also use an existing Requirement Table.
2. [Specify the scope](#) for the table.
3. Set the context element (i.e., a Block as the table context) to perform the context-specific analysis.

The Requirement Table with the *Property*, *Bounds*, *Value*, and *Margin* columns is created, and passing/failing Requirements are marked (see an image below).

Criteria

Scope (optional): REGULAR Requirements,COMMON_Requirements {39} ... Filter:

Context (optional): SUV_REGULAR ...

Requirement Verification: ☐ Pass ☐ Fail

#	△ Name	Text	Property	Bounds	Value	Margin
1	1 SUV_REGULAR Requirements					
2	1.1 Spring Coils	Spring shall have <u>less than 8</u> coils.	suspension.spring.coils : Real	<8	7	1
3	1.2 Spring Deflection Distance	Spring shall have <u>not more than 108</u> -mm deflection distance.	suspension.spring.deflectionDistance : diameter[metre]	<=108	132	-24
4	1.3 Spring Free Length	The spring shall <u>have a free length of 200</u> mm.	suspension.spring.freeLength : distance[millimetre]	=200	160	-40
5	1.4 Spring Outer Diameter	The diameter shall be <u>less than 105 mm and more than 95</u> mm.	suspension.spring.outerDiameter : diameter[millimetre]	(95;105)	85	-10
6	1.5 Shock Absorber Length	Overall shock absorber length shall be at <u>maximum of 600</u> .	suspension.shockAbsorber.length : distance[millimetre]	<=600	450	150
7	1.6 Shock Absorber Weight	Shock absorber shall weight <u>not more than 4</u> kg.	suspension.shockAbsorber.weight : mass[kilogram]	<=4	3	1
8	1.7 Tire Diameter	The tires shall <u>have 18</u> -inch rolling diameter.	suspension.wheel.tire.diameter : Integer	=18	17	-1
9	1.8 Tire Height	The tire height shall be <u>not less than 45 and not more than 60</u> .	suspension.wheel.tire.height : distance[millimetre]	[45;60]	50	5
10	1.9 Tire Width	The tire width shall be <u>between 205 and 270</u> millimeters.	suspension.wheel.tire.width : distance[millimetre]	[205;270]	185	-20
11	1.10 Rotor Diameter	The brake rotors shall <u>not exceed 0.28</u> meter diameter.	brake.rotor.rotorOuterDiameter : diameter[millimetre]	<=0.28	0.29	-0.01
12	1.11 Pad Center Length	The Pad Center Length shall be <u>between 0.075 and 0.14</u> meters.	brake.pad.padLength : length[metre]	[0.075;0.14]	0.15	-0.01
13	1.12 Brake Pad Life	Brake pads shall have a projected life of <u>at least 57500</u> km.	brake.pad.padLifeSpan : distance[kilometre]	>=57500	90000	32500
14	1.13 Pad Width	The Pad width shall be <u>more than or equals 45e-3 and less than 65e-3</u> meters.	brake.pad.padWidth : diameter[metre]	(0.045;0.065)	0.042	-0.003

Using the Requirement patterns mechanism, the constraint extracted from the Requirement text is shown in the **Bounds** column. The **Value** column shows the **initial value** (if it exists) or the default value of the Value Property. Finally, the **Margin** column displays the difference between the calculated and required values.

Verification context in tables

- If the table **context** is left unspecified, properties satisfying the Requirements are collected from the entire model.

Requirements verification limitations

Table context specification limits

The table **context** should be specified with a Block that is recursively composed of no more than 10,000 parts. Otherwise, the **Property**, **Value**, and **Margin** columns are displayed as empty, and the verification analysis is not performed. In such a case, error icons are displayed in the column headers (see below).

You can also carry out Requirements verification analysis in an Instance table. While the verification in a Requirement table allows you to perform the analysis for multiple instances at once via properties provided by the classifier specified for the table, the verification in an Instance table allows you to perform the analysis for a single instance.

To perform automated Requirements verification in Instance Table, you need to select the context that is recursively composed of no more than 10,000 parts.

Requirements refined by Constraint Blocks

- Create an Instance Table.
- Specify the classifier for the table (i.e. the Block containing the properties satisfying the requirements).
- Set the scope for the table or add individual instances to the table.
- Select the properties to be displayed in table columns by Constraint Blocks.

5. In the table toolbar, click the **Options** button and select **Enable Patterns-Based Verification** to enable the verification. The simulation configuration option is selected and performs the verification, which is visible via the highlighted cells based on their verification status (definition is specified in the Legend).

If an Instance is defined as the Requirement Table context, the Requirements verification analysis does not consider the Simulation configuration option.

Criteria

Classifier: SUV REGULAR

Scope (optional): Instances

Filter:

Remember Failure Status

Verification Status: ☐ Pass ☐ Fail

#	Name	suspension.spring.coils	suspension.spring.deflectionDistance	suspension.spring.outerDiameter
1	suv_regular_v2	8	112	80
2	suv_regular_v1	9	100	85
3	suv_regular_v6	9	110	101
4	suv_regular_v5	9	100	101
5	suv_regular_v3	6	108	100
6	suv_regular_v4	7	90	106

Requirement 1.1 - "Spring shall have less than 8 coils." is satisfied.

Requirement 1.1 - "Spring shall have less than 8 coils." is satisfied.


Performing automated Requirements verification using simulation

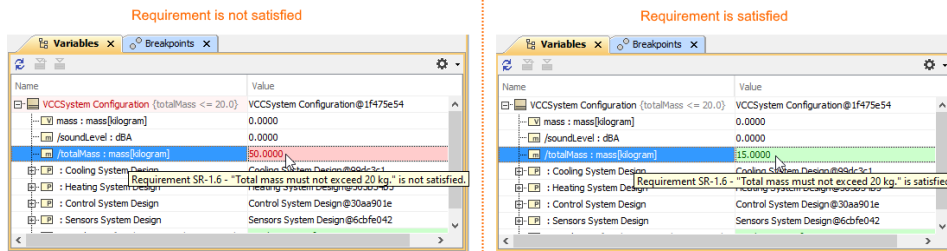
With the help of simulation, you can perform automatic Requirements verification.



To perform automatic Requirements verification, you must have the Cameo Simulation Toolkit installed. [How to install >>](#)

To perform automatic Requirements verification using simulation

1. Right-click the Block, which contains the Value Property.
 2. From the shortcut menu, select **Simulation > Run**.
 3. In the **Question** dialog, click **Yes** to load the validation rules and validate the model before the simulation or **No** to simulate the model without validating it.
 4. In the Simulation window, click  or press F8 to start the simulation.
- The result indicating whether or not the value is satisfied is shown in the Variables pane. In the following figure, you can see when the Requirement is not satisfied (highlighted in red) and satisfied (highlighted in green). You can change the value directly in the Value cell, and the Requirement constraint is checked automatically.



Additional features of Cameo Simulation Toolkit
[Learn more about how to perform Verification for a single element >>](#)

[Learn more about how to validate the model against a set of validation rules before executing it >>](#)

Webinar: Automated Requirements Verification