Trade study analysis

A trade study or trade-off study is the activity of a multidisciplinary team to identify the most balanced technical solutions among a set of proposed viable solutions (System Engineering Manual, Federal Aviation Administration, 2006).

A trade study is used to compare with a number of alternative solutions to see whether and how well they satisfy a particular set of criteria. Each solution is characterized by a set of measures of effectiveness (often abbreviated “moe’s”) that corresponds to evaluation criteria and has a calculated value or value distribution. The moe’s for a given solution is then evaluated using an objective function (often called a cost function or utility function), and the results for each alternative are compared to select a preferred solution.

Cameo Simulation Toolkit has built-in support for trade study analysis. The CarBreakingAnalysis sample model is used as a demonstration for trade study analysis through the following steps.

1. Use **TradeStudy «Block» «Analysis»** through the following step:
   - From the **MD Customization for SysML::analysis patterns::trade study** package, drag **TradeStudy «Block» «Analysis»** into the Block Definition diagram (BDD).

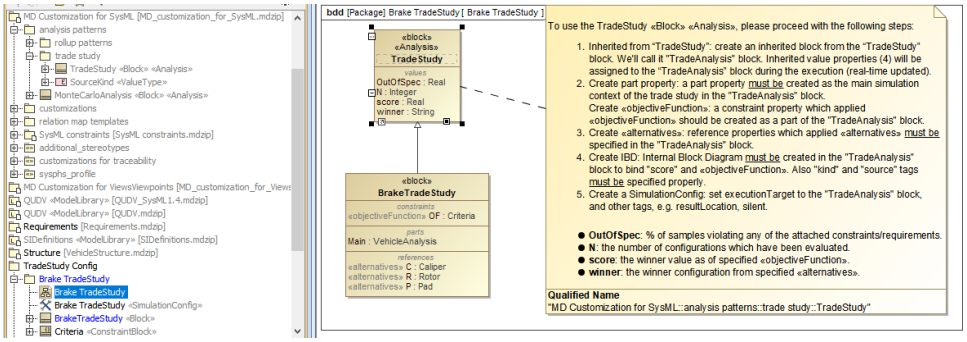
     > **Note**
     >
     > The **TradeStudy** package is available in all SysML projects. If the **MD Customization for SysML** package is not visible, click in the Containment tree pane and select **Show Auxiliary Resources**.

     The **TradeStudy** Block contains the following four value properties which will be inherited by the *TradeAnalysis* Block.

   - **OutOfSpec: Real**: counts how much percent of sample value that violates any attached constraints or Requirements.
   - **N: Integer**: the number of configurations evaluated.
   - **score: Real**: set as the **winner** value of the «objectiveFunction».
   - **winner: String**: used to keep the **winner** configuration as a string.
2. Create the *TradeAnalysis* Block by doing the following:
   a. Inherit the **TradeStudy** (Block).
   Create a new Block and Generalization from the **TradeStudy** Block so that the new *TradeAnalysis* Block is inherited, e.g., an instance of *BrakeTradeStudy*. Those four inherited value properties will be assigned to the *TradeAnalysis* Block and updated real-time during the execution.
   b. Add main simulation context as Part.
   Add the main Block of simulation context as a Part property of the *TradeAnalysis* Block, e.g., *Main : VehicleAnalysis*.
   c. Create **«objectiveFunction»**.
   An objective function is a special type of constraint Block used to determine the overall value of any weighted alternatives in terms of weighted criteria. You must create a constraint Block containing related constraint parameters and a constant expression for determining the best weighted value, e.g., *Criteria*.

   > **Note**
   >
   > - The specification of the constraint must be an equation with LHS = RHS, where LHS contains only one parameter to bind with *TradeAnalysis::^score*, and RHS can contain multiple parameters (bound with corresponding value properties) to evaluate as **winner** criteria.
   > - The output of **«objectiveFunction»** will be compared with previous weighted results. If an alternative is greater, it will be set as the **winner** (maximum value by default). However, if you want the minimum value, use a negative value, e.g., *value = -distance*.

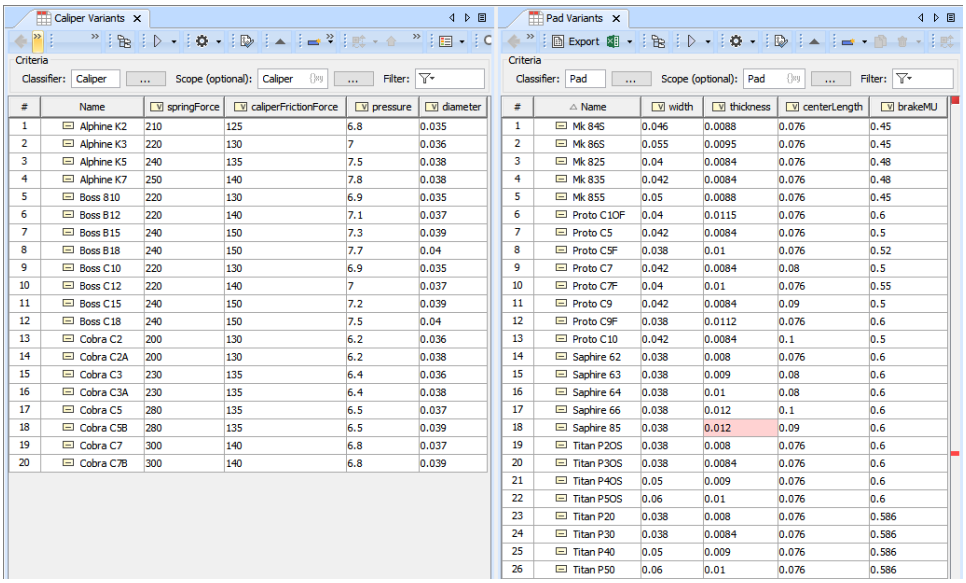
   d. Add **«objectiveFunction»**.
   Create the constraint property in the *TradeAnalysis* Block typed by the created constraint Block, and apply **«objectiveFunction»**, e.g., *O F : Criteria*.
   e. Create **«alternatives»**.
   Create reference properties typed by a Block of alternatives and apply **«alternatives»** (*Qualified Name*: *MD Customization for SysML*::*ad ditional_stereotypes*::*alternatives*) to those properties, e.g., *C : Caliper*, *R : Rotor*, and *P : Pad*, as shown in the figure below.



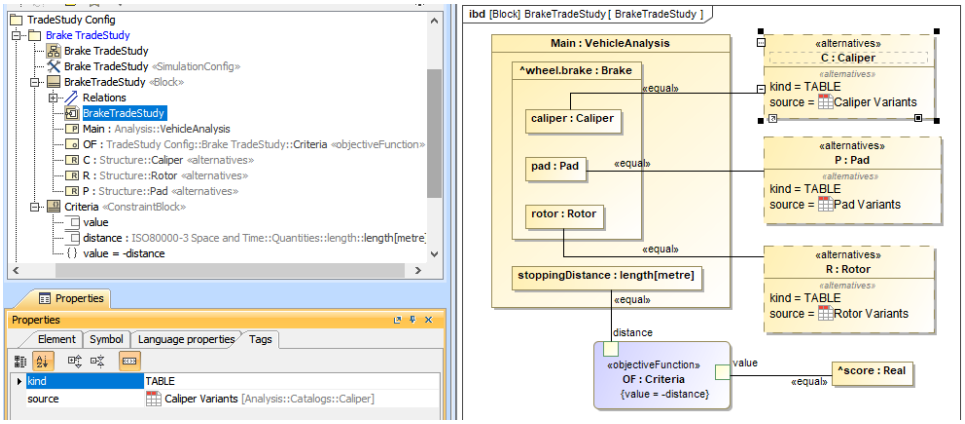
Structure of the TradeAnalysis Block.

3. Create binding of the *TradeAnalysis* Block through the following steps:

a. Create an Internal Block diagram (IBD).
   Create an IBD of the *TradeAnalysis* Block. You must select Parts which have been set as **«objectiveFunction»** and **«alternatives»** to display. You must also display *^score* (inherited property) in the diagram.
b. Bind **«objectiveFunction»**.
   You must bind *^score* with a Binding Connector to the LHS parameter of **«objectiveFunction»**, e.g., *^score – value*. You must also bind a value property of the main simulation context with a Binding Connector to the RHS parameter of **«objectiveFunction»**, e.g., *stoppingDistance – distance*.
c. Bind **«alternatives»**.
   You must bind each **«alternatives»** with a Binding Connector to each Part property.
d. Set kind/source tags.
   For each **«alternatives»**, **source** depends on the following **kinds** through the **Tags** settings.
   - **kind = TABLE**: **source** must be an instance table which must be the same Classifier as the alternative property, as shown in the two figures below. However, sorting of rows in the table is not necessary.

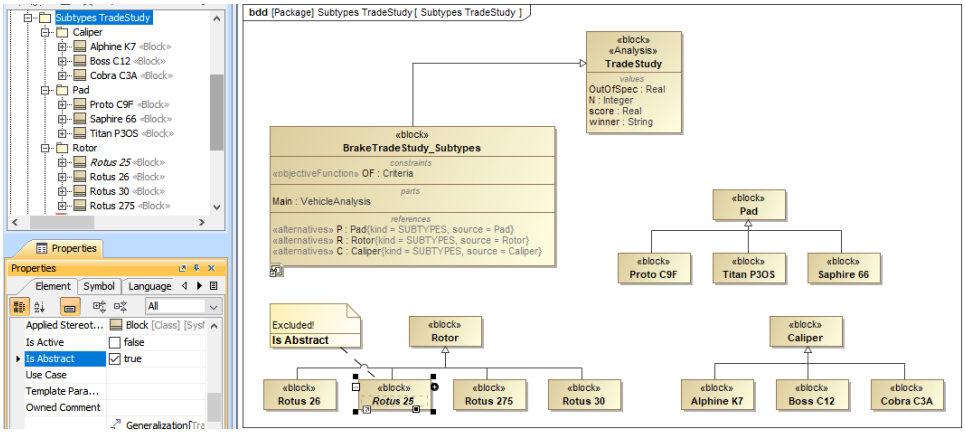


TABLE, kind of «alternatives», must be the same Classifier as the alternative property.

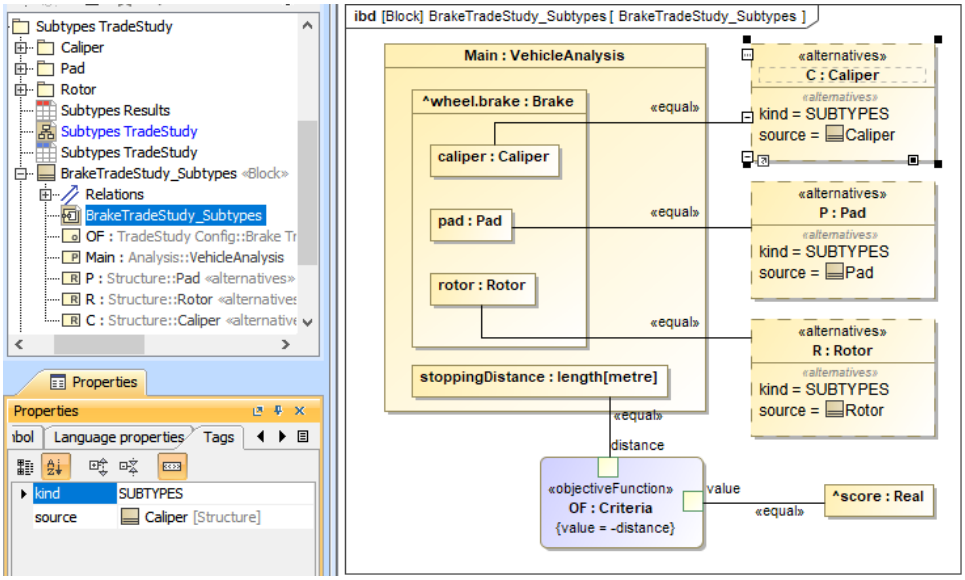


Binding of the TradeAnalysis Block in the Internal Block diagram (kind = TABLE).

- **kind = SUBTYPES**: **source** must be a parent Block of subtypes which must be the same type as the alternative property, as shown in the two figures below. Also, the parent Block will not be evaluated as well as any Blocks which have *Is Abstract* = true.

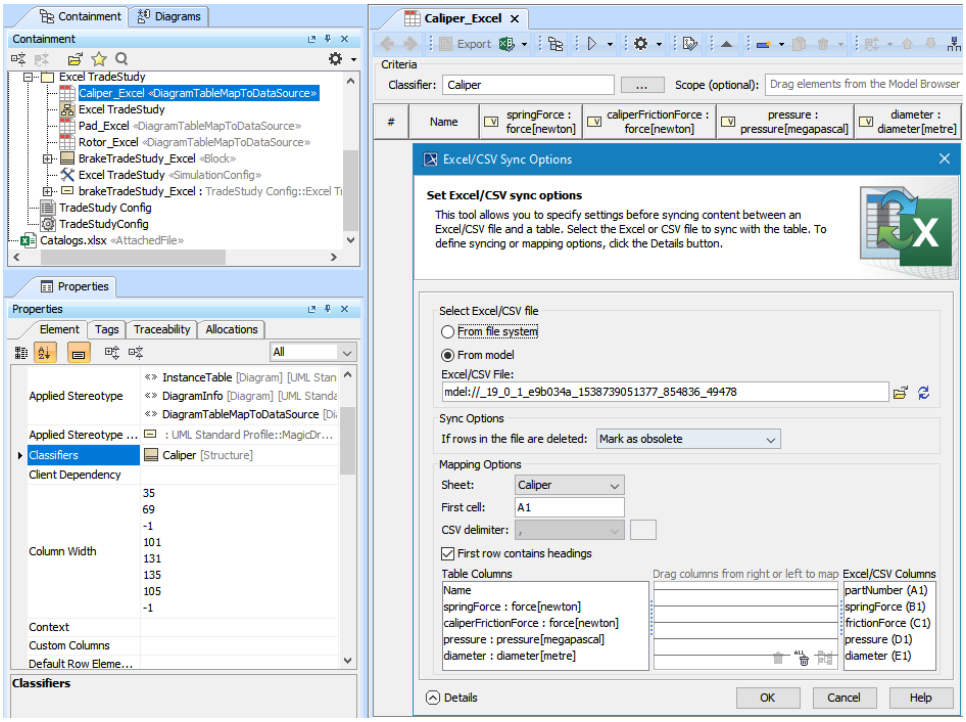


SUBTYPES, kind of «alternatives», must be the same type as the alternative property.

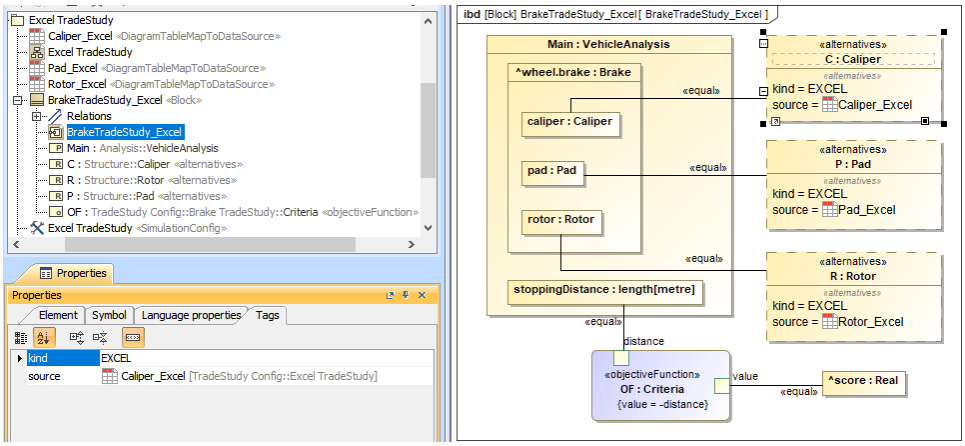


Binding of the TradeAnalysis Block in the Internal Block diagram (kind = SUBTYPES).

- **kind = EXCEL**: **source** must be an instance table linked to an Excel file, and table columns must be mapped to Excel/CSV columns as shown in the two figures below (see also Sync with Excel or CSV files). However, you do not need to use the **Read from File** command from the **Publish** Excel toolbar to load data into the table. The Excel Import plugin is required for the **EXCEL** kind (see also Excel Import Plugin Documentation).



EXCEL, kind of «alternatives», must be linked to an Excel file, and table columns must be mapped to Excel/CSV columns.



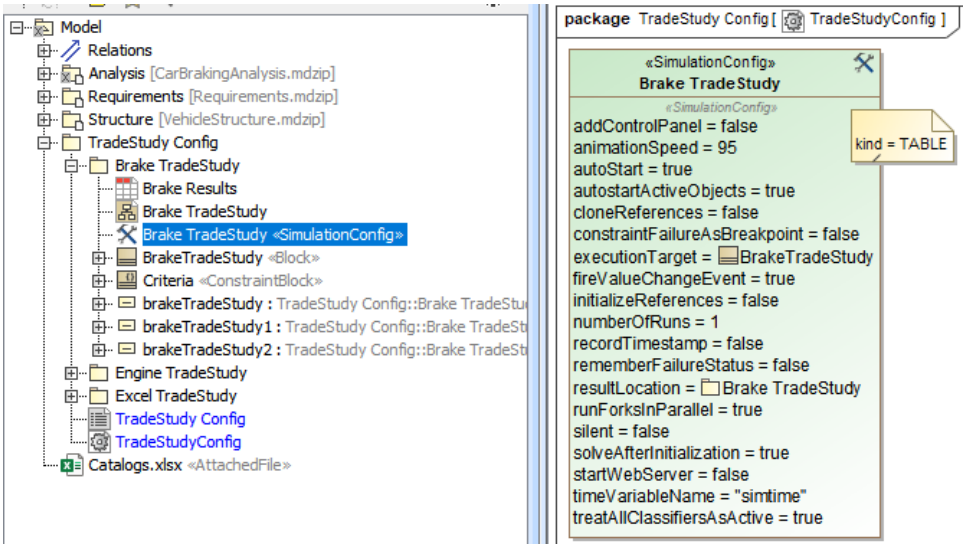
Binding of the TradeAnalysis Block in the Internal Block diagram (kind = EXCEL).

4. Create SimulationConfig and other settings.
   You can create a SimulationConfig, so that it can specify the **resultLocation** of this **TradeStudy** to a package, e.g., instances of the winning configuration will be saved. Also, the following tags can be set, as shown in the figure below.
   a. **executionTarget**
      executionTarget must be set to the *TradeAnalysis* Block, e.g., *BrakeTradeStudy* in the sample. *executionTarget* can be an instance of the *TradeAnalysis* Block so that the instance can be reconfigured.
   b. **resultLocation**
      A package/instance table must be specified so an instance with related information will be saved after running the simulation. The information includes **N**, **OutOfSpec**, **score**, **winner**, and other elements.

> **Note**
>
> - If you do not create a SimulationConfig and run the *TradeAnalysis* Block directly, **TradeStudy** will not be triggered, but the *TradeAnalysis* Block will be run as normal .
> - If the **executionTarget** of a SimulationConfig is the *TradeAnalysis* Block, **TradeStudy** execution will override other executions, so Monte Carlo simulation will not be triggered even if **numberOfRuns** is more than 1.

   c. **silent**
      You can set the **silent** tag accordingly to see the animation.
   d. **rememberFailureStatus**
      You can use *rememberFailureStatus* when evaluating those weighted alternatives. Any alternatives violating any of the attached constraints/Requirements will not be considered as the **winner**, but they will be used in the calculation of **OutOfSpec**.
   e. Tags neglected

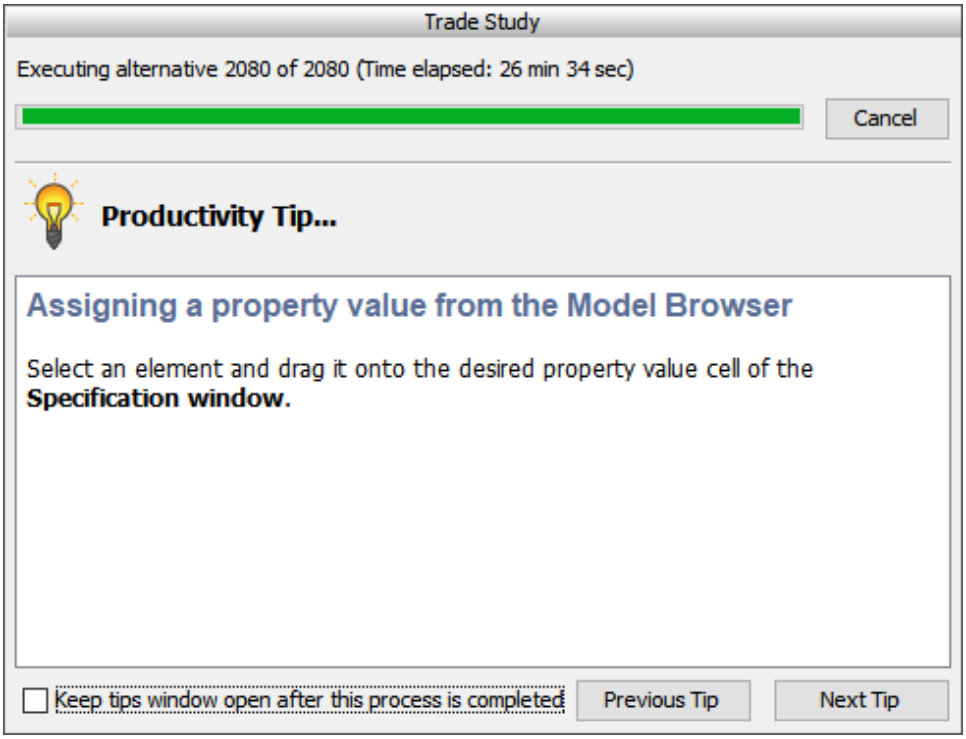
You can neglect **animationSpeed**, **constraintFailureAsBreakpoint**, **UI**, and **autoStart**.



SimulationConfig created for other settings, e.g., executionTarget and resultLocation.

5. Run the SimulationConfig.
   When running the SimulationConfig, the information of **TradeStudy** will be printed in the **Console** pane. The Simulation pane will be disabled, but all warning/errors will be printed. There is a progress bar shown with the description of **Executing alternative [n] of [total iterations] (Time elapsed: [time])**, and the **Cancel** button that allows canceling **Trade Study** as shown in the figure below. Simulation Toolkit automatically iterates all variants and instantiates all possible configurations in memory. For example, *Brake* which contains the combinations of *Pad* (26 instances), *Caliper* (20 instances), and *Rotor* (4 instances) will have 26 x 20 x 4 = 2,080 configurations. The **winner** value on each iteration will be compared directly with the **score** value property.

> **Note**
>
> - The **Starting Math Engine** progress bar will be shown in this sample because MATLAB is used as the external evaluator. See also Integration with MATLAB.
> - Any alternatives violating any of the attached constraints/Requirements will not be considered as the **winner**. They will be used in the calculation of **OutOfSpec**.
> - **rememberFailureStatus** of a SimulationConfig will be used when evaluating those alternatives.
> - In some cases, you can click **Unlock** in the **Simulation** pane to see execution details during the execution.



A progress bar shown with description and the Cancel button during the simulation.

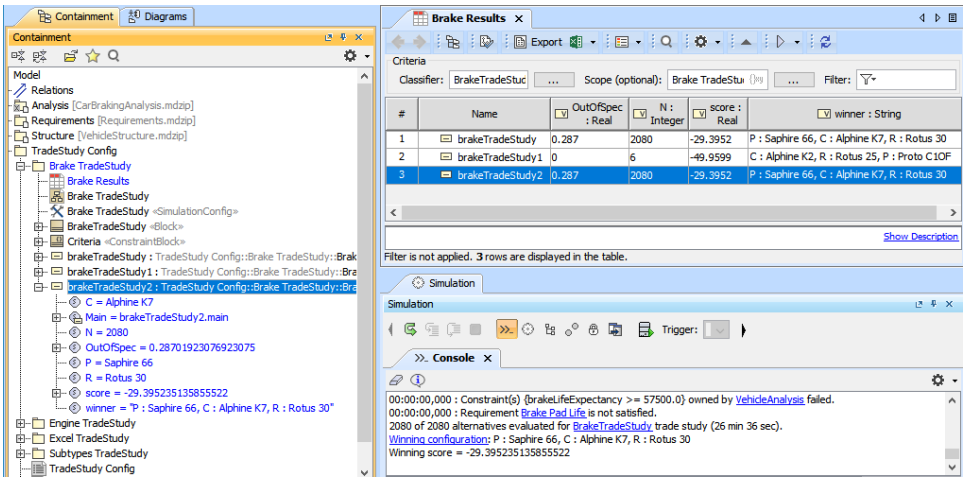
6. Get the **TradeAnalysis** result.

When the simulation is either completed or canceled, **winning** information will be printed on the **Console** pane in the following three lines as shown in the figure below.

- The first line shows the number of iterations (completed/canceled) of all alternatives for ***executionTarget*** with elapsed time.
- The second line displays the *winning* configuration from the **winner** string.
- The third line is the **winning score** from the *^score*.

> **Note**
>
> The **winner** string is printed with the formats as follows:
> *AlternativeProperty.Name1 : StringKind [, AlternativeProperty.Name2 : StringKind, AlternativeProperty.Name3 : StringKind, ...]*
>
> where *StringKind* will apply the following rules, depending on **kind** of alternative:
>
> - *kind=TABLE*: then *StringKind*=InstanceName, e.g., P : Saphire 66, C : Alphine K7, R : Rotus 30.
> - *kind=SUBTYPES*: then *StringKind*=subtypesName, e.g., power : Diesel, support : Wheels, stopping : Brakes.
> - *kind=EXCEL*: then *StringKind*=#Row, e.g., R : #5, P : #21, C : #21, where *Row* is the number of Excel rows.

The result instance will be saved at the location as specified in **resultLocation** of SimulationConfig. You can create an instance table, set a Classifier to the *TradeAnalysis* Block, and set *Scope* to the package of the results.



The TradeAnalysis result (in the last 3 lines) is printed on the Console pane, and the result instance is saved into a package and presented in the instance table.