SIEMENS

Mechatronics Concept Designer

Quickstart

and

User Manual

May 12th, 2010

Table of Contents

1 Introduction and Overview 5

1.1 Purpose of this document 5

1.2 What is “Mechatronics Concept Designer“? 5

1.2.1 Functional Machine Design 5

1.2.2 Early System Validation 6

1.2.3 Multi-Disciplinary Support 8

1.2.4 Modularity and Reuse 9

1.3 How does “Mechatronics Concept Designer” fit into the design workflow? 10

2 Quick Start 13

2.1 The Simulation Engine” 14

2.2 Simple robot example 18

2.2.1 Manage requirements 18

2.2.2 Functional decomposition 19

2.2.3 Link requirements and functions 20

2.2.4 Implement functions 21

2.2.5 Simple Kinematics 23

2.2.6 Actuators 25

2.2.7 Time based Operations 27

2.2.8 Sensors 28

2.2.9 Event based operations 30

2.3 Cam example 31

2.3.1 Define a cam 31

2.3.2 Inspector 34

3 The navigators 36

3.1 The Function Navigator 36

3.2 The Physics Navigator 38

3.3 The Sequence Editor 41

4 Basic Features 43

4.1 Define design requirements 43

4.2 Create Functional Model 43

4.3 Define Mechanical Concept (Geometry) 44
4.3.1 Define basic principles of the machine based on the rough geometries 44
4.3.2 Create association link between components and function model 44
4.4 Define Mechanical Concept (Physics) 46
4.4.1 Add Physics Properties 46
4.4.2 Add Kinematic Behavior 56
4.4.3 Constraints 68
4.5 Add Generalized Actuators 70
4.5.1 Speed Control 71
4.5.2 Position Control 71
4.5.3 Export Sensor and Actuator List 72
4.6 Define time based controllers 73
4.6.1 Define operations 73
4.6.2 Define sequence of operations 74
4.7 Continuous behavior 75
4.7.1 Define a Motion Profile 75
4.7.2 Define a gear 78
4.7.3 Define a cam 79
4.8 Add Generalized Sensors 80
4.8.1 Define a Collision Sensor 81
4.8.2 Continue to build the sensor/actuator list 83
4.9 Add event based behavior 83
4.9.1 Add event based operations 83
4.9.2 Add a behavioral object 86
4.10 Knowledge capture and efficient design based on reusable objects 89
4.11 Monitor the simulation 91
4.12 Preferences 91
4.12.1 General 91
4.12.2 Physics Engine 92
4.12.3 Inspector 93
4.13 Instable Simulation 93
5 Support other design disciplines 95

5.1 Systems Engineering and Functional Modeling 95
5.2 Mechanical Design 96
5.3 Electrical Design 96
5.4 Software Design 96
6 Addendum 97
6.1 EN 61346-2 97
7 Contact 103

Referenced documents

(1) „Teamcenter 8.1 Getting Started with Teamcenter", Publication Number PLM00002 D getting_started_teamcenter.pdf
(2) "Teamcenter 8.1 Requirements Manager Guide", Publication Number PLM00038 C requirements_manager.pdf
(3) "VDW Richtlinie Funktionsbeschreibung", VDW Corneliusstraße 4, 60325 Frankfurt am Main, Phone: 069-756081-0, vdw@vdw.de
(4) „NX Basics for Mechatronics", April 2010 nx_basics.pdf

1 Introduction and Overview

Thank you very for participating in the Early Adopter Program of Mechatronics Concept Designer. This program gives you the opportunity to apply Mechatronics Concept Designer to your application and by your feedback shape the future releases. This will help us to deliver a solution that perfectly fits to your demands.

1.1 Purpose of this document

This document will help you to get a quick overview of Mechatronics Concept Designer and gives you a quick start to efficiently work with it.

Mechatronics Concept Designer is based on the NX CAD platform and offers many features that are used for sophisticated CAD design. To learn about the basic CAD design features like sketching, 3D-modeling, assembly management or data import and export you might have to refer to the corresponding NX documentation.

This document will give you an overview how to perform a functional machine design approach with Mechatronics Concept Designer and Teamcenter. Both a closely linked together and offer a great solution to design you machine starting with requirements, map them to a functional model and evaluate concept alternatives that are the foundation for your detailed design.

1.2 What is "Mechatronics Concept Designer"?

1.2.1 Functional Machine Design

Mechatronics Concept Designer is a solution that transforms the machine creation process into an efficient mechatronics design approach, so it should not be seen as an isolated product. This will significantly reduce the time to market. One of the main instruments in this is the functional model, which forms the foundation to provide an interdisciplinary view of the "mechatronics system" machine.

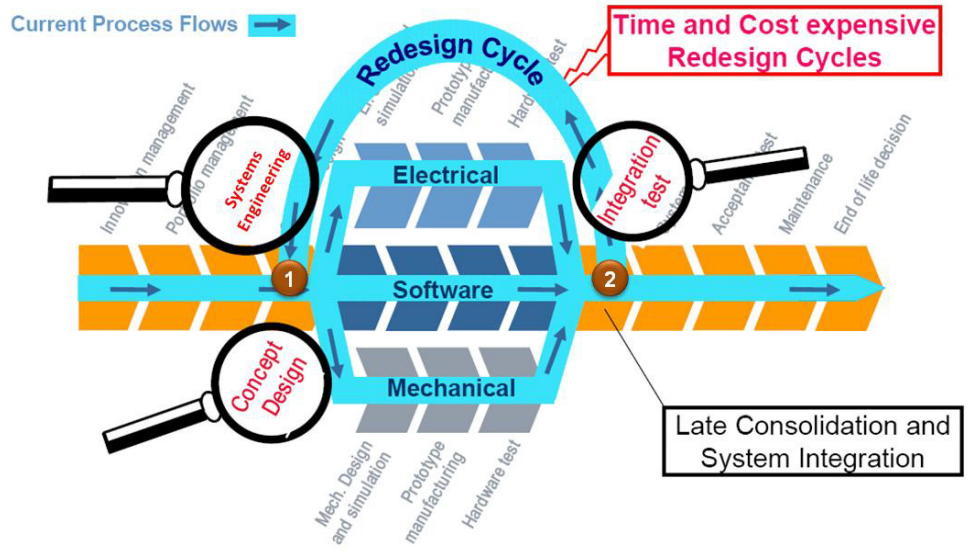


Figure 1: Overall Design Process

Figure 1 shows the typical steps in the design process of a mechatronics system. Many problems in an interdisciplinary design context occur when the disciplines meet each other when the whole system is integrated near the end of the process(2). In many cases, these problems are caused by the loose connection of the detailed design disciplines, including Mechanical, Electrical/Fluid and Software. The different departments do not collaborate to synchronize their work.

Mechatronics Concept Designer will help to lay the foundation for collaboration in detailed design by supporting the early design phase (1) with a functional design approach.

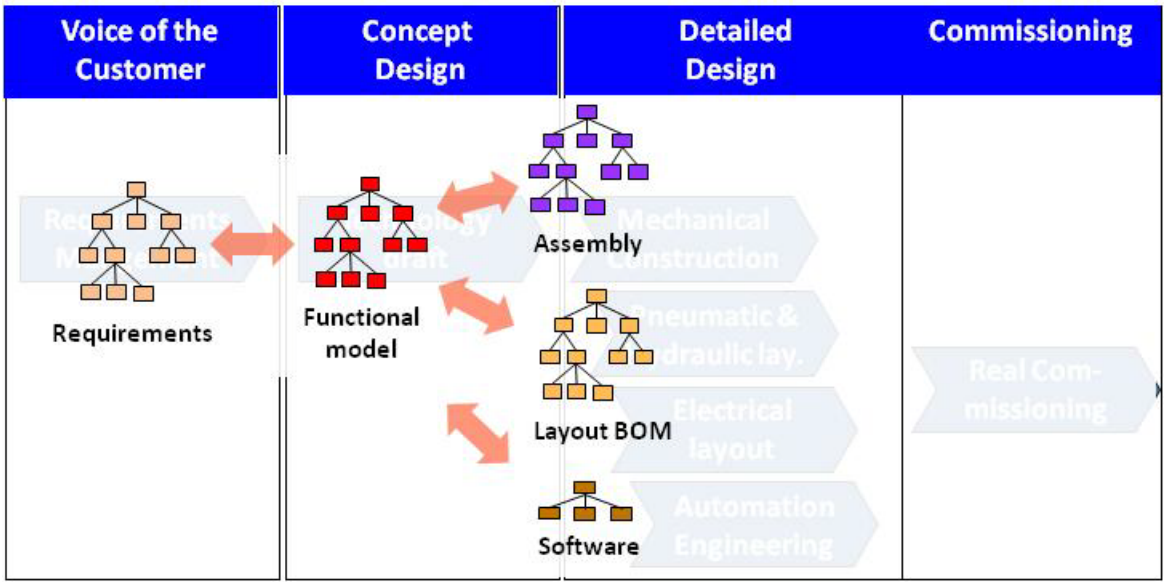


Figure 2: The functional data structure

The functional model provides the link between the data management of the different disciplines and the requirements. This enables the traceability of the customer demand data down to the design departments. Further, the functional model provides a supporting structure to come up with initial design concepts and has the features to perform an evaluation of design alternatives.

1.2.2 Early System Validation

In addition to this, Mechatronics Concept Designer introduces a verification technology that is built on a new simulation engine. This will help to validate concept designs at a very early stage of the development process. This will save a lot of money since the costs to correct an error increase later in the development process.

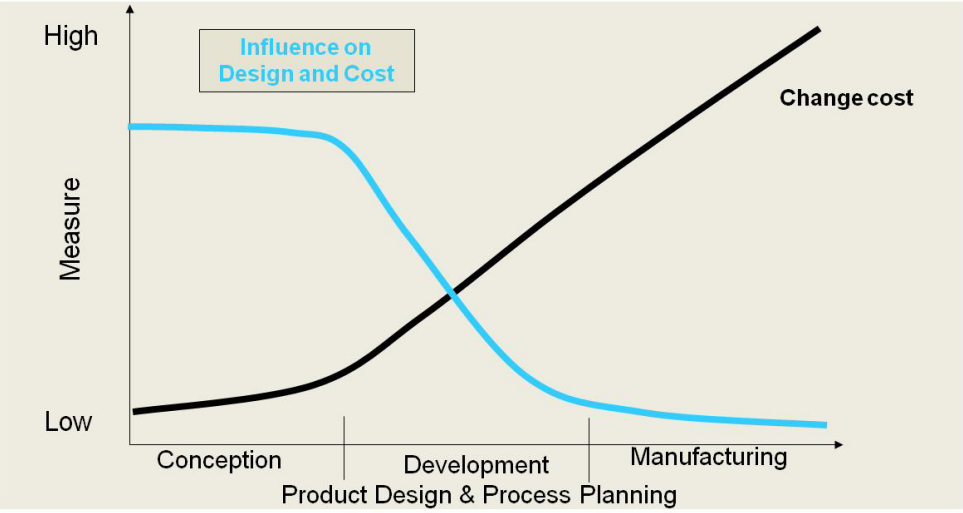


Figure 3: Increasing change cost over the maturity of the design

The simulation technology adds value in several ways:

- It is easy to use. The modeling of the physical world is very simple. In a few steps you get to a physical definition of your machine concept and your desired machine behavior.
- It covers kinematics, dynamics, collisions, actuators, springs, cams and much more. Everything you need to validate your machine concept.
- It has excellent performance. Just switch it on and you will immediately see your machine design working in real time.
- It is interactive. While the simulation is running you can use your mouse pointer to apply forces to objects on the screen. This will help to anticipate unforeseen behavior and see how the system handles errors.
- It supports material flow with multiple objects. During the simulation you can automatically generate new objects and introduce them to the system, transported by conveyers or processed by an operation.
- It gives you insight into all relevant physical values. Using the “inspector” during the simulation you can click on an object in the scene and will you get its physical runtime parameters like it’s actual position, speed, rotation and much more.

These advantages also come with some restrictions which one has to keep in mind when applying the simulation capabilities of Mechatronics Concept Designer.

- The simulation solver applies simplified equations. Therefore the accuracy of the simulation is not sufficient to make detailed multi-body analyses, such as frequency analysis.
- The collision shapes are approximated. The simulation engine does not provide mesh-to-mesh collision detection, it uses simplified bodies that come close to the original shape. Therefore Mechatronics Concept Designer should not be used for detailed collision analysis of complex, concave shapes.
- Mechatronics Concept Designer does not support flexible bodies (e.g. paper).

- Mechatronics Concept Designer does not simulate the transformation (e.g. melting) or deformation (e.g. material removal of a milling process) of bodies.

1.2.3 Multi-Disciplinary Support

Following this functional machine design approach, Mechatronics Concept Designer facilitates interdisciplinary concept design up front. The following disciplines can jointly work on a project:

- The mechanical engineer will create the design based on 3D shapes and kinematics.
- The electrical engineer will help to select and position sensors and actuators.
- The automation programmer will use Mechatronics Concept Designer to design the basic logical behavior of the machine, starting with time based behavior and then defining the event based control.

This is not duplicate work because Mechatronics Concept Designer provides output that can be reused for subsequent disciplines.

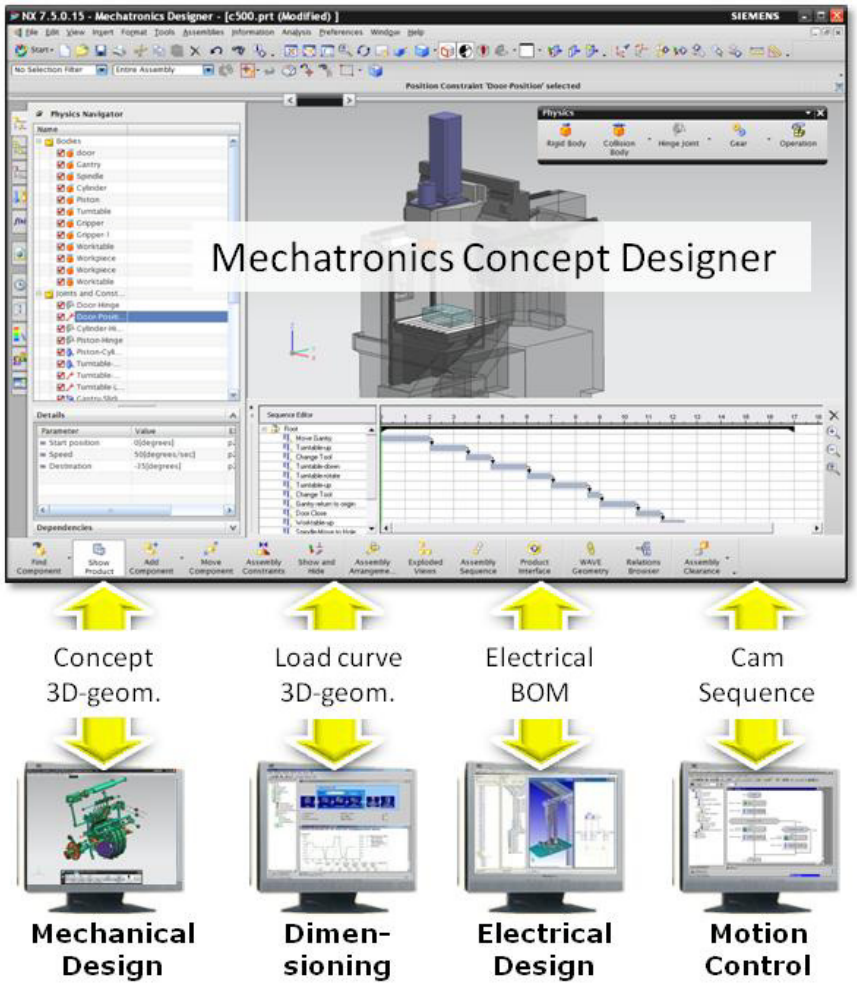


Figure 4: Output of Mechatronics Concept Designer

- **Mechanical Design:** Mechatronics Concept Designer is based on a CAD platform so it provides all the features that are needed for sophisticated CAD design. It is seamlessly integrated with NX but can export to many other CAD tools such as:
    - Catia V5
    - ProEngineer
    - SolidWorks
    - JT (CAD neutral)
- **Dimensioning:** Mechatronics Concept Designer can simulate the dynamics of a machine axis and export the load and speed cycle. This can be used in a dimensioning tool like "SINAMICS MICROMASTER SIZER" to support the selection of the best fitting motor.
  This feature is not supported in the current release.
- **Electrical Design**: During the concept design with Mechatronics Concept Designer, a Sensor and Actuator list will be built. This list can be exported and used in an ECAD tool to help build the layout.
  This feature is not supported in the current release.
- **Automation Design**: Mechatronics Concept Designer has the ability to define the Sequence of Operations in a Gantt Chart. This can be exported in a PLCOpen XML format which has been standardized by the AutomationML group. This can be reused for further detailing in the Automation Engineering.
- **Motion Control Design**: The continuous behavior of synchronized axes can be defined in functions that drive a cam. These functions can be exported in XML and reused in Motion Control Engineering tools for further refinement.
  This feature is not supported in the current release.

1.2.4 Modularity and Reuse

Modularity and Reuse are the keys to maximizing design efficiency. The ability to capture knowledge in components and store them in a library enables the reuse of this knowledge in other projects. This increases design quality because designs are based on already proven concepts and it speeds up development because one does not have to redo tasks which already have been performed. Mechatronics Concept Designer supports the definition of "functional units" such as those in the VDW Standard "Funktionsbeschreibung".

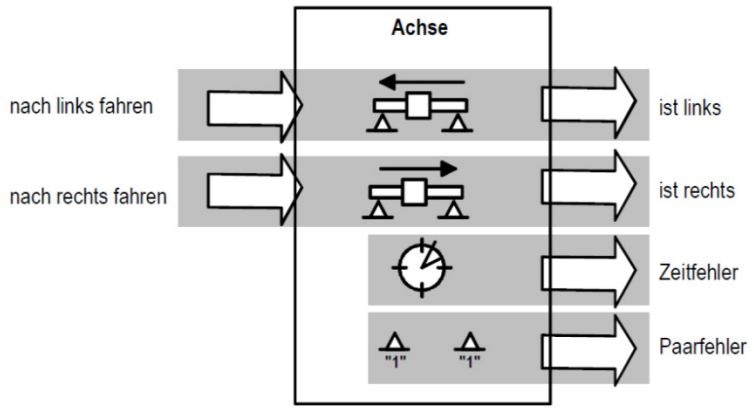


Figure 5: Picture from the VDW Standard "Funktionsbeschreibung"

With Mechatronics Concept Designer you can store mechatronics data in one part file (file extension .prt):

- Graphical 3D-data
- Physical data like kinematics, dynamics, …
- Sensors and Actuators with their interfaces
- Cams and functions that are performed by this functional unit
- Operations that are performed by the functional unit

By simply adding this part file to your design you include all of this information in your system. This is the foundation of a step-by-step process to build a library of proven, reusable objects.

1.3 How does "Mechatronics Concept Designer" fit into the design workflow?

As you saw in Figure 1, Mechatronics Concept Designer is designed to support the early design phase that provides the basic machine concept including the mechanical, electrical/fluid and software aspects.

Figure 6 gives an overview of the typical design steps that are performed in the early design phase. These are supported by the Mechatronics Concept Designer solution with Teamcenter.

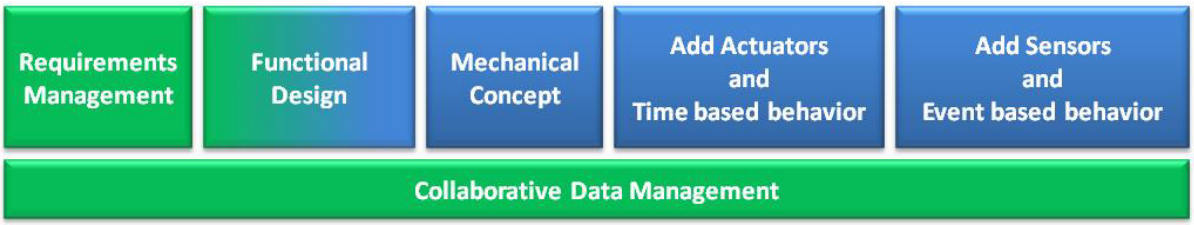


Figure 6: Basic workflow supported by Teamcenter (green) and Mechatronics Concept Designer (blue)

Mechatronics Concept Designer's output will be further refined in the detailed design disciplines. These will rely on existing tools that already exist in the different development departments.

Figure 7: Downstream disciplines that benefit from the data structures provided by Mechatronics Concept Designer and Teamcenter

The following table gives an overview of the typical design steps.

Action	Description	Tool
Requirements Management	• Gather requirements • Structure requirements, e.g. based on a template structure • Add "derived" requirements • Link requirements to each other • Add more details to requirements using embedded tools like MS WORD	Teamcenter
Functional Design	• Define basic functions of the system. • Create a hierarchy based on a functional decomposition • Create and maintain alternatives for the functional design • Reuse functional units	Teamcenter
Mechanical Machine Concept	• Define rough 3D outline of the basic solution concept • Assign mechanical implementation objects to functional tree • Add kinematics and dynamics	Mechatronics Concept Designer, (Teamcenter)

Add abstract Actuators	• Add two types of actuators:
	  ○ Speed constraint
	  ○ Position constraint
		Mechatronics Concept Designer

Define time based Operations	• Define how the actuators are controlled by operations • Arrange the sequence of operation with a time based notion • Assign operations to the corresponding functions in the function tree	Mechatronics Concept Designer, (Teamcenter)
Add Sensors	• Add sensors that are triggered by collisions of system elements with sensor objects	Mechatronics Concept Designer
Define event based Operations	• Define operations that are triggered by events generated by the sensors or other objects in the mechatronics systems (like the position of an actuator) • Assign operations to the corresponding functions in the function tree	Mechatronics Concept Designer, (Teamcenter)

Refine the mechanical design	• Refine the rough 3D outline of the solution • Create more detailed geometry • Assign the geometry to the corresponding functions in the function tree • Add additional elements like bolts, washers, holes, etc. *The following step is not supported in the current release of Mechatronics Concept Designer.* • Select the best fitting motors based on the simulation results from Mechatronics Concept Designer.	CAD design tool (e.g. NX CAD), (Teamcenter) SIZER
Create the electrical and fluid layout	*The following step is not supported in the current release of Mechatronics Concept Designer.* • Create the electrical and fluid layout based on the sensors and actuators defined in the concept design • Add further components that are needed to detail the layout plan • Manage the net list and electrical BOM in Teamcenter • Assign new items to the function tree • Perform 3D routing in the CAD tool • Execute consistency checks to synchronize electrical and mechanical data structures	ECAD tool, 3D routing tool (e.g. NX routing), (Teamcenter)
Develop the automation software	• Export Sequence of Operation in PLCOpen XML from MCD • Develop PLC code • Create drive configuration *The following step is not supported in the current release of Mechatronics Concept Designer.* • Define the detailed cams to synchronize multiple axes based on the exported cams from Mechatronics Concept Designer	Automation Engineering Tools, Drive Configuration Tools

2 Quick Start

This chapter will help you to quickly get familiar with Mechatronics Concept Designer. It will use example files that are provided with this Quick Start Guide. So you have the opportunity to try out these examples on your own Mechatronics Concept Designer.

Once you have started Mechatronics Concept Designer please make sure that the Mechatronics Concept Designer application is selected:

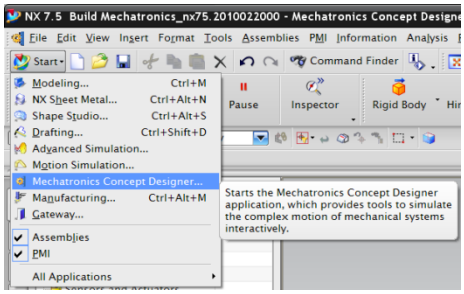


Figure 8: Mechatronics Concept Designer Application

In Teamcenter managed mode you will see the following example data in the Teamcenter database.

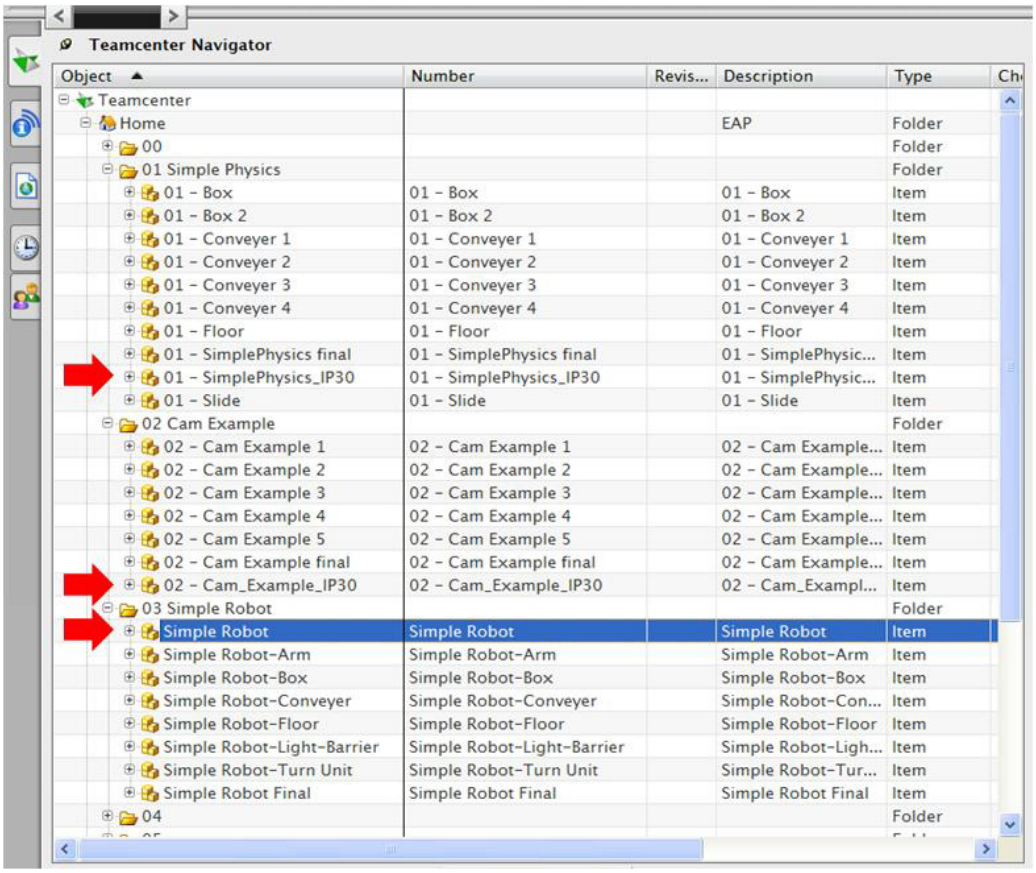


Figure 9: Mechatronics Concept Designer Application

The red arrows point to the files that are important for this demo.

2.1 The Simulation Engine"

For this exercise please open the file: "01 - SimplePhysics_IP30.prt"

Once you have opened this part file please select the "Physics Navigator".

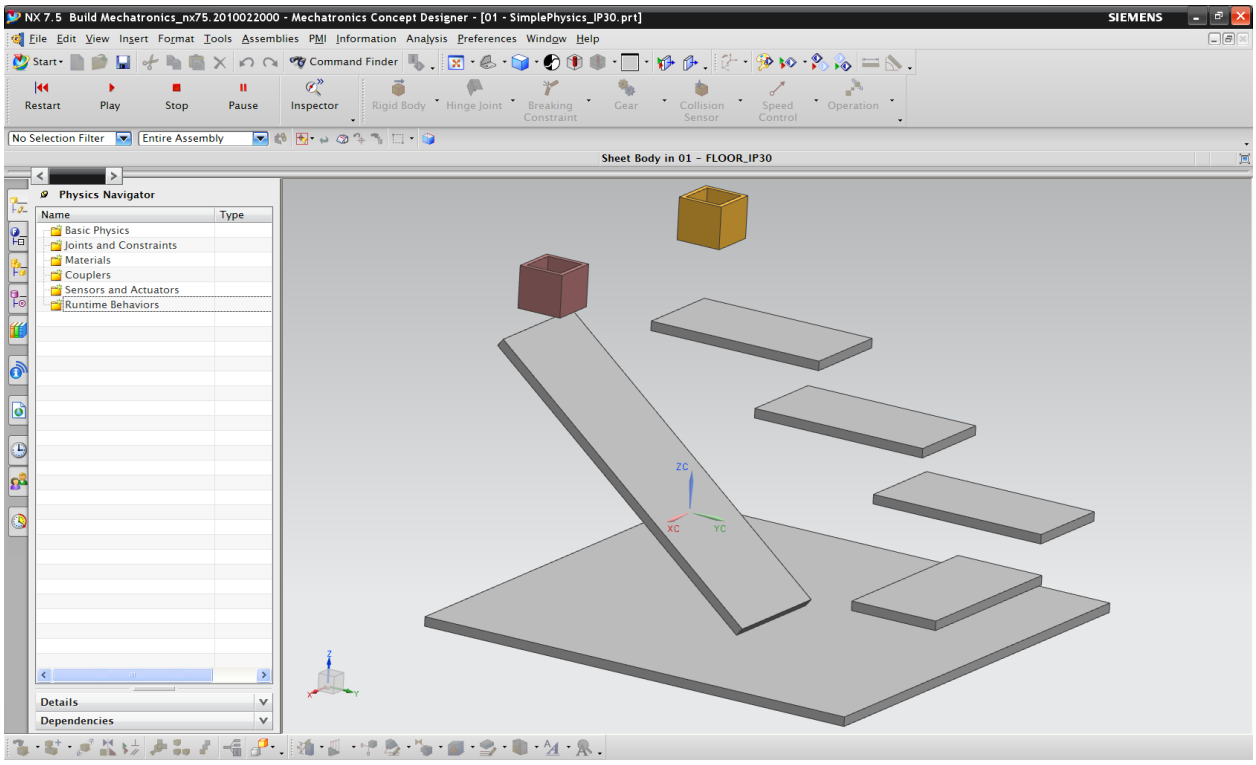


Figure 10: Physics Navigator

The Physics Navigator shows several structures of elements that define the physical behavior of the machine. In this example we will only deal with the "Basic Physics", namely the two elements "Rigid Body" and "Collision Body".

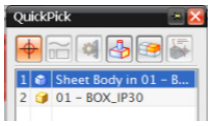
- A Rigid Body defines an element that is moveable. Once you assigned this to a 3D object in your scene it will respond to all the forces in your systems (e.g. gravity). Objects without a rigid body are completely stationary.
- A Collision Body defines how elements collide with other elements that also have a collision body. Objects without a collision body pass through other objects.

If you start the simulation 

nothing will happen.

Now assign a rigid body 

. Select the left box (1). Don't change any other parameters. Just assign the name "Box 1" (2). You might have to choose the type of object that you want to select:



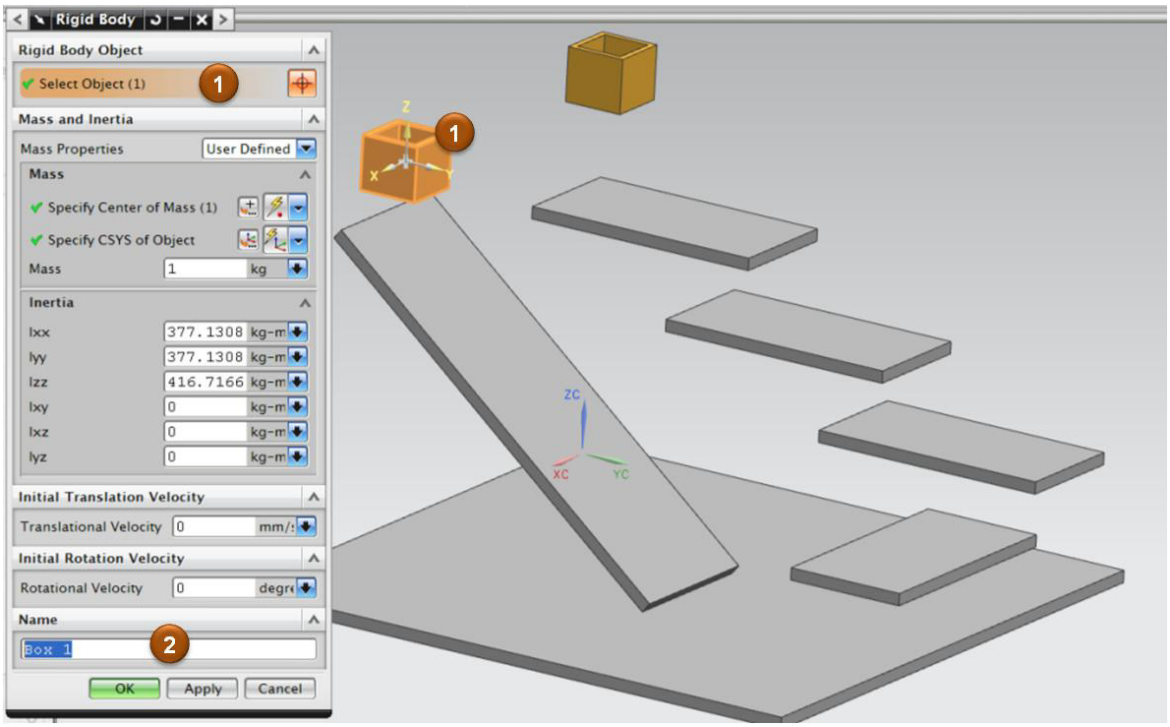


Figure 11: Define Rigid Body

Now when you start the simulation you will see that the Box is falling down. The forces of gravity are applied to it. Go on and make the second box a rigid body, too. Call it "Box 2". You will find these two boxes in the structure of the Physics Navigator.

The next step is to make these Boxes collide with their environment. Therefore define a collision body 

. Select the box (1) and assign a name (2). At this point we will not care about the other parameters. The collision shape should be the default value "box".

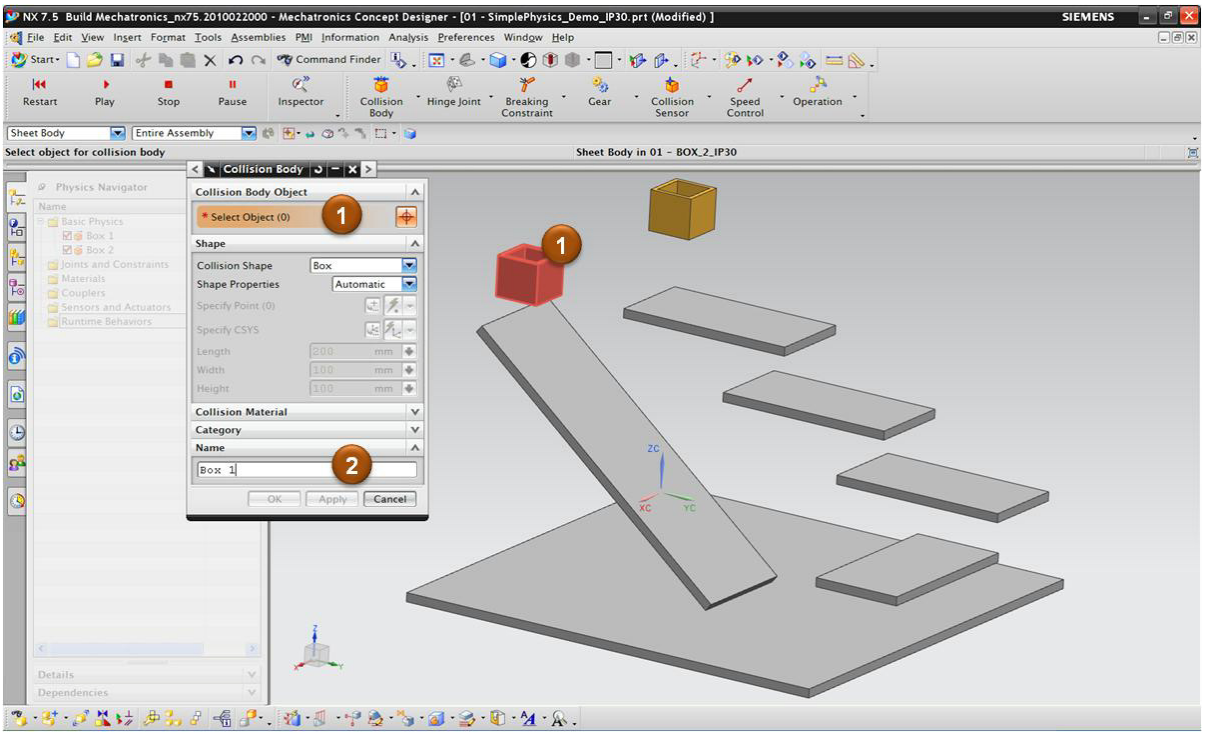


Figure 12: Define Collision Body

Repeat this for every 3D object in the scene and name the conveyers “Conveyer x”. You should get the following list:

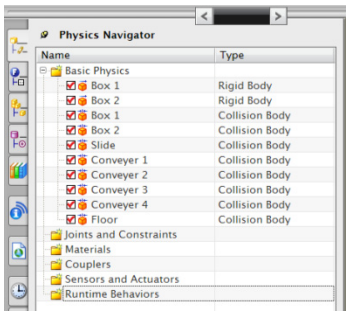


Figure 13: Basic physical elements in the scene

When you start the simulation the boxes are both falling down while they are colliding with the other elements in the scene.

The last step is to define a transport surface 

. Create a transport surface object and select the top face of one of the conveyer elements (1) as its face. Then, select a vector (2) (in this case YC) that is pointing into the direction you want the conveyer move. You can also define the speed of the movement (3).

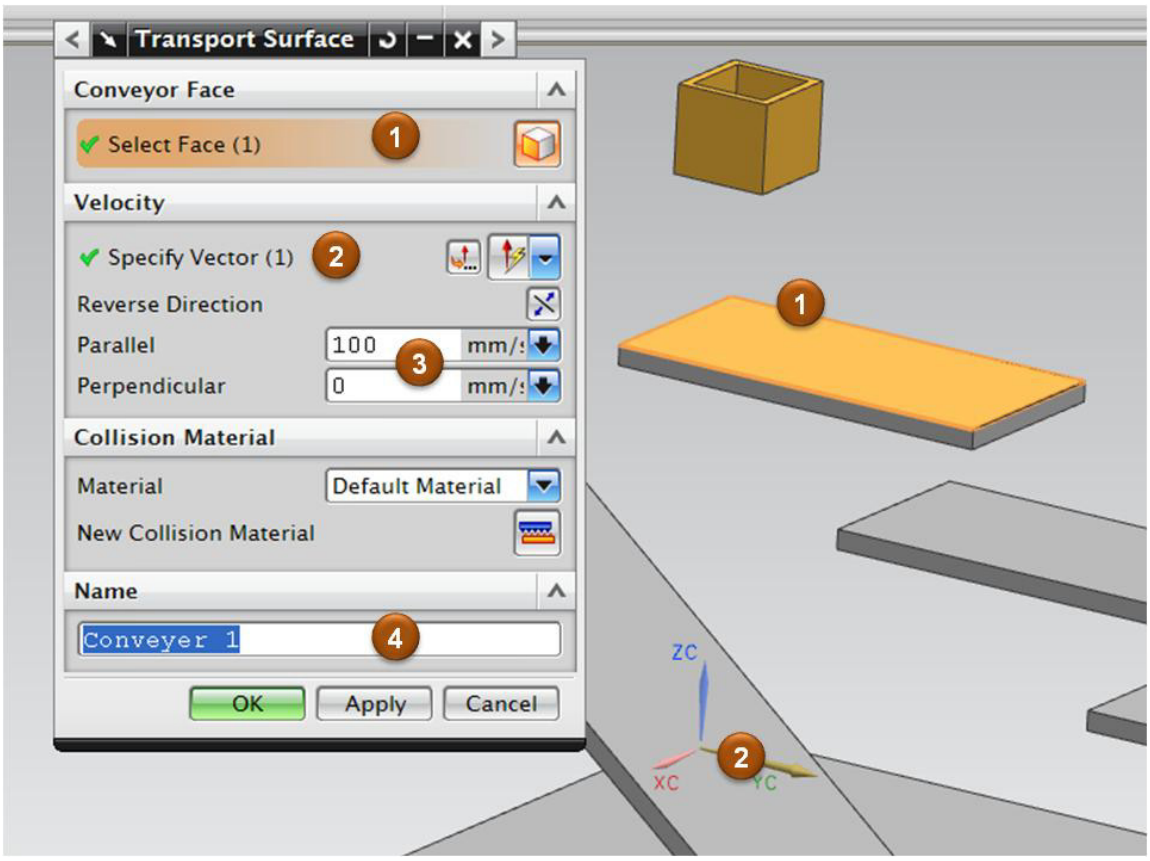


Figure 14: Assign a conveyer surface

This you can repeat for all four conveyers. The simulation will show that the two boxes are falling down or transported to the floor.

While the simulation is running you can use the mouse cursor to apply forces to the objects in the scene. In this case you can manually move the two boxes around and make them fall down from the floor.

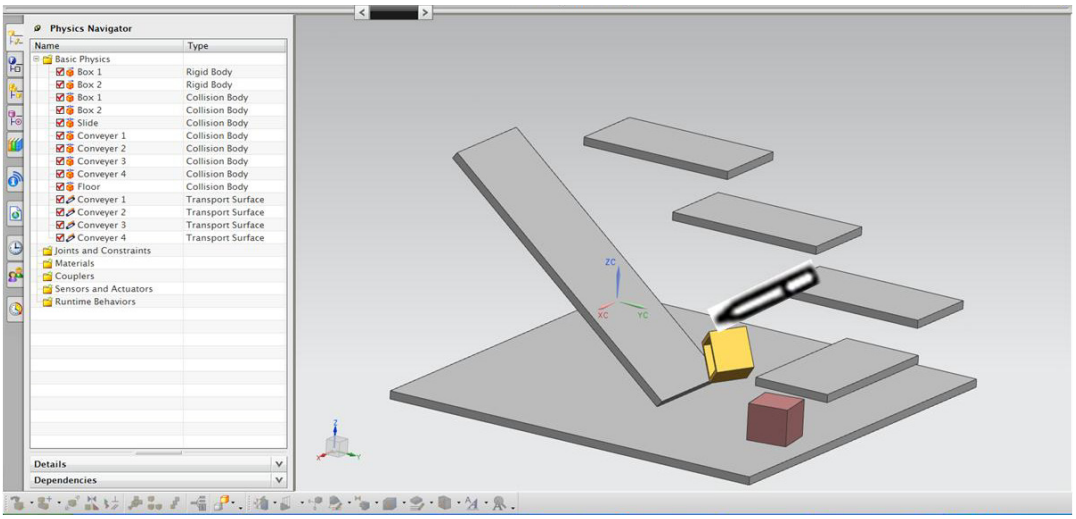


Figure 15: Interact with the running simulation

2.2 Simple robot example

2.2.1 Manage requirements

Enter the virtual PC and start the Rich Client of Teamcenter 8. Under my Teamcenter change to the directory “03 – Simple Robot”. Right click the function tree and send to Structure Manager.

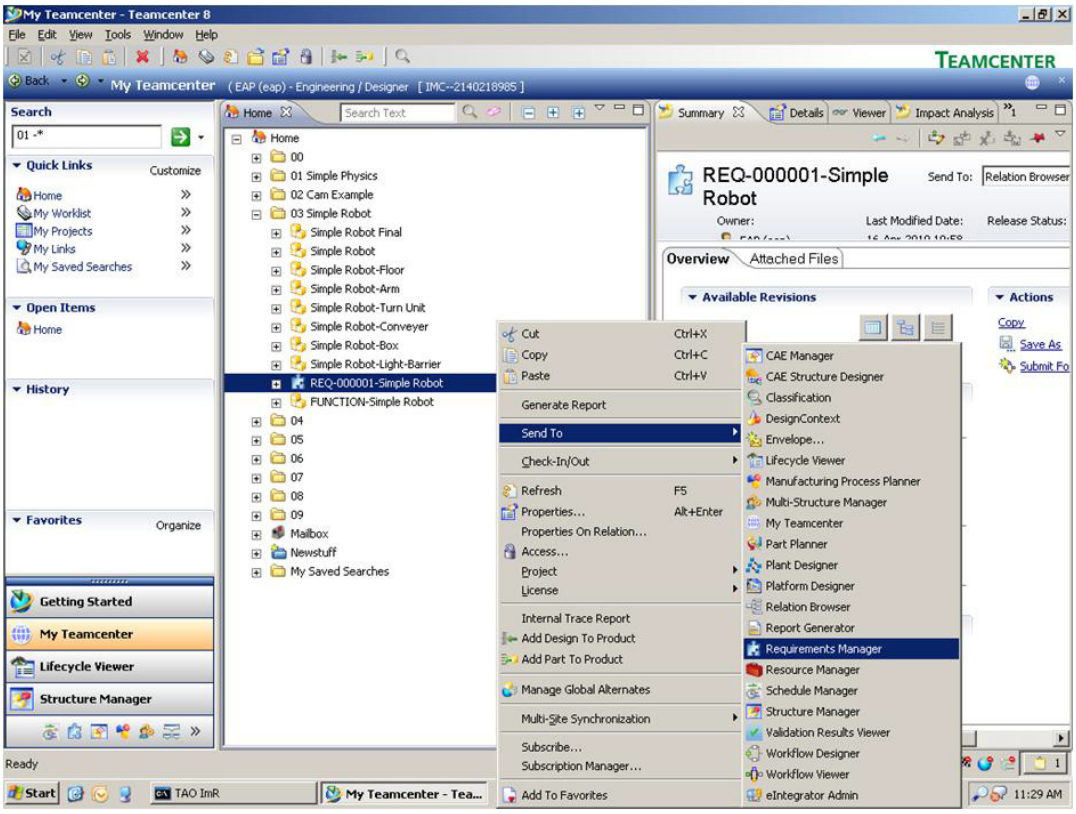


Figure 16: Open Requirements Tree in the Requirements Manager

Add a new requirement : *“The speed of the conveyer belt should be 0.5 m/sec.”*

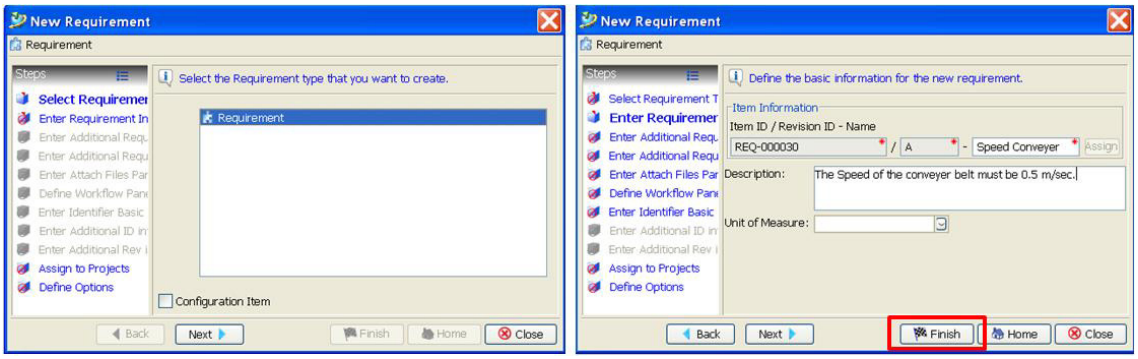


Figure 17: Add a requirement

If can add multiple requirements in a hierarchical structure. For this example it is enough just to work with one example requirement.

2.2.2 Functional decomposition

Enter the virtual PC and start the Rich Client of Teamcenter 8. Under my Teamcenter change to the directory “03 – Simple Robot”. Right click the function tree and send to Requirements Manager.

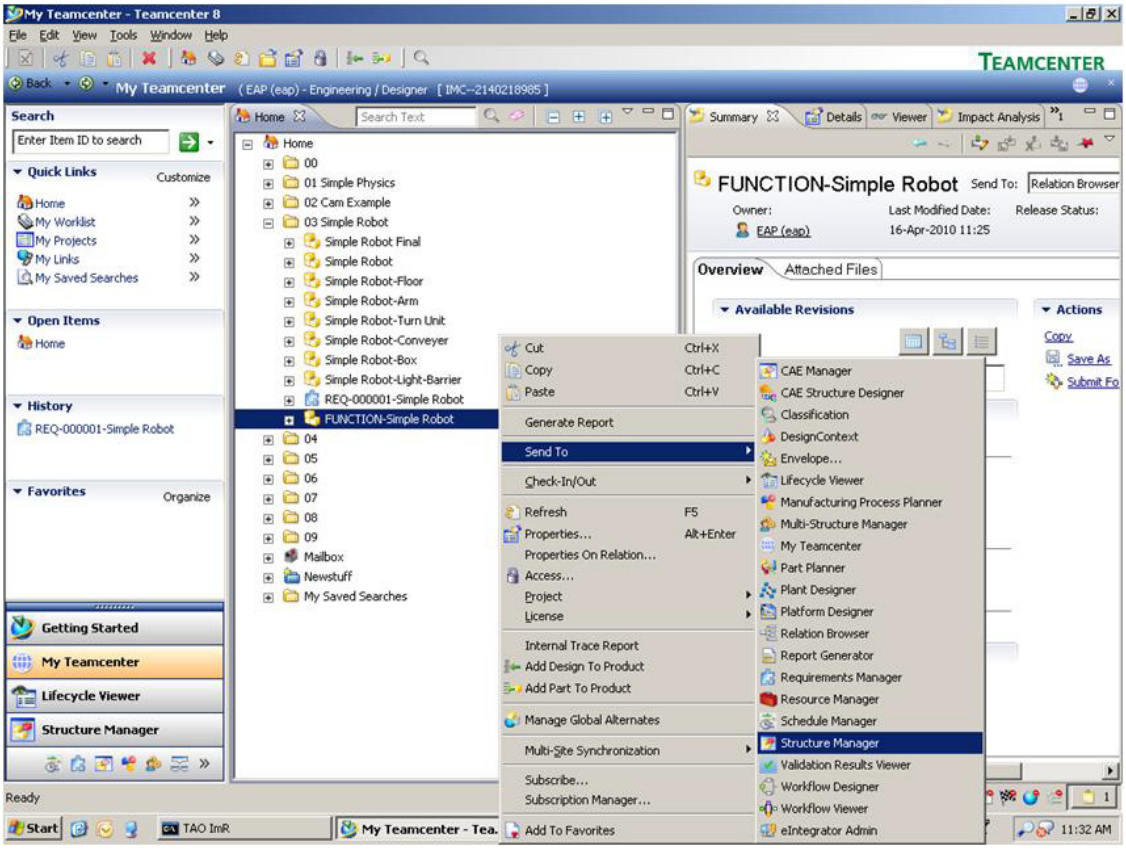


Figure 18: Open Function Tree in the Structure Manager

In the Structure Manager perform the functional decomposition by adding new items “Functionality”.

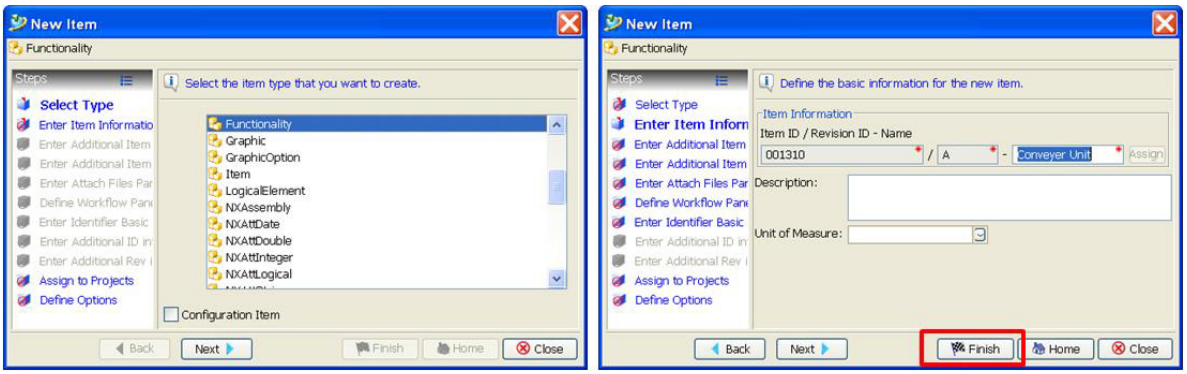


Figure 19: Add Item for the Functional Decomposition

Assign the Item ID and the Revision ID automatically. Create a functional decomposition like this:

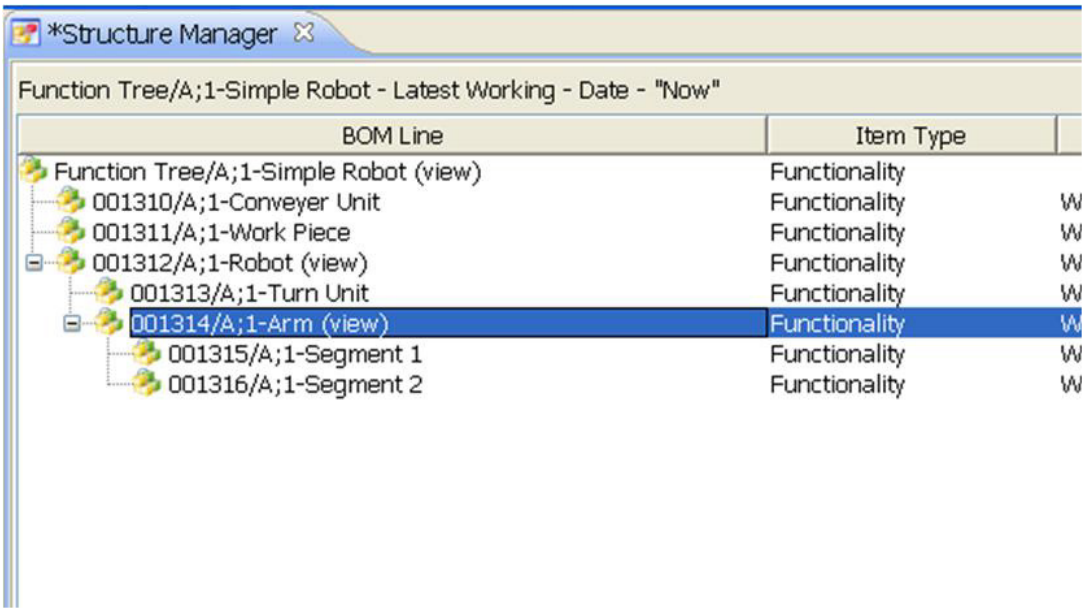


Figure 20: Simple Functional Decomposition

2.2.3 Link requirements and functions

Now open both the Function Tree and the Requirements Tree in the Requirements Manager and select "Split panel".

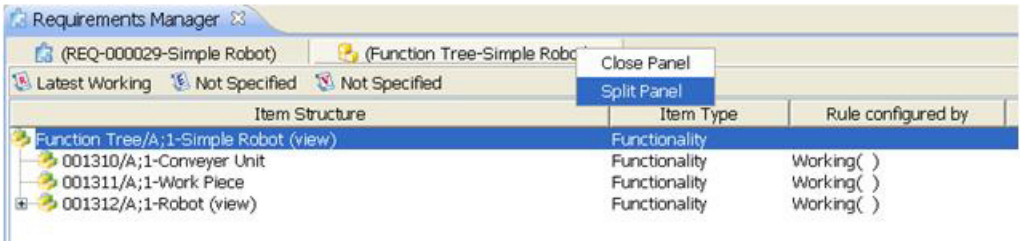


Figure 21: Split Panel

Now create a trace link. First select the requirement and start a trace link (). Then select a function and end a trace link ().

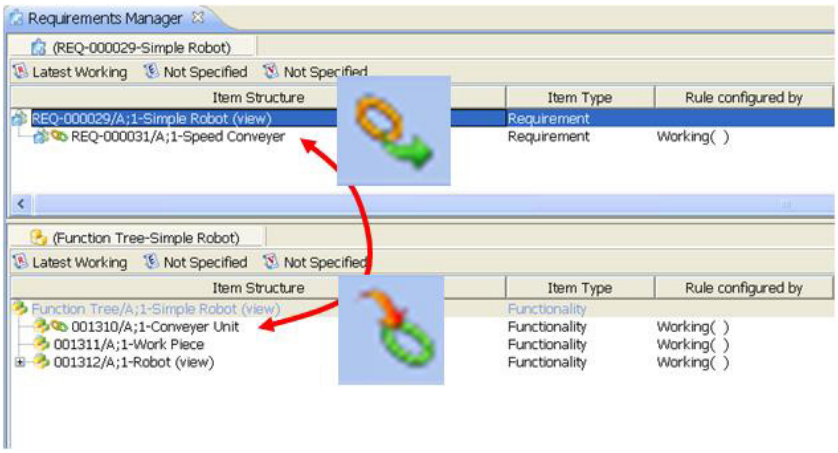


Figure 22: Create a trace link

2.2.4 Implement functions

Please open the part file: “Simple Robot.prt”. You will see a conveyer belt and a simple robot with two arm segments and a turn unit.

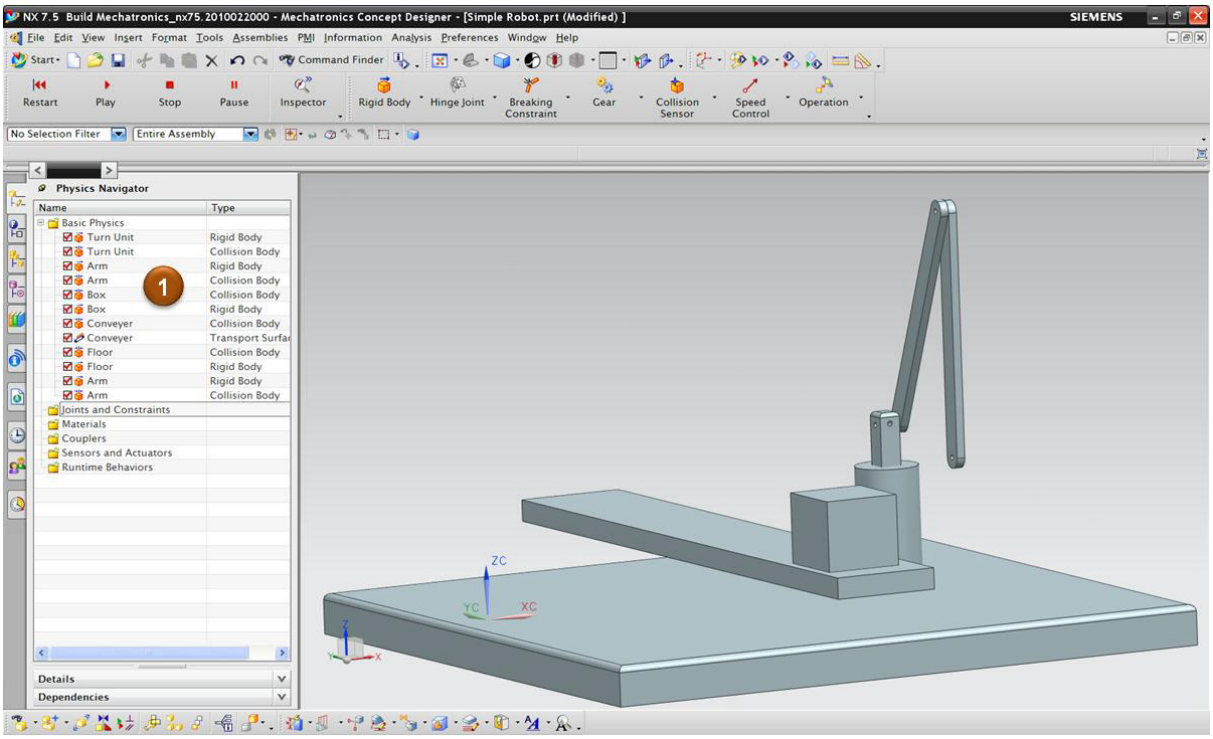


Figure 23: Example with Simple Robot and Conveyer

Load a functional model from Teamcenter with a right-mouse click into the Function Navigator.

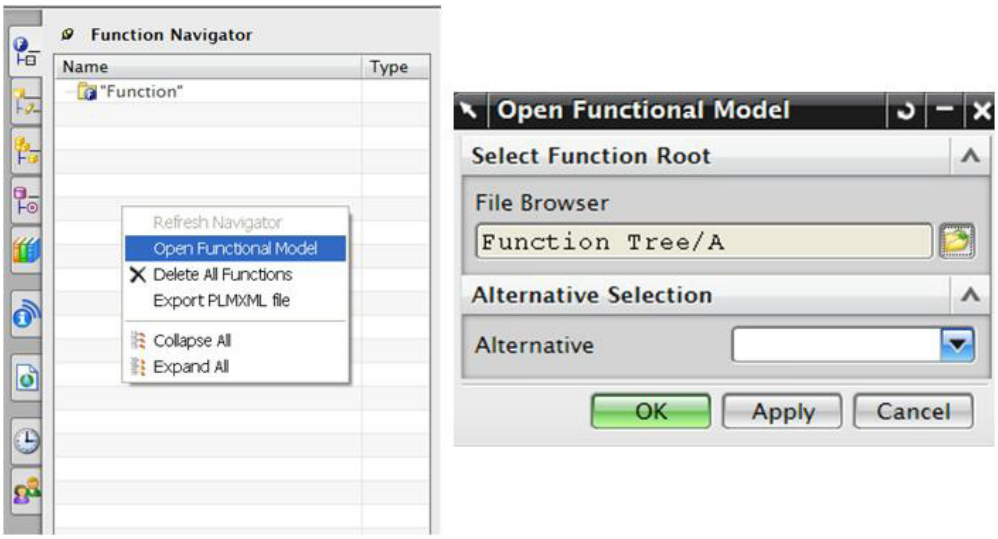


Figure 24: Open a functional model

The Function Navigator shows the functional decomposition. Mechatronics Concept Designer has automatically added two folders:

- The **Components** folder links elements from the 3D-design.
- The **Operations** folder links operations from the Sequence Editor.

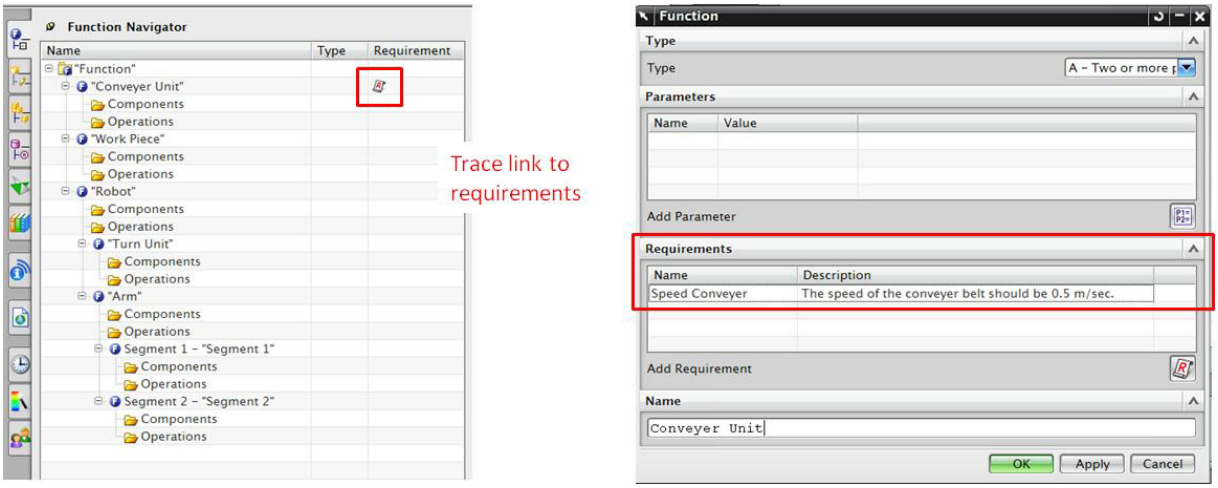


Figure 25: Function Navigator show imported function tree from Teamcenter

Double click the little icon in the requirements column to view the linked requirements.

To link implementations to the corresponding “Components” folder of a function right-click on the folder and select “Add existing component”. Now pick a component in the 3D design. Same procedure to link Operations to the Operations folder.

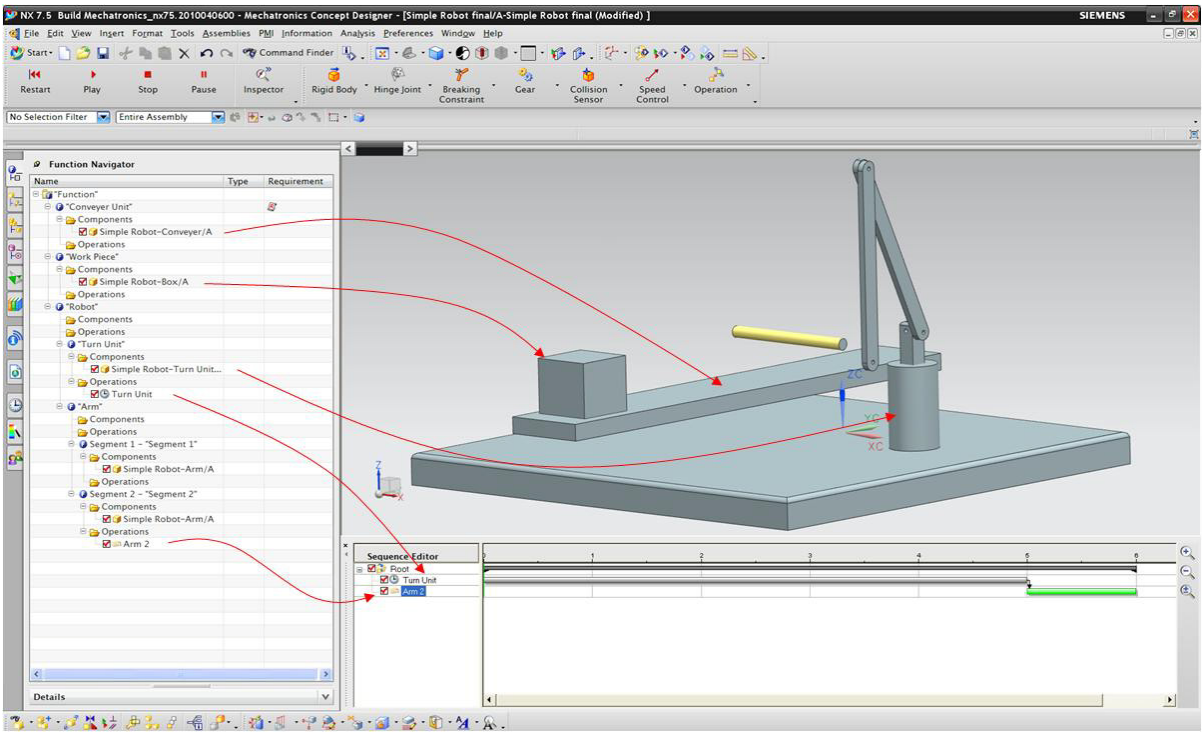


Figure 26: Function Navigator show imported function tree from Teamcenter

2.2.5 Simple Kinematics

The Physics Navigator shows that basic physics elements like rigid bodies, collision shapes and transport surfaces have already been assigned. Now start the simulation. The floor and the robot are falling down due the forces of gravity. Apply the fixed joint 

.

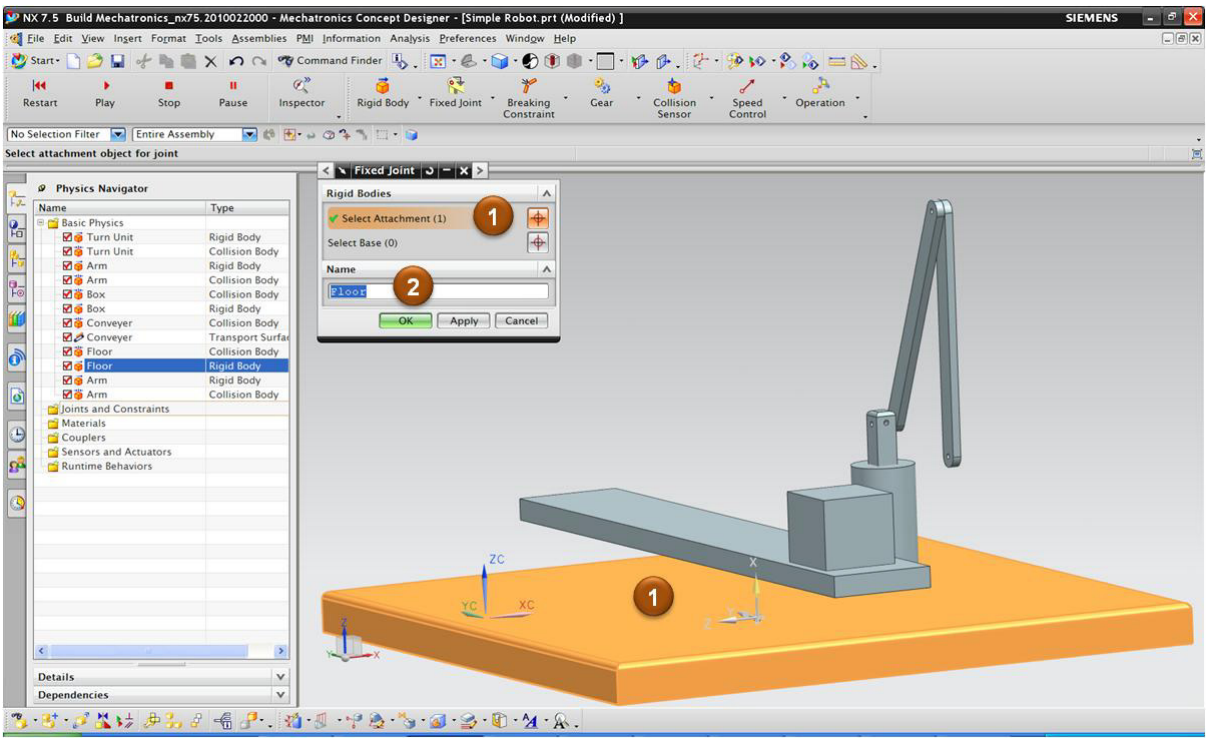


Figure 27: Apply fixed joint to the floor

The “fixed joint” creates a rigid link between two bodies. Select the floor 1 as “Attachment” and leave the “Base” empty. In this case the floor will be hooked up with the background. Assign the name “Floor” 2.

When you now start the simulation you will find out that the arms on the robot are falling down. They have to be linked by a hinge joint 

.

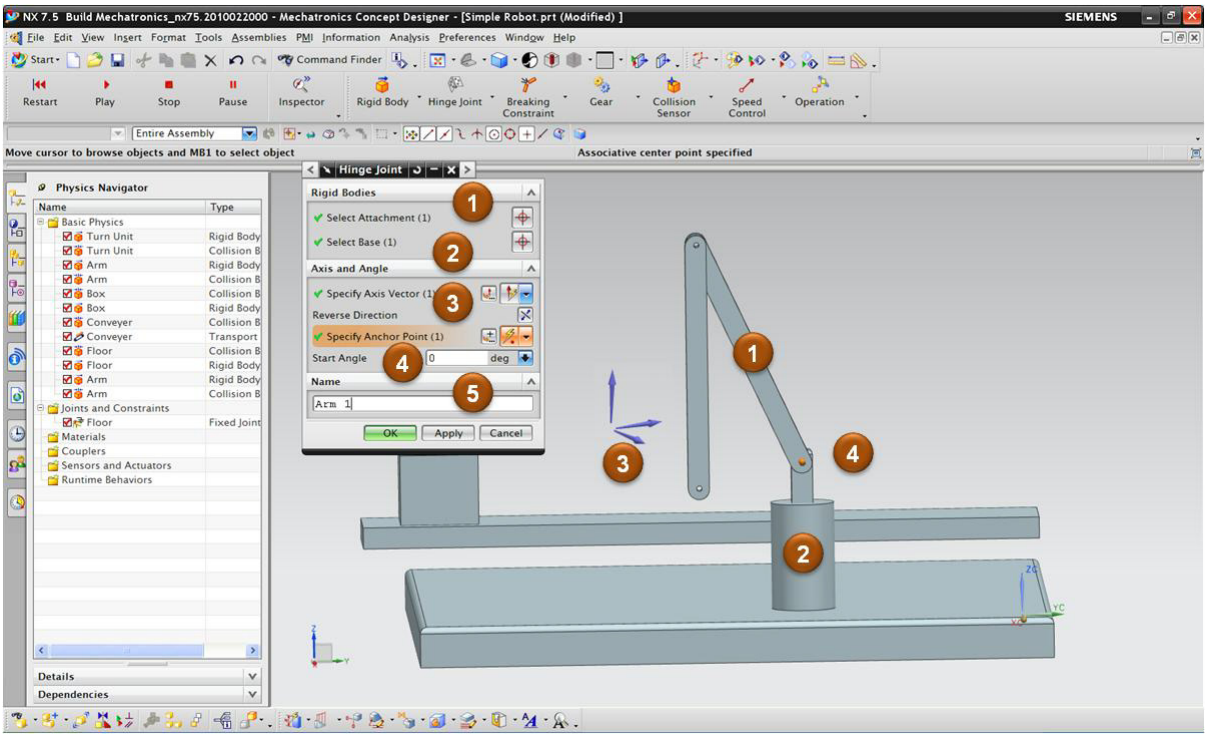


Figure 28: Define "Hinge Joint" to connect the arm to the turn unit.

Select the arm as the "Attachment" (1). Select the turn unit as "Base" (2) and define the vector of the turn axis (3). Select the center of the hole in the arm as "Anchor Point" (4) and assign the name "Arm 1" (5).

Then perform the same exercise to connect arm 2 with arm 1. Starting the simulation now will cause the robot to fall over. Therefore define a "Hinge joint" and link the turn unit with the floor.

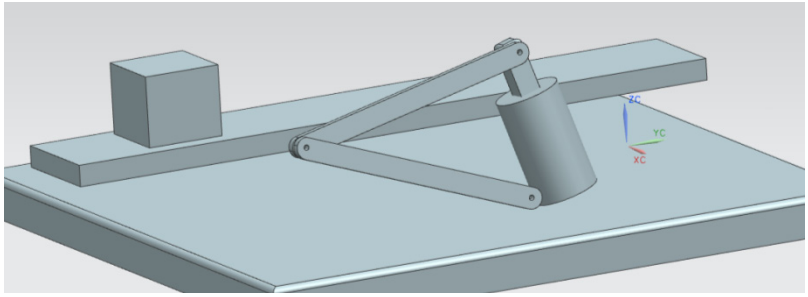


Figure 29: Robot is falling over

When you now start the simulation you can drag the arms of the robot with your mouse pointer.

2.2.6 Actuators

The kinematics is now defined and we can assign the actuator to drive the axes of the robot 

.

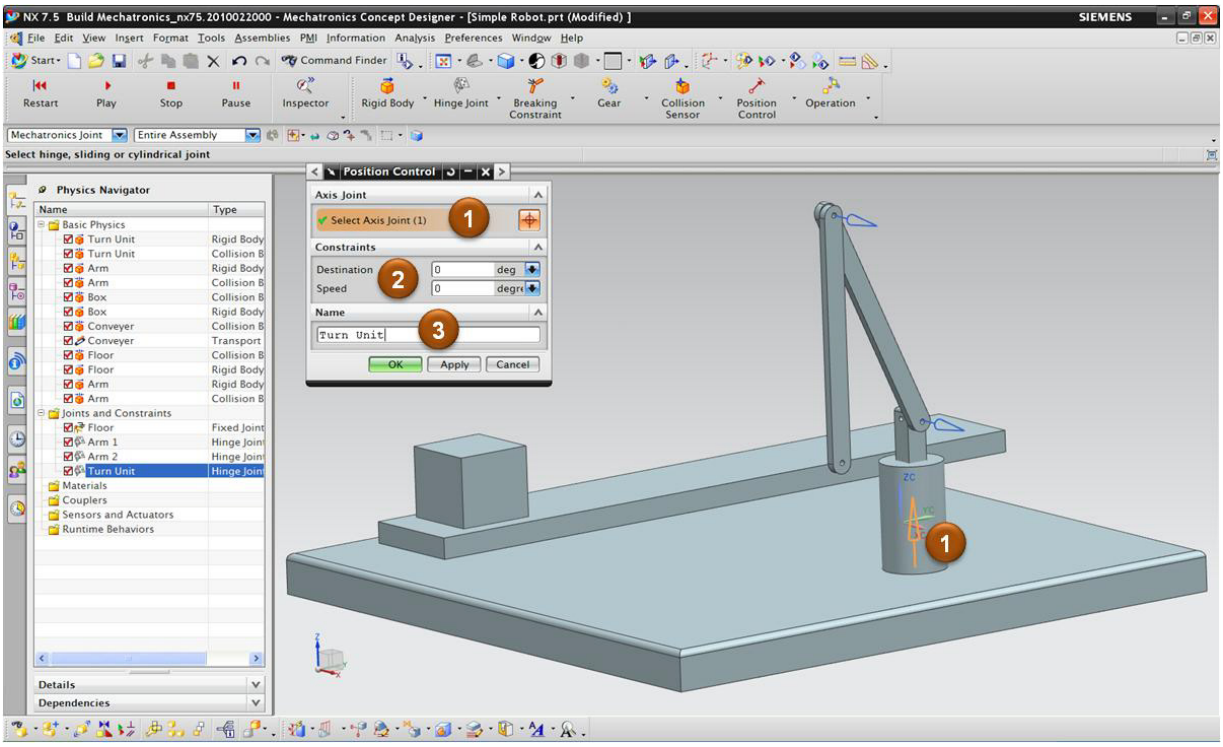


Figure 30: Assign a "Position Control"

Select the axis of the turn unit (1). Set Destination and Speed to zero (2) and assign the name "Turn Unit" (3). Perform the same exercise for the two axes of the robot arms. Starting the simulation, the three actuators will keep the arms and the turn unit in the position zero. It is not possible to move them with the mouse pointer. Now the "Physics Navigator" should look like this.

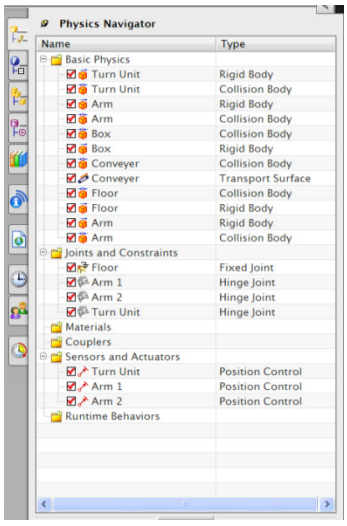


Figure 31: Physics Navigator

You can find that the Sensor and Actuator list now contains three actuators.

2.2.7 Time based Operations

To make the robot move in a time based manner we have to define operations 

.

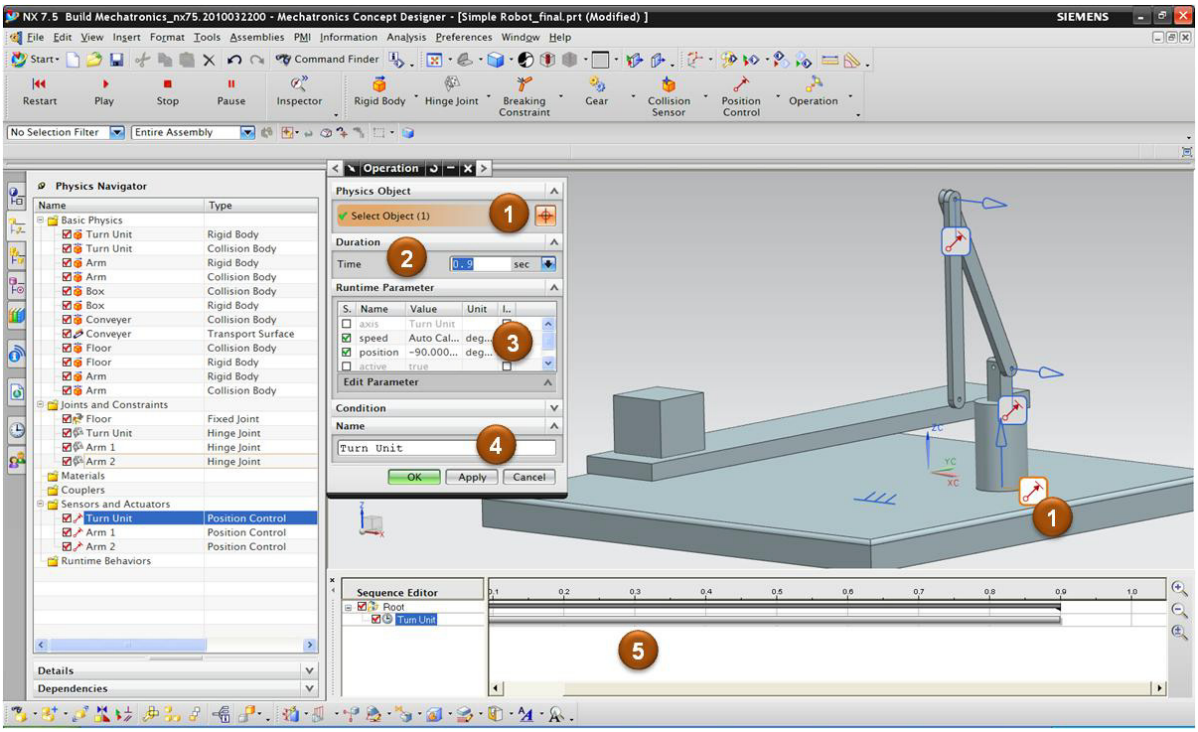


Figure 32: Define an operation

To get a pretty good overview of the machine the actuators are highlighted with a symbol  . Select the actuator “Turn Unit” (1) and set the time to “0.9 sec” (2). In this section (3) you can access the interface of the actuator and define the position “-90°”. Check the speed and let it be auto-calculated based on time and position. The name should be ”Turn Unit” (4). All the operations will appear in the Sequence Editor (5).

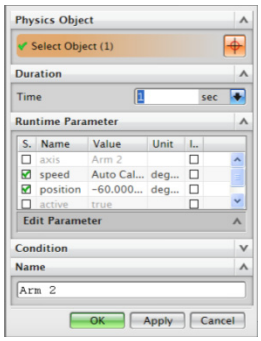


Figure 33: Define the second operation

Repeat this exercise for the actuator “Arm 2” and set the time to “1 sec” and the position to “-60°”.

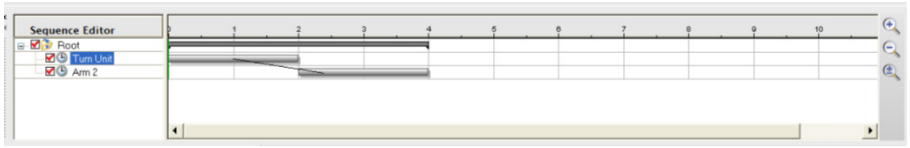


Figure 34: Define the sequence of the operations

To define that the operation “Arm 2” happens after the operation “Turn Unit” drag a line out of the bar of “Turn Unit” and connect it to the bar of “Arm 2”.

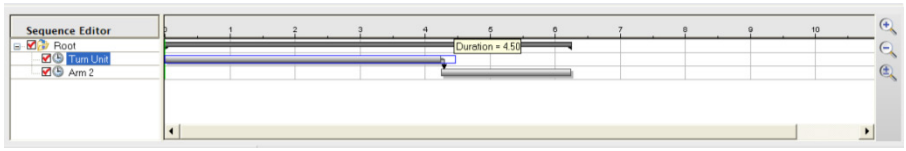


Figure 35: Define the sequence of the operations

To make the robot toss the box from the belt pick the end of the first bar and drag it to 5 secs.

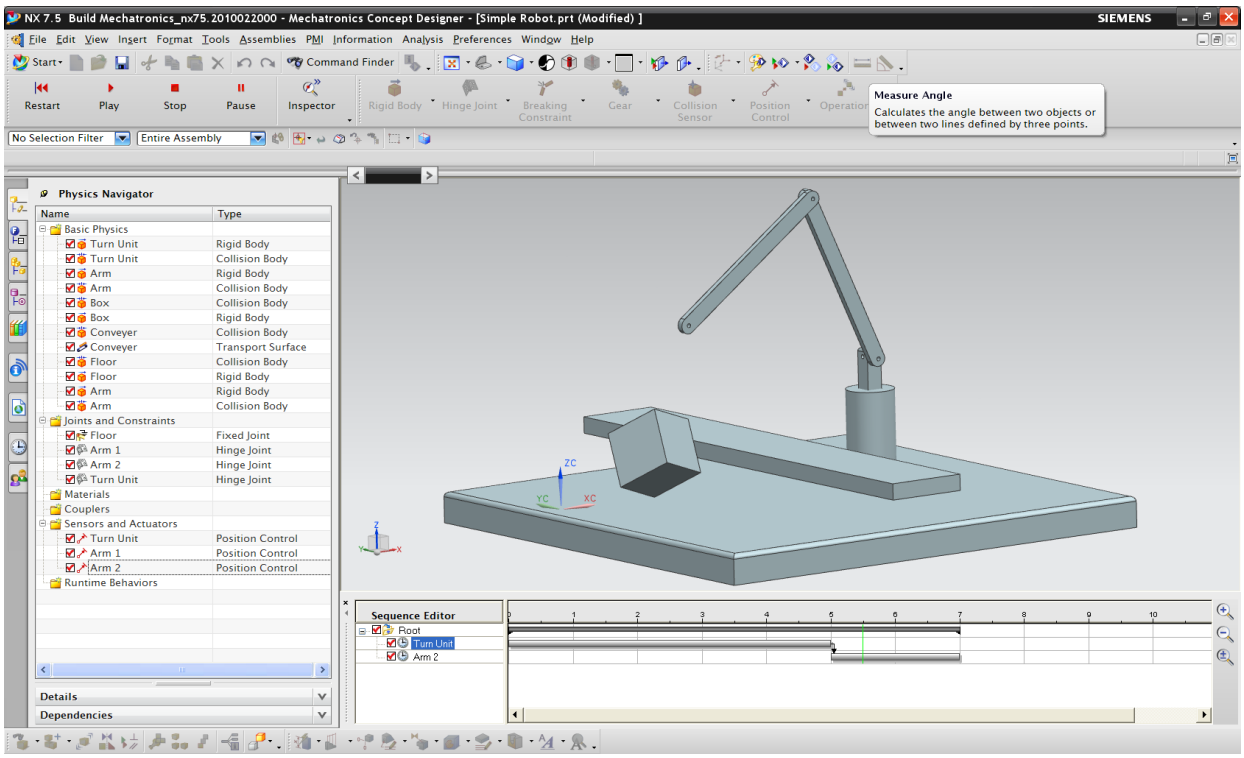


Figure 36: The box is tossed from the conveyer

2.2.8 Sensors

To add a sensor to the scene we first have to define a shape that could act as a light barrier. Therefore switch to the “Assembly Navigator” and check the part “Simple Robot-Light-Barrier”.

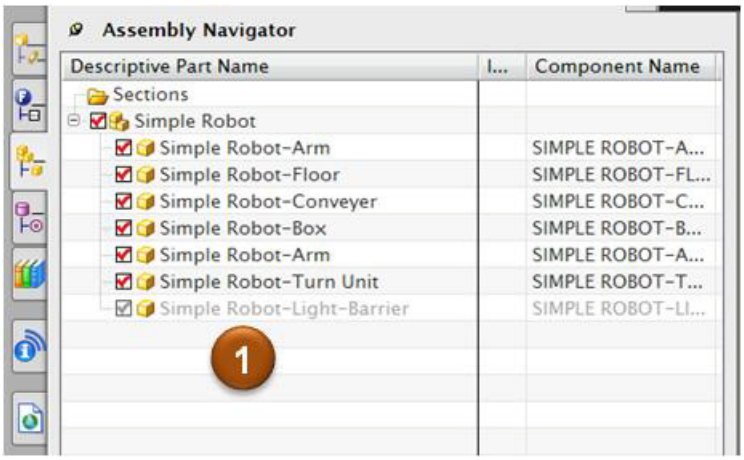


Figure 37: Add the shape for a light barrier

Now define this shape to be the light barrier 

.

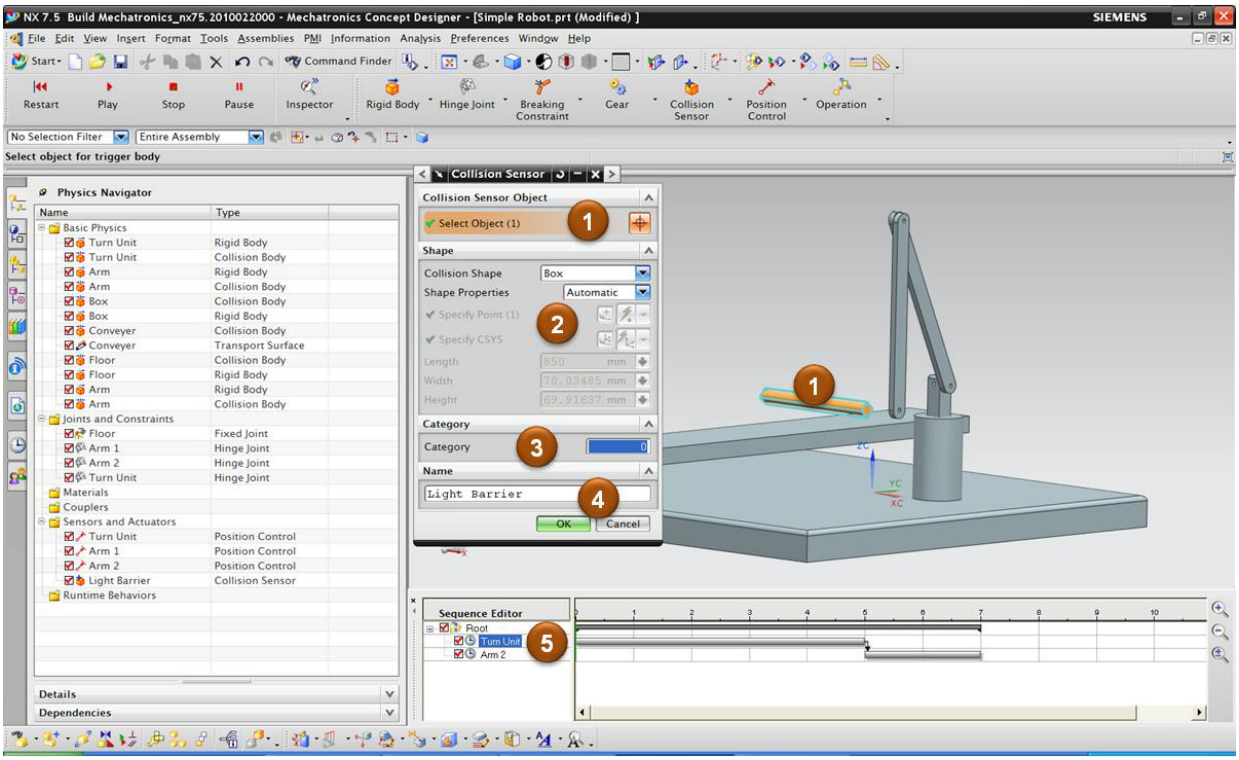


Figure 38: Add the shape for a light barrier

Select the 3D object 1. Don't make changes to the parameters 2 and 3. Assign the name "Light Barrier" 4.

2.2.9 Event based operations

Now double click the operation “Arm 2” (5) in Figure 38.

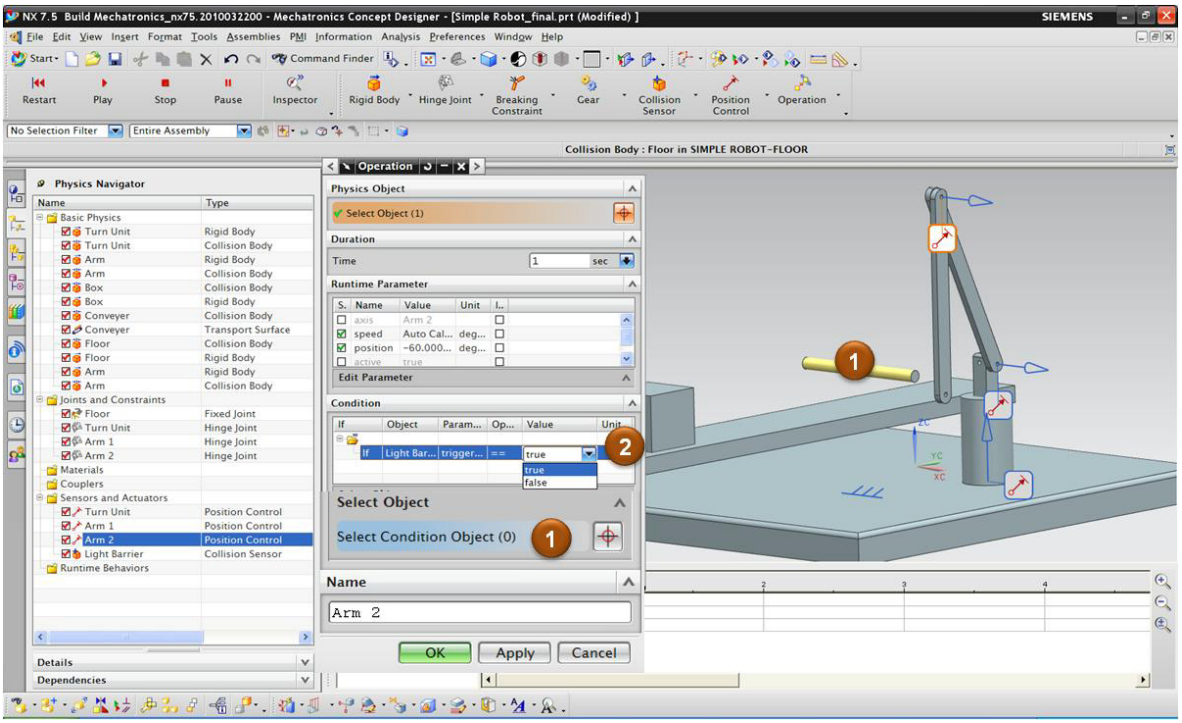
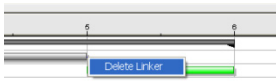


Figure 39: Change the operation to wait for the event of the “Collision Sensor”

Select the sensor shape as a “Condition Object” (1) and define the condition

If | Light Barrier | trigger | == | true” (2), the bars denote the separation between table elements.

Please also delete the link between the two operations with a right mouse click.



The operation “Arm 2” is now dynamically invoked when the box is colliding with the light barrier. The green color is indication

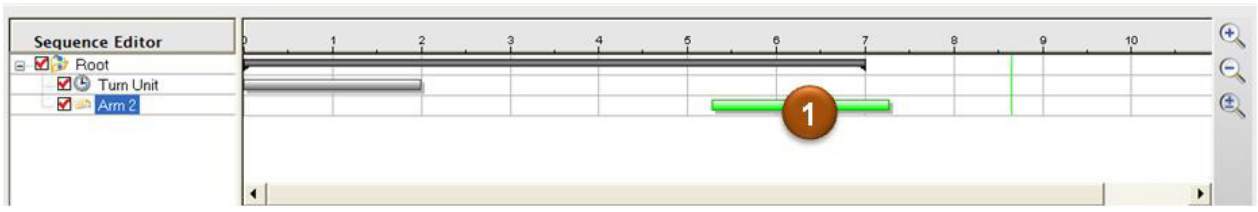


Figure 40: Event based operation “Arm 2”

2.3 Cam example

2.3.1 Define a cam

Please open the file “02 - Cam_Example_IP30.prt”. An assembly with multiple axes shows up.

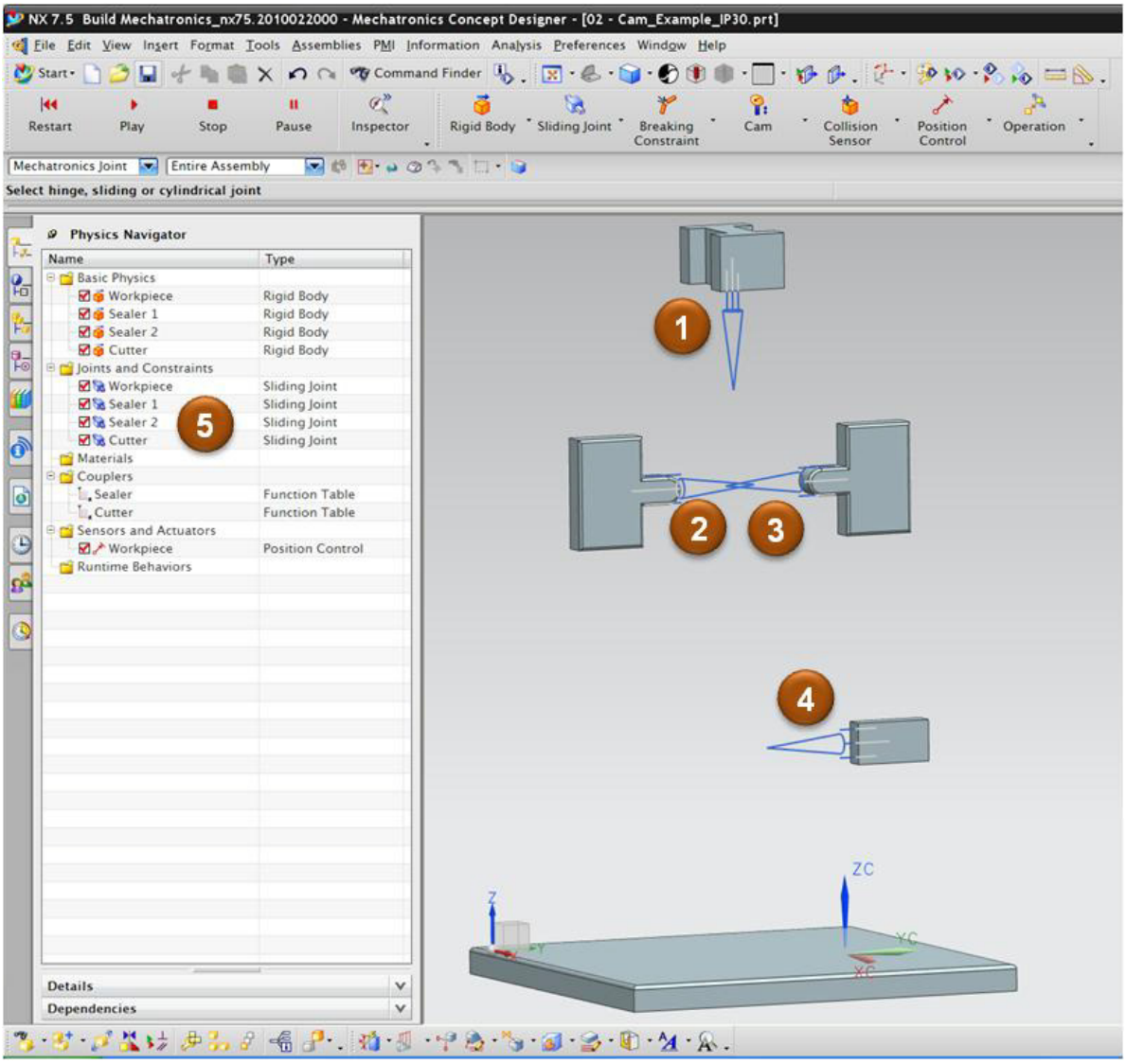


Figure 41: Event based operation “Arm 2”

The master axis (1) is driven by an actuator that moves the work piece with constant velocity down to the floor. The other axes are “slave axes” ((2), (3), (4)) and will be synchronized to the motion of the master axis. The Physics Navigator shows the corresponding “Sliding Joints” and two items with the type “Function Table” (5).

The function table defines how the slave axis is synchronized with the master axis. Double-click on the function table Sealer. This function table will be used to synchronize the motion of the two axis 2 and 3.

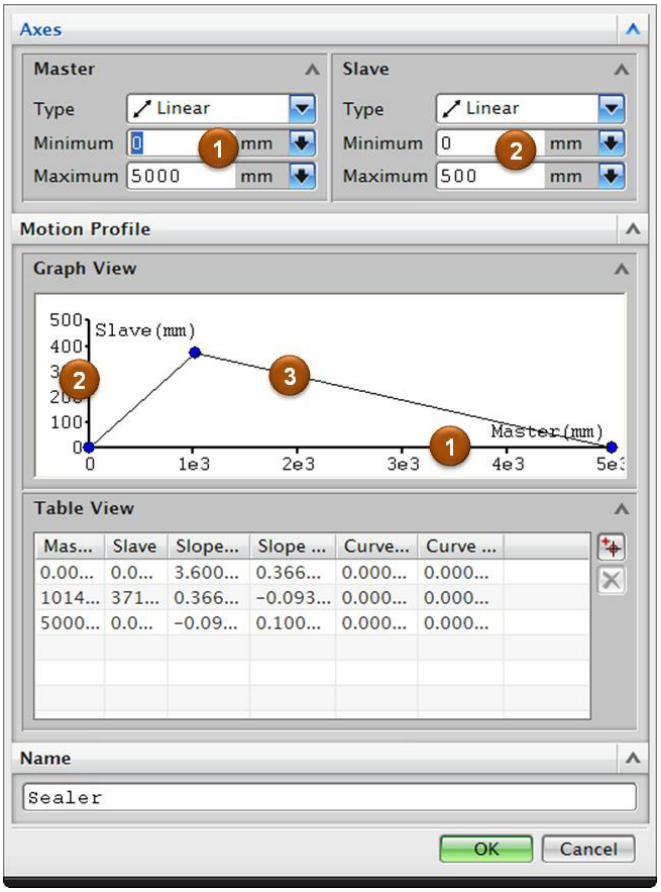


Figure 42: Cam Editor

The cam editor shows the master axis 1 with a length of 5000mm which you can find in the function chart on the X-axis of the coordinate system. The slave axis 2 has a length of 500mm and is represented by the Y-axis in the function table. The curve 3 defines how the slave axis is following the master axis.

Now close this dialog and create the corresponding coupler between the axes 

.

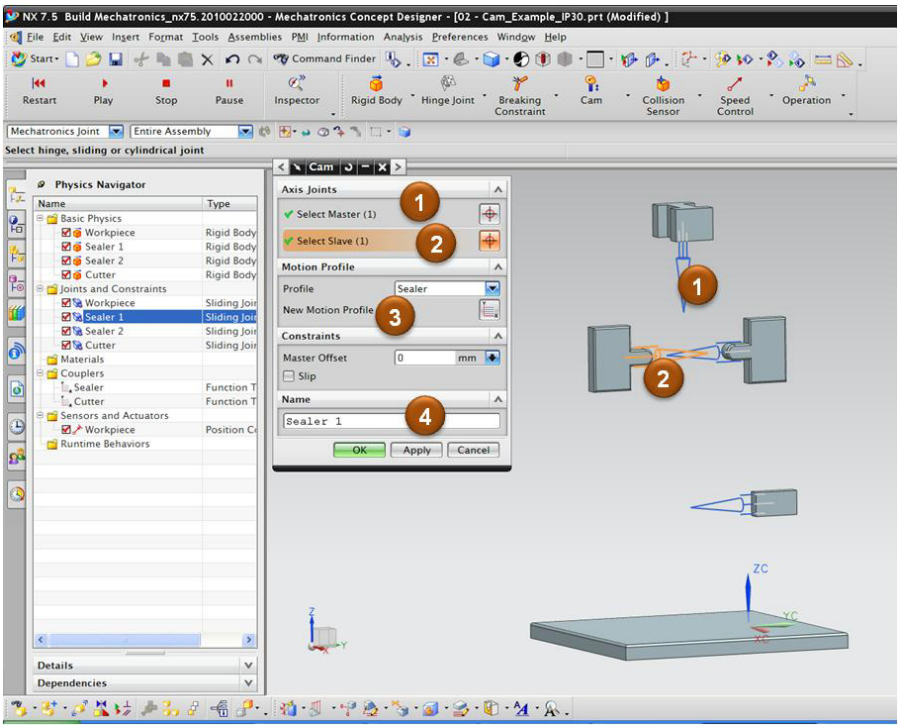


Figure 43: Define the coupler for the cam

Select the master axis 1 and the slave axis 2. Assign the function table 3 and the name 4. Please repeat this for the remaining slave axis and link them all to the same master axis.

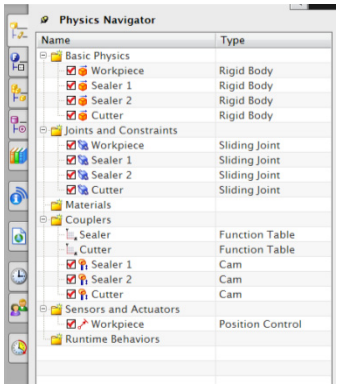


Figure 44: Physics Navigator with all cams

Start the simulation and you will see how the motion of all axes is coordinated by the cams.

You can now play around with the functions and adapt the behavior of the slave axes. Double-click the “Cutter” function table.

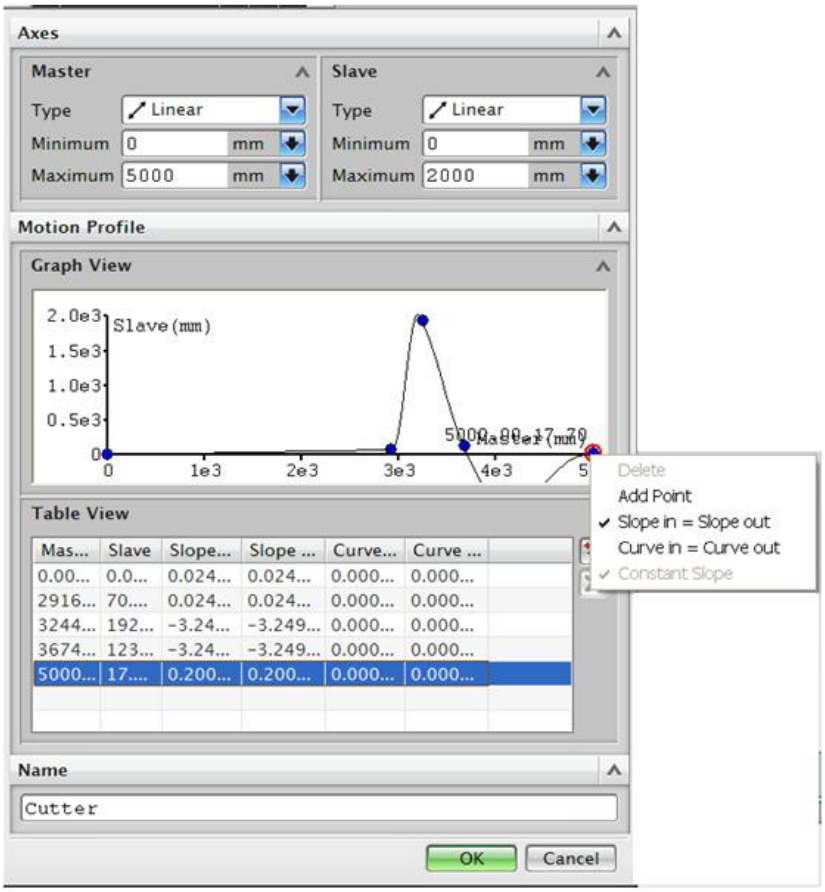


Figure 45: Edit the curve of the cam

You can drag the points, add new one with the “Right-Mouse-Click” and edit parameters of the points. If you set “slope in = slope out” the cam editor will automatically generate a smooth curve that can be driven by a real actuator. Just start the simulation to immediately see the effect of your changes.

2.3.2 Inspector

The inspector monitors runtime parameters of objects in the scene. Open the inspector window 

and start the simulation. Now you can point to objects in the scene (1) and will immediately see basic runtime parameters of this object (2).

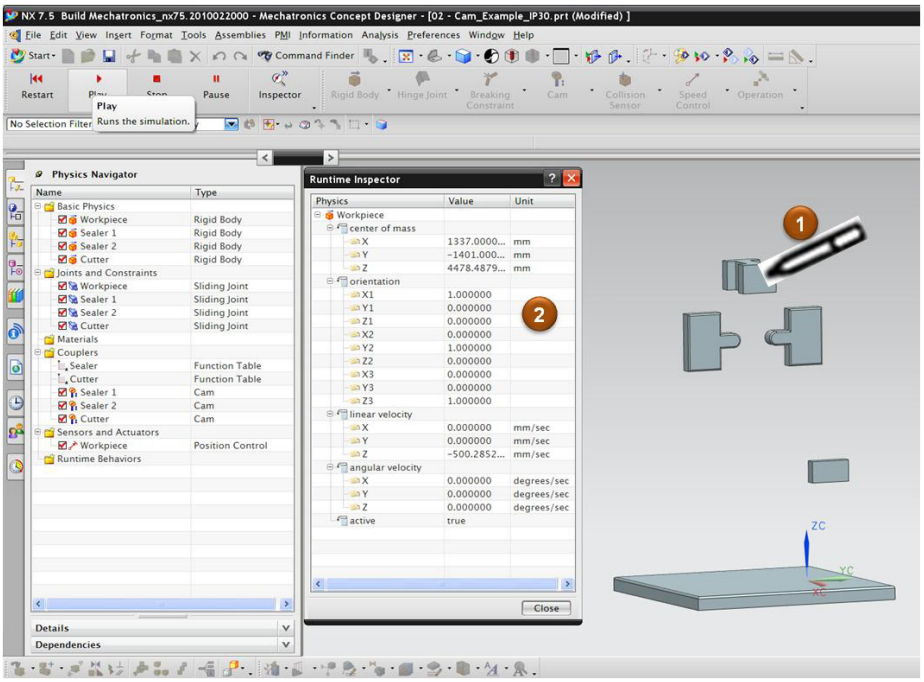


Figure 46: Edit the curve of the cam

3 The navigators

Mechatronics Concept Designer introduced three new navigators. The Function Navigator, the Physics Navigator and the Sequence Editor.

3.1 The Function Navigator

The Function Navigator provides the link from Mechatronics Concept Designer to the functional decomposition and the requirements stored in Teamcenter. You can create and maintain both structures in Teamcenter. Teamcenter provides a feature to create a trace link (1) to match requirements and functions. In addition Teamcenter maintains variants of the functional decomposition. Once you load the functional decomposition into the Function Navigator of Mechatronics Concept Designer (2) you have to decide which alternative you want to access. Via the trace links the Function Navigator provides access to the corresponding requirements (3).

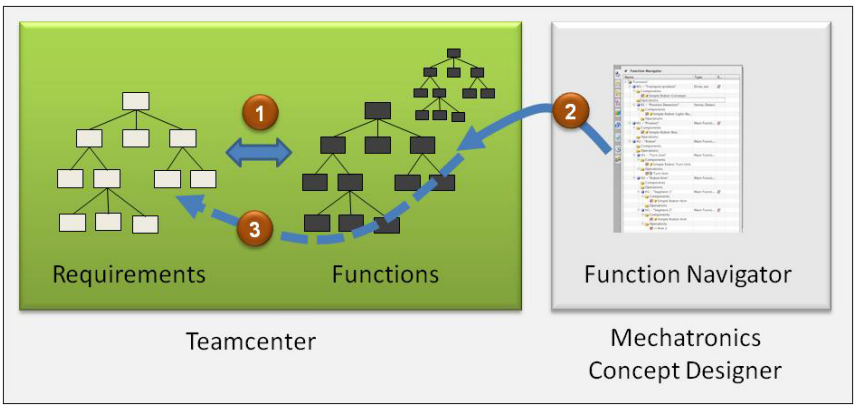


Figure 47: Access Functions and Requirements in Teamcenter

The Function Navigator shows the following information about a function in the columns:

- Name of the function: this can be assigned by the user.
- Type of the function: this is a function type according to a specific classification. Mechatronics Concept Designer is currently supporting the European Standard “EN 61346-2” (see chapter 6.1 on page 97).
- Requirements: a symbol in this column indicates that requirements for this function are accessible.

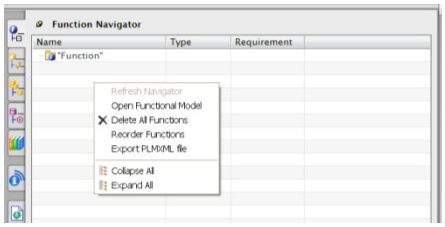


Figure 48: Edit the curve of the cam

A right mouse click in the Function Navigator offers the following items:

Feature	Description
Open Functional Model	This will load a new Functional Model • In Teamcenter Mode load a functional structure from the database (item type: function). • In Native Mode load a PLMXML file from the file system.
Delete All Functions	This will delete all functions in the Function Navigator.
Export PLMXML file	Save the current functional structure in the Function Navigator in a PLMXML file.
Collapse All	Collapse all items of the structure displayed in the Function Navigator.
Expand All	Expand all items of the structure displayed in the Function Navigator.

There are additional functions when you right-click on items in the Function Navigator.

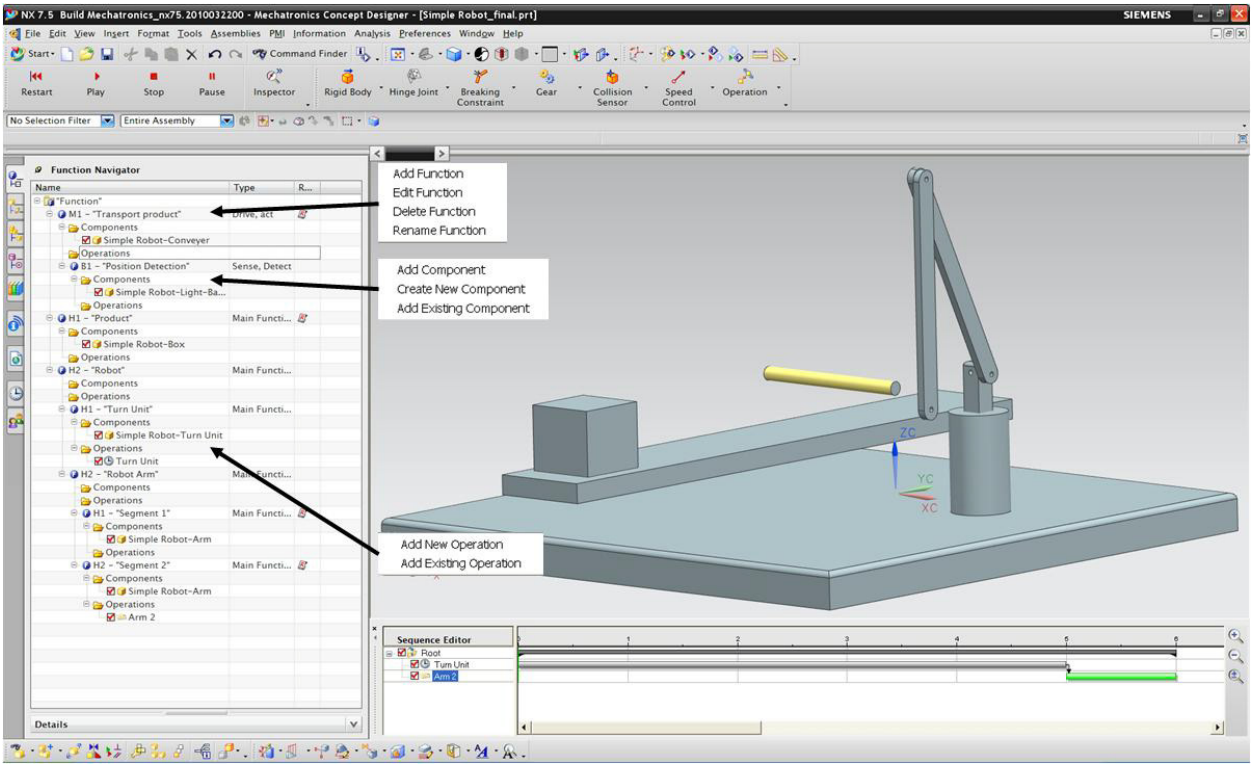


Figure 49: The Function Navigator

Item in the Function Navigator	Item in the pull down menu	Description
Function	Add Function	Adds a new function on hierarchy level below the selected function.
	Edit Function	Edit the selected function. Also possible with double-click.
	Delete Function	Deletes the selected function.
	Rename Function	Change the name of the selected function.
Components	Add Component	Adds a component to the function that is not yet part of the current assembly. This component (part file) can be selected in Teamcenter or in the file system and will be added to the assembly and to selected component folder.
	Create New Component	Creates a new empty part file that is added to the current assembly and to the selected component folder.
	Add Existing Component	Pick a component from the current available assembly. Just pick it in the 3D representation.
Operations	Add New Operation	Create a new operation and add to the selected “Operations” folder.
	Add Existing Operation	Select an operation from the Sequence Editor and add to the selected “Operations” Folder.

3.2 The Physics Navigator

The Physics Navigator gives an overview of the physical and logical behavior that has been assigned to the design.

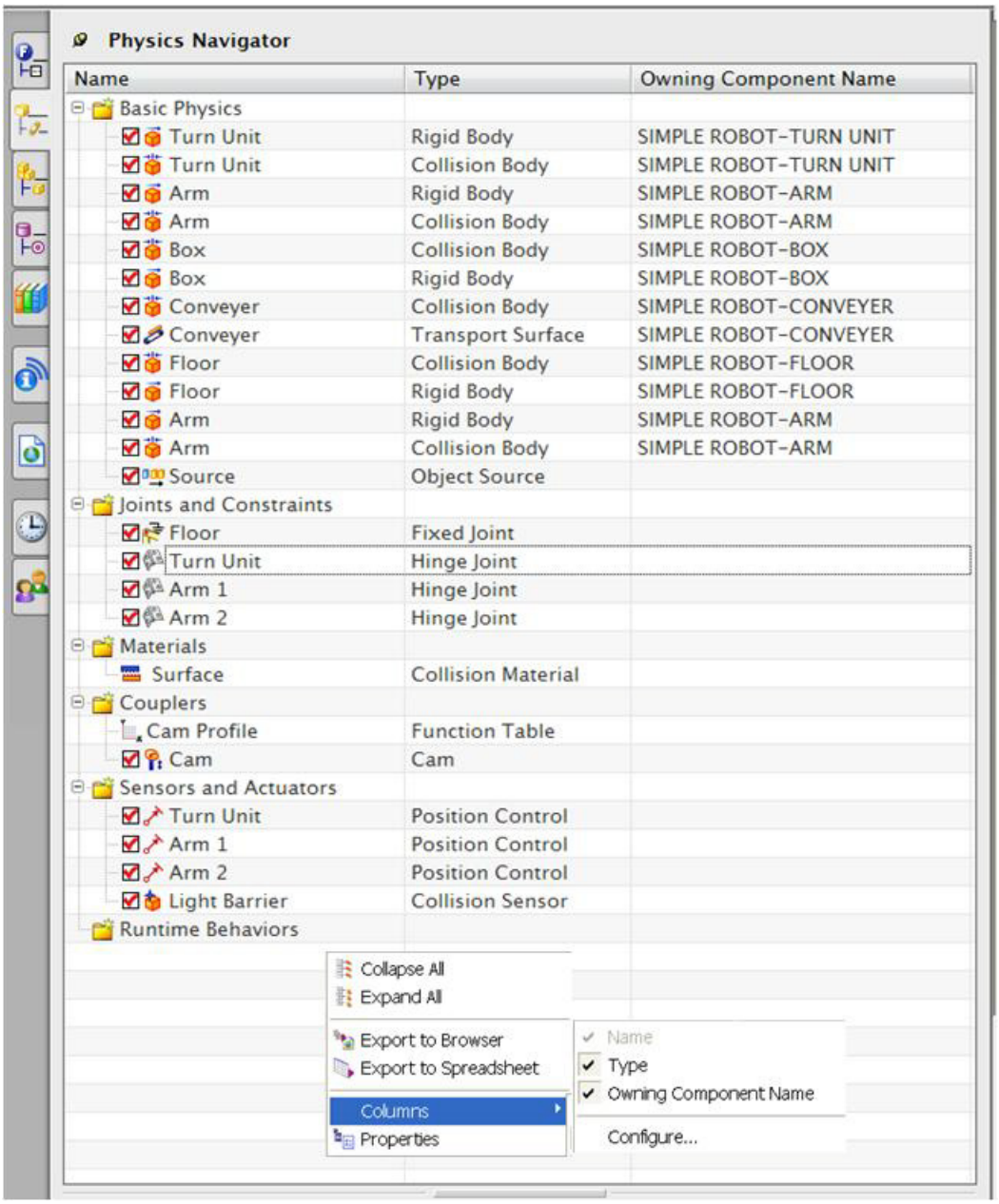


Figure 50: The Physics Navigator

Type of Structure	Contains the following item types
Basic Physics	Items that define the basic physical behavior of the mechanical system: • Rigid Body • Collision Body • Object Source • Object Sink • Transport Surface
Joints and Constraints	Joints that build kinematic chains and constraints to restrict the motion of bodies: • Hinge Joint • Sliding Joint • Ball Joint • Fixed Joint • Angular Spring Joint • Linear Spring Joint • Angular Limit Joint • Linear Limit Joint • Breaking Joint • Prevent Collision
Materials	Items that define collision properties of objects: • Collision Material
Couplers	Definition how two axis are synchronized: • Motion Profile • Gear • Cam
Sensors and Actuators	Current list of Sensors and Actuator that are built into the mechatronics system: • Speed Control • Position Control • Collision Sensor
Runtime Behaviors	Items that represent runtime programming code in C#.

A right mouse click in the Physics Navigator offers the following items:

Feature	Description
Collapse All	Collapse all items of the structure displayed in the Function Navigator.
Expand All	Expand all items of the structure displayed in the Function Navigator.
Export to Browser	Export the structure to HTML that can be displayed in a browser. This is especially helpful to export the Sensor- and Actuator List.
Export to Spreadsheet	Export the structure to Excel. This is especially helpful to export the Sensor- and Actuator List.
Columns	Select columns that show addition information about the items in the list.
Properties	Select columns that show addition information about the items in the list.

3.3 The Sequence Editor

The Sequence Editor shows all Operations that have been defined for the mechatronics system. Basically there are two types of operations:

-  Operations that are purely triggerd by the time. They start and end at the point of time that is indicated by their position on the time scale. These operations are shown in the time line with a gray bar (1).
-  Operations that are somehow dependent on a not time based event. That could be:
  - A certain event fires and meets the starting condition of the operation.
  - For operations that control a position constraint: the position and the speed determine when the operation ends.

  These operations are shown as a green bar (2).

Both types of operation can be linked (3). Just drag a line from one operation and drop it on the other. The link can be deleted with a right-click.

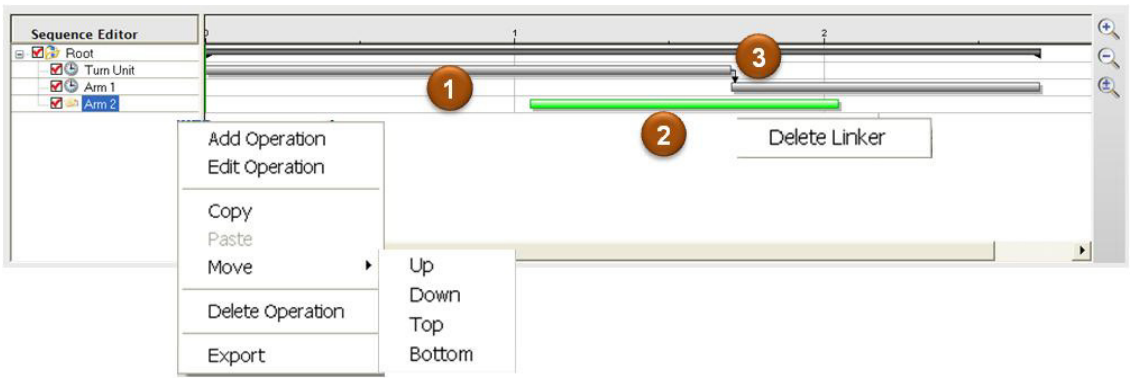


Figure 51: The Sequence Editor

A right-click on an Operation offers the following authoring features.

Menu item	Description
Add Operation	Adds a new operation to the mechatronics system.
Edit Operation	Edit the selected operation
Copy	Copy the selected operation.
Paste	Paste the operation in the clipboard.
Move	Move an operation: up, down, to the top or to the bottom.
Delete Operation	Delete the selected operation
Export	Export the operations to the PLCOpen XML format which has been standardized by the AutomationML group.

4 Basic Features

4.1 Define design requirements

The initial step for a design or redesign of a mechatronics system is the management of requirements. This comprises:

- Building requirement specification structures
- Create a requirement specification
- Importing a requirement specification structure from Microsoft Office Word
- Creating a requirement specification structure from a template
- Viewing a requirement specification structure
- Editing requirement content

The document (2) in chapter 2 gives deeper insights how to perform these tasks with Teamcenter.

To start working with Teamcenter the document (1) provides a general overview of Teamcenter.

4.2 Create Functional Model

To create the Functional Model in Teamcenter or Mechatronics Concept Designer it is helpful to have a guideline. The document (3) introduces the concept of a functional decomposition in order to structure mechatronics data of a machine.

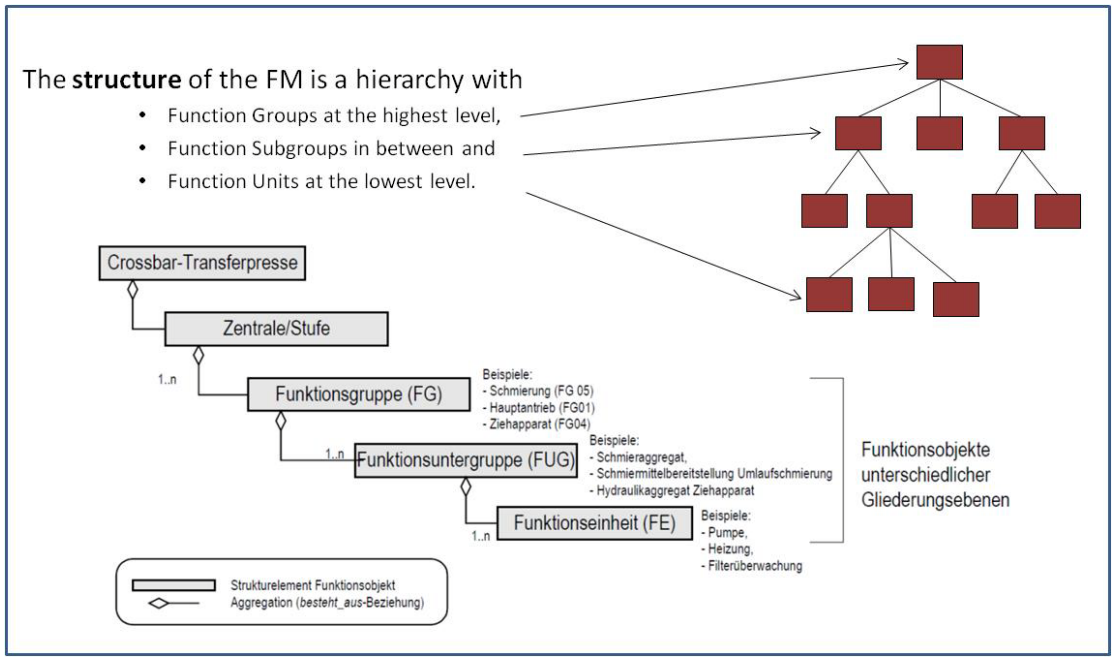


Figure 52: Functional decomposition according to the VDW - reference (3)

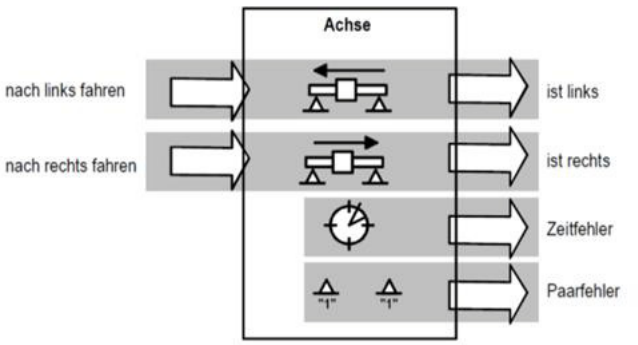


Figure 53: Functional Object according to the VDW - reference (3)

The basic elements are Functional Objects

- They contain data of multiple disciplines.
- They provide a dedicated interface.
- They are reusable by classification and instantiation.
- A functional object contains sensors and actuators that are changing and measuring physical values.

The document (2) in chapter 3 shows how to link items like functions with requirements.

4.3 Define Mechanical Concept (Geometry)

The idea of Mechatronics Concept Designer is to implement the elements of the functional structure with mechanical elements. These elements will be added to the corresponding function in the function tree.

4.3.1 Define basic principles of the machine based on the rough geometries

The basic mechanical design can be designed using the modeling features of CAD. The document (4) will provide additional information how to design in a CAD environment.

- Chapter 1: Introduction to NX
- Chapter 2: Selecting objects
- Chapter 3: Sketching
- Chapter 4: Modeling basics
- Chapter 5: Assemblies basics
- Chapter 6: Reuse Library

4.3.2 Create association link between components and function model

Mechanical elements have to be mapped to the function structure (see chapter 3.1 on page 36). There a basically three mechanisms to add a component to the components folder of the function tree:

Add Component

Adds a component to the function that is not yet part of the current assembly. This component (part file) can be selected in Teamcenter or in the file system and will be added to the assembly and to the selected component folder.

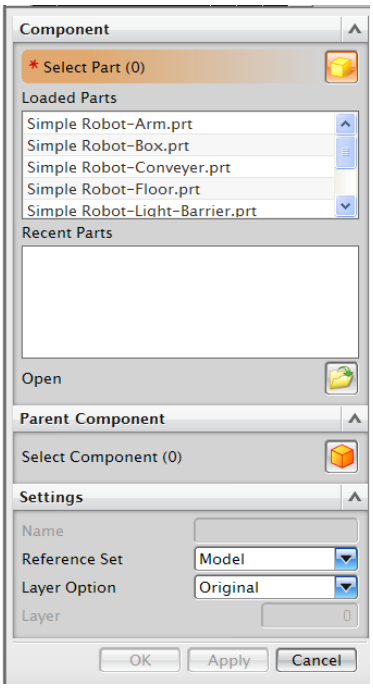


Figure 54: Dialog to add a component

For further reference see document (4) chapter “Add Component Overview” in page 245.

Create New Component

Creates a new empty part file that is added to the current assembly and add this to the selected component folder.

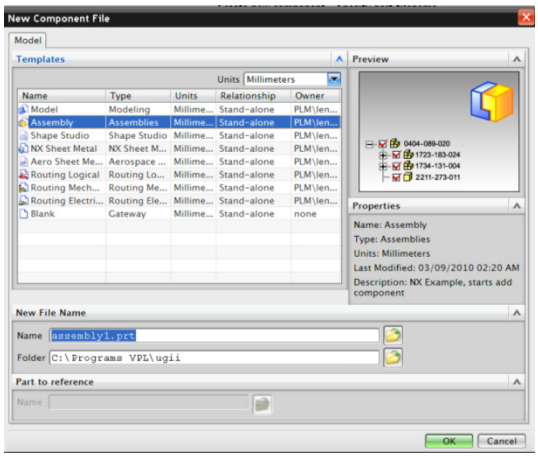


Figure 55: Dialog to add a new component

For further reference see document (4) in chapter “Create New Component overview” on page 242.

Add Existing Component

Pick a component from the current assembly. Just select it in the 3D representation.

4.4 Define Mechanical Concept (Physics)

Physics properties will be added to define the machine behavior from a mechanical point of view. This is the foundation for the interaction with other disciplines and the simulation of the mechatronics system.

4.4.1 Add Physics Properties

4.4.1.1 Rigid Body

A rigid body is an element of the system that can move around and act as mass in the mechatronics system. Once a geometric object is a defined to be a rigid body it responds to forces, e.g. it is falling down due to the gravity. A rigid body has a dynamic state:

State	Mathematical expression	Initial Value
Position	$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$	Defined by the position of the selected 3D geometries
Orientation	$\begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}$	Defined by the position of the selected 3D geometries
Translational Velocity	$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$	Default: 0. User defined (see below).
Angular Velocity	$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$	Default: 0. User defined (see below).
Mass	m	Default: automatically determined based on some standard material properties. User defined (see below).
Inertia	$\begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$	Default: automatically determined based on some standard material properties. User defined (see below).

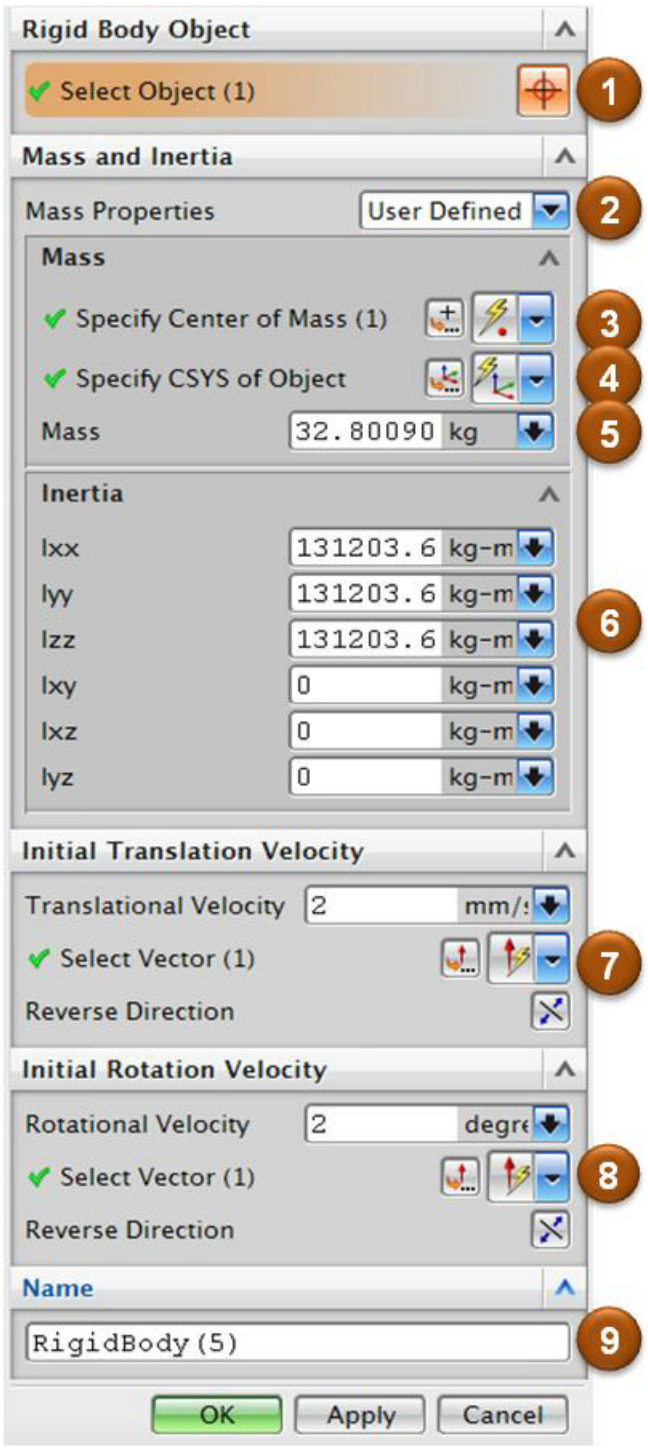
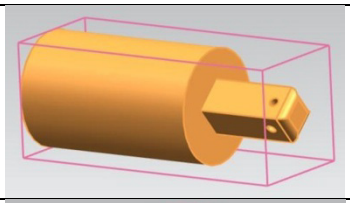
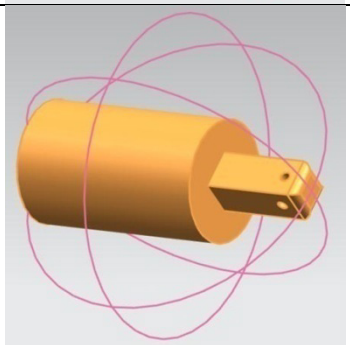
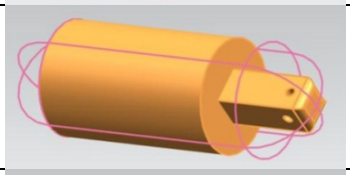
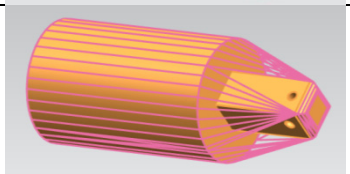
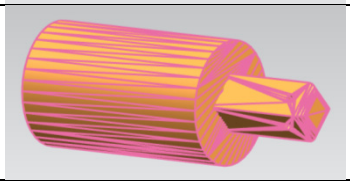


Figure 56: Definition of a rigid body

	Parameter	Description
1	Objects	Pick one or more 3D objects. All objects combined will act as one rigid body. For further reference how to select objects see document (4) chapter “Selecting objects” on page 13.
2	Mass Properties	Usually it is the best to set this parameter to “Automatic”. In this case Mechatronics Concept Designer will care for the calculation of the mass properties. “User defined” requires the user to enter the corresponding parameters manually.
3	Center of Mass	Select a point that will be the center of mass of the rigid body. For further reference how to select objects see document (4) chapter “Selecting objects” on page 13.
4	CSYS of the Object	Define the coordinate system that is the reference for the definition of the inertia parameters. For further reference regarding CSYS definition see document (4) in chapter “Datum CSYS overview” on page 141.
5	Mass	The mass that takes effect at the “Center of Mass”.
6	*Inertia*	*Define the Inertia matrix* $\begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$.
7	Translational Velocity	Initial translational velocity $\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$ of the rigid body defined by a scalar and a vector.
8	Rotational Velocity	Initial rotational velocity $\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$ of the rigid body defined by a scalar and a vector.
9	Name	User defined name of the rigid body.

4.4.1.2 Collision Body

Mechatronics Concept Designer provides a very efficient calculation of collisions. The reason for the high performance is the use of simplified collision shapes. Mechatronics Concept Designer supports the following collision shapes:

Type	Picture	Performance of the calculation
Box		High
Sphere		High
Capsule		High
Convex		Medium
Mesh *Not supported.*		Low

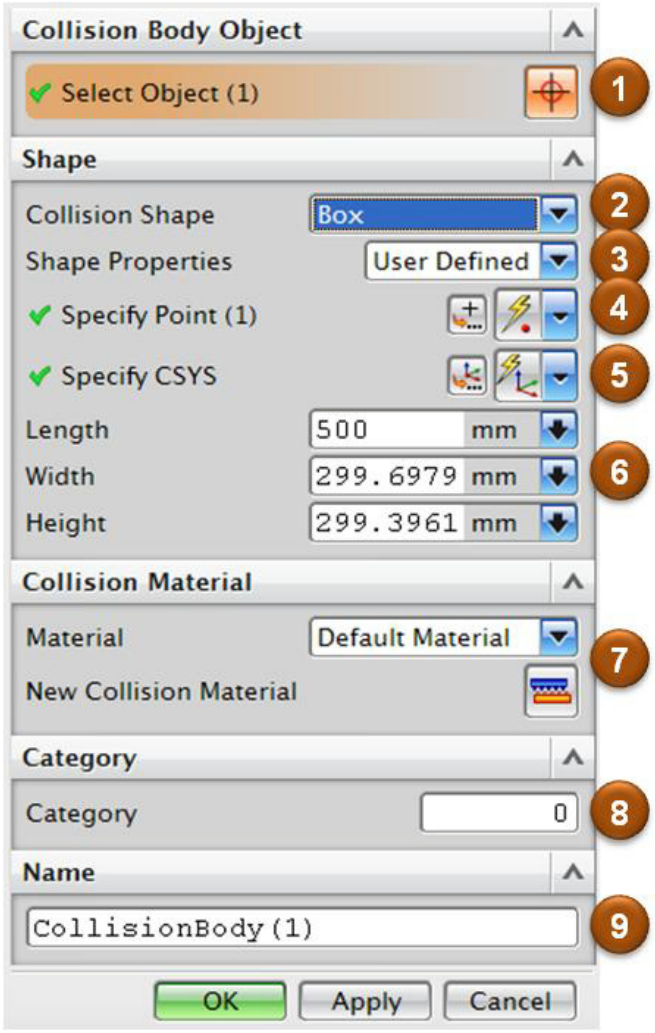


Figure 57: Definition of a collision body

A collision shape is defined by the following parameters:

	Parameter	Description
1	Objects	Pick one or more 3D objects. The collision shape will be calculated considering all selected objects. For further reference how to select objects see document (4) chapter “Selecting objects” on page 13.
2	Collision Shape	Type of collision shape. See above.
3	Shape properties	By default the shape properties are automatically calculated. “User defined” requires the user to enter the necessary parameters.
4	Point	Center point of the selected type of collision shape. For further reference how to select objects see document (4) chapter “Selecting objects” on page 13.
5	CSYS	CSYS (local coordinate system) for the collision shape. For further reference regarding CSYS definition see document (4) in chapter “Datum CSYS overview” on page 141.
6	Basic dimensions of the collision shape	Dimensions of the collision shape. These will differ depending on the kind of shape selected.
7	Material	The material determines the following behavior: • Static friction • Dynamic friction • Restitution (bouncing)
8	Category	Only collision bodies in the same category are considered to collide with each other. If there are many bodies in the scene this feature helps to reduce the time to calculate all collisions.
9	Name	Name of the collision body.

To define the collision shape of complex bodies it helps to separately select the facets of these bodies and assign a collision shape for each.

4.4.1.3 Transport Surface

The Transport Surface is a physical property that turns any selected flat surface into a kind of “conveyer belt”. Once another body lies on this surface it is transported into a specific direction.

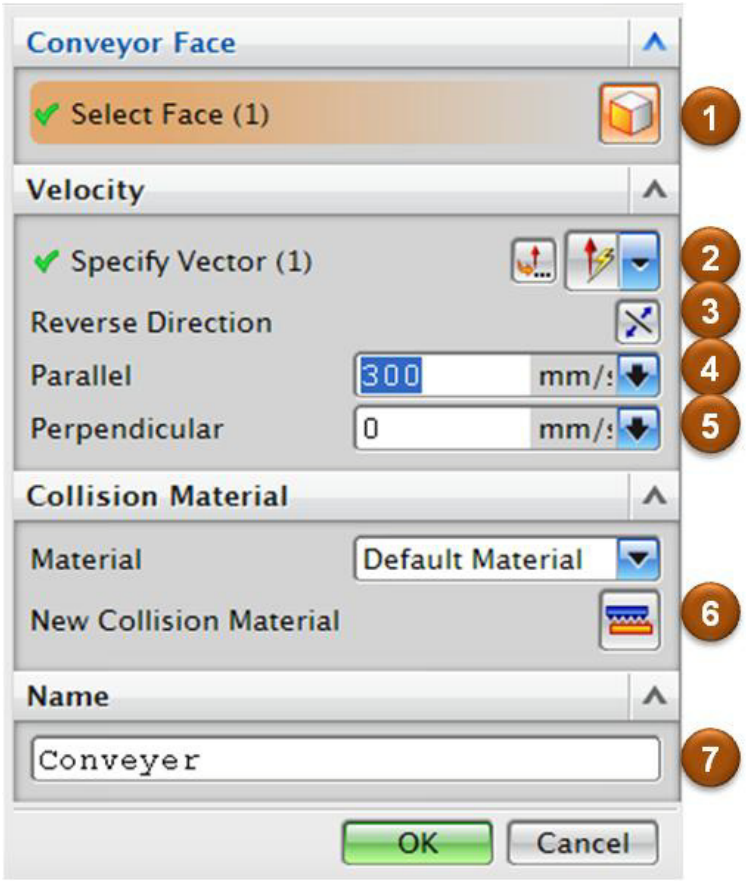


Figure 58: Definition of a transport surface

	Parameter	Description
1	Face	Pick the face of a body. For further reference how to select objects see document (4) chapter “Selecting objects” on page 13.
2	Vector	Vector that is pointing into the direction of the transport. For further reference how to select objects see document (4) chapter “Selecting objects” on page 13.
3	Reverse direction	Change the direction of the vector.
4	Parallel	Speed in the direction of the selected vector.
5	Perpendicular	Speed in the perpendicular direction of the selected vector.
6	Material	The material determines the following behavior: • Static friction • Dynamic friction • Restitution (bouncing)
7	Name	Name of the transport surface

4.4.1.4 Collision Material

The Collision Material is referred to by collision bodies and transport surfaces. It determines the basic behavior of a collision.

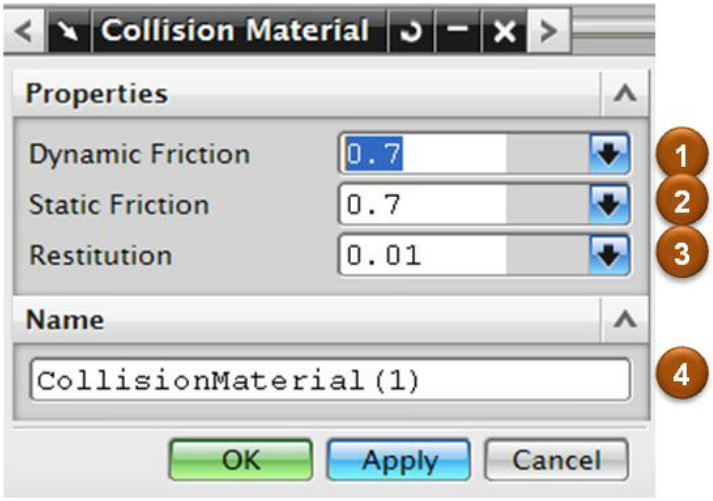


Figure 59: Definition of the collision material

	Parameter	Description
1	Dynamic Friction	Friction of body that is already moving.
2	Static Friction	Friction of a body that is not moving.
3	Restitution	Capability of the material to absorb the force that is applied or reflect it.
4	Name	Name of the Material

4.4.1.5 Object Source

Object Source and Object Sink are very powerful features to handle the multiple appearances of the same object. This is especially the case for material flow use cases. The Object Source creates objects in a specific time interval and the Object Sink deletes objects that are colliding. The example below shows

how multiple boxes are created by the Object Source and how they are deleted once they collide with the Object Sink.

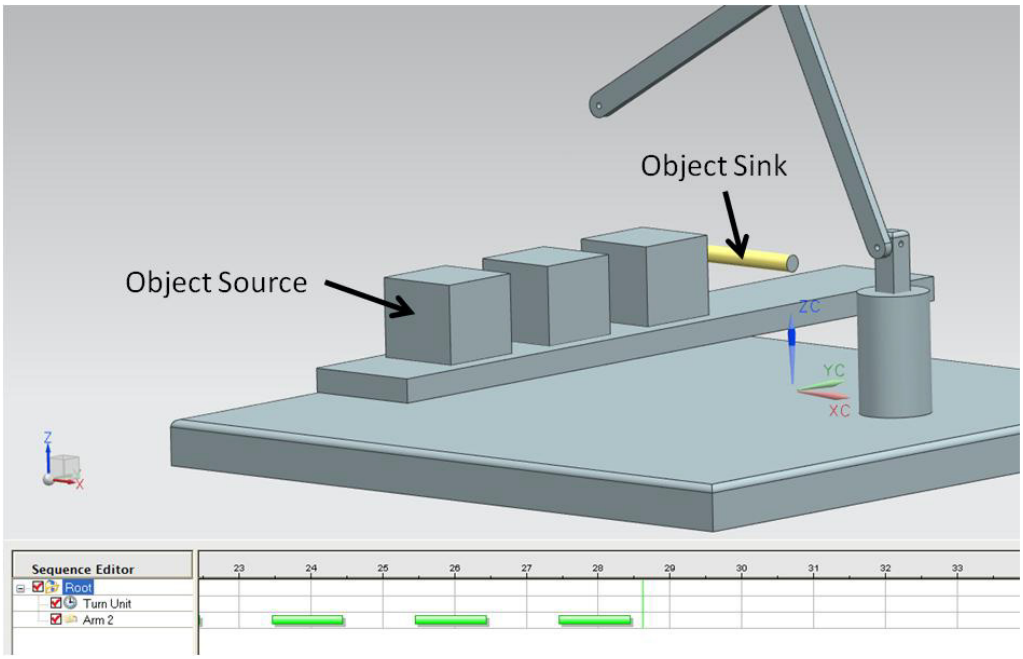


Figure 60: Example for Object Source and Object Sink

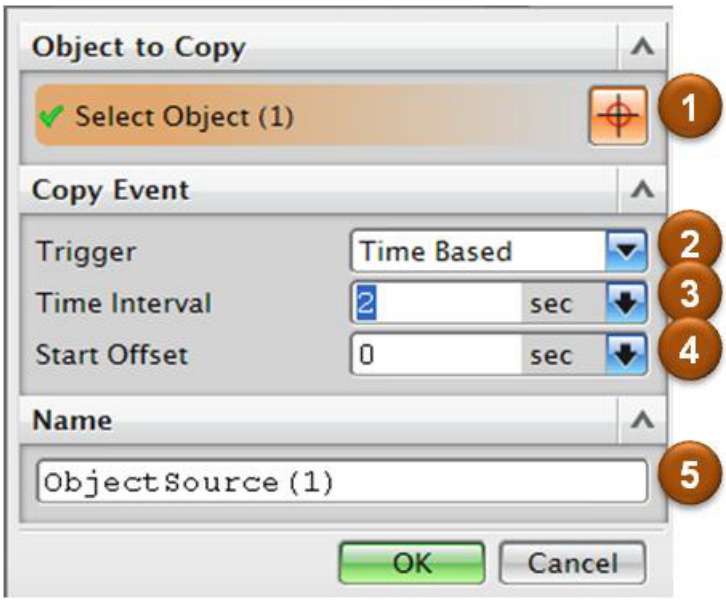


Figure 61: Definition of an Object Source

	Parameter	Description
1	Object	Select an object that should replicate itself automatically. For further reference how to select objects see document (4) chapter “Selecting objects” on page 13.
2	Trigger	• Replication in a specific time interval or • only one time per activation
3	Time Interval	Time interval in secs
4	Start Offset	How many seconds to wait until the first object is created.
5	Name	Name of the Object Source

4.4.1.6 Object Sink

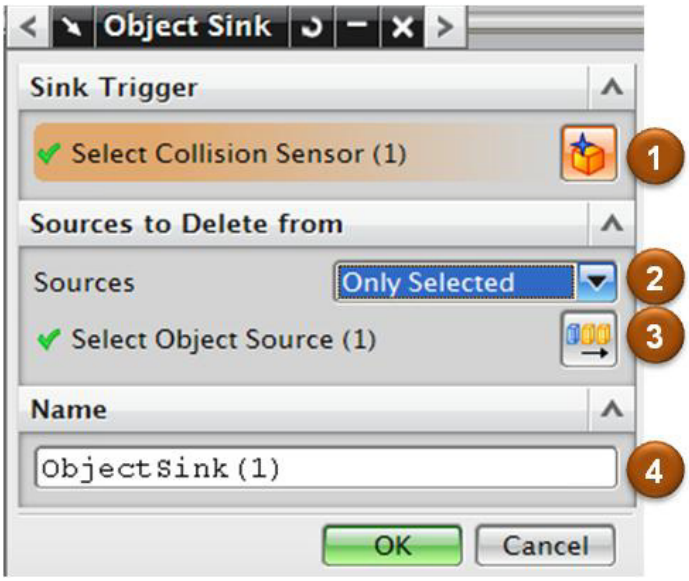


Figure 62: Definition of an Object Sink

	Parameter	Description
1	Collision Sensor	Select a collision sensor (see chapter 4.8.1 on page 81). For further reference how to select objects see document (4) chapter “Selecting objects” on page 13.
2	Sources	Determines if only objects generated by a specific source should be considered by the Object Sink.
3	Object Source	Only objects from this source are deleted by the Object Sink.
4	Name	Name of the Object Sink.

4.4.2 Add Kinematic Behavior

Joints are the basic elements to build a kinematic chain out of rigid bodies. They define how two rigid bodies are linked and which kind of movement they perform relative to each other.

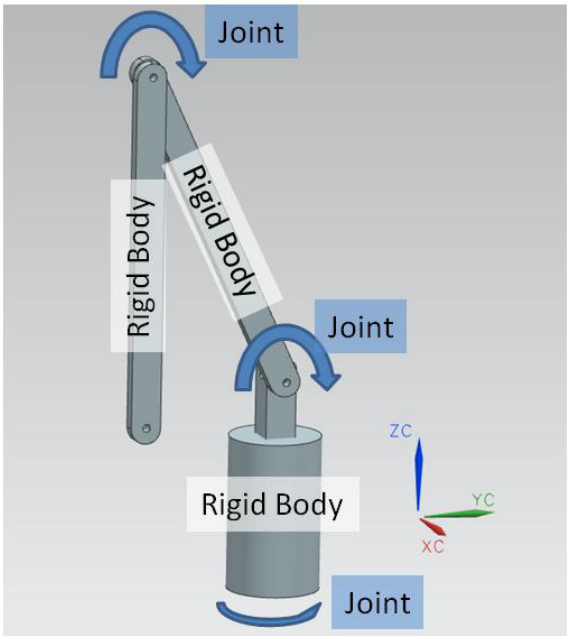


Figure 63: Example for a kinematic chain

4.4.2.1 Hinge Joint

The hinge joint performs a fixed rotation along an axis.

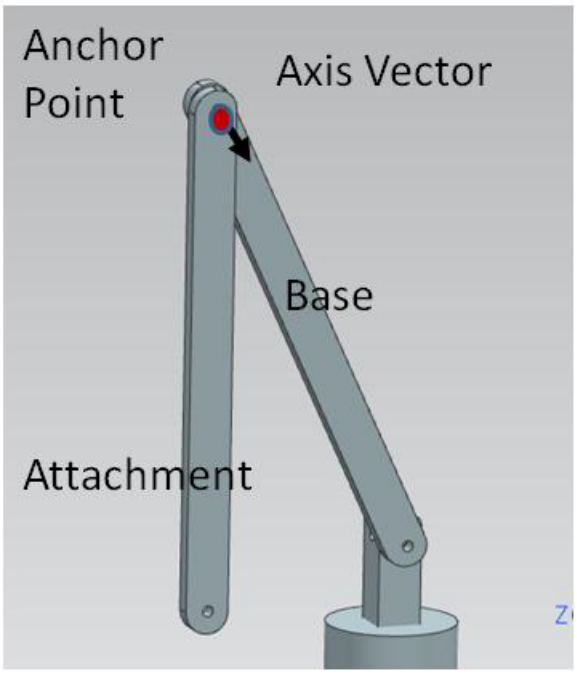


Figure 64: Basic parameters of a hinge joint

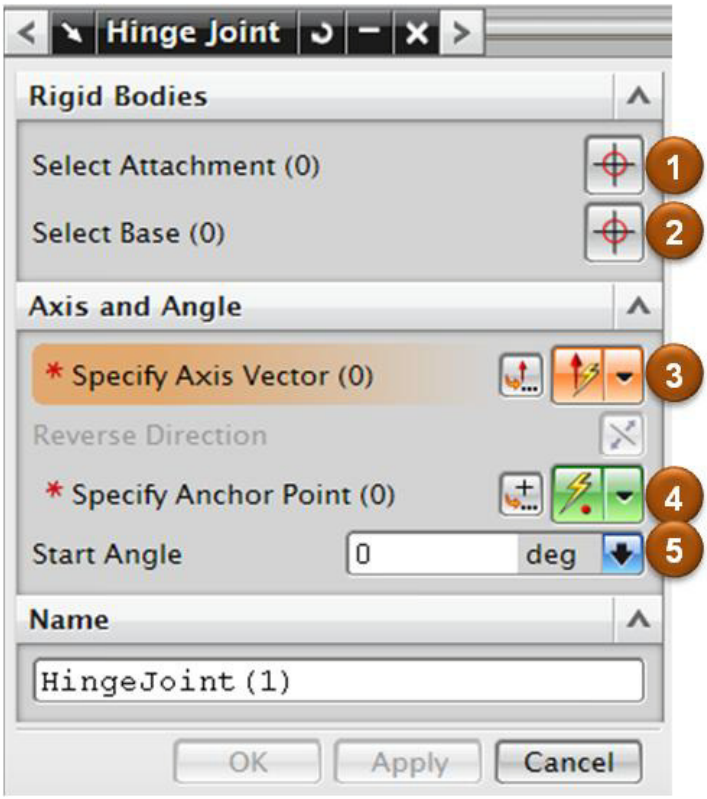
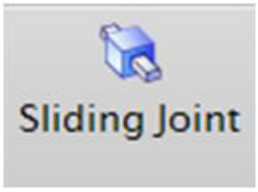


Figure 65: Basic parameters of a hinge joint

	Parameter	Description
1	Attachment	Rigid Body that should be constrained by the hinge joint.
2	Base	Rigid Body that this Attachment should be linked to. If this parameter is empty the Attachment is linked to the background.
3	Axis Vector	The vector that is the axis for the rotation.
4	Anchor Point	Anchor point of the vector.
5	Start Angle	Initial position of the attachment relative to its position on the screen when the simulation is not running.

4.4.2.2 Sliding Joint



The sliding joint performs a fixed linear movement along a vector but does not allow the bodies to rotate with respect to each other.

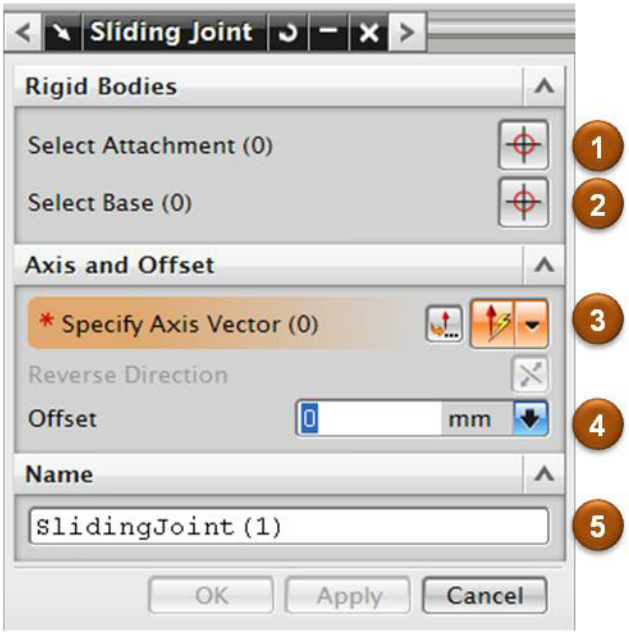
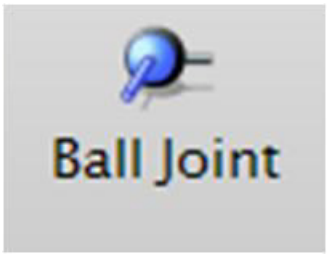


Figure 66: Definition of a Sliding Joint

	Parameter	Description
1	Attachment	Rigid Body that should be constrained by the sliding joint.
2	Base	Rigid Body that this Attachment should be linked to. If this parameter is empty the Attachment is linked to the background.
3	Axis Vector	The vector that is the axis for the linear movement.
4	Offset	Initial position of the attachment relative to its position on the screen when the simulation is not running.
5	Name	Name of the sliding joint.

4.4.2.3 Ball Joint



A Ball Joint constraints two bodies connected at a point, both can rotate freely. It is comparable to a Hinge Joint (see chapter 4.4.2.1) but there is no need to define a vector for the motion.

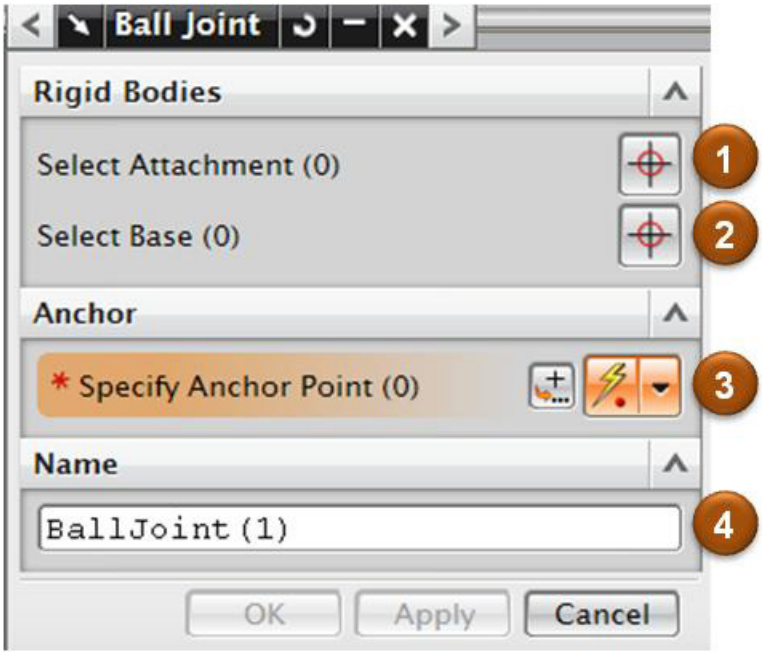


Figure 67: Definition of a Ball Joint

	Parameter	Description

1	Attachment	Rigid Body that should be constraint by the sliding joint.
2	Base	Rigid Body that this Attachment should be linked to. If this parameter is empty the Attachment is linked to the background.
3	Anchor Point	Anchor point of the two rigid bodies
4	Name	Name of the Ball Joint.

4.4.2.4 *Cylindrical Joint*

The Cylindrical Joint rotates like the Hinge Joint but can also telescope along the rotation axis.

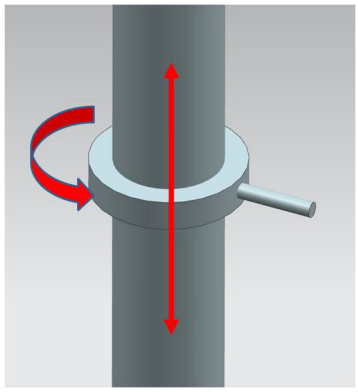


Figure 68: Example of a Cylindrical Joint

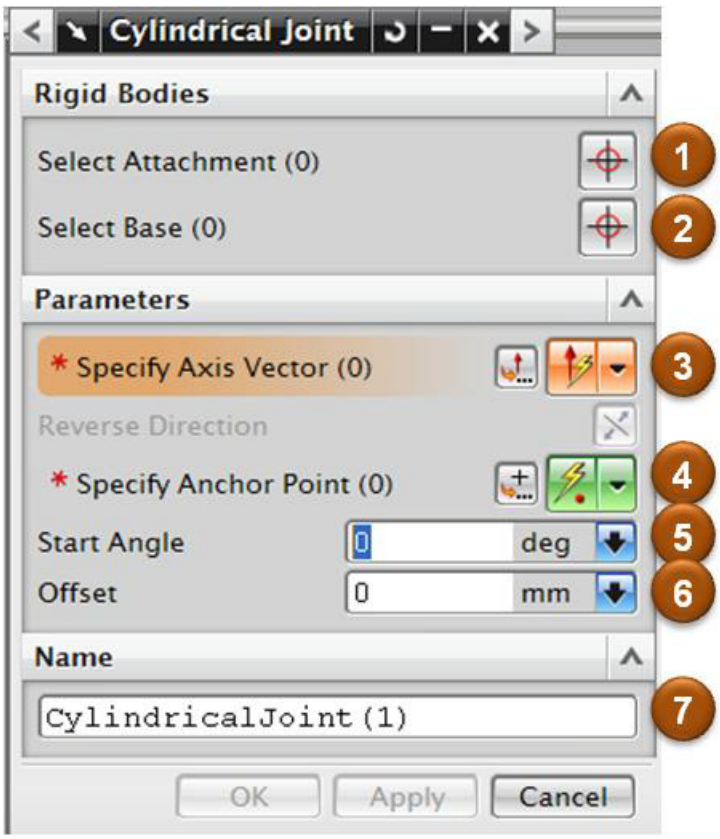
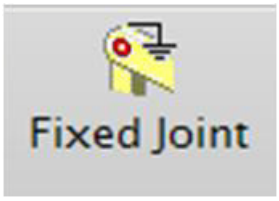


Figure 69: Basic parameters of a Cylindrical Joint

	Parameter	Description
1	Attachment	Rigid Body that should be constraint by the hinge joint.
2	Base	Rigid Body that this Attachment should be linked to. If this parameter is empty the Attachment is linked to the background.
3	Axis Vector	The vector that is the axis for the rotation.
4	Anchor Point	Anchor point of the vector.
5	Start Angle	Initial position of the attachment relative to its position on the screen when the simulation is not running with respect to the turn movement.
6	Offset	Initial position of the attachment relative to its position on the screen when the simulation is not running with respect to the linear movement.
7	Name	Name of the Cylindrical Joint

4.4.2.5 Fixed Joint



The Fixed Joint provides a rigid connection of two rigid bodies.

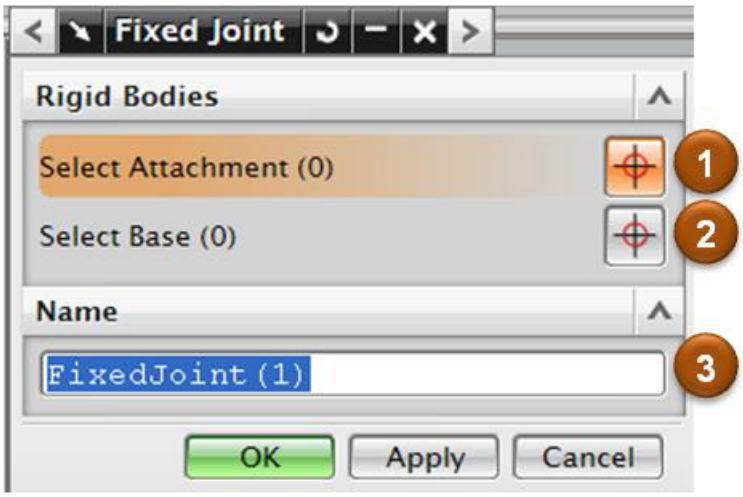


Figure 70: Basic parameters of a Cylindrical Joint

	Parameter	Description
1	Attachment	Rigid Body that should be constraint by the hinge joint.
2	Base	Rigid Body that this Attachment should be linked to. If this parameter is empty the Attachment is linked to the background.
3	Name	Name of the Fixed Joint

4.4.2.6 Angular Spring Joint

The Angular Spring Joint generates a torque when angles between bodies change. These joints are functional but do not have much user interface. The angle is determined by two vectors that are relative to each of the objects being attached – the vectors' direction is in the global coordinate space. When the constrained objects move, the assigned vector maintains its relative position to the object. The angle

between the vectors determines the value the joint is going to be restricted. Springs include both a spring force and a damping parameter. The angle is unsigned, that is the objects can move in either direction to achieve the max value in a limit constraint or apply a force in a spring.

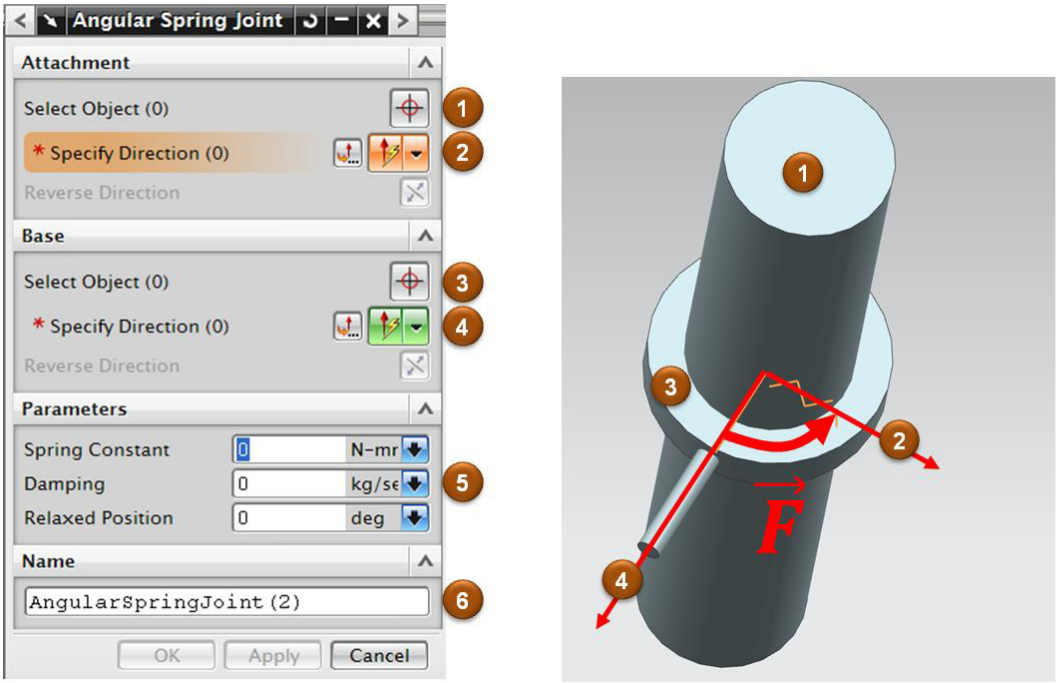


Figure 71: Basic parameters of an Angular Spring Joint

	Parameter	Description
1	Attachment Object	First Rigid Body.
2	Attachment Direction	Vector that will serve to determine the angle between the attachment and the base body.
3	Base Object	Second Rigid Body.
4	Base Direction	Vector that will serve to determine the angle between the attachment and the base body.
5	Parameters of the spring	• Spring Constant • Damping • Relaxed Position
6	Name	Name of the Angular Spring Joint

4.4.2.7 Linear Spring Joint

The Linear Spring Joint generates an opposing force when bodies travel apart. This constraint works similar like the Angular Spring Joint. While the Angular Spring Joint is looking at the angle of two rotating object, the Linear Spring Joint is looking at the distance between two moving objects.

These joints are functional but do not have much user interface. For linear joints, the distance is determined by offset points – the points specified in the global space. When the constrained objects move, the assigned point maintains its relative position to the object. The distance between the points determines the value the joint is going to be restricted. Springs include both a spring force and a damping parameter. The distance is unsigned, that is the objects can move in either direction to achieve the max value in a limit constraint or apply a force in a spring.

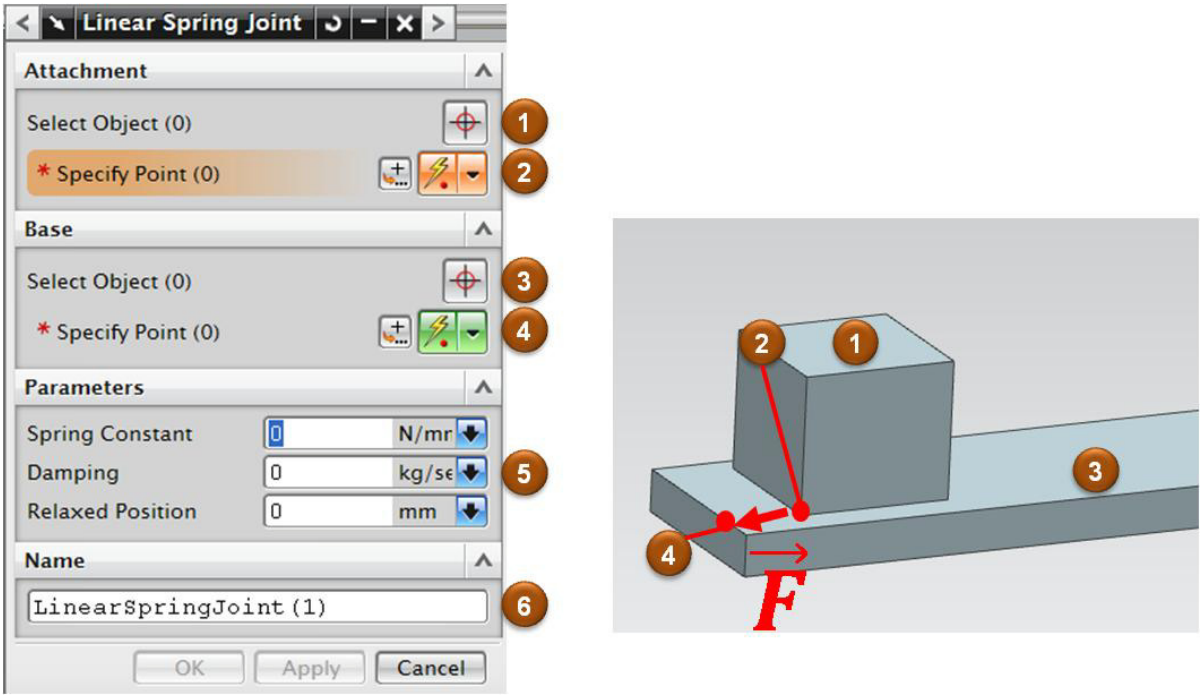
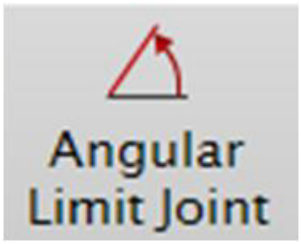


Figure 72: Basic parameters of a Linear Spring Joint

	Parameter	Description
1	Attachment Object	First Rigid Body.
2	Attachment Point	Point of the attachment that will serve to determine the distance between the attachment and the base body.
3	Base Object	Second Rigid Body.
4	Base Point	Point of the base that will serve to determine the distance between the attachment and the base body.
5	Parameters of the spring	• Spring Constant • Damping • Relaxed Position
6	Name	Name of the Linear Spring Joint

4.4.2.8 Angular Limit Joint



The Angular Limit Joint prevents bodies from rotating past a given angle. These joints are functional but do not have much user interface. The angle is determined by two vectors that are relative to each of the objects being attached – the vectors' direction is in the global coordinate space. When the constrained objects move, the assigned vector maintains its relative position to the object.

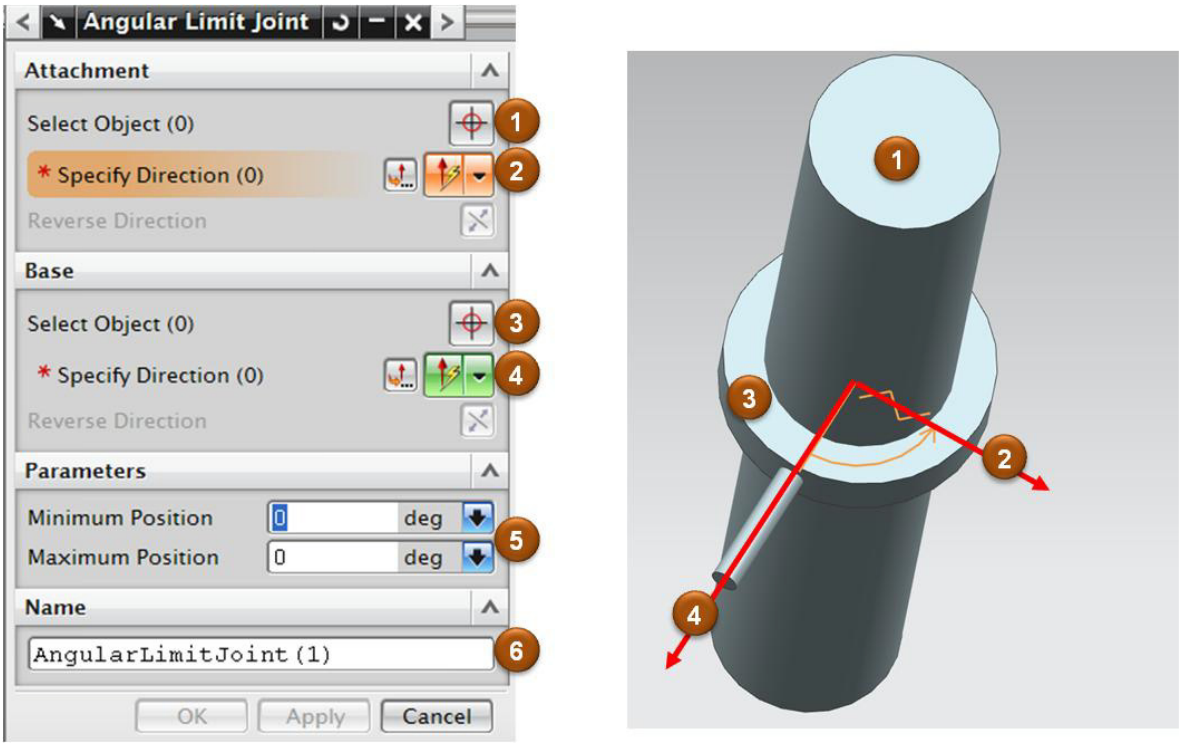
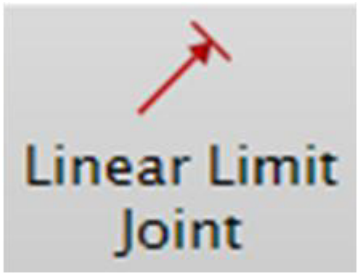


Figure 73: Basic parameters of an Angular Limit Joint

	Parameter	Description
1	Attachment Object	First Rigid Body.
2	Attachment Direction	Vector that will serve to determine the angle between the attachment and the base body.
3	Base Object	Second Rigid Body.
4	Base Direction	Vector that will serve to determine the angle between the attachment and the base body.
5	Limiting parameters	• Minimum angle • Maximum angle
6	Name	Name of the Angular Spring Joint

4.4.2.9 Linear Limit Joint



The Linear Limit Joint prevents bodies from moving beyond a given distance. For linear joints, the distance is determined by offset points – the points specified in the global space. When the constrained objects move, the assigned point maintains its relative position to the object.

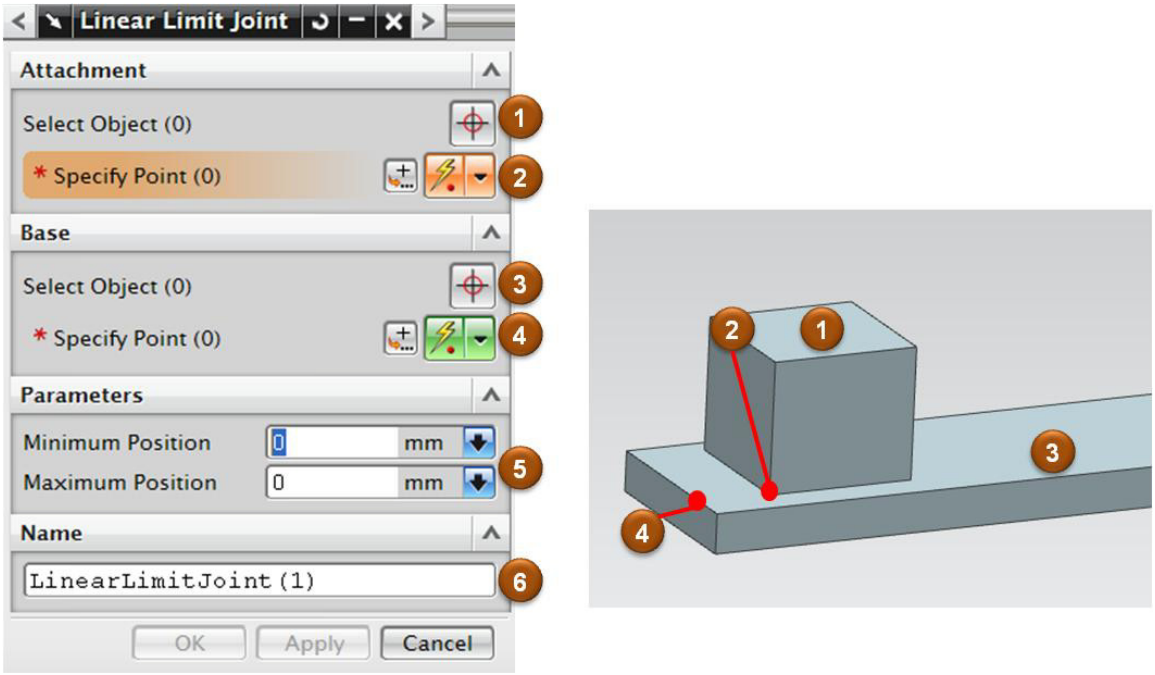
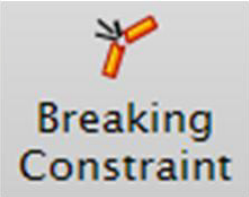


Figure 74: Basic parameters of a Linear Limit Joint

	Parameter	Description
1	Attachment Object	First Rigid Body.
2	Attachment Point	Point of the attachment that will serve to determine the distance between the attachment and the base body.
3	Base Object	Second Rigid Body.
4	Base Point	Point of the base that will serve to determine the distance between the attachment and the base body.
5	Limiting parameters	• Minimum position • Maximum position
6	Name	Name of the Linear Spring Joint

4.4.3 Constraints

4.4.3.1 Breaking Constraint



The Breaking Constraint defines the maximum force or torque that breaks a specific joint. “Breaking” means that this joint will no longer constrain the motion of the former linked rigid bodies.

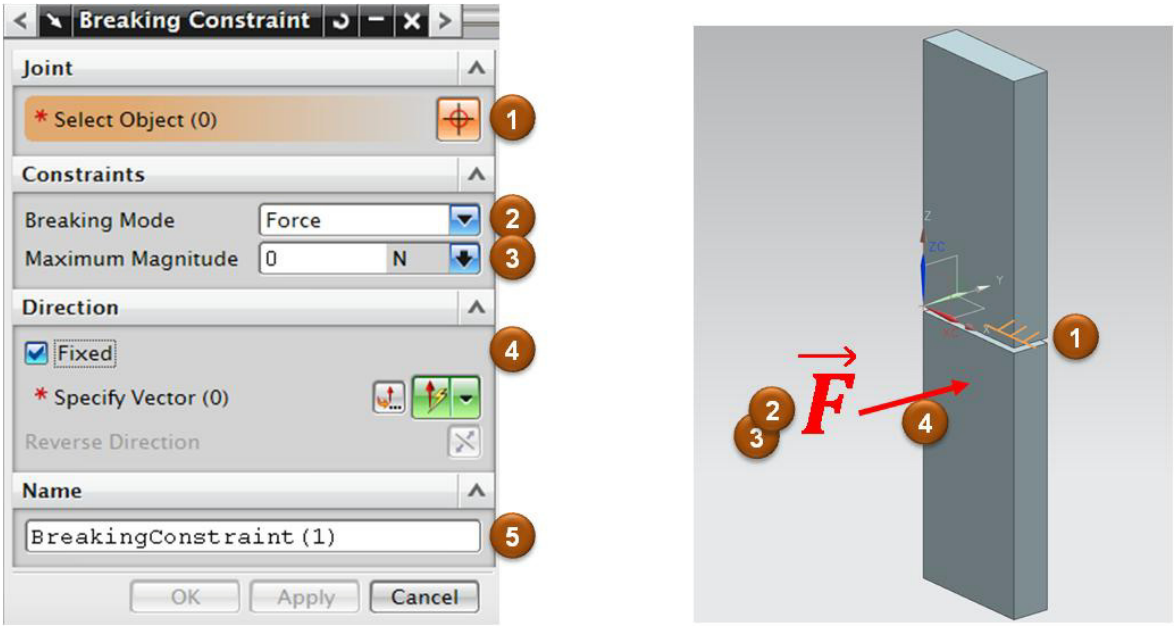


Figure 75: Basic parameters of a Breaking Constraint

	Parameter	Description
1	Object	Select a joint.
2	Breaking Mode	Defines if the breaking condition is driven by force or torque.
3	Maximum Magnitude	Maximum value of the torque or the force.
4	Vector	Optional vector that defines the direction of the force. This fixed direction of the breaking constraint is relative to the base of the joint. If no vector is defined forces from any direction are considered.
5	Name	Name of the Breaking Constraint

4.4.3.2 Prevent Collision

The Prevent Collision Constraint defines that two bodies (with a collision shape) no longer collide with each other. In this case they two bodies will permeate each other. Prevent Collision can also be used to prevent a collision with a Collision Sensor (see chapter 4.8.1 on page 81).

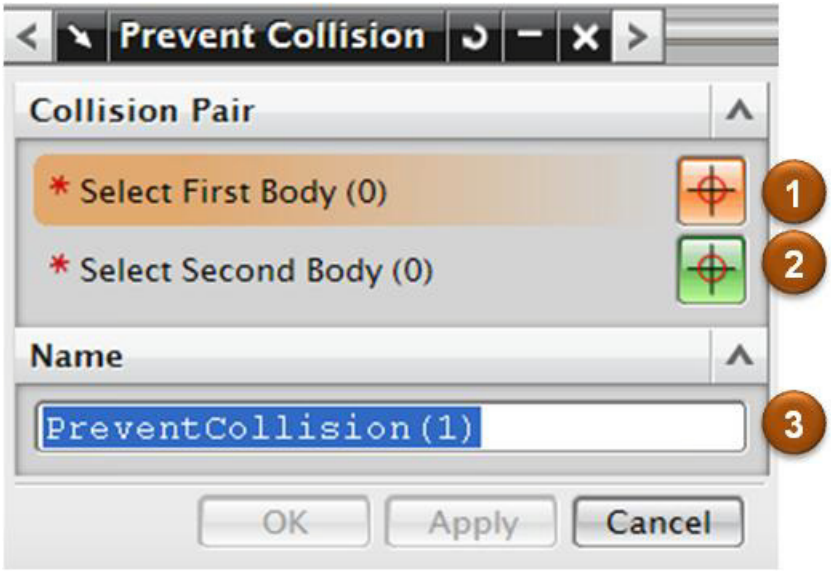
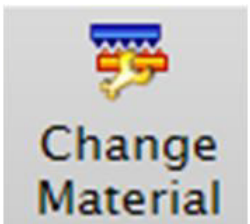


Figure 76: Basic parameters of a Prevent Collision Constraint

	Parameter	Description
1	First Body	First of two bodies
2	Second Body	Second of two bodies
3	Name	Name of the Prevent Collision Constraint

4.4.3.3 Change Material



While “Prevent Collision” avoids collision behavior of two bodies, “Change Material” defines the collision behavior of two specific bodies. The collision behavior is defined by the Material.

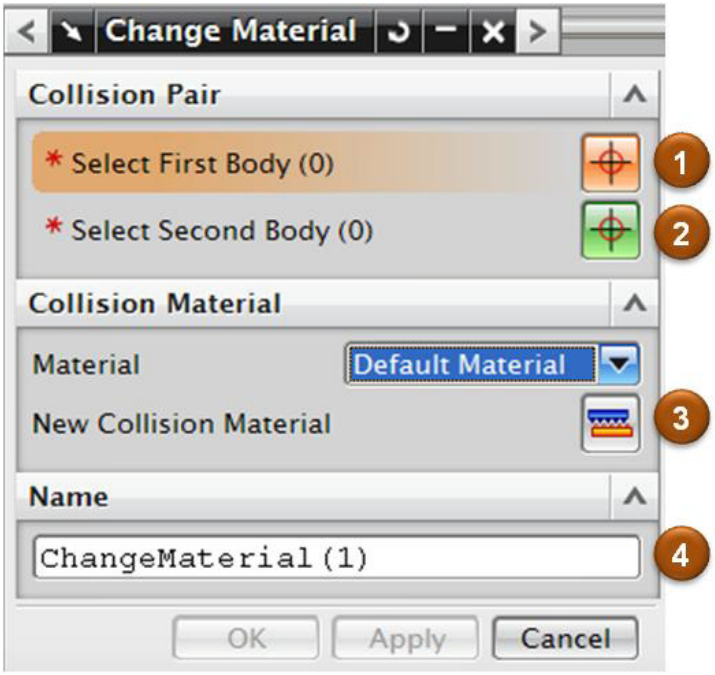


Figure 77: Basic parameters to define the material for the collision of two bodies

	Parameter	Description
1	First Body	First of two bodies
2	Second Body	Second of two bodies
3	Collision Material	Select a predefined material from the list or create a new material that is then automatically added to the list.
4	Name	Name of “Change Material”

4.5 Add Generalized Actuators

The Sensor- and Actuator List is built by adding Generalized Actuators. A “Generalized Actuator” means an actuator that is not matched with a real actuator, e.g. it has no 3D shape and no specific motor behavior like first-order time-delay or intrinsic inertia. Mechatronics Concept Designer offers two types of actuators: speed control and position control.

4.5.1 Speed Control

The Speed Control sets the movement of an axis (which is identified by a joint) to a predefined constant speed.

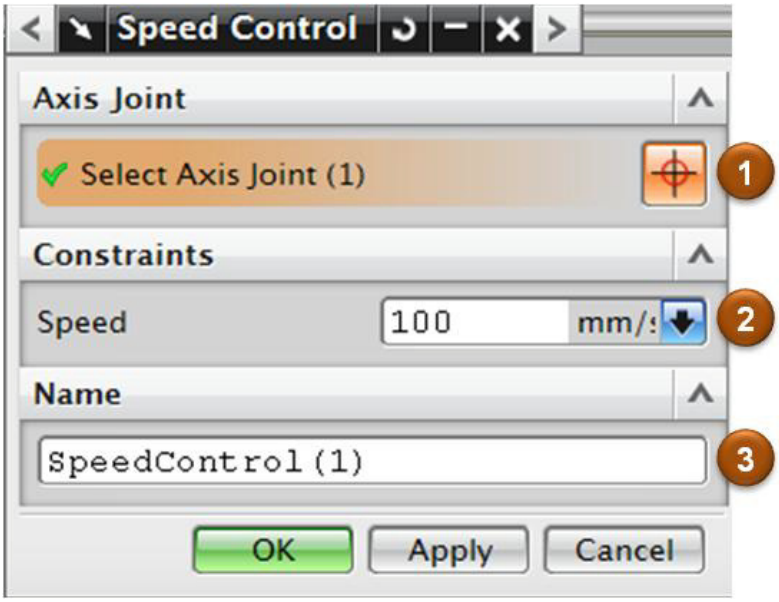


Figure 78: Basic parameters of the Speed Control

	Parameter	Description
1	Axis Joint	Joint that should be moved by the actuator.
2	Speed	Constant speed of the movement. Depending on 1 this is either an angular speed or a linear speed.
3	Name	Name of the Speed Control

4.5.2 Position Control

The Position Control moves an axis to a predefined position with a predefined speed. So the time to complete this movement will be $time = \frac{Position}{Speed}$.

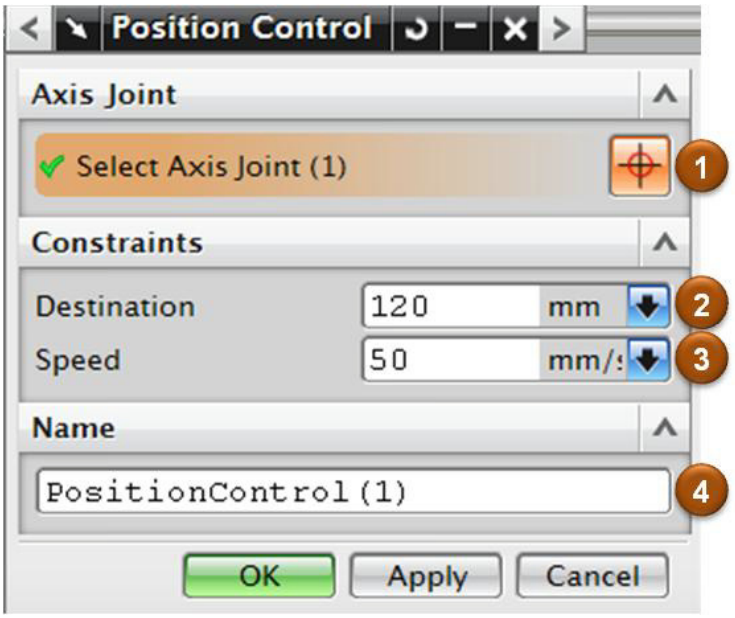


Figure 79: Basic parameters of the Position Control

	Parameter	Description
1	Axis Joint	Joint that should be moved by the actuator.
2	Position	Set position that the axis should be moved to.
3	Speed	Constant speed of the movement. Depending on 1 this is either an angular speed or a linear speed.
4	Name	Name of the Position Control

4.5.3 Export Sensor and Actuator List

As explained in chapter 3.2 on page 38 the sensor actuator list can be exported to an Excel or HTML format.

Name	Type	Owning Component Name
Basic Physics		
Turn Unit	Rigid Body	SIMPLE ROBOT-TURN UNIT
Turn Unit	Collision Body	SIMPLE ROBOT-TURN UNIT
Arm	Rigid Body	SIMPLE ROBOT-ARM
Arm	Collision Body	SIMPLE ROBOT-ARM
Box	Collision Body	SIMPLE ROBOT-BOX
Box	Rigid Body	SIMPLE ROBOT-BOX
Conveyer	Collision Body	SIMPLE ROBOT-CONVEYER
Conveyer	Transport Surface	SIMPLE ROBOT-CONVEYER
Floor	Collision Body	SIMPLE ROBOT-FLOOR
Floor	Rigid Body	SIMPLE ROBOT-FLOOR
Arm	Rigid Body	SIMPLE ROBOT-ARM
Arm	Collision Body	SIMPLE ROBOT-ARM
Joints and Constraints		
Floor	Fixed Joint	
Turn Unit	Hinge Joint	
Arm 1	Hinge Joint	
Arm 2	Hinge Joint	
Materials		
Couplers		
Sensors and Actuators		
Turn Unit	Position Control	
Arm 1	Position Control	
Arm 2	Position Control	
Light Barrier	Collision Sensor	
Runtime Behaviors		

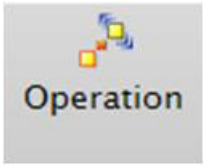
	A	B	C
1	Name	Type	Owning Component Name
2	Basic Physics		
3	Turn Unit	Rigid Body	SIMPLE ROBOT-TURN UNIT
4	Turn Unit	Collision Body	SIMPLE ROBOT-TURN UNIT
5	Arm	Rigid Body	SIMPLE ROBOT-ARM
6	Arm	Collision Body	SIMPLE ROBOT-ARM
7	Box	Collision Body	SIMPLE ROBOT-BOX
8	Box	Rigid Body	SIMPLE ROBOT-BOX
9	Conveyer	Collision Body	SIMPLE ROBOT-CONVEYER
10	Conveyer	Transport Surface	SIMPLE ROBOT-CONVEYER
11	Floor	Collision Body	SIMPLE ROBOT-FLOOR
12	Floor	Rigid Body	SIMPLE ROBOT-FLOOR
13	Arm	Rigid Body	SIMPLE ROBOT-ARM
14	Arm	Collision Body	SIMPLE ROBOT-ARM
15	Joints and Constraints		
16	Floor	Fixed Joint	
17	Turn Unit	Hinge Joint	
18	Arm 1	Hinge Joint	
19	Arm 2	Hinge Joint	
20	Materials		
21	Couplers		
22	Sensors and Actuators		
23	Turn Unit	Position Control	
24	Arm 1	Position Control	
25	Arm 2	Position Control	
26	Light Barrier	Collision Sensor	
27	Runtime Behaviors		
28			

Figure 80: Export in HTML and Excel format

4.6 Define time based controllers

Operations are the building blocks to define the time based and event based behavior of the mechatronics system. They are automatically added to the Sequence Editor (see chapter 3.3 on page 41).

4.6.1 Define operations



An operation is a control element that can access any object of the mechatronics system. Typically an operation will control an actuator like in the real world but it might also access other objects like joints and constraints. Every object provides an interface 3 which is exposed to the operation. When the operation is activated it will change the parameters of the object to the desired value.

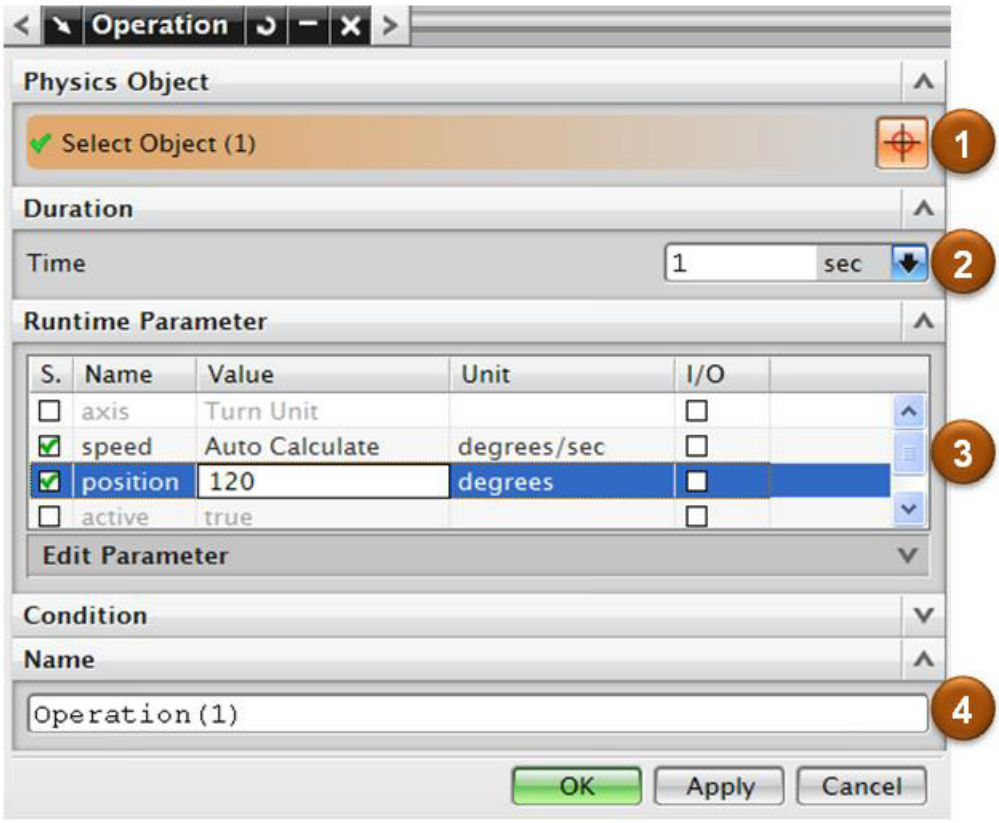


Figure 81: Basic parameters of the Operation

	Parameter	Description
1	Object	Select an actuator that will be controlled by this operation. It is also possible to select any other object and change parameters. *For example changing the attachment of a fixed joint can attach an object to a gripper at runtime.*
2	Time	Duration of the operation. The speed of a Position Constraint will be automatically calculated to match the time.
3	Interface	The interface section shows all accessible parameters of the object. By activating the parameter in the first column the corresponding parameter will be changed by the operation. The user has to define the value of the parameter. *The column I/O defined if this parameter will become a signal that is accessible from external applications. This is not yet implemented in Mechatronics Concept Designer.*
4	Name	Name of the Operation

4.6.2 Define sequence of operations

Chapter 3.3 on page 41 gives an overview how the Sequence Editor is used to define the sequence of operations.

4.7 Continuous behavior

Continuous behavior defines the coordination of multiple axes. In the past this has been realized by mechanical means. Today mechanical components are replaced by servo drives and intelligent motion control (e.g. SIEMENS SIMOTION). Eliminating mechanics from the mechanical system means a higher dynamic and performance. It also offers a higher degree of flexibility and less wear. Motion Control is becoming a decisive factor and Mechatronics Concept Designer helps to define the initial Motion Control Concept.

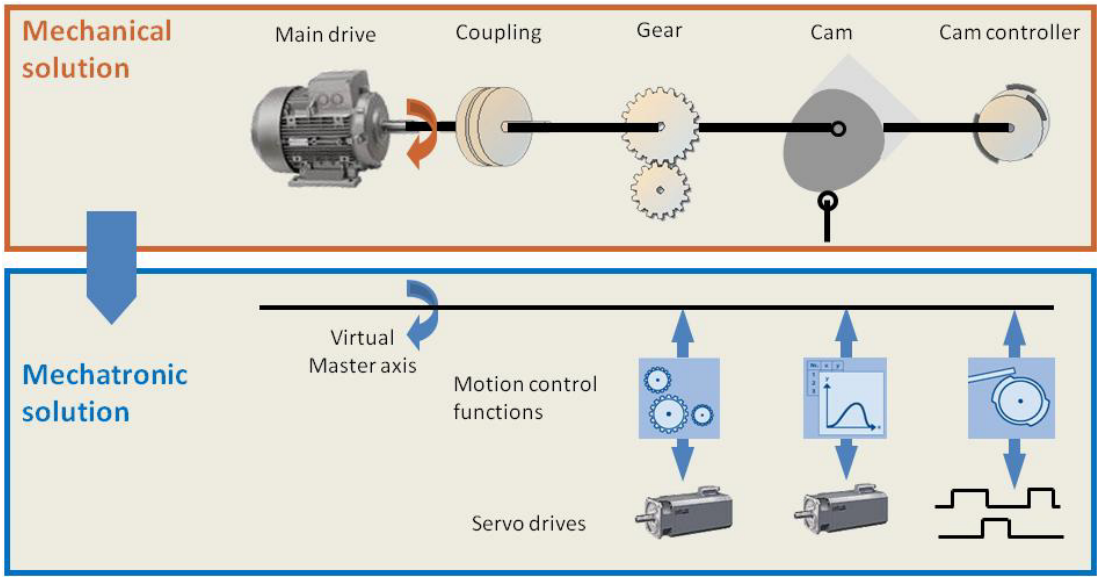


Figure 82: Mechanical solutions replaced by Mechatronics solutions

Mechatronics Concept Designer offers features to define and simulate cams and gears.

4.7.1 Define a Motion Profile

The Motion Profile is a graph that defines the continuous coordination curve between a master and a slave axis.

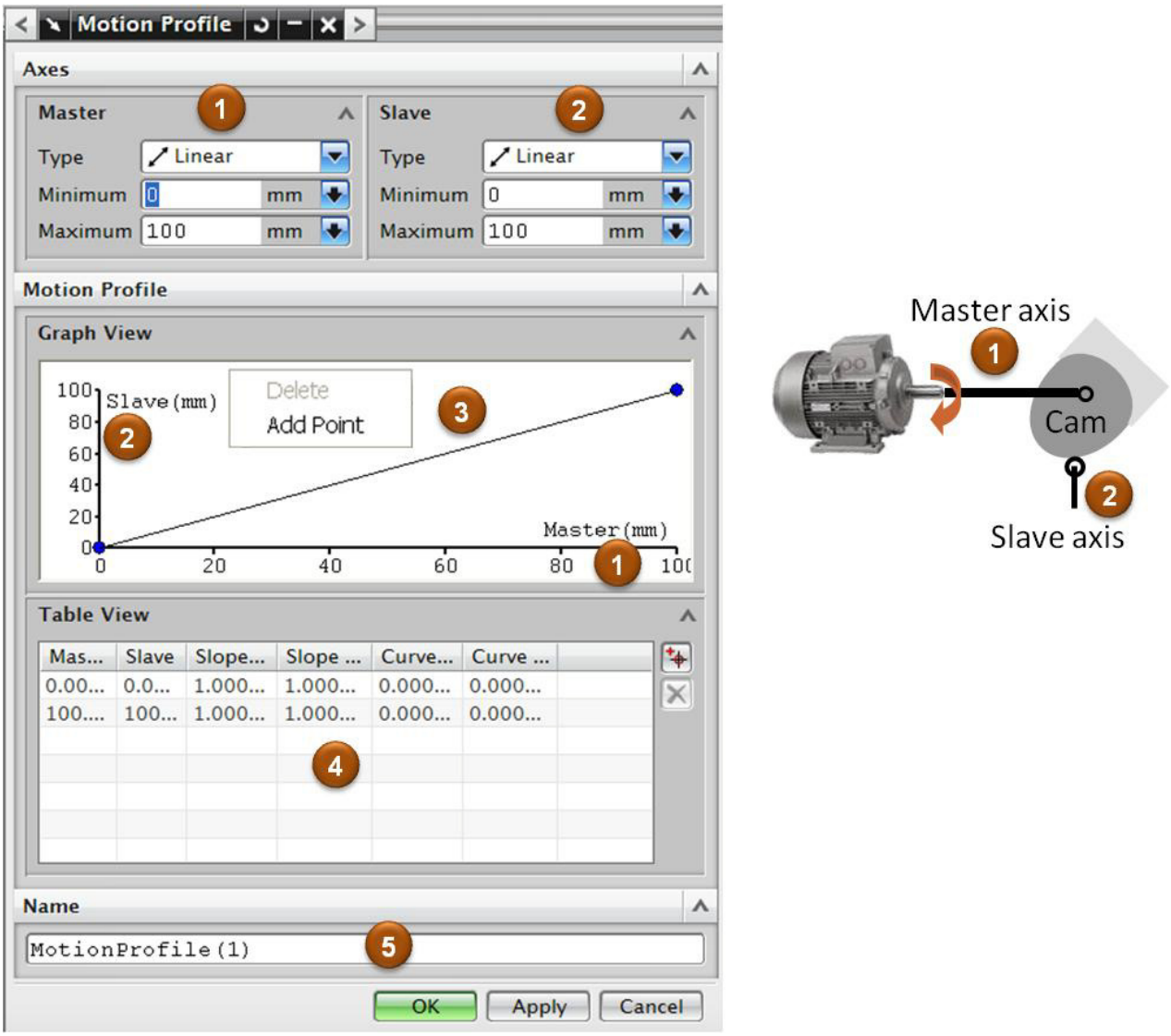


Figure 83: Definition of the Motion Profile

	Parameter	Description
	Master axis	Type of the master axis: • Linear axis • Rotary axis • Time (in this case the cam will serve as a start ramp) Minimum and maximum values the axes.
	Slave axis	Type of the slave axis: • Linear axis • Rotary axis Minimum and maximum values the axes.
	Curve	Curve that defines the cam. With right mouse click points can be added to the curve or be deleted from the curve. It is possible to drag points to shape the curve.
	Table of the points	This table shows all points of the curve. The values of the curve can be edited: • Master position Position on the master axis • Salve position Position on the slave axis • Slope in First derivative (speed) of the curve at the left side of the point. • Slope out First derivative (speed) of the curve at the right side of the point. • Curve in Second derivative (acceleration) of the curve at the left side of the point. • Curve out Second derivative (acceleration) of the curve at the right side of the point.
	Name	Name of the Motion Profile

A right click on a point provides a pull down menu to edit parameters of the corresponding point. These serve to make the curve as smooth as possible and define segments with constant speed.

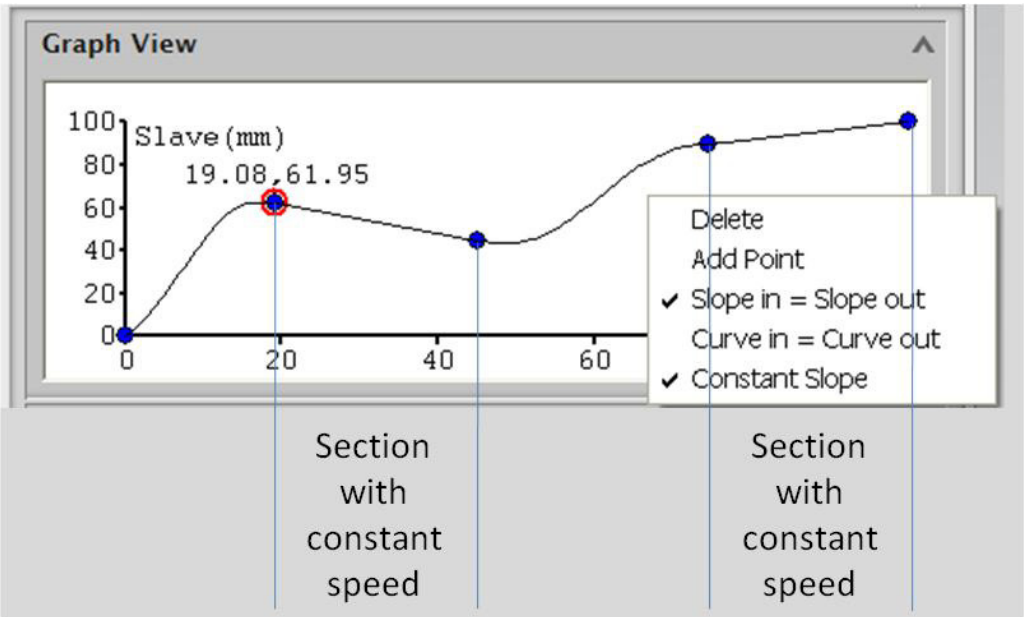
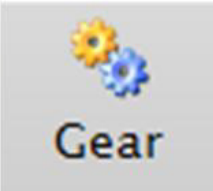


Figure 84: Definition of the segments of a motion profile

- Slope in = Slope out:
  Continuous first derivative at this point.
- Curve in = Curve out:
  Continuous second derivative at this point.
- Constant Slope:
  The segment to the right of this point will have constant slope.

4.7.2 Define a gear



A gear forces two rotating axes to maintain a constant turn ratio. It is a special case of a cam defined by a motion profile with constant speed.

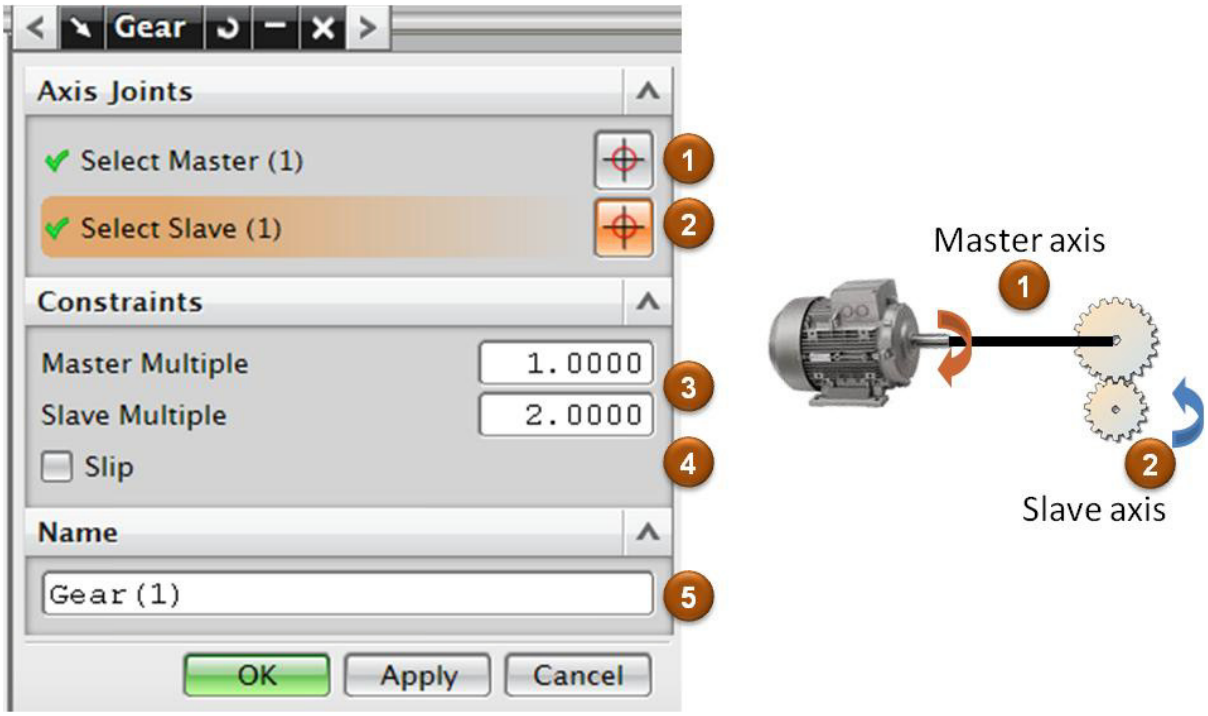
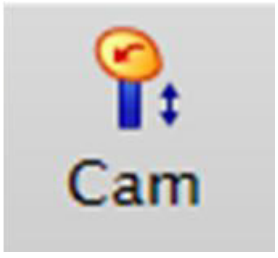


Figure 85: Basic parameters of a Gear

	Parameter	Description
1	Master	Select a rotary joint to be the master.
2	Slave	Select a rotary joint to be the slave.
3	Gear ratio	The gear ratio is defined by: $\frac{Master\ Multiple}{Slave\ Multiople}$
4	Slip	The gear will allow some slip which is the case for a belt drive.
5	Name	Name of the Gear

4.7.3 Define a cam



A Cam forces two axes to maintain a relationship determined by a graph (motion profile).

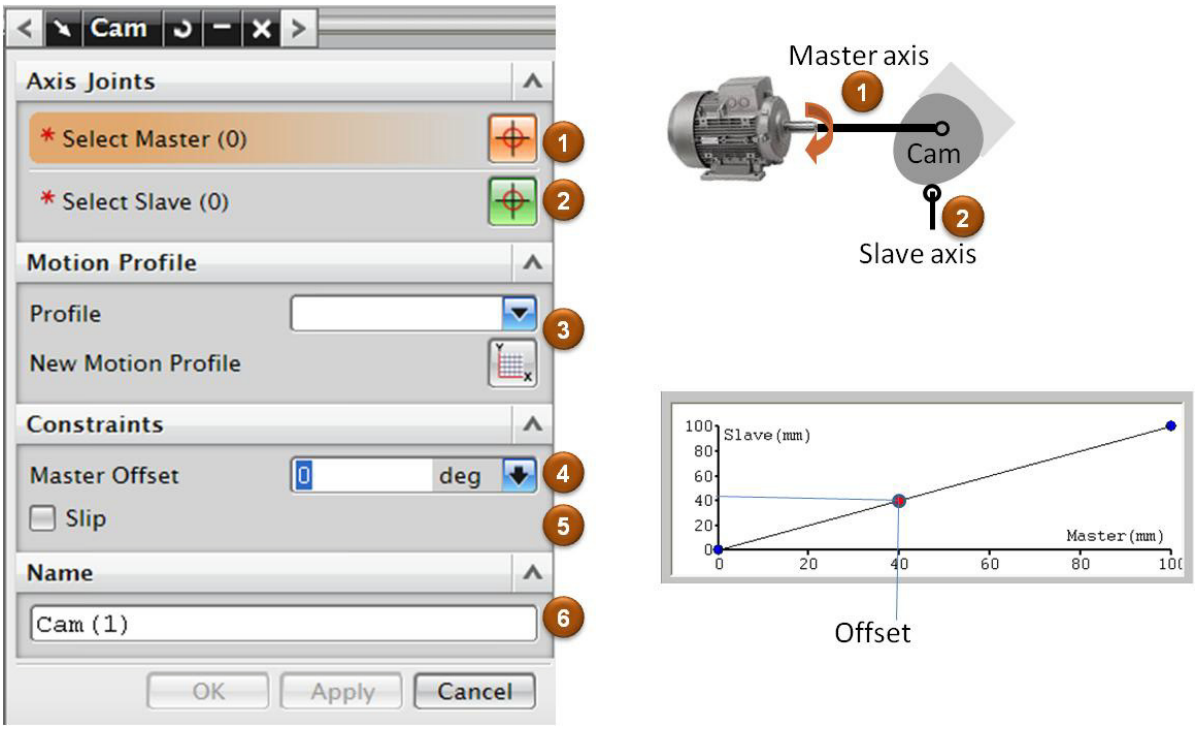


Figure 86: Basic parameters of a Cam

	Parameter	Description
1	Master	Select a joint to be the master.
2	Slave	Select a joint to be the slave.
3	Profile	Select a predefined Motion Profile from the list or create a new one.
4	Master offset	Defines the starting point of the cam on the master axis. *In this version of Mechatronics Concept Designer the corresponding starting point of the slave axis is not calculated. It will start at point 0 on the slave axis.*
5	Slip	The Cam will allow some slip which is the case for a belt drive.
6	Name	Name of the Cam

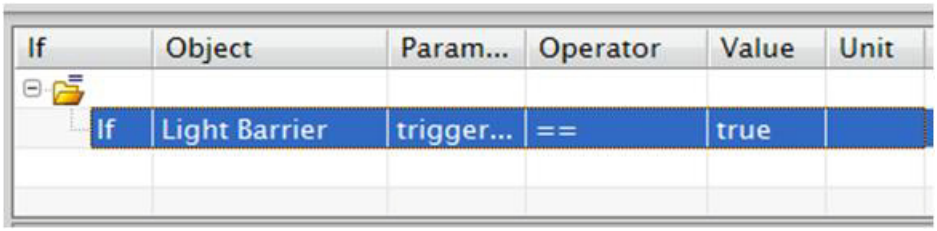
4.8 Add Generalized Sensors

To develop the event based behavior of the mechatronics system a runtime feedback is needed. Mechatronics Concept Designer offers two means to access runtime data

- Each object of the system model (e.g. actuators, joints, conveyers) provide an interface to access their specific data. Two examples:
  - A Position Controller provides speed and position.

- A Transport Surface provides parallel speed and perpendicular speed.

- To provide immediate feedback about the interaction of bodies in the scene Mechatronics Concept Designer allows to define a Collision Sensor.
  The Collision Sensor provides the runtime parameter “trigger”. It is set to “true” if a collision is detected.



4.8.1 Define a Collision Sensor

A Collision Sensor is attached to a body and detects the presence of other items within its bounds. The basic definition of a Collision Sensor is comparable to a Collision Body (see chapter 4.4.1.2 on page 48).

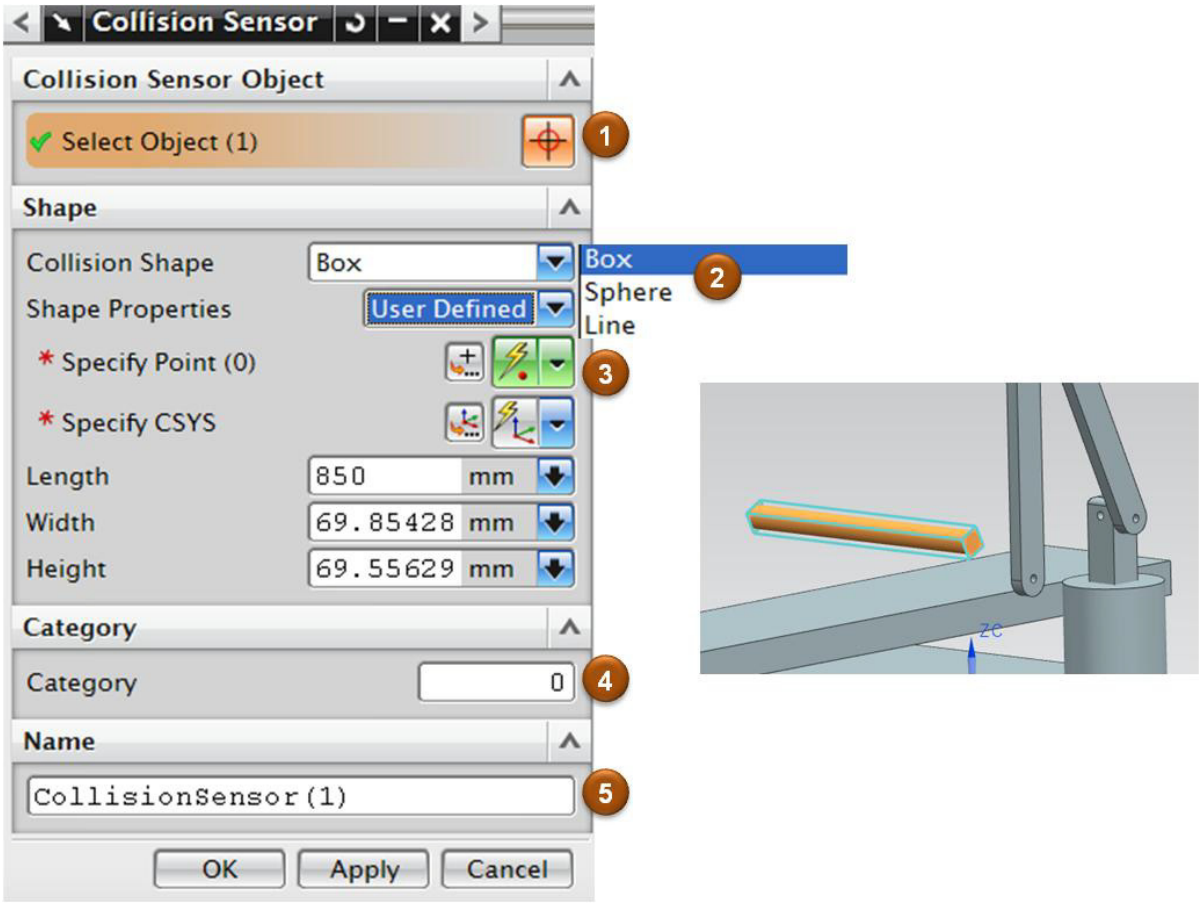


Figure 87: Basic parameters of a Collision Sensor

	Parameter	Description
1	Object	Pick one 3D objects.
2	Collision Shape	Type of collision shape • Box • Sphere • Line
3	Shape properties	By default the shape properties are automatically calculated. "User defined" requires the user to enter the necessary parameters.
	Point	Center point of the selected type of collision shape. For further reference how to select objects see document (4) chapter "Selecting objects" on page 13.
	CSYS	CSYS (local coordinate system) for the collision shape. For further reference regarding CSYS definition see document (4) in chapter "Datum CSYS overview" on page 141.
	Basic dimensions of the collision shape	Dimensions of the collision shape. These will differ depending on the kind of shape selected.
4	Category	Only collision bodies in the same category are considered to collide with each other. If there are many bodies in the scene this feature helps to reduce the time to calculate all collisions.
5	Name	Name of the Collision Sensor

4.8.2 Continue to build the sensor/actuator list

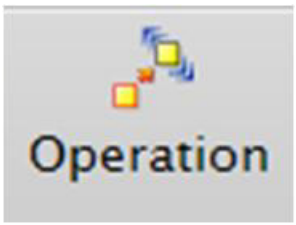
The Collision Sensor is added to the list of sensors and actuators (see chapter 3.2 and chapter 4.5.3).

4.9 Add event based behavior

Based on sensors the time based behavior (see chapter 4.6 on page 73) can be extended to event based behavior. The operations are not only invoked at a defined point of time. They dynamically respond to a certain event of the mechatronics system. This behavior can be defined

- by adding conditions to operation or
- by programming C# code based on events from the mechatronics system.

4.9.1 Add event based operations



Mechatronics Concept Designer distinguishes between three kinds of operations:

- For a purely time based operation both the start and end is determined by the time (1).
- A time based operation that is linked to a position constraint with defined speed and position has an undetermined duration. Time is automatically derived based on the runtime situation (2).

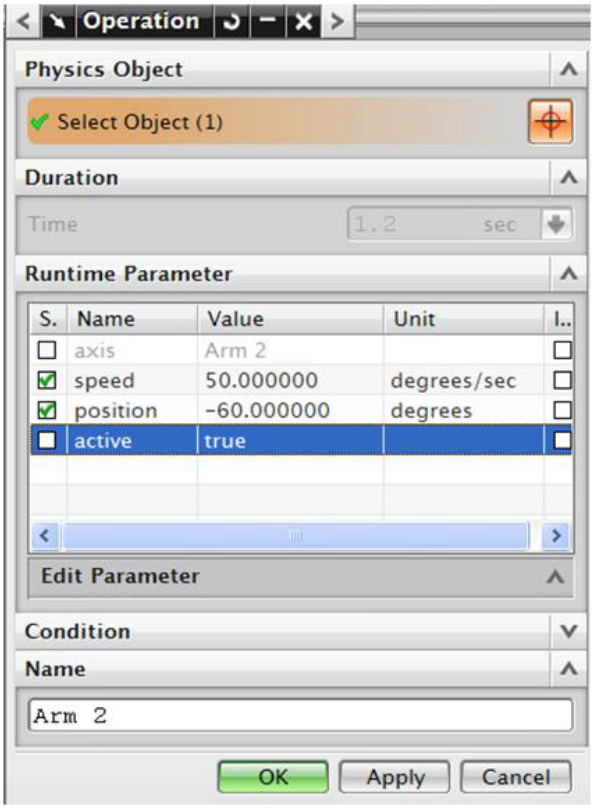


Figure 88: Operation with dynamic calculation of the time

- An event based operation is invoked by a certain event, e.g. a collision with a collision sensor. In this case both the point of time for start and end are not predetermined (3). The earliest point of time the trigger condition is considered, is the start time on the time scale. If you want that operation to be sensitive for the event all the time, move it to the beginning of the time scale.

The Sequence Editor shows the time based operation as a gray bar, the time based operation with undetermined end as blue bar and the event based operation as a green bar.

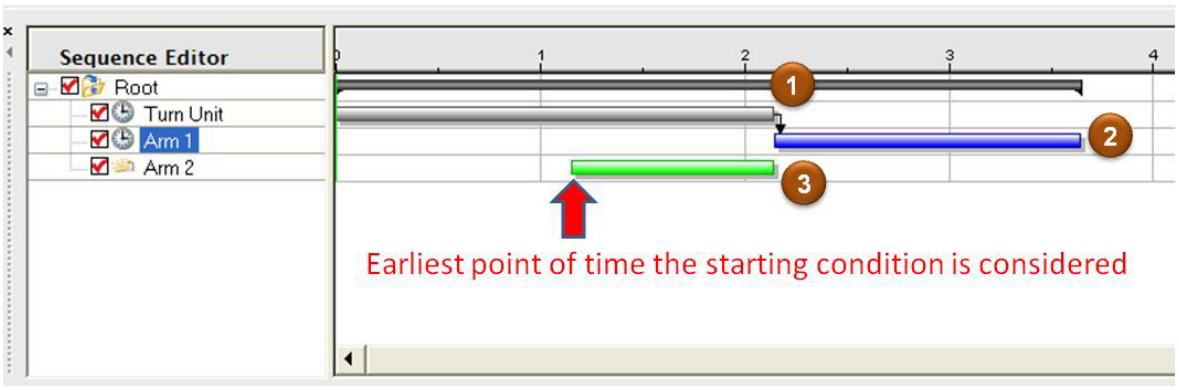


Figure 89: Time based and event based operations

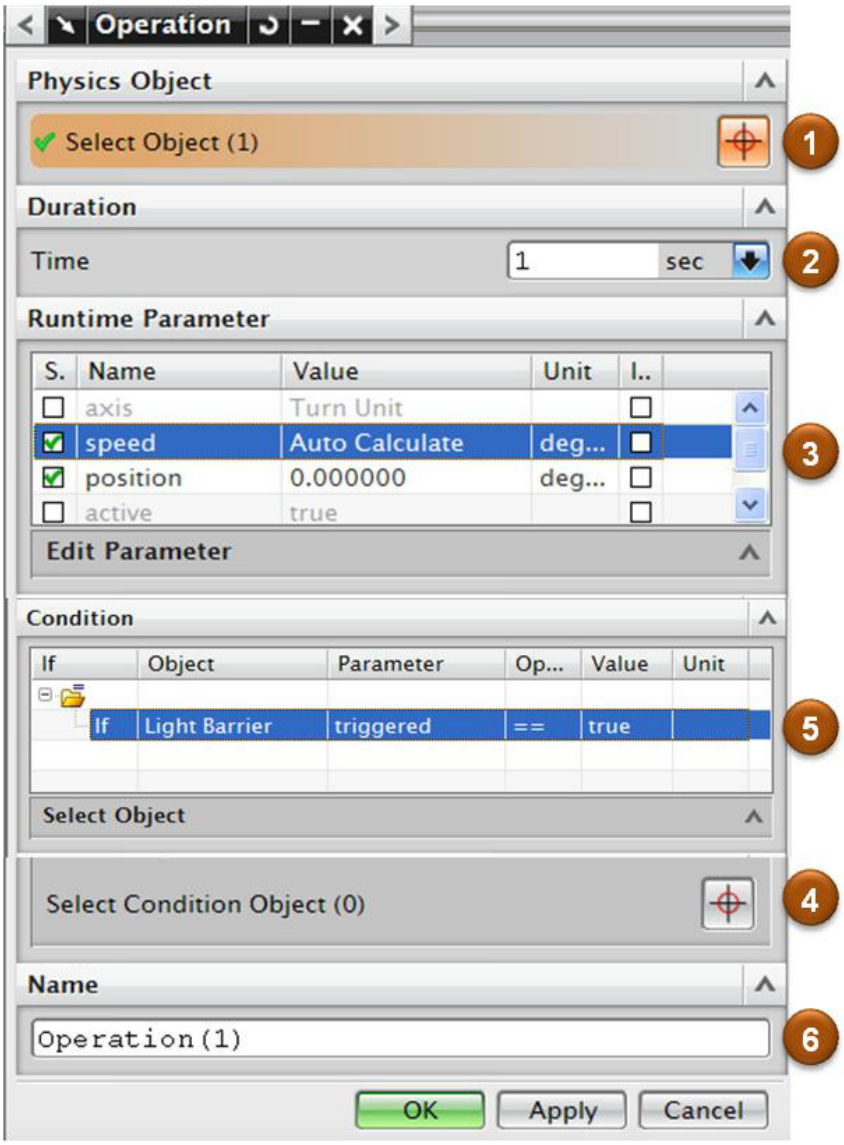
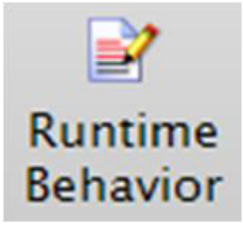


Figure 90: Basic parameters of an Operation with condition

	Parameter	Description
1	Object	Select an actuator that will be controlled by this operation. It is also possible to select any other object and change parameters. *For example changing the attachment of a fixed joint can attach an object to a gripper at runtime.*
2	Time	Duration of the operation. Grayed if dynamically calculated.
3	Interface	The interface section shows all accessible parameters of the object. By activating the parameter in the first column the corresponding parameter will be changed by the operation. The user has to define the value of the parameter. *The column I/O defined if this parameter will become a signal that is accessible from external applications. This is not yet implemented in Mechatronics Concept Designer.*
4	Condition Object	The Condition Object provides the runtime parameters to determine the start condition of the operation.
5	Condition	The condition is defined as an equation. If the equation is true the operation is fired.
6	Name	Name of the Operation

4.9.2 Add a behavioral object



"Runtime Behavior" is a very generic approach to link C# programming capabilities with objects of the mechatronics system.

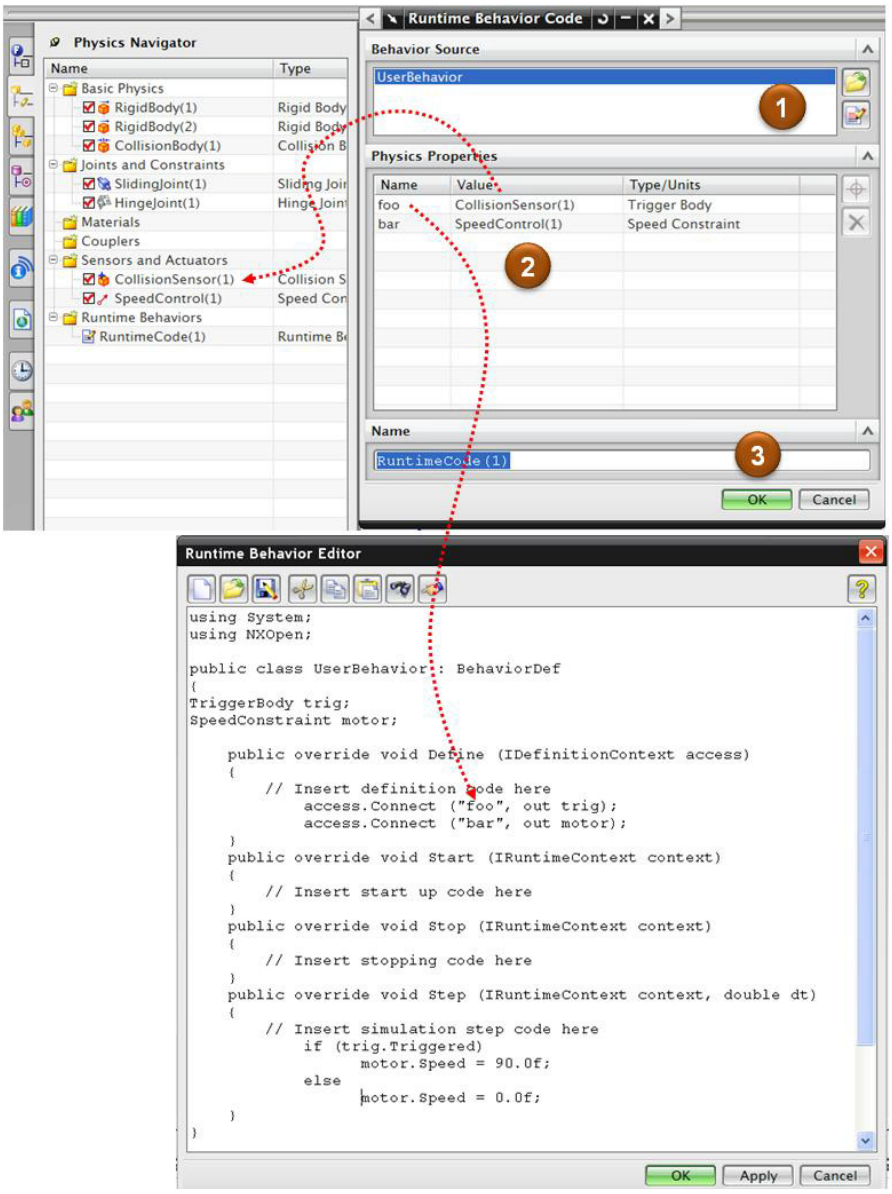


Figure 91: Basic parameters of an Operation with condition

	Parameter	Description
1	Behavior Source	C# code of the behavior. This can be loaded from a file ( ) and edited in a built-in editor ( ).
2	Physics Properties	Link objects in the source code with objects of the mechatronics model. The following command is used to create the object in C#. access.Connect ("*name*", out *obj*);
3	Name	Name of the Behavior Object

```
using System;
using NXOpen;

public class UserBehavior : BehaviorDef
{
TriggerBody trig;
SpeedConstraint motor;

   public override void Define (IDefinitionContext access)
   {
      // Insert definition code here
           access.Connect ("foo", out trig);
           access.Connect ("bar", out motor);
   }
   public override void Start (IRuntimeContext context)
   {
      // Insert start up code here
   }
   public override void Stop (IRuntimeContext context)
   {
      // Insert stopping code here
   }
   public override void Step (IRuntimeContext context, double dt)
   {
      // Insert simulation step code here
           if (trig.Triggered)
             motor.Speed = 90.0f;
           else
             motor.Speed = 0.0f;
   }
}
```

Figure 92: Example code

The C# code has to implement the following methods:

Method	Description
public override void Define (IDefinitionContext access)	Basic definition of all object links
public override void Start (IRuntimeContext context)	Executed once when the simulation is started.
public override void Stop (IRuntimeContext context)	Executed once when the simulation is stopped.
public override void Step (IRuntimeContext context, double dt)	Executed at each step of the simulatoin

The detailed description of the C# classes is provided in a separate document that is not yet available.

4.10 Knowledge capture and efficient design based on reusable objects

To support reuse of mechatronics knowledge Mechatronics Concept Designer encapsulates multi disciplinary data in one part file. To maximize design efficiency these predefined units have to be maintained in a reuse library. They can easily be dragged into a mechatronics systems design and will provide multi-disciplinary data to the mechatronics overall system.

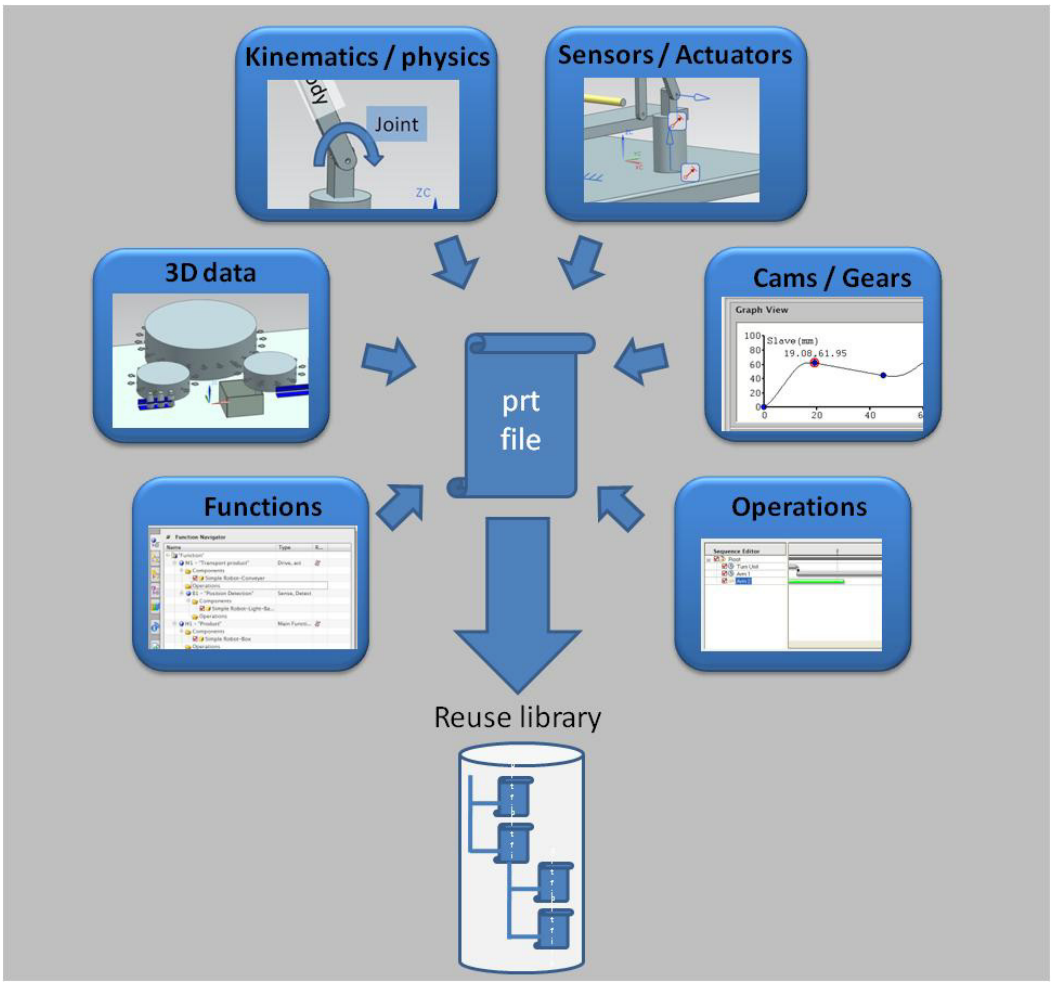


Figure 93: Reusable functional units

The mechatronics data become visible in the Physics Navigator when you right click on the corresponding part file in the Assembly Navigator and “Replace Reference Set – Entire Part”.

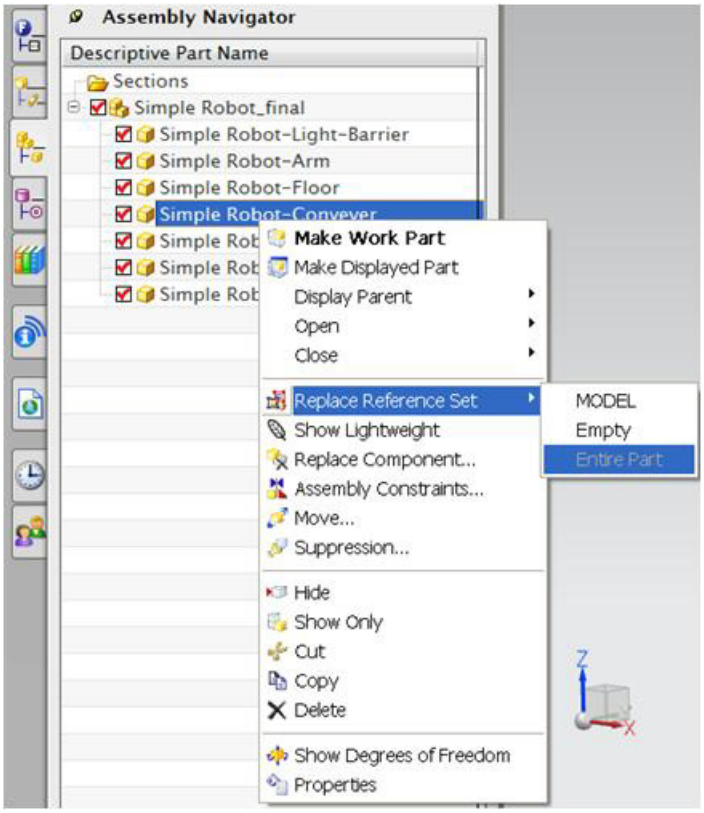


Figure 94: Replace Reference Set – Entire Part

Operations of a sub parts are displayed in the Sequence Editor as a "Compound Operation". This operation is representing the sub operations of the corresponding "mechatronics" part.

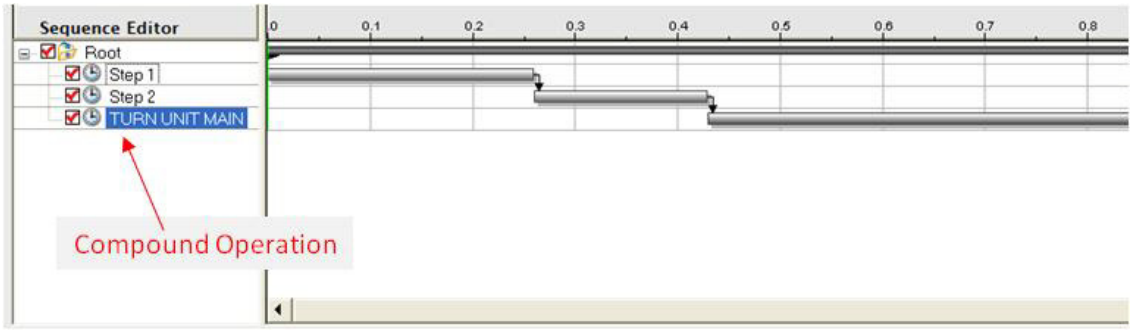


Figure 95: Compound Operation of the part "TURN UNIT MAIN"

In order to edit these operations the corresponding part has to become the Work Part.

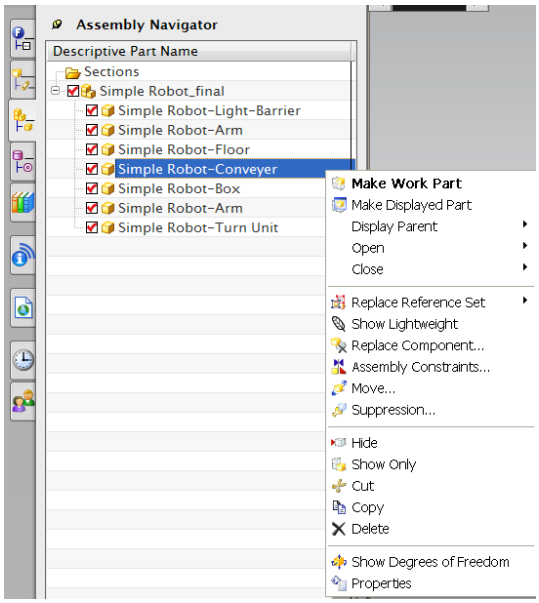


Figure 96: Make a part a "Work Part"

4.11 Monitor the simulation

To monitor runtime parameters the inspector will help to analyze the system (see chapter 2.3.2 on page 34).

4.12 Preferences

4.12.1 General

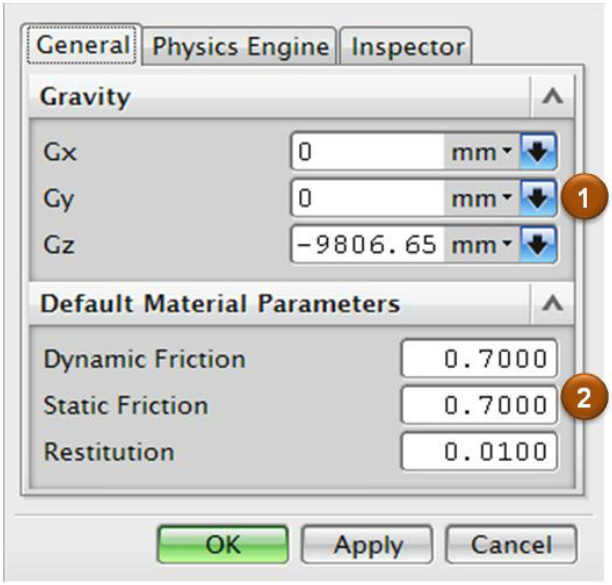


Figure 97: General preferences

	Parameter	Description
1	Gravity	Value of the gravity in x, y and z direction.
2	Default Material Parameters	Default value for the material if no specific parameters are defined.

4.12.2 Physics Engine

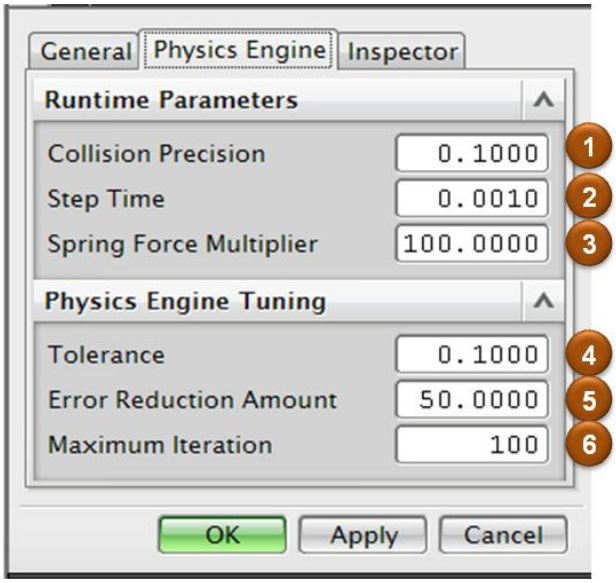


Figure 98: Physics Engine preferences

	Parameter	Description
1	Collision Precision	Determines the granularity of the collision detection. Two objects can be considered to be touching if their surfaces come this close. The objects may also penetrate one another by this amount. A larger value will be more efficient but cause more penetration.
2	Step Time	Determines the smallest time increment. Physics calculations are performed once for every time step. Actions do not occur between time steps. A larger value will improve system performance but will reduce accuracy.
3	Spring Force Multiplier	Applies to the rigidity of joints. A larger value makes joints more rigid (less flexible) but a value too high will cause instability.
4	Tolerance	The amount of distance that objects can have between them and still be considered to be aligned by a joint. A larger value will be solved more quickly but joints will have more play in their positioning.
5	Error Reduction Amount	A factor applied to determine how quickly joint positions are solved. A larger value will cause the solver to pull joints together in fewer steps, but a value that is too high will be unstable.
6	Maximum Iteration	The maximum number of iterations (within a single time step) that the solver will use to solve for the positions of the joints and bring all their positions to be within the tolerance. A larger value will give the

	solver more time to bring a large number of joints into position but it will potentially take longer to solve.

4.12.3 Inspector

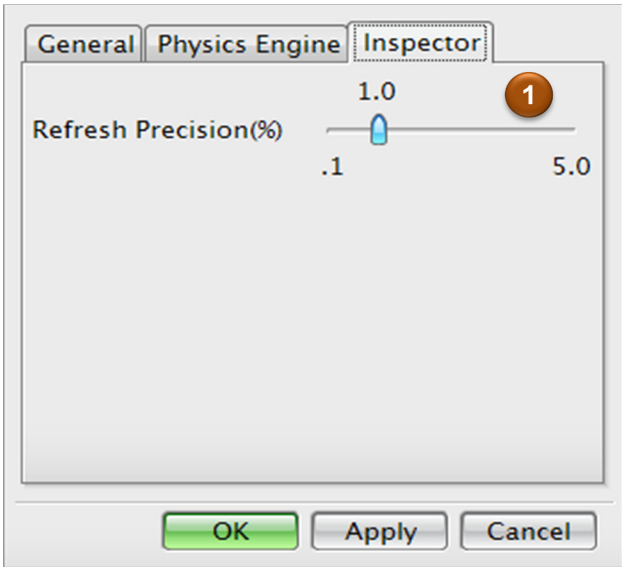


Figure 99: Inspector preferences

	Parameter	Description
1	Refresh Precision	Refresh rate of the values in the inspector window.

4.13 Instable Simulation

The simulation might show unexpected behavior. Bodies are moving without any “obvious” reason, items are vibrating or joints are showing strange deformations.

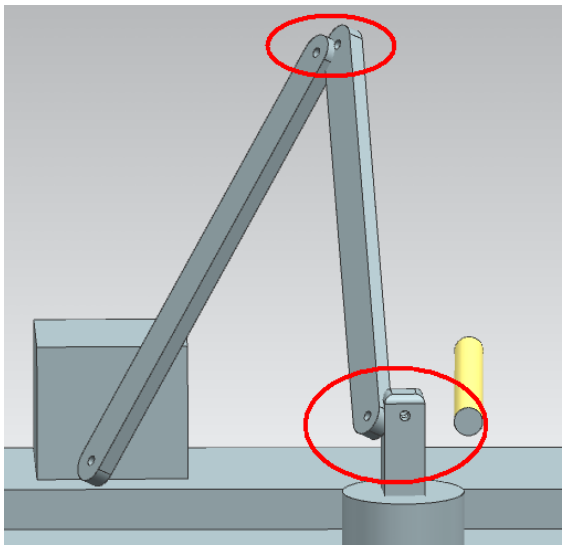


Figure 100: Example for instable simulation

Here is a short list of reasons for this kind of behavior:

- The system is over constrained. Constraints are working against each other and the weaker constraint breaks. E.g. in the picture above you see that the robot is colliding with another object and the collision constraint is working against the joints.
  -> Define all constraints carefully and avoid constraints that are not necessary for your design.
- The simulation is going slow when you have used too many constraints, e.g. very complex kinematic chains with many cams or many objects with complex collision bodies especially when you use the object source (see chapter 4.4.1.5).
  -> Define constraints or collision bodies only if you really need them.
  -> Define only collision bodies that are necessary. Not every body in the scene needs to be a collision body.
  The collision does not work properly if the collision body is ultra thin.
  -> Define collision bodies that have a volume and avoid collision bodies that are just a face.
- Collision bodies interpenetrate at the starting point of the simulation. In this case the simulation will apply a force to these bodies to bring them into a non-overlapping state. This force will lead to an unexpected movement.
  -> Make sure that you position collision bodies properly. Keep in mind that the collision body does not have exactly the same shape like the corresponding 3D body.
  -> Use collision categories (see chapter 4.4.1.2 on page 48) to avoid unwanted collisions.
- The model is very small e.g. the dimensions of active bodies are less than 5mm. So physical properties like inertia, mass, torque and force are very small and the system is very easy to be unstable.
  -> Avoid to create systems with too small absolute dimensions.
- The attachment direction of an angular spring joint is not perpendicular with the hinge direction. So there is a torque between the normal of two angular spring directions and hinge direction.
  -> Properly define the vector of mechanical forces that are applied to the system.
- The anchor point of a joint is far away from the mass center. This might cause instable behavior. The anchor point should be near the mass center.
  -> Select anchor points as close as possible to the center of the mass of corresponding bodies.
- Rigid bodies that are linked with other bodies are leaving the visible scene, e.g. they are falling down due to the gravity. In this case they fall into “infinity” but still impact the current system.
  -> Don’t forget that bodies that have left the scene might still impact the system.
- A chain of multiple cams might amplify the movement of the first element in the chain. If the first element is just a little instable the last element in the chain will reflect this instability in a magnified way.
  -> Carefully think about the application of cams and define them properly.
- The parameters of the physics engine (see chapter 4.12.2 on page 92) are not set properly.
  -> Adjust the setting to improve the behavior of the simulation.

5 Support other design disciplines

As explained in chapter 1.2.3 on page 8 Mechatronics Concept Designer is part of the overall design process and offers various interfaces to exchange data with other applications.

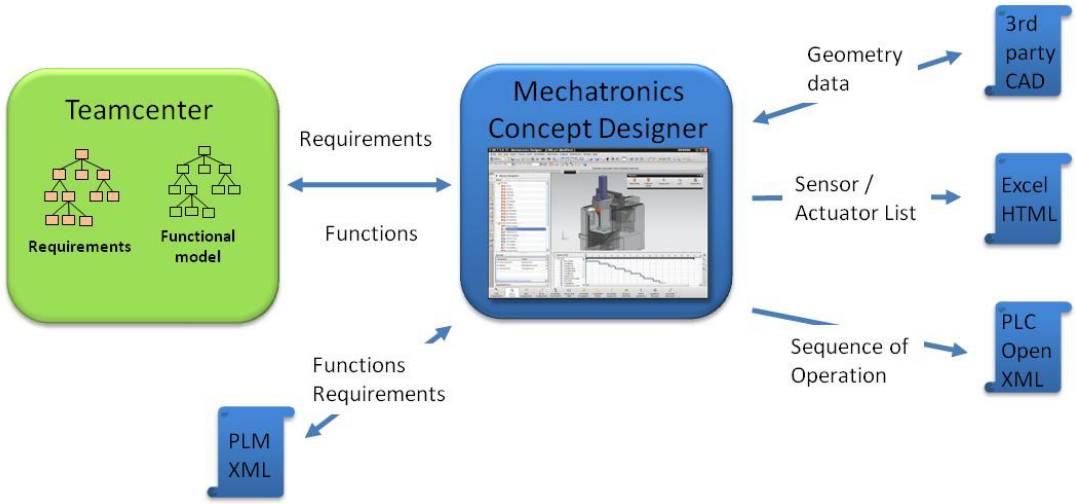


Figure 101: Interfaces of Mechatronics Concept Designer

5.1 Systems Engineering and Functional Modeling

Mechatronics Concept Designer supports the following data exchange:

- Load functions and requirements from Teamcenter
- Synchronize data with Teamcenter (refresh)

Use Case	Teamcenter to MCD	MCD to Teamcenter
Add a new function	x	x
Change a function	x	create a new revision in Teamcenter
Delete a function	x	not possible
Add a new requirement	x	x
Change a requirement	x	create a new revision in Teamcenter
Delete a requirement	x	not possible

- Export and Import Functions and Requirements in PLMXML.

Further information can be found in chapter 3.1 on page 36.

5.2 Mechanical Design

Mechatronics Concept Designer supports data exchange of CAD data in all major CAD formats.

5.3 Electrical Design

Mechatronics Concept Designer exports the Sensor and Actuator list.

See chapter 4.5.3 on page 72.

5.4 Software Design

Mechatronics Concept Designer exports operations to the PLCOpen XML format which has been standardized by the AutomationML group.

See chapter 3.3 on page 41.

Free viewers are available:

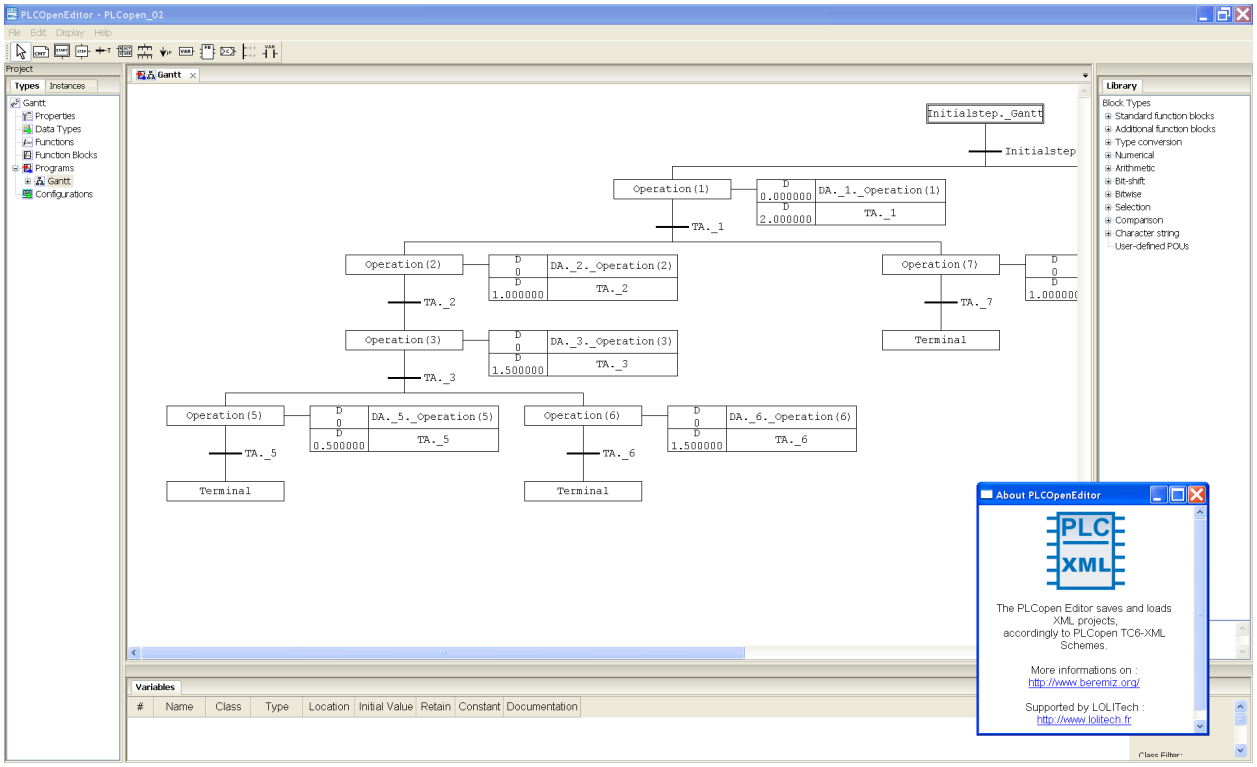


Figure 102: PLCopen Editor from Beremiz

6 Addendum

6.1 EN 61346-2

IEC 61346 is an electro technical standard entitled "Industrial systems, Installations and Equipment and Industrial Products — Structuring Principles and Reference Designations". It sets voluntary standards on how to structure systems and generate reference designations.

Code	Object class definition	Examples for terms describing the purpose or task of objects	Examples for typical mechanical/fluid objects	Examples for typical electrical objects
A	Object serving two or more purposes Note - This class is only for objects for which no main function can be identified.			
B	Object for monitoring and detecting or determining events or conditions.	• detecting • measuring *) • monitoring • sensing • weighing *) *) picking-up of values	• orifice plate (for measuring) • sensor	• detector • fire detector • limit switch • measuring element • measuring shunt • microphone • movement detector • photo cell • pilot switch • position switch • proximity switch • proximity sensor • sensor • smoke sensor • tachogenerator • temperature sensor • video camera

C	Object for storing material, energy, or information	• recording • storing	• barrel • cistern • container • hot water accumulator • steam accumulator • tank • vessel	• buffer (store) • capacitor • event recorder *) • memory • tape recorder *) • video recorder *) • voltage recorder *) *) mainly storing function
D	Reserved for futurestandardization			
E	Object for generating heat,cold, and/or light or other radiation.	• cooling • heating • lighting • radiating • transmitting	• boiler • furnace • heater • gas lamp • heat exchanger • nuclear reactor • paraffin lamp • radiator	• antenna • boiler • fluorescent lamp • freezer • heater • lamp *) • lamp bulb • laser • luminaire • maser • radiator • refrigerator *) see also H
F	Object for protecting directly or indirectly a flow, personnel or equipment from dangerous or unwanted conditions.	• absorbing • insulating • guarding • preventing • protecting • securing • shielding	• air bag • buffer • fence • pipe-break valve • rupture disk • safety belt • safety valve • shield • vacuum valve	• Buchholtz relay • cathodic protection anode • earthing electrode • Faraday cage • fuse • insulator • protective relay • miniature circuit breaker • cable screen • shield • surge diverter • thermal overload relay • thermal overload release

G	Object for generating a flow of material, energy, or signals.	• generating • producing • pumping • transporting	• pump • blower • conveyor (driven) • fan • lift • manipulator • vacuum pump • ventilator	• dry cell battery • coil *) • dynamo • generator • hand inductor • power generator • rotating generator • signal generator • solenoid *) • wave generator *) see also M
H	Object for presenting signals or information *) in optical, audible or tactile (touchable) form.*) e.g. positions, levels, conditions, alarms, announcements Note - Objects for presenting measured values: see class P	• alarming • communicating • displaying • indicating • informing • presenting • printing • warning	• acoustical signal device • clock • bell • display unit • indicating device • printer • sight glass	• acoustical signal device • clock • bell • display unit • indicating device • LED • loudspeaker • optical signal device • printer • signal lamp • signal vibrator
J	Reserved for future standardization			

K	Object for processing (receiving, treating and providing) signals for discrete and/or continuous control of other objects.	• closing *) • continuous controlling • delaying • opening *) • positioning • postponing • switching • synchronizing *) of information circuits	• fluid feedback controller • pilot valve • valve positioner	• all-or-nothing relay • analogue integrated circuit • automatic paralleling device • binary integrated circuit • contactor relay • delay element • delay line • electronic valve • electronic tube • feedback controller • measuring relay • programmable controller • synchronizing device • time relay • transistor
L	Reserved for future standardization			
M	Object supplying kinetic energy for actuating/driving other objects.	• actuating • driving	• combustion engine • fluid actuator • fluid cylinder • fluid motor • heat engine • mechanical actuator • spring-loaded actuator • turbine • water turbine • wind turbine	• actuator • actuating coil • electric motor • linear motor
N	Object for producing or reshaping material	• crushing • cutting • forging • forming • fractionating • grinding • milling • pulverising • rolling • turning	• crusher • drop forge • mill • turning lathe	

P	Object for presenting measured, metered, integrated, or calculated values.	• counting • measuring *) • metering *) indicating function	• balance • clock • flow meter • gas meter • indicator (mechanical) • manometer • glass gauge • thermometer • watermeter	• clock • continuous line recorder • event counter • Geiger counter • recording voltmeter • synchronoscope • voltmeter • wattmeter • watt-hour meter
Q	Controlled switching and variation of energy, signal or material flow (regarding signals in control loops see K and S)			
R	Limiting and stabilization of motion or flow of energy, information or material			
S	Object providing an interface for manual input of information into or selecting information from a system.	• influencing • manually controlling • operating • selecting	• push button valve • selector switch	• keyboard • light pen • mouse • push button switch • selector switch • set-point adjuster
T	Object for converting one form of energy or information into another form of the same kind of energy respectively information.	• amplifying • compressing • converting • expanding • modulating • transforming	• fluid amplifier • gear box • measuring transducer • measuring transmitter • pressure intensifier • torque converter	• AC/DC converter • amplifier • demodulator • frequency changer • measuring transducer • measuring transformer • measuring transmitter • modulator • power transformer • rectifier • rectifier station • signal converter • signal transformer • transducer

U	Object for keeping other objects in a defined position.	• bearing • carrying • holding • supporting	• beam • bearing • block • cable ladder • cable tray • console • corbel • foundation • hanger • insulator • mounting plate • mounting rack • pylon • roller bearing • roll stand	
V	Object for separating, combining, or mixing material, energy, or information.	• filtering • mixing • separating • stirring	• centrifuge • filter • mixer • rake • separator • sieve	• filter • induction stirrer
W	Object conducting, guiding or leading material, energy or information from one place to another.	• conducting • guiding • leading	• duct • hose • ladder • mirror • link (mechanical) • roller table (not driven) • pipe • shaft	• busbar • cable • conductor • information bus • optical fibre • through bushingwaveguide
X	Object for establishing a static connection	• connecting *) • coupling *) • joining *) static	• flange • hook • hose fitting • pipeline fitting • quick-release coupling • terminal block	• connector • plug connector • terminal • terminal block • terminal strip
Y	Reserved for future standardization			
Z	Reserved for future standardization			

7 Contact

If you have any questions please contact:
Dr. **Matthias Lenord**
I IA PL OAT
Address:
10824 Hope Street
Cypress, CA 90630
Phone:
+1.714.952.5380
eMail:
matthias.lenord@siemens.com

About Siemens PLM Software

Siemens PLM Software, a business unit of Siemens Industry IA (Industry Automation), is a leading global provider of product lifecycle management (PLM) software and services with 4.6 million licensed seats and 51,000 customers worldwide. Headquartered in Plano, Texas, Siemens PLM Software's open enterprise solutions enable a world where organizations and their partners collaborate through Global Innovation Networks to deliver world-class products and services.

For more information on Siemens PLM Software products and services, visit http://www.siemens.com/plm.