

TKC: A Framework for ADAMS/View Customization with diverse application examples

Chris Verheul, SayField International

Taken from the MS Software SimAcademy Presentation of February 19, 2015

Agenda

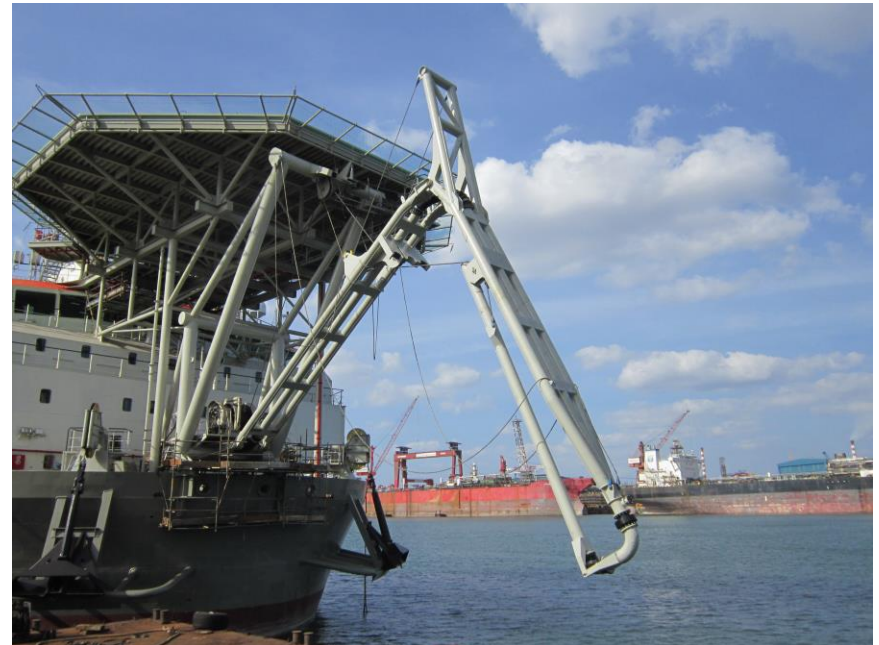
- Introduction
- **Typical examples of using ADAMS/View in Benelux**
 - Mechatronics: Ball Balancing Robot
 - Offshore: Dredging Ship Bowsprit Dynamics
- **Building Component Based ADAMS/View Robot Model**
 - Components → Toolkit → Model Setup → Assembly Model
 - Configure Toolkit and register Objects
 - Define Parametric Components
 - Toolkit File Structure
 - Create Model Instance using Components
 - Concluding Remarks

Mechatronics: Ball Balancing Robot Simulations (icw Alten Mechatronics)

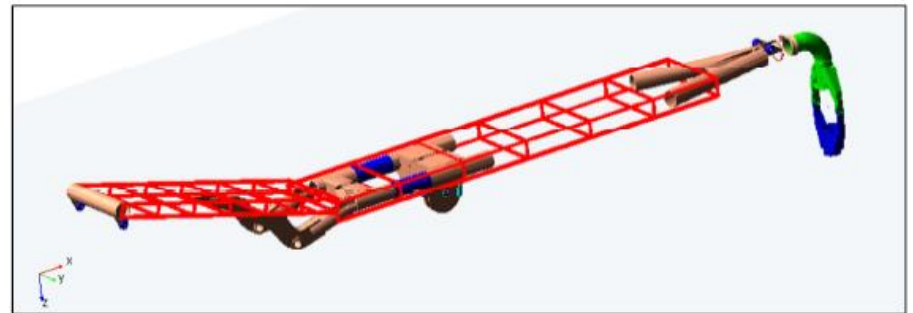
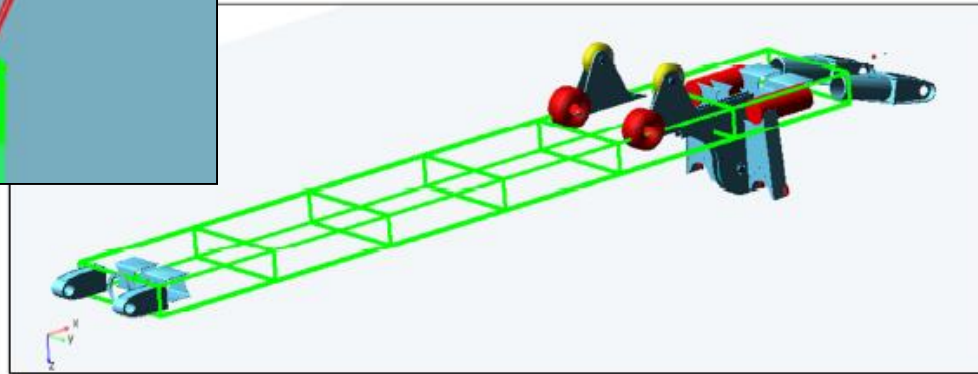
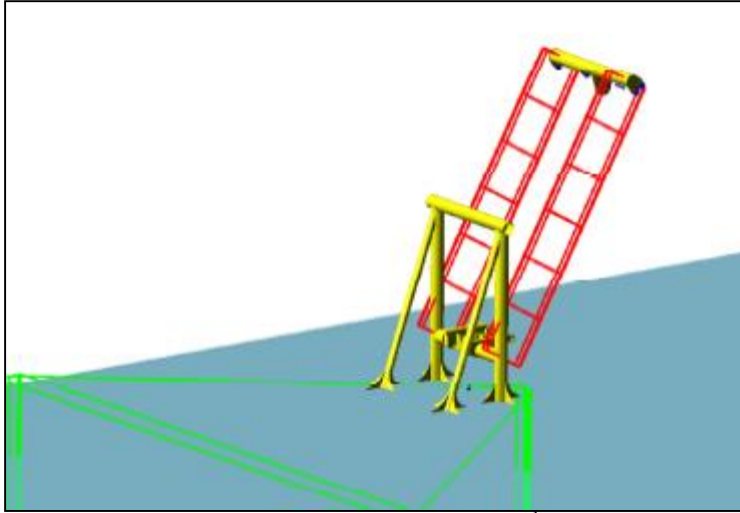
- Ball Balancing Robot Model:
 - Multibody Simulation Test
 - Testing of Control Algorithms
 - Analyze behaviour with Local Cameras
 - Predict effect of modifications:
 - Omniwheels contact to Ball
 - Additional Casters to maintain stability at extreme angles



Offshore: Deployment of a Dredging Bowsprit (icw: Boskalis N.V.)



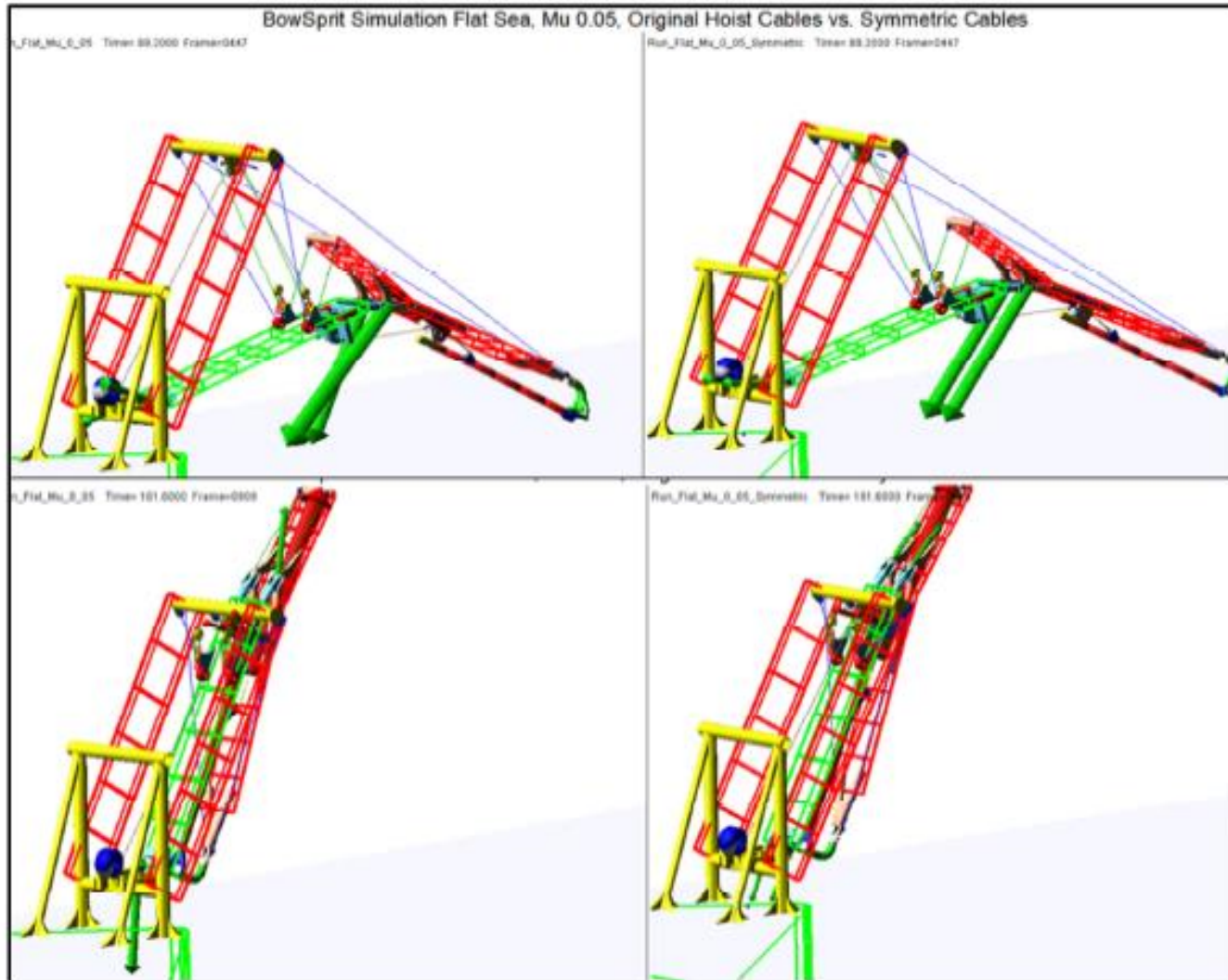
Simulation Approach: Independent Model Components



Main Frame Component Parameters

Bowsprit Part3				
Name	Type	Value	Units	Info
Visuals		Group	Non-Functional Visual Data	
Col_EndPart	Color	.colors.BLUE		Color of End Part
Col_Strut	Color	.colors.RED		Color of Strut Parts
Col_Force	Color	.colors.YELLOW		Color of Connection Forces
Col_Joint	Color	.colors.CYAN		Color of Constraints and Motions
Inertia_Dynamics		Group	Parts Inertia and other Dynamics Data	
M_Beg_Part	Real	(100.0(kg))	Mass	Begin Part Mass
I_Beg_Part	Real	((5.0, 50.0, 50.0)(kg-m**2))	Inertia	Begin Part Inertia (X-Along, Y-Vertical)
X_Cm_Beg	Real	(0.5m)	Length	X-Loc. of Begin Part Cm wrt. Begin Reference
M_End_Part	Real	(10.0(kg))	Mass	End Part Mass
I_End_Part	Real	((1.0, 1.0, 1.0)(kg-m**2))	Inertia	End Part Inertia (X-Along, Y-Vertical)
X_Cm_End	Real	(-0.5m)	Length	X-Loc. of End Part Cm wrt. End Reference
X_Beg_Strut	Real	(0.8m)	Length	X-Loc. of Strut Begin wrt. Begin Reference
Rad_Pivot	Real	(3.5E-002m)	Length	Pivot Joint Friction Torque Radius
Mu_Pivot	Real	5.00E-02	No_Units	Friction Mu on Begin Side Pivot
Os_Pivot	Real	(1.0(deg/sec))	Angular_Velocity	Full developed Slip Omega for Pivot Friction
Parts_Geometry		Group	SpritPart End Parts and Geometry Data	
Y_Beg_Pivot	Real	(0.44m)	Length	Y-Loc. of Beg Pivots to Carrier wrt. Reference
Z_Beg_Pivot	Real	(0.0m)	Length	Z-Loc. of Beg Pivots to Carrier wrt. Reference
X_End_Part	Real	(10.925m)	Length	X-Loc. of End Part wrt. Reference Begin Part
Discrete_Dynamics		Group	Discrete Flexibility Strut Parameters	
Num_Segments	Integer	6		Nr. of Segments in the Discrete Strut
Lock_Ends	List(No,Yes)	No		Lock Fixation of the Strut End Parts
Density	Real	(7801.0(kg/meter**3))	Density	Part Strut Material Density
Youngs_Mod	Real	(2.07E+010(N/m**2))	No_Units	Part Strut Material Youngs Modulus
Poissons_Rat	Real	0.29	No_Units	Part Strut Material Poisson Ratio
Damping	Real	(5.0E-003(sec))	Time	Damping Ratio in the Part Strut (secs)
Rad_Strut	Real	(0.175m)	Length	Part Strut Radius
Thk_Strut	Real	(1.2E-002m)	Length	Part Strut Wall Thickness
Parts_CAD_Data		Group	End Parts CAD Graphics Files	
Solids_Lib	List(UseF, Safe, BBC)	UseF		PARTS CAD Solids Library
CAD_SpritPart	File	__Sprit_Part3		Begin Part Solid File
CAD_Beg_Part	File	Sprit_Part3_Begin		Begin Part Solid File
CAD_End_Part	File	Sprit_Part3_End		End Part Solid File
Hoist_Init_Setup		Group	SpritPart Initial and Hoist Cable Data	
Loc_Pear	Real	((-3.03, 0.35, 0.3)m)	Length	Loc. of Hoist Pear wrt. Ref. End Part
Ori_Pear	Real	(0.0 * 90.0d + 1 * 135d)	Angle	Ori. of Hoist Pear wrt. Ref. End Part
Damp_Pear	Real	(0.1(N-m-sec/deg))	Torsion_Damping	Revolute Pear Rotational Damping
L_Sleeve	Real	(1.7m)	Length	Hoist Cable Length to Bumper
Ang_Init	Real	(0.0deg)	Angle	Initial Angle of SpritPart wrt. Reference

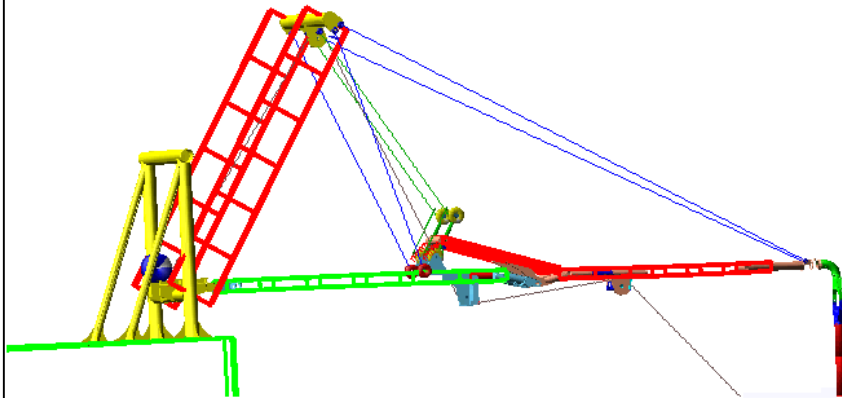
Bowsprit ADAMS/View Model Assembly



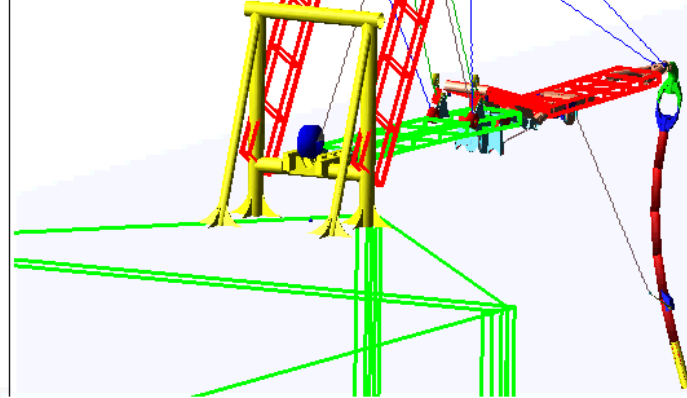
Results of Modal Dynamics Analysis

Bowsprit Oscillatory Modes 328, ..., 331

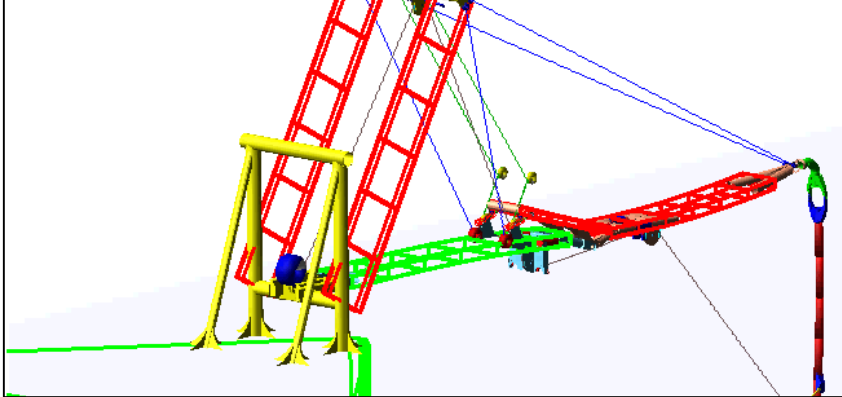
EIG_1 Mode=328 Frequency= 1.9618 (Hz)



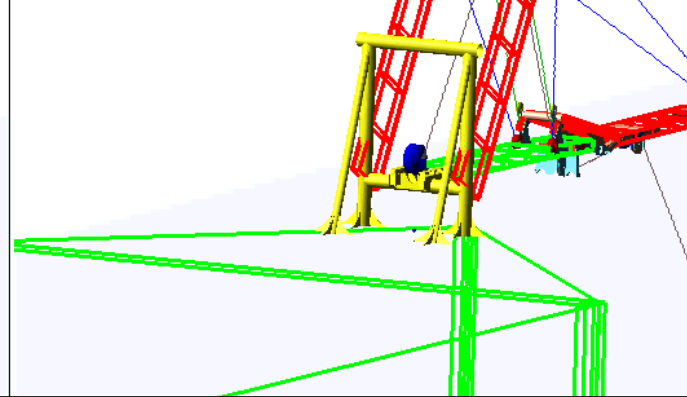
EIG_1 Mode=329 Frequency= 2.4140 (Hz)



EIG_1 Mode=330 Frequency= 2.4879 (Hz)



EIG_1 Mode=331 Frequency= 3.0189 (Hz)



Building Component Toolkits to make Models

Which **functionally independent components** ?

- Setting up a Components Toolkit Framework
 - GUI Dialog & Input Parameters for each Component
 - → Testing Component GUI
- Functional Works: Creating Component Topology and Data
 - → Testing Component Functionality
- Manually re-assemble the Components to create a Robot
 - Testing Component Interactions
 - → Testing Model Dynamics
- Extracting the Model to a Model Assembly
 - Testing the Assembly
 - (Improving Assembly by adding Parameters)

Robot Arm Model: System overview

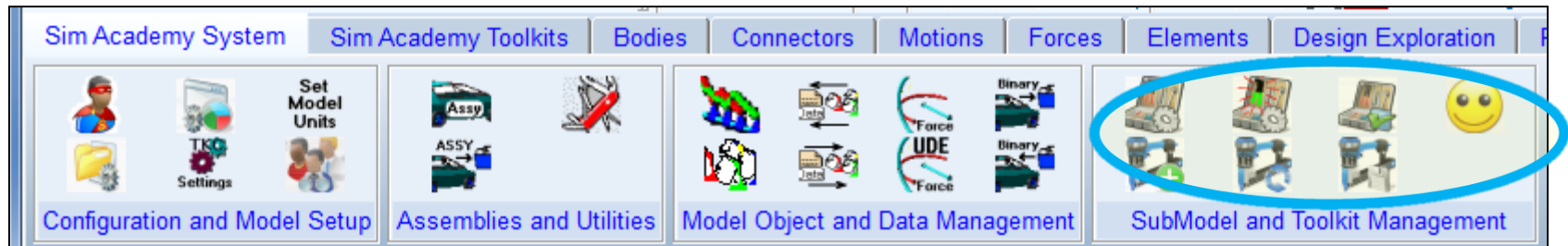


Proposed Robot Model Toolkit Components

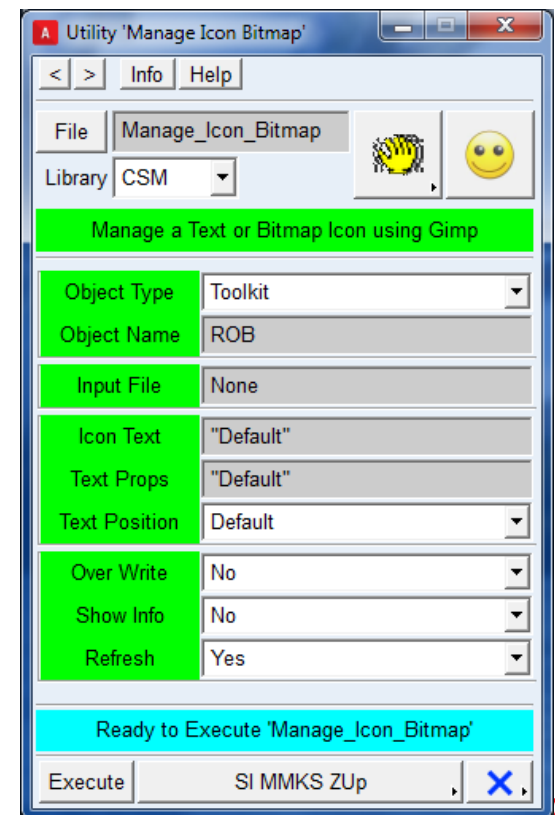
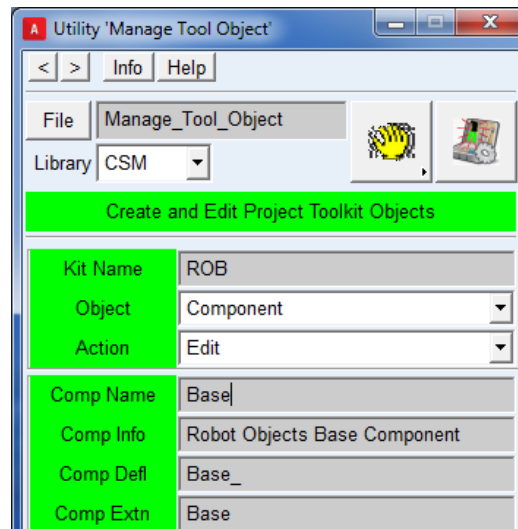
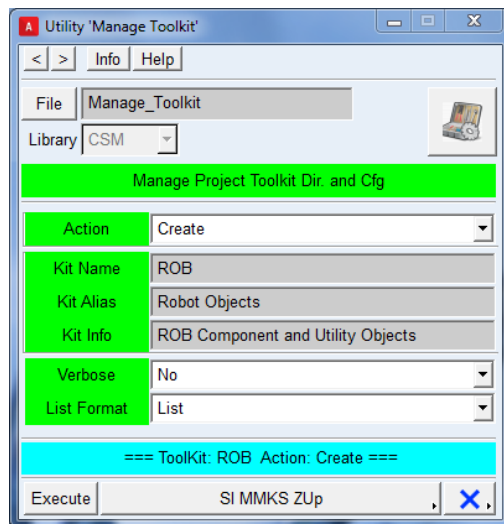
Toolkit **ROB**: *Robot Objects and Building Blocks*

- **Base Component:**
 - Connected to environment on a *Carrier* Body
 - At location of a *Reference* Marker
 - Different versions are available: Generic, Basic, ...
- **Arm Component:**
 - Connected to environment on a *Carrier* Body (Base, Arm, ...)
 - At location of a *Reference* Marker
 - Different versions available: Generic, Single, Double, ...
- **Gripper Component:**
 - Etc etc etc...
- **Load Component:**
 - Located at *Reference* Marker
 - Can be in contact with robot components

Step 1: Configure & Register Toolkit Objects



Create the Toolkit ROB, define the Icon and start creating Components



Configure & Register Toolkit *Objects*

Shortcuts to Create Component Parameters and Sub-Types (use '?' and '#')

```

=== The Number of Component Parameter Default Codes is: 5 ===

1  Code: Ref      Name: Reference      Type: Marker:D=(None)      Info: '[N] Marker for Component [C]'
2  Code: Car      Name: Carrier        Type: Part:D=Ground        Info: '[N] Part of Component [C]'
3  Code: DCn      Name: Data_Contact   Type: All(GUM.Contact.Data):D=(None) Info: '[N] for Component [C]'
4  Code: COb      Name: Comp_Object    Type: All(C_UDE):D=(None)   Info: 'Input [N] to Component [C]'
5  Code: DCo      Name: Comp_Data      Type: All(D_UDE):D=(None)   Info: 'Input [N] Object to Component [C]'

=== The Number of Component SubType Default Codes is: 5 ===

1  Code: AGen     File_Extn: Comp_Generic   Info: 'Generic Version of [C]'
2  Code: CGen     File_Extn: [C]_Generic     Info: 'Generic Version of [C]'
3  Code: Basic    File_Extn: [C]_Basic      Info: 'Basic [C]'
4  Code: Rigid    File_Extn: [C]_Rigid      Info: 'Rigid Version of [C]'
5  Code: Flex     File_Extn: [C]_Flex       Info: 'Flexible Version of [C]'
    
```

Comp Name	Base
Comp Info	Robot Objects Base Component
Comp Defl	Base_
Comp Extn	Base_*
Parm Name	Carrier
Parm Type	Part:D=Ground
Parm Info	Carrier Part of Component Base
Parm Call	Create
Subtype Name	Generic
Subtype Info	Generic Robot Objects Base Component
Subtype Extn	Base_Generic

Kit: ROB Action: Edit

Execute SI MMKS ZUp X

Comp Name	Gripper.Ref.Car..Generic..Basic
Comp Info	Robot Objects Gripper Component
Comp Defl	Gripper_
Comp Extn	Gripper

Kit: ROB Action: Edit

Execute SI MMKS ZUp X

Shortcuts applied
in Comp. creation

Toolkit ROB Configuration Ready !!



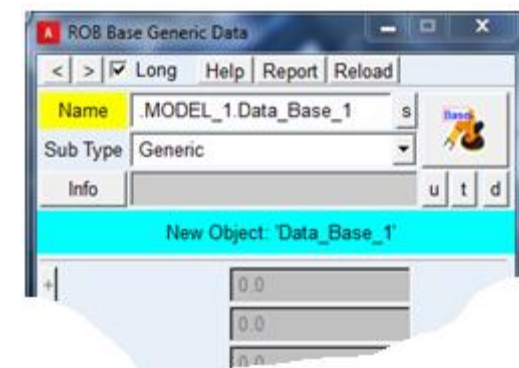
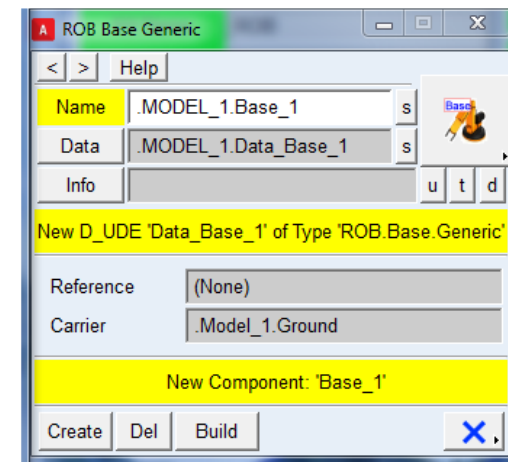
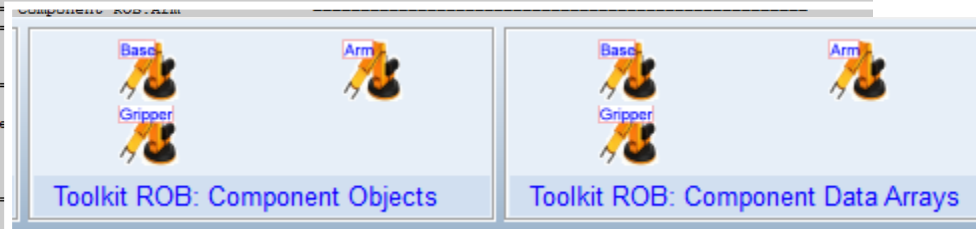
```
===== List of Toolkit Components =====
= Nr.== Name == Default_Name == Params == STypes ===== Comment =====
1 Base Base_ 2 2 'Robot Objects Base Component'
2 Arm Arm_ 2 3 'Robot Objects Arm Component'
3 Gripper Gripper_ 2 2 'Robot Objects Gripper Component'
```

```
===== Component ROB.Base =====
===== Defines 2 Input Parameters =====
= Nr.== Name == Type == Comment ==
1 Reference Marker:D=(None) 'Reference Marker for Component Base'
2 Carrier Part:D=Ground 'Carrier Part of Component Base'
```

```
===== Defines 2 SubTypes =====
= Nr.== Name == Data_Extension == Comment ==
1 ROB.Base.Generic Base_Generic 'Generic Robot Objects Base Component'
2 ROB.Base.Simple Base_Simple 'Simple Robot Objects Base Component'
```

```
===== Defines 2 =====
= Nr.== Name ==
1 Reference
2 Carrier

===== Defines 3 =====
= Nr.== Name ==
1 ROB.Arm.C
2 ROB.Arm.Single Arm_Single 'Single Robot Objects Arm Component'
3 ROB.Arm.Double Arm_Double 'Double Robot Objects Arm Component'
```



Step 2: Define Parametric Component Topology

Different Options available:

1. Manual ADAMS/View editing of Component objects plus Parameters
 - + Standard Aview work (for experienced users)
 - - Correct parametrisation requires *some caution and experience*
2. Modest parametric object creation followed by *encapsulation* in Component
 - + Quick action and results, also allows to encapsulate **existing models**
 - - May result in *messy* component structure (automated parametrics??)
3. Using scripting in component and data creation macro files
 - - Requires a *programming* spirit of the user
 - + Designated approach for **complex component topology**
4. Using the TKC *Component Builder*
 - + **Direct and interactive parametrization of Blocks of Objects**
 - - May seem a bit *overwhelming* at first (many dialog actions)

TKC Component Builder: *Generic vs. Specific*

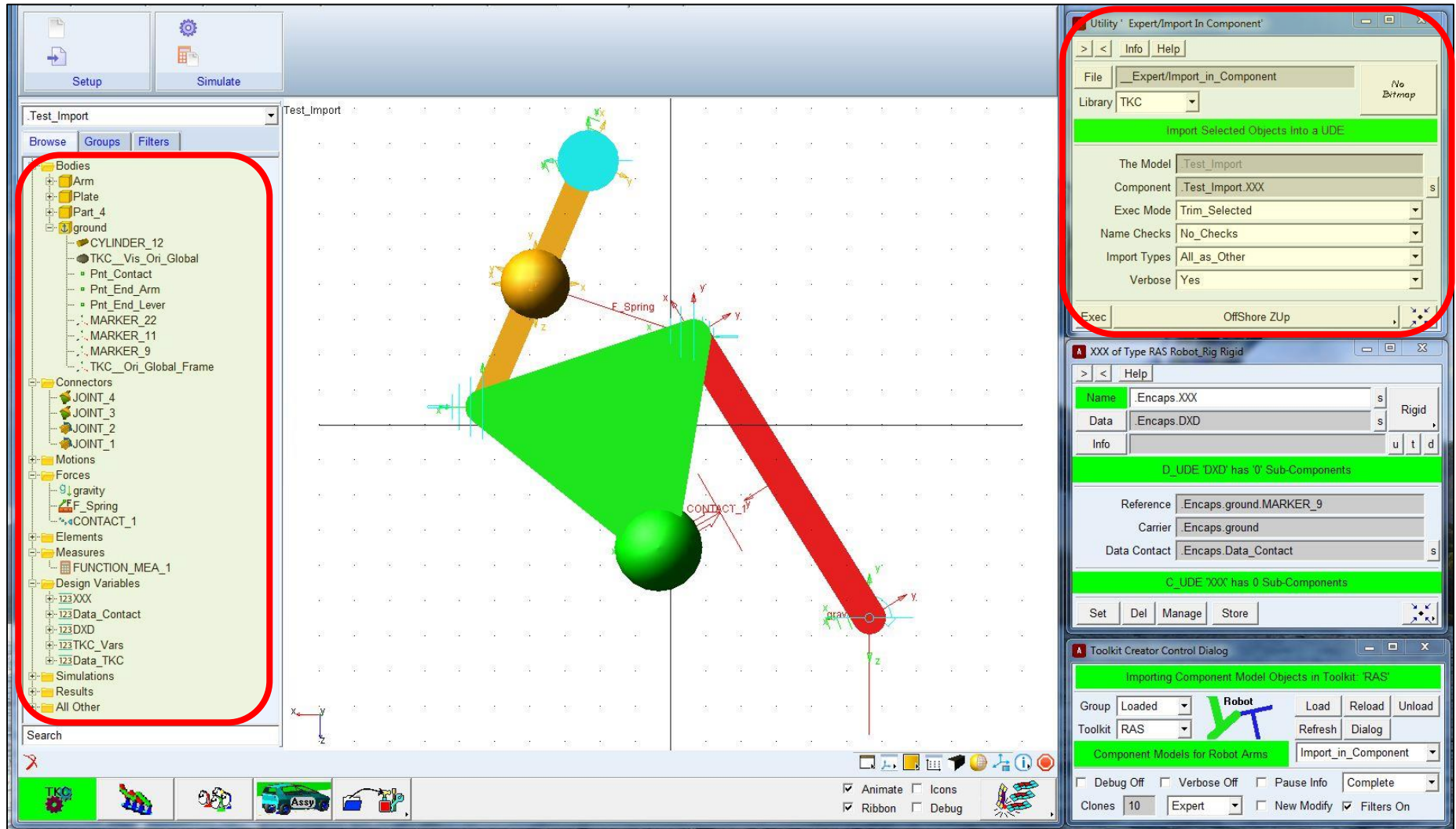
- Component Builder **adds blocks of objects** to components
- Blocks can be as simple as a marker or as complex as a component
- *Generic* components are completely defined by the component builder
- Both block parameters and topology are stored in component data array
 - + Generic components are generated *on-the-fly*
 - - Requires some caution when combining with 'specific' topology.
- Blocks must be converted to **specific** component objects (**merge all**) before component storage to their Toolkit definition macros
- Blocks can be added to a specific component (i.e. for project extensions):
 - i.e.: References of arbitrary range of cranes on a ship hull ...

Defining object hierarchy in ADAMS Models

Different approaches are available and possible.

1. Submodels: .Model_1.Model_2.Model_3
 - Is used in Template based ADAMS products
 - Can be used in Aview as well, but is still relatively recent
2. User Defined Entities (UDEs)
 - also apply the 'dot' hierarchy approach
 - How to distinguish from .Model_1.Part_1.Marker_1
3. Groups in groups in groups....
4. Apply naming convention to 'tag' a hierarchy
 - TKC uses a **double underscore** to tag objects inside a component
 - Can be used to nest to a variable level of depth
 - Extreme example: .Model_1.Robot__Base__Bearing__Outer_Race

Encapsulation in Component Start Situation



Encapsulation in Component Finished

The screenshot displays the MSC Software RAS (Robot Arm Simulator) interface. The main workspace shows a 3D model of a robot arm assembly with a green triangular plate, a yellow cylindrical arm, and a red cylindrical arm. The left sidebar contains a hierarchical tree view of the model's components, including Bodies, Connectors, Motions, Forces, Elements, Measures, Design Variables, Simulations, Results, and All Other. The right sidebar shows the 'Expert/Import In Component' dialog box, which is used for encapsulating components. The 'Import Types' dropdown is set to 'All', and the 'Exec' button is highlighted. Below this, the 'XXX of Type RAS Robot_Rig Rigid' dialog box is visible, showing the 'Encaps.XXX' component with its data and reference settings. At the bottom, the 'Toolkit Creator Control Dialog' is shown, indicating the import of component models for the robot arms.

Utility 'Expert/Import In Component'

> < Info Help

File: Expert/Import_in_Component

Library: TKC

==== Zero Remaining Objects in Select List ====

The Model: Test_Import

Component: Test_Import.XXX

Exec Mode: Execute_Import

Name Checks: Bodies_Only

Import Types: All

Verbose: Yes

Exec: OffShore ZUp

XXX of Type RAS Robot_Rig Rigid

> < Help

Name: Encaps.XXX

Data: Encaps.DXD

Info: Rigid

D_UE 'DXD' has 0 Sub-Components

Reference: Encaps.ground.MARKER_9

Carrier: Encaps.ground

Data Contact: Encaps.Data_Contact

C_UE 'XXX' has 0 Sub-Components

Set Del Manage Store

Toolkit Creator Control Dialog

Importing Component Model Objects in Toolkit: RAS

Group: Loaded

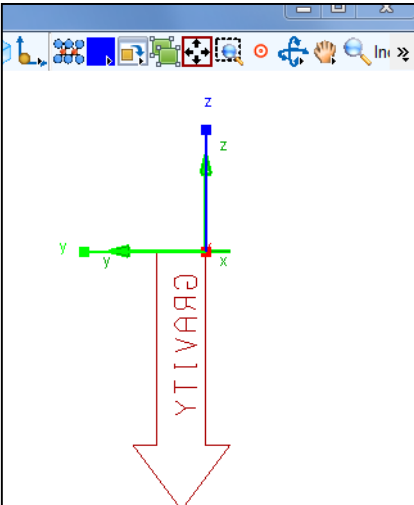
Toolkit: RAS

Component Models for Robot Arms: Import_in_Component

Debug Off Verbose Off Pause Info Complete

Clones: 10 Expert New Modify Filters On

Component Builder: Creating a Motion Driven Base Component



Component Builder: Creating a Motion Driven Base Component

The image displays the Component Builder interface for creating a motion-driven base component. The interface is divided into several panels:

- Top Left Panel:** Shows the 3D model of the component, a coordinate system (x, y, z), and a red arrow labeled "GRAVITY" pointing downwards.
- Top Right Panel (ROB Base Generic):** Contains the component's name (.MODEL_1.Base_1), data (.MODEL_1.Data_Base_1), and reference (.MODEL_1.ground.Ref_Model). It also shows the carrier (.MODEL_1.ground) and a status bar indicating "Component: 'Base_1' has 0 Sub-Components".
- Bottom Left Panel (Data_Base_1 of Type ROB.Base):** Shows the component's name (.MODEL_1.Data_Base_1), sub type (Generic), and a table of properties:

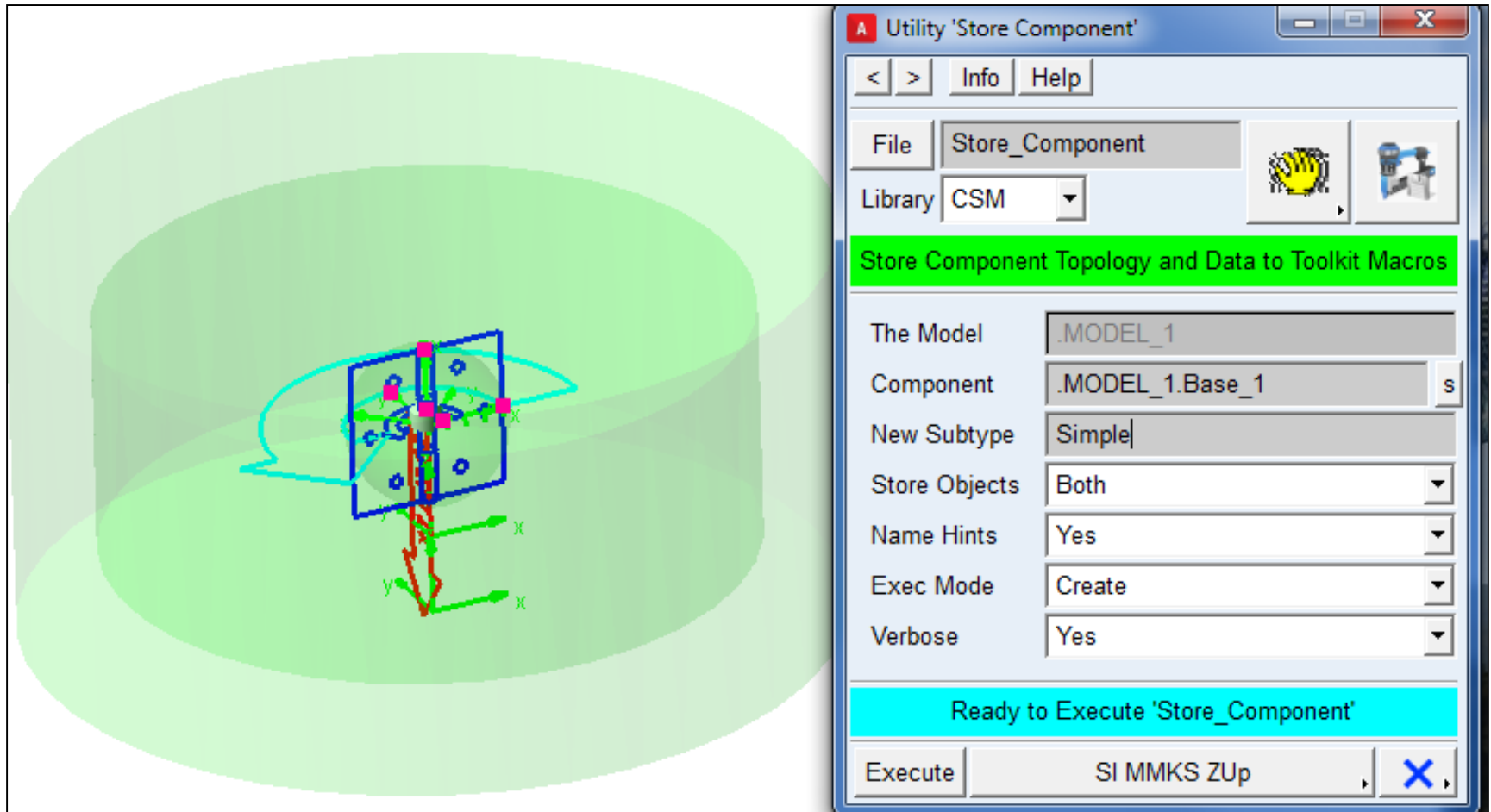
Property	Value	Subcom
Ref_Pivot	GUM.Marker	Subcom
Ref_Pivot Ori	(0.0d)	Real
	0.0	
	0.0	

- Bottom Middle Panel (Sub-Comp 'Ref_Pivot' of 'Base_1' is a 'GUM M...'):** Shows the component's type (GUM), marker (None), and name (Ref_Pivot). It also shows the reference (Marker Ref_Pivot) and a status bar indicating "1: 'Ref_Pivot' is a 'Data_Owner'".
- Bottom Right Panel (Sub-Comp 'Mot_Pivot' of 'Base_1' is a 'GUM Motion'):** Shows the component's type (GUM), motion (Steps), and name (Mot_Pivot). It also shows the reference (Motion_Steps Mot_Pivot) and a status bar indicating "5: 'Mot_Pivot' is a 'Data_Owner'".
- Right Panel (Object 'Data_Base_1' of Type 'ROB.Base'):** Shows the component's name (.MODEL_1.Data_Base_1), sub type (Generic), and a table of properties:

Property	Value
Ref_Pivot	GUM.Marker
Prt_Pivot	GUM.Part.Density
Geo_Pivot	GUM.Geometry.Cylinder
Jnt_Pivot	GUM.Constraint
Mot_Pivot	GUM.Motion.Steps

A large blue arrow points from the Component Builder interface towards the right, indicating the flow of the process.

Storing Component Topology and Parameters in Toolkit Files



ROB Toolkit File Structure

..	<DIR>
Bitmaps	<DIR>
Util	<DIR>
_ROB.cfg	5 k
Arm_Double.mac_cre	1 k
Arm_Double.mac_dat	1 k
Arm_Generic.mac_cre	1 k
Arm_Generic.mac_dat	1 k
Arm_Single.mac_cre	1 k
Arm_Single.mac_dat	1 k
Base_Generic.mac_cre	1 k
Base_Generic.mac_dat	1 k
Base_Simple.mac_cre	1 k
Base_Simple.mac_dat	1 k
Gripper_Basic.mac_cre	1 k
Gripper_Basic.mac_dat	1 k
Gripper_Generic.mac_cre	1 k
Gripper_Generic.mac_dat	1 k

```

!
Marker Create Marker = $Carrier.$'Name'__Ref_Pivot &
Loc = (LOC_RELATIVE_TO({0.0, 0.0, 0.0}, $Reference)) &
Ori = (ORI_RELATIVE_TO({$Data_Name.Ori_Pivot, 0.0d, 0.0d}, $Reference))
!
!
=====
! == Part Prt_Pivot =====
!
!
Part Create Rigid Name Part = $The_Model.$'Name'__Prt_Pivot &
Loc = (LOC_RELATIVE_TO({0.0, 0.0, 0.0}m, $Carrier.$'Name'__Ref_Pivot)) &
Ori = (ORI_RELATIVE_TO({0.0, 0.0, 0.0}d, $Carrier.$'Name'__Ref_Pivot))
!
Marker Create Marker = $The_Model.$'Name'__Prt_Pivot.Reference &
Loc = (LOC_RELATIVE_TO({0.0, 0.0, 0.0}m, $Carrier.$'Name'__Ref_Pivot)) &
Ori = (ORI_RELATIVE_TO({0.0, 0.0, 0.0}d, $Carrier.$'Name'__Ref_Pivot))
!
Marker Create Marker = $The_Model.$'Name'__Prt_Pivot.cm &
Loc = (LOC_RELATIVE_TO($Data_Name.Loc_Cm_Pivot, $The_Model.$'Name'__Prt_Piv
Ori = (ORI_RELATIVE_TO($Data_Name.Ori_Cm_Pivot, $The_Model.$'Name'__Prt_Piv
!
Marker Create Marker = $The_Model.$'Name'__Prt_Pivot.Geo_Pivot_Ref &
Loc = (LOC_RELATIVE_TO(-0.5 * $Data_Name.Length_Pivot * {0.0, 0.0, 1.0}, $T
Ori = (ORI_RELATIVE_TO({0.0, 0.0, 0.0}d, $The_Model.$'Name'__Prt_Pivot.Refe
!
Marker Create Marker = $The_Model.$'Name'__Prt_Pivot.Ref_Robot_Arm &
Loc = (LOC_RELATIVE_TO($Data_Name.Loc_Robot_Arm, $The_Model.$'Name'__Prt_Pi
Ori = (ORI_RELATIVE_TO({0.0d, 90.0d, 0.0d}, $The_Model.$'Name'__Prt_Pivot.B

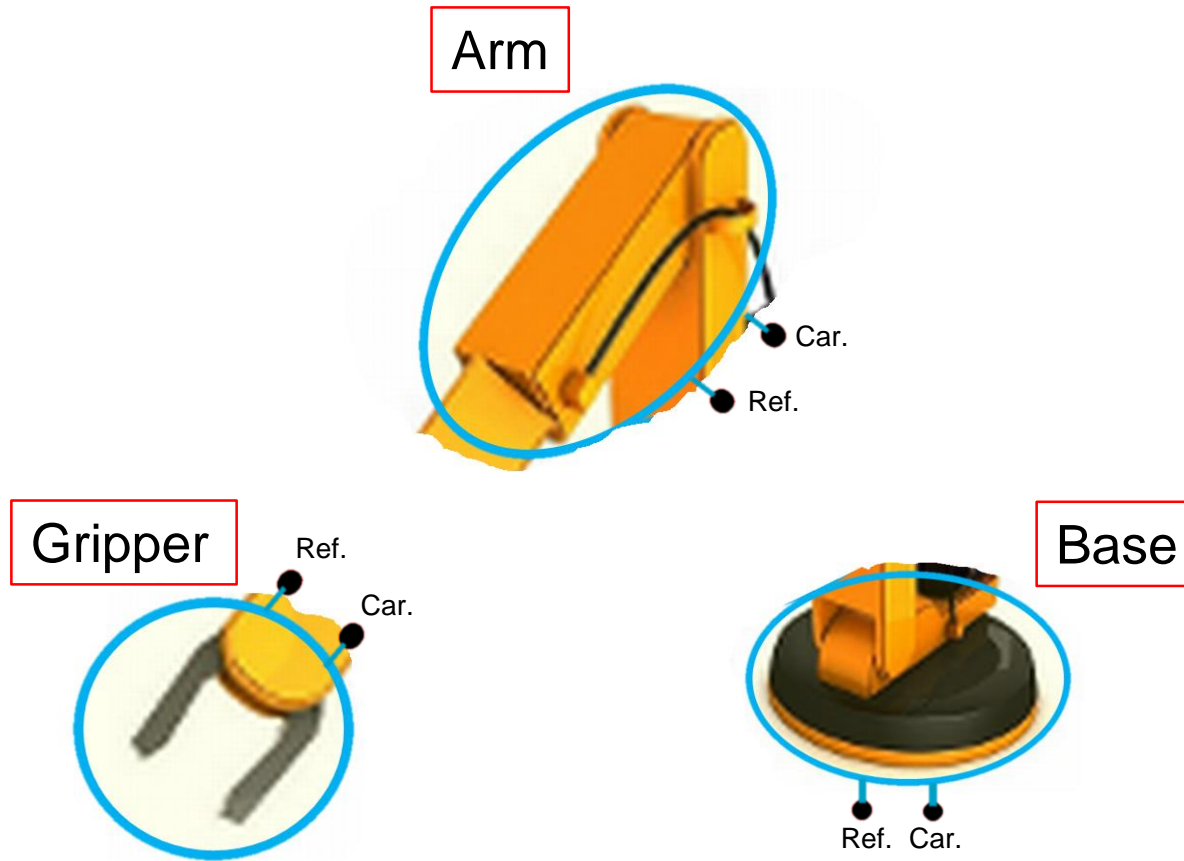
```

```

!
Var Create Var = $Name._Grp_Inertia_Data String = "End=Ori_Cm_Pivot" Comment = "Robot Base In
!
Var Create Var = $Name.Mass_Pivot Real = (10.0kg) Units = Mass Comment = "Mass of Part
Var Create Var = $Name.Inert_Pivot Real = ({1.0, 1.0, 1.0}(kg-m**2)) Units = Inertia Comment = "Main Inertia
Var Create Var = $Name.Icross_Pivot Real = ({0.0, 0.0, 0.0}(kg-m**2)) Units = Inertia Comment = "Cross-Inertia
Var Create Var = $Name.Loc_Cm_Pivot Real = ({0.0, 0.0, 0.0}m) Units = Length Comment = "Loc. of Part
Var Create Var = $Name.Ori_Cm_Pivot Real = ({0.0, 0.0, 0.0}d) Units = Angle Comment = "Ori. of Part
!

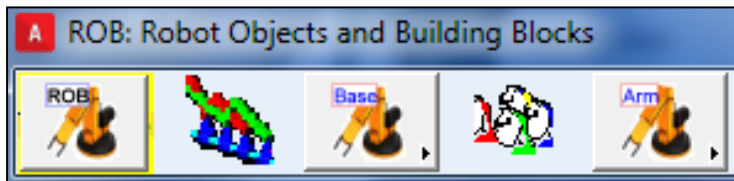
```

Robot Components Overview

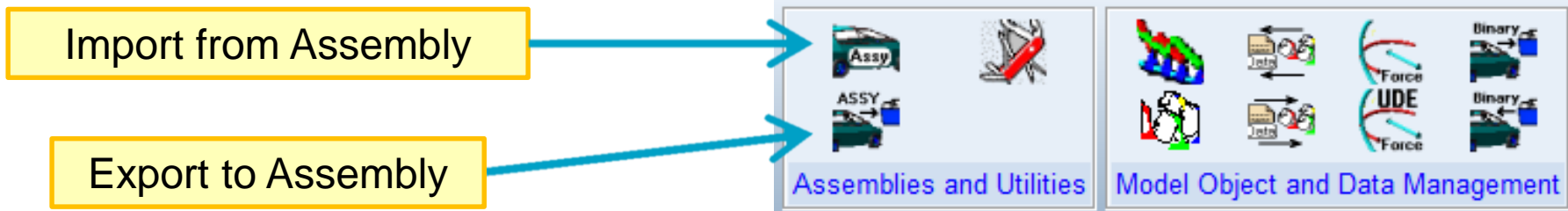


Step 3: Creation of Components Based Robot

- Component definitions are now stored in Toolkit macro files (or generic)
- We can now use component dialogs to create a full robot



- Next, the robot model can be exported to an assembly file



- This assembly can be used as central repository of the robot model
- Multiple assembly files can be created and maintained

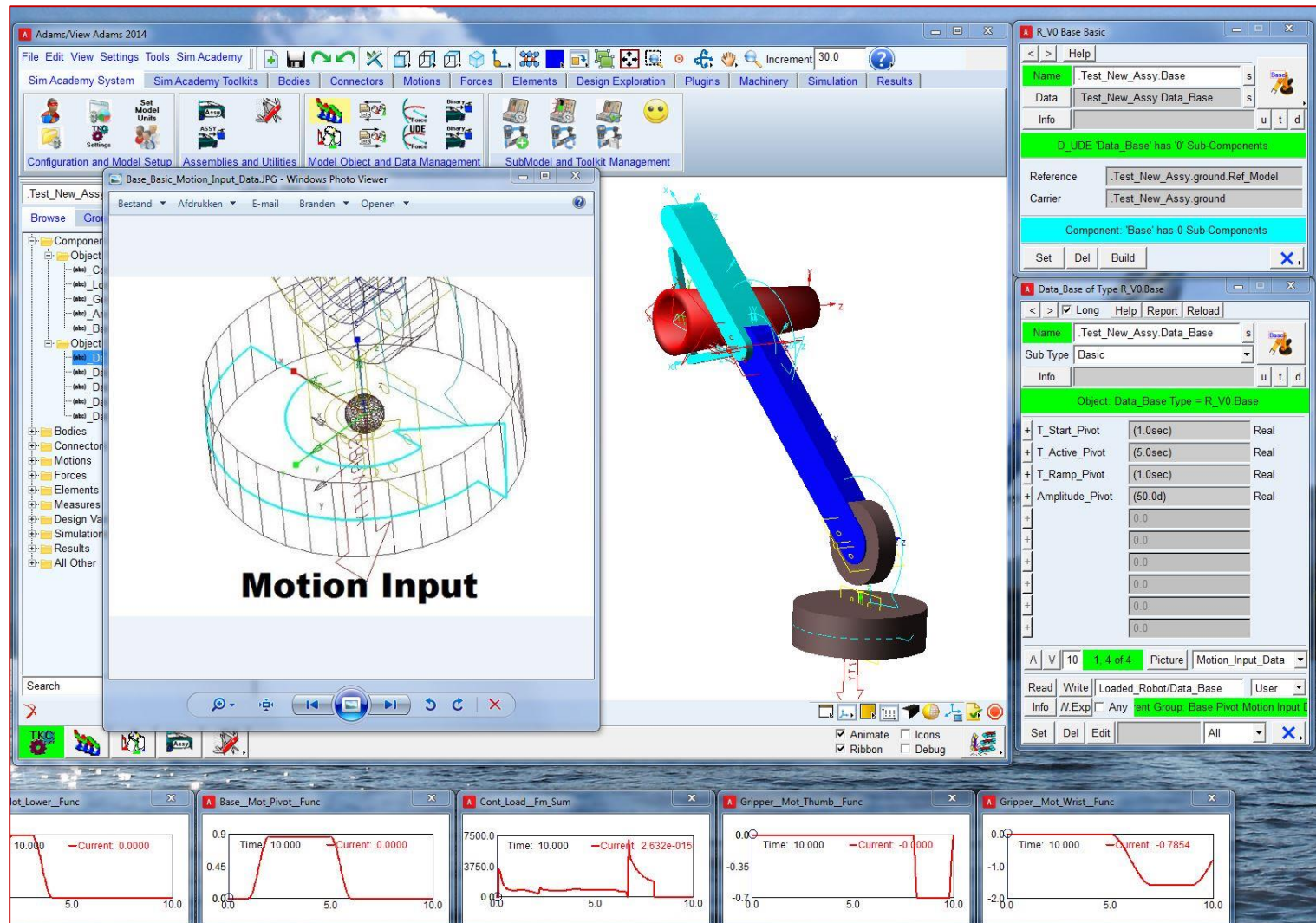
Typical Assembly File Structure

- Assembly header with arbitrary number of arguments for user input of:
 - Data Files,
 - Component types (i.e. Rigid vs. Flex)
- Data Objects Creation from Data Library Files
- Model Backbone
 - i.e. factory floor, range of markers or testrig Component
- Model Components
- Final Adjustments,
 - Model modifications and additions
 - Simulation settings

Final Simulation Project Results

- Range of robot simulation models stored in assembly files
- Model data stored in data library
- Model data and components are created from assembly
- **Model assembly is separated from component updates**
- Assemblies can be expanded, updated and improved by:
 - Manual coding (standard Aview syntax)
 - Automatic model extraction

TKC ROB Toolkit: Sample Component View



Final Remarks

- TKC is used in the Benelux area by majority of the ADAMS users
 - Using TKC creates more commitment to the ADAMS product and empowers them to manage more complex models and projects
- TKC is now commercially available as an ADAMS/View add-on product
 - World wide available to ADAMS/View users
- A TKC installation consists of:
 - The TKC Core + *Shared* Toolkits (GUM, DFM, CSM,...),
 - Selected *Site* toolkits for a wide range of industries:
Aerospace, Offshore, Heavy Industries, Consumer Products,
General Machinery, Oil Industry, Medical, Packaging,
 - Your Proprietary Toolkits !!!
 - How about encapsulating your current Aview models ??
- Please contact us in case you want to start using TKC in AView:
 - www.sayfield.nl or chris.verheul@sayfield.nl
 - And/or through your local MSC.Software agency