SmartPart: A Reconfigurable Modular Strategy for Knowledge Capture and Reuse

Brian Prasad

bprasad@parker.com

Jeff Rogers

jrogers@parker.com

Parker Hannifin Corporation

Aerospace Group
Control Systems Division
14300 Alton Parkway
Irvine, CA 92618

Strategic Charter

To be the premier provider of motion and control systems for our global customers

Mission

Our mission is to provide unequaled value through

superior performance
technical innovation
speed and responsiveness
premier customer service
financial strength

to our customers, company, team members, and community

Vision

JustOne TEAM

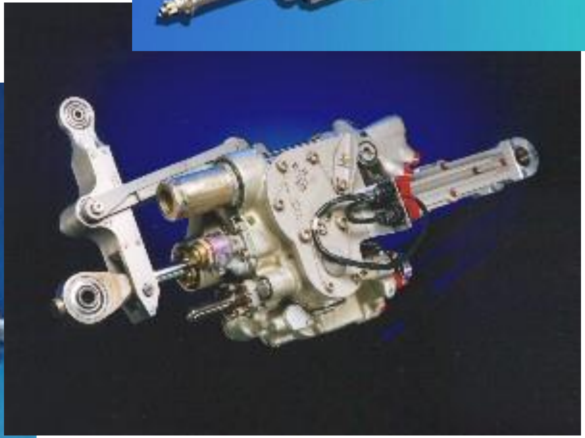
...our lean vision

Continuous Improvement

Vision

The Spirit of Innovation CSD

Innovation



Parker PLM Strategy?

Systems Engineering View
Configure and reconfigure solutions based on changing specs-requirements as often as you may wish

JustOne Approach
Think "families" of parts; identify a base model from which various "part or product subsystems" can be adapted to meet the same system/customer specs

Knowledge-Driven Automation (KDA)
Develop an architecture to "generatively" build a product tree for each possible situation, based on rules [or rule bodies].

Parker PLM Strategy?

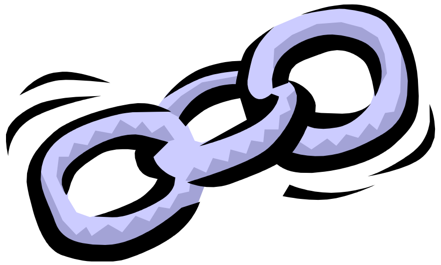
Part Intelligence
Employ a template-based approach for building intelligent objects; Identify rules and "generate" a top-down structure for each product scenario. (Use scripting for rules, not geometry)

Part-Independent Communication
Pass dependent parameters from "systems" to "subsystems" to "components" to "features" during "decomposition" and vice-versa during "aggregation"

Product Configuration
Employ a modular approach for building configuration families. As modules are "instantiated" and constraints are satisfied, "new leaves" are made and hung on the product tree

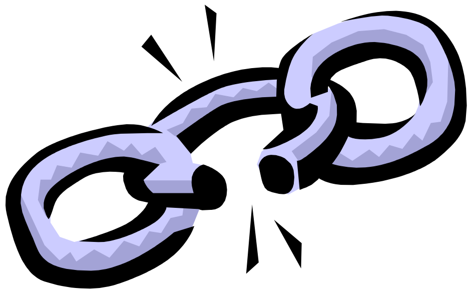
How Do We Build Intelligence Into Parts With PLM Today?

- In CATIA V5 tools, the most common ways are for designers to write KWA/KWE rules, design tables, UDFs, and formulas on the geometric attributes of parts
- By doing so, designers embed the valuable engineering rules and equations in properties that are linked to an intrinsic geometric feature
- The two ("engineering rules" and "geometric properties") becomes inseparable.



How Do We Build Intelligence Into Parts With PLM Today?

- If we change some features in the geometry (due to needed maintenance), previously written rules may not work
- As such, after each addition or subtraction of new geometric features, the associated rules or links need to be rebuilt
- If by accident a portion of the CATIA geometry gets corrupted, or interruptions occur while running CATIA, links break, the part model needs to be aborted, forcing the entire CATIA model to be reworked

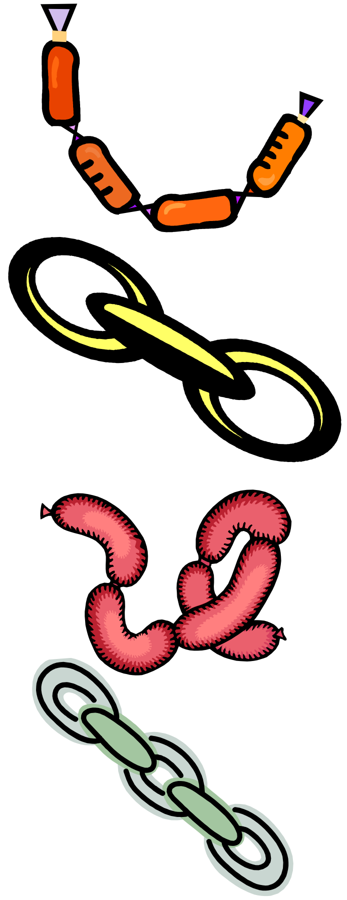


These Shortcomings Prompted The Following Questions…

- Can we separate sizing rules from a “base parametric geometry”?
- Can we flow information so that it’s reusable across various geometric models?
- Can the necessary links be established dynamically at run time?

Ways To Link Two Templates?

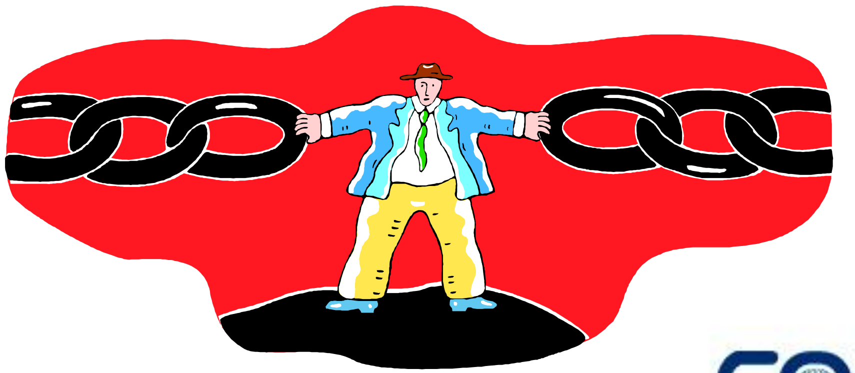
- There are many ways to link 2 templates (or models) in CATIA V5. The most popular ways are…
  - PowerCopy and combine the two into "one BIG CATPart"
  - KWA/Reaction and Action linking method
  - Formulae/ "External Parameters" linking method
  - Functional Calls (GetAttribute/SetAttribute) linking method
  - Excel table attribute linking via design table and "associations"



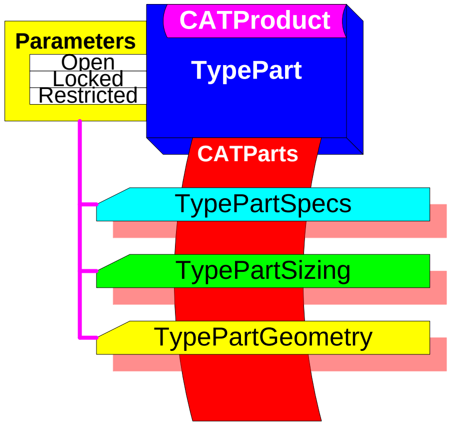
How to Build Reconfigurable Parts?

- Over the years, Parker has investigated many such modeling and knowledge reuse requirements.

- After careful analysis of all such available options, Parker developed **a novel concept for building modular and reconfigurable SmartParts.**



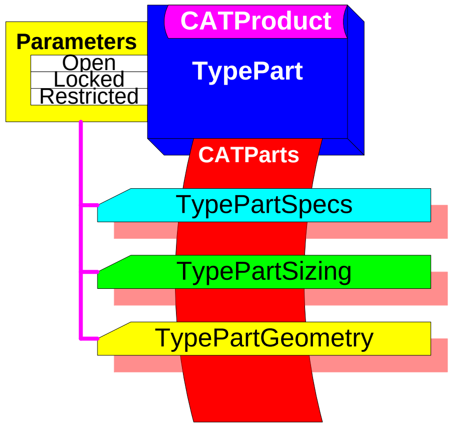
What is a SmartPart?



…a modular approach for building intelligent parts, features, or objects.

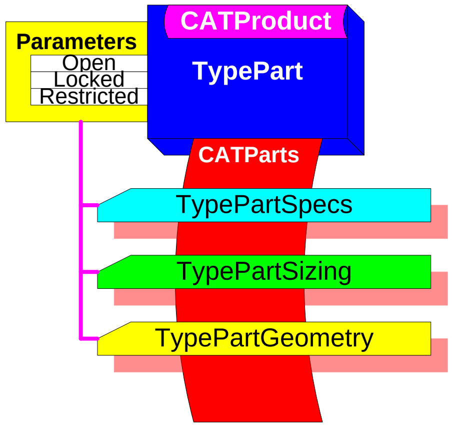
The SmartPart template was introduced in our KDA (Knowledge-Driven Automation) Project at Parker. It was central to building a flexible system architecture for KDA in 2004.

What is a SmartPart?



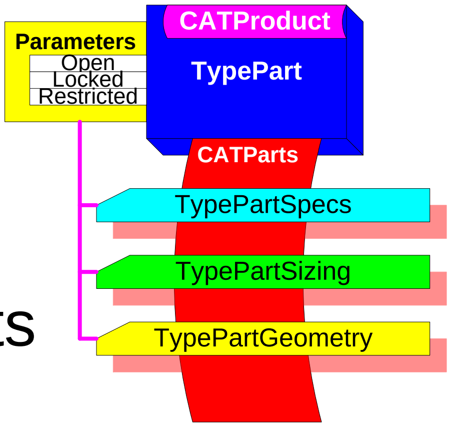
- SmartPart is a concept of building intelligent physical parts based on “Structure Similarity”
- SmartPart concept logically decomposes and defines a physical part information into two or more of its components. (One or more of the CATIA V5 feature definition tools could be used here).
  - Some examples of feature definition tools are: V5 PKT/UDF, a PowerCopy, a PKT part template, any V5 CATPart (w/wo geometry), a method, a set of equations, or simply a collection of parameters sets

Parker SmartPart–Strategy?



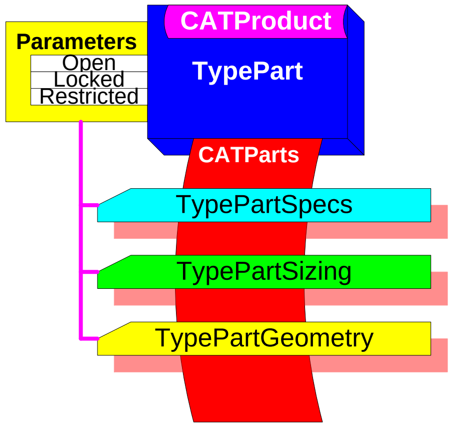
- Separates “Geometric” from “Resizing” (engineering-centered) parameters
- Built on three sets of CATPart (templates)
  - XY**Specs**.CATParts
  - XY**Sizing**.CATParts
  - XY**Geometry**.CATParts

What Constitute a Specs Component?



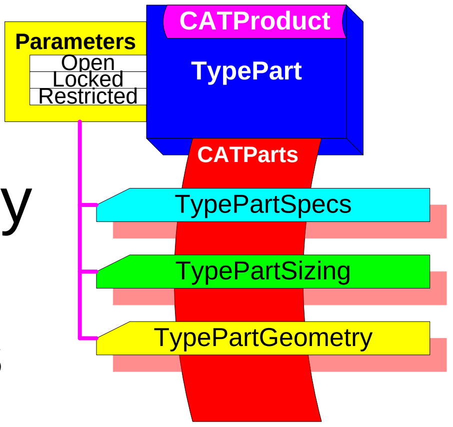
- Standalone and Reusable
- Specs component (say XYSpecs.CATPart) has NO geometry.
- Separates independent customer requirements from sizing parameters—explicit, clear, and system modifiable
- Organizes specification rules into "Spec templates" (like material tables, specification table, fatigue table, rules, etc.) engineers can instantiate to expedite their design and analysis tasks
- Parameters in Specs.CATPart can be modified by an external system (If this is used as a module of a larger system)

What Constitute a Sizing Component?



- Standalone and Reusable
- Sizing component (say XYSizing.CATPart) has NO geometry
- Engineering-intent rules are used to control the functional intent of the design, which in turn are related to a set of geometric parameters but no resulting geometry is created in this component.
- Captures "sizing" rules into reusable "living templates" (like design table, catalogues, scripts, rules, system of equation solvers, etc.)

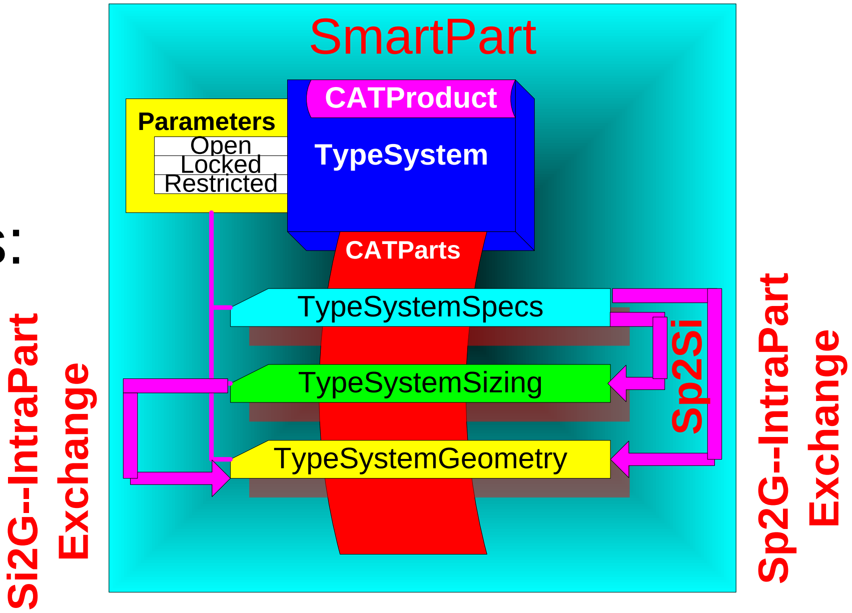
What Constitute a Geometry Component?



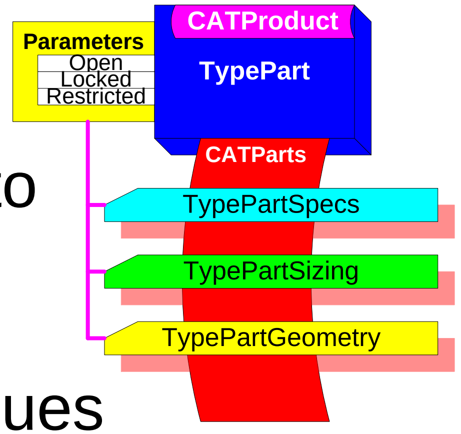
- Standalone and Reusable
- Employs best practices in parametric geometry components (say XYGeometry.CATPart)
- Captures more than just geometry. It captures product design intent (i.e. life-cycle functional intent) in addition to parametric geometry.
  - Embeds all sorts of geometric rules, formulae, parameters, relations, and constraints.
  - Applies standards via design tables.
- Allows CATIA V5 models that are standalone, instantiable, generic, and reusable across their life–cycle.

A Dynamic Linking Method

- Parker developed a VB-based *SearchAttribute* Method
- It searches 2 select components: (say a driving-component and a driven-component).
- It identifies a set of user-defined parameters that are matched (common across the 2 Parts).
- It then sets (equates) values of the matched user-defined parameters in the driven-component with the driving-component.

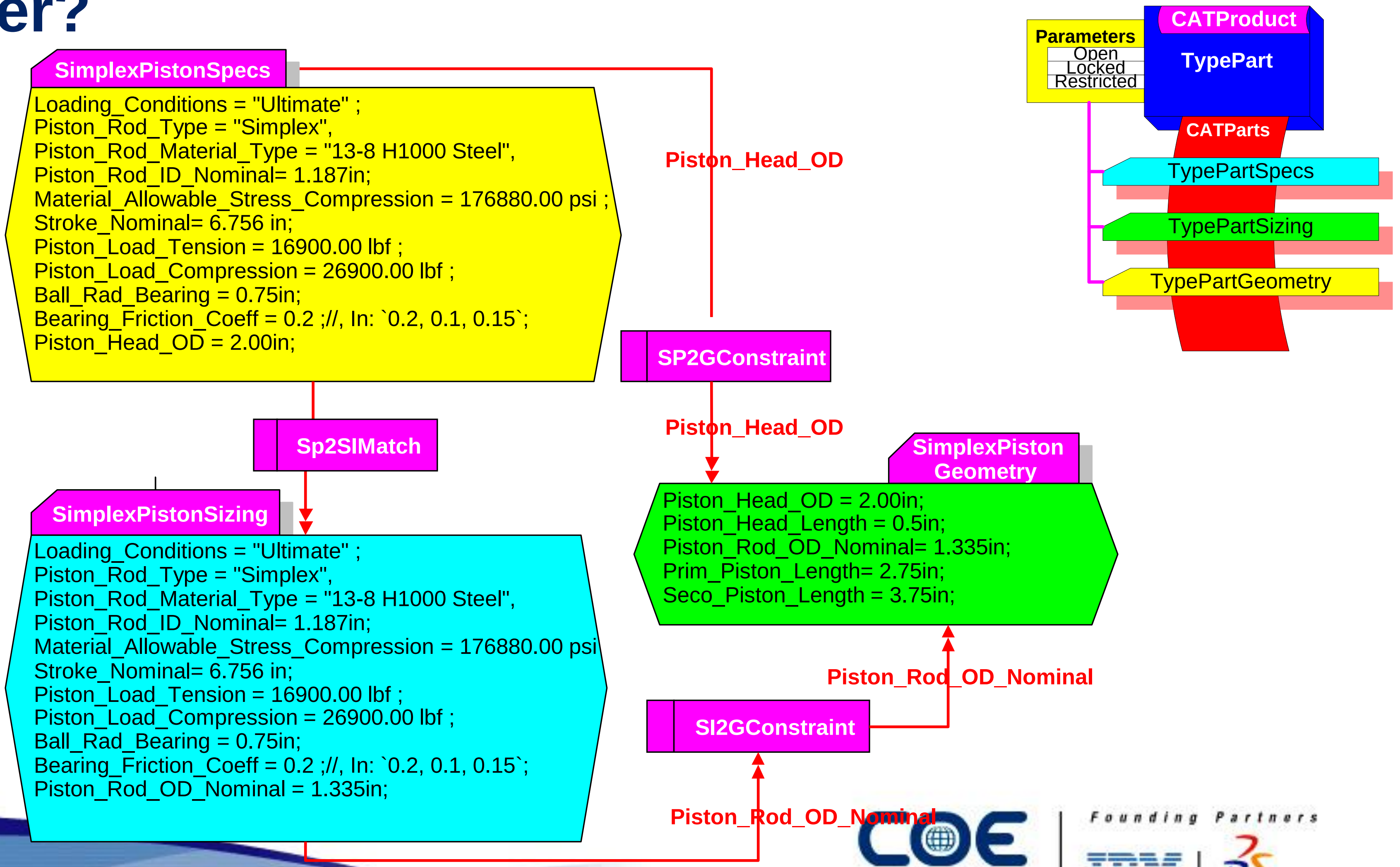


How Components are Communicating with Each Other?



- A generalized method (written in VBScript) -- called ***SearchAttribute*** Method is employed to pass/exchange information across Intra-part components.
- ***SearchAttribute*** Method controls passing values of "matched attributes" across various pair of components.
- This way ***SmartPart*** definition becomes generic and "reusable." (e.g., the same configurator is used to create 14 Actuator Smart Parts in minutes).

How Components are Communicating with Each Other?



SimplexPistonSpecs

Loading_Conditions = "Ultimate" ;
Piston_Rod_Type = "Simplex",
Piston_Rod_Material_Type = "13-8 H1000 Steel",
Piston_Rod_ID_Nominal= 1.187in;
Material_Allowable_Stress_Compression = 176880.00 psi ;
Stroke_Nominal= 6.756 in;
Piston_Load_Tension = 16900.00 lbf ;
Piston_Load_Compression = 26900.00 lbf ;
Ball_Rad_Bearing = 0.75in;
Bearing_Friction_Coeff = 0.2 ;//, In: `0.2, 0.1, 0.15`;
Piston_Head_OD = 2.00in;

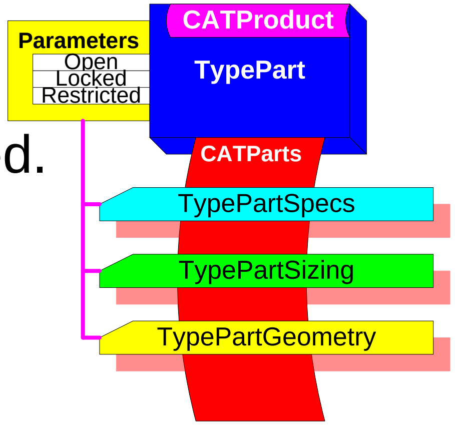
SimplexPistonSizing

Loading_Conditions = "Ultimate" ;
Piston_Rod_Type = "Simplex",
Piston_Rod_Material_Type = "13-8 H1000 Steel",
Piston_Rod_ID_Nominal= 1.187in;
Material_Allowable_Stress_Compression = 176880.00 psi
Stroke_Nominal= 6.756 in;
Piston_Load_Tension = 16900.00 lbf ;
Piston_Load_Compression = 26900.00 lbf ;
Ball_Rad_Bearing = 0.75in;
Bearing_Friction_Coeff = 0.2 ;//, In: `0.2, 0.1, 0.15`;
Piston_Rod_OD_Nominal = 1.335in;

SimplexPiston Geometry

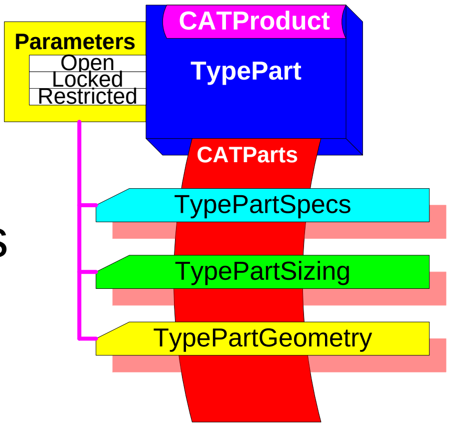
Piston_Head_OD = 2.00in;
Piston_Head_Length = 0.5in;
Piston_Rod_OD_Nominal= 1.335in;
Prim_Piston_Length= 2.75in;
Seco_Piston_Length = 3.75in;

Why employ a "Part-independent" SearchAttribute Methods?



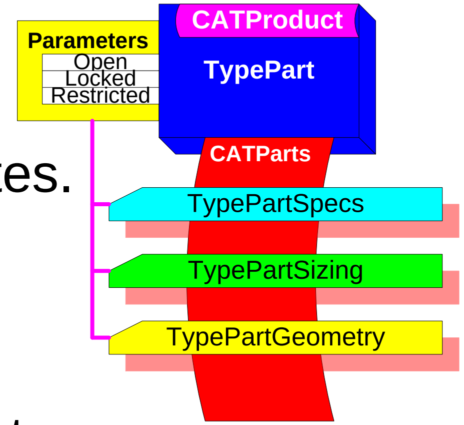
- With "***Part-independent***" ***SearchAttribute*** method, SmartPart components can be independently developed.
- Each component may constitute a semi-independent element of the Part structure. For example, often specification is semi-independent of sizing, and sizing is semi-independent of geometry.
- No "External Parameters" links are needed at any place. Original SmartPart components, (templates, formulas, structure, etc.) remain intact.
- The ***SearchAttribute*** method is reusable across various (any 2-components pairs) of a smartPart concept.
- Plug and Play provision is maintained throughout
- The ***SearchAttribute*** method is also reusable across system to its subsystems deployment stage.
- Less clutter and each template is easy to understand and maintain.

Why employ a “Part-independent” SmartPart Concept?



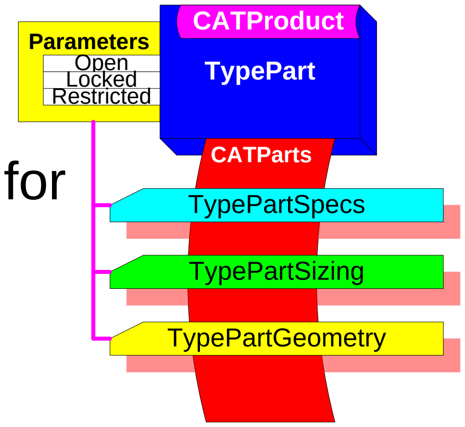
- Once a ***SmartPart*** definition is set, it can be reused many times.
- A single ***SmartPart*** definition creates 14 physical parts of a Tandem Actuator, 14 Parts of Simplex and 14 Parallel Parts in 3 modes: restricted, open and locked.
  - Without this ***SmartPart*** definitions, users would have to create altogether (14*3*3 = 126 CATParts.)
  - The specified ***SmartPart*** CATpart is created on a fly, on demand and based on your inputs at runtime -- requirements (e.g. Actuator_type and Security_type).

Why employ a “Part-independent” SmartPart Concept?



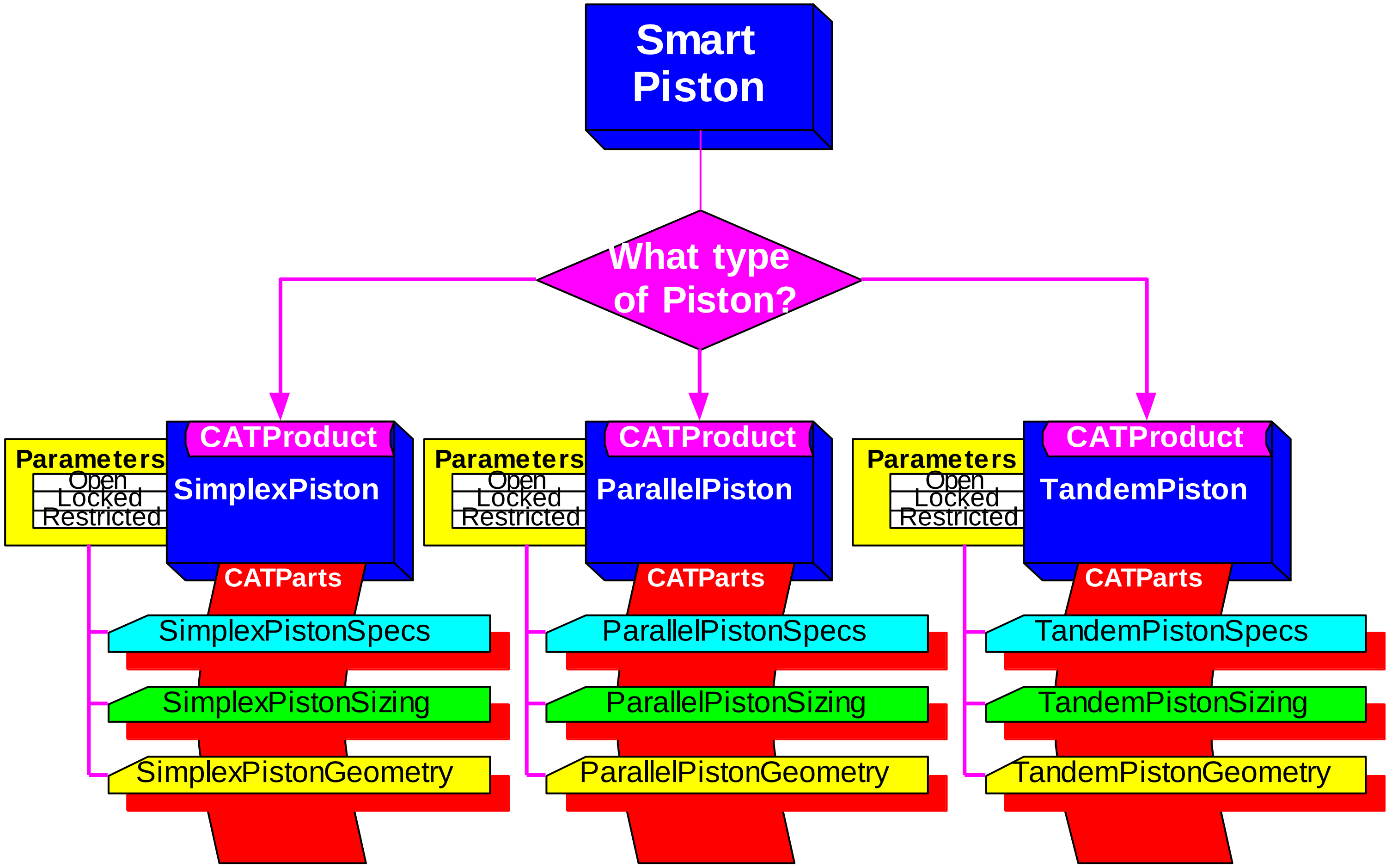
- No chances of introducing any human error.
  - Typing is eliminated by a pull_down choice attributes.
- It is done very rapidly and accurately.
- Even after ***SmartPart*** Product tree structure is build, saved, components are interchangeable.
  - You can plug out any one of the ***SmartPart*** templates (say Sizing, by another sizing and the system would perform as designed.
  - Rules will operate on the replaced component as if nothing is changed.

What is the "SmartPart Process"?

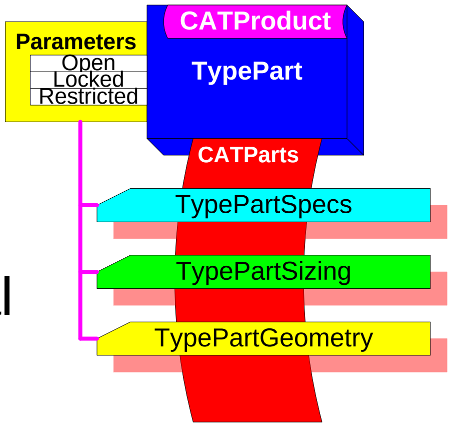


- ***SmartPart process*** is designed for building Parker family of products – (e.g., ***JustOne*** Part configurator for each part in the product assembly).
- It consists of the following ***4 toolkit*** components:
  - ***SmartPart Configurator***
  - ***Smart Part Concept***
  - ***SearchAttribute Methods,***
    - A visual-based method
    - A generative-script method
  - ***Best Practices***.
    - Use "functional, ***template-based***, feature-based" modeling approach for capturing knowledge and modeling components in CATIA V5.
    - Follow a top-down ***system-engineering*** approach to product design and development.

How Smart Part configurator works?

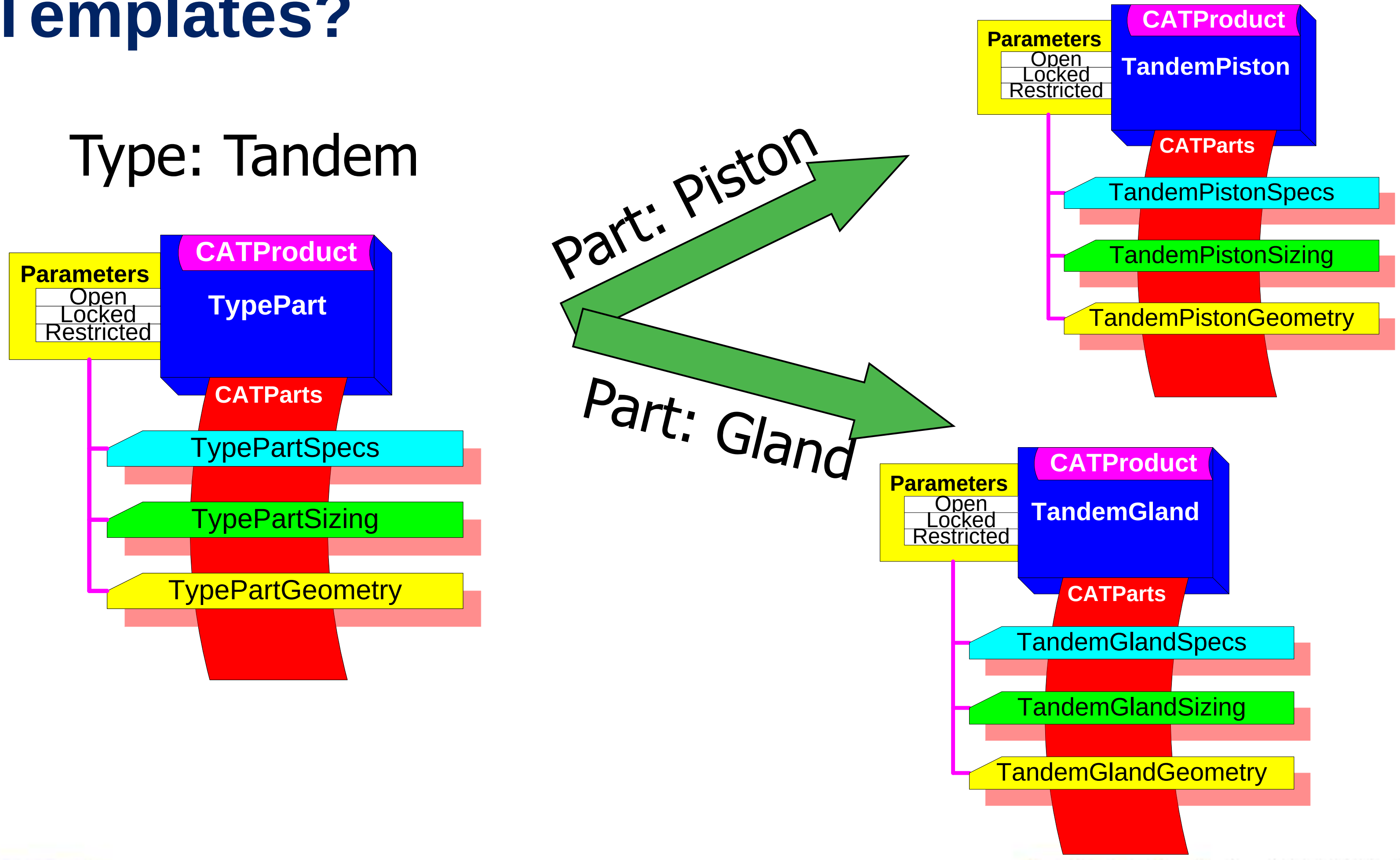


Building SmartParts: Knowledge Capturing Strategy



- Employ a KBE “capturing (scripting) language” to construct a reconfigure-able and smart model of the product
- Built collections of intelligent reconfigure-able functional parts
- Built Architecture for configuring products based on users/customers requirements
- Each components/modules is driven by same set of input specifications
- Assemble a reconfigure-able product assembly (CATProduct) -- where components (CATParts) are itself reconfigure-able.

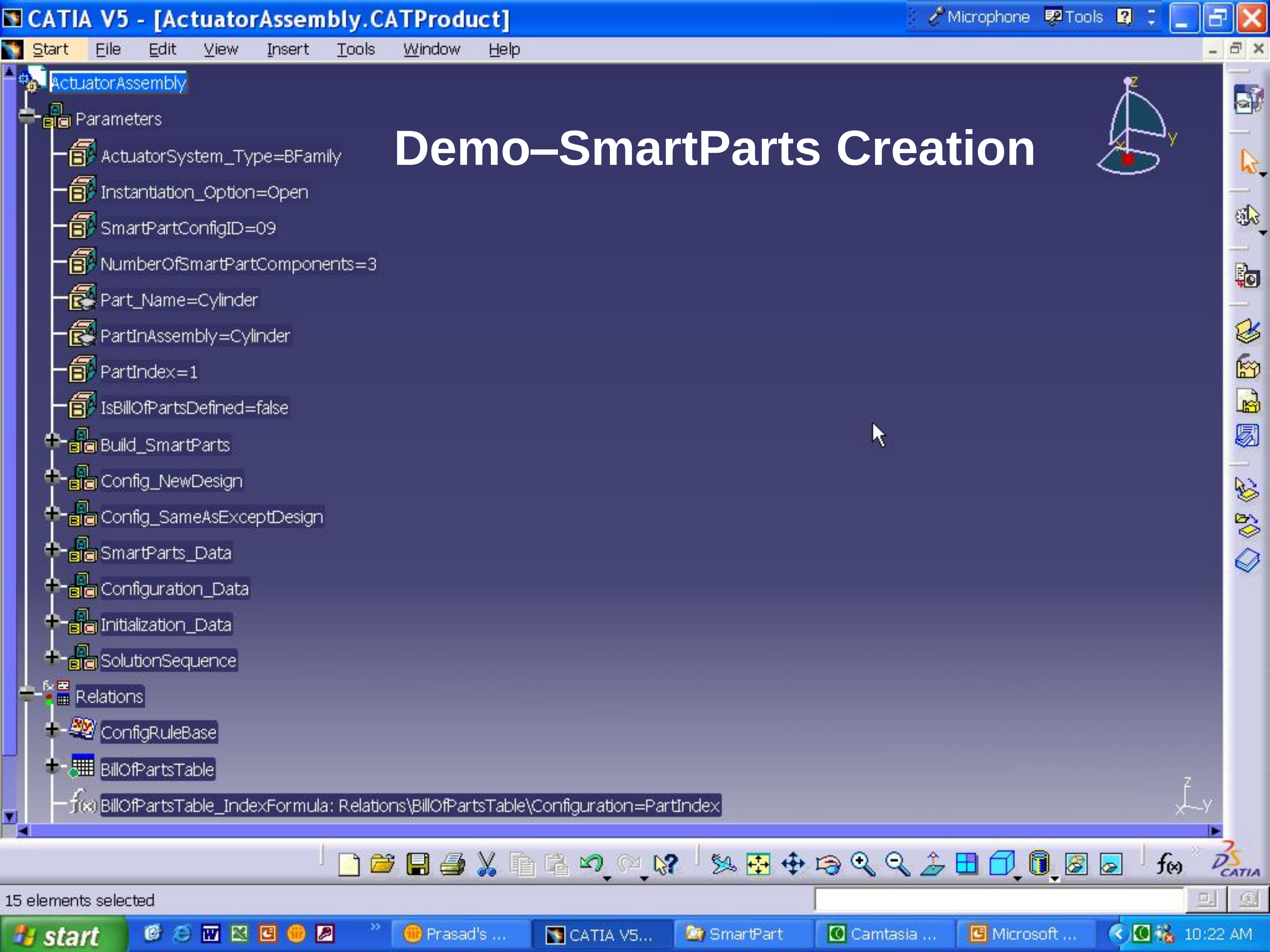
Rapid Creation of SmartParts based on Reusable Templates?



Demo–What's in play?

- Three templates for each part
- Other design tables in background
- A new CATProduct for each part

Demo–SmartParts Creation

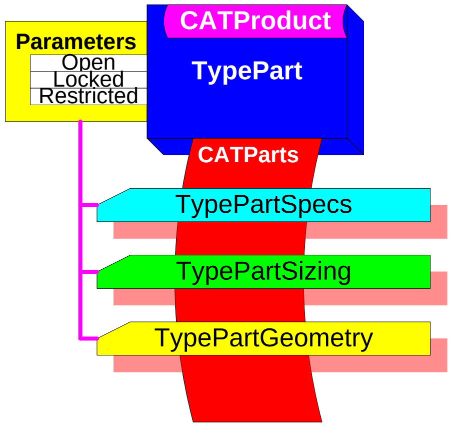


CATIA V5 - [ActuatorAssembly.CATProduct]

ActuatorAssembly
- Parameters
  - ActuatorSystem_Type=BFamily
  - Instantiation_Option=Open
  - SmartPartConfigID=09
  - NumberOfSmartPartComponents=3
  - Part_Name=Cylinder
  - PartInAssembly=Cylinder
  - PartIndex=1
  - IsBillOfPartsDefined=false
  - Build_SmartParts
  - Config_NewDesign
  - Config_SameAsExceptDesign
  - SmartParts_Data
  - Configuration_Data
  - Initialization_Data
  - SolutionSequence
- Relations
  - ConfigRuleBase
  - BillOfPartsTable
  - BillOfPartsTable_IndexFormula: Relations\BillOfPartsTable\Configuration=PartIndex

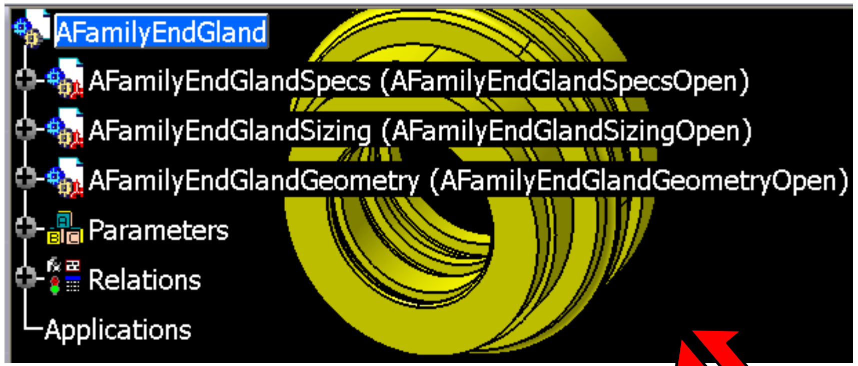
15 elements selected

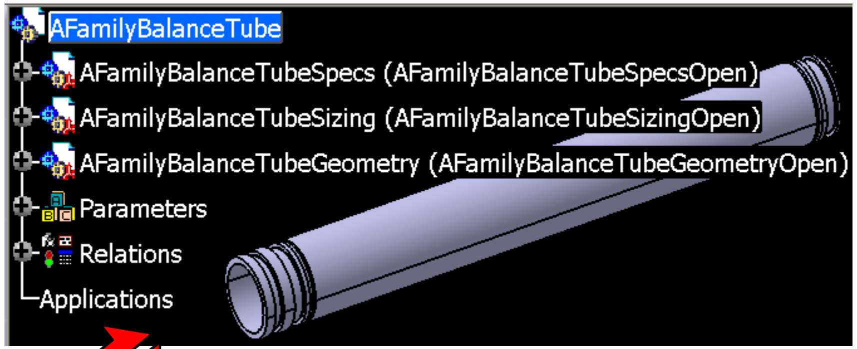
Demo–Salient points



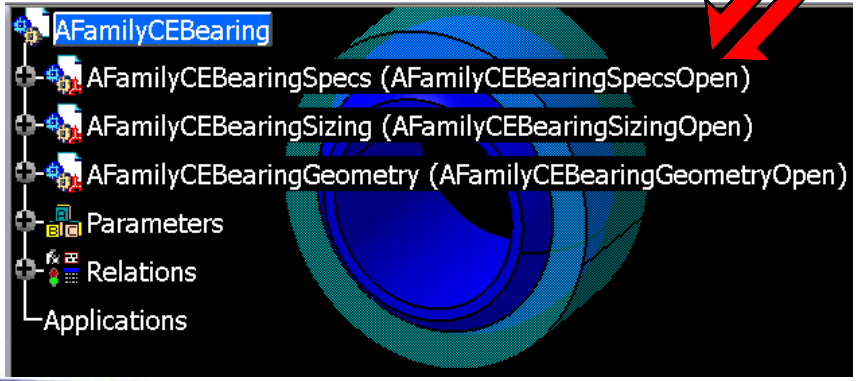
- Defined new rules for creating SmartParts
  - Reusability
- Rules fired to build new product tree
  - Extensibility
- Each product tree has three components of SmartPart
  - Systematization
- Intra-part relations were established to bind components of the SmartPart
  - External links eliminated, maintainability

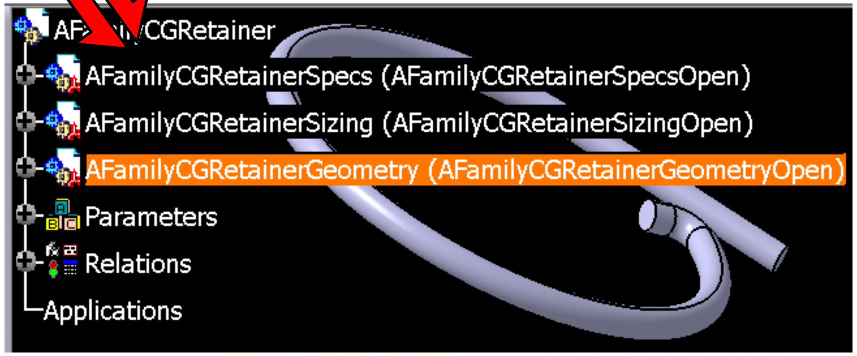
SmartPart Wizard- Examples


AFamilyEndGland
- AFamilyEndGlandSpecs (AFamilyEndGlandSpecsOpen)
- AFamilyEndGlandSizing (AFamilyEndGlandSizingOpen)
- AFamilyEndGlandGeometry (AFamilyEndGlandGeometryOpen)
- Parameters
- Relations
- Applications

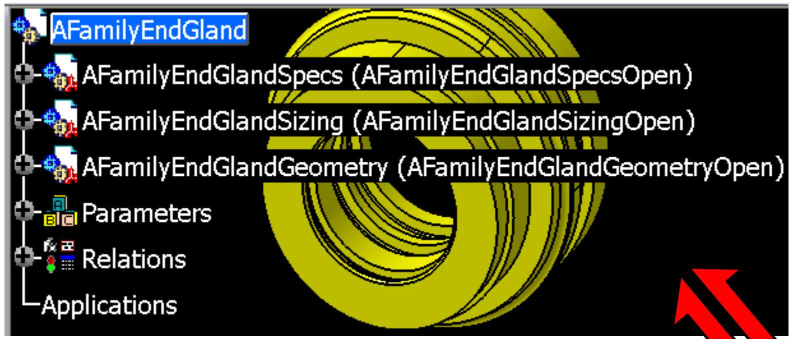

AFamilyBalanceTube
- AFamilyBalanceTubeSpecs (AFamilyBalanceTubeSpecsOpen)
- AFamilyBalanceTubeSizing (AFamilyBalanceTubeSizingOpen)
- AFamilyBalanceTubeGeometry (AFamilyBalanceTubeGeometryOpen)
- Parameters
- Relations
- Applications

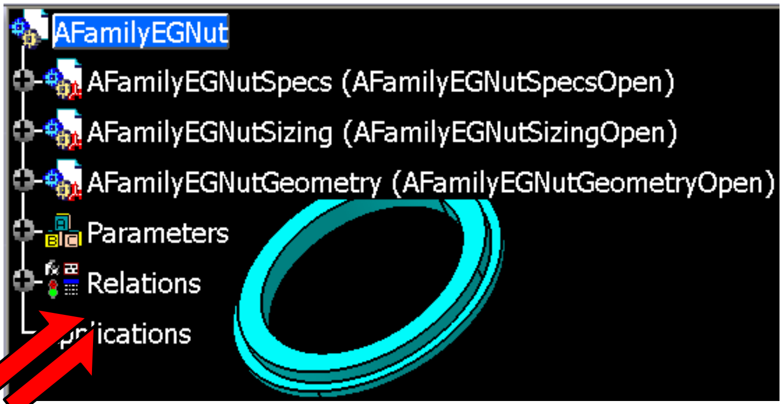
Part Wizard


AFamilyCEBearing
- AFamilyCEBearingSpecs (AFamilyCEBearingSpecsOpen)
- AFamilyCEBearingSizing (AFamilyCEBearingSizingOpen)
- AFamilyCEBearingGeometry (AFamilyCEBearingGeometryOpen)
- Parameters
- Relations
- Applications

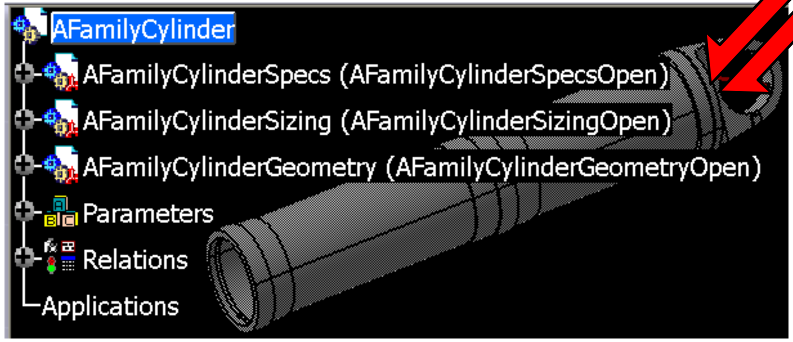

AF[illegible]CGRetainer
- AFamilyCGRetainerSpecs (AFamilyCGRetainerSpecsOpen)
- AFamilyCGRetainerSizing (AFamilyCGRetainerSizingOpen)
- AFamilyCGRetainerGeometry (AFamilyCGRetainerGeometryOpen)
- Parameters
- Relations
- Applications

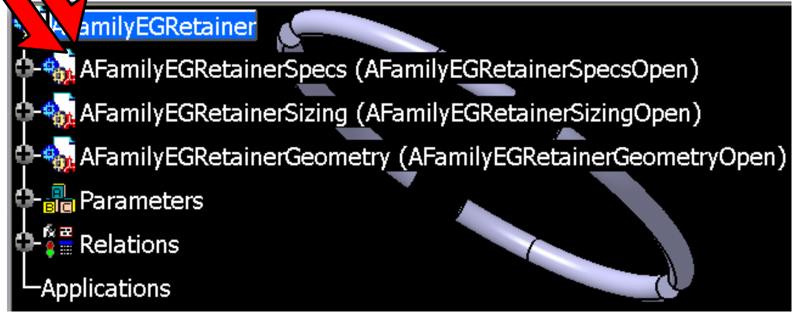
SmartPart Wizard- Examples


AFamilyEndGland
AFamilyEndGlandSpecs (AFamilyEndGlandSpecsOpen)
AFamilyEndGlandSizing (AFamilyEndGlandSizingOpen)
AFamilyEndGlandGeometry (AFamilyEndGlandGeometryOpen)
Parameters
Relations
Applications

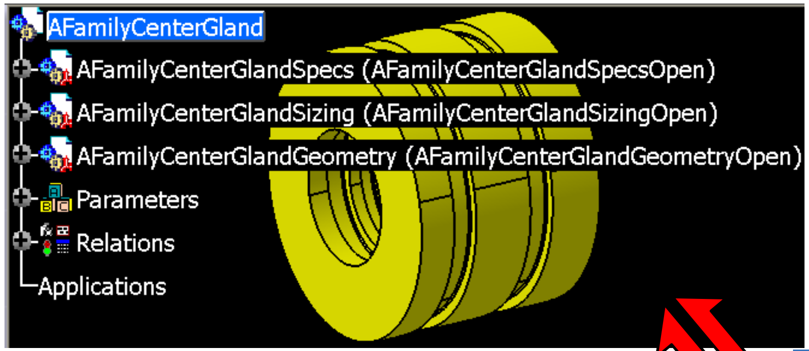

AFamilyEGNut
AFamilyEGNutSpecs (AFamilyEGNutSpecsOpen)
AFamilyEGNutSizing (AFamilyEGNutSizingOpen)
AFamilyEGNutGeometry (AFamilyEGNutGeometryOpen)
Parameters
Relations
Applications

Part Wizard

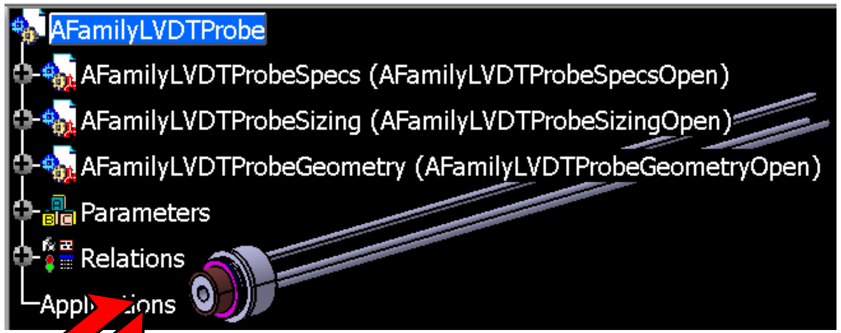

AFamilyCylinder
AFamilyCylinderSpecs (AFamilyCylinderSpecsOpen)
AFamilyCylinderSizing (AFamilyCylinderSizingOpen)
AFamilyCylinderGeometry (AFamilyCylinderGeometryOpen)
Parameters
Relations
Applications


AFamilyEGRetainer
AFamilyEGRetainerSpecs (AFamilyEGRetainerSpecsOpen)
AFamilyEGRetainerSizing (AFamilyEGRetainerSizingOpen)
AFamilyEGRetainerGeometry (AFamilyEGRetainerGeometryOpen)
Parameters
Relations
Applications

SmartPart Wizard- Examples

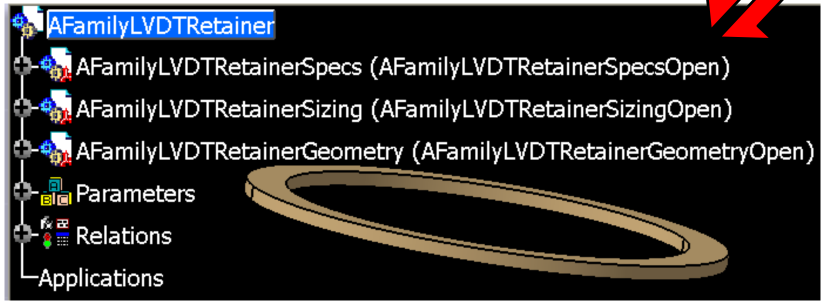


AFamilyCenterGland
AFamilyCenterGlandSpecs (AFamilyCenterGlandSpecsOpen)
AFamilyCenterGlandSizing (AFamilyCenterGlandSizingOpen)
AFamilyCenterGlandGeometry (AFamilyCenterGlandGeometryOpen)
Parameters
Relations
Applications

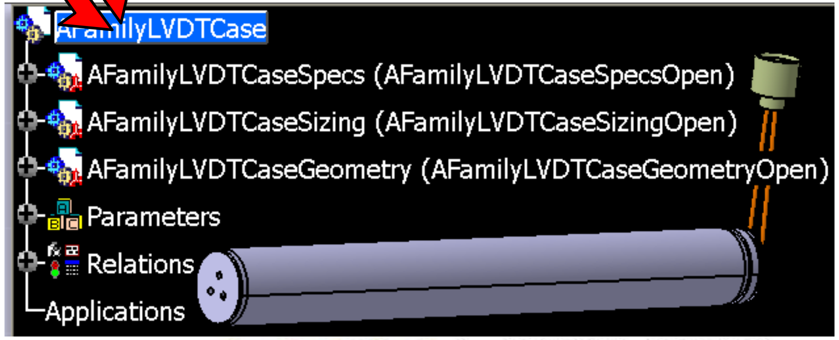


AFamilyLVDTProbe
AFamilyLVDTProbeSpecs (AFamilyLVDTProbeSpecsOpen)
AFamilyLVDTProbeSizing (AFamilyLVDTProbeSizingOpen)
AFamilyLVDTProbeGeometry (AFamilyLVDTProbeGeometryOpen)
Parameters
Relations
Applications

Part Wizard

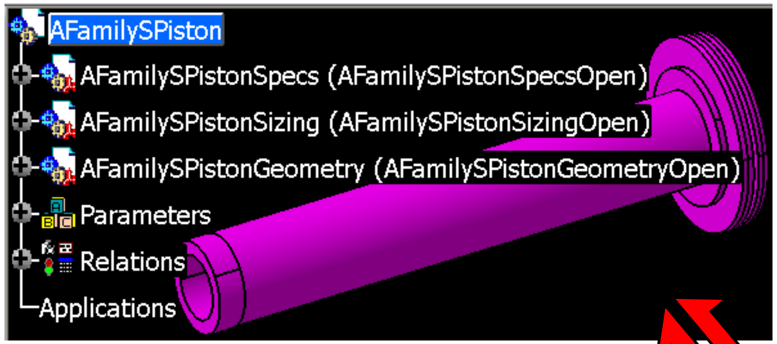


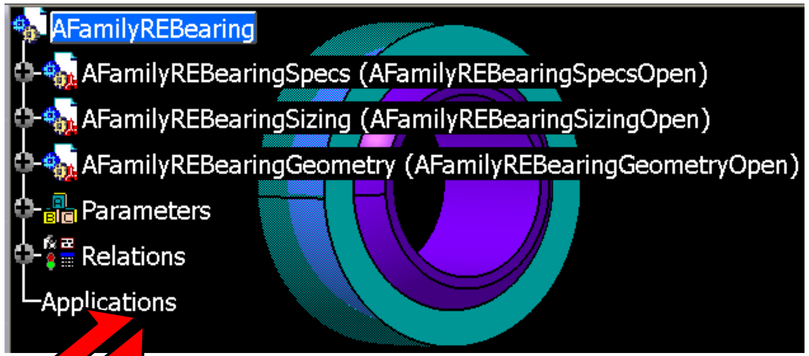
AFamilyLVDTRetainer
AFamilyLVDTRetainerSpecs (AFamilyLVDTRetainerSpecsOpen)
AFamilyLVDTRetainerSizing (AFamilyLVDTRetainerSizingOpen)
AFamilyLVDTRetainerGeometry (AFamilyLVDTRetainerGeometryOpen)
Parameters
Relations
Applications



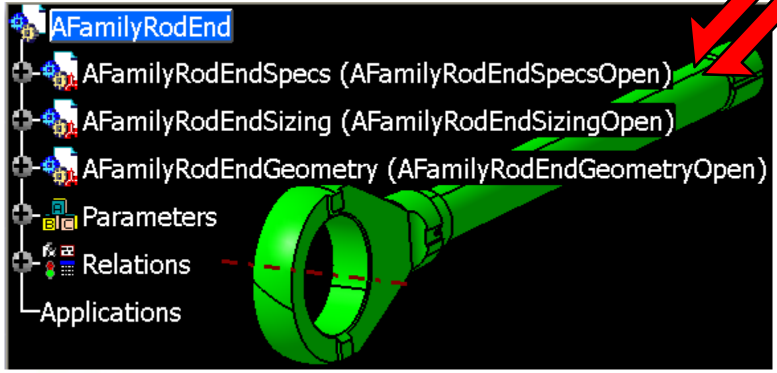
AFamilyLVDTCase
AFamilyLVDTCaseSpecs (AFamilyLVDTCaseSpecsOpen)
AFamilyLVDTCaseSizing (AFamilyLVDTCaseSizingOpen)
AFamilyLVDTCaseGeometry (AFamilyLVDTCaseGeometryOpen)
Parameters
Relations
Applications

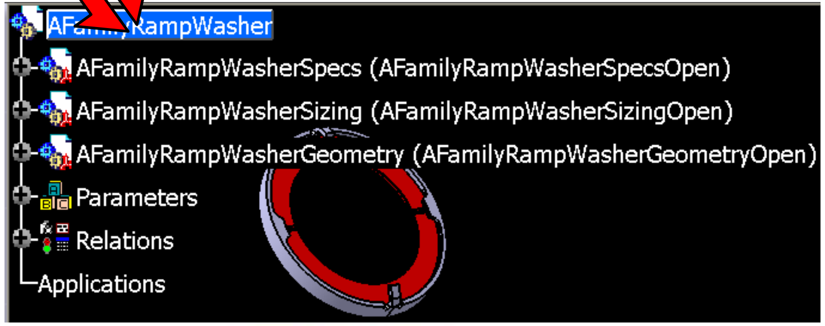
SmartPart Wizard- Examples


AFamilySPiston
AFamilySPistonSpecs (AFamilySPistonSpecsOpen)
AFamilySPistonSizing (AFamilySPistonSizingOpen)
AFamilySPistonGeometry (AFamilySPistonGeometryOpen)
Parameters
Relations
Applications


AFamilyREBearing
AFamilyREBearingSpecs (AFamilyREBearingSpecsOpen)
AFamilyREBearingSizing (AFamilyREBearingSizingOpen)
AFamilyREBearingGeometry (AFamilyREBearingGeometryOpen)
Parameters
Relations
Applications

Part Wizard


AFamilyRodEnd
AFamilyRodEndSpecs (AFamilyRodEndSpecsOpen)
AFamilyRodEndSizing (AFamilyRodEndSizingOpen)
AFamilyRodEndGeometry (AFamilyRodEndGeometryOpen)
Parameters
Relations
Applications


AFamilyRampWasher
AFamilyRampWasherSpecs (AFamilyRampWasherSpecsOpen)
AFamilyRampWasherSizing (AFamilyRampWasherSizingOpen)
AFamilyRampWasherGeometry (AFamilyRampWasherGeometryOpen)
Parameters
Relations
Applications

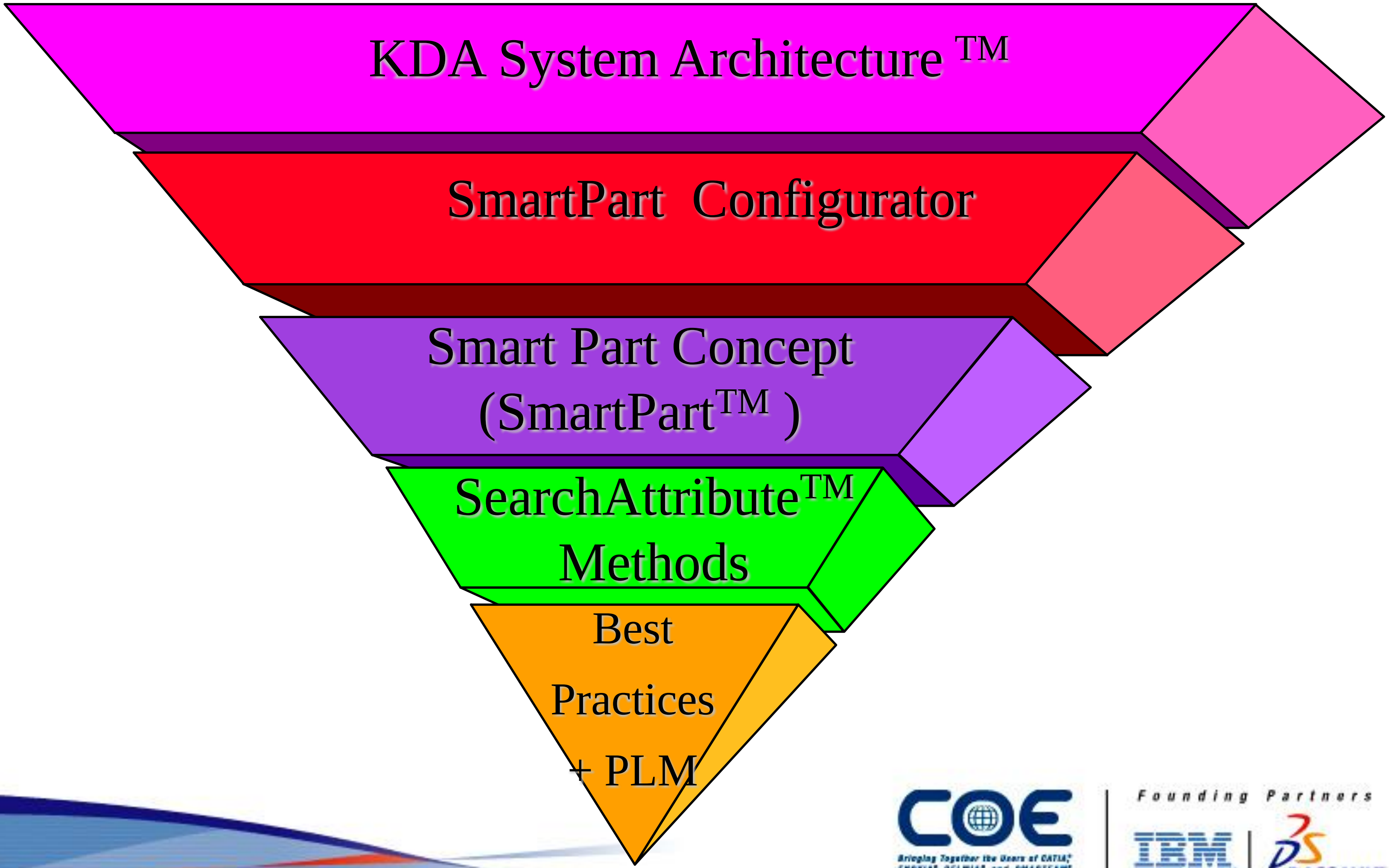
Why employ a "Part-independent" SmartPart Concept?

- With ***SmartPart,*** no broken links nightmare – one can plug and play.
- ***SmartPart*** concept enables concurrent engineering.
- Different groups of people can work on their individual tasks (Specs, Sizing, Geometry) independently, since you don't have to have the actual results, only provisions for receiving common inputs and outputs.
  - You don't have to edit or build any links. **Rules define the links for you.**
  - System matches the inputs and outputs at runtime based on rules already existing.

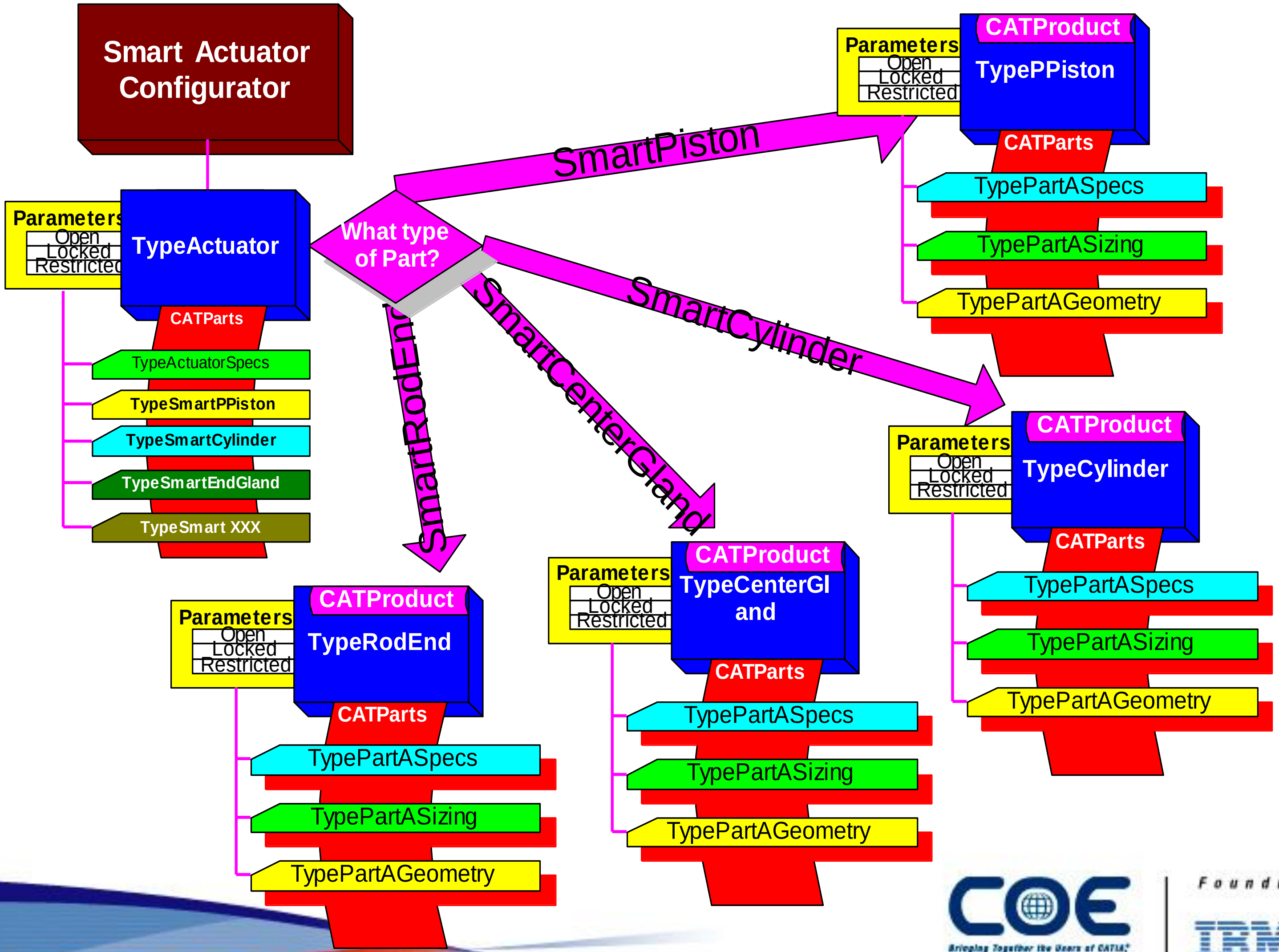
Why employ a “Part-independent” SmartParts Concept?

- If a new Sizing method is available, the old one can be replaced with this new one -- without modifying anything in the tree. The ***system*** architecture takes care of this interchange automatically.
- No limitations on the number of components of ***SmartPart*** .
- Using the concept one can easily design a 2-elements ***SmartPart*** or a 4-elements ***SmartPart*** .
- SmartPart Configurator is “Part –independent” and components are modular, interchangeable (plug and play).
- ***SmartPart*** definition is generic and “reusable.” (e.g., the same Actuator configurator is used to create 14 Actuator Smart Parts.

SmartPart is a part of Our KDA System Architecture

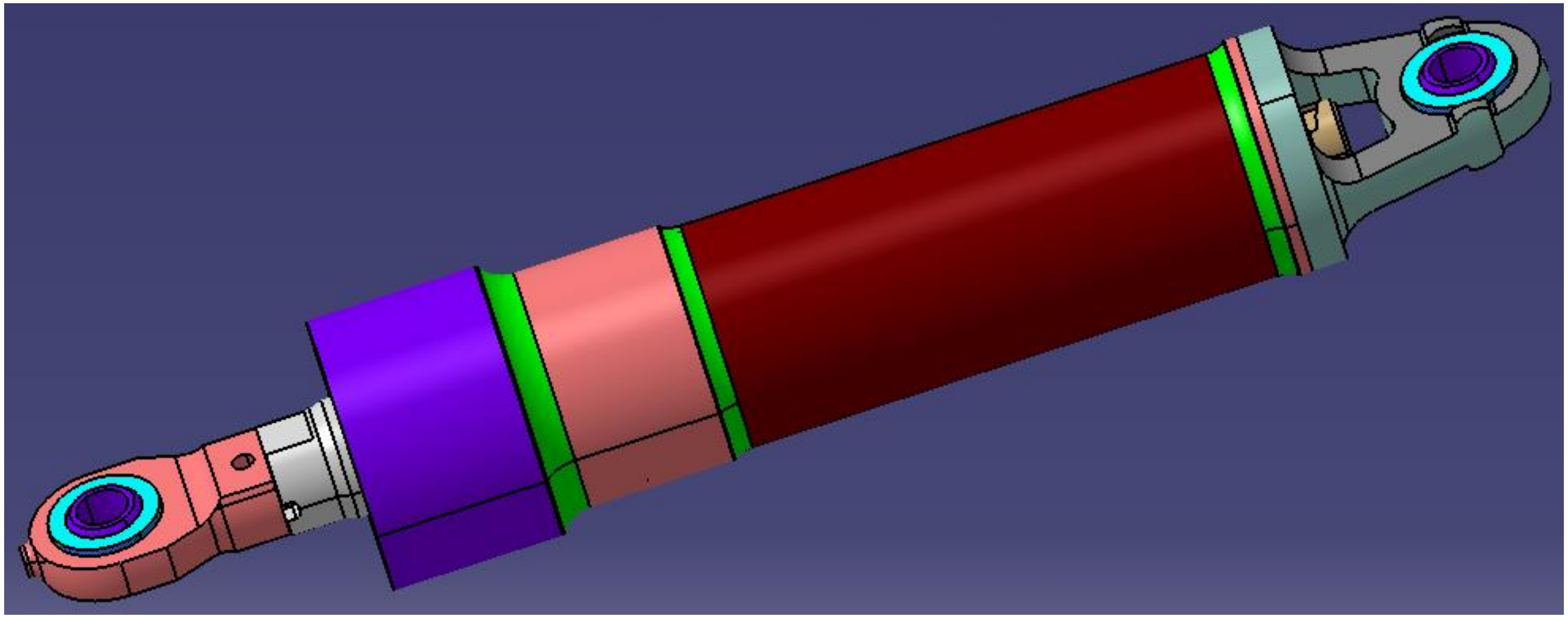

KDA System Architecture TM
SmartPart Configurator
Smart Part Concept (SmartPartTM)
SearchAttributeTM Methods
Best Practices + PLM

SmartPart Definition is Reusable?

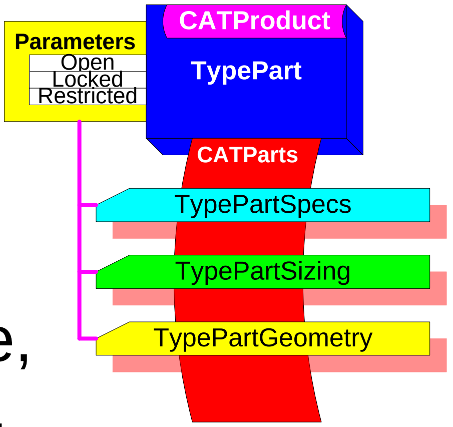


Another Major Benefits of SmartParts…

- For Creating Smart Assembly– Family of Actuators

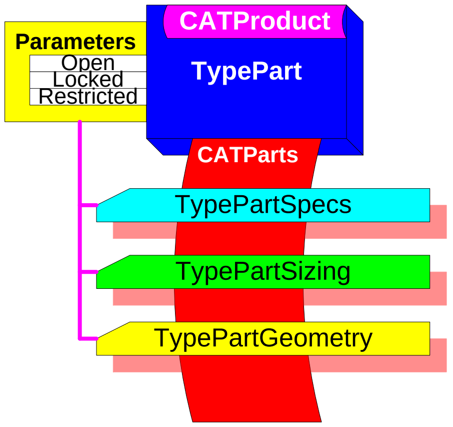


Our SmartPart Deployment Strategies



- Integrate design standards into “SmartPart” templates engineers could instantiate to expedite their design and analysis tasks
- Follow ***KDA Process*** (e.g., KDA architecture, product configurator, SmartPart configurator, etc.), to develop wizards to help engineers and designers come up with alternative designs quickly in early development phases based on historical knowledge and best practices.

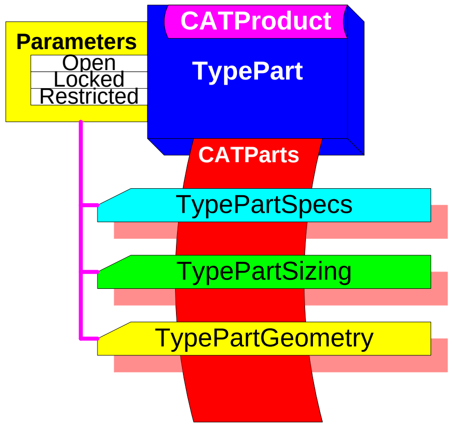
Concluding Remarks



- If components are defined, building SmartParts is a trivial exercise.
- Due to independence of the implicit matching method used for passing attribute-values across its component parts, the results are remarkable.
- Process is dynamic and assembly exhibits Parts' independence,
- SmartPart can freely accepts both sizing, geometrical (such as morphing) and topological (such additions and subtractions of parts) changes.

Concluding Remarks

- No breakage possibilities of any “links” exists since no “static links” were created or employed.
- Provide greater plug-and-play flexibilities for interchanging parameters, exchanging its features, or replacing its components.
- It also allows the various components of the SmartPart to be reconfigured and reused in either standalone mode or in combination with other solutions.



Thank You

bprasad@parker.com, jrogers@parker.com

Parker

www.parker.com