cadexchanger.medium.com /crash-course-on-cad-data-part-4-f2f2ba1e165e

Crash course on CAD data. Part 4 - CAD Exchanger - Medium

CAD Exchanger ∷ 2021/6/10

Previous parts

Part 1. How did this all begin?

Part 2. How is 3D data presented in CAD?

Part 3. Precise geometry and tessellation. B-Reps and meshes

CAD model structure. Parts and assemblies

Up until now we've been talking about matters concerning only a single physical object. Such an object in CAD terminology is called *part*. When working with more complex models, consisting of multiple parts we have to deal with groups or even hierarchies of nested groups of parts. These groups are called *assemblies* in the CAD world and the hierarchy of parts and assemblies is sometimes called *product structure*.

Let's take a look at the following piston-crankshaft assembly model in CAD Exchanger Cloud:

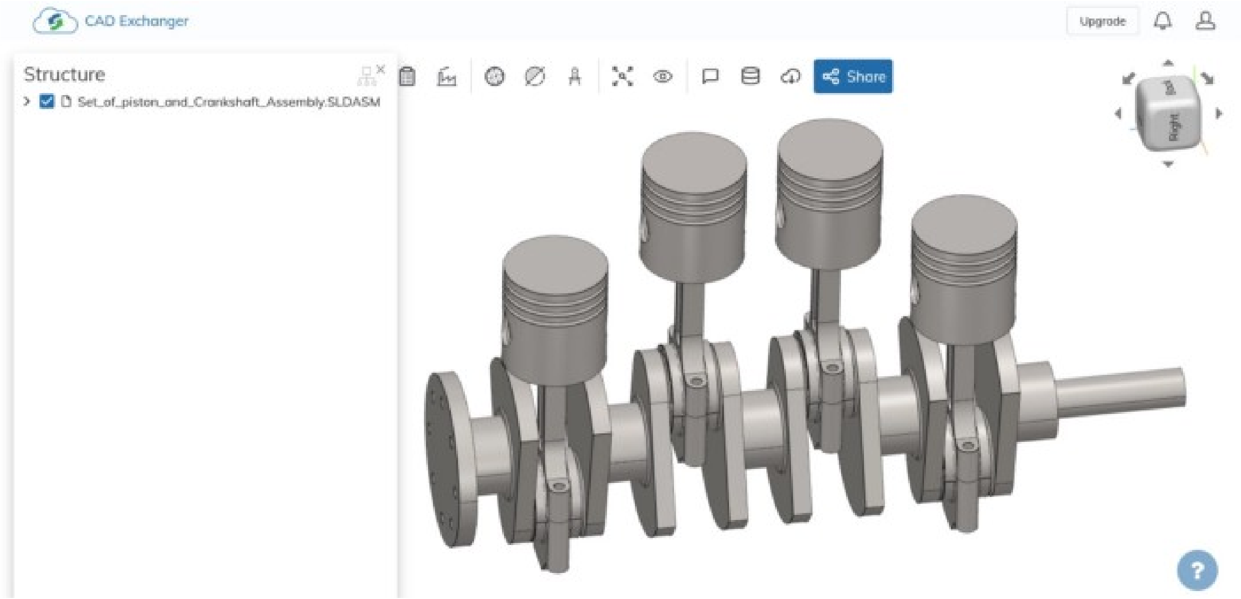
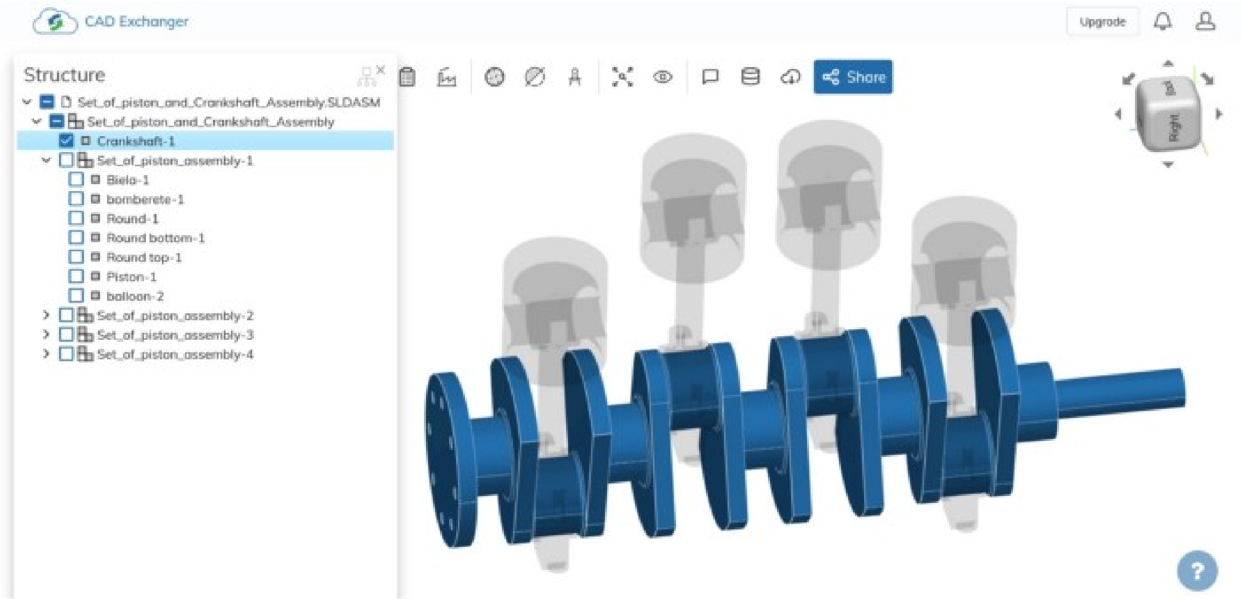


Fig. 1. Piston-and-crankshaft assembly

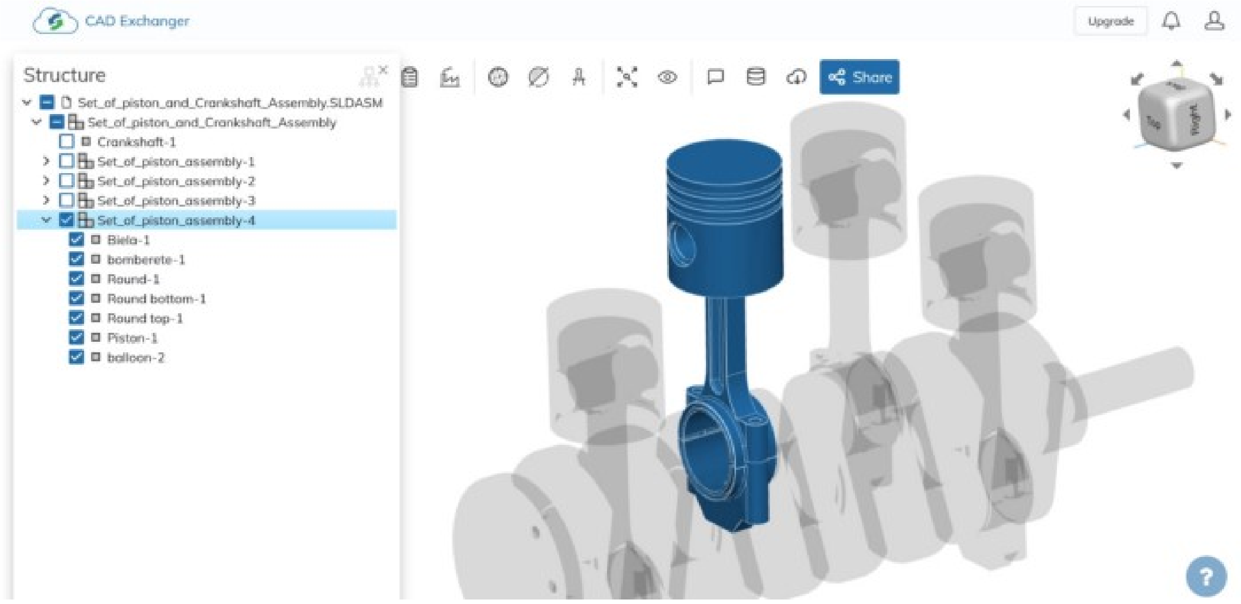
If we expand the tree in the Structure view we will see the parts and assemblies comprising the model. The entire model is contained in a single assembly "Set_of_piston_and_Crankshaft_Assembly" (see Fig.1). The crankshaft is a single part (see Fig.2), while each piston-connecting rod combination is a whole assembly in itself — "Set_of_piston_assembly-1" (see Fig.3). In turn, it contains its elements as parts — piston in "Piston-1", connecting rod in "Biela-1" and "bomberete-1", as well as a few other elements (see Fig.3 and Fig.4).



Structure

- ■ Set_of_piston_and_Crankshaft_Assembly.SLDASM
  - ■ Set_of_piston_and_Crankshaft_Assembly
    - ☑ Crankshaft-1
    - □ Set_of_piston_assembly-1
      - □ Biela-1
      - □ bomberete-1
      - □ Round-1
      - □ Round bottom-1
      - □ Round top-1
      - □ Piston-1
      - □ balloon-2
    - □ Set_of_piston_assembly-2
    - □ Set_of_piston_assembly-3
    - □ Set_of_piston_assembly-4

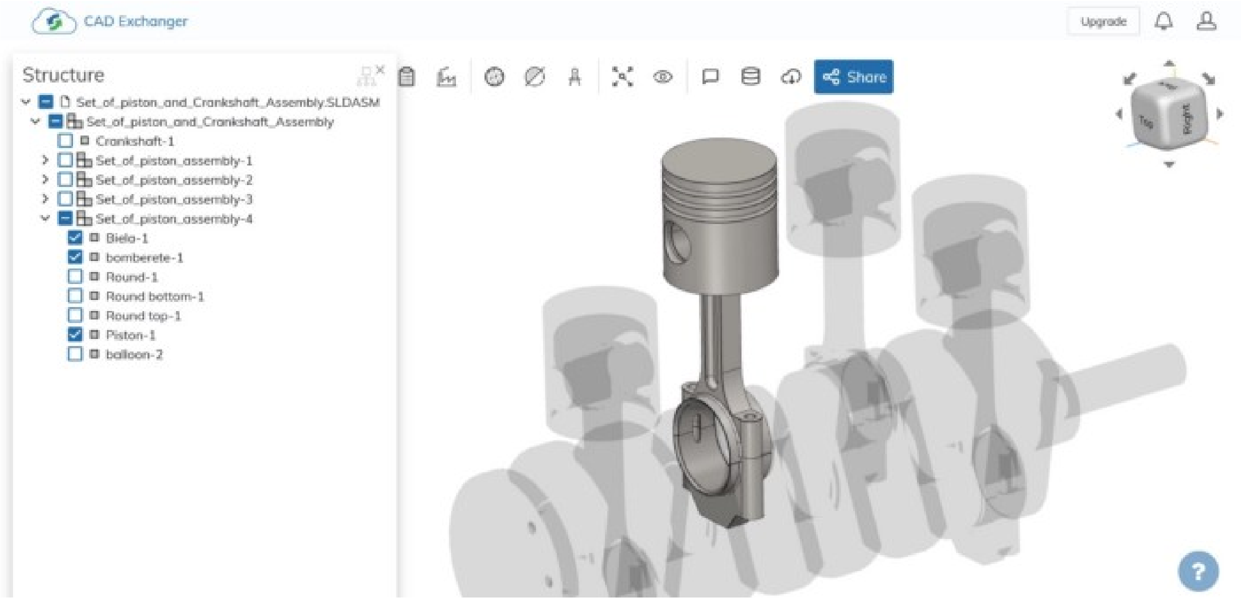
Fig. 2. Crankshaft is a single part



Structure

- ■ Set_of_piston_and_Crankshaft_Assembly.SLDASM
  - ■ Set_of_piston_and_Crankshaft_Assembly
    - □ Crankshaft-1
    - □ Set_of_piston_assembly-1
    - □ Set_of_piston_assembly-2
    - □ Set_of_piston_assembly-3
    - ☑ Set_of_piston_assembly-4
      - ☑ Biela-1
      - ☑ bomberete-1
      - ☑ Round-1
      - ☑ Round bottom-1
      - ☑ Round top-1
      - ☑ Piston-1
      - ☑ balloon-2

Fig. 3. “Piston-connecting rod” group is a sub-assembly



Structure

- ■ Set_of_piston_and_Crankshaft_Assembly.SLDASM
  - ■ Set_of_piston_and_Crankshaft_Assembly
    - □ Crankshaft-1
    - □ Set_of_piston_assembly-1
    - □ Set_of_piston_assembly-2
    - □ Set_of_piston_assembly-3
    - ■ Set_of_piston_assembly-4
      - ☑ Biela-1
      - ☑ bomberete-1
      - □ Round-1
      - □ Round bottom-1
      - □ Round top-1
      - ☑ Piston-1
      - □ balloon-2

Fig. 4. “Piston-connecting rod” sub-assembly with some hidden parts

At this point you might be asking: why even bother with defining all these groups and sub-groups? It all seems very tedious, is there a benefit to splitting your model in such a way? Indeed, the benefits might not become obvious until we have to work with truly large designs. I can point out two specific advantages:

- **Complexity management.** When working on a detailed design of a complicated machine (e.g. internal combustion engine), being able to consider only a small portion of the model during a particular design session is important both conceptually (there’s only so much context a human brain can hold) and practically (wielding a massive detailed model on a screen at all times is hard on the hardware).
- **Component reuse.** Over time a design firm is going to accumulate a library of common parts which can be simply dropped into assembly in the next project. CAD systems generally supply libraries with some universal standard parts (most commonly fasteners).

How does this structure arise during the design? Normally it would be a result of an iterative top-down design approach. The model is originally a single part, but as the design becomes more and more detailed, its portions are split off into separate parts and re-inserted into the whole model as components of some assembly.

CAD systems normally provide separate sets of tools for part and assembly modeling. While part modeling provides tools that change the geometry of the part (covered in Part 3 of this series), assembly modeling tools are all about connecting parts and subassemblies together, placing them in relation to each other. While it’s possible to meticulously drag the models in 3D space trying to put them in just the right spots, a better (and standard in the CAD world) approach is to *use constraints (mates)* — conditions which limit the relative motion of assembly components. Common basic constraints include:

- Distance and angle
- Coincidence
- Concentricity
- Perpendicularity or parallelism
- Tangency

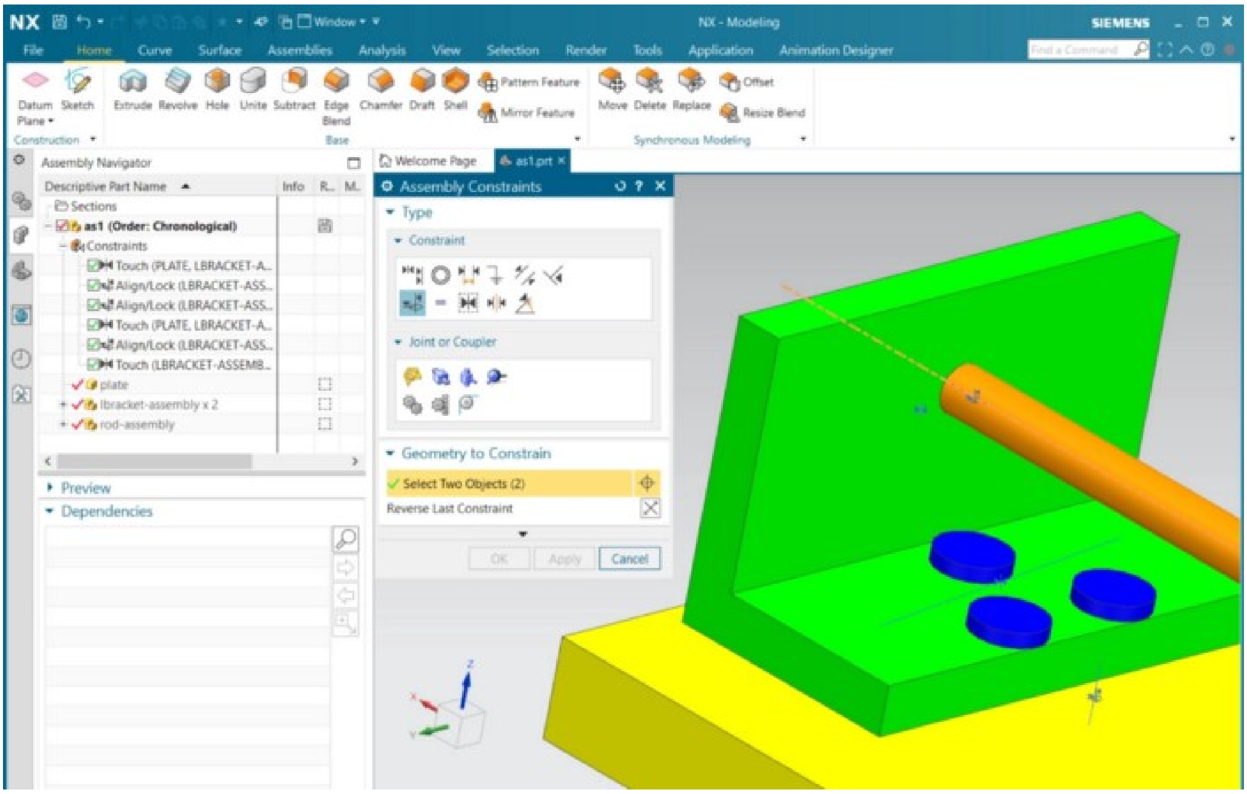
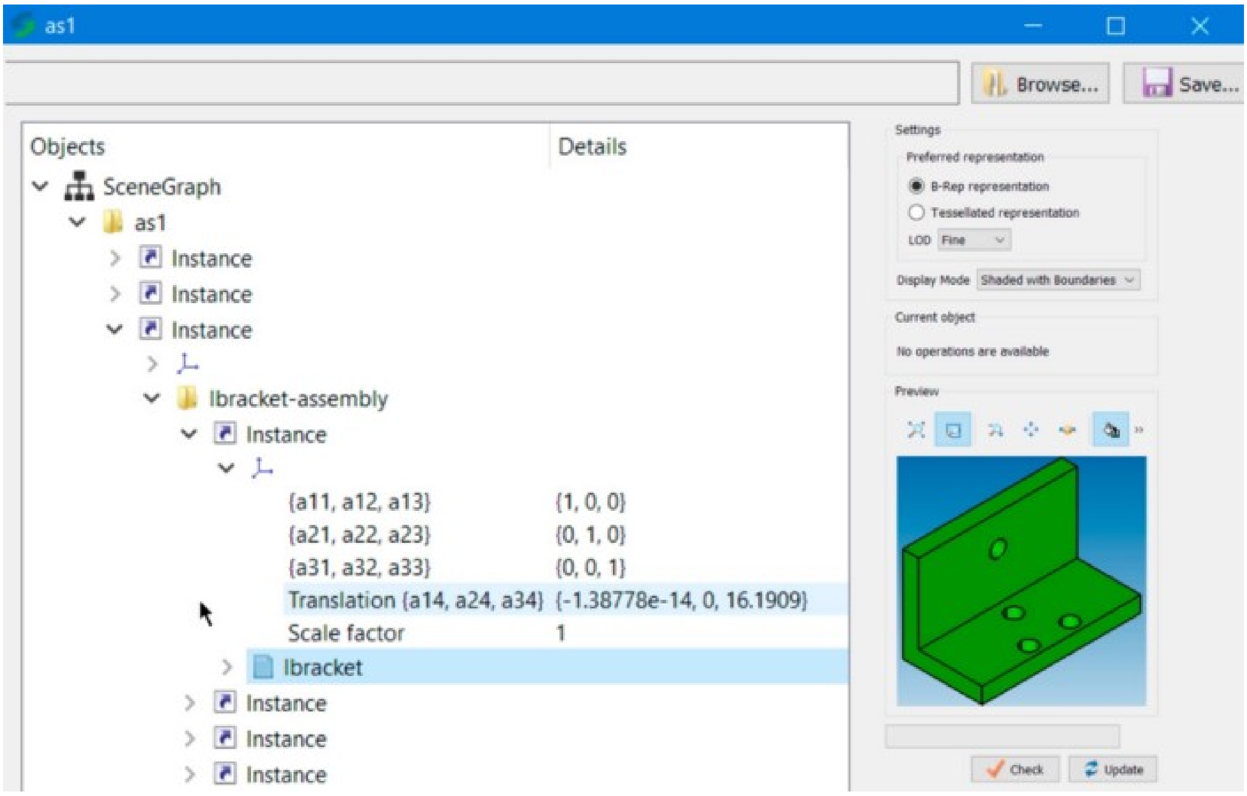


Fig. 5. Applying Align/Lock constraint in Siemens NX to place the orange rod in the hole in the green l-bracket

There are also other, more advanced types of constraints, for example those related to systems of gears. Once the constraints are all applied, the actual positions of the components are then computed automatically by a CAD system component known as *constraint solver*. This step is needed because ultimately the assembly has to be displayed on screen and the part triangles need to be shifted into their actual positions in order to be rendered correctly. Constraints themselves are not easy to use for this task, but the transformations computed by constraint solver are just perfect.



Objects	Details
SceneGraph	
as1	
Instance	
Instance	
Instance	
lbracket-assembly	
Instance	
{a11, a12, a13}	{1, 0, 0}
{a21, a22, a23}	{0, 1, 0}
{a31, a32, a33}	{0, 0, 1}
Translation {a14, a24, a34}	{-1.38778e-14, 0, 16.1909}
Scale factor	1
lbracket	
Instance	
Instance	
Instance	

Settings
Preferred representation
● B-Rep representation
○ Tessellated representation
LOD Fine
Display Mode Shaded with Boundaries
Current object
No operations are available
Preview

Fig. 6. The resulting transformation matrix of an I-bracket part in the assembly above

Since parts and assemblies are so vital to CAD designs, naturally, most CAD formats also have ways to represent parts and assemblies. There are two approaches to doing that. Some formats have a single (monolithic) file for the entire model and all the assemblies and parts are described within that file. Another approach (especially favored by industrial CAD systems) is to place each part and assembly into a separate file on disk.

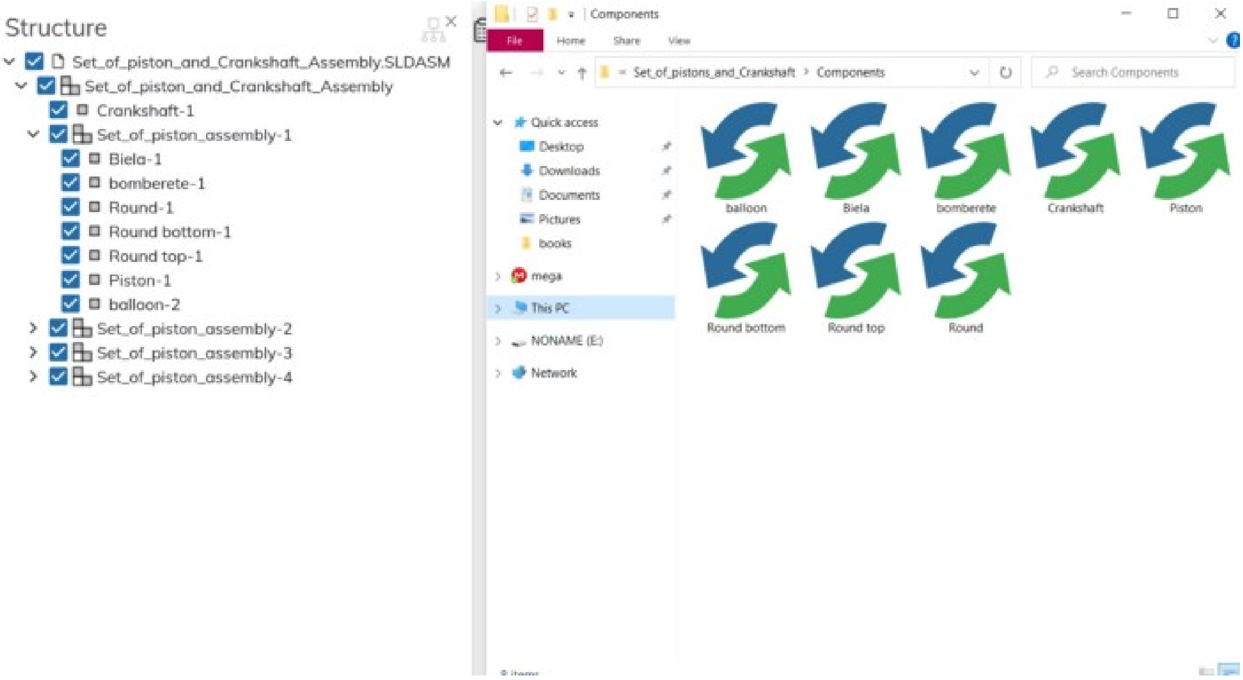


Fig. 7. SolidWorks model structure in CAD Exchanger Cloud and SolidWorks files it's composed of

The choice between these approaches represents a trade-off: separate files enable granular data storage — no need to load up the entire assembly when you only want to work on one part. On the other hand, separate files can get lost in transmission, they are misplaced and can't be loaded into assembly — these are the issues you don't have with monolithic-file models. Examples of multi-file formats are native formats of various CAD systems — SolidWorks, CATIA, PTC Creo and Siemens NX. Examples of monolithic-file formats are IGES, Parasolid-XT and ACIS-SAT. There are also formats that can work both ways, e.g. STEP and JT.

This part concludes our study of the 2 cornerstones of CAD data — geometry and structure representations. However, CAD models usually contain other information — and sometimes lots of it. Stay tuned and we'll cover the remaining types in the future installments.

Related readings:

CAD Exchanger SDK: traversing the hierarchy of assemblies and parts

CAD Exchanger SDK examples: exploring assembly (BOM), creating assembly

Next parts:

Part 5. Metadata

Part 6. Visual attributes

Originally published at https://cadexchanger.com on April 23, 2021.