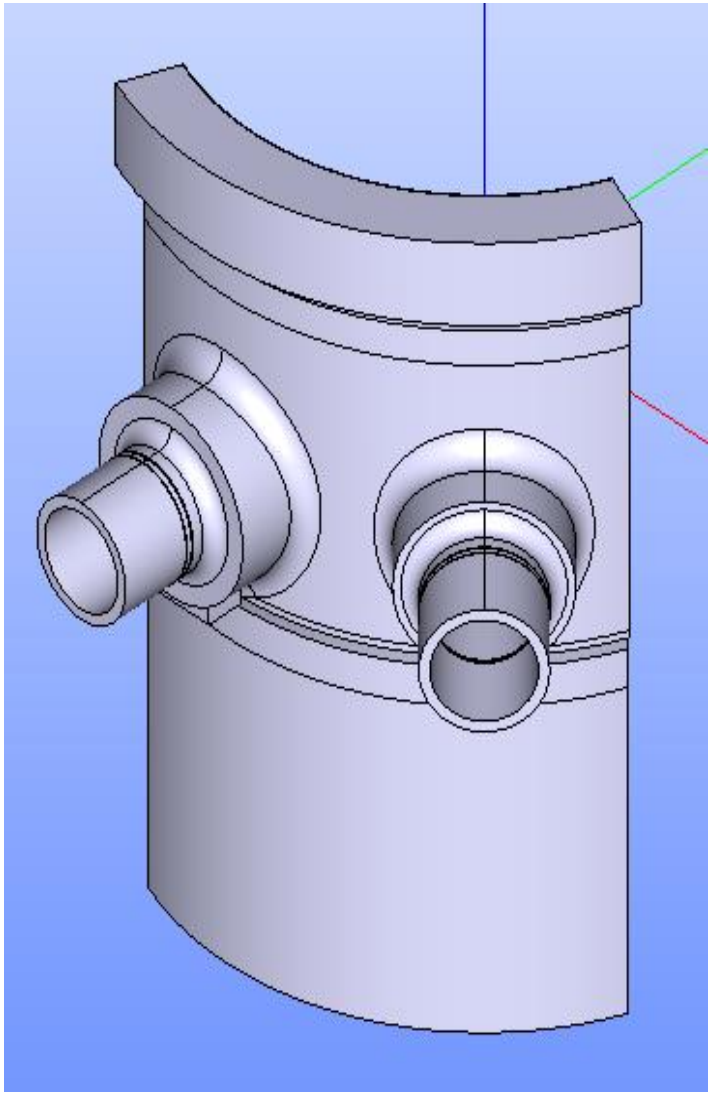
A short introduction to SHAPER



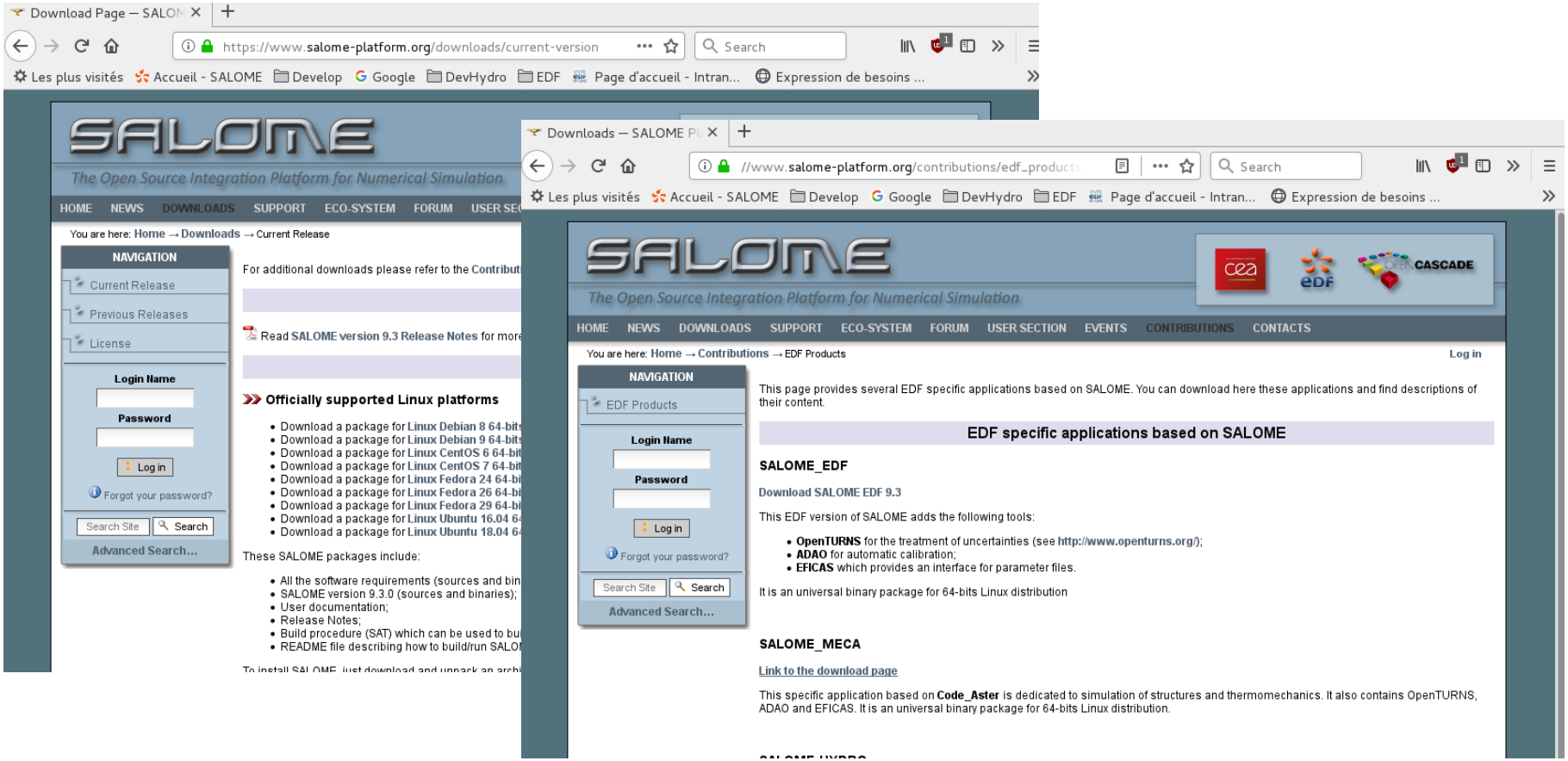
Raphaël MARC
EDF lab Saclay
Département PERICLES, groupe i2C

Shaper : Official Realease

- **In version 9.3 of SALOME made available on May 15, 2019 here:**
  - Generic Platform, Linux(es) and Windows:
    - https://www.salome-platform.org/downloads/current-version
  - EDF contributions (Debian + Ubuntu Versions, Mechanics, CFD, Hydraulics…):
    - https://www.salome-platform.org/contributions/edf_products
- **The major new features of SALOME 9.3 version are:**
  - The new CAD parametric and variational Modeler SHAPER,
  - The possibility to realize educational visualizations thanks to OSPRay available in ParaView
  - A new module YDEFX for parametric studies
  - Python 3

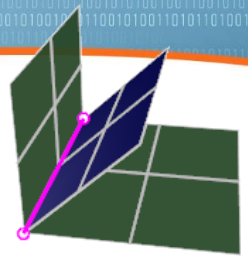
Installation

- https://www.salome-platform.org/downloads/current-version
- https://www.salome-platform.org/contributions/edf_products



Shaper : Why ?

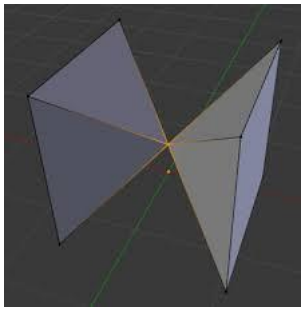
- **Context:**
  - SALOME: an open source integration platform for numerical simulations
    - Designed for fields Physics (mainly finites elements)
    - CAD, meshing, visualization, computation workflows, HPC Clusters…
  - Co-developed by EDF & CEA with OpenCascade
  - EDF-CEA shared analysis: the code, architecture, ergonomics of the old CAD module (GEOM) are obsolete
- **Major stakes:**
  - More efficient drawing of CAD models (gain in studies efficiency)
  - maintain the competence in CAD (EDF, CEA, OpenCascade)
  - ➔ development started in 2014

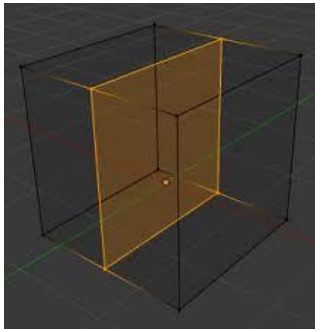


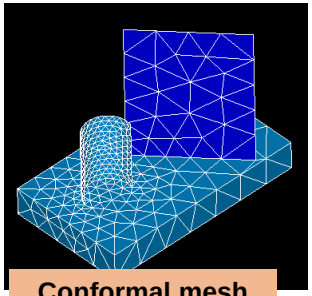
Shaper : Why ?

Why develop a new CAD modeler ourselves?

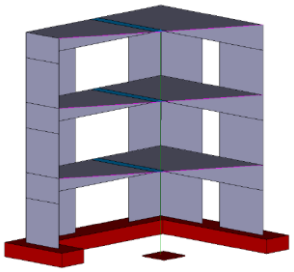
- To stay in complete control of our simulation chain
- ➔ meet the fine needs of R & D and the needs of engineering:
  - ➔ Integration of specific solvers,
  - ➔ Development of domain specific applications
- A CAD modeler dedicated to simulation:
  - Group of shapes (vertices, edges, faces, solids etc.)
  - Conformity of shapes:
    - non manifold geometries (ex: an edged shared by more than 2 faces)
    - partition
    - multi-dimensionnal connected geometries
    - Conformal meshable geometries
  - Python scripting

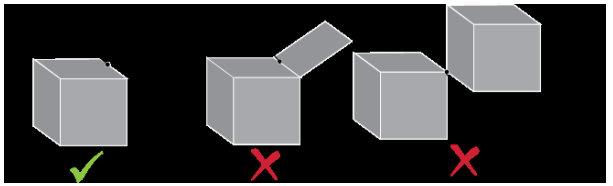






Conformal mesh



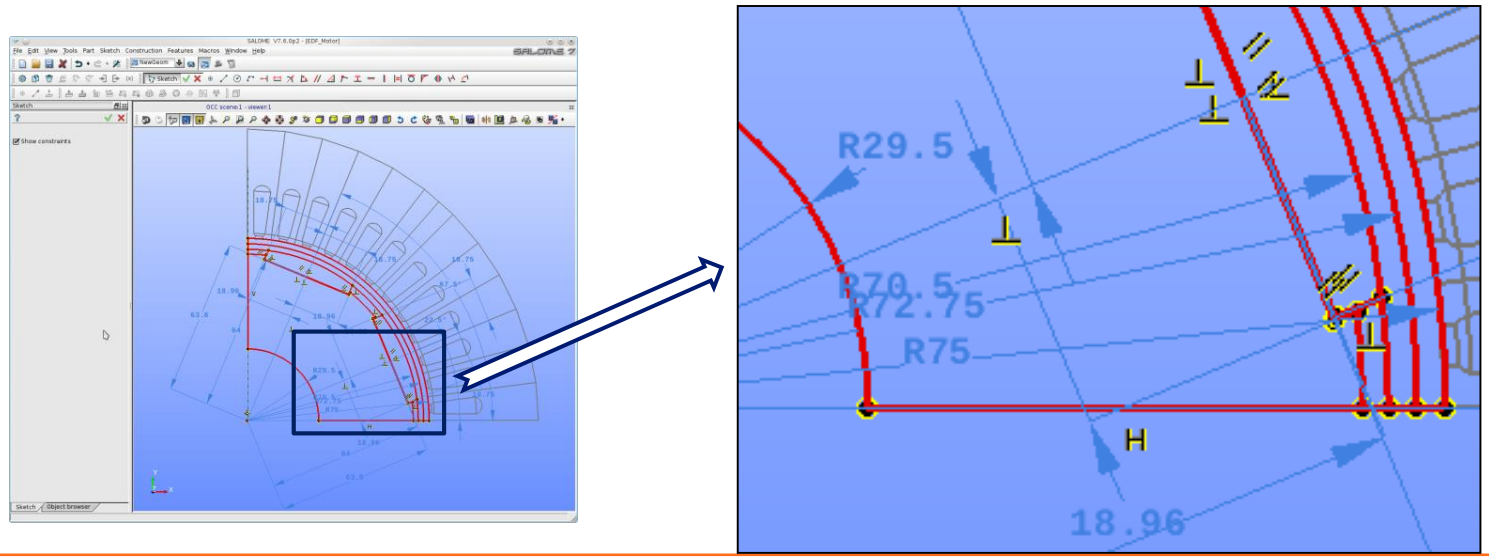


CEA ENISTAT model

Shaper: Main Principles

- Parametric and variational
- 3D parts assembly
- promote drawing with the HMI
- CAD for simulation

- **1) Interactive design on 2D sketches (variational) :**
  - Similar to industrial drafting practices
  - Application of constraints: parallelism, distance, coincidence ...



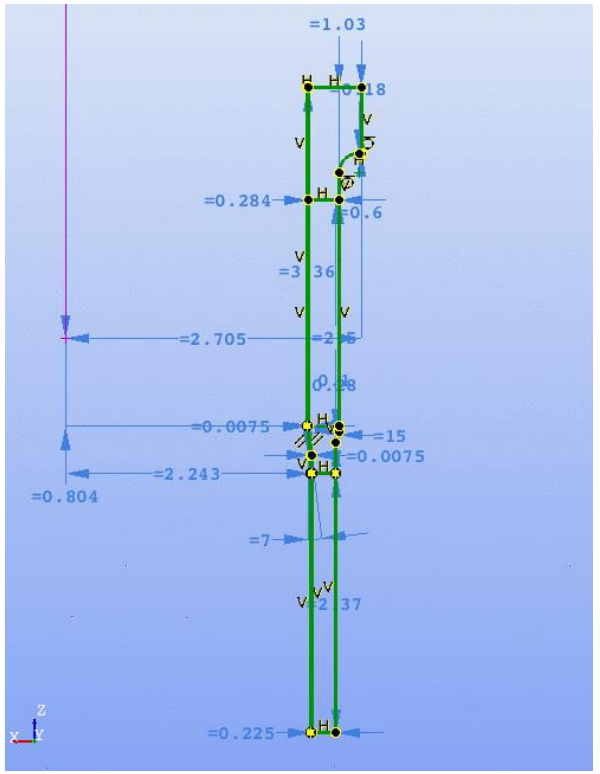
Shaper: Main Principles

2) Volume forming:

- by extrusion, revolution, sweeping
- Then Boolean operations (CSG)…

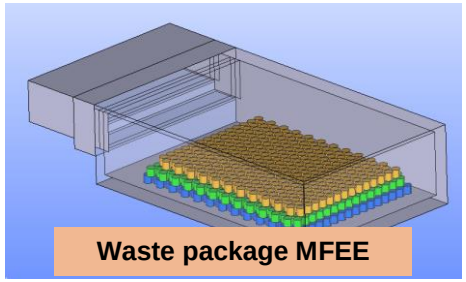
Parametric:

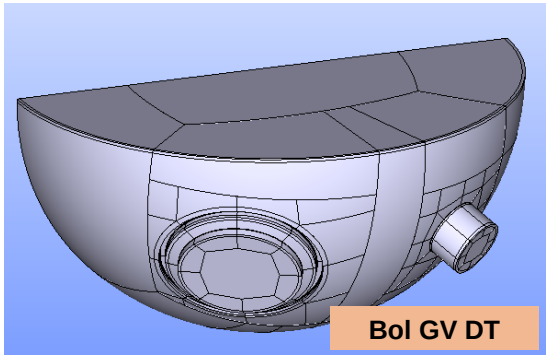
- Each parameter can be edited and modified
- The shape is automatically updated

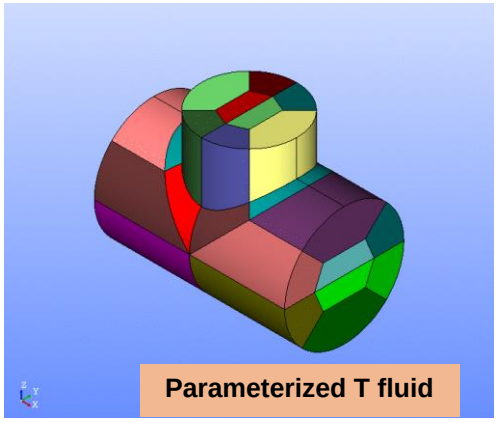


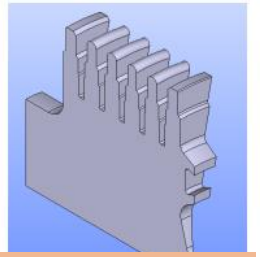
HOW? Piloting developments :

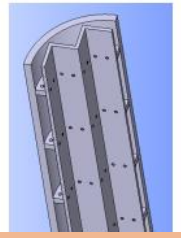
- By needs (8 users clubs)
- With target models (EDF)
- Ergonomics inspired by SolidWorks and FreeCAD

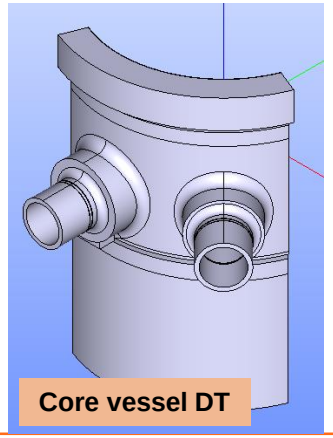

Waste package MFEE

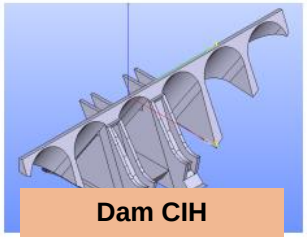

Bol GV DT

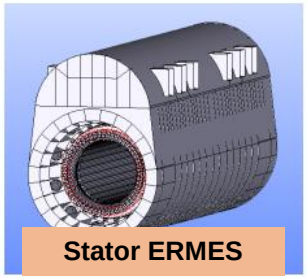

Parameterized T fluid

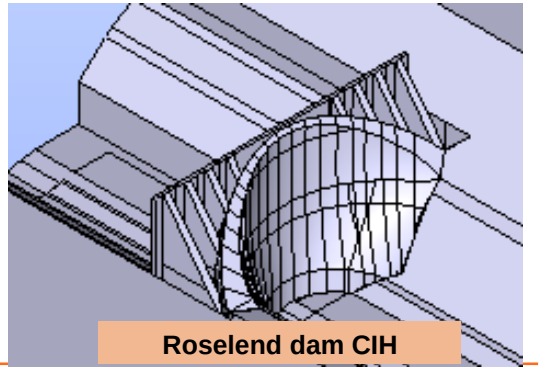

Attache ailette DT


Cloison internes cuve MFEE


Core vessel DT


Dam CIH


Stator ERMES


Roselend dam CIH

HOW much?

Cost:

EDF: 0,5 eng.year on average since 2014.

CEA

OpenCascade:

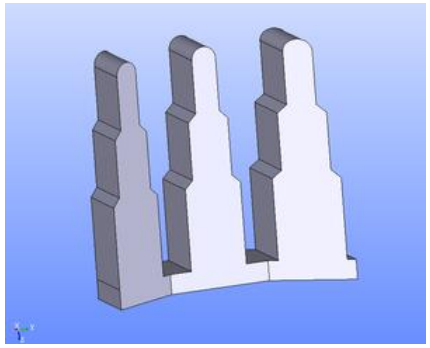
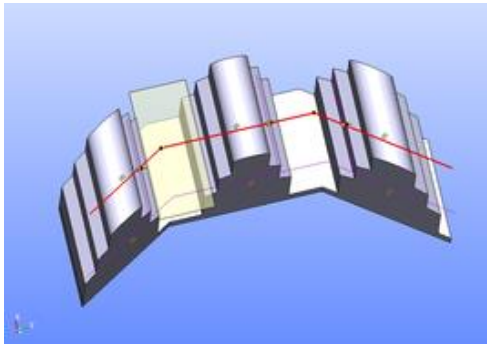
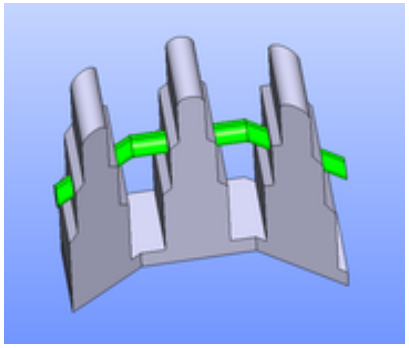
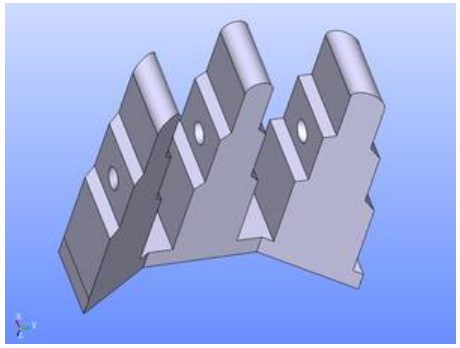
- Pre study 2013 : 6 eng.month
- Development 2014-2019 : 135 eng.month (2700 man-hours)
  - Specification: 5 eng.month
  - Debug : …

Number of lines of code:

- 193 000 lines of C++
- 28 500 lines of Python
- 1 500 C++ classes

geom: differences, recovery (of old studies)

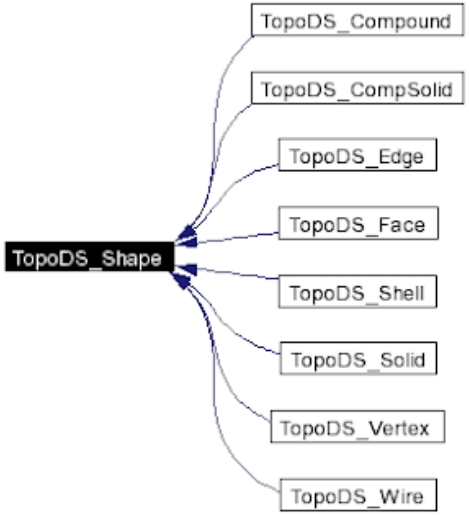
- **Differences:**
  - GEOM: neither parametric nor variational
  - GEOM: the drawing with the HMI not very ergonomic
    - ➔ the use of Python scripting with geompy is very often needed.
  - SHAPER: the goal is to allow to do as much as possible interactively with the HMI, without programming.
  - The groups creation functions in SHAPER are still a bit limited: to be completed in the next versions
- **No recovery of GEOM Python scripts or HDF templates but:**
  - Import of GEOM models in SHAPER
  - Possibility of modification / addition. For instance, below:

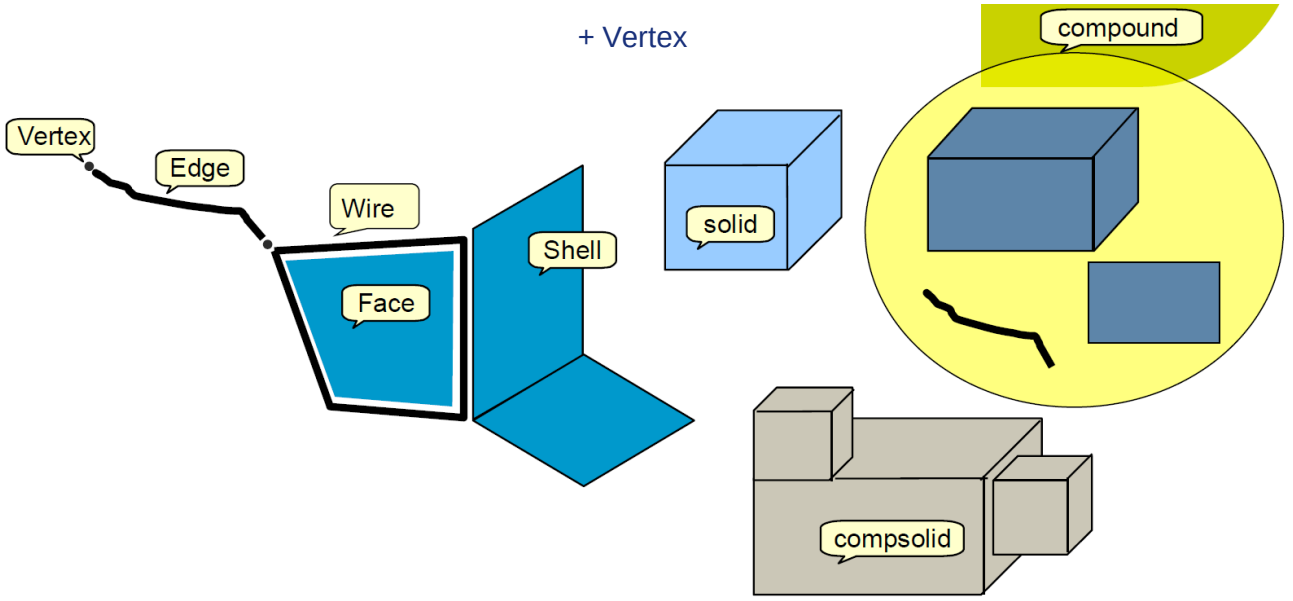





CAD vocabulary

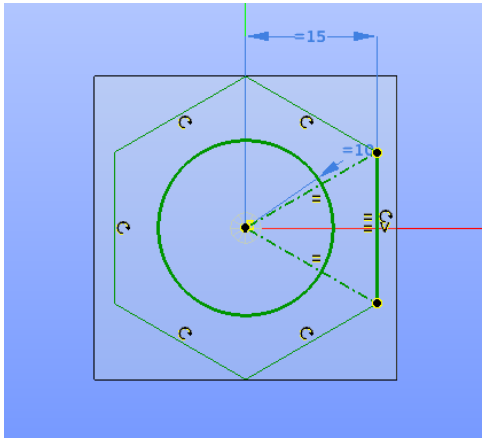
- OB, Object browser
- Part
- Partset: parts assembly
- Sketch: 2D drawing
- Shape
- hierarchy

+ Compound
  + <any type of shapes>
+ Compsolid
  + Solid
    + Shell
      + Face
        + Wire
          + Edge
            + Vertex

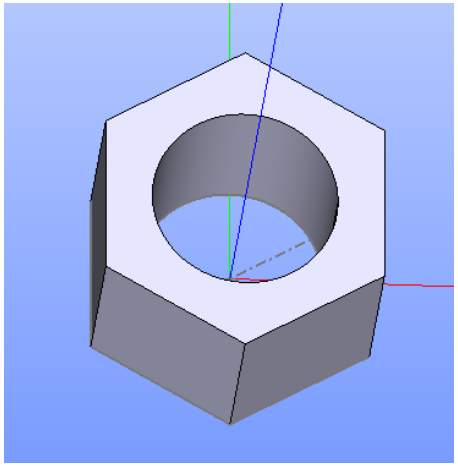




SKETCH – constraints – cotation: hands-on



- **SKETCH**
  - equilateral triangle: draw, constraints and dimension (h = 15)
  - angular copy of the opposite side to the origin
    - 6 rotations
  - Circle, select origin, define the radius (r = 10)

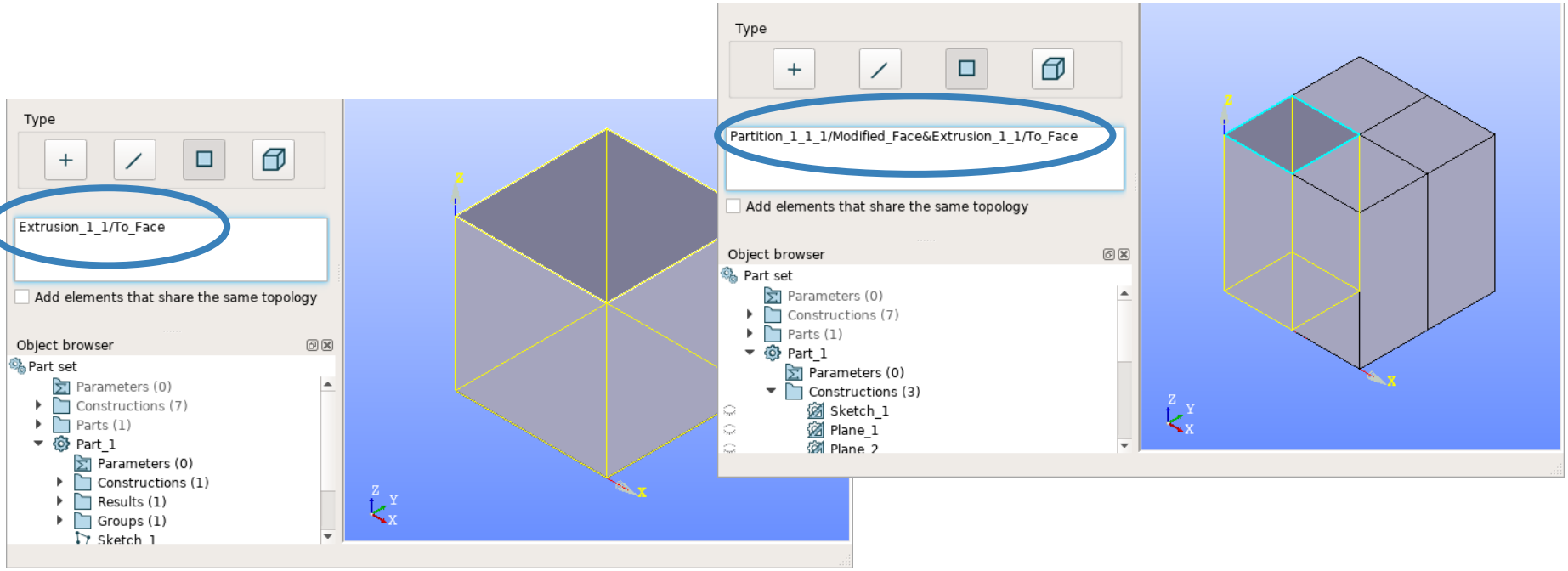


- **VOLUME**
  - Extrusion (h = 20)

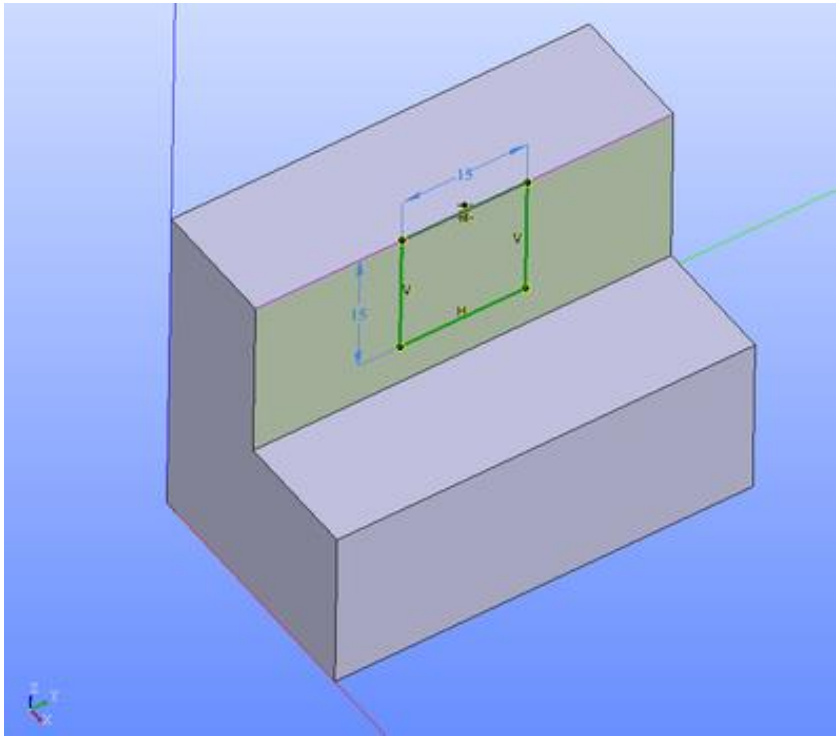
Naming: the base of parameterization

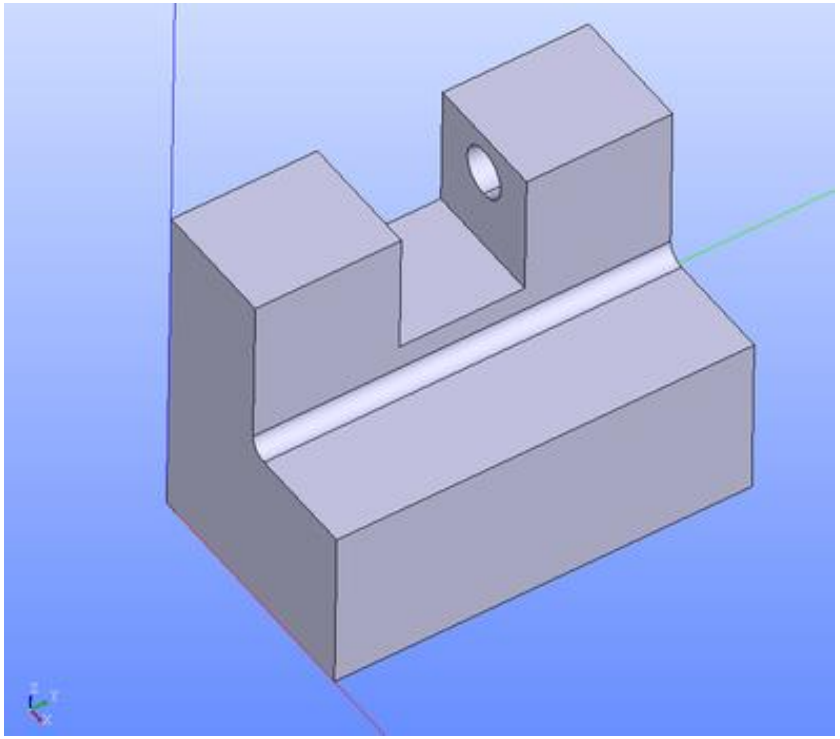
Naming Algorithm:

- Robust and stable naming of shapes.
- To allow to identify and follow a shape along the construction steps (in spite of its geometric or topologic modifications)

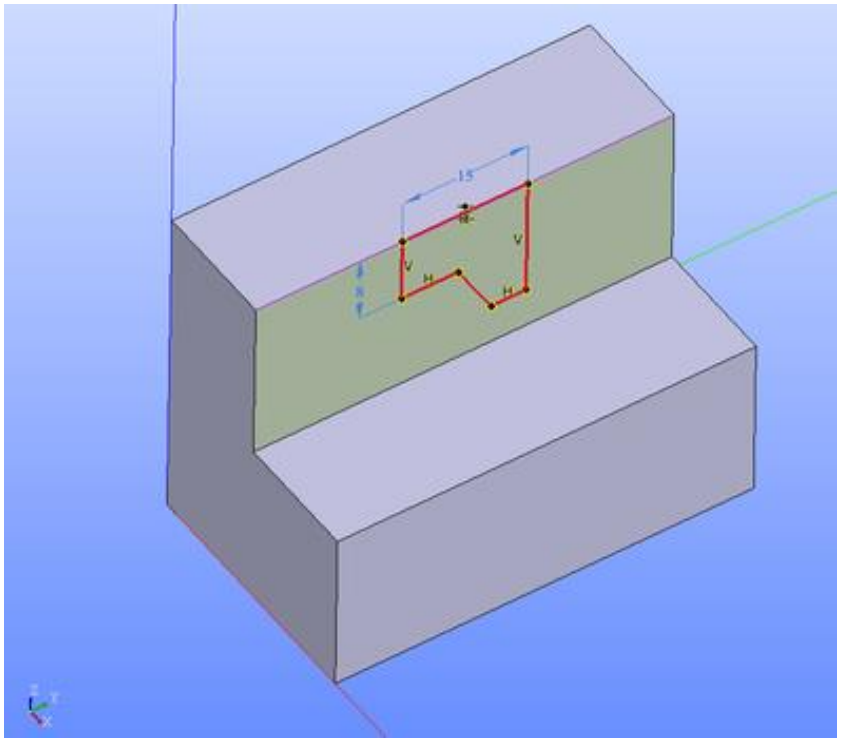


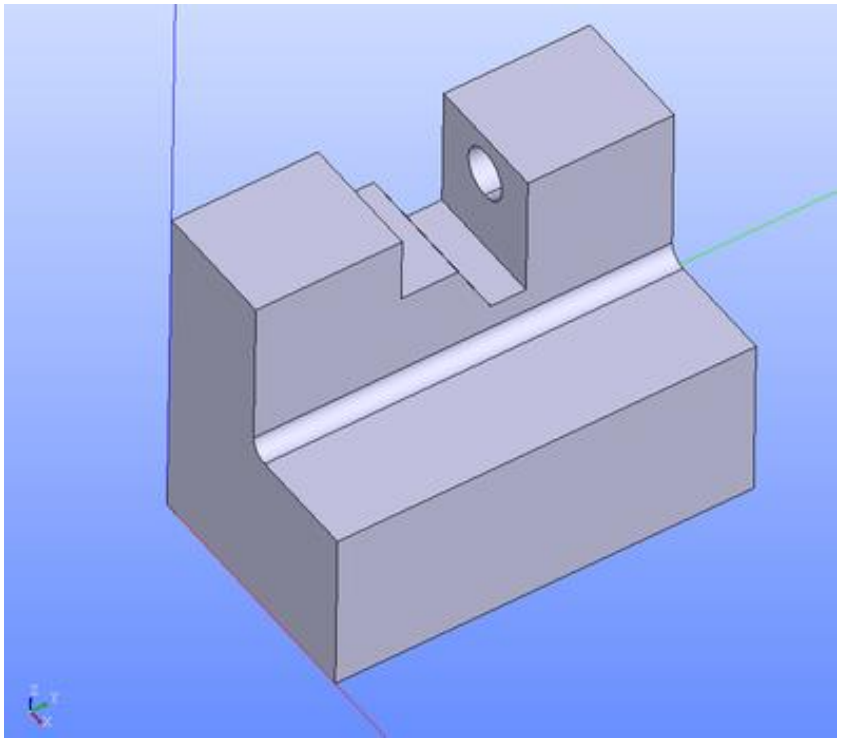
Naming effect on parameterization: 1





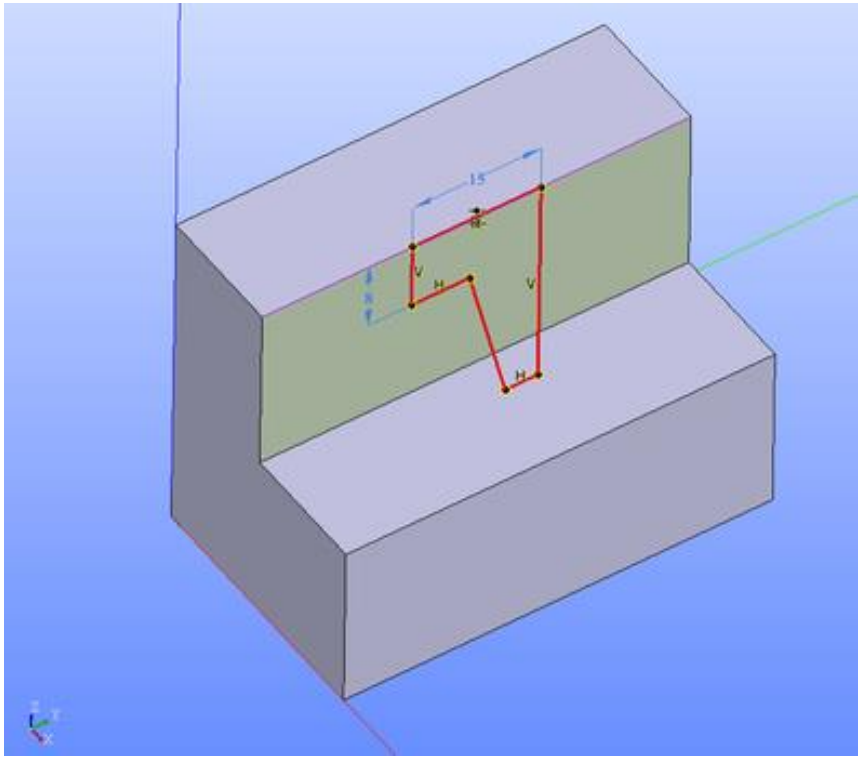
Naming effect on parameterization: 2

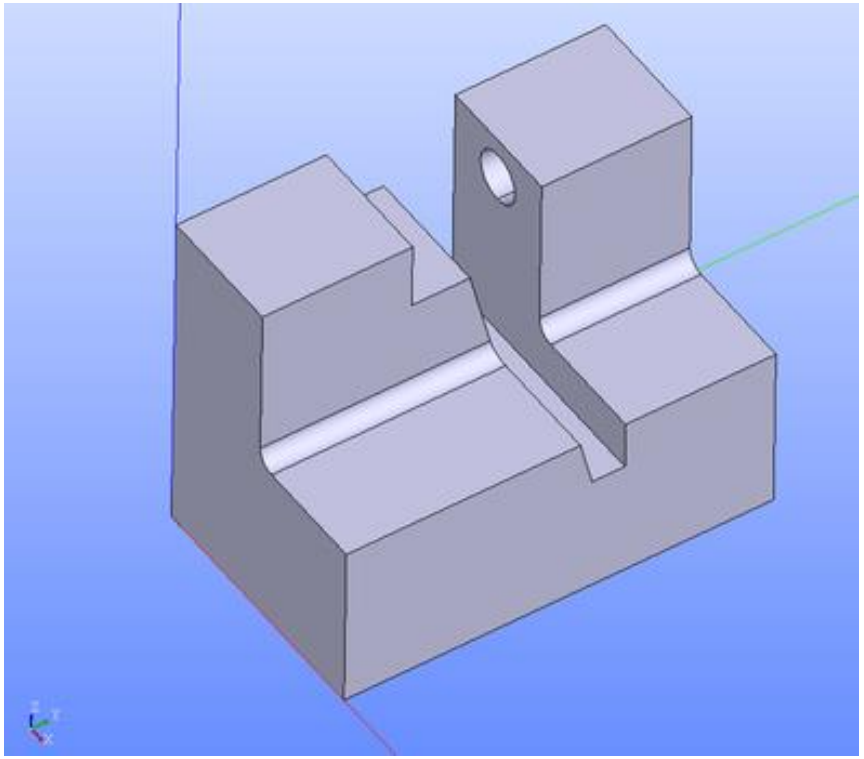




Naming effect on parameterization: 3

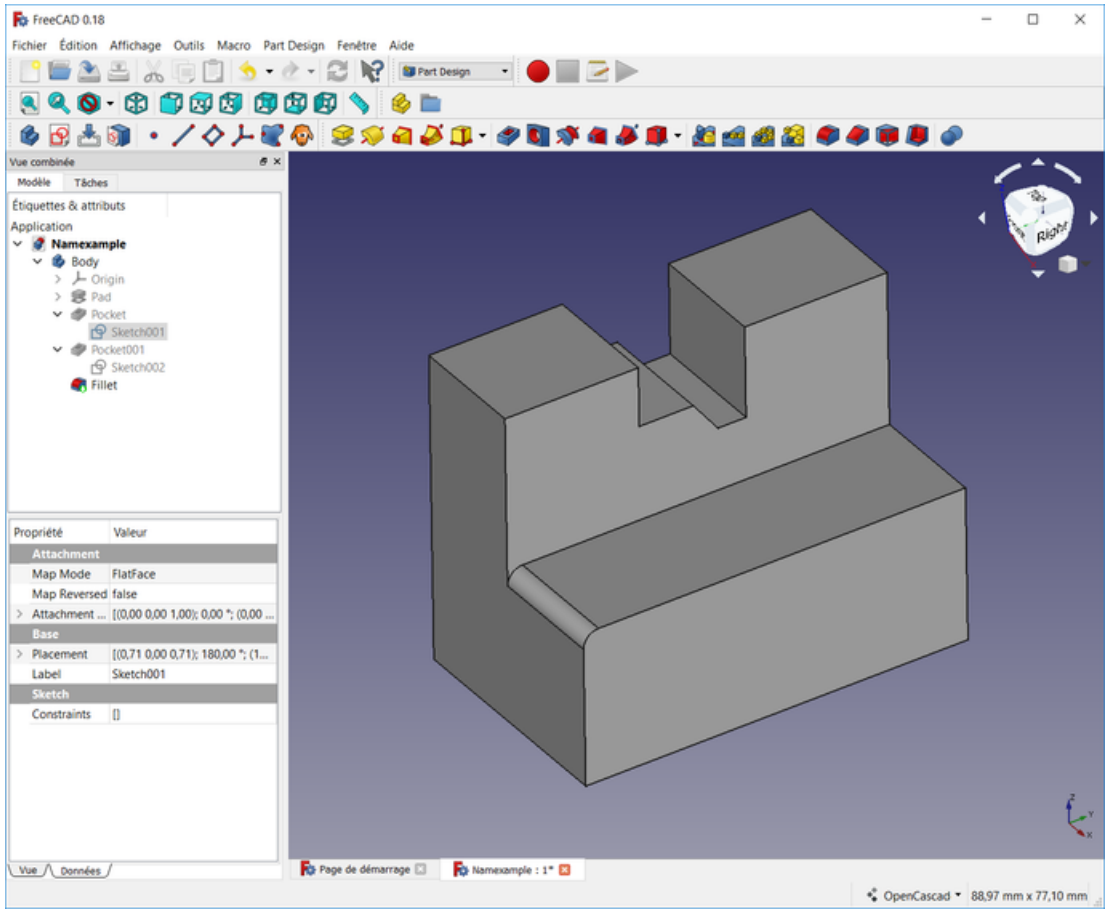
- The hole and the fillet are preserved despite the topological modification of the sketch created for the extrusion / cut.



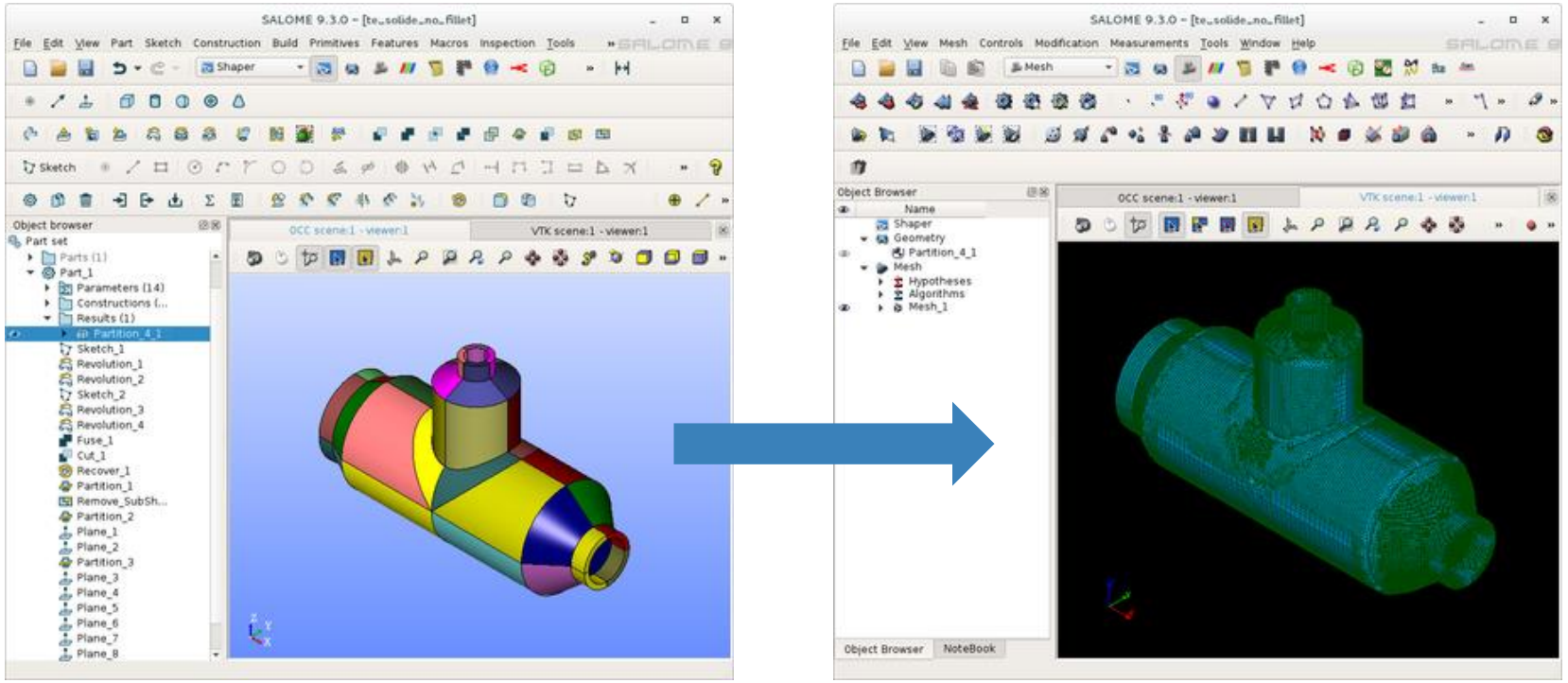


The same example in freecad…

… does not work:



Meshing with groups



Python Dump

- **Part Menu ➔ a specific SHAPER Dump** which generates:
  - A file <filename>.py ➔ « topologic » dump using the naming
  - A file <filename>_geo.py ➔ geometric dump using coordinates

Topologic dump:

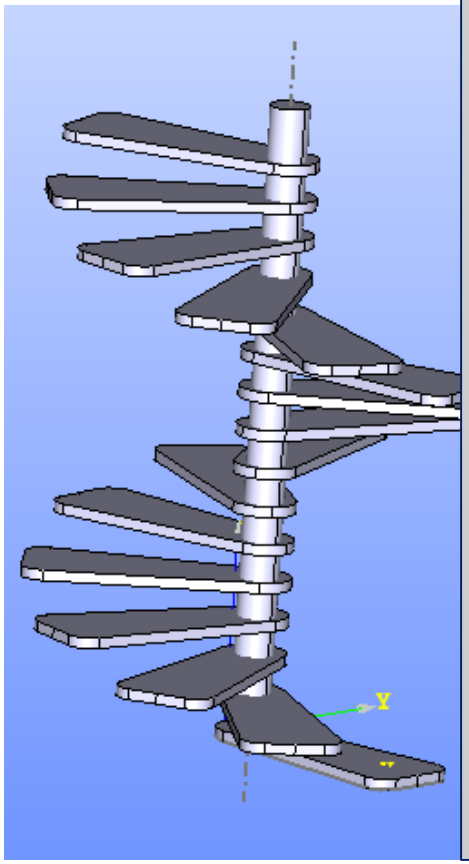
```
Sketch_1 = model.addSketch(Part_1_doc, model.selection("FACE", "Box_1_1/Top"))
SketchProjection_1 = Sketch_1.addProjection(model.selection("VERTEX",
"[Box_1_1/Back][Box_1_1/Left][Box_1_1/Top]"), False)
```

Geometric dump:

```
Sketch_1 = model.addSketch(Part_1_doc, model.selection("FACE", (5, 5, 5) ))
SketchProjection_1 = Sketch_1.addProjection(model.selection("VERTEX", (0, 0, 5) ),
False)
```

- **File Menu ➔ Dump** : dump of all the SALOME modules, SHAPER included (topologic dump)
- Upward Compatibility (for Python scripts and HDFs) ➔ dedicated test base

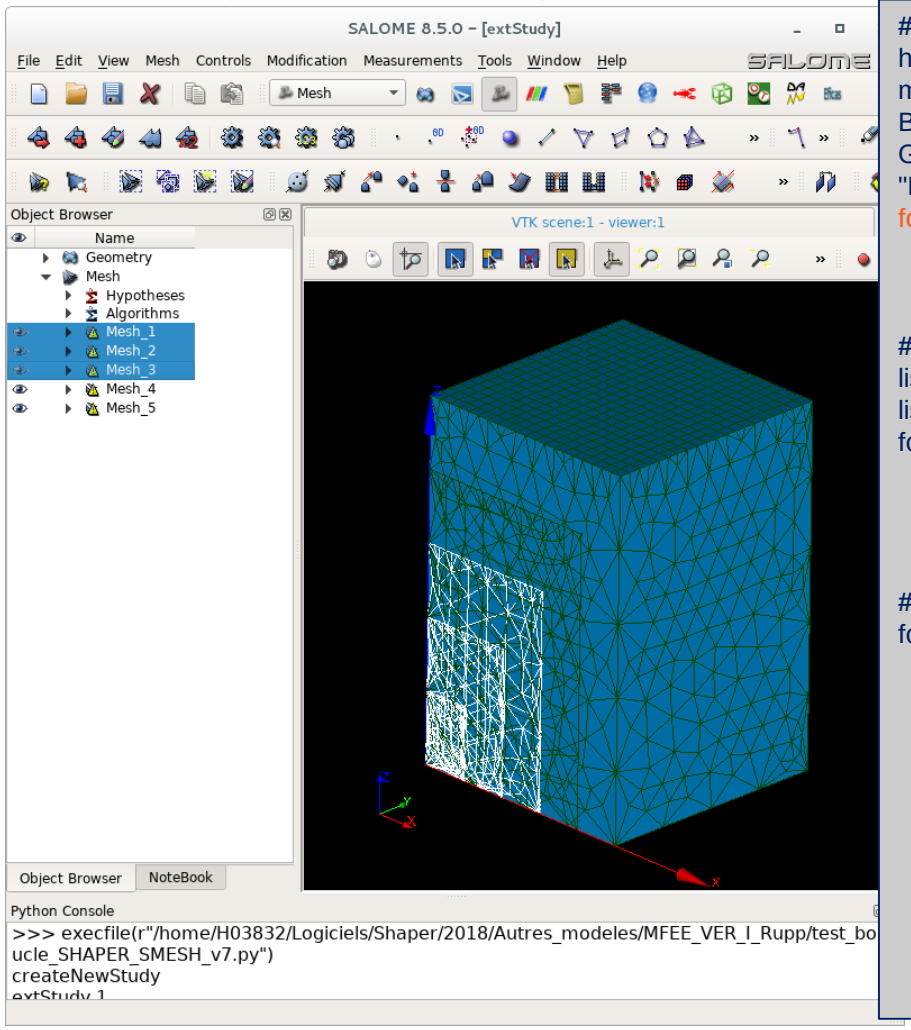
Python Scripting



```
# Draw the sketch of a step
# Draw the axis (sketch and extrusion)
model.do()
# First step:
Extrusion_2 = model.addExtrusion(Part_1_doc, [model.selection("FACE", "Sketch_2/Face-SketchCircle_1_2f")],
model.selection(), 300, 0)
# rotation axis:
Axis_4 = model.addAxis(Part_1_doc, model.selection("FACE", "Extrusion_2_1/Generated_Face_1"))
model.do()
# Loop on the steps: Z translation, recover  then rotation (only by script) :
stairFeature = Extrusion_1
for step in range(15):
    Translation = model.addTranslation(Part_1_doc, [stairFeature.result()], model.selection("EDGE", "PartSet/OZ"), "h")
    Recover = model.addRecover(Part_1_doc, Translation, [stairFeature.result()])
    Rotation = model.addRotation(Part_1_doc, [Translation.result()], model.selection("EDGE", "Axis_1"), "angle")
    model.do() # next transaction
    stairFeature = Rotation # store the next step feature to translate/rotate it in the next iteration
model.end()
```

Reuse of GEOM scripts: published shapes will produce a "dead" BREP (not parametrized) in SHAPER.

Geometry meshing loop




```
### SHAPER component
h=model.addParameter(partSet, "h", str(10))
model.addParameter(partSet, "c", "h*2./3.")
Box_1 = model.addBox(Part_1_doc, "c", "c", "h")
Group_1 = model.addGroup(Part_1_doc, [model.selection("FACE",
"Box_1_1/Top")])
for i in range(1,6):  # 5 cubes créés
    h.setValue(i*10)
    model.do()

### GEOM component
liste_shapes_GEOM=[]
liste_groupes_GEOM=[]
for i,f in enumerate(liste_xao):
    (imported, Box, [], [Group], []) = geompy.ImportXAO(f)
    liste_shapes_GEOM.append(Box)
    liste_groupes_GEOM.append(Group)

### SMESH component
for i,S in enumerate(liste_shapes_GEOM):
    Mesh = smesh.Mesh(S)
    smesh.SetName(Mesh.GetMesh(), 'Mesh_'+str(i+1))
    Regular_1D = Mesh.Segment()
    MEFISTO_2D = Mesh.Triangle(algo=algo2D)
    NETGEN_3D = Mesh.Tetrahedron()
    Group_1=liste_groupes_GEOM[i]
    Regular_1D_1 = Mesh.Segment(geom=Group_1)
    Mesh.Quadrangle(algo=smeshBuilder.QUADRANGLE,geom=Group_1)
    isDone = Mesh.Compute()
    Sub_mesh_1 = Mesh.GetSubMesh( Group_1, 'Sub-mesh_1' )
    Mesh.ExportMED(path+"mesh_"+S.GetName()+".med")
```

Python addons

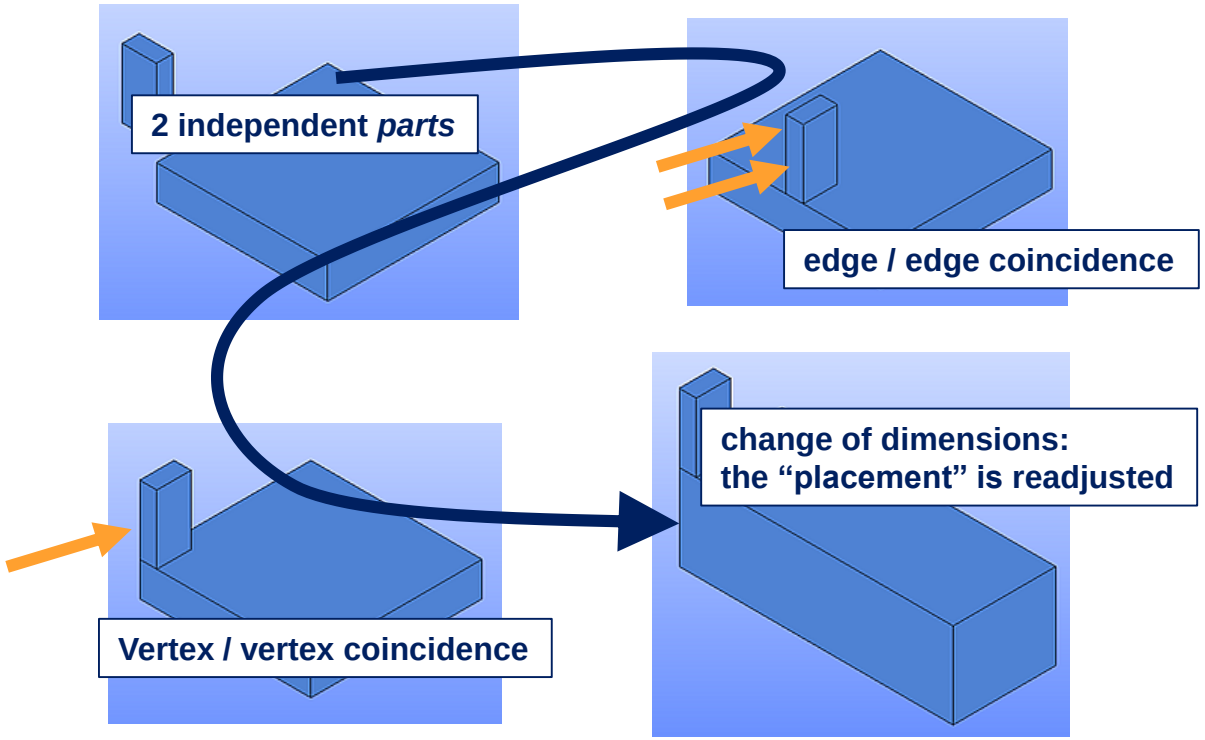
- It is possible to create custom features.
- The corresponding folder should be created for each feature at ../sources/src/PythonAddons/macros.
- A feature description includes 4 files:
  - widget.xml with a description of the property panel,
  - __init__.py,
  - feature.py with python commands,
  - icon.png with image of button in the toolbar (the file is located at sub-folder /icons).

Example: the rectangle in the sketcher

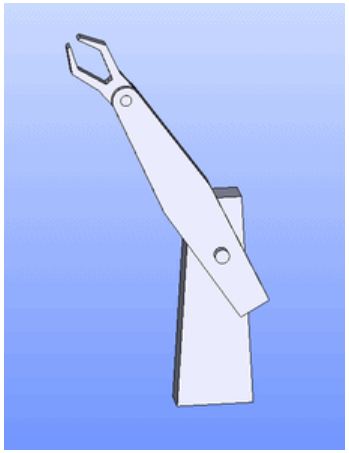
…/salome/Shaper_9.3.0/Salome-V9_3_0-c9/modules/SHAPER_V9_3_0/share/doc/salome/gui/SHAPER/addons_Features/rectangleFeature.html

Parts Assembly

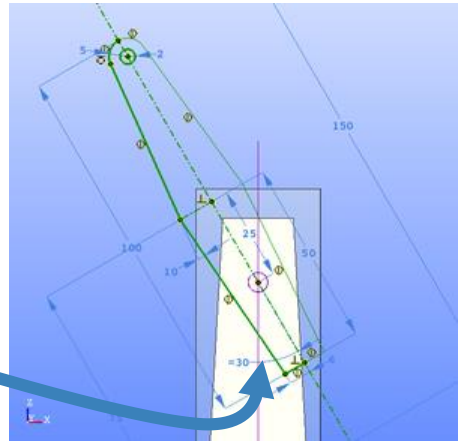
- “Placement” of parts at the level of the partset or bodies (results) within a part.
- Accumulation of placement constraints (descent face, edge, point)



Mobile parts Assembly



setting the angle by parameter in the sketch



Partset : advantages

- *PartSet* = assembly of *Parts*

Pros:

- Modularity: guarantee the independence of the parts between them
- Draw parts moving relatively to each other
- In the long term, re-use (several times) of the same part

Cons:

- no sharing between Parts while it's easy to share within the same part ...

Method:

- Draw a common sketch at the Parset level ➔ it will be shared by all the Parts
- Define parameters at PartSet level
- Move / copy parts with the functions of the Part menu: placement, translation, rotation, symmetry etc.
- Warning: no "recover" possible

THANK YOU