PUC RIO

Universidade Federal Fluminense

Computer Graphics
for Engineering

TU Graz
Graz University of Technology

ifb

numsim
Numerical simulation in technical sciences

Geometric and Solid Modeling

Luiz Fernando Martha
André Pereira

Graz, Austria
June 2014

Outline

- Motivation
- Solid Modeling
- Modeling in Engineering
- Geometric Modeling
- Parametric Modeling

Introduction

Drawing

Traditional Approach – First CAD Generation

(*Computer Aided Design*)

Drawing

Traditional Approach – First CAD Generation

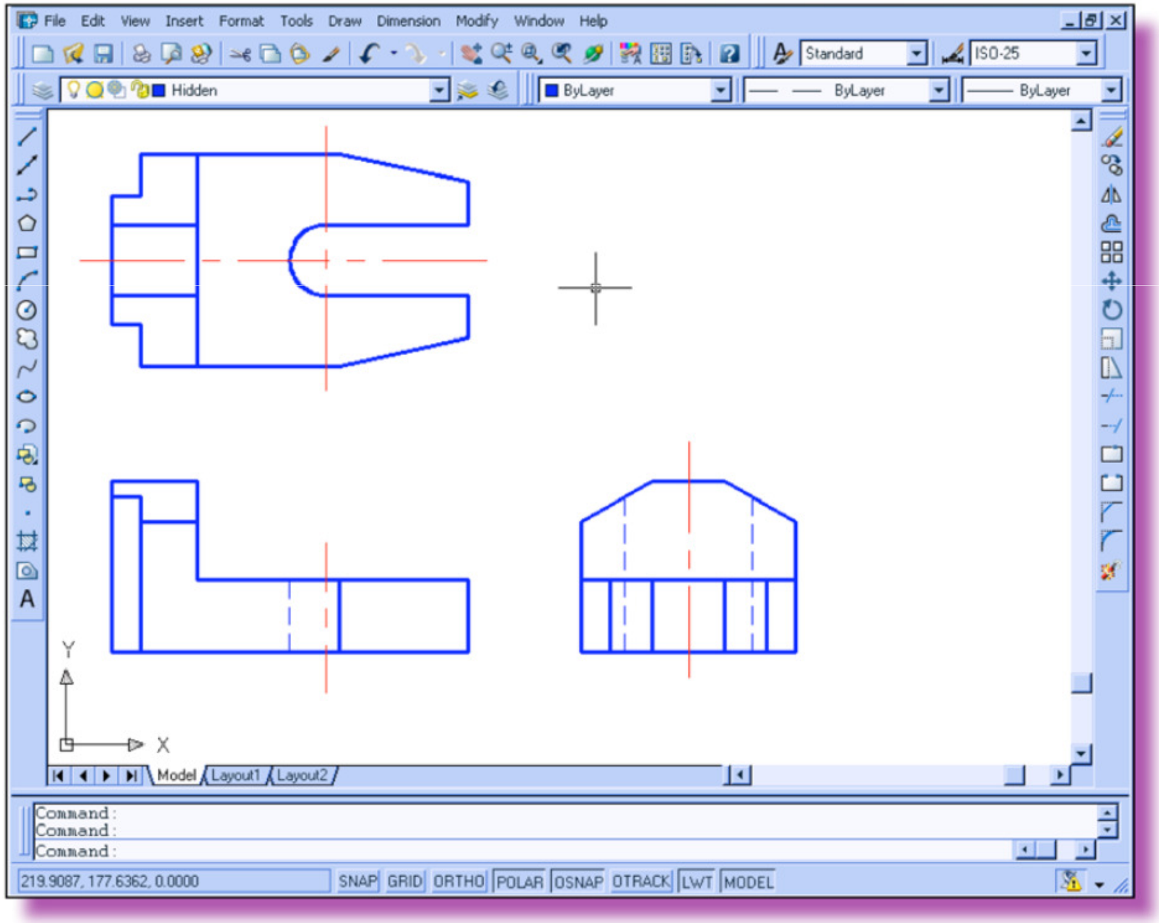
(*Computer Aided Design*)

The first generations of CAD were just in 2D, basically substituting the pencil and paper.

Drawing

Traditional Approach – First CAD Generation
(*Computer Aided Design*)

[SHIH2006]



The first generations of CAD were just in 2D, basically substituting the pencil and paper.

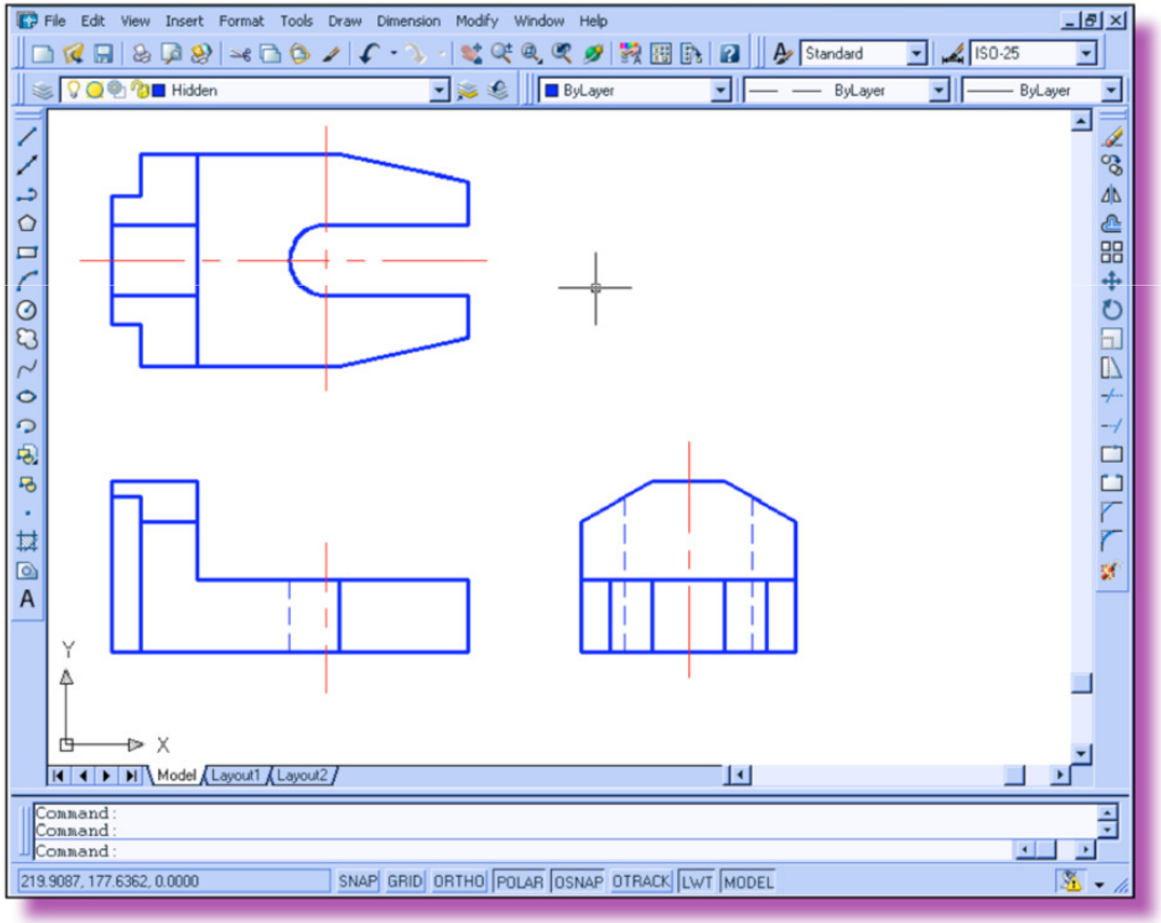
The so popular AutoCAD, distributed for the first time in 1981, gained popularity and is one of the leading CAD systems.

Drawing

Traditional Approach – First CAD Generation

(*Computer Aided Design*)

[SHIH2006]



The first generations of CAD were just in 2D, basically substituting the pencil and paper.

The so popular AutoCAD, distributed for the first time in 1981, gained popularity and is one of the leading CAD systems.

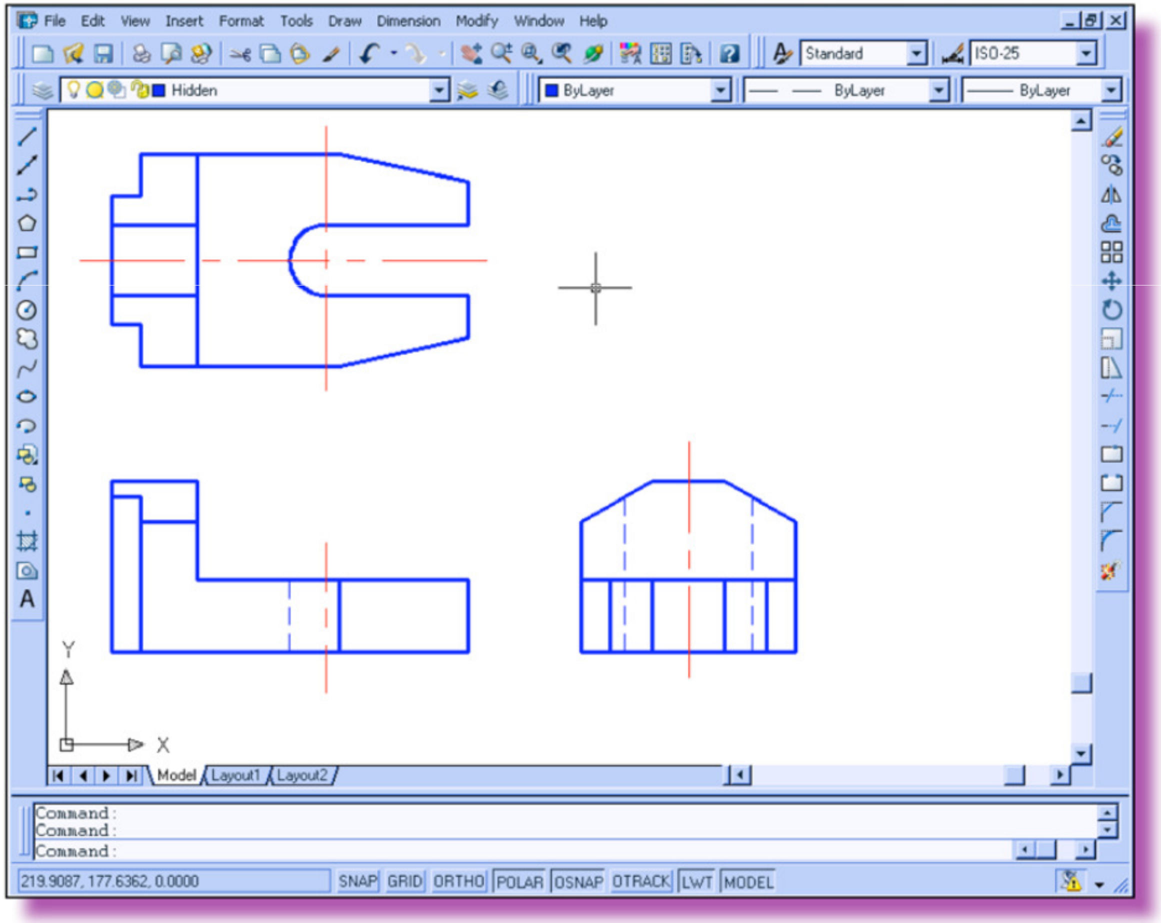
Even today, many companies use 2D CAD to create designs.

Drawing

Traditional Approach – First CAD Generation

(*Computer Aided Design*)

[SHIH2006]



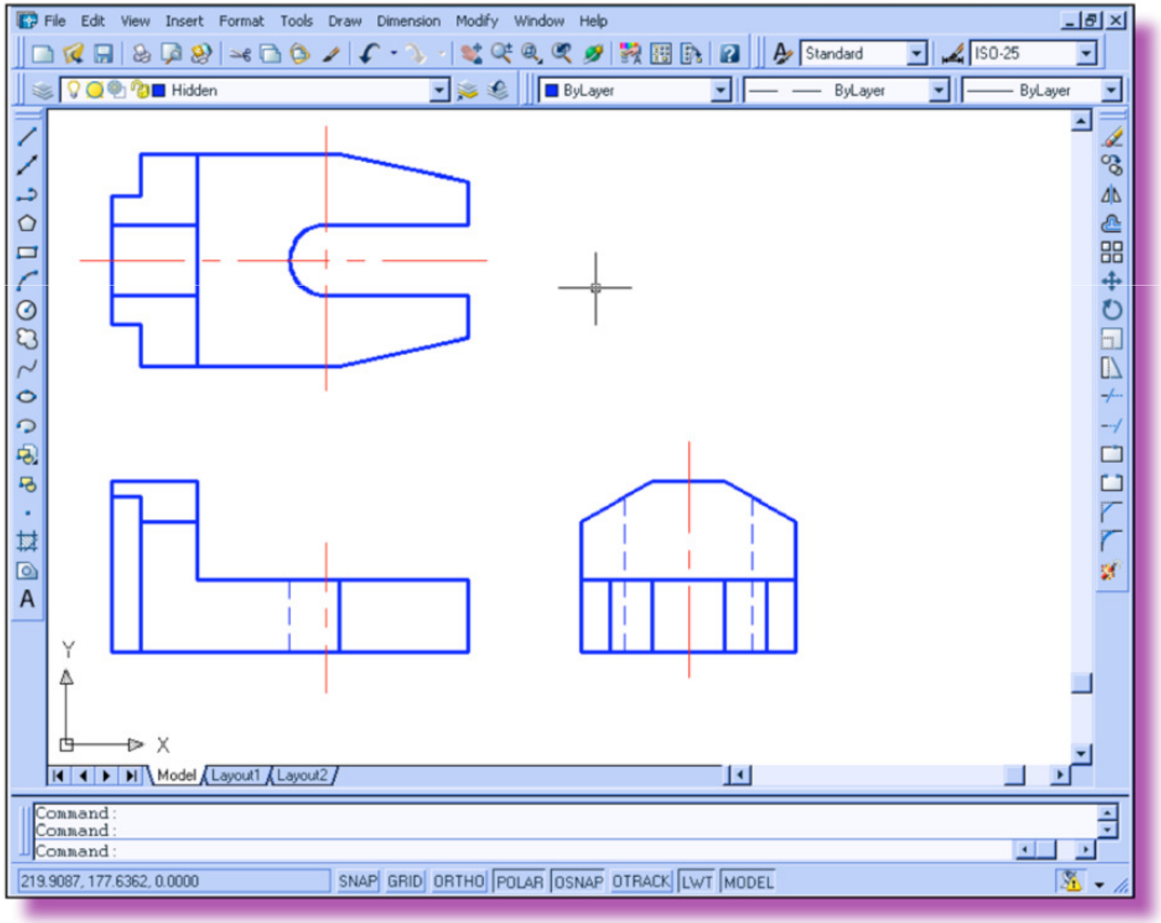
This approach requires knowledge of the actual dimensions of the project, being no flexible at all.

Drawing

Traditional Approach – First CAD Generation

(*Computer Aided Design*)

[SHIH2006]



This approach requires knowledge of the actual dimensions of the project, being no flexible at all.

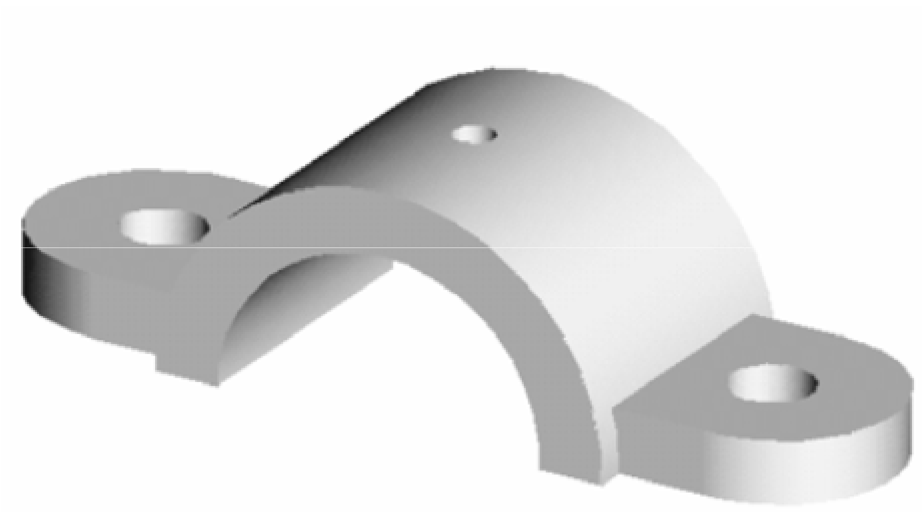
Note in the Figure that:

(1) The creation of these views require the knowledge of the dimensions.

(2) Each of the three views is created and edited completely independent of the others.

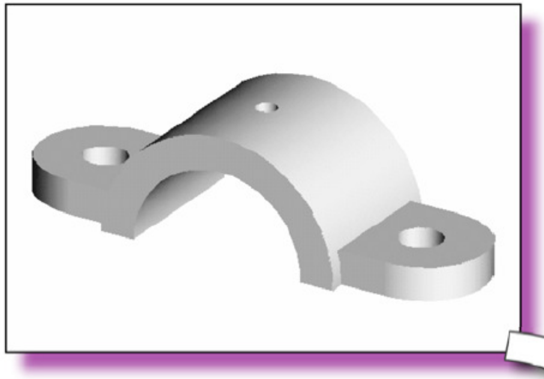
Solid Modeling

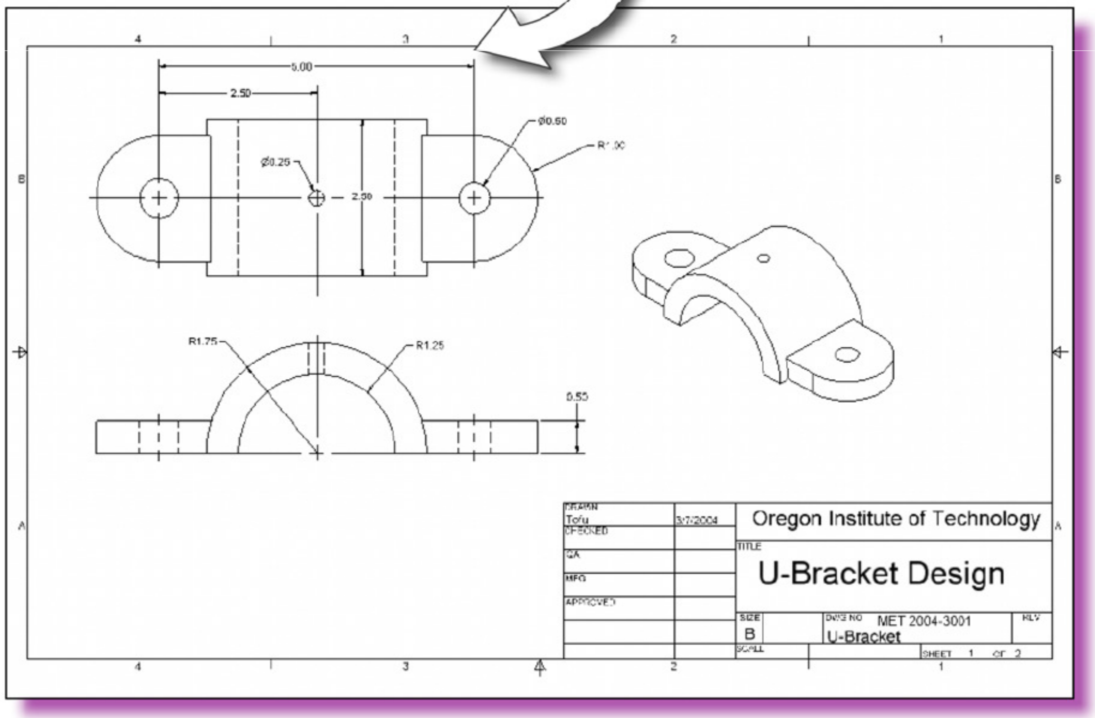
Solid Modeling



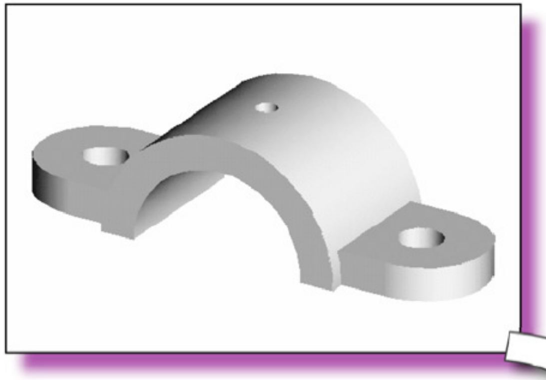
Solid Modeling

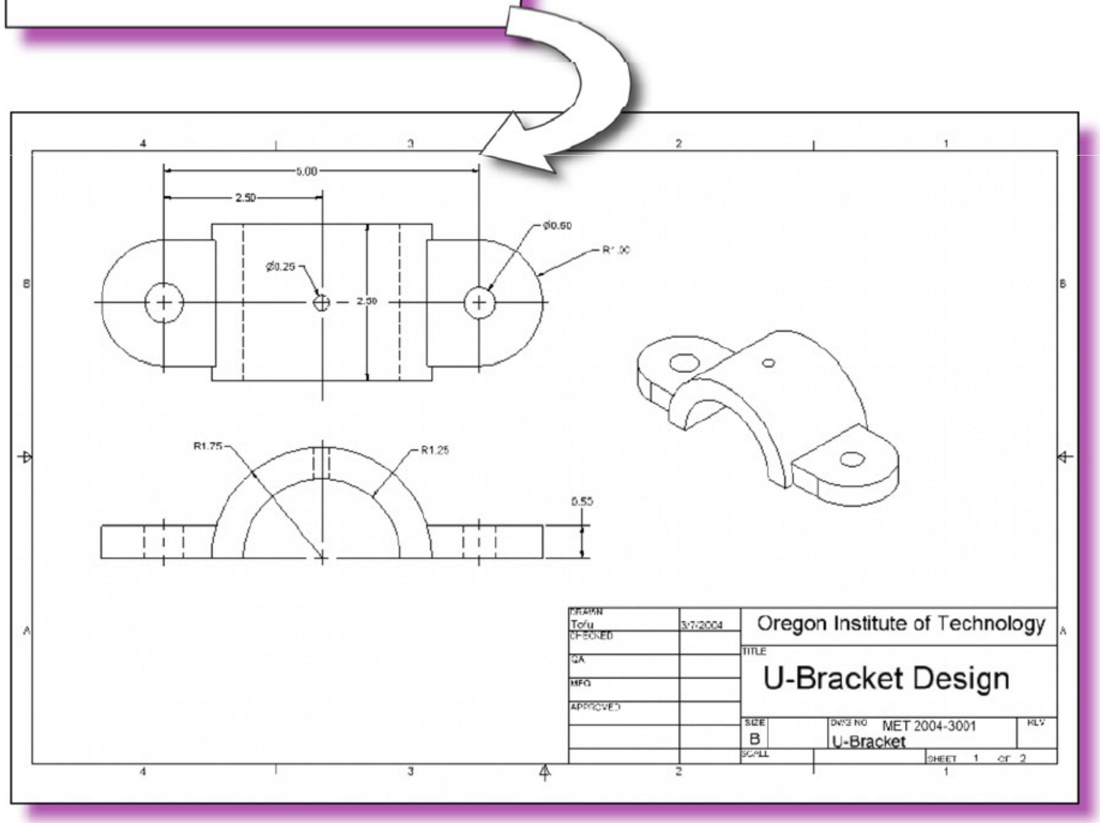
The 2D drawings are generated from the 3D model.





Solid Modeling



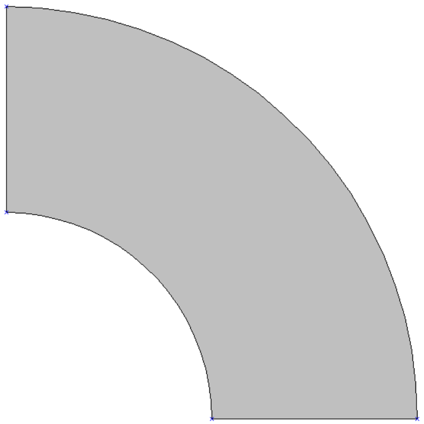


The 2D drawings are generated from the 3D model.

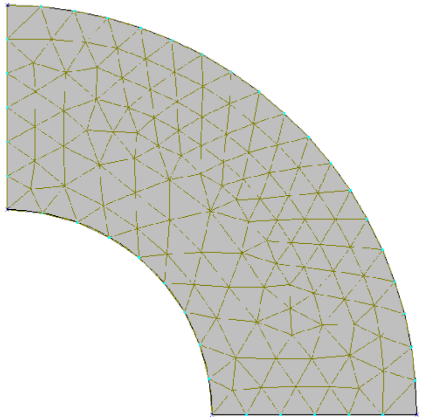
Modifications are automatically updated.

Modeling in Engineering

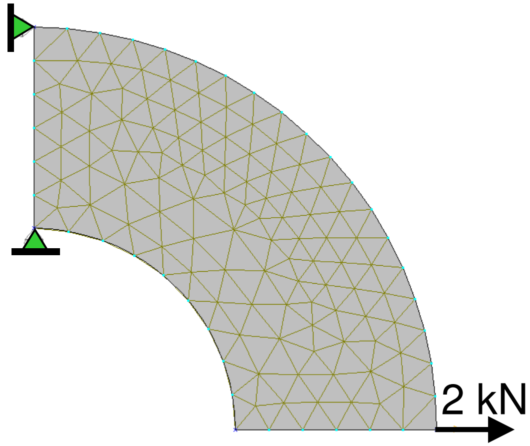
Traditional FE Simulation Process



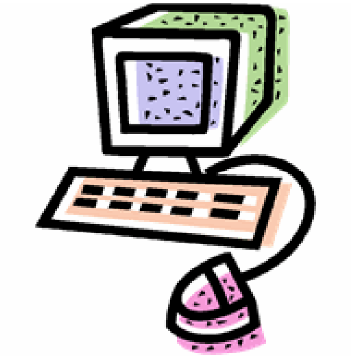
1. Build geometric model



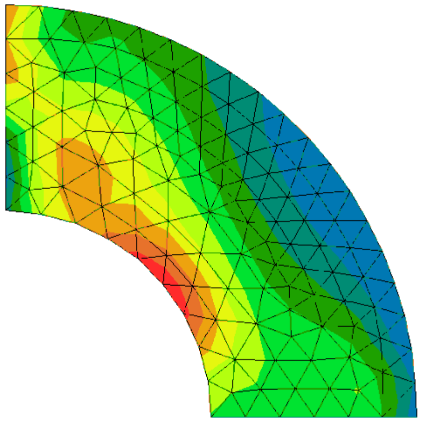
2. Mesh



3. Apply boundary conditions

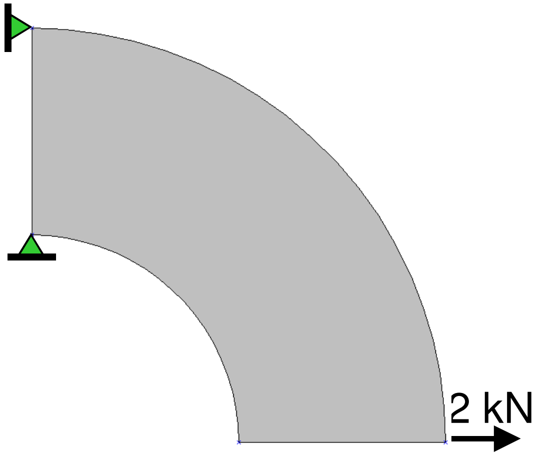


4. Computational analysis

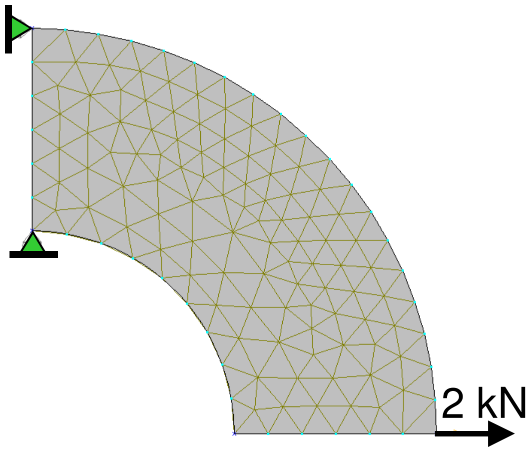


5. Result visualization

Geometry-based Simulation Process

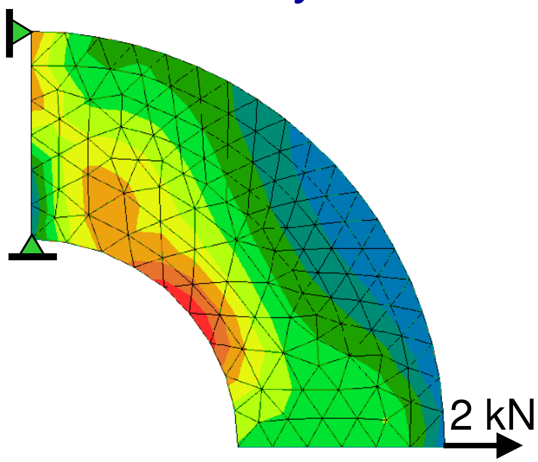


1. Geometric modeling, apply attributes and boundary conditions



2. FE mesh generation, apply boundary conditions

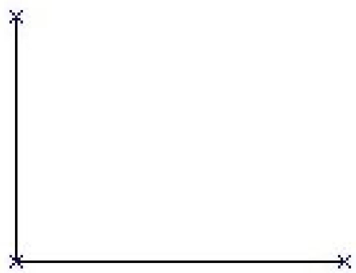
3. Computational analysis



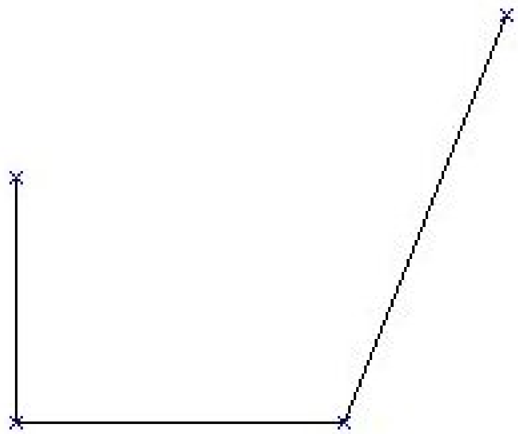
4. Result visualization

Construction of a Simple FE Model

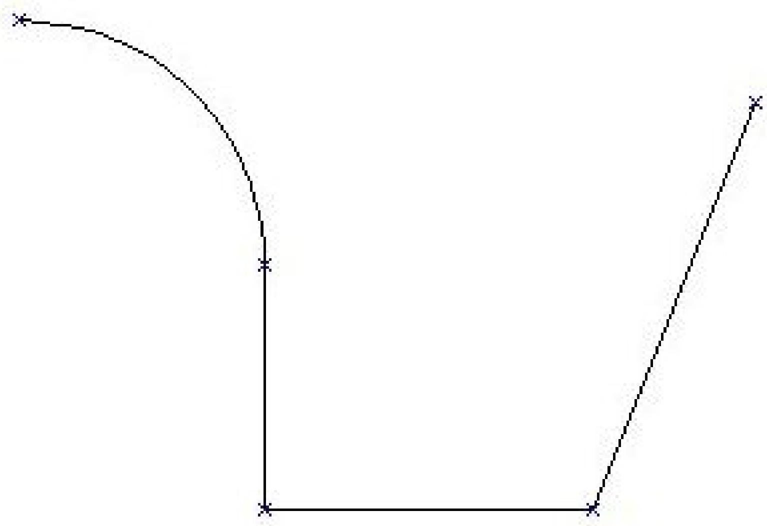
Construction of a Simple FE Model



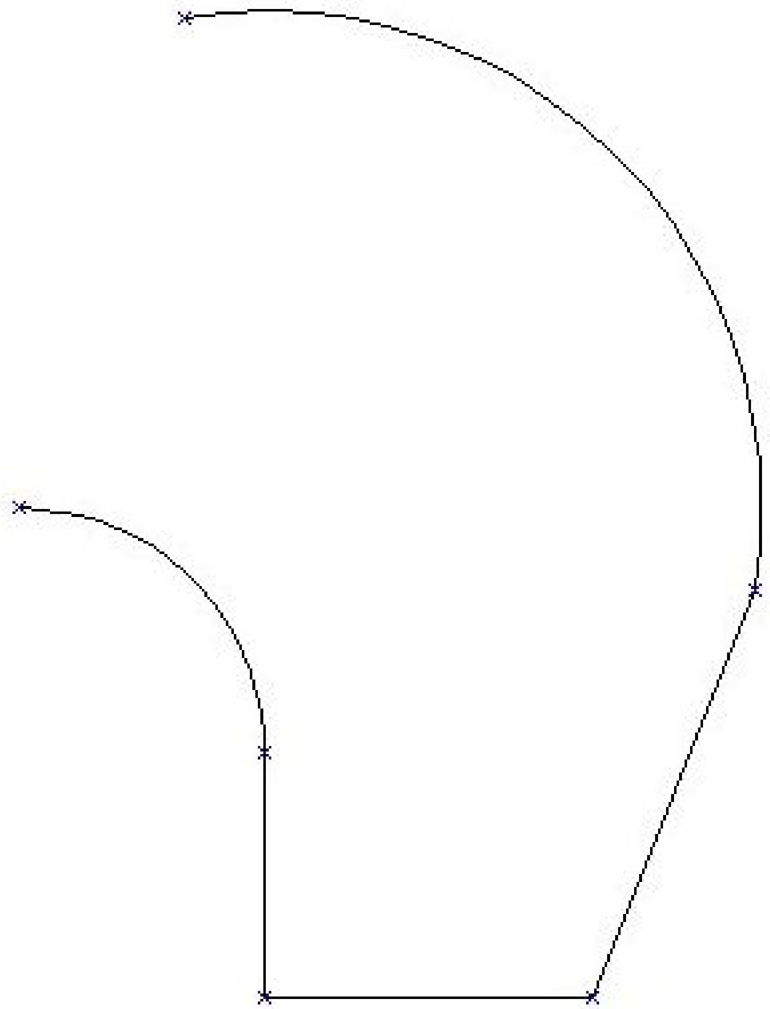
Construction of a Simple FE Model



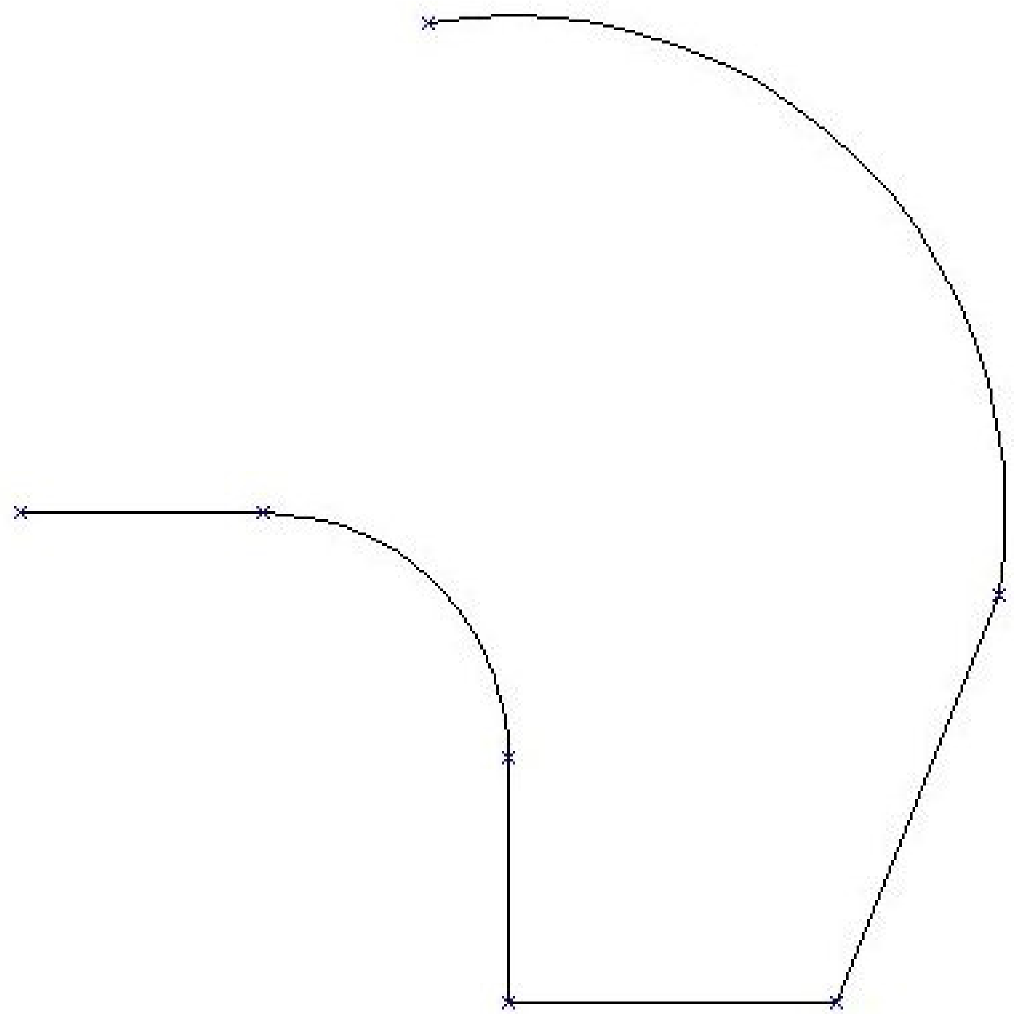
Construction of a Simple FE Model



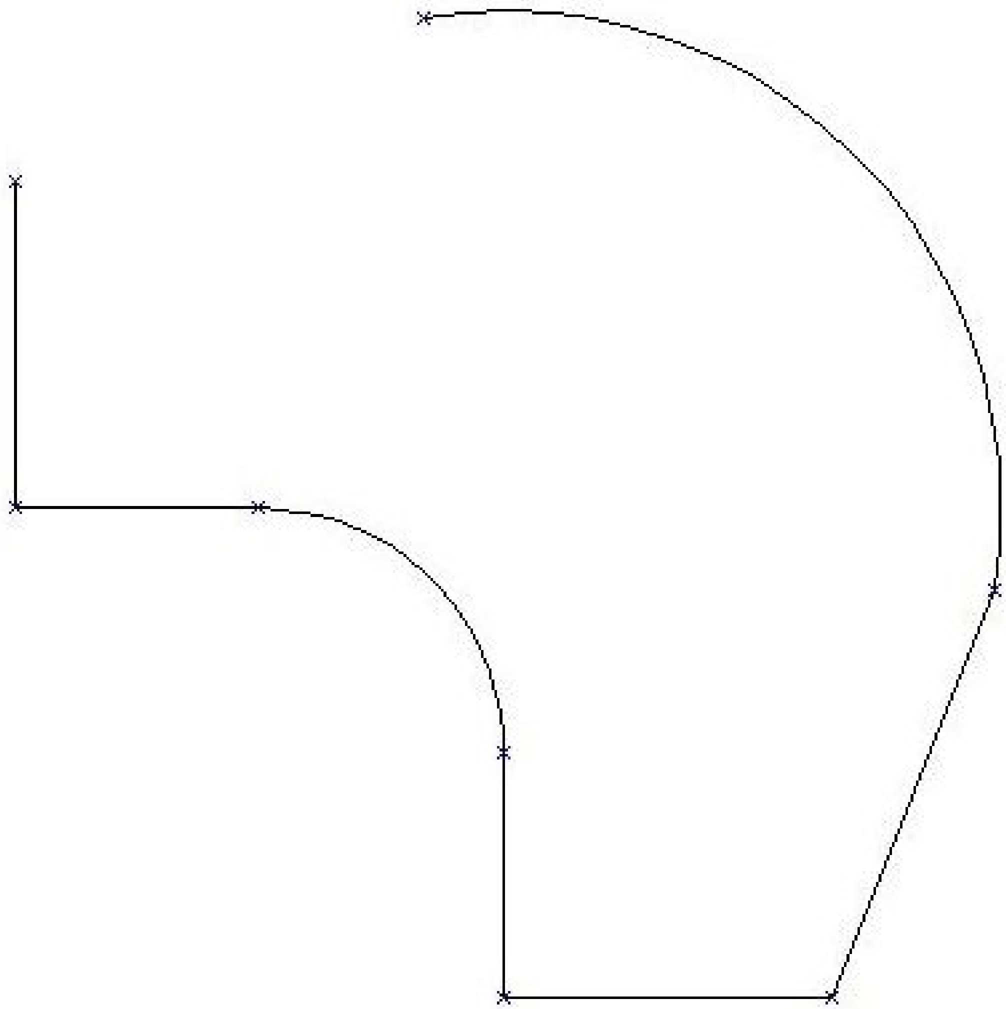
Construction of a Simple FE Model



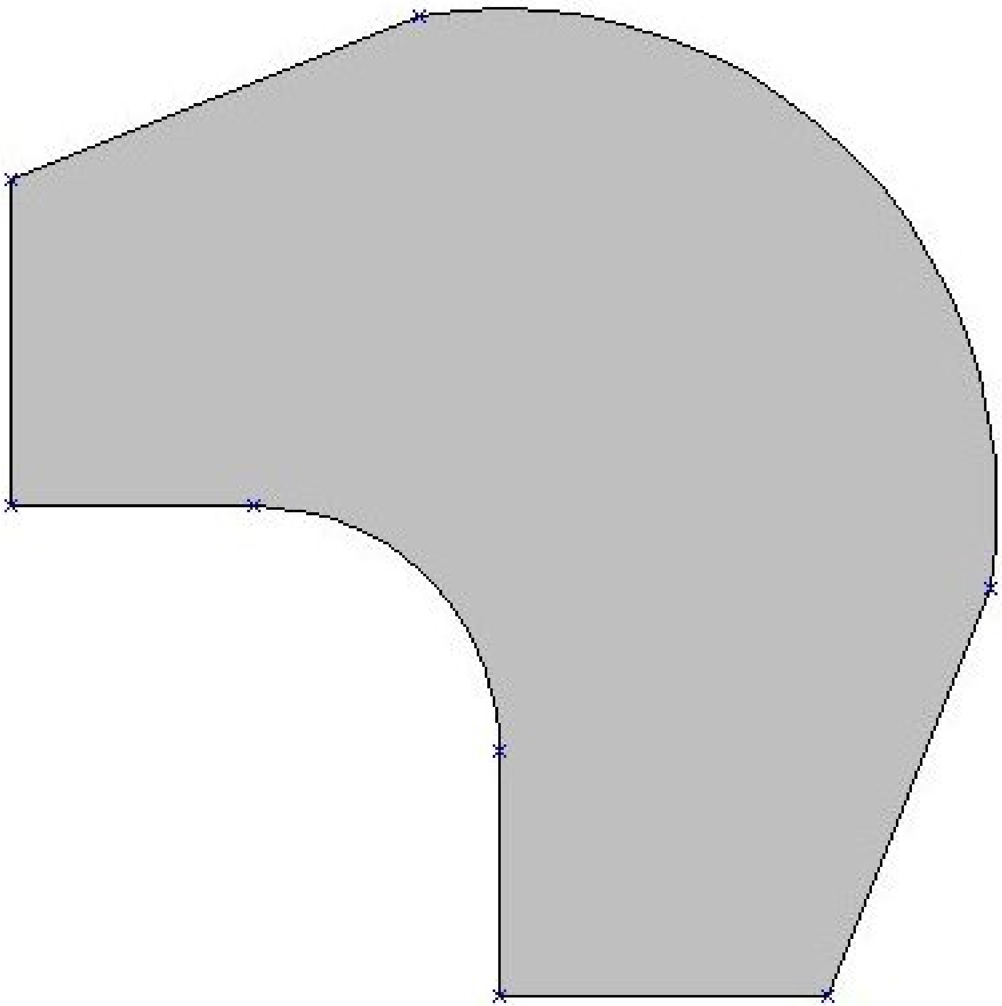
Construction of a Simple FE Model



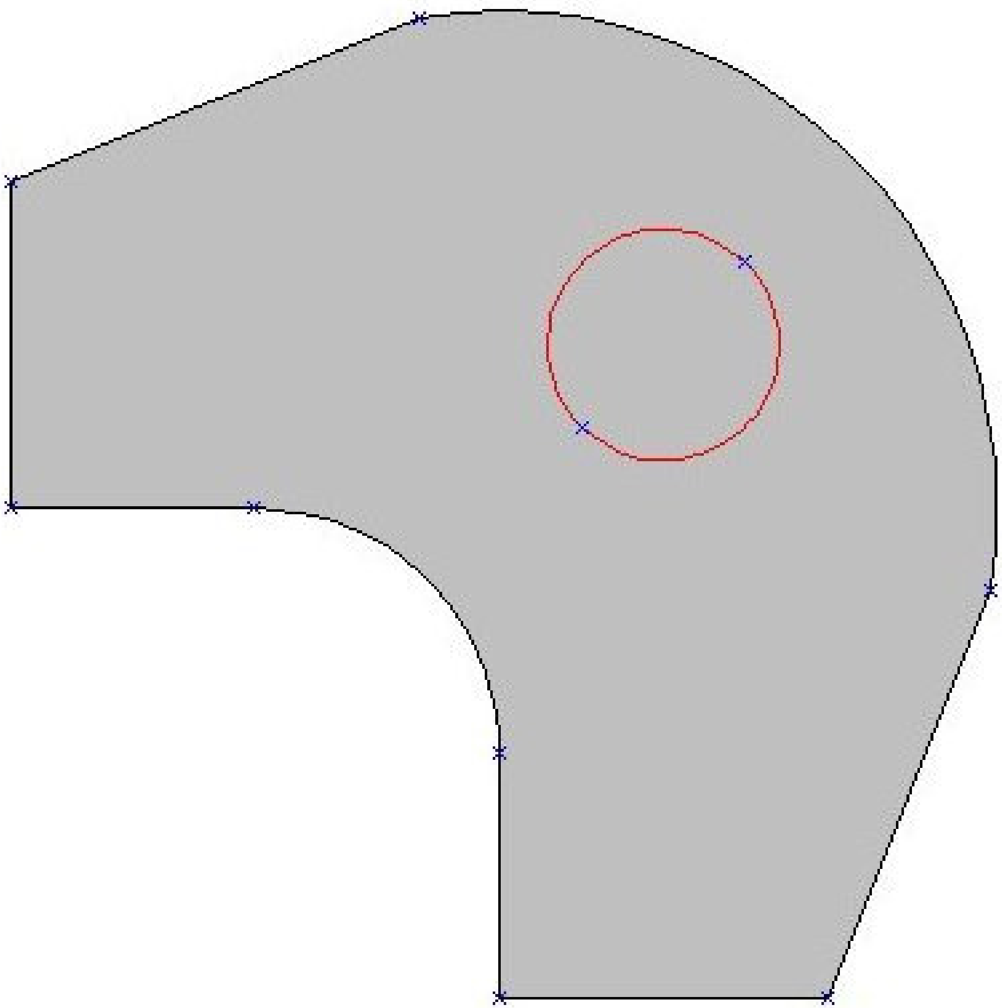
Construction of a Simple FE Model



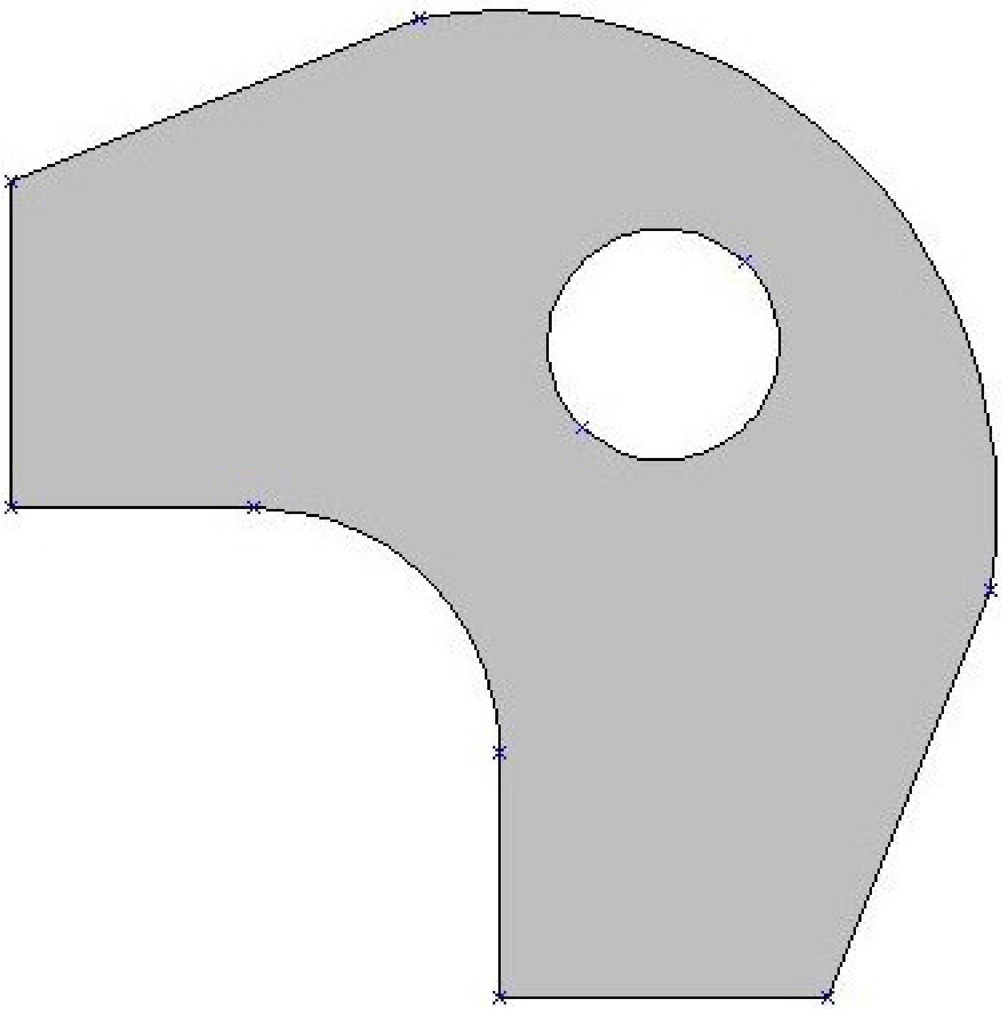
Automatic region recognition



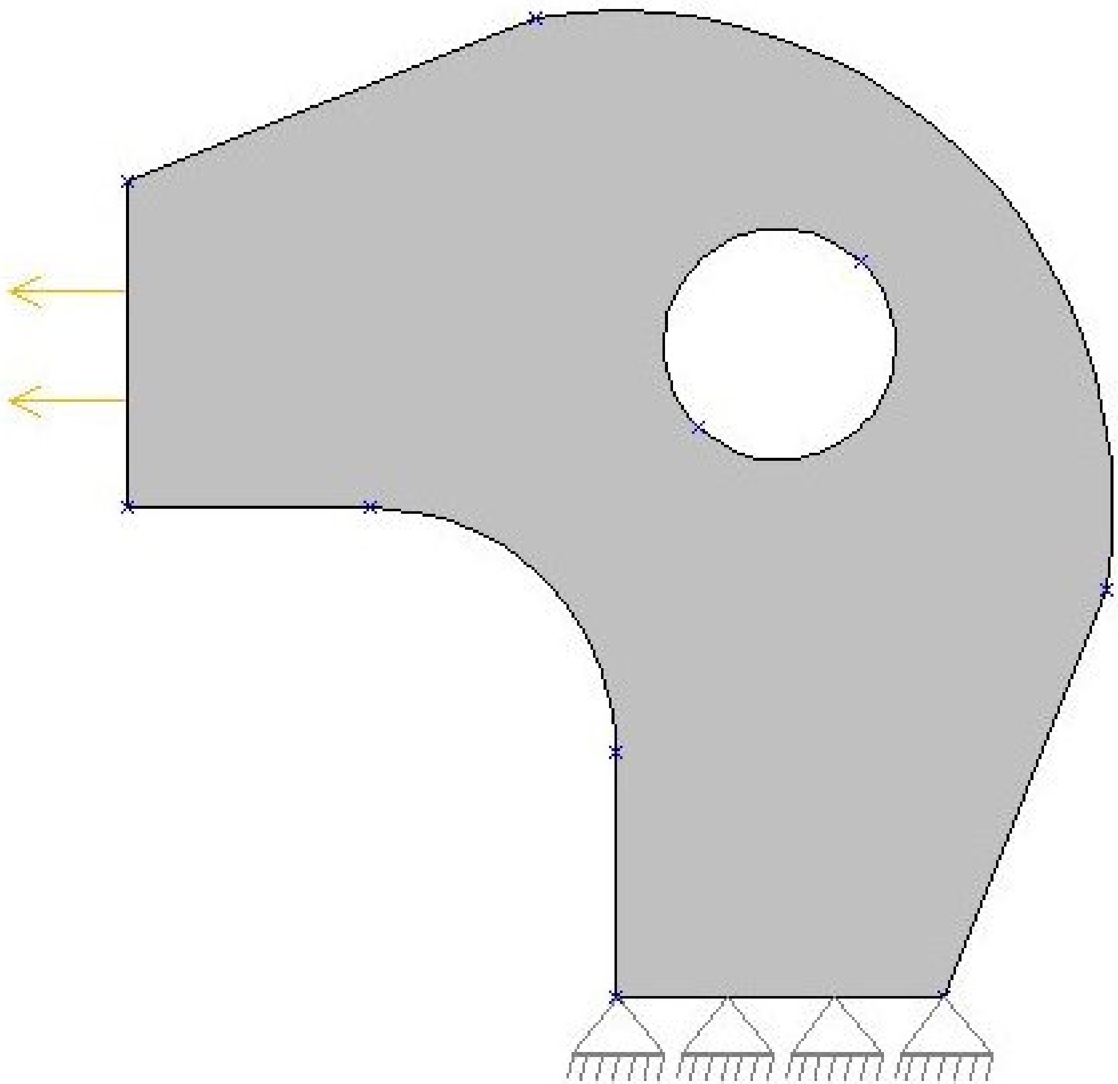
Creating a hole



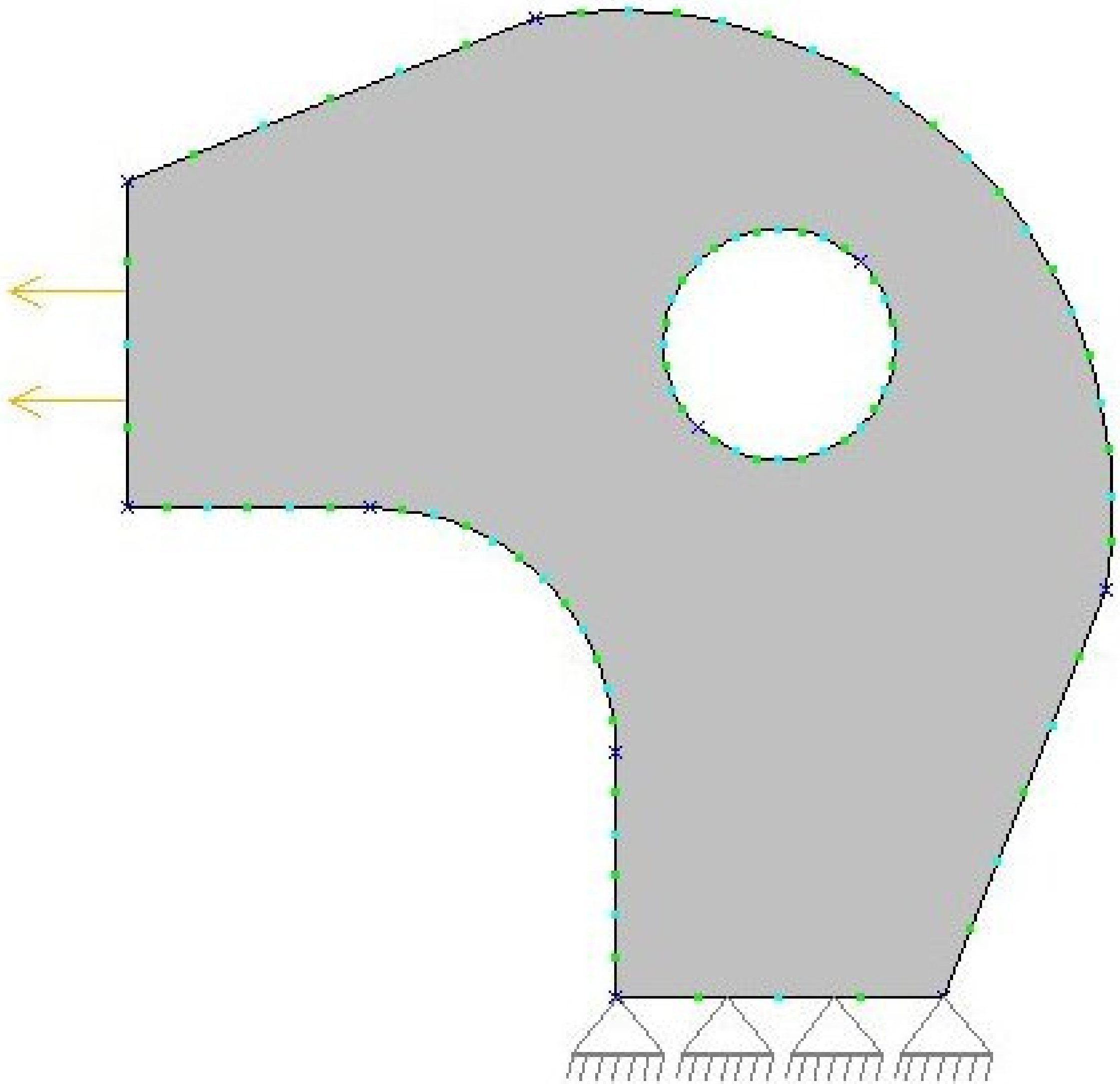
Assigning hole attribute



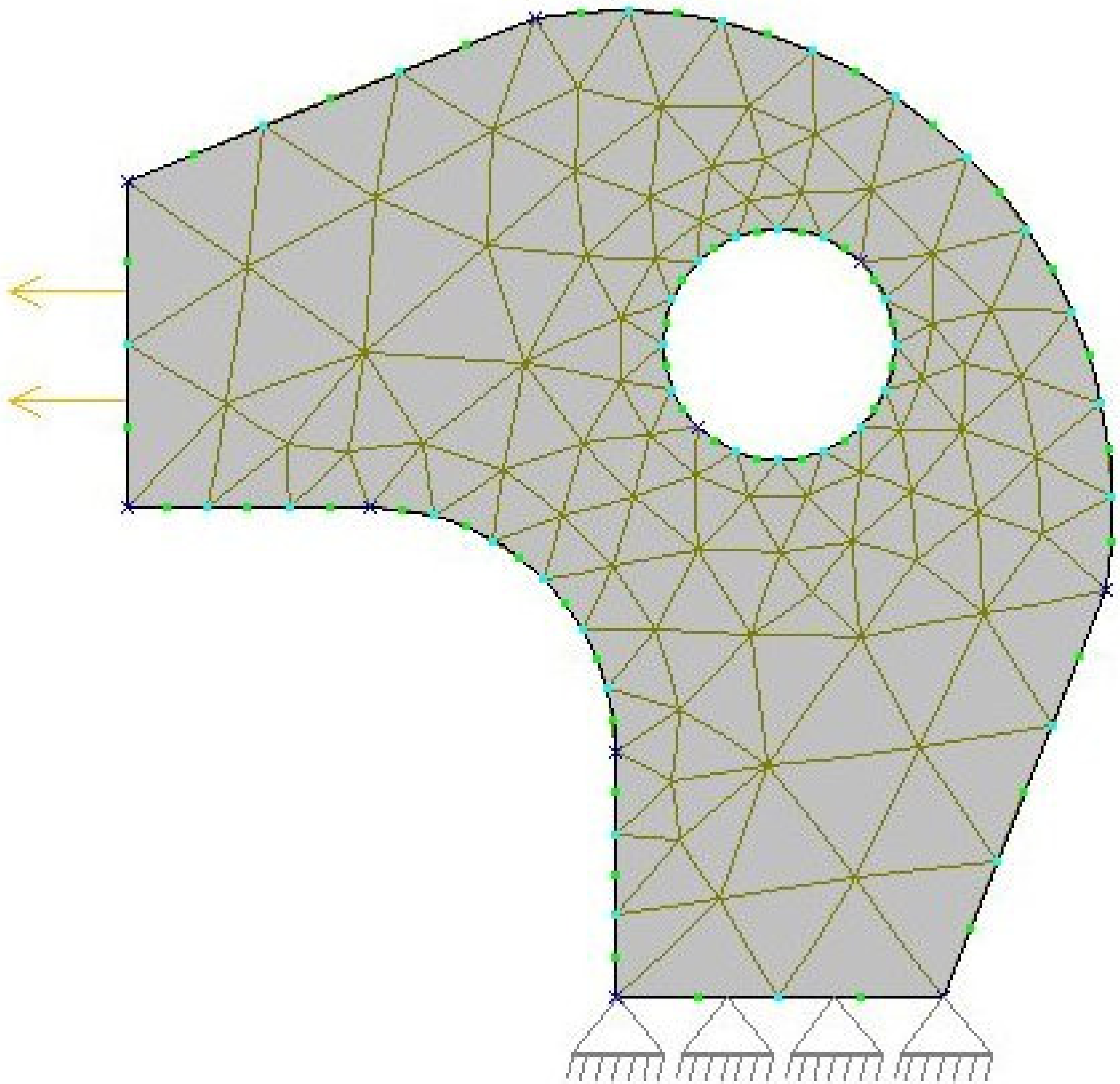
Applying attributes to geometry



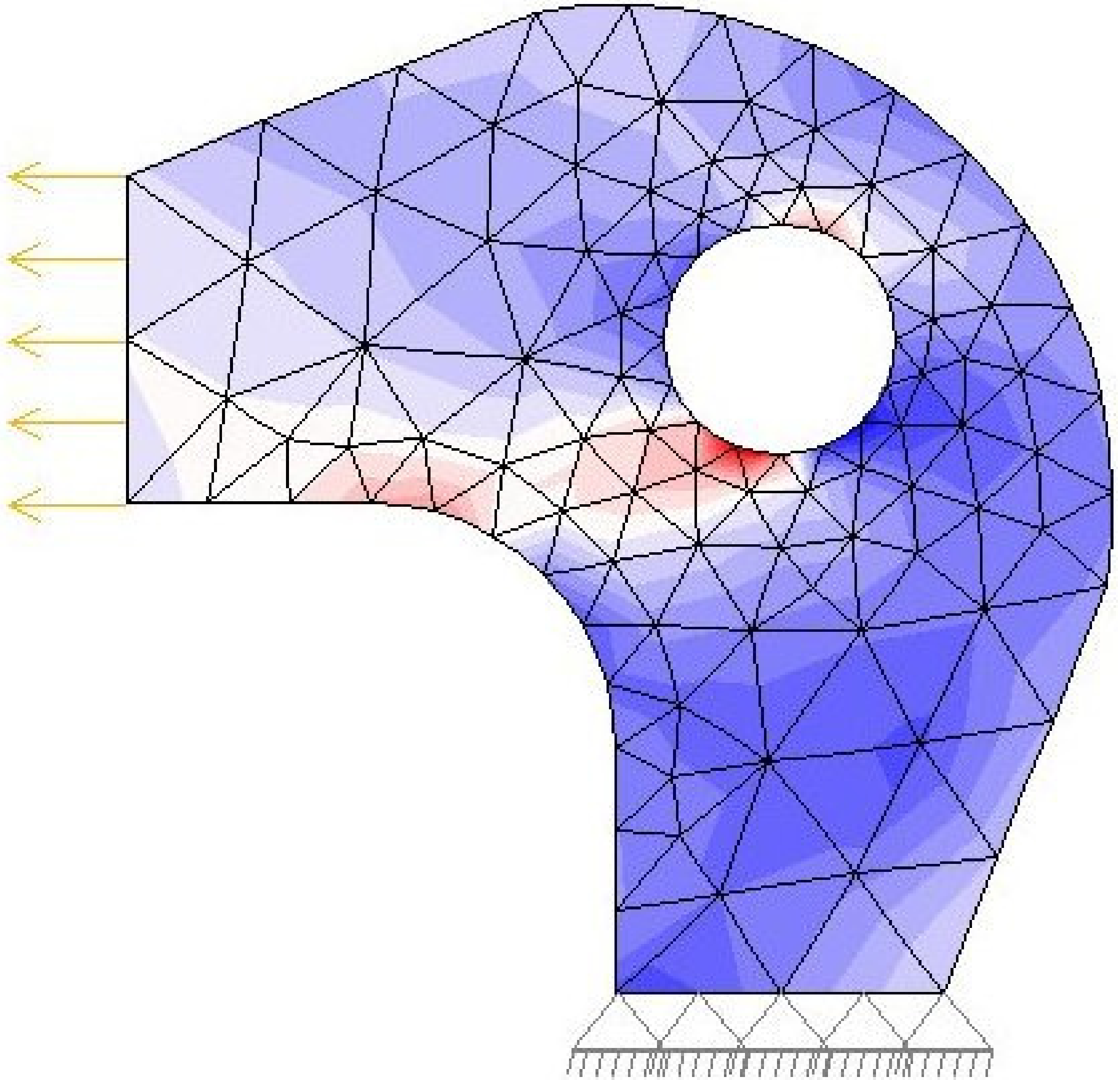
Defining meshing refinement parameters: boundary subdivision



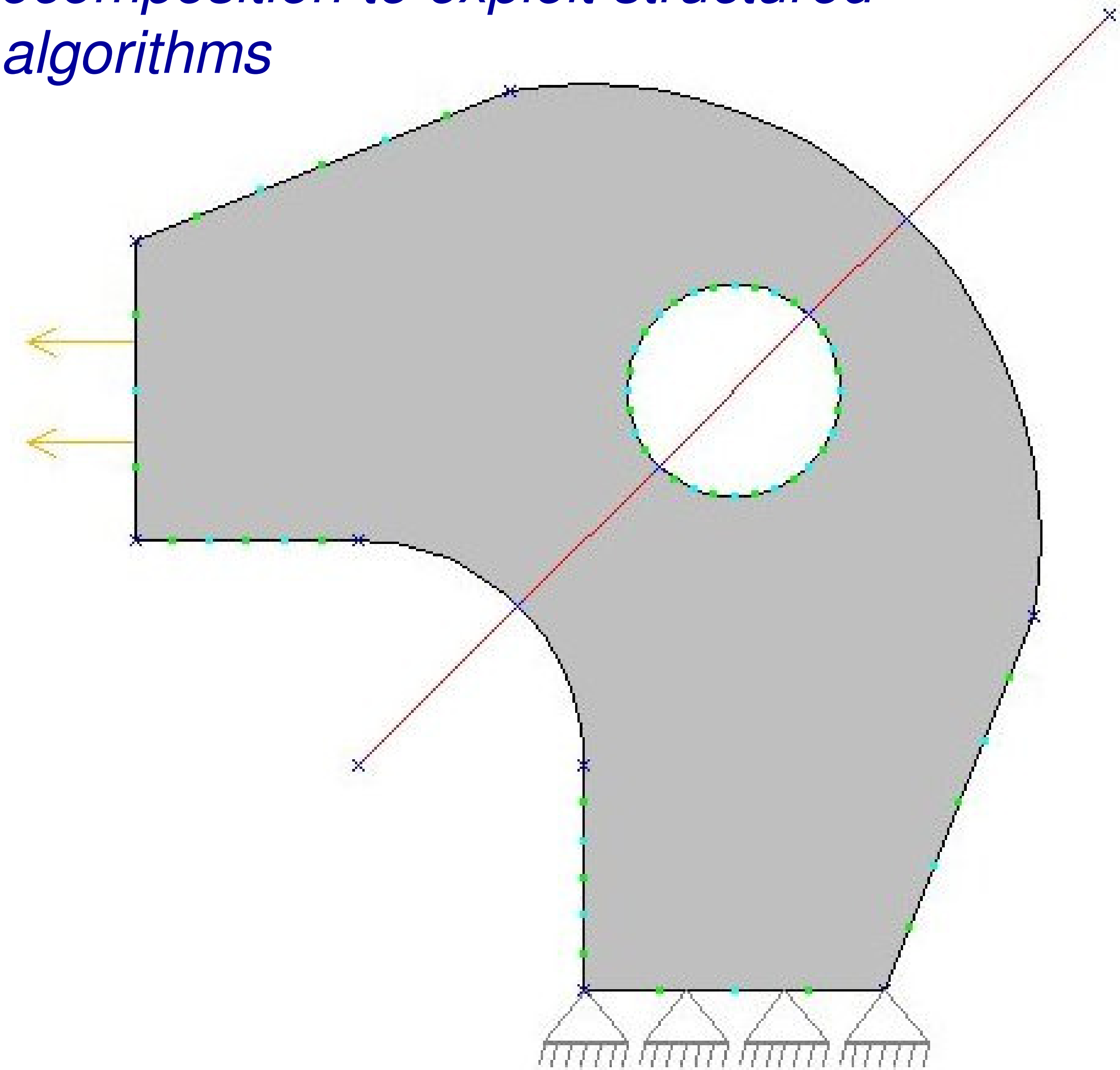
Automatic unstructured mesh generation



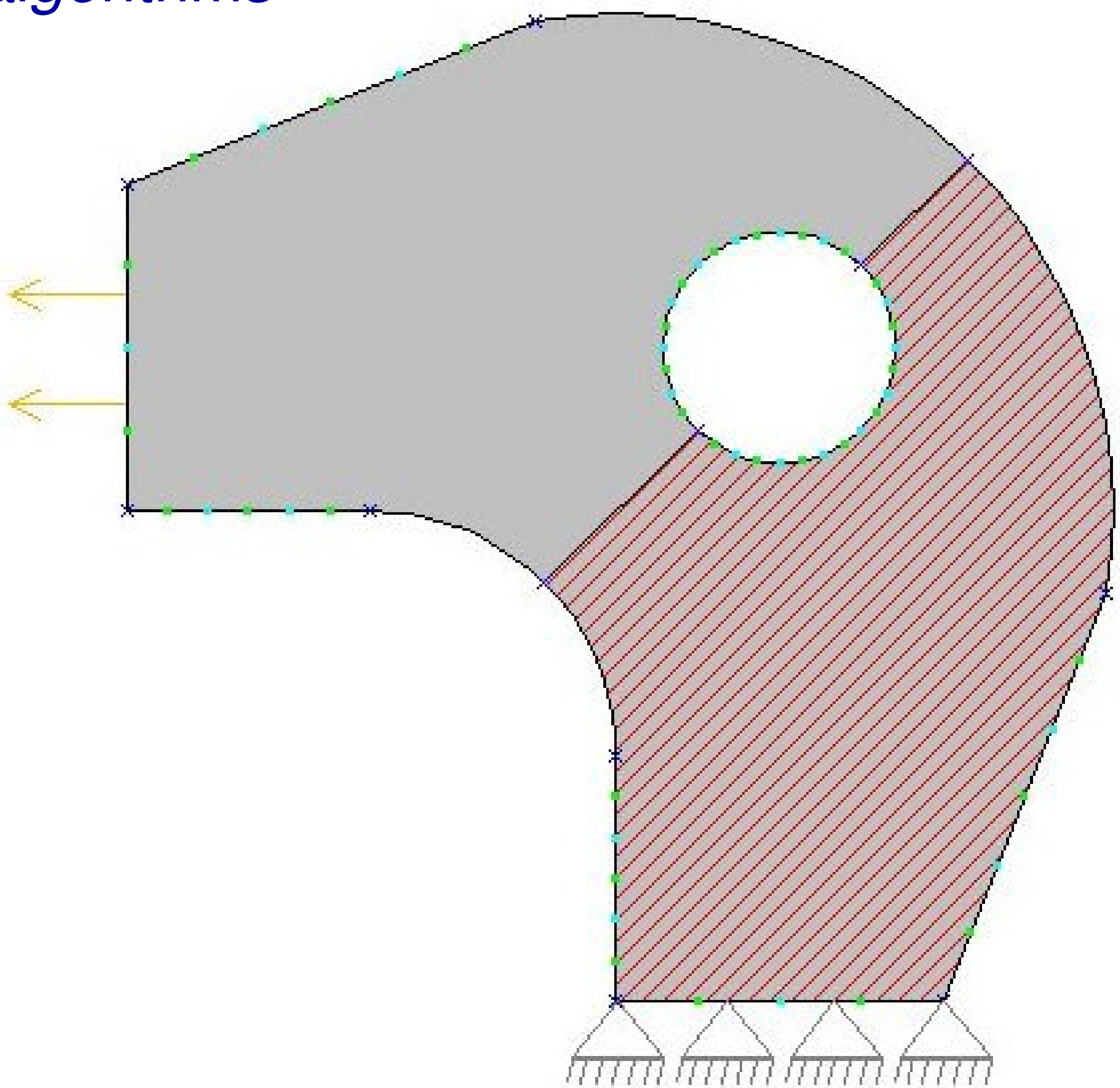
Attributes automatically assigned to mesh entities



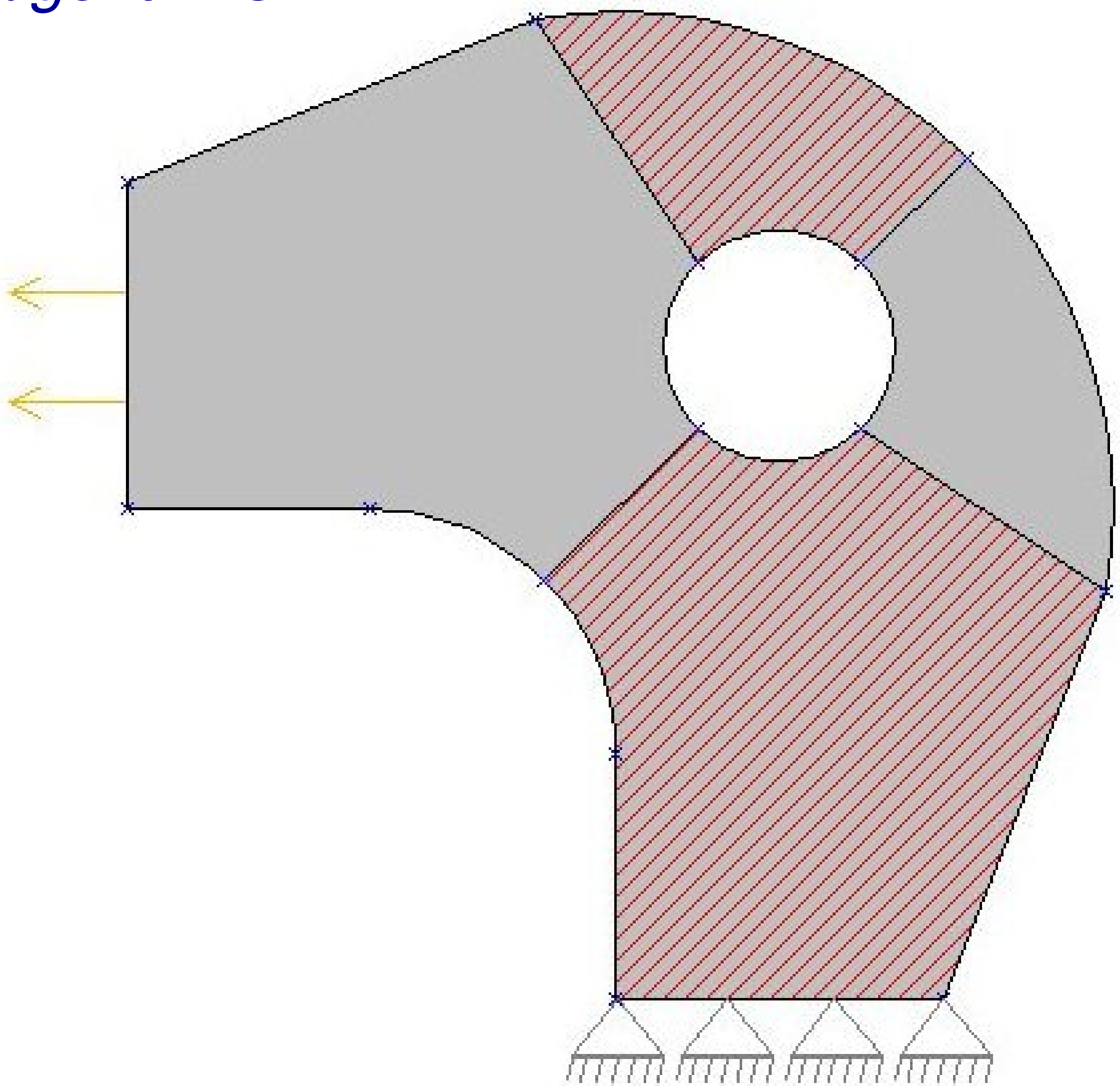
Region decomposition to exploit structured meshing algorithms



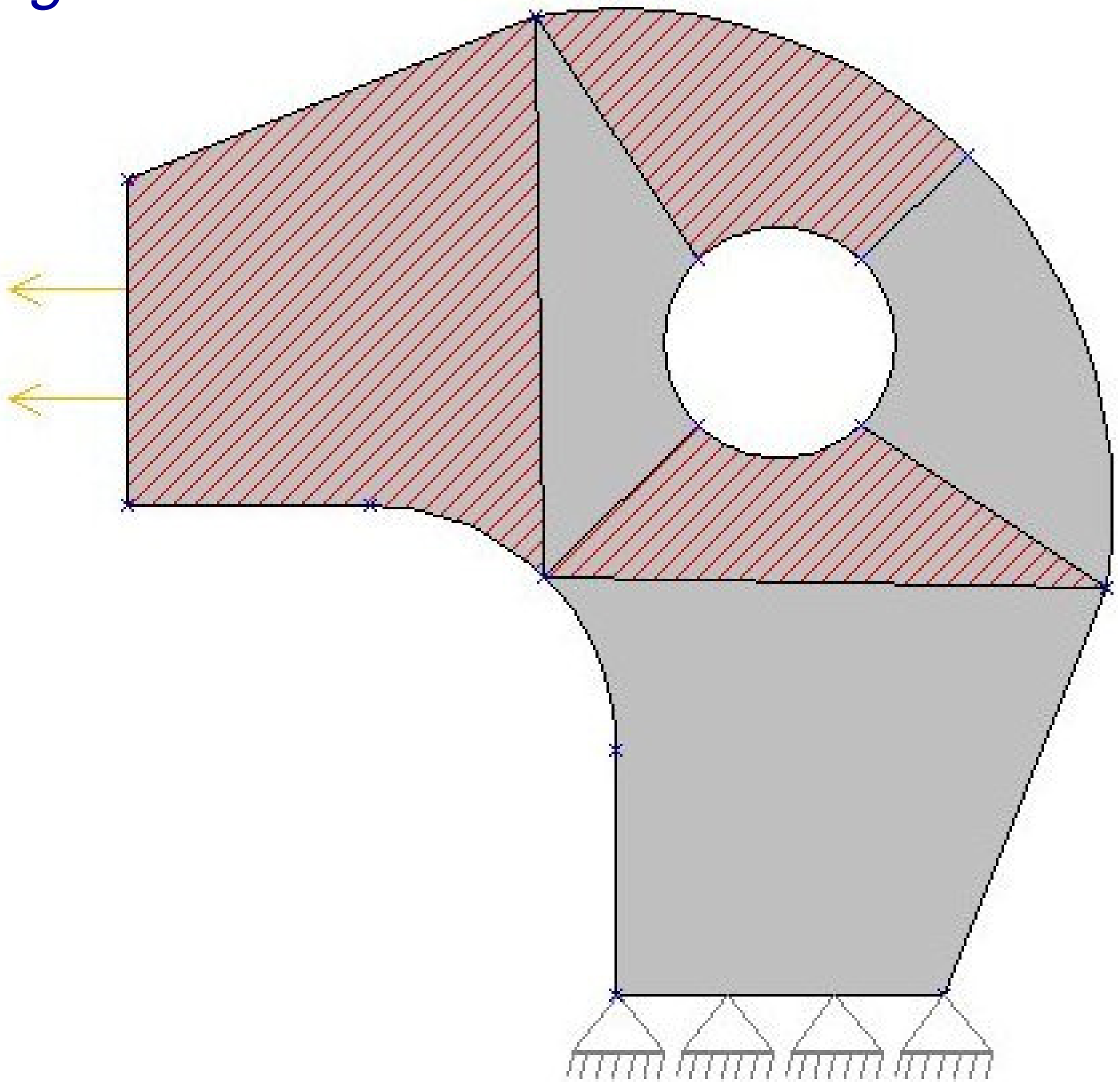
Region decomposition to exploit structured meshing algorithms



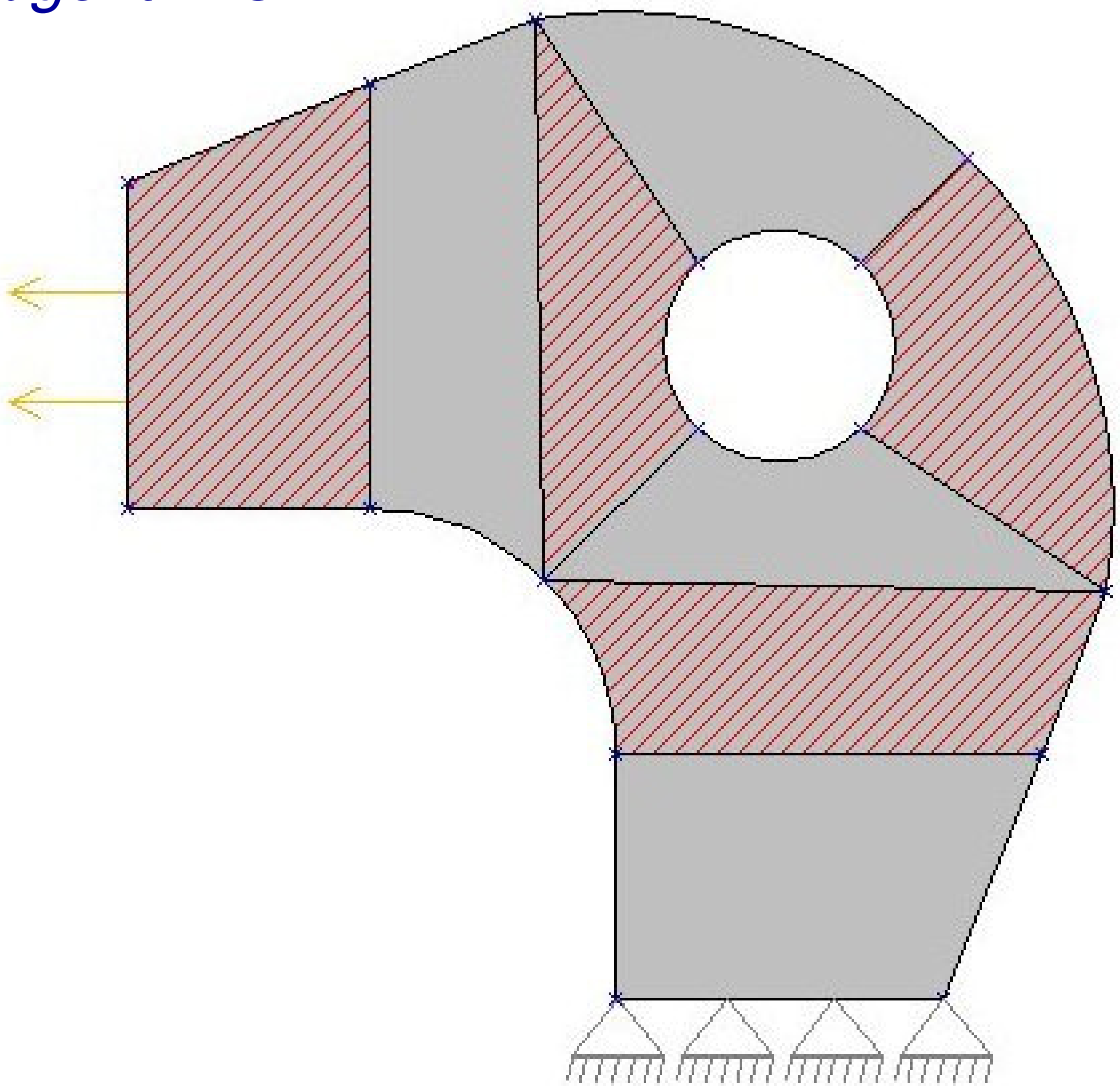
Region decomposition to exploit structured meshing algorithms



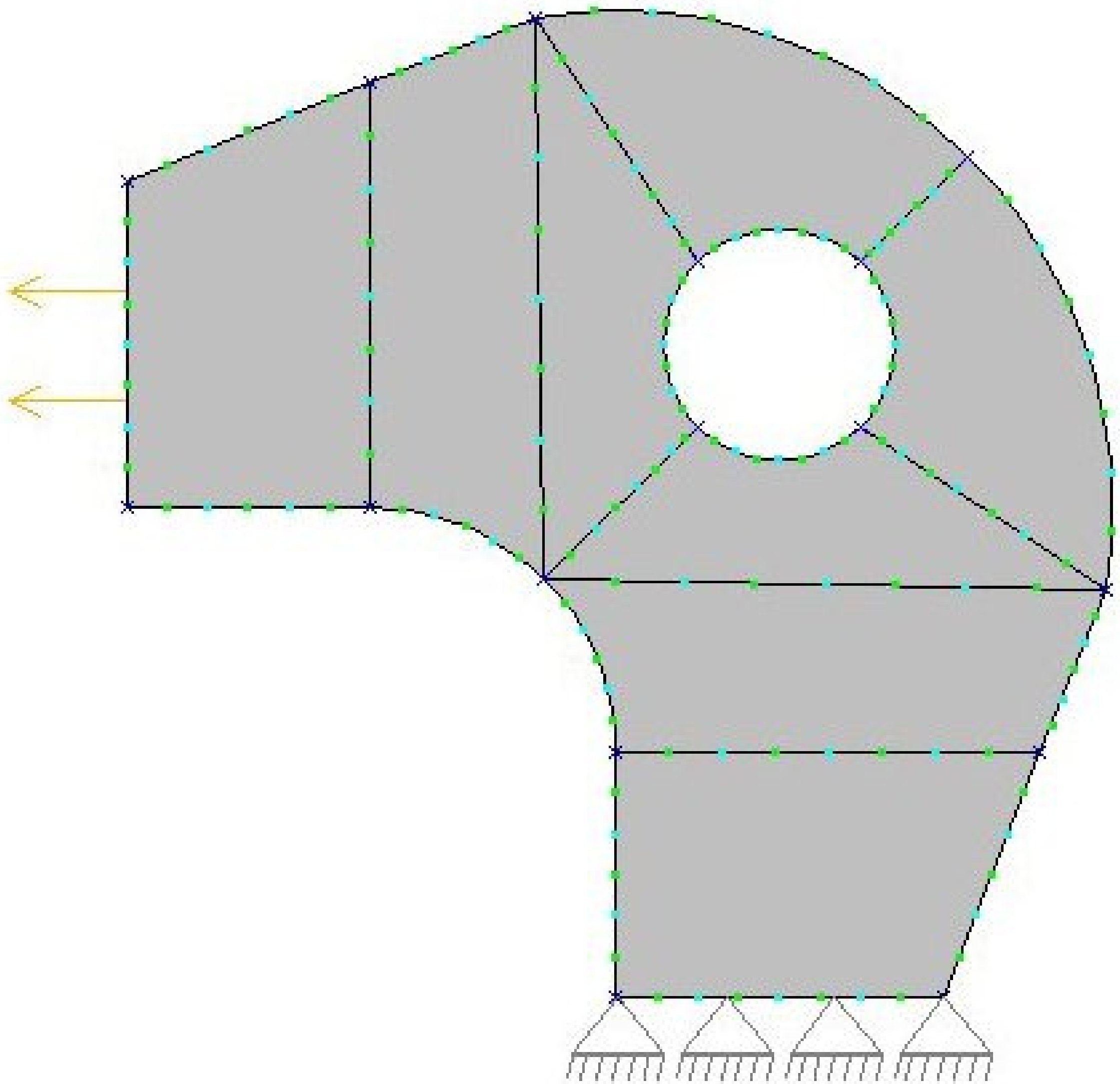
Region decomposition to exploit structured meshing algorithms



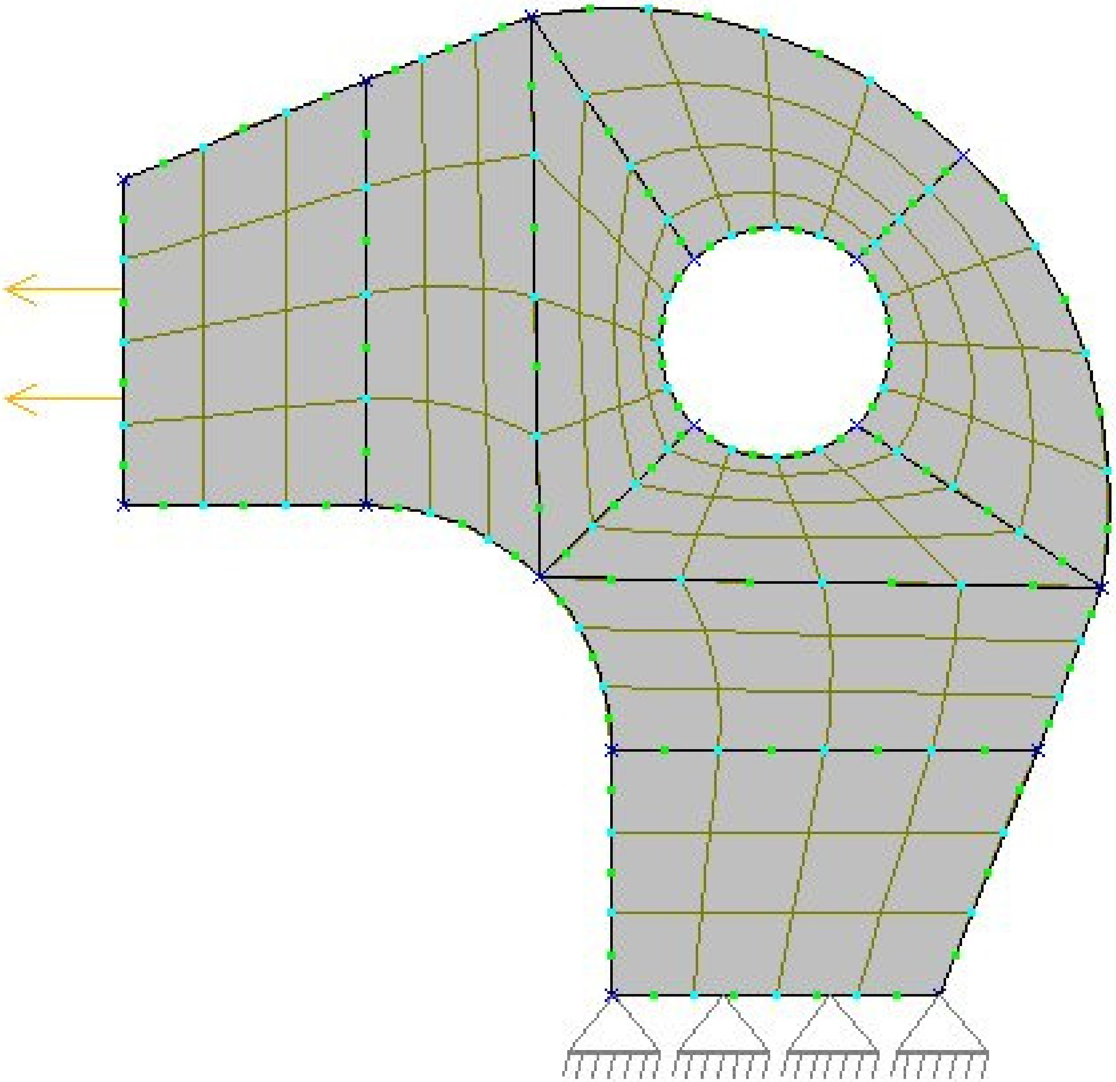
Region decomposition to exploit structured meshing algorithms



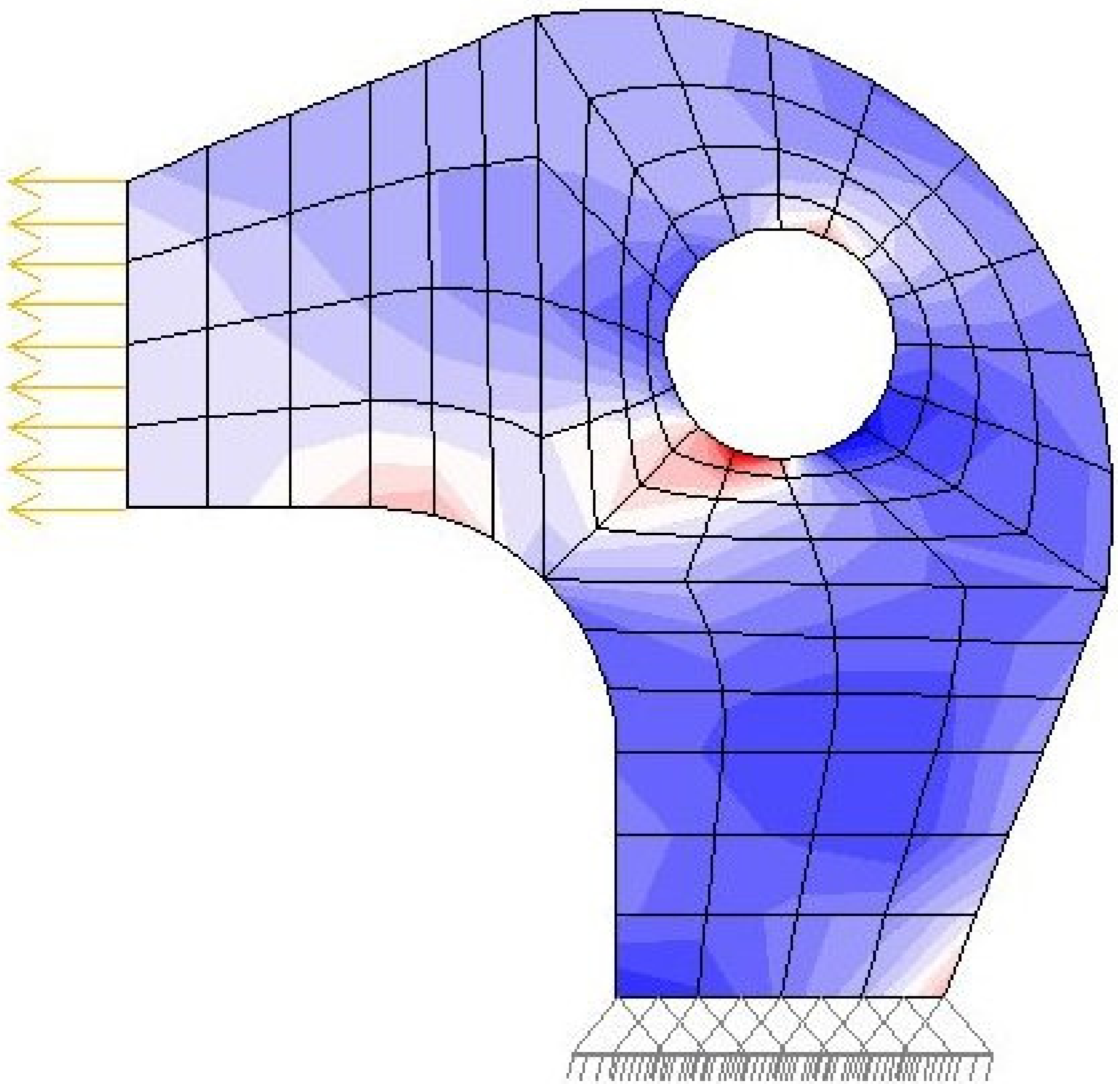
Defining meshing refinement parameters: boundary subdivision



Automatic unstructured mesh generation

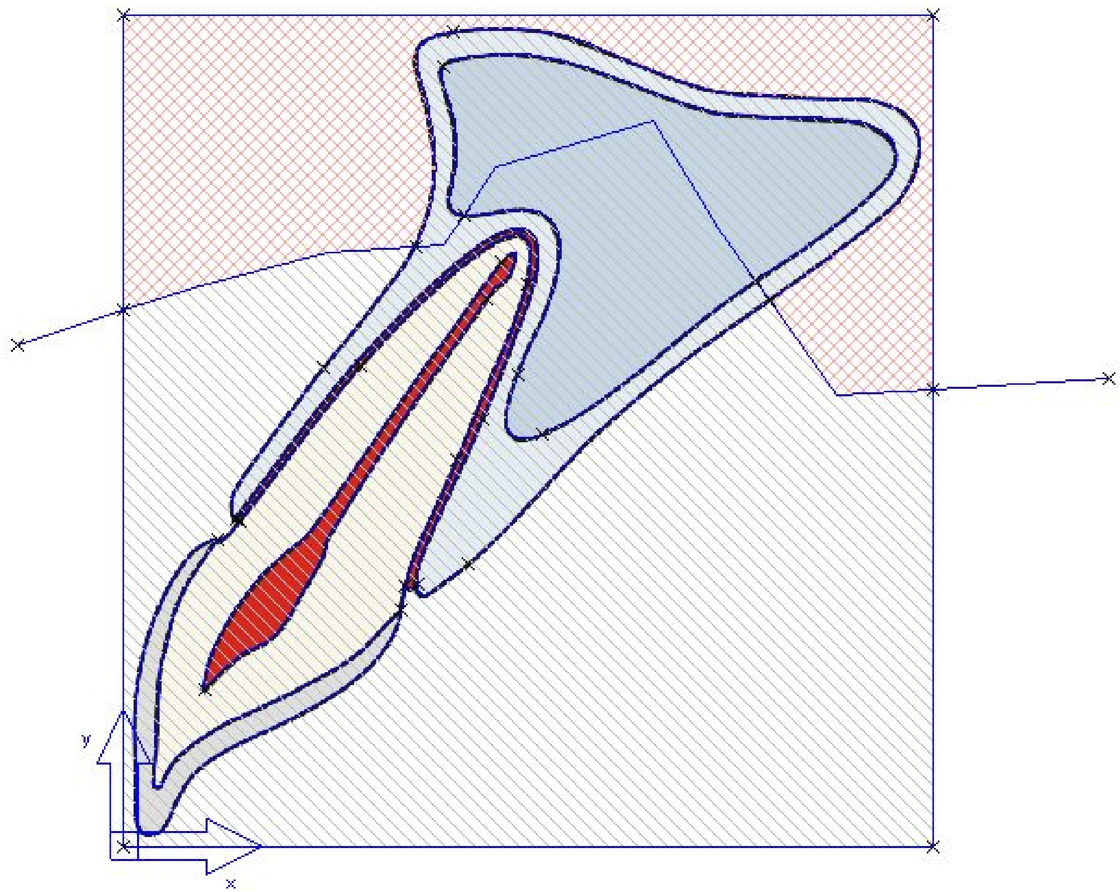


What is the technology behind this?
What issues we have to address?



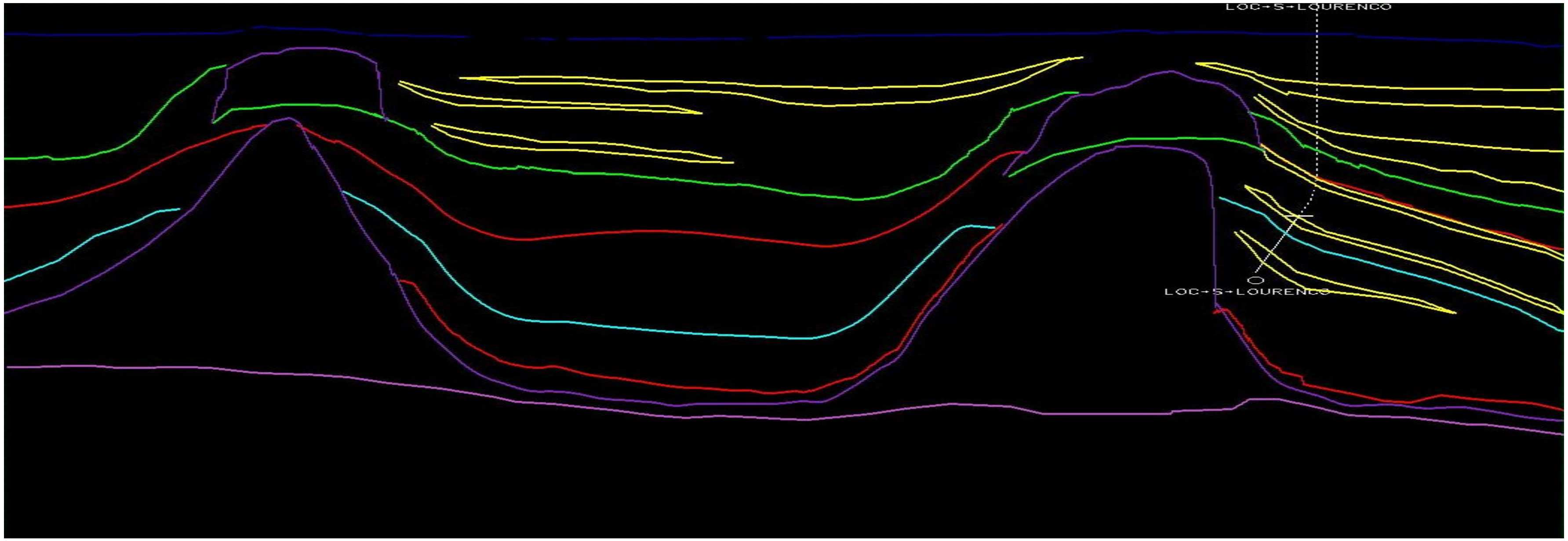
Generic Space Subdivion: Many Applications

An environment in which curves and surfaces are inserted randomly.
Automatic region recognition and full adjacency information.



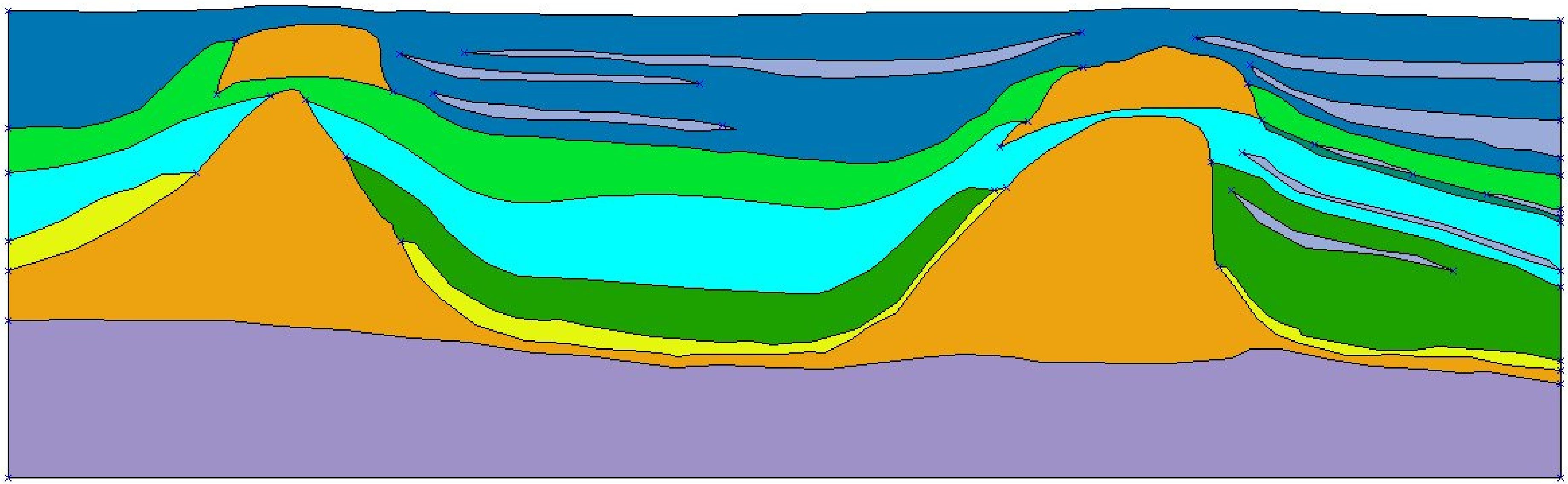
2D Subsurface Simulation Modeling

Sidon-Tiro



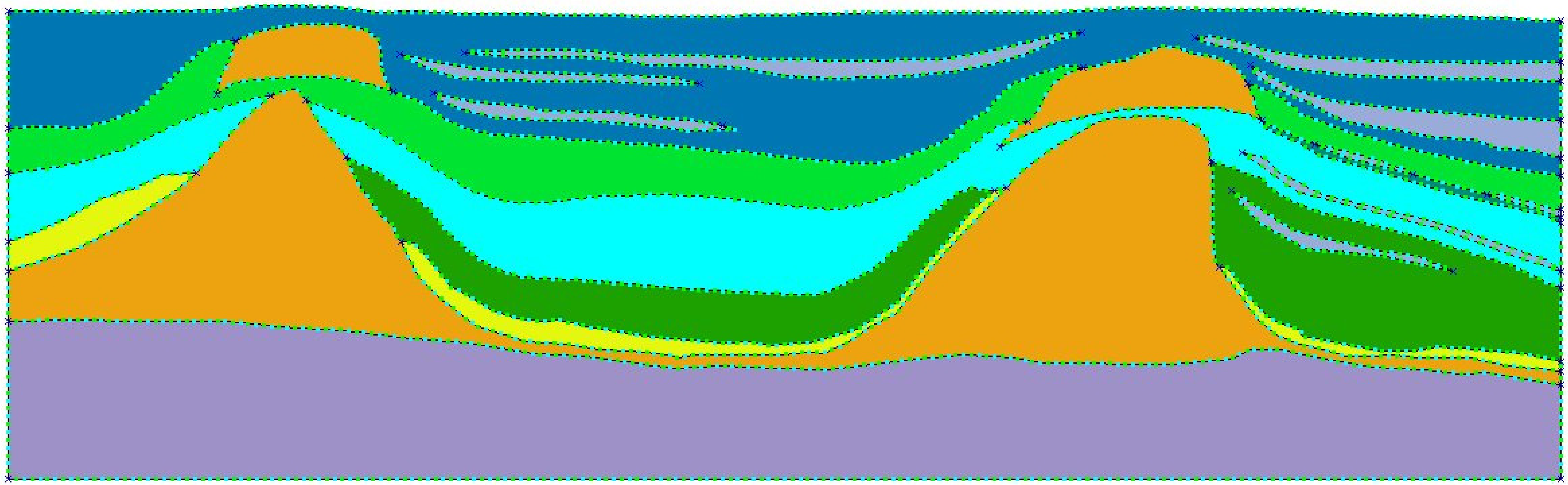
2D Subsurface Simulation Modeling

Curve digitalization



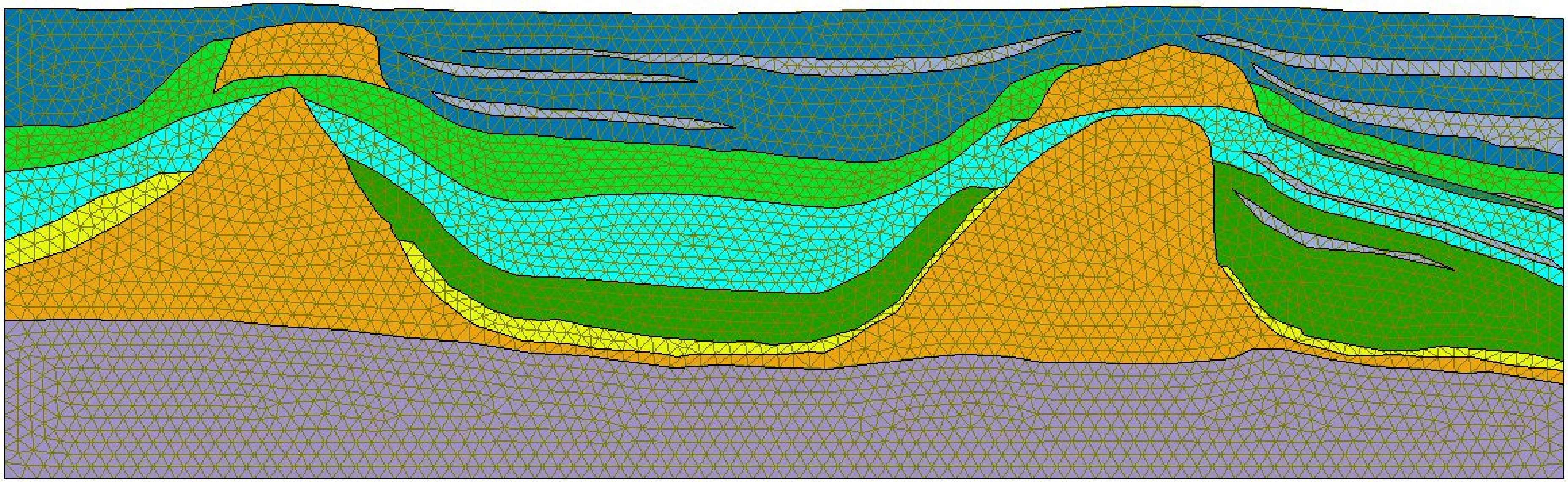
2D Subsurface Simulation Modeling

Curve subdivision



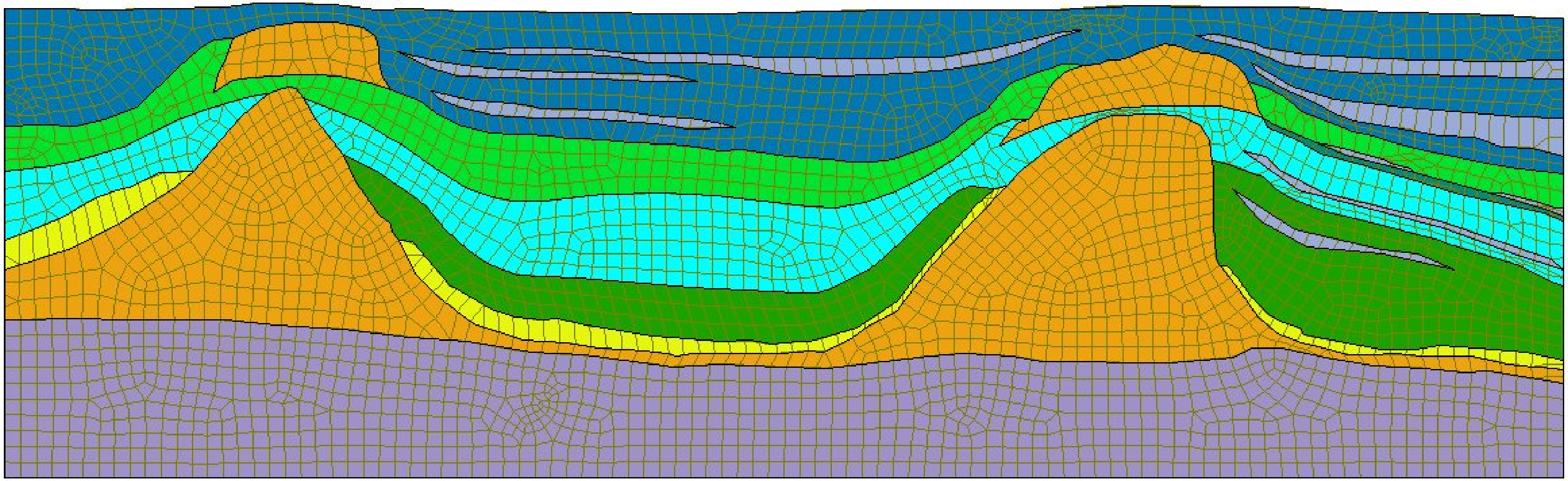
2D Subsurface Simulation Modeling

Mesh generation: triangular elements



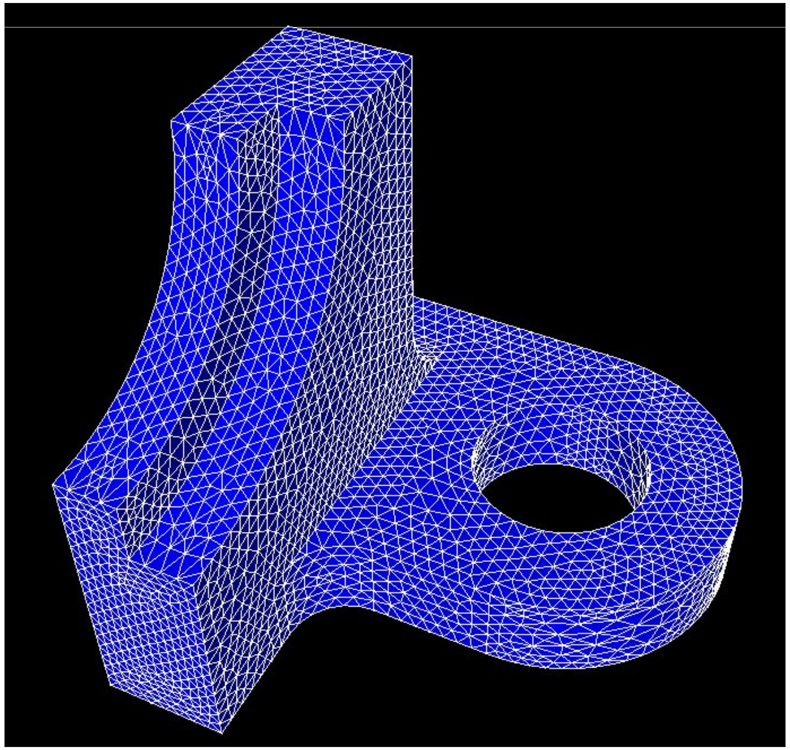
2D Subsurface Simulation Modeling

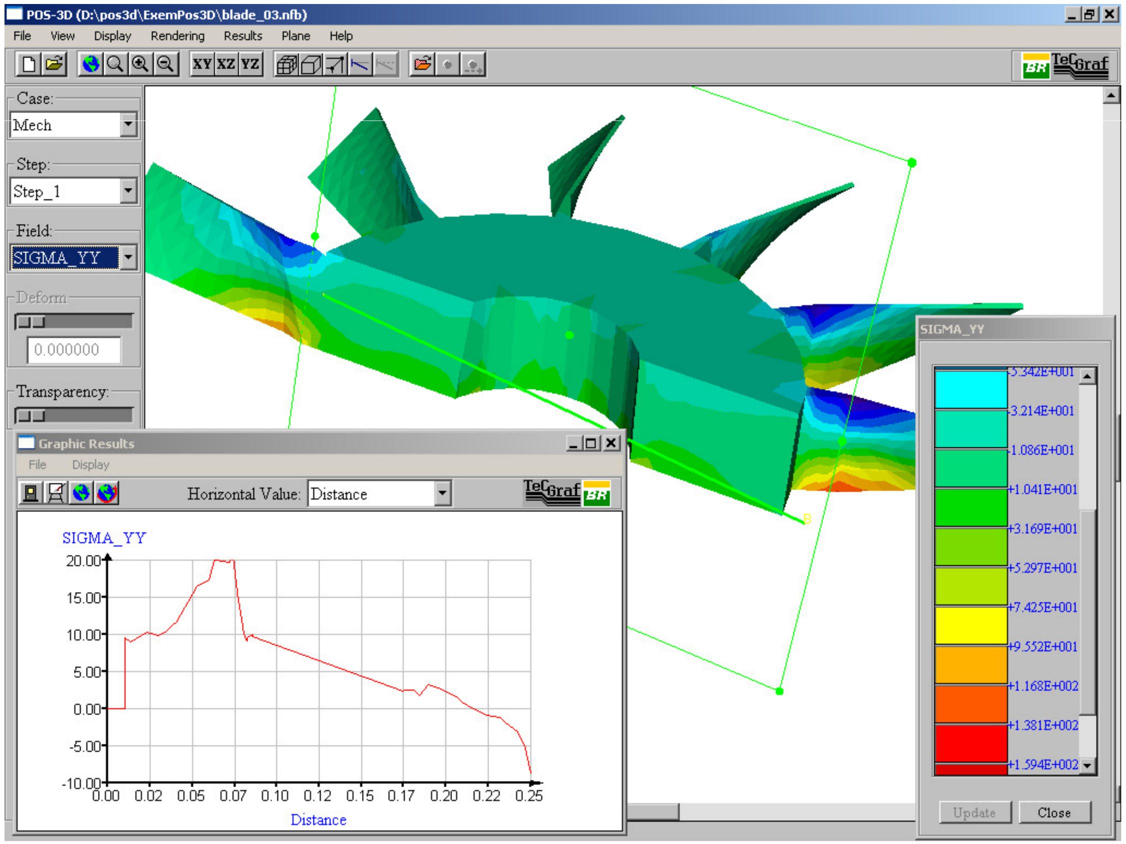
Mesh generation: quadrilateral elements



Requirements for Underlying Data Representation

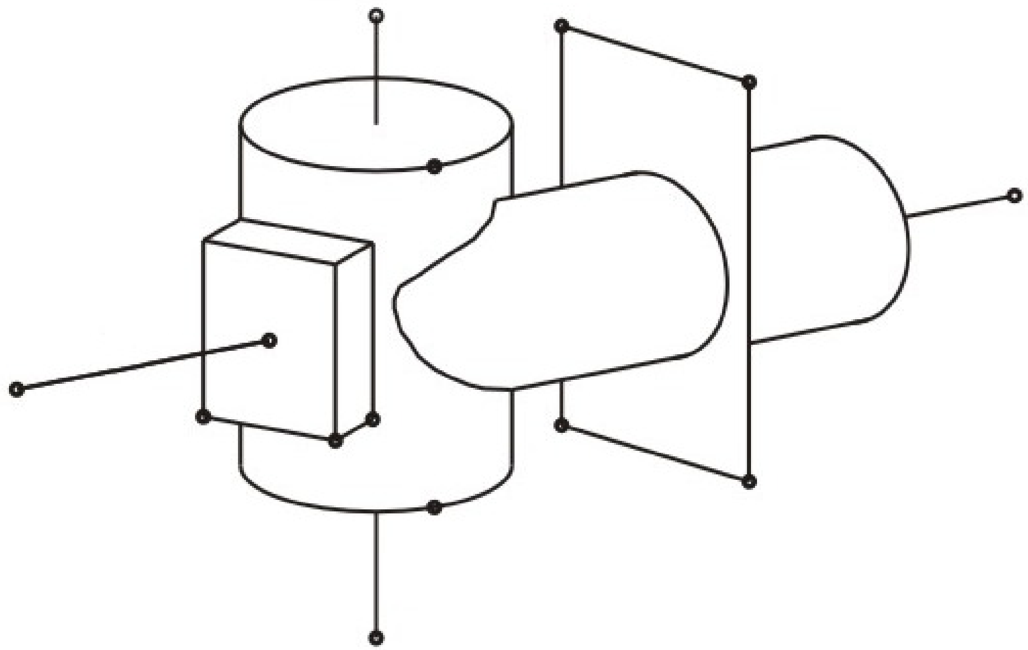
– ***The data structures must provide a natural navigation across all phases of a simulation: pre-processing (model creation), numerical analysis, and post-processing (model results visualization).***





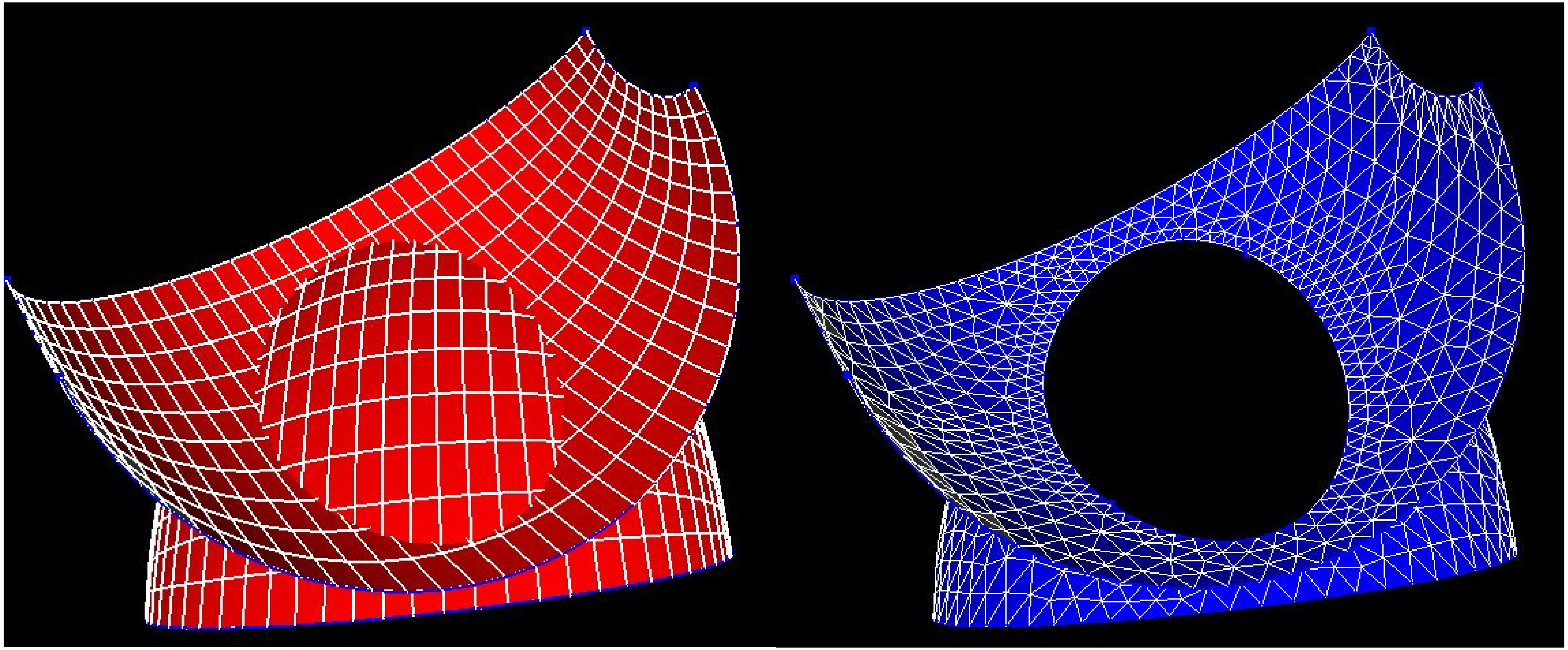
Requirements for Underlying Data Representation

– ***The data structures must take into account that the simulation may induce, at least temporarily during model creation, geometric objects (curves and surfaces) that are inconsistent with the target final model. This requires a non-manifold topology representation capability.***



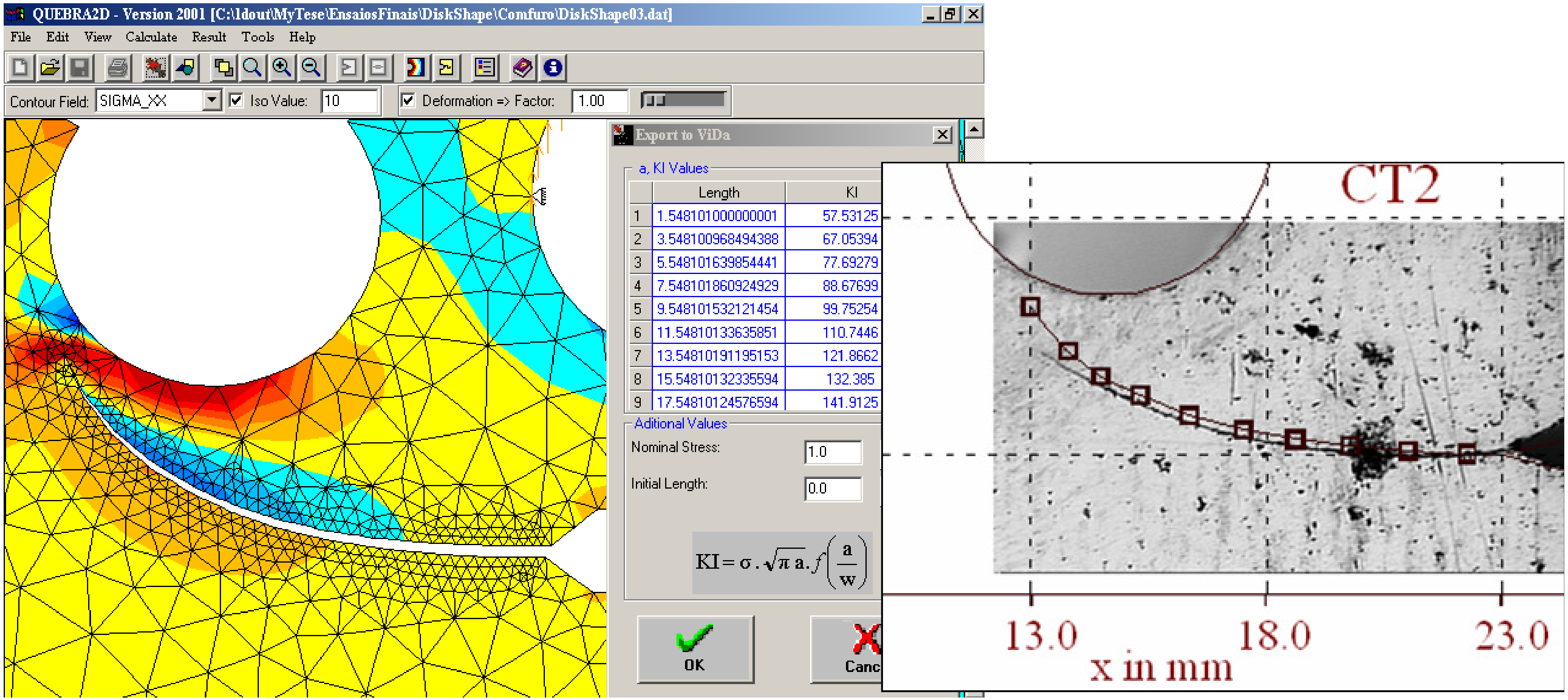
Requirements for Underlying Data Representation

– ***The data structure should aid in key aspects of geometric modeling, such as surface intersection and automatic region recognition, as well as in surface and solid finite element mesh generation in arbitrary domains.***



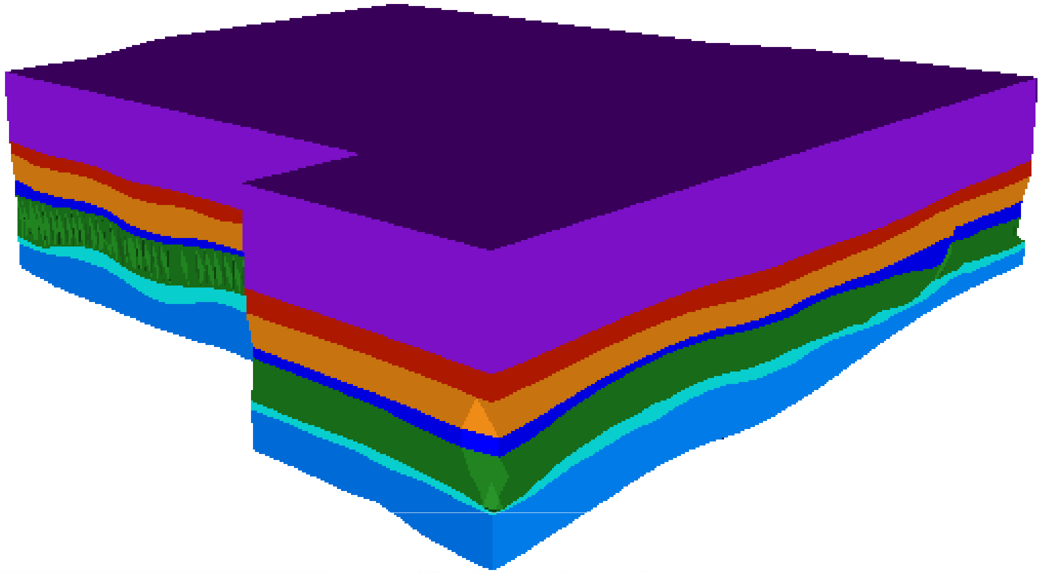
Requirements for Underlying Data Representation

- *The data structure must provide for efficient geometric operators, including automatic intersection detection and processing.*
- *This is necessary in simulations with evolving topology and geometry.*

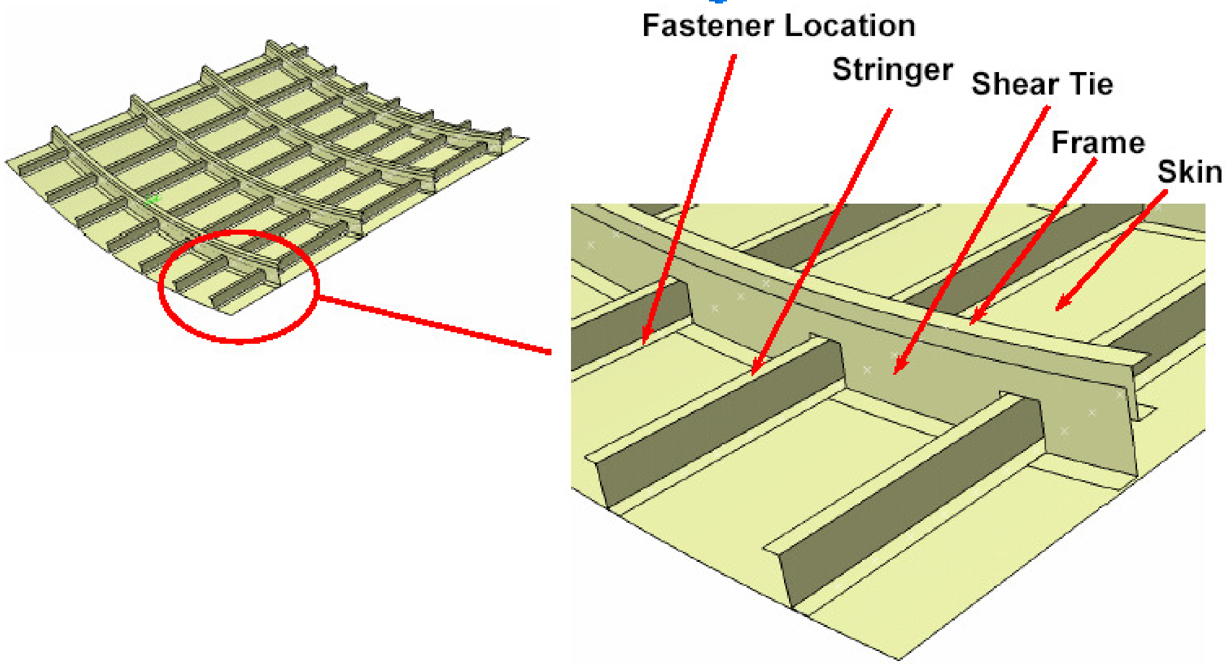


The need for non-manifold modeling

Multi-region modeling

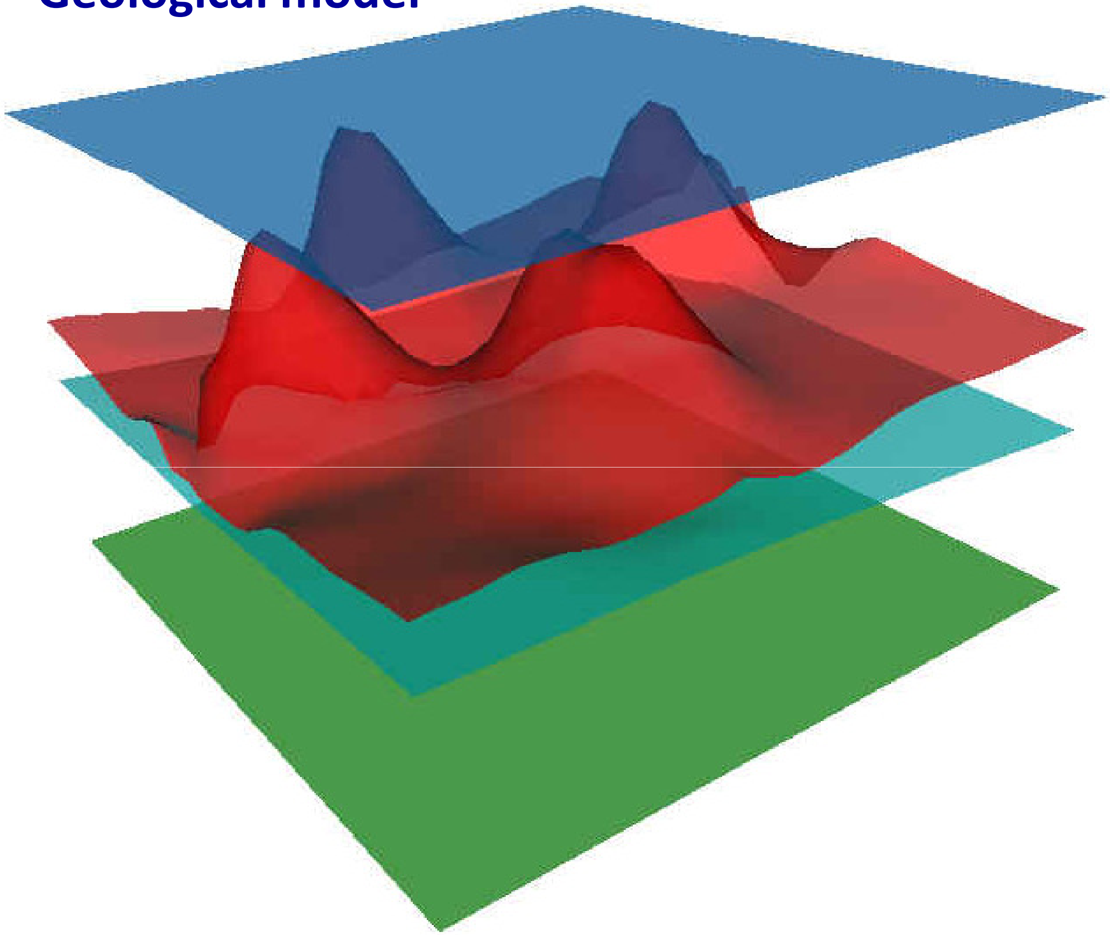


Degenerated structures

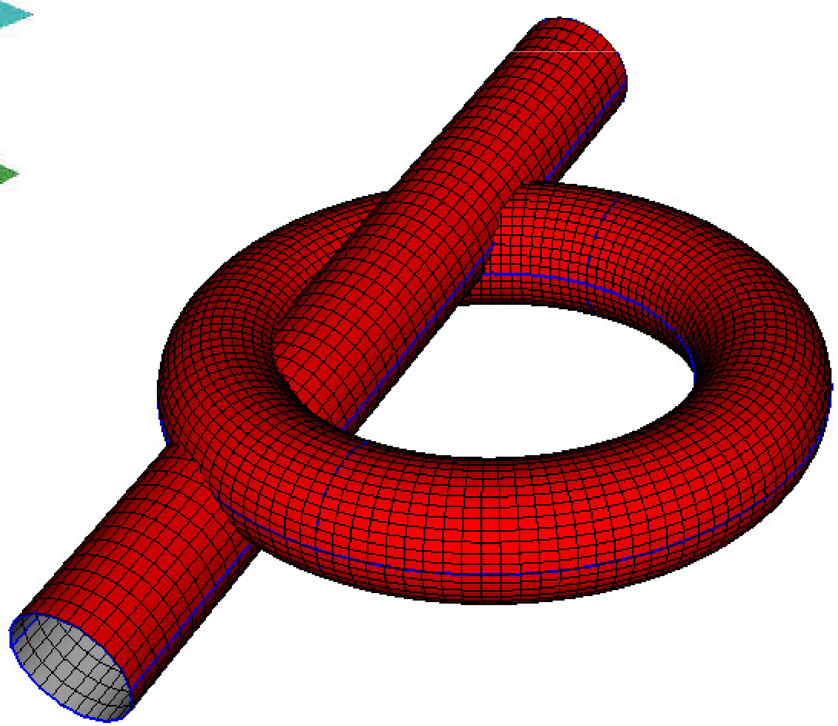


Natural Modeling: surface patches as primitives

Geological model

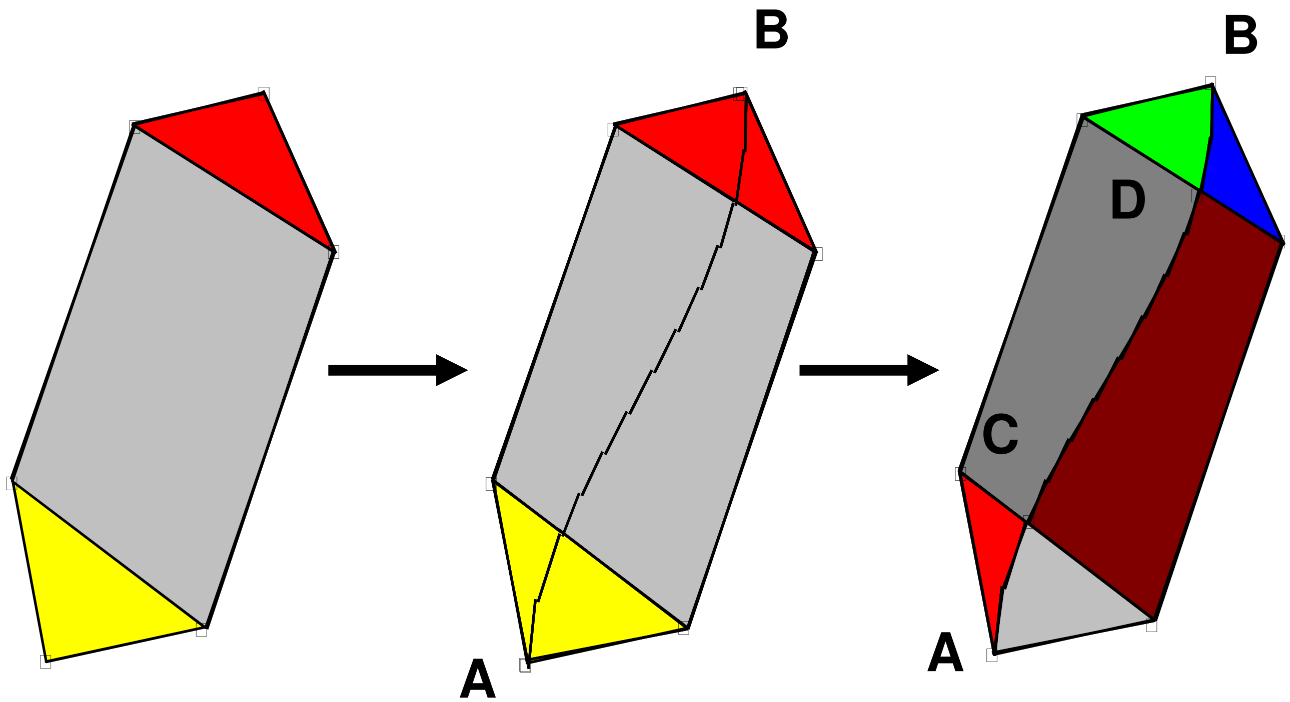


Manufactured model



Ideal Environment: complete space subdivision

Space subdivision in 2D: high level operations



User action
+ basic function

System
response

Modeling in Engineering

Computer Graphics

Research areas:

- Geometric Modeling

  needs due to visual inspections

- Visualization

Representation forms:

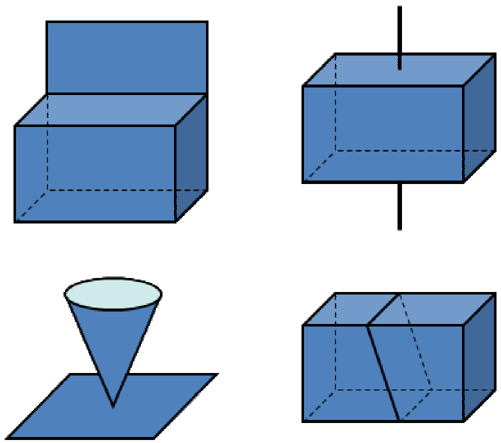
- Wireframe Modeling Technique
- Surface Modeling Technique
- Solid Modeling Technique

are traditionally constrained to work only with two-manifold solids.

Non-manifold Modeling

embods all of the capabilities of the three modeling forms in a unified representation.

allows topological conditions such as



has superior flexibility

but at a cost of a larger size and more complex data structure

Modeling in Engineering

Computer Graphics

Research areas:

- Geometric Modeling

  ↓ *needs due to visual inspections*

- Visualization

Representation forms:

- Wireframe Modeling Technique
- Surface Modeling Technique
- Solid Modeling Technique

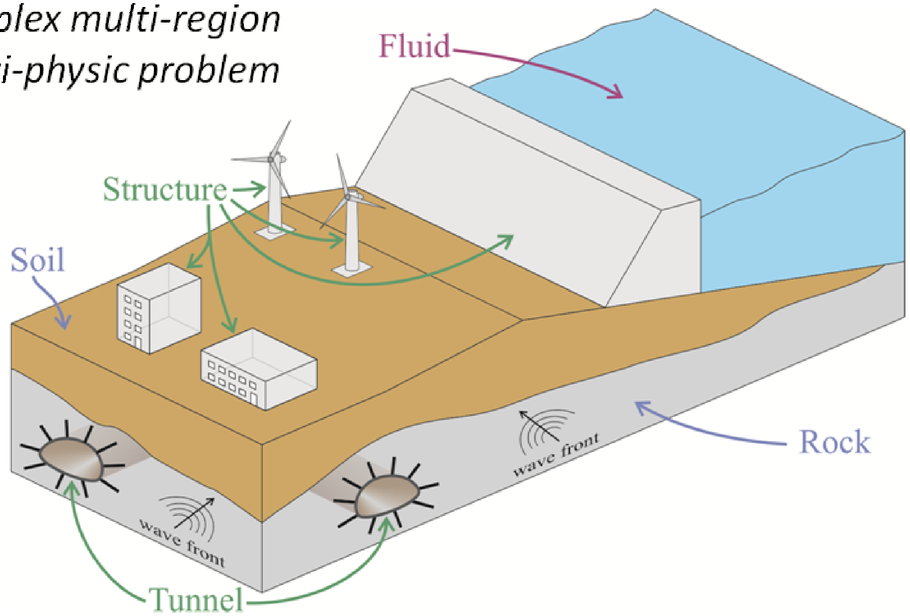
are traditionally constrained to work only with two-manifold solids.

↓

Non-manifold Modeling

embods all of the capabilities of the three modeling forms in a unified representation.

complex multi-region multi-physic problem



Geometric Modeling

Geometric Modeling

- Creation, manipulation, maintenance and analysis of representations of geometric forms of two and three dimensional objects.

- Application in several fields, such as movie production, design of industrial mechanical parts, scientific visualization and reproduction of objects for analysis in engineering.

Geometric Modeling

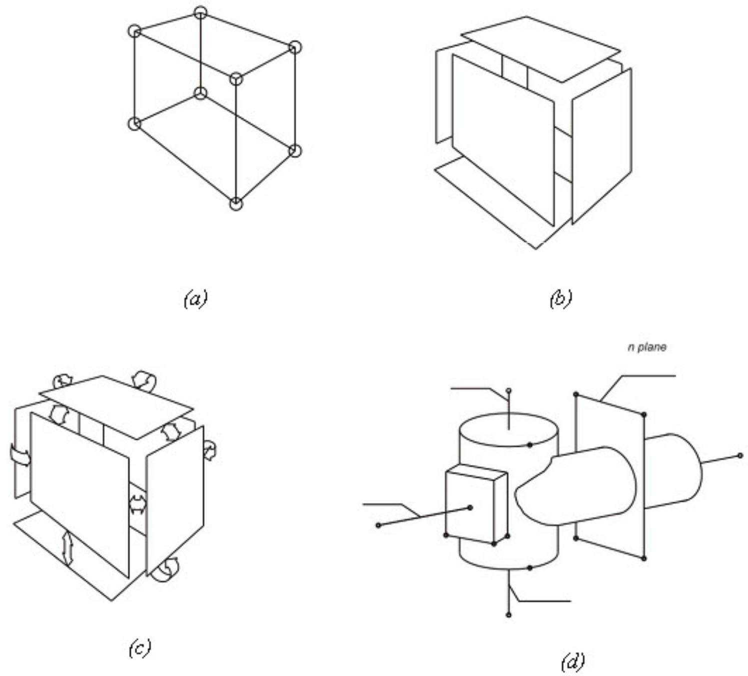
- **Historic Evolution:**

  a) Wire Modeling

  b) Surface Modeling

  c) Solid Modeling

  d) Non-manifold Modeling



Geometric Modeling

- **Strategies for Representing Solids**

  - Decomposition Models

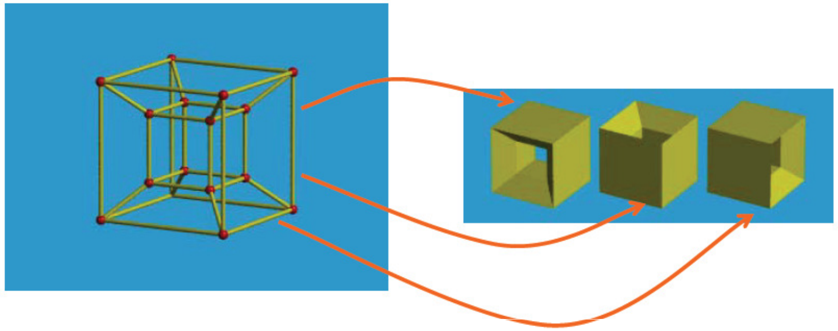
  - B-Rep Models

  - Constructive Models (CSG)

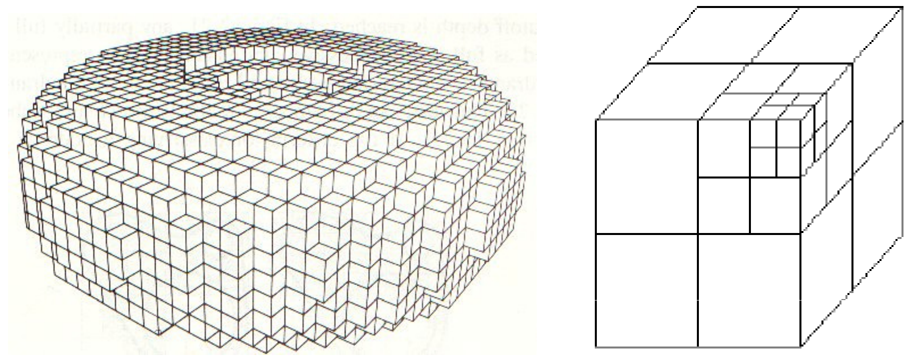
  - Hybrid Models

Geometric Modeling

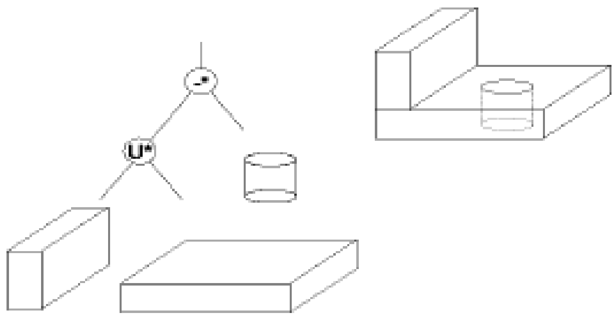
Wire Frame



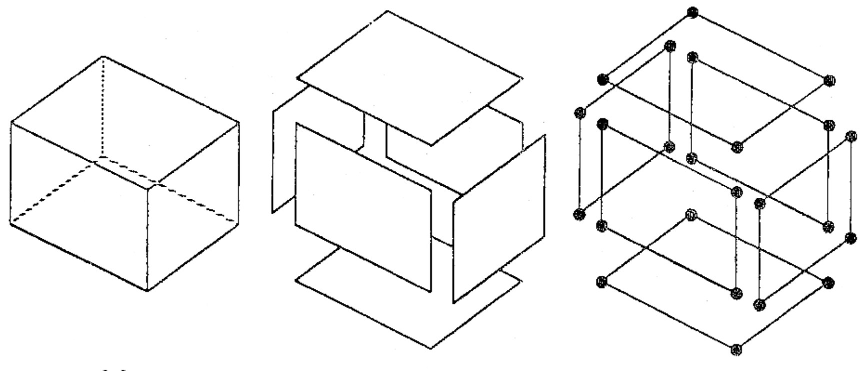
Cell Decomposition / Space Enumeration



Constructive Solid Geometry (CSG)

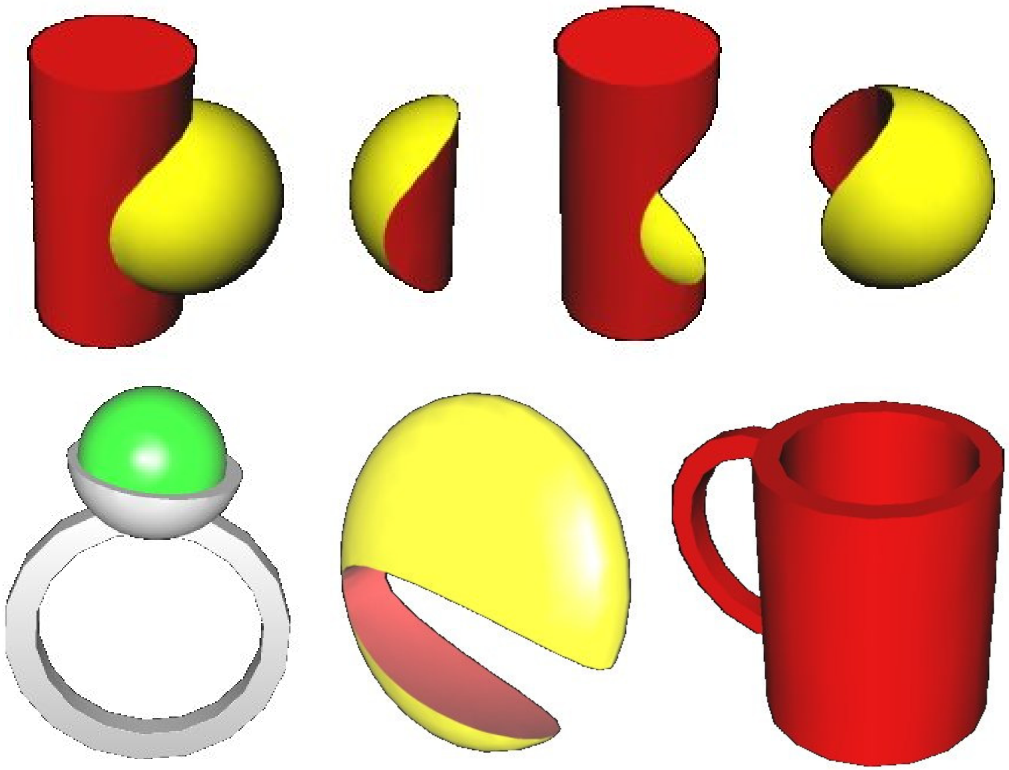


Boundary Representation (B-Rep)



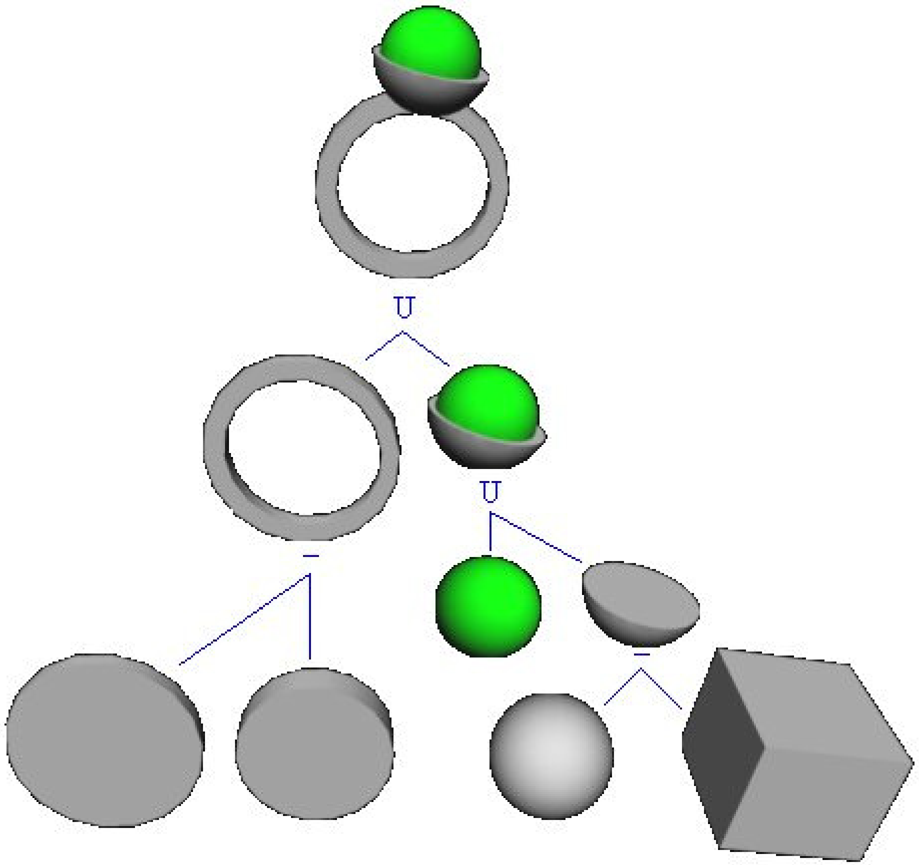
Geometric Modeling

- The Constructive Solid Geometry (CSG) uses Boolean operations and rigid body motions into simple primitives to build more complex solid objects.



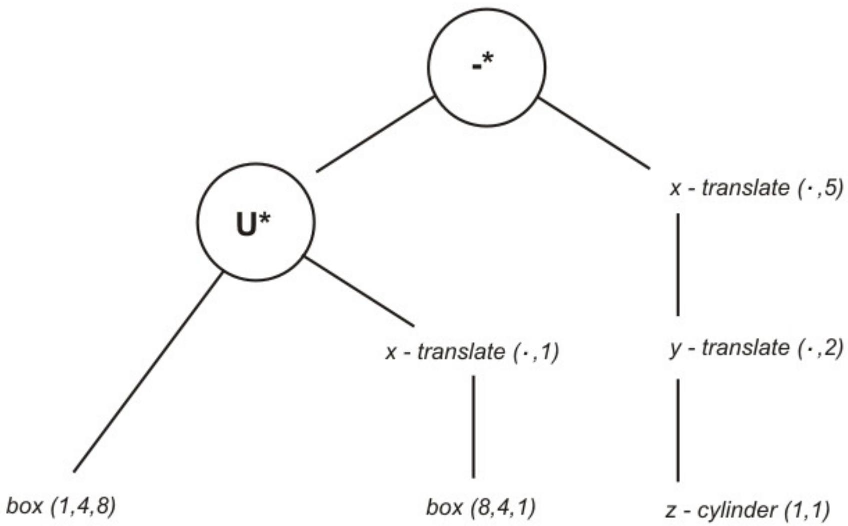
Geometric Modeling

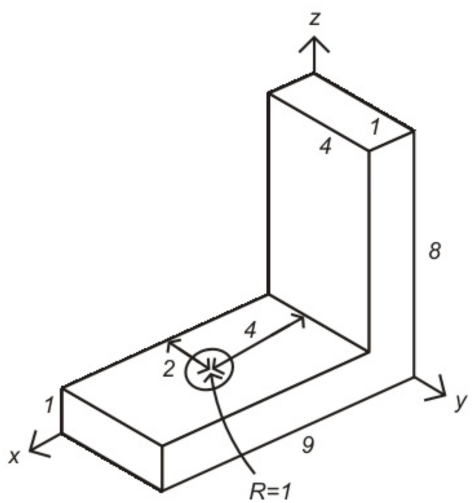
- **CSG Tree**



Geometric Modeling

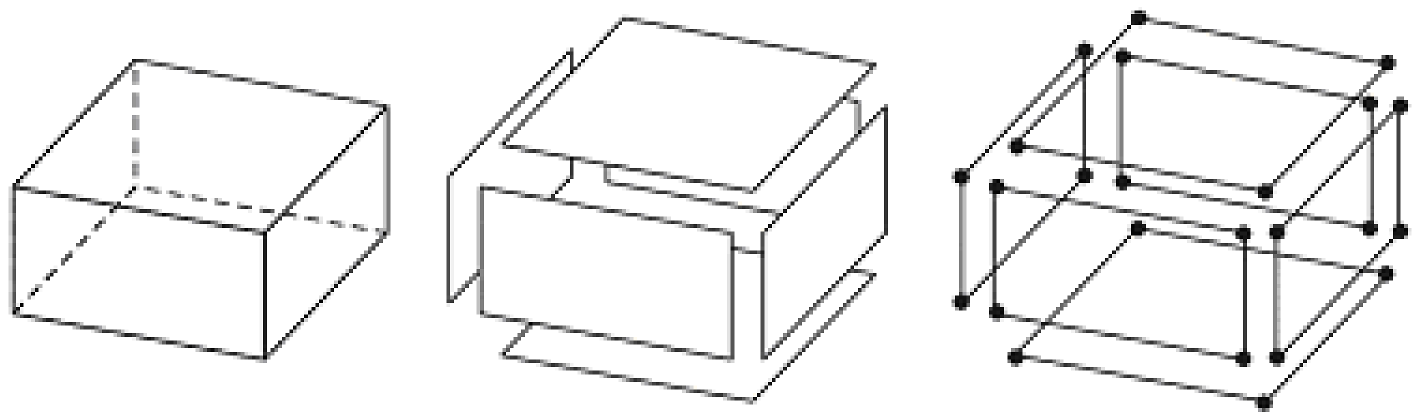
- **CSG Tree**





Geometric Modeling

- B-Rep models explicitly use the adjacency relations among the topological elements (vertices, edges and faces) to define the topological boundary of the objects.



Geometric Modeling

- **Non-manifold Modeling**

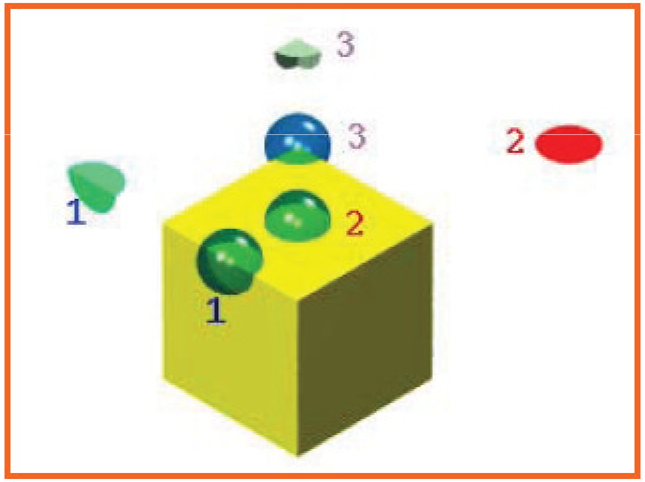
  - Aggregates all the capabilities of the previous three types of modeling.

  - Removes restrictions to the domain of the analyzed models.

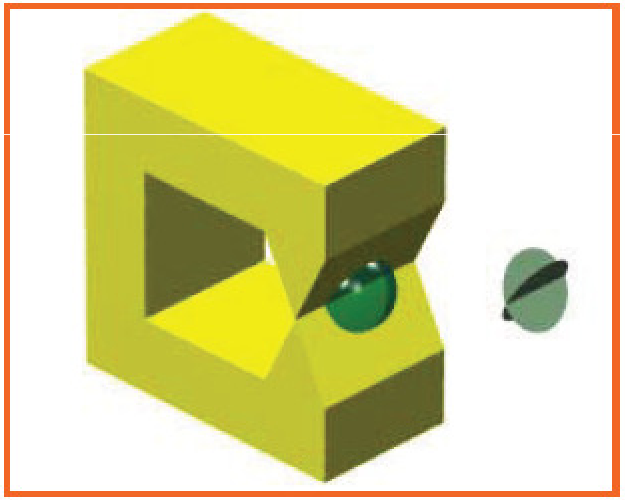
  - Allows the representation of internal or dangling structures of lower dimensions.

Geometric Modeling

Manifold



Non-manifold



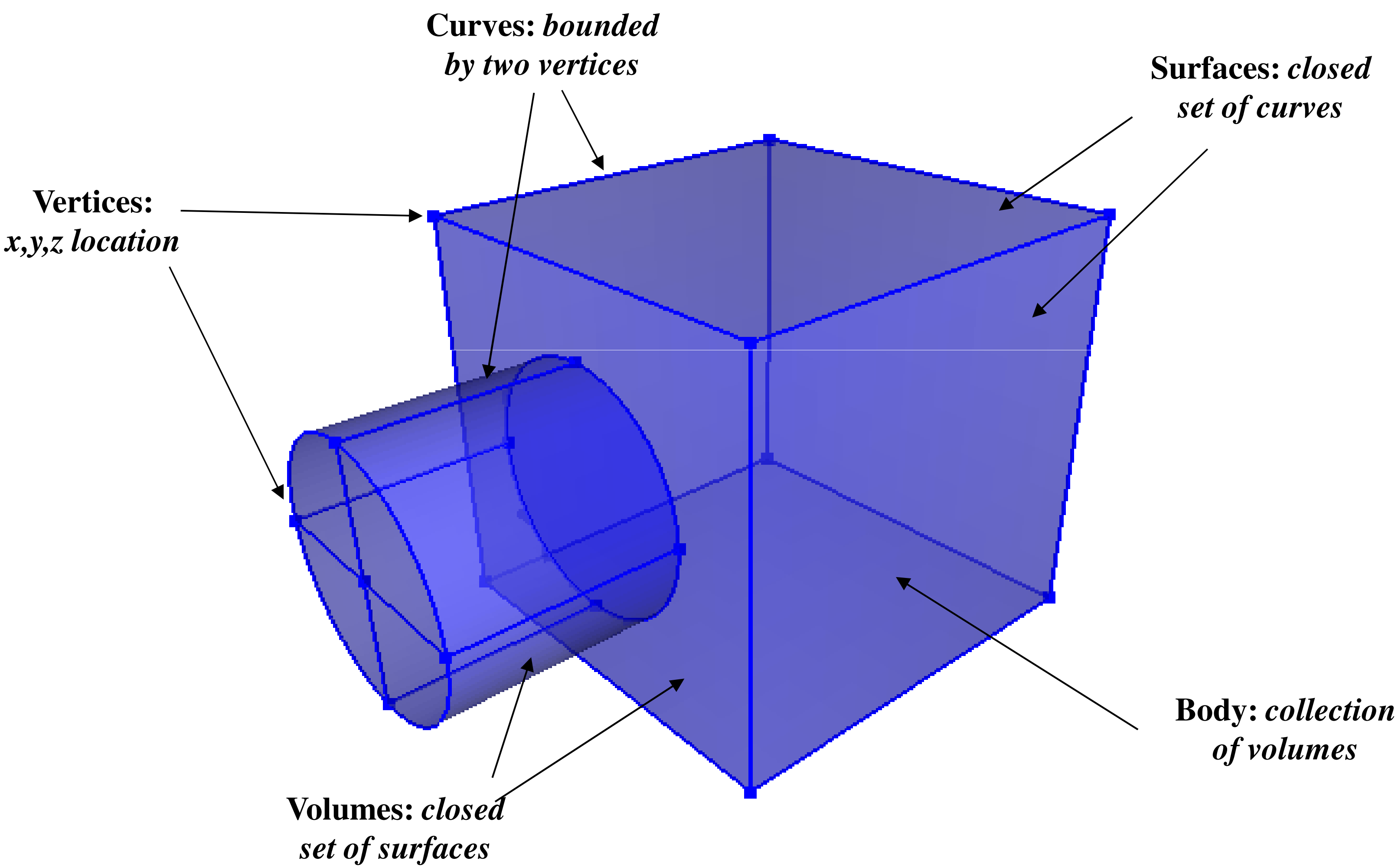
Geometric Modeling

- **Topology and Geometry**

  - **Geometry –** set of complete and essential information to define the shape and spatial location of objects.

  - **Topology –** subset of information obtained from the geometry of the object. Invariant after applying geometric transformations to the object.

Geometric and Topological Entities



Geometric Modeling

- **Using the topology as basis for a modeling system:**

  1) Stability of the System

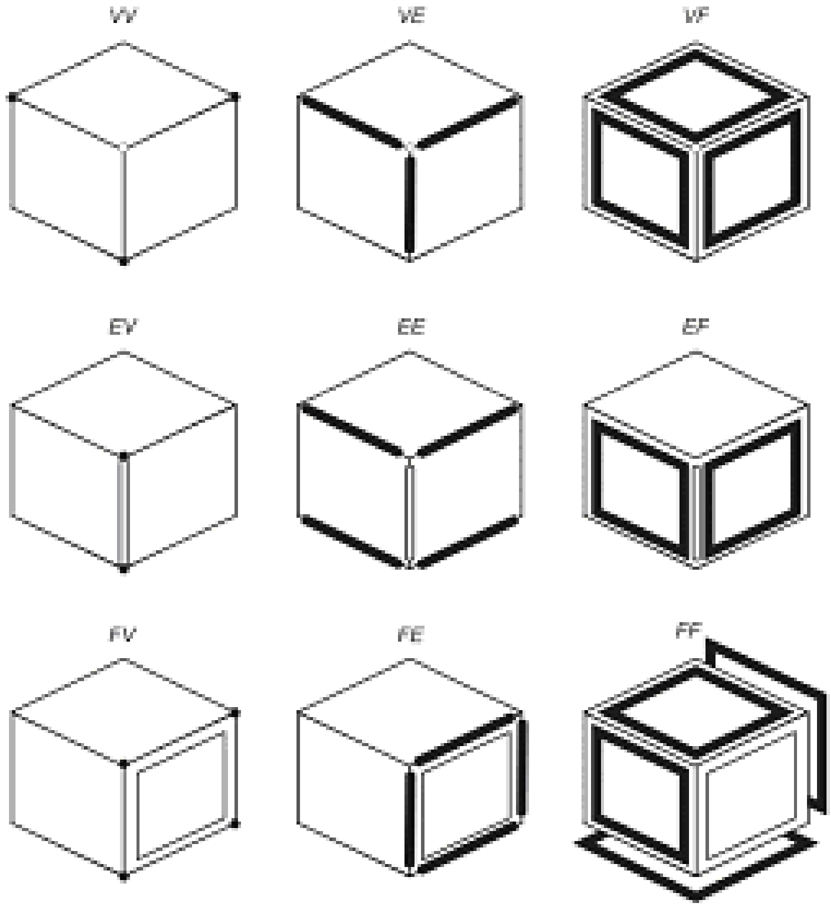
  2) Avoiding numerical errors

  3) Separation of geometric and topological information

Geometric Modeling

- **Adjacency Relationships**
  - ➢ Connectivity among the topological elements
  - ➢ Extracted from the geometric information of the model
  - ➢ Use as a base of modeling framework, ensuring the implementation of algorithms simpler and more efficient
  - ➢ Determination of a minimal set of sufficient adjacency relationships

Geometric Modeling



Adjacency relation among vertices, edges and faces

Geometric Modeling

- **Topological Data Structures**
  - Systematization and organization of topological information of a model from the storage of a sufficient set of adjacency relations.
  - Main topological elements: vertices, edges and faces.
  - Additional topological elements: loops, shells, regions, vertex uses, half-edges, edge uses, loop uses, face uses.

Geometric Modeling

- **Topological Data Structures**

  - Examples of data structures established in *manifold* modeling:
    - *Winged-edge*
    - *Half-edge*

  - Data structures established in *non-manifold* modeling:
    - *Radial Edge*

Winged-Edge Topological Data Structure

Winged-Edge (Baumgart, 1972)

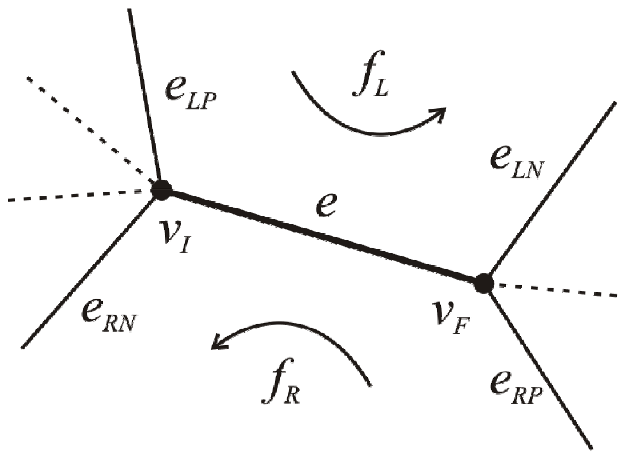


Table of Vertexes

v	x	y	z	e_I

Table of Faces

f	e_I

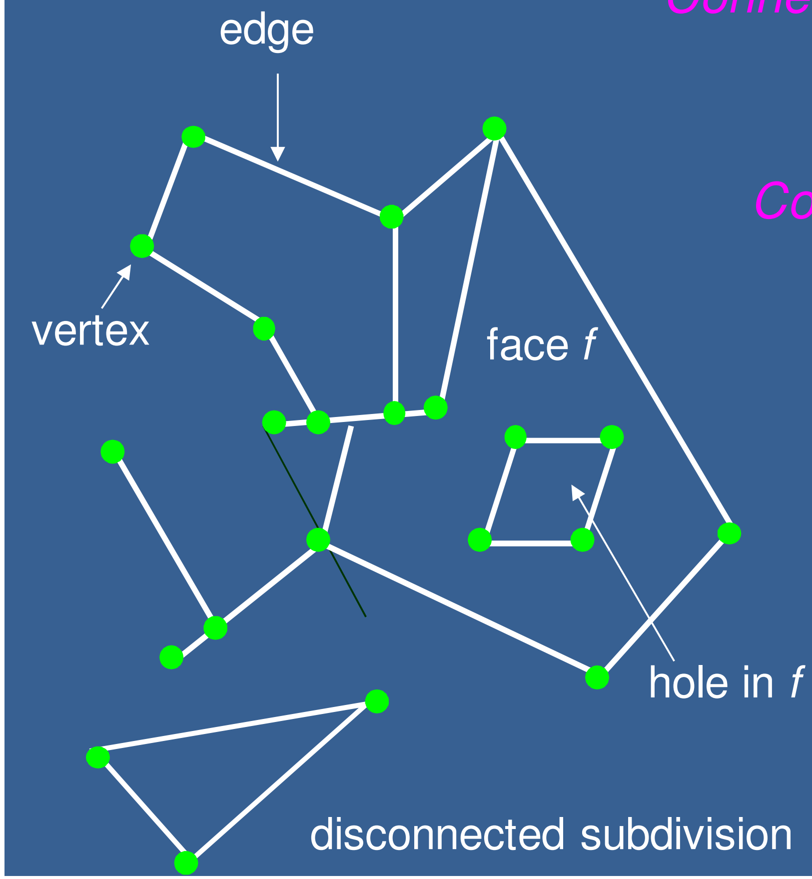
Table of Edges

e	v_I	v_F	f_L	f_R	e_{LP}	e_{LN}	e_{RP}	e_{RN}

Topological Data Structure - Planar Subdivision

Induced by planar embedding of a graph.

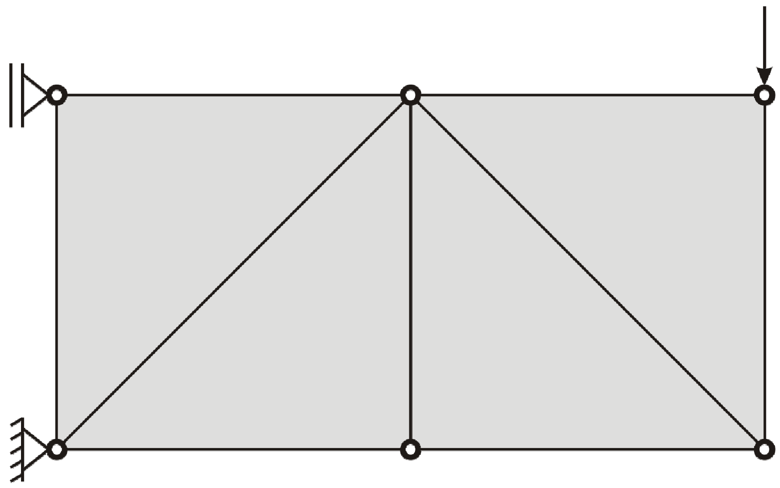
Connected if the underlying graph is.



Complexity = #vertices + #edges + #faces

Typical operations:

- Walk around a face.
- Access one face from an adjacent one via a common edge.
- Visit all the edges adjacent to a vertex.



Legend:

geometry

point

curve

surface

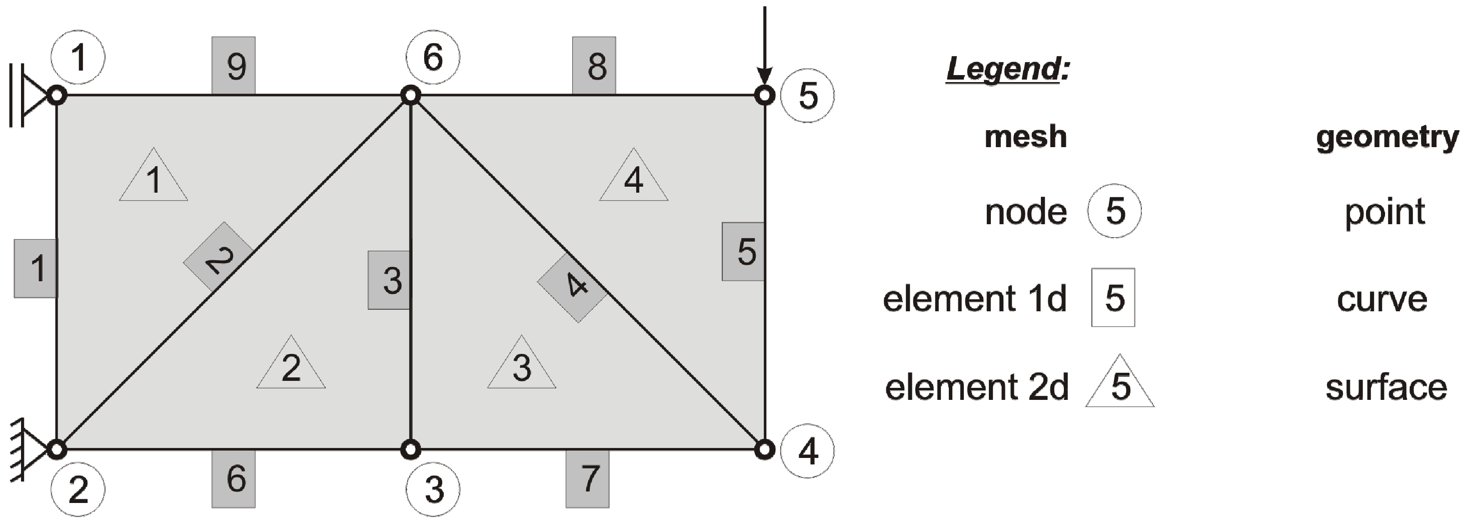


Table of nodes

N	x	y	z
1	0	1	0
2	0	0	0
3	1	0	0
4	2	0	0
5	2	1	0
6	1	1	0

Table of Incidences

E	N_1	N_2	N_3
1	1	2	6
2	2	3	6
3	6	3	4
4	4	5	6

E	N_1	N_2
1	1	2
2	2	6
3	6	3
4	6	4
5	4	5
6	2	3
7	3	4
8	5	6
9	6	1

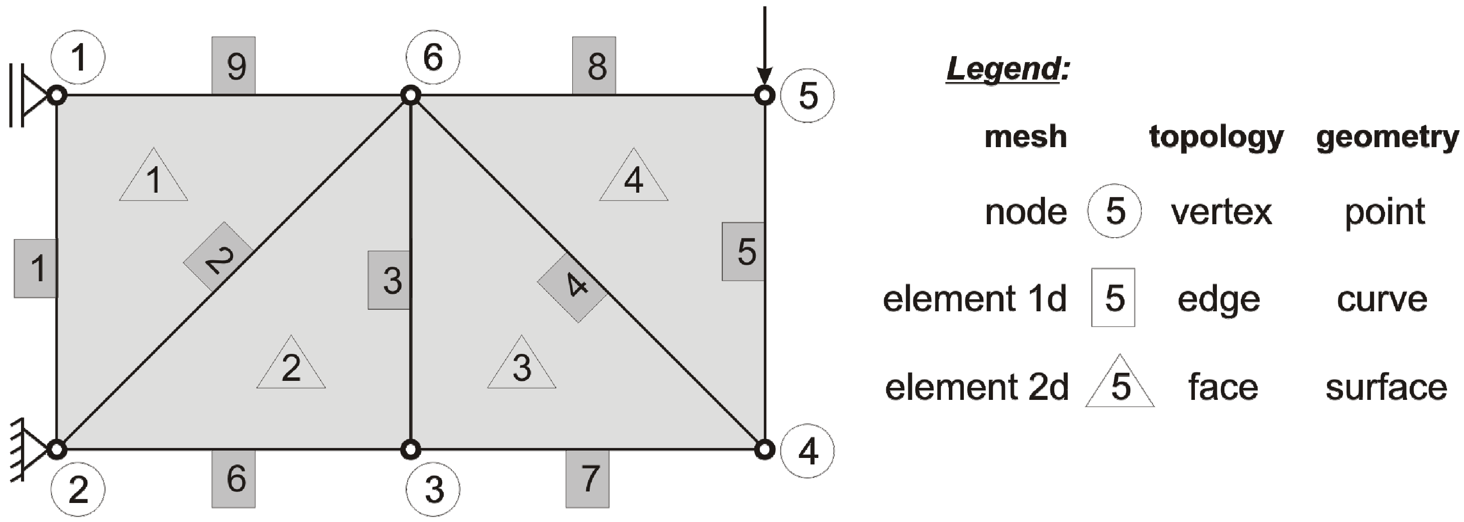


Table of nodes

N	x	y	z
1	0	1	0
2	0	0	0
3	1	0	0
4	2	0	0
5	2	1	0
6	1	1	0

Table of Incidences

E	N_1	N_2	N_3
1	1	2	6
2	2	3	6
3	6	3	4
4	4	5	6

E	N_1	N_2
1	1	2
2	2	6
3	6	3
4	6	4
5	4	5
6	2	3
7	3	4
8	5	6
9	6	1

Table of vertexes

v	x	y	z	e_I
1	0	1	0	1
2	0	0	0	6
3	1	0	0	7
4	2	0	0	4
5	2	1	0	8
6	1	1	0	2

Table of faces

f	e_I
0	1
1	2
2	6
3	4
4	4

Table of edges

e	v_I	v_F	f_L	f_R	e_{LP}	e_{LN}	e_{RP}	e_{RN}
1	1	2	1	0	9	2	6	9
2	2	6	1	2	1	9	3	6
3	6	3	3	2	4	7	6	2
4	6	4	4	3	8	5	7	3
5	4	5	4	0	4	8	8	7
6	2	3	2	0	2	3	7	1
7	3	4	3	0	3	4	5	6
8	5	6	4	0	5	4	9	5
9	6	1	1	0	2	1	1	8

Euler Operators

From a topological viewpoint, the simplest solids are those that have a closed orientable surface and no holes or interior voids. We assume that each face is bounded by a single loop of adjacent vertices; that is, the face is homeomorphic to a closed disk. Then the number of vertices V, edges E, and faces F of the solid satisfy the *Euler* formula:

$$V - E + F - 2 = 0$$

This fact is easily proved by induction on the surface structure. Extensions to this formula have been made that account for faces not being homeomorphic to closed disks, the solid surface not being without holes, and the solid having interior voids, as reviewed next.

[HOFFMANN1992]

We consider the possibility that the solid has holes, but that it remains bounded by a single, connected surface. Moreover, each face is assumed to be homeomorphic to disk. For example, the torus has one hole, and the object in Figure **A** has two. It is a well-known fact that such solids are topologically equivalent, i.e., *homeomorphic*, to a sphere with zero or more handles. For example, the object of Figure **A** is homeomorphic to a sphere with two handles, the latter shown in Figure **B**. The number of handles is called the *genus* of the surface. In general, with a genus G, the numbers of vertices, edges, and faces obey the *Euler–Poincaré* formula:

$$V - E + F - 2(1 - G) = 0$$

[HOFFMANN1992]

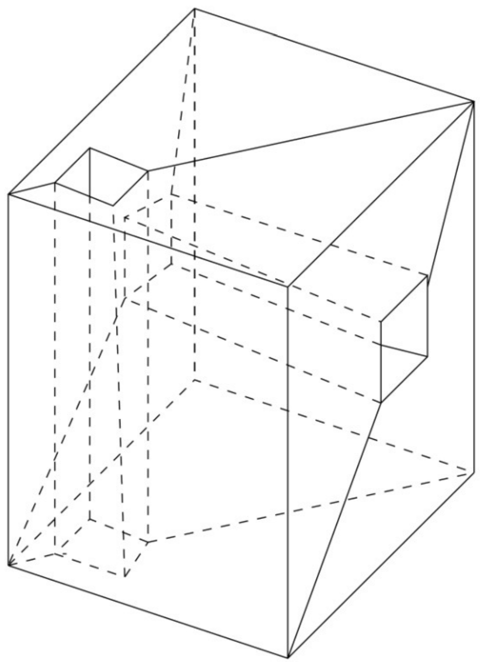


Figure A An Object with Two Holes and with Faces Homeomorphic to Disks

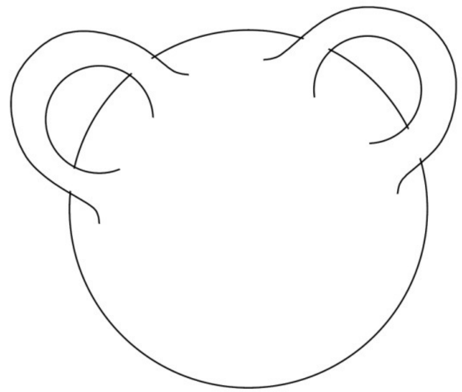
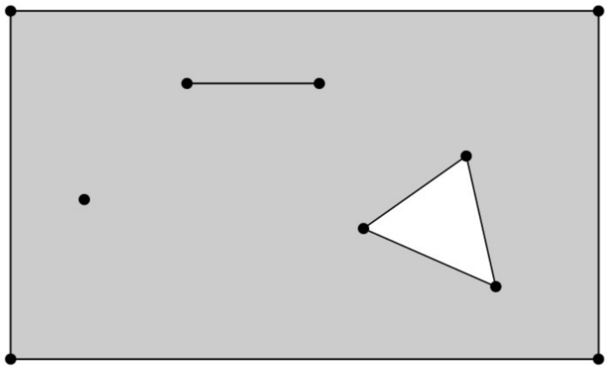


Figure B A Surface of Genus 2

Next, we further generalize by adding the possibility of internal voids. These voids are bounded by separate closed manifold surfaces, called *shells*. The number of shells will be denoted by S. Finally, we relax the requirement that a face is bounded by a single loop of vertices, but require that each face can be mapped to the plane. Thus, a sphere missing at least one point can be a face. In Figure **C**, a face is shown with four bounding loops. Note that one of these loops consists of a single vertex, and another one of two vertices connected by an edge. To account for faces of this complexity, we must count, for each face, the number of bounding vertex loops. For the face in Figure **C**, this number is four. With L the total number of loops, the relationship among the number of faces, edges, vertices, loops, and shells, and the sum G of each shell's genus, is then

$$V - E + F - (L - F) - 2(S - G) = 0$$

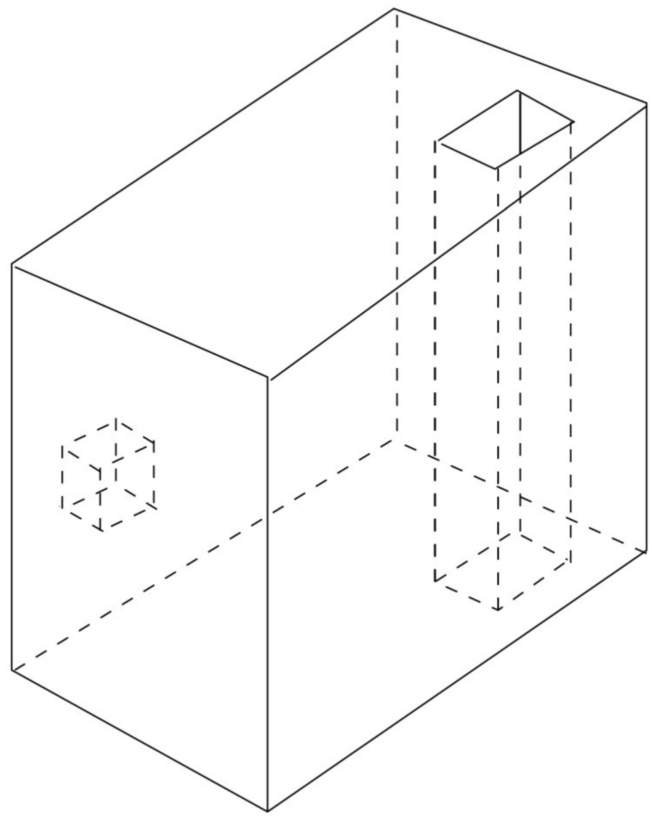


[HOFFMANN1992]

Figure C A Face with Four Bounding Loops

An example solid illustrating this relationship is shown in Figure D.

We may think of the quantities V, E, F, L, S, and G as existing in an abstract six-dimensional space. The relationship among them is then the equation of a hyperplane. Since the values of the variables must be non-negative integers, we might view the relation as defining a lattice on this hyperplane. For each solid with a given topological structure, there corresponds a point in this lattice.



[HOFFMANN1992]

Figure D Solid with 24 Vertices, 36 Edges, 16 Faces, 18 Loops, 2 Shells, and Genus Sum 1

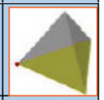
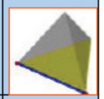
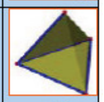
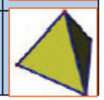
Euler Operators

Operator Name	Meaning	V	E	F	L	S	G
MEV	Make an edge and a vertex	+1	+1				
MFE	Make a face and an edge		+1	+1	+1		
MSFV	Make a shell, a face and a vertex	+1		+1	+1	+1	
MSG	Make a shell and a hole					+1	+1
MEKL	Make an edge and kill a loop		+1		-1		

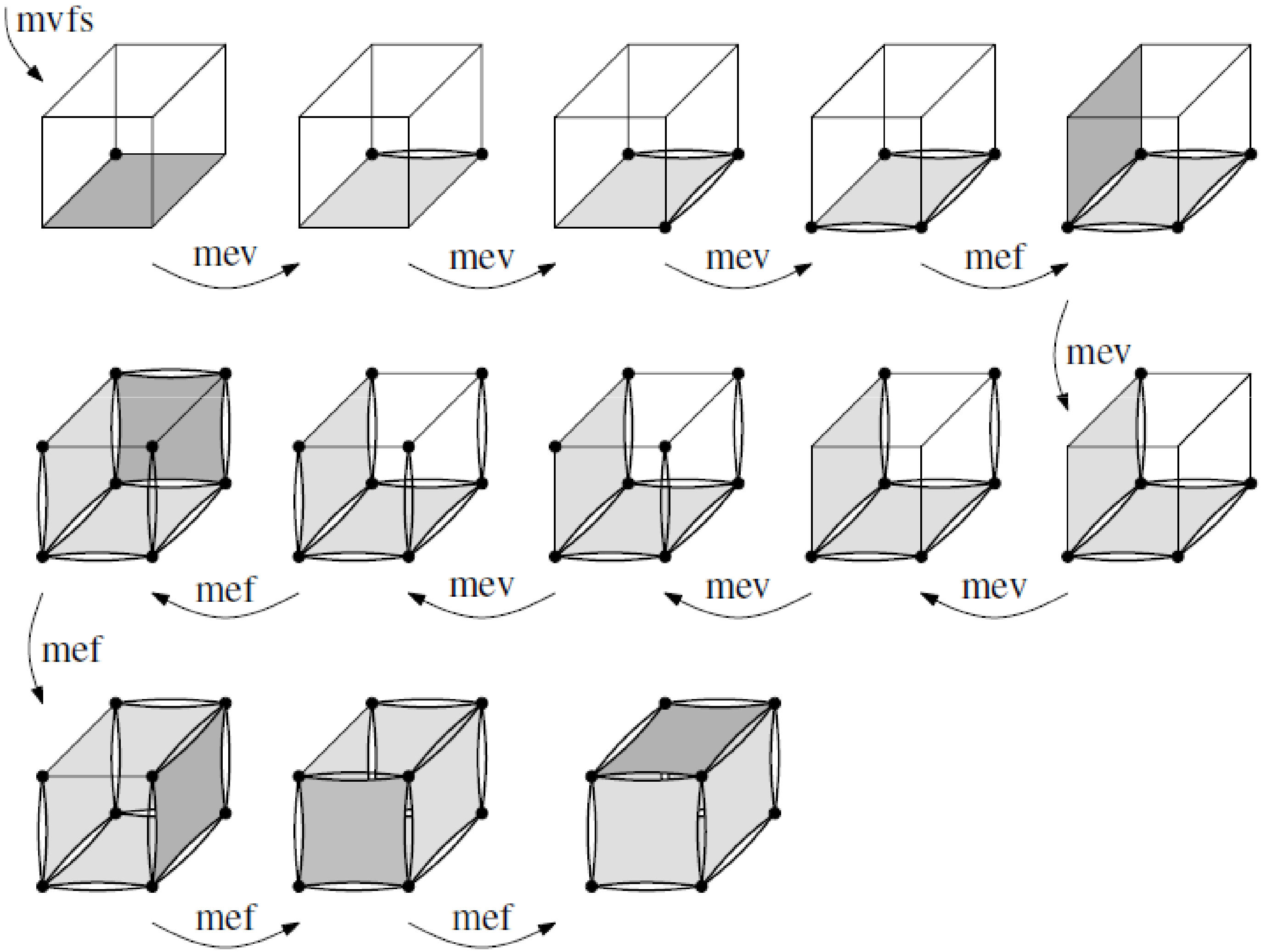
$$V - E + F - (L - F) - 2(S - G) = 0$$

Euler Operators

Operator Name	Meaning	V	E	F	L	S	G
MEV	Make an edge and a vertex	+1	+1				
MFE	Make a face and an edge		+1	+1	+1		
MSFV	Make a shell, a face and a vertex	+1		+1	+1	+1	
MSG	Make a shell and a hole					+1	+1
MEKL	Make an edge and kill a loop		+1		-1		

Operator Name	*Meaning*	*V*	*E*	*F*	*L*	*S*	*G*	**Result**
MSFV	**Make a shell, a face and a vertex**	**+1**		**+1**	**+1**	**+1**		
MEV	**Make an edge and a vertex**	**+1**	**+1**					
MEV	**Make an edge and a vertex**	**+1**	**+1**					
MEV	**Make an edge and a vertex**	**+1**	**+1**					
MFE	**Make a face and an edge**		**+1**	**+1**		**+1**		
MFE	**Make a face and an edge**		**+1**	**+1**		**+1**		
MFE	**Make a face and an edge**		**+1**	**+1**		**+1**		

Using Euler Operators to Construct a Solid



Half-Edge Topological Data Structure

Half-Edge (Mäntylä, 1988)

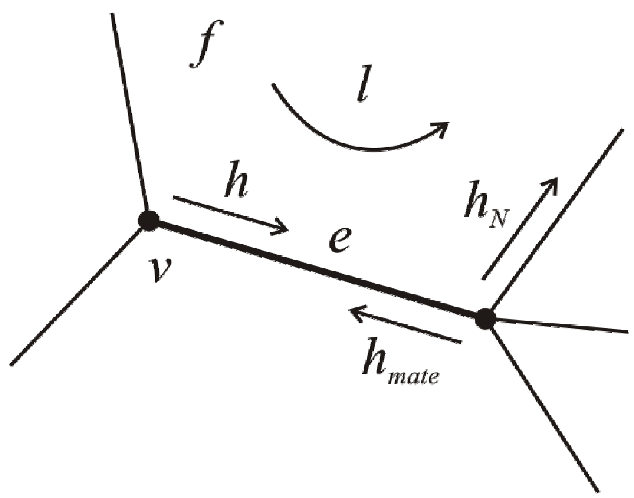


Table of vertexes

v	x	y	z	h

Table of half-edges

h	e	v	l	h_N

Table of edges

e	h_1	h_2

Table of loops

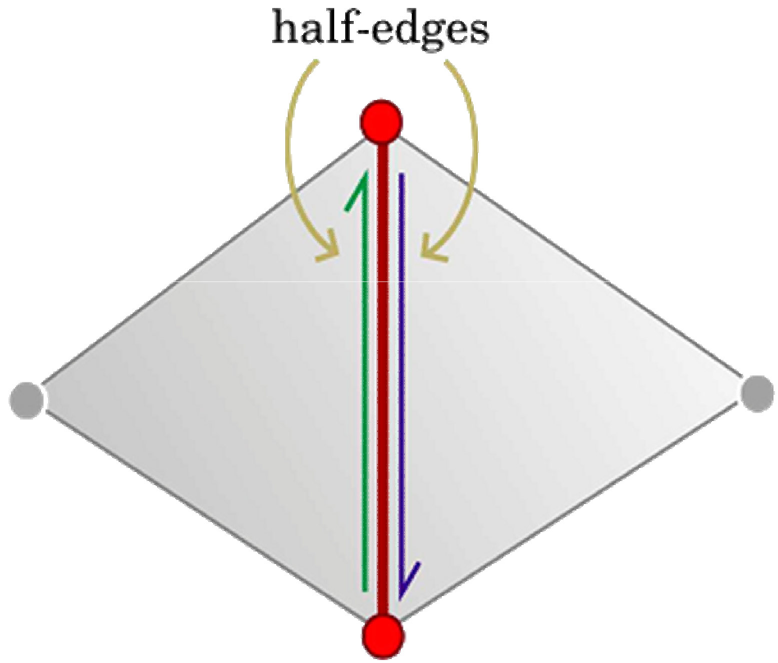
l	h	f	l_N

Table of faces

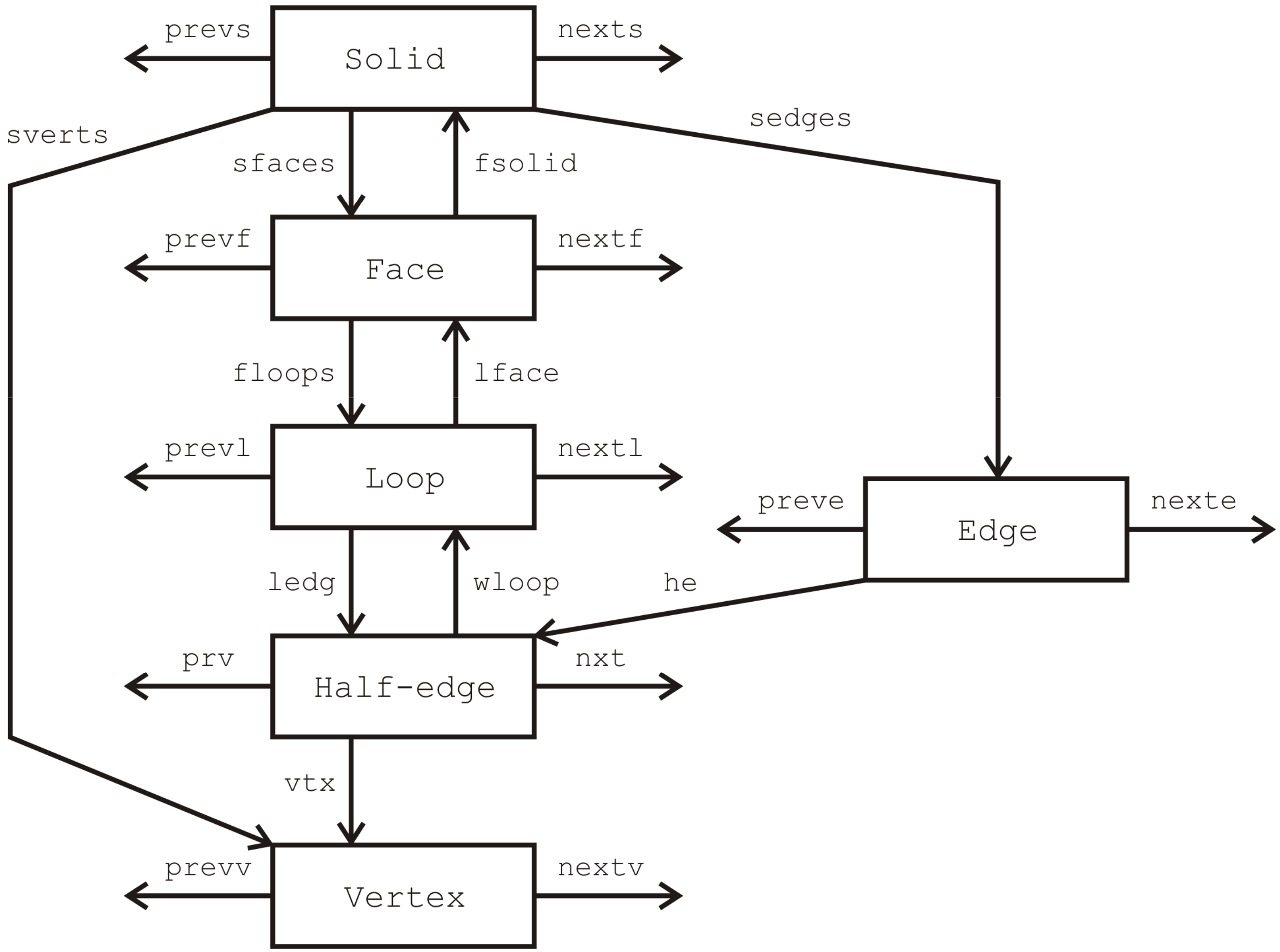
f	l_{out}	l_{in}

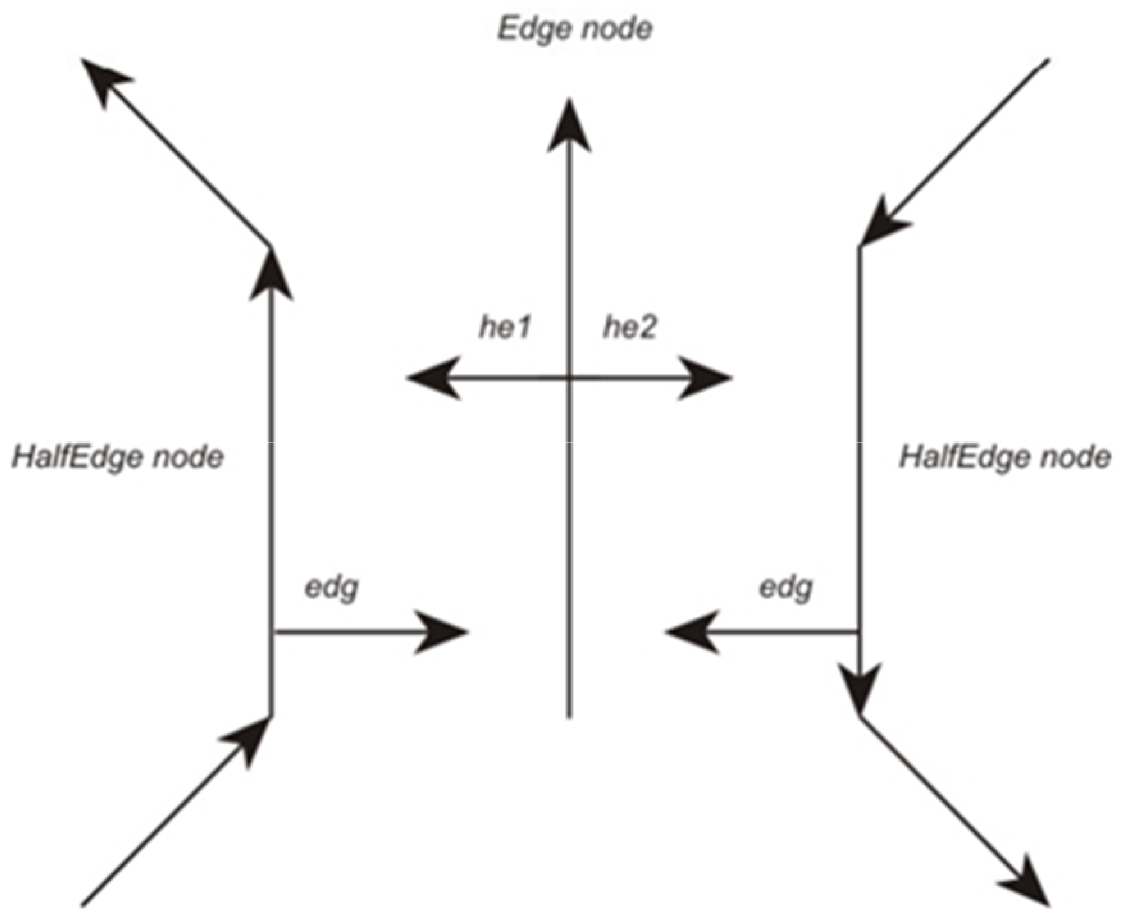
Hierarchy of Topological Levels

- Solid
- Face
- L*oop*
- *Half-Edge*
- Vertex
- Edge*



Half-Edge Data Structure Entities

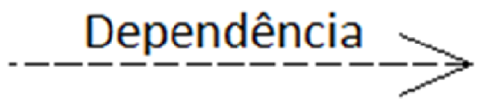



Edge node
he1
he2
HalfEdge node
HalfEdge node
edg
edg

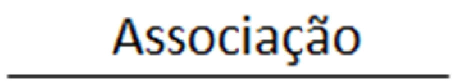
- Elements
  - Objects
  - Classes

- UML Diagrams
  - Use Cases
  - Robustez*
  - Sequence
  - Classes

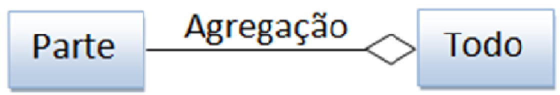
- Relationships
  - Dependency

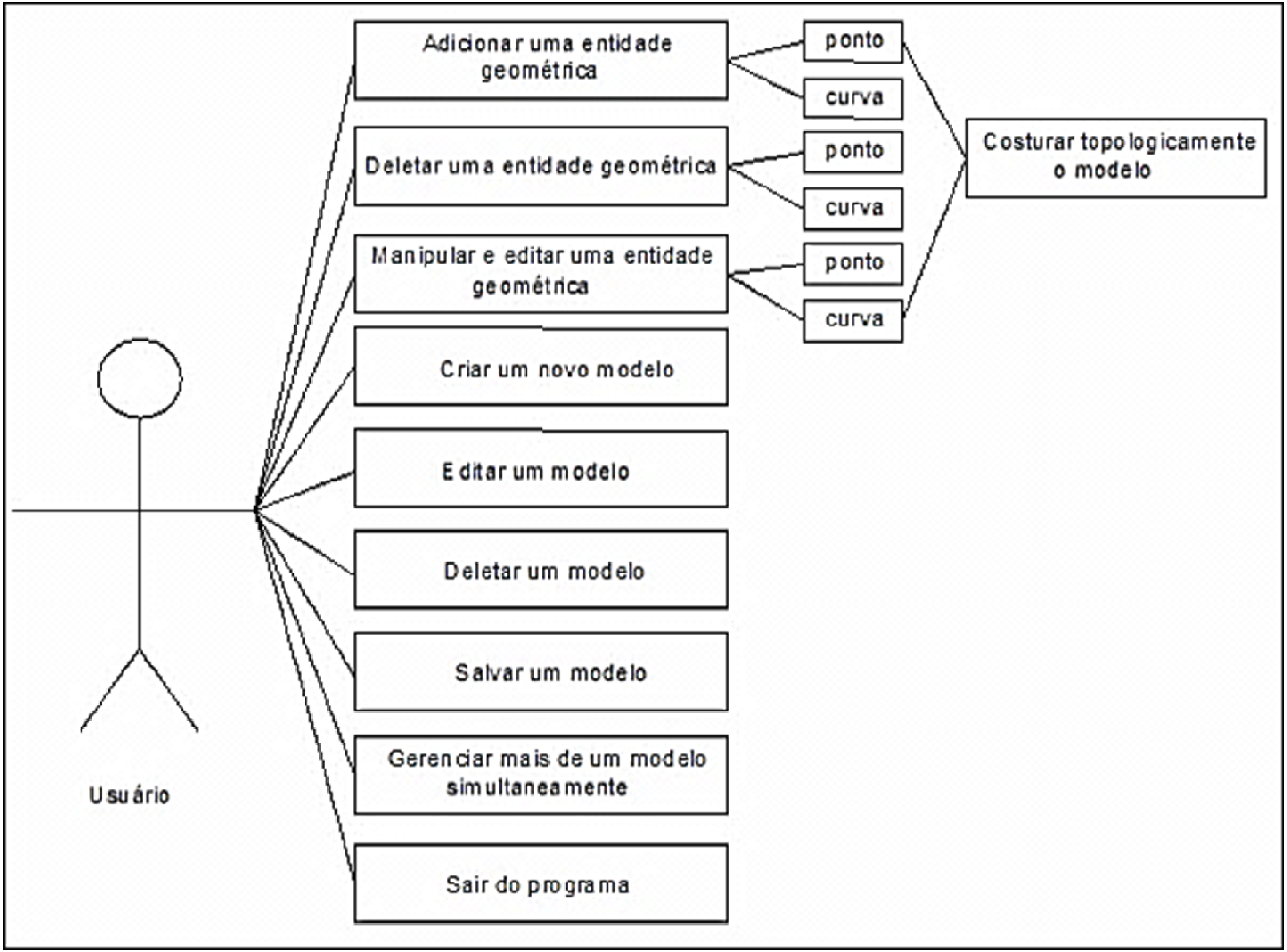


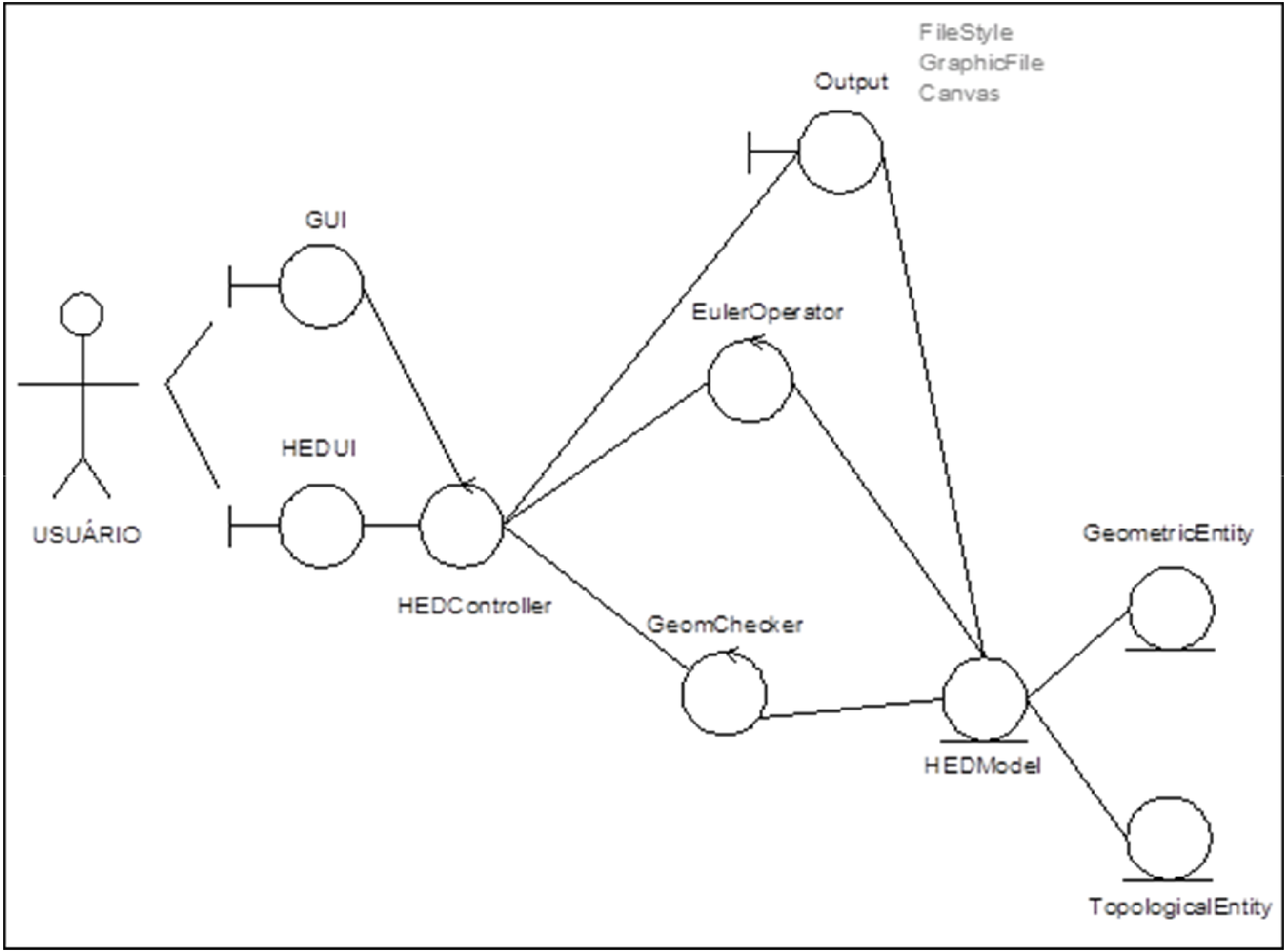
  - Association



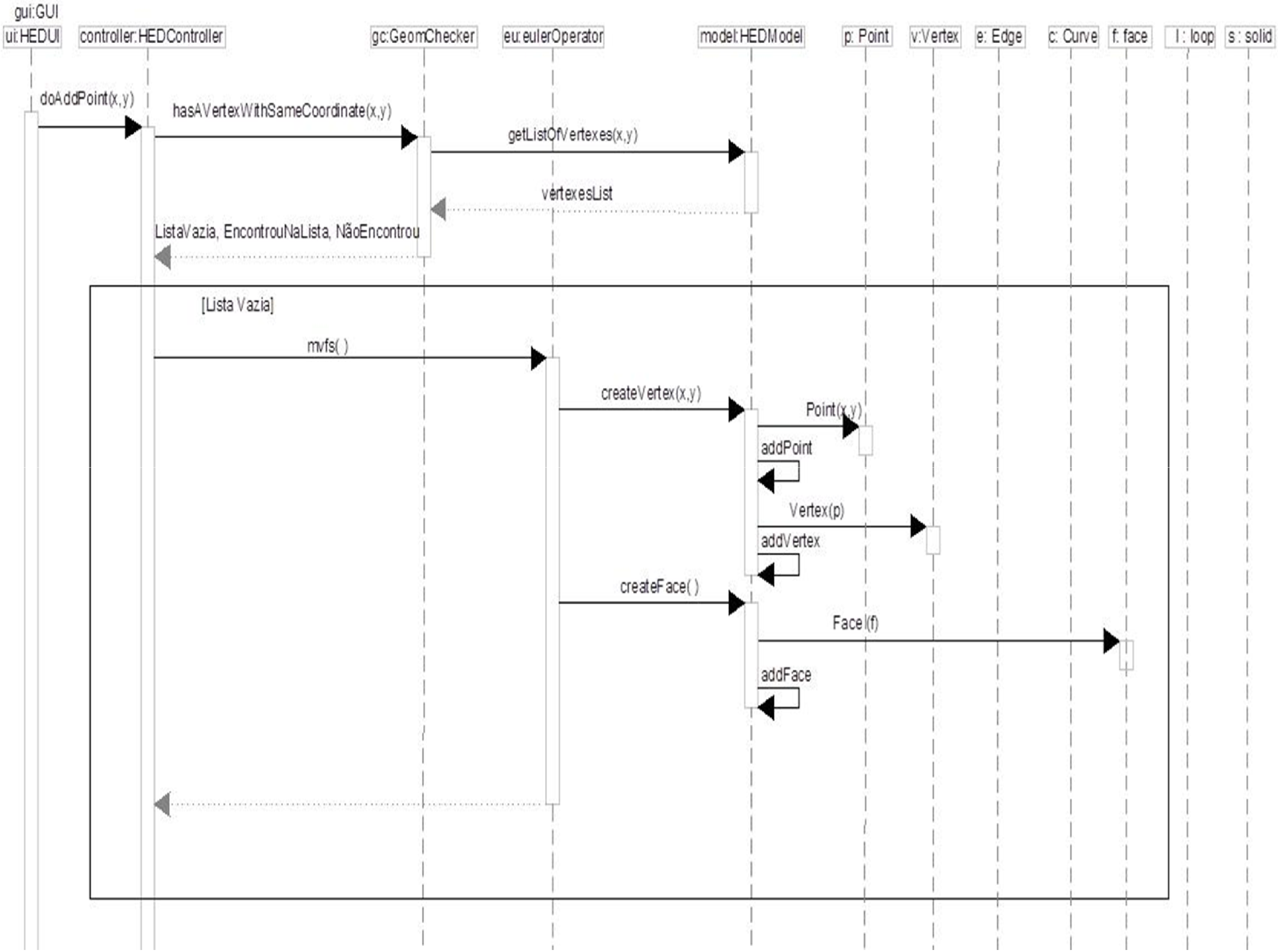
  - Aggregation

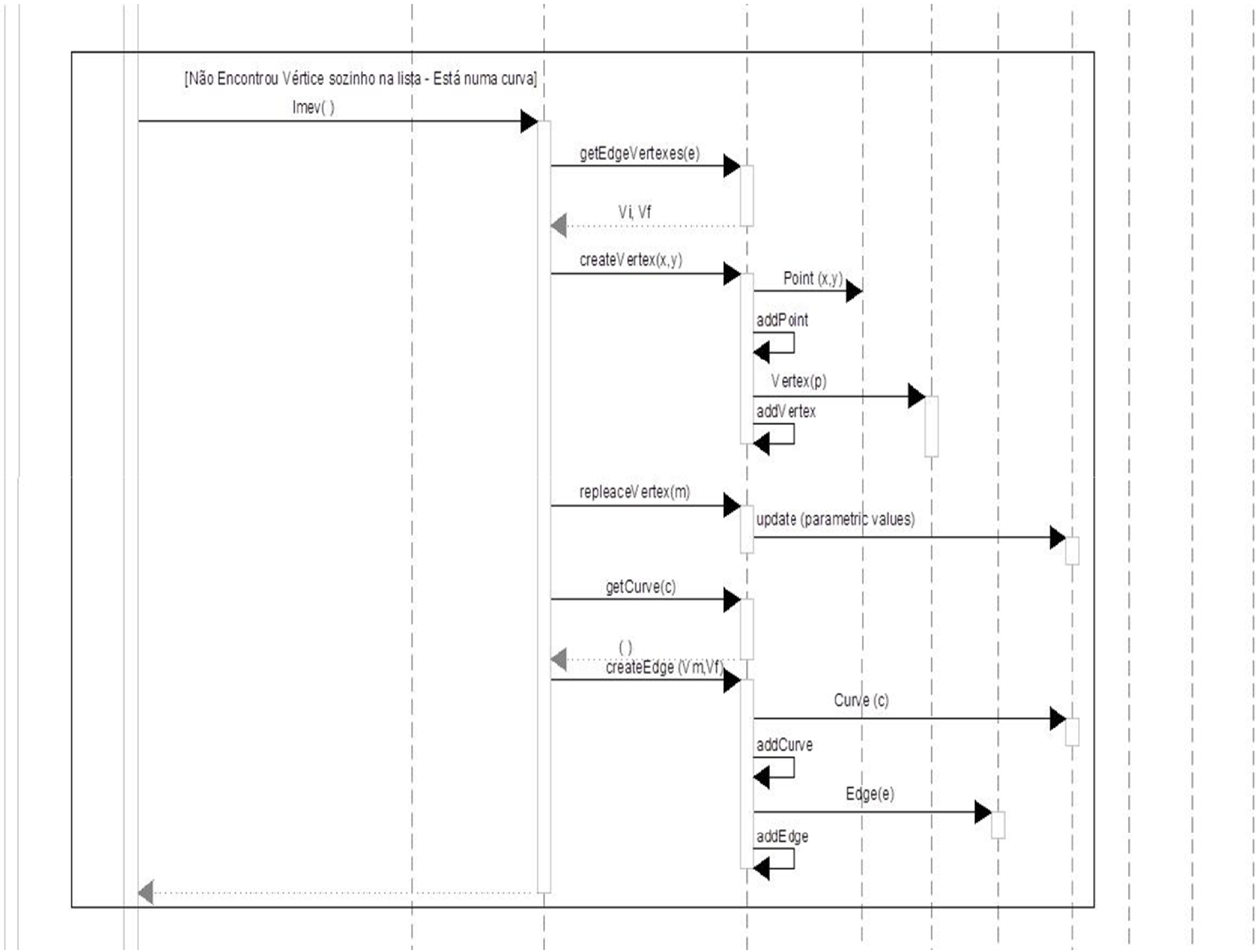



Adicionar uma entidade geométrica
ponto
curva
Deletar uma entidade geométrica
ponto
curva
Costurar topologicamente o modelo
Manipular e editar uma entidade geométrica
ponto
curva
Criar um novo modelo
Editar um modelo
Deletar um modelo
Salvar um modelo
Gerenciar mais de um modelo simultaneamente
Sair do programa
Usuário

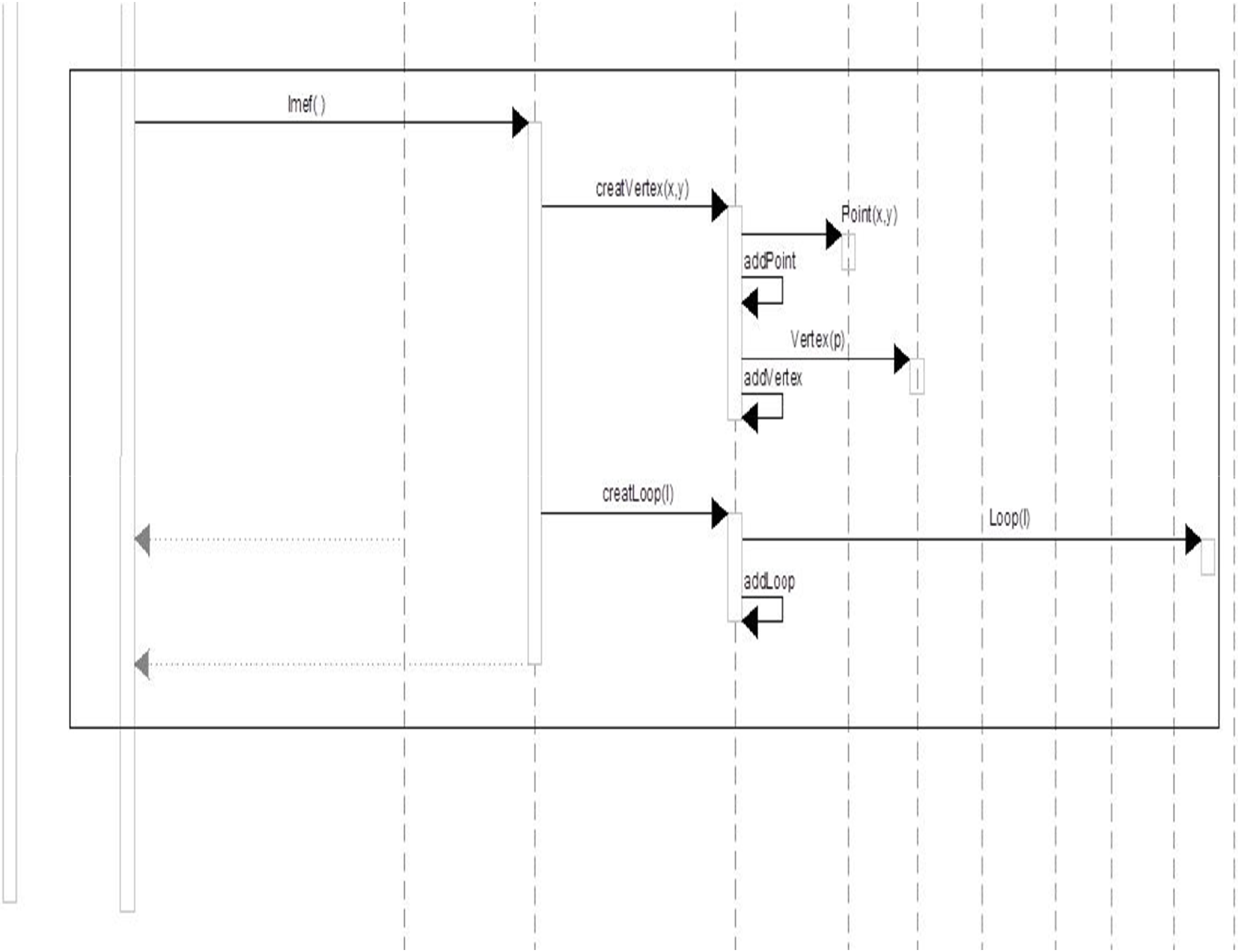


FileStyle
GraphicFile
Canvas
Output
GUI
EulerOperator
USUÁRIO
HEDUI
HEDController
GeomChecker
GeometricEntity
HEDModel
TopologicalEntity

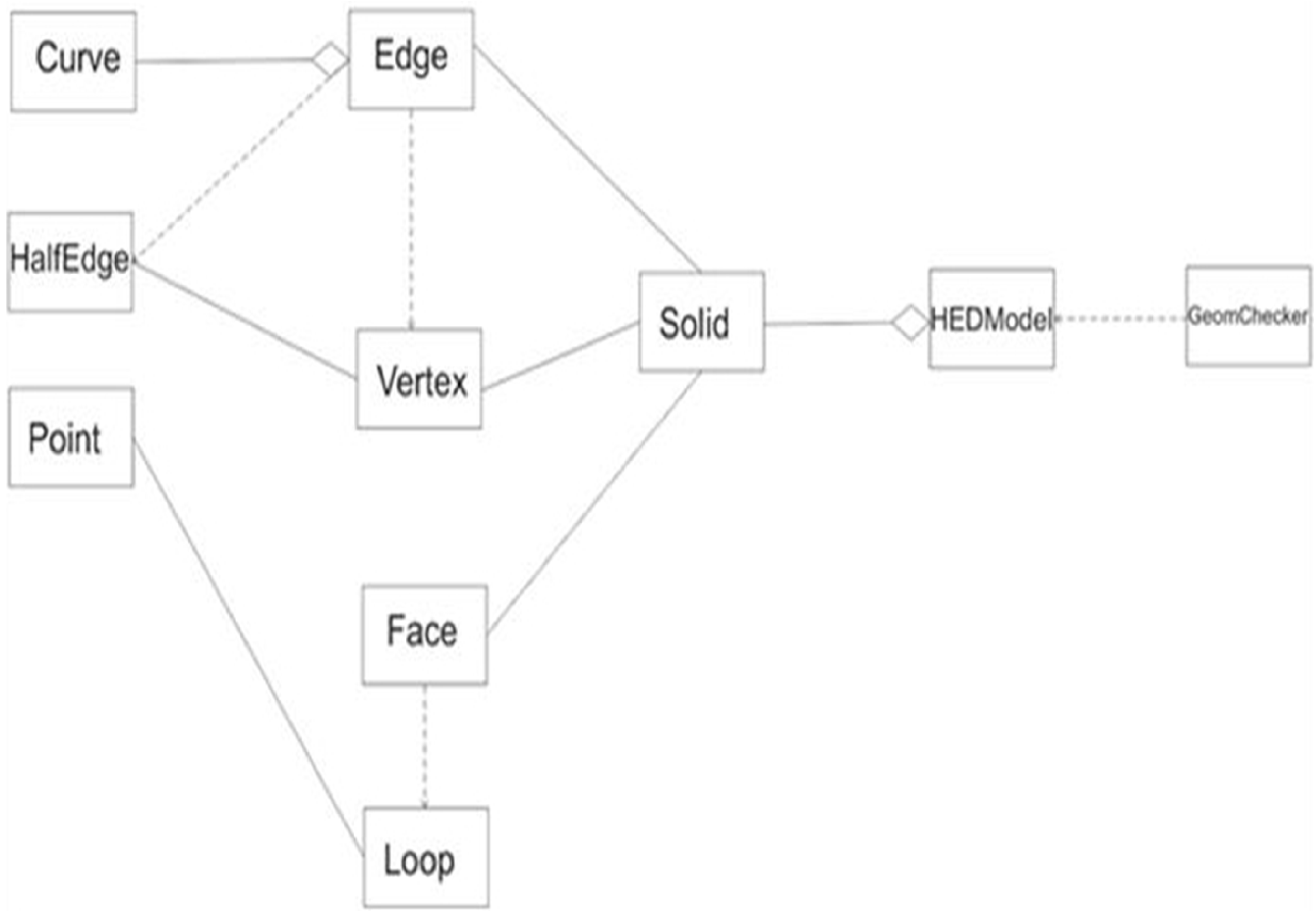

gui:GUI
ui:HEDUI
controller:HEDController
gc:GeomChecker
eu:eulerOperator
model:HEDModel
p: Point
v:Vertex
e: Edge
c: Curve
f: face
l : loop
s : solid
doAddPoint(x,y)
hasAVertexWithSameCoordinate(x,y)
getListOfVertexes(x,y)
vertexesList
ListaVazia, EncontrouNaLista, NãoEncontrou
[Lista Vazia]
mvfs( )
createVertex(x,y)
Point(x,y)
addPoint
Vertex(p)
addVertex
createFace( )
Face(f)
addFace

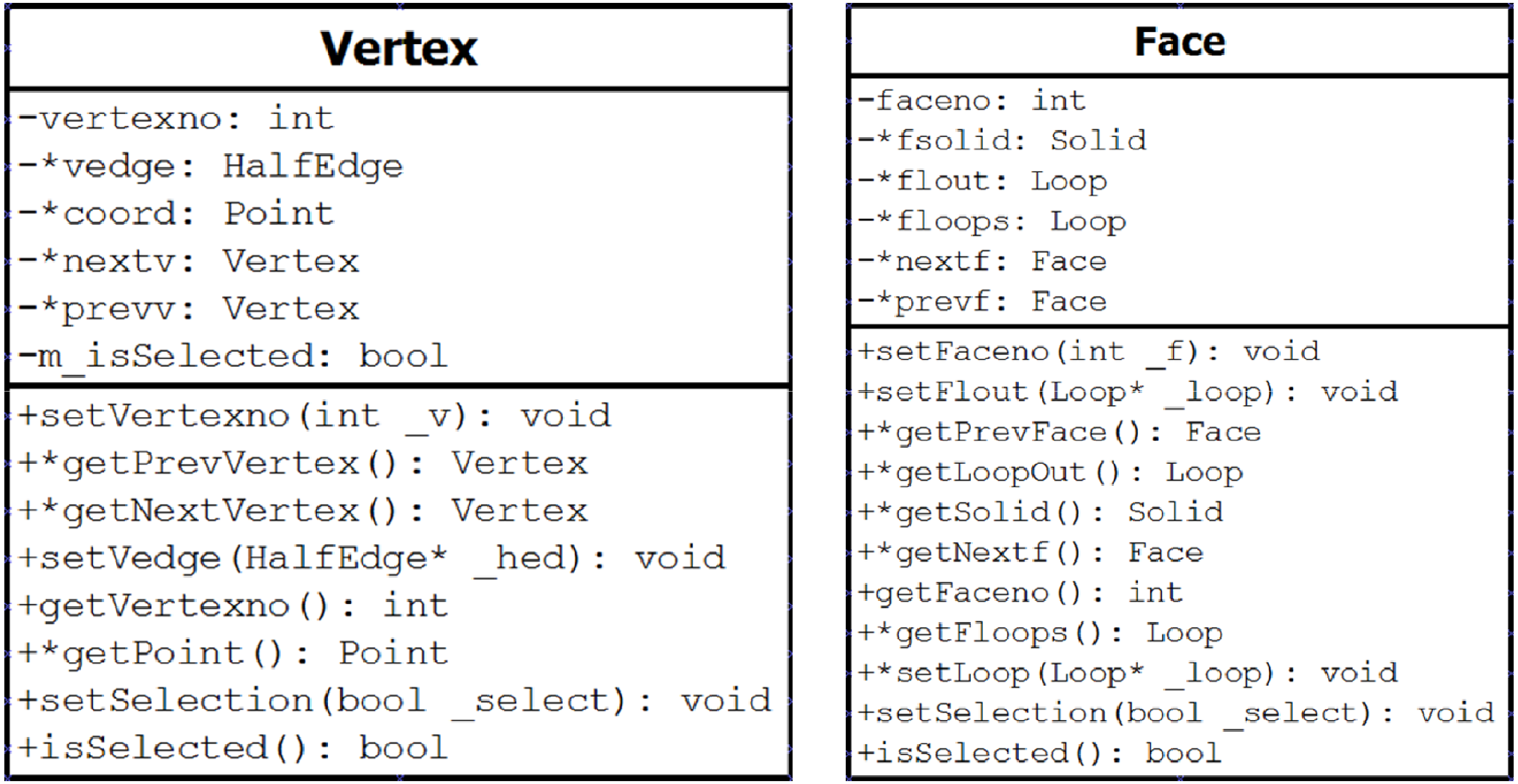


[Não Encontrou Vértice sozinho na lista - Está numa curva]
lmev( )
getEdgeVertexes(e)
Vi, Vf
createVertex(x,y)
Point (x,y)
addPoint
Vertex(p)
addVertex
repleaceVertex(m)
update (parametric values)
getCurve(c)
( )
createEdge (Vm,Vf)
Curve (c)
addCurve
Edge(e)
addEdge



lmef( )
creatVertex(x,y)
Point(x,y)
addPoint
Vertex(p)
addVertex
creatLoop(l)
Loop(l)
addLoop


Curve
Edge
HalfEdge
Solid
HEDModel
GeomChecker
Vertex
Point
Face
Loop


Vertex
-vertexno: int
-*vedge: HalfEdge
-*coord: Point
-*nextv: Vertex
-*prevv: Vertex
-m_isSelected: bool
+setVertexno(int _v): void
+*getPrevVertex(): Vertex
+*getNextVertex(): Vertex
+setVedge(HalfEdge* _hed): void
+getVertexno(): int
+*getPoint(): Point
+setSelection(bool _select): void
+isSelected(): bool
Face
-faceno: int
-*fsolid: Solid
-*flout: Loop
-*floops: Loop
-*nextf: Face
-*prevf: Face
+setFaceno(int _f): void
+setFlout(Loop* _loop): void
+*getPrevFace(): Face
+*getLoopOut(): Loop
+*getSolid(): Solid
+*getNextf(): Face
+getFaceno(): int
+*getFloops(): Loop
+*setLoop(Loop* _loop): void
+setSelection(bool _select): void
+isSelected(): bool

Edge

```
-*he1: HalfEdge
-*he2: HalfEdge
-*nexte: Edge
-*preve: Edge
-*curve: Curve
```

```
+*getPrevEdge(): Edge
+*getNextEdge(): Edge
+*getHe1(): HalfEdge
+getHe2(): HalfEdge
+*getFirstVertex(): Vertex
+*getSecondVertex(): Vertex
+*addhe(Vertex* v, HalfEdge* where, EOrientation sign)
+setSelection(bool _select): void
+isSelected(): bool
```

HEDUI

```
-controle: HEDController
```

```
+addPoint(double x, double y, double z): void
+addCurve(double x1, double y1, double z1,
           double x2, double y2, double z2): void
+addTriangle(double x1, double y1, double z1,
              double x2, double y2, double z2,
              double x3, double y3, double z3): void
```

EulerOperator

```
-lmev (HalfEdge* he1, HalfEdge* he2, int v, double x, double y, double z): void
-*lmef (HalfEdge* he1, HalfEdge* he2, int f): Face
-lkmer (HalfEdge* he1, HalfEdge* he2): void
```

```
+mvfs(HEDModel& _model, int s, int f, int v,
       double x, double y, double z): void
+mev(HEDModel& _model, int s, int f1, int f2,
      int v1, int v2, int v3, int v4, double x,
      double y, double z): void
+mef(HEDModel& _model, int s, int f1, int v1,
      int v2, int v3, int v4, int f2): void
```

GeomChecker

```
-computerLineLineIntersection(double x1, double y1, double x2, double y2,
                              double x3, double y3, double x4, double y4): int
```

```
+hasAnyVertex(HEDModel& _model): bool
+hasVertexWithGivenCoords(HEDModel& _model,
                          int& id, double x,
                          double, y, double z,
                          double _tol): bool
+liesOnCurve(HEDModel& _model, double x,
             double y, double z): bool
+intersectsVertex(HEDModel& _model, double* curvePts,
                  int&n, double* intersPts): bool
+intersectsCurves(HEDModel& _model, double* curvePts,
                  int&n, double* intersPts): bool
```

HEDController

```
-*eulerop: EulerOperator
-model: HEDModel
-geomChecker: GeomChecker
```

```
+doAddPoint(double x, double y, double z,
            double tol): void
+doAddCurve(double x1, double y1, double z1,
           double x2, double y2, double z2,
            double tol): void
+doAddTriangle(double x1, double y1, double z1,
              double x2, double y2, double z2,
               double x3, double y3, double z3): void
+doSelection(double x, double y, double z,
             double tol): void
+unselect(): void
+doManipulate(): void
+doCreate(): void
+doEdit(): void
+doDelete(): void
+doSave(): void
+doManage(): void
+doExit(): void
+getModel(): HEDModel
```

Solid

```
-solidno: int
-*sfaces: Face
-*sedges: Edge
-*sverts: Vertex
-*nexts: Solid
-*prevs: Solid
```

```
+setSolidno(int _s): void
+*getFace(): Face
+*getEdge(): Edge
+*getVertex(): Vertex
+*getNexts(): Solid
+*getSfaces(): Face
+getSolidno(): int
+setFace(Face* _face): void
+setVertex(Vertex* _vertex): void
+setEdge(Edge* edge): void
```

HEDModel

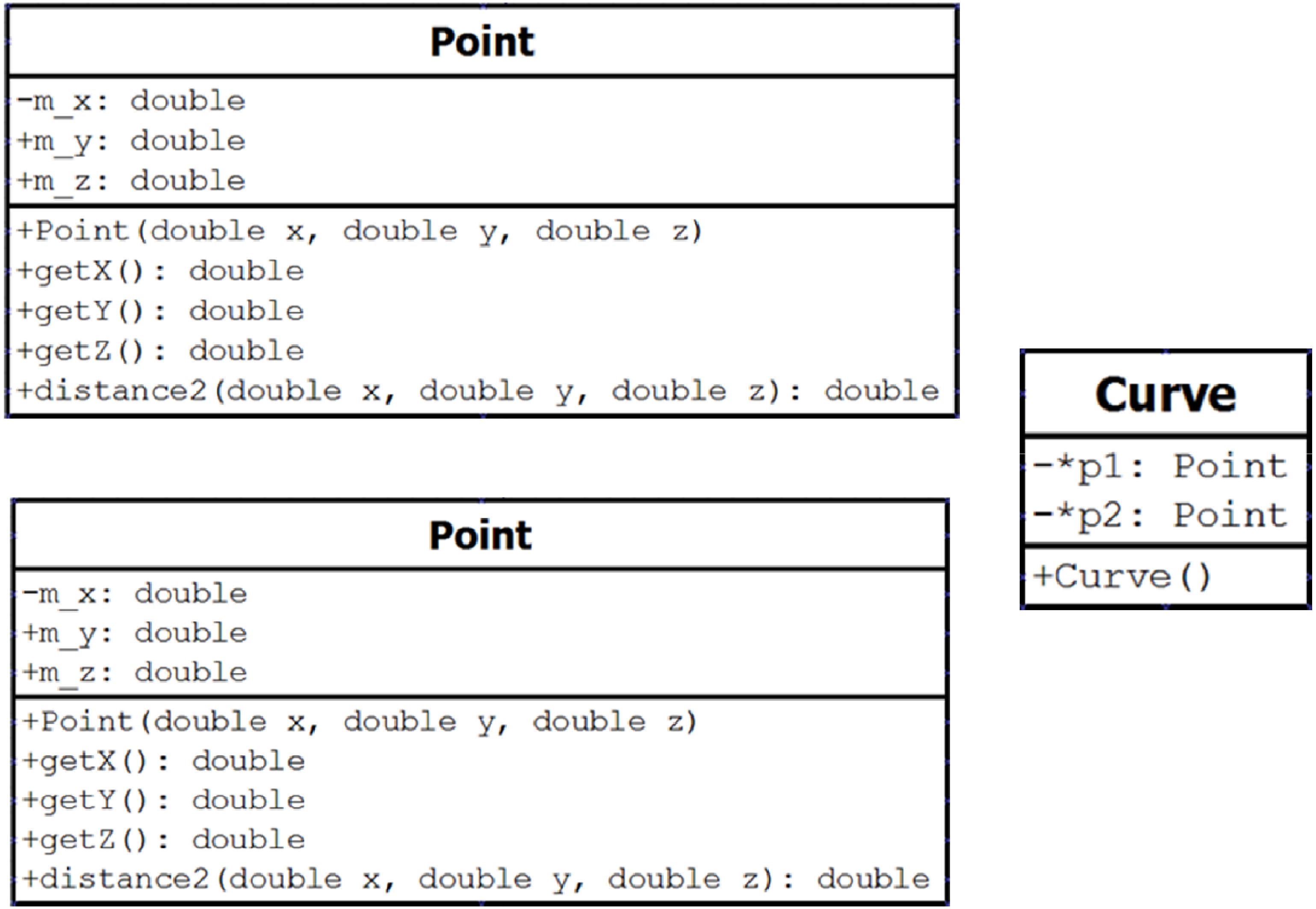
```
-*firsts: Solid
```

```
+*getFirsts(): Solid
+setFirsts(Solid* _firsts): void
+*getsolid(int sn): Solid
+*fface(Solid* s, int fn): Face
+*fhe(Face* f, int vn1, int vn2): HalfEdge
+*getVertex(double _x, double _y, double _z,
            double _tol): Vertex
+*getEdge(double _x, double _y, double _z,
          double _tol): Edge
+*getFace(double _x, double _y, double _z,
          double _tol): Face
+getNumberOfVertexes(): int
+getNumberOfFaces(): int
+getEdgeVertexes(): int
+replaceVertex(): void
+getCurve(): void
```

Loop

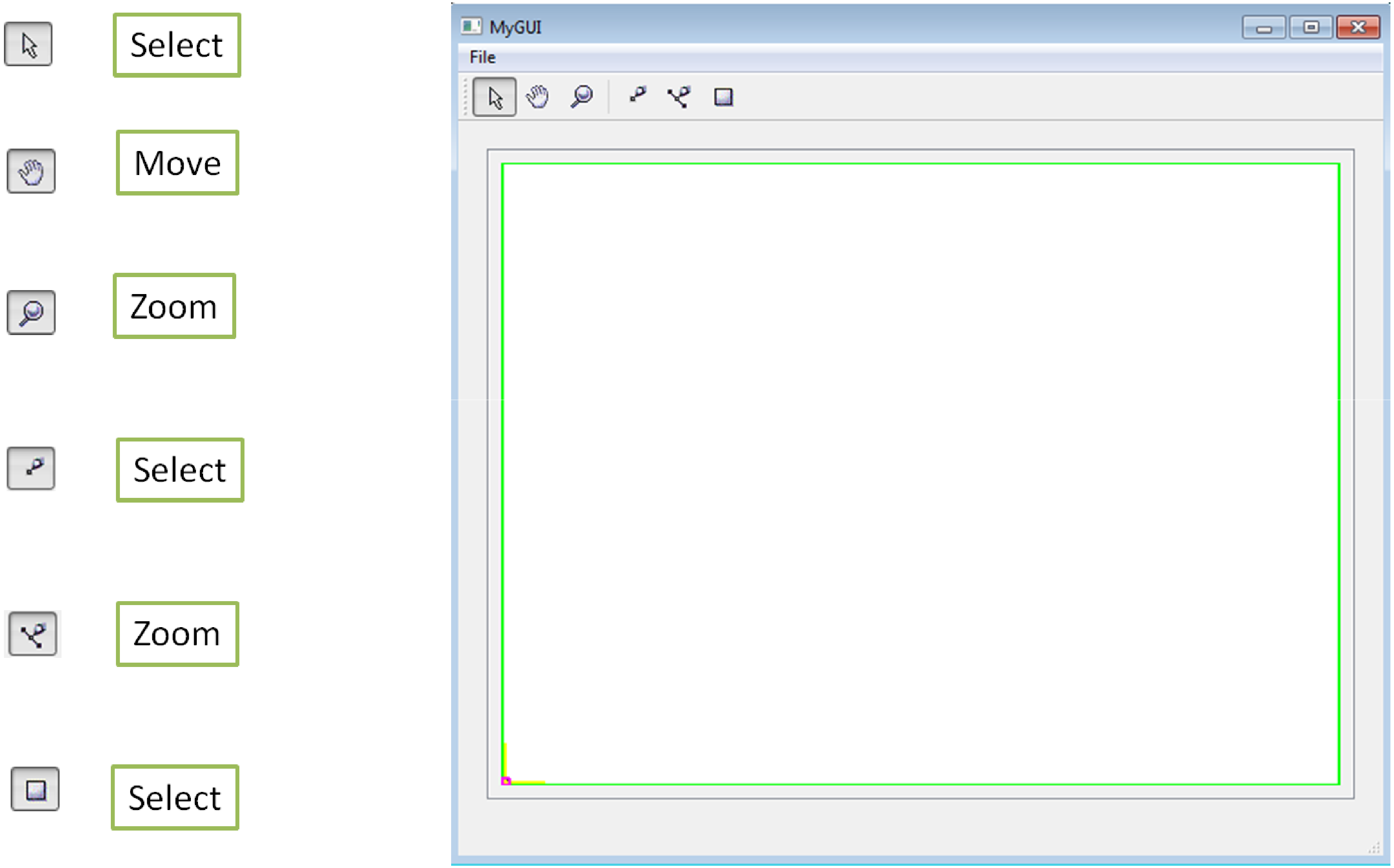
```
-*ledg: HalfEdge
-*lface: Face
-*nextl: Loop
-*prevl: Loop
```

```
+setLedg(HalfEdge* _newhe): void
+*getPrevLoop(): Loop
+*getFace(): Face
+*getNextl(): Loop
+*getLedg(): HalfEdge
```


Point
-m_x: double
+m_y: double
+m_z: double
+Point(double x, double y, double z)
+getX(): double
+getY(): double
+getZ(): double
+distance2(double x, double y, double z): double
Curve
-*p1: Point
-*p2: Point
+Curve()
Point
-m_x: double
+m_y: double
+m_z: double
+Point(double x, double y, double z)
+getX(): double
+getY(): double
+getZ(): double
+distance2(double x, double y, double z): double

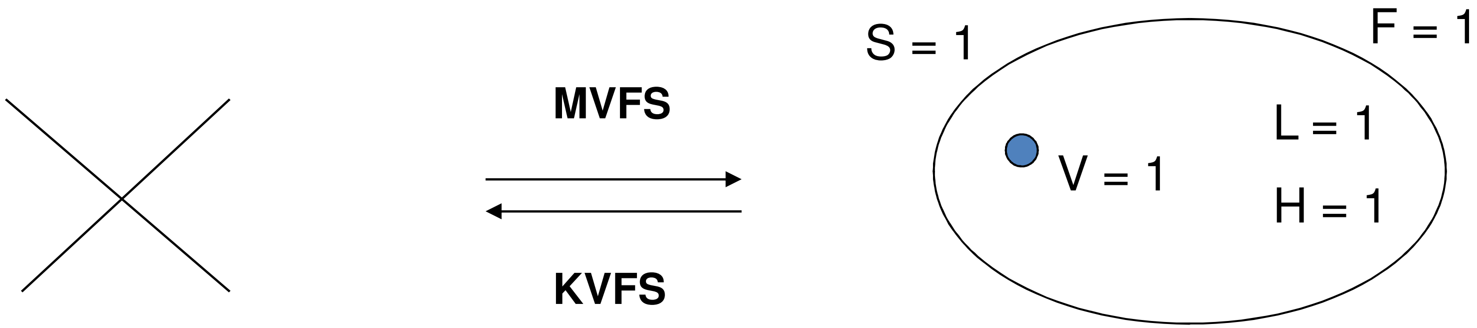
- **<u>Euler Operator - LMEV</u>:**

```
void EulerOperator::lmev(HalfEdge* he1, HalfEdge* he2, int v, double x, y, z)
{
        Loop* loop = he1->getLoop();
        Face* face = loop->getFace();
        Solid* solid = face->getSolid();
        Edge* newedge = new Edge(solid);
        Point* point = new Point(x,y,z);
        Vertex* newvertex = new Vertex(point,solid);
        newvertex->setVertexno(v);
        HalfEdge* he = he1;
        while(he != he2)
        {
                he->setVtx(newvertex);
                he = he->mate()->getNxt();
        }
        newedge->addhe(he2->getVtx(), he1, MINUS);
        newedge->addhe(newvertex, he2, PLUS);
        newvertex->setVedge(he2->getPrv());
        he2->getVtx()->setVedge(he2);
}
```


Select
Move
Zoom
Select
Zoom
Select
MyGUI
File

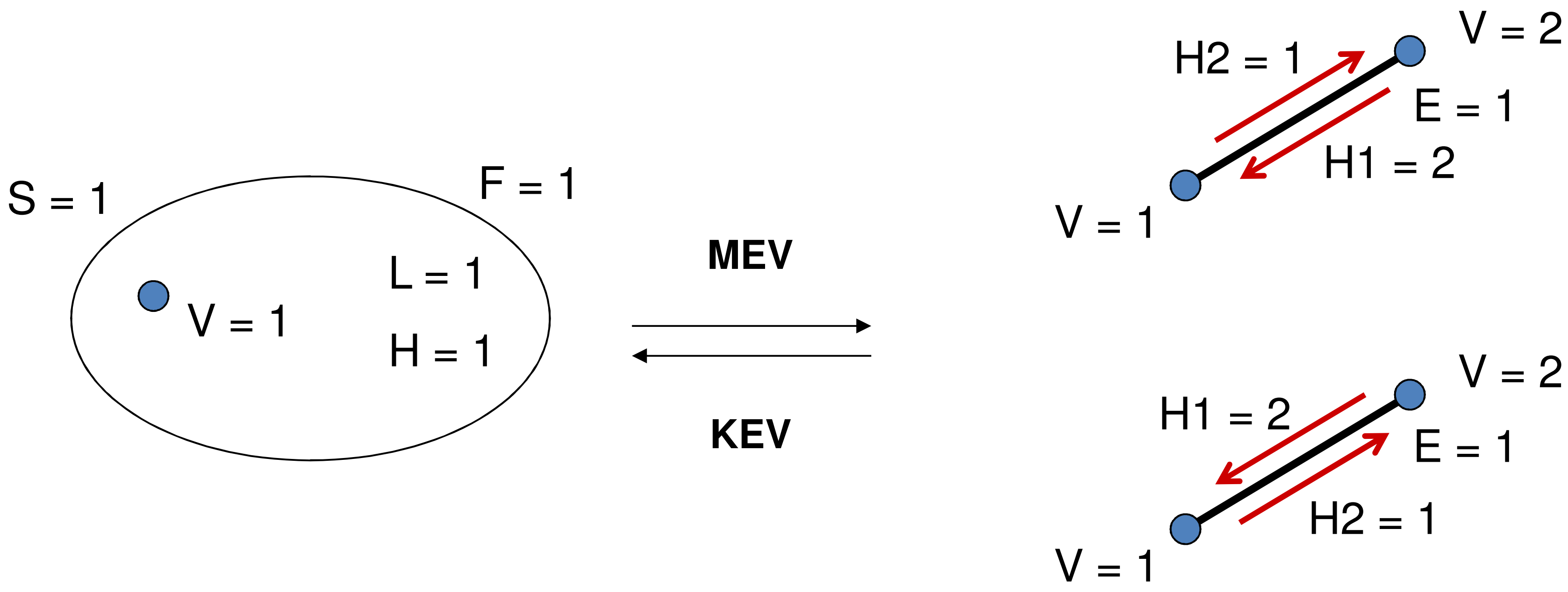
MVFS

V = 1 (H = 1) N = 0 P = 0
H = 1 (V = 1, E = 0, L = 1) N = 0 P = 0
E = 0 (H1 = 0, H2 = 0) N = 0 P = 0
L = 1 (H = 1, F = 1) N = 0 P = 0
F = 1 (S = 1, LOUT = 0 / LOOPS = 1) N = 0 P = 0
S = 1 (V = 1, F = 1, E = 0) N = 0 P = 0



MEV

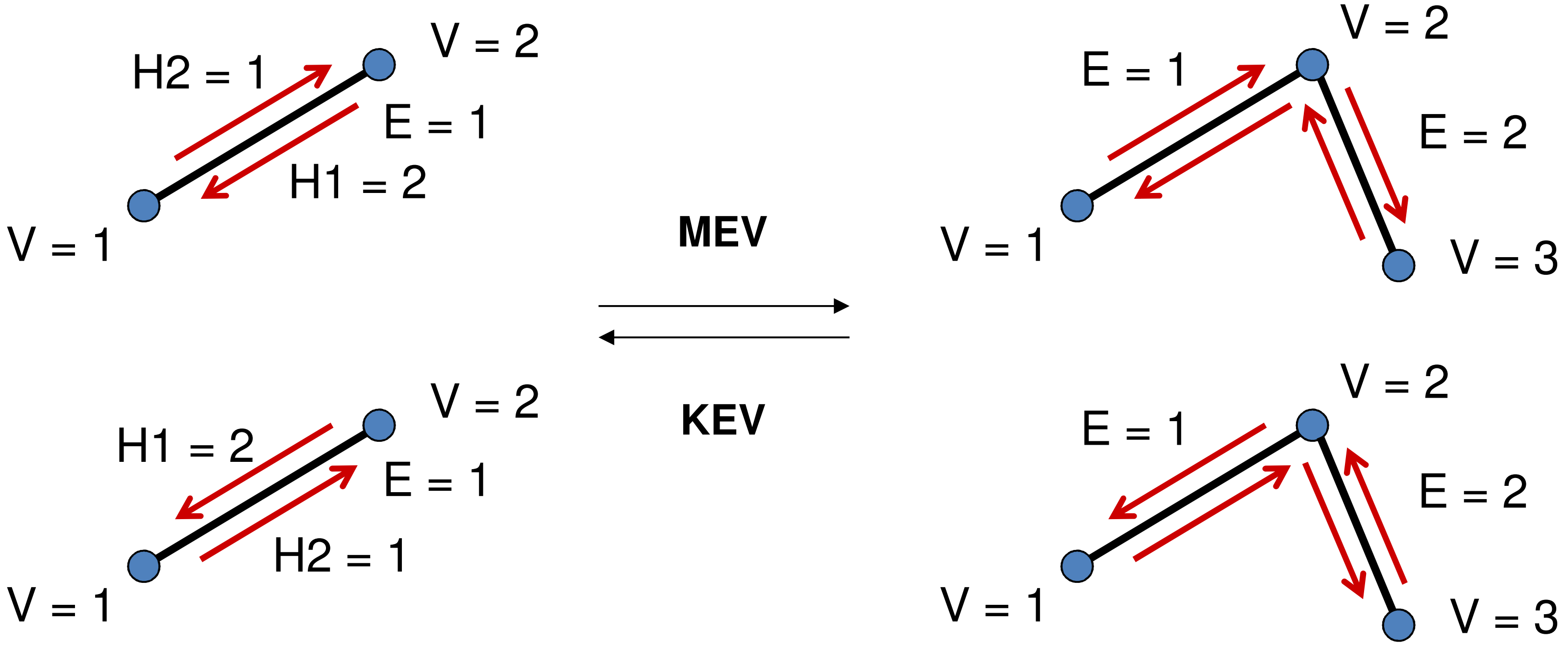
V = 1 (H = 1) N = 0 P = 2
V = 2 (H = 2) N = 1 P = 0
H = 1 (V = 1, E = 1, L = 1) N = 2 P = 2
H = 2 (V = 2, E = 1, L = 1) N = 1 P = 1
E = 1 (H1 = 2, H2 = 1) N = 0 P = 0
L = 1 (H = 2, F = 1) N = 0 P = 0
F = 1 (S = 1, LOUT = 0 / LOOPS = 1) N = 0 P = 0
S = 1 (V = 1, F = 1, E = 1) N = 0 P = 0



MEV

For a single strip there is no definition of the sequence (ccw nor ucw)

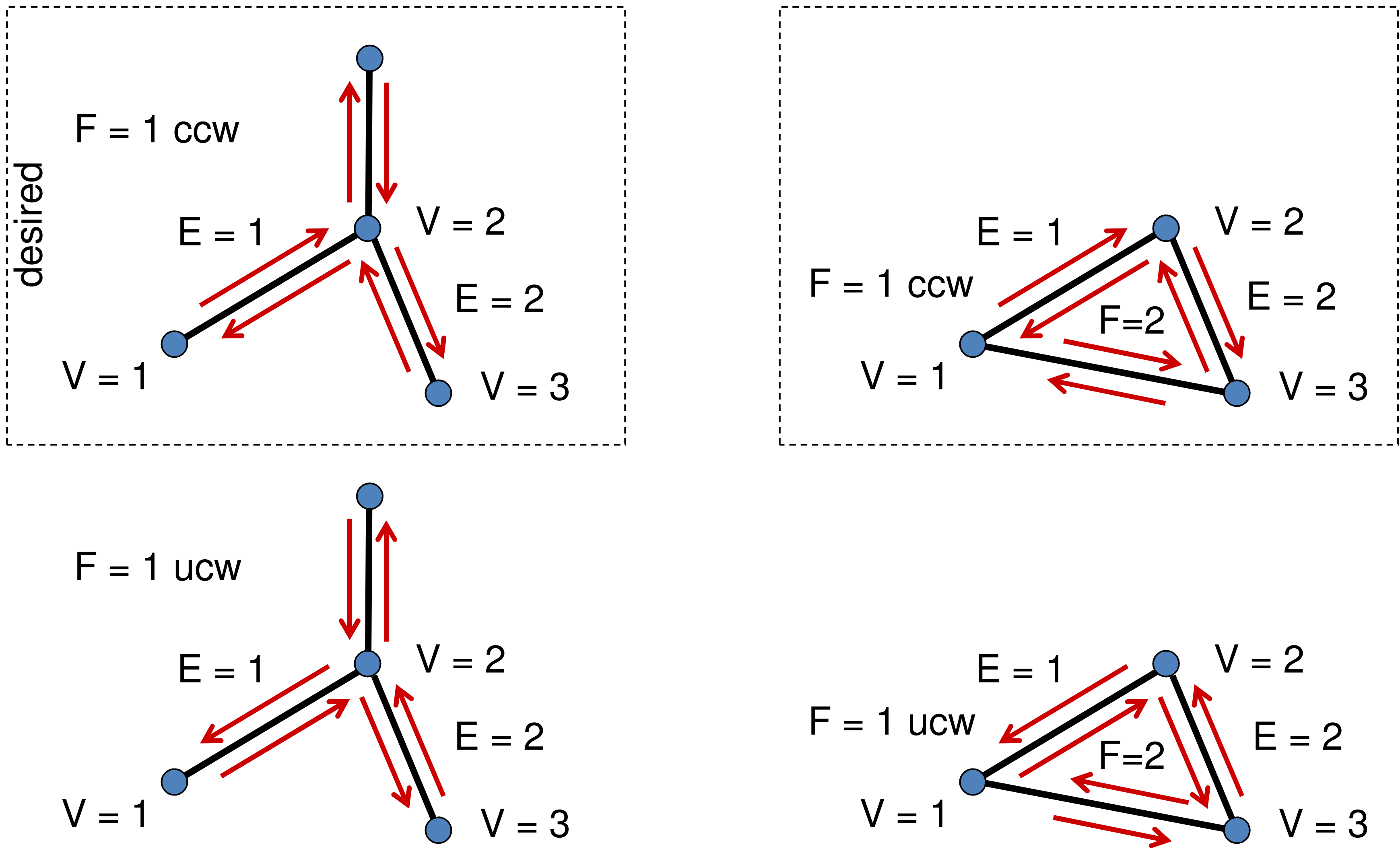
V = 1 (H = 1) N = 0 P = 2
V = 2 (H = 3) N = 1 P = 3
V = 3 (H = 4) N = 2 P = 0
H = 1 (V = 1, E = 1, L = 1) N = 3 P = 2
H = 2 (V = 2, E = 1, L = 1) N = 1 P = 4
H = 3 (V = 2, E = 2, L = 1) N = 4 P = 1
H = 4 (V = 3, E = 2, L = 1) N = 2 P = 3
E = 1 (H1 = 2, H2 = 1) N = 0 P = 2
E = 2 (H1 = 3, H2 = 4) N = 1 P = 0
L = 1 (H = 2, F = 1) N = 0 P = 0
F = 1 (S = 1, LOUT = 0 / LOOPS = 1) N = 0 P = 0
S = 1 (V = 1, F = 1, E = 1) N = 0 P = 0



Defines the sequence if occurs two situations:

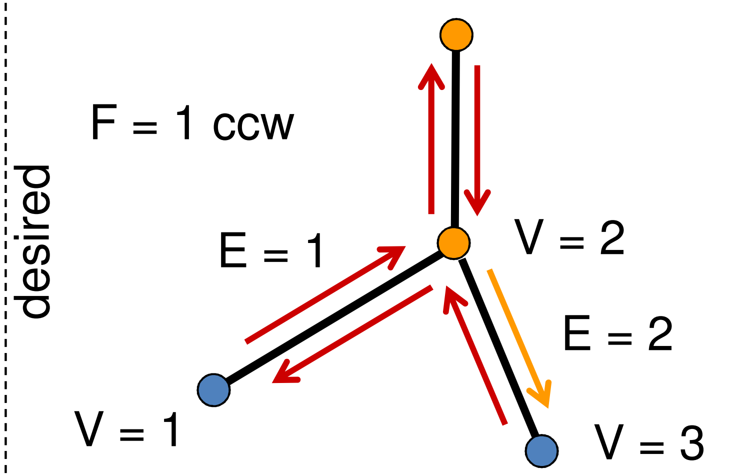
a MEV between two edges

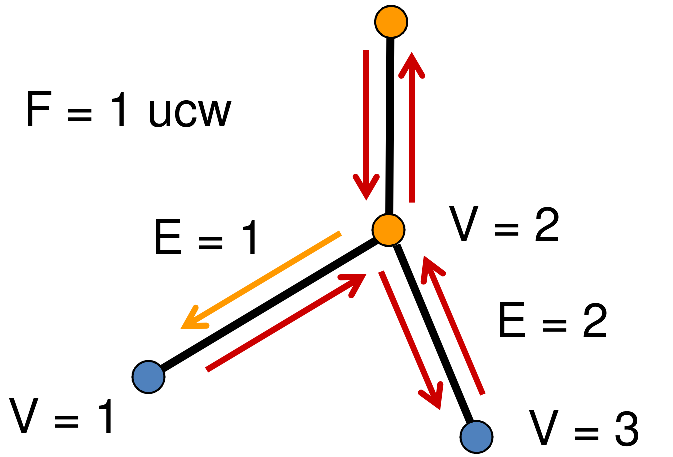
the first MEF



Which are the parameter to define each situation?

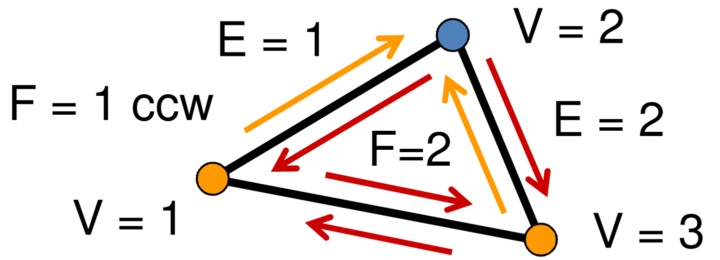
MEV

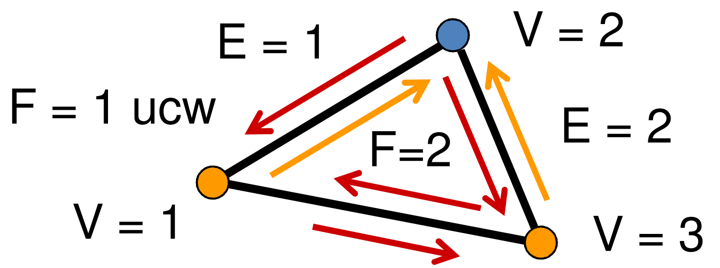




MEF

In this case, the half-edge of edge 2 (if it is the first parameter of MEF) receives the new face/loop. It is decided if the new loop area is positive! TIP: Always keep the first face with negative area (as the outside face).





Non-manifold Geometric Modeling

Geometric Modeling

- **Topology in *non-manifold* representations**

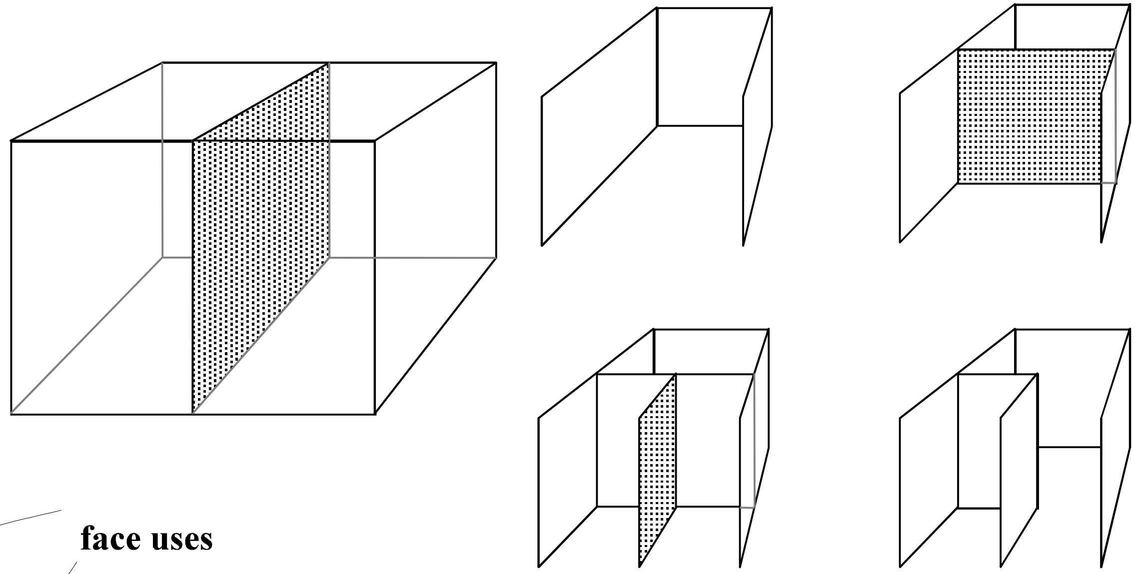
  - Application areas of geometric modeling that take advantage of the additional features of non-manifold representation

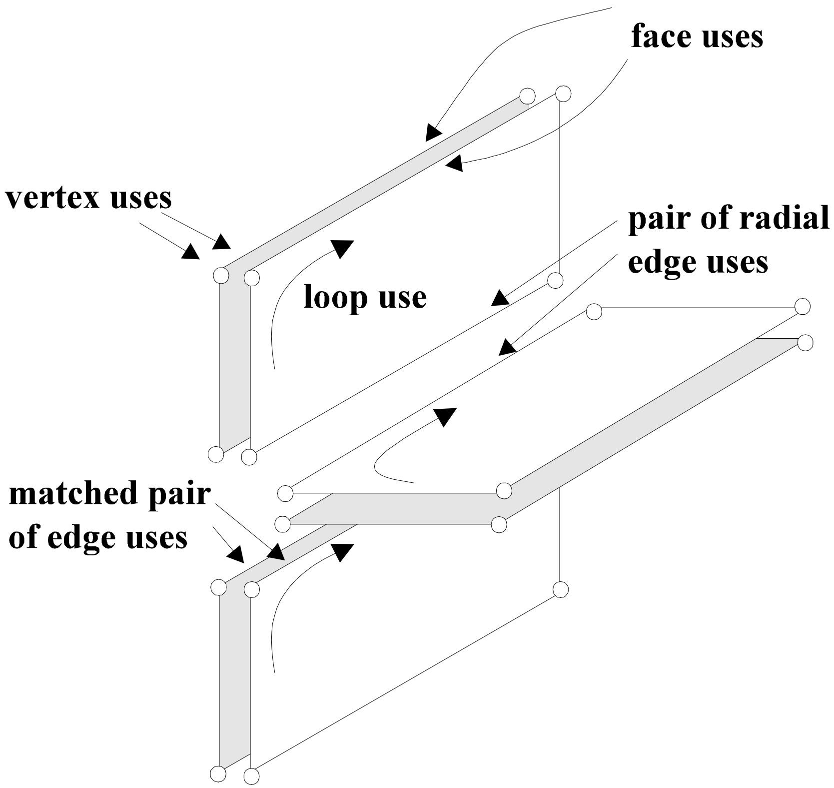
    - **Modeling** – transition between models, regions detections, storing arbitrary geometric information

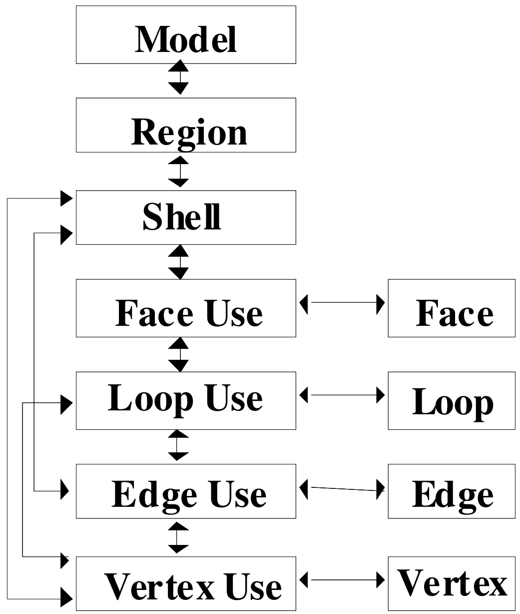
    - **Analysis** – implementation of building tools and simultaneous analysis of the model

    - **Representation of heterogeneous objects** – regions with common volumes, coincident faces, internal structures, solids consisting of different materials

Radial-Edge
(Weiler 1986)







Parametric Modeling

Parametric Modeling

MCAD (*Mechanical Computer Aided Design*)

Relatively new Technology.

Parametric Modeling

MCAD (*Mechanical Computer Aided Design*)

Relatively new Technology.

Its development has been occurring since 40 years in parallel to the development of hardware technology.

It was first introduced in the late 1980s, and recently became the new paradigm of mechanical CAD designs.

Parametric Modeling

MCAD (*Mechanical Computer Aided Design*)

Relatively new Technology.

Its development has been occurring since 40 years
in parallel to the development of hardware technology.

It was first introduced in the late 1980s, and
recently became the new paradigm of mechanical CAD designs.

It has increased the CAD technology
at the level of being very powerful design tools.

It automates the design and reviewing procedures
by using *parametric features*.

Parametric Modeling

The word **parametric** means that the settings of the project geometry, such as dimensions, can be changed at any time in the design process.

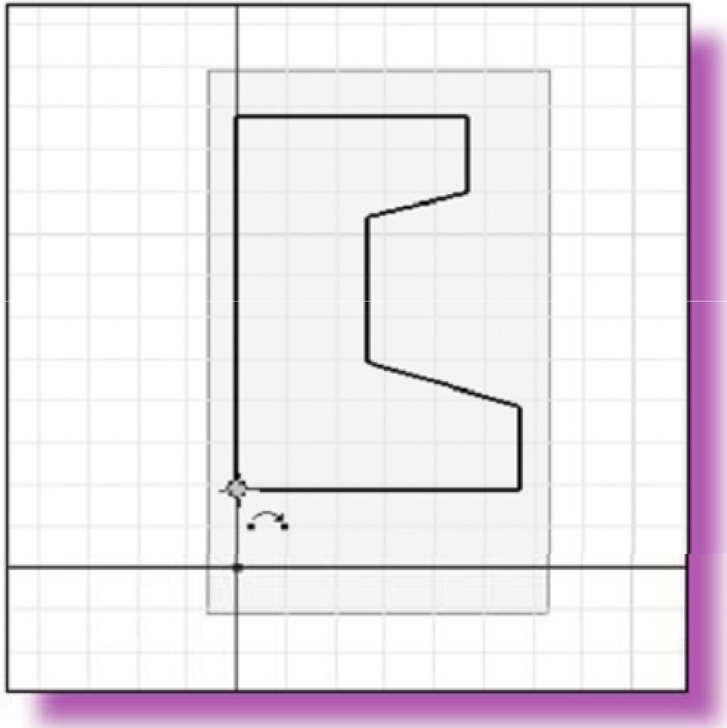
Parametric modeling is so named because of the design of parameters or variables that are modified during the project simulation process.

Vocabulary and Formalization:

- *Features*
- *Part*
- *Constrains*
- *Assembly*
- *Sketch*

Parametric Modeling

Sketcher

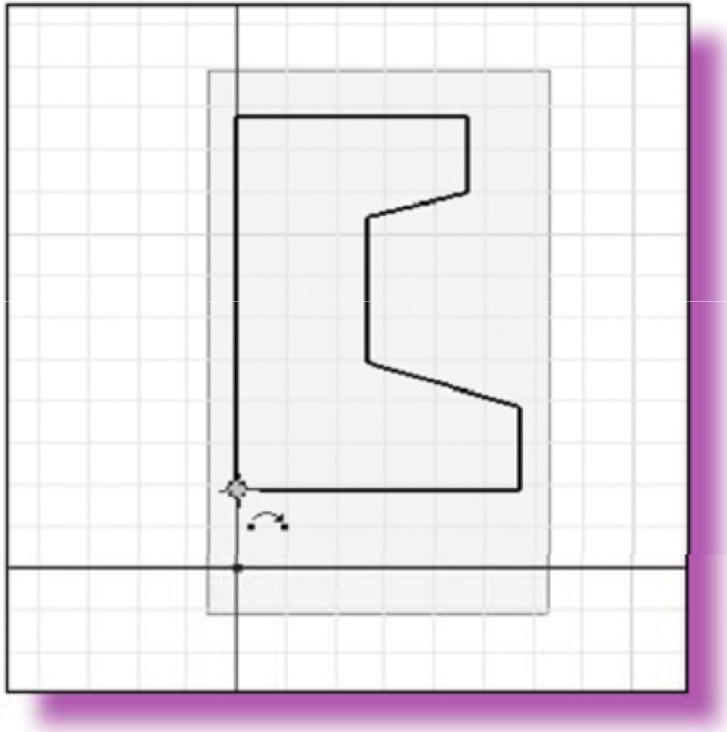


Parametric Modeling

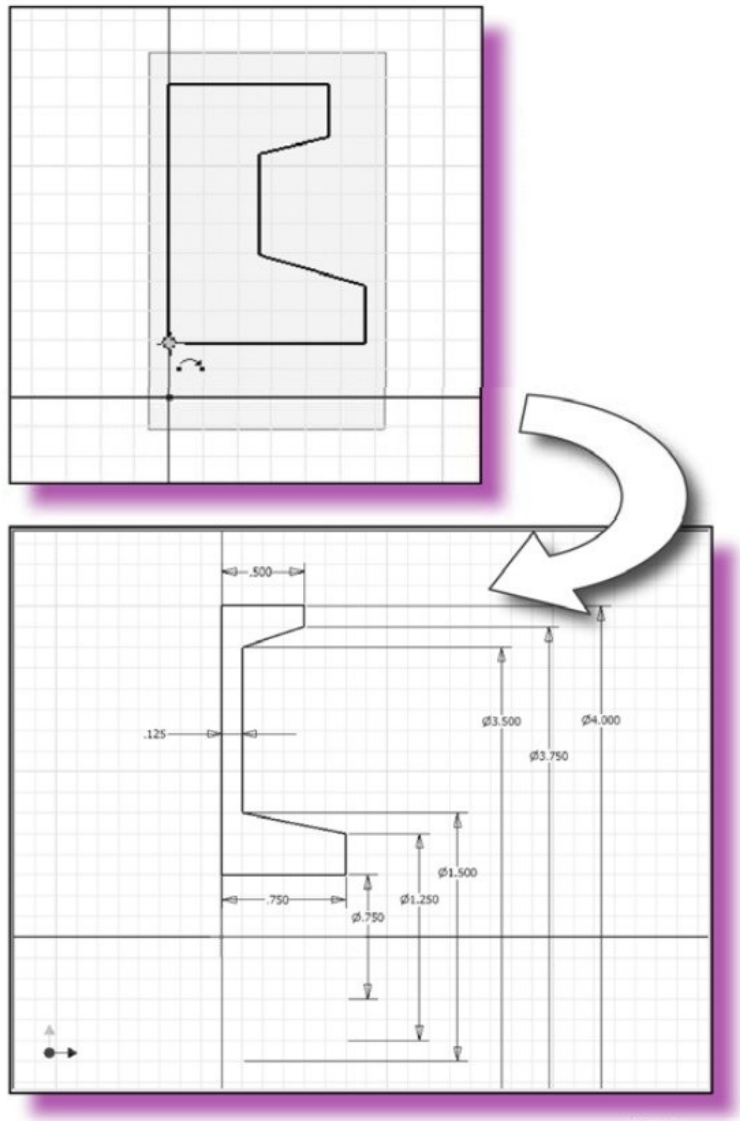
Geometric Tools:

- **Point**: Draws a point
- **Arc**: Draws an arc segment from center, radius, start angle and end angle
- **Circle**: Draws a circle from center and radius
- **2-point Line**: Draws a line segment from 2 points
- **Polyline (multiple-point line)**: Draws a line made of multiple line segments
- **Rectangle**: Draws a rectangle from 2 opposite points
- **Fillet**: Makes a fillet between two lines joined at one point. Select both lines or click on the corner point, then activate the tool.
- **Trimming**: Trims a line, circle or arc with respect to the clicked point.
- **External Geometry**: Creates an edge linked to external geometry.
- **Construction Mode**: Toggles an element to/from construction mode. A construction object will not be used in a 3D geometry operation.

Sketcher



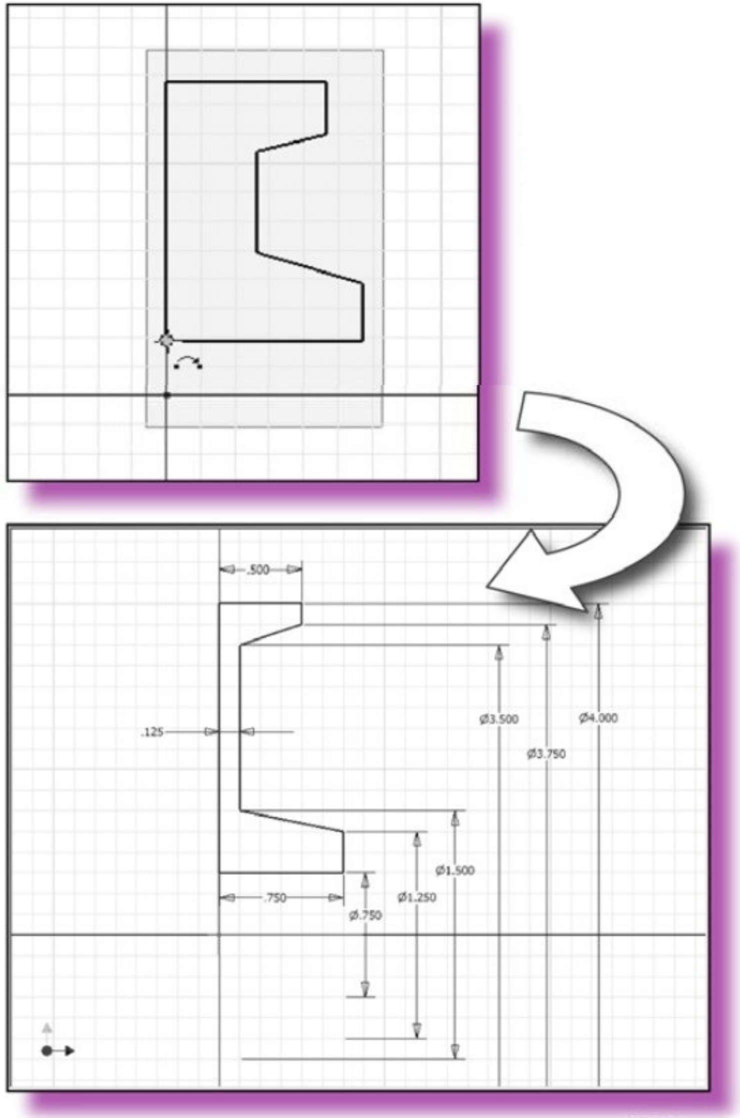
Parametric Modeling



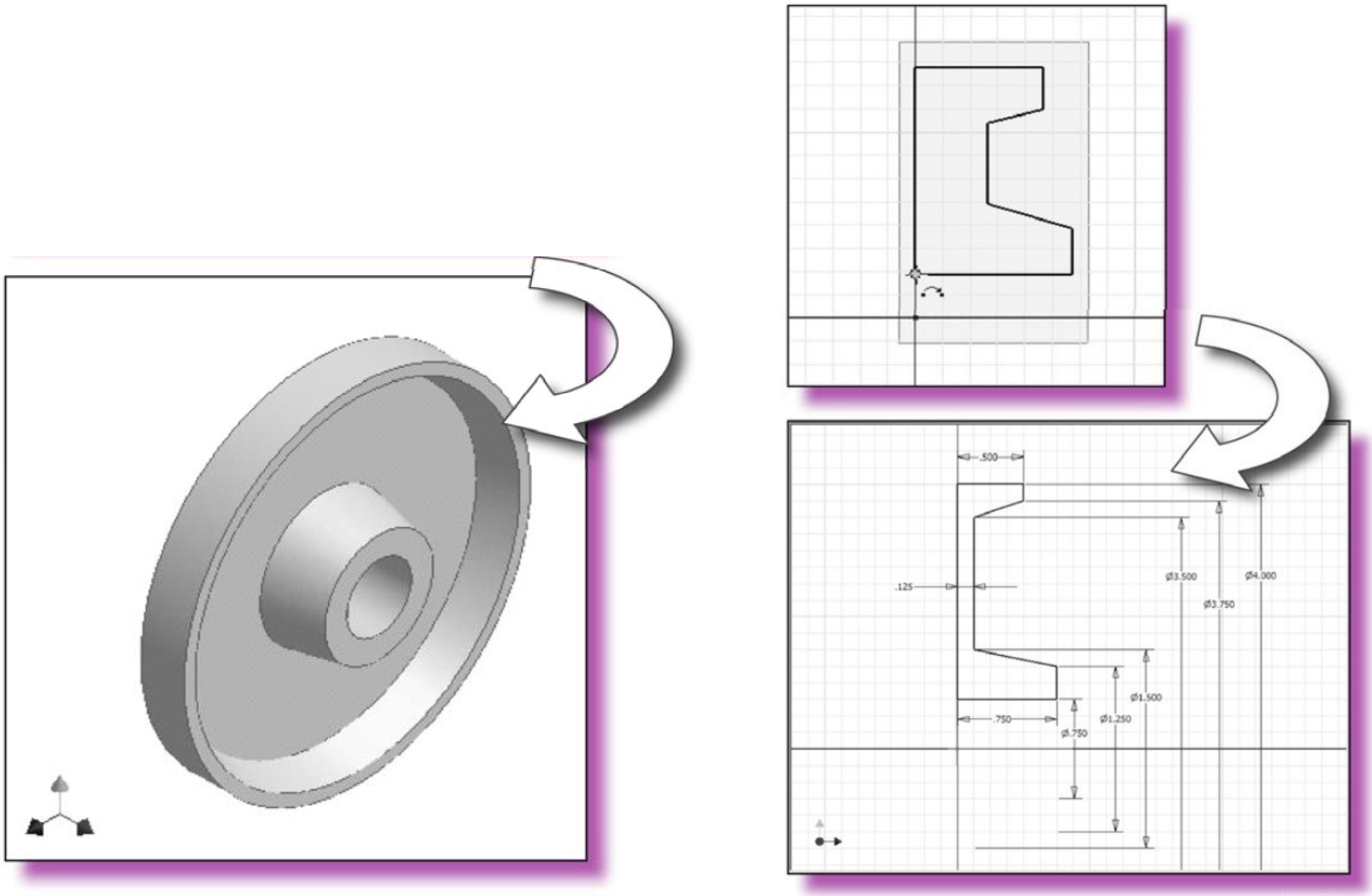
Parametric Modeling

Applying Constrains:

- **Lock**: Creates a lock constraint on the selected item by setting vertical and horizontal dimensions relative to the origin (dimensions can be edited afterwards).
- **Coincident**: Creates a coincident (point-on-point) constraint between two selected points.
- **Point On Object**: Creates a point-on-object constraint on selected items.
- **Horizontal Distance**: Fixes the horizontal distance between 2 points or line ends. If only one item is selected, the distance is set to the origin.
- **Vertical Distance**: Fixes the vertical distance between 2 points or line ends. If only one item is selected, the distance is set to the origin.
- **Vertical**: Creates a vertical constraint to the selected lines or polylines elements. More than one object can be selected.
- **Horizontal**: Creates a horizontal constraint to the selected lines or polylines elements. More than one object can be selected.
- **Length**: Creates a length constraint on a selected line.
- **Radius**: Creates a radius constraint on a selected arc or circle.
- **Parallel**: Creates a parallel constraint between two selected lines.
- **Perpendicular**: Creates a perpendicular constraint between two selected lines.
- **InternalAngle**: Creates an internal angle constraint between two selected lines.
- **Tangent**: Creates a tangent constraint between two selected entities, or a colinear constraint between two line segments.
- **Equal Length**: Creates an equality constraint between two selected entities. If used on circle or arcs, the radius will be set equal.
- **Symmetric**: Creates a symmetric constraint between 2 points with respect to a line.



Parametric Modeling



Parametric Modeling

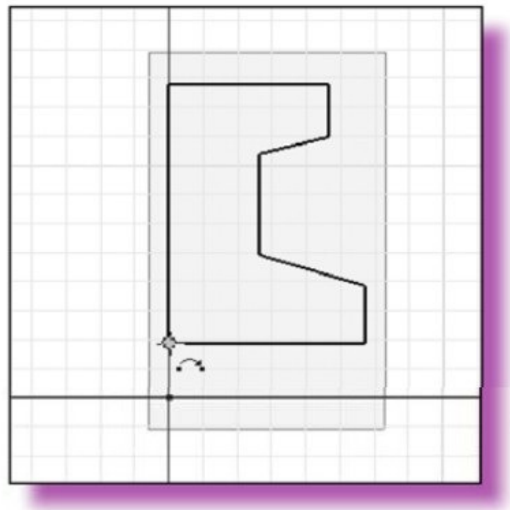
Features:

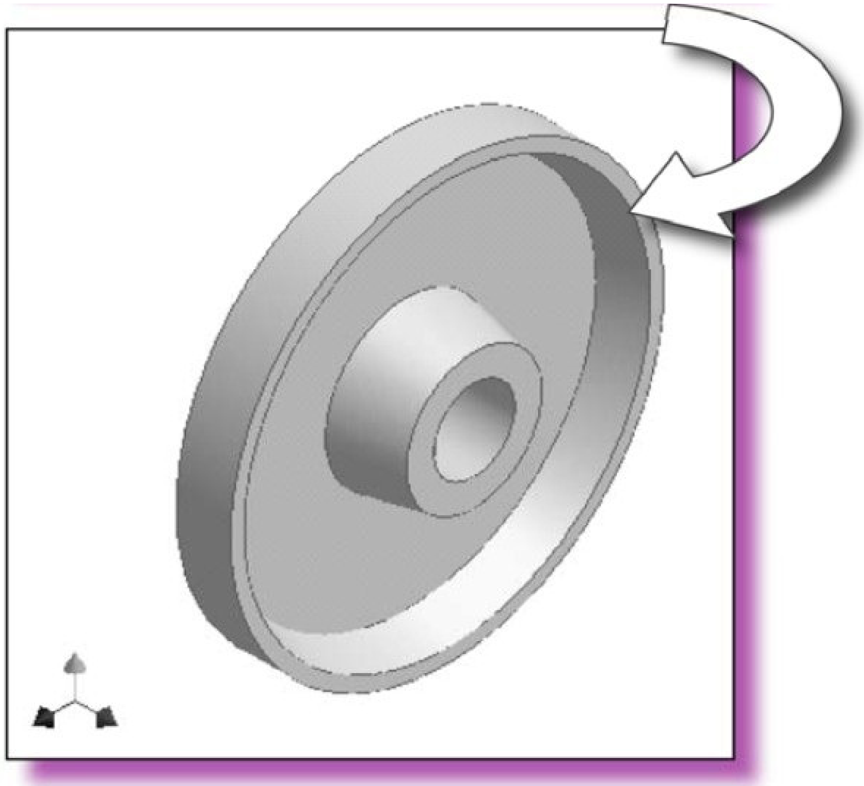
- Extrude

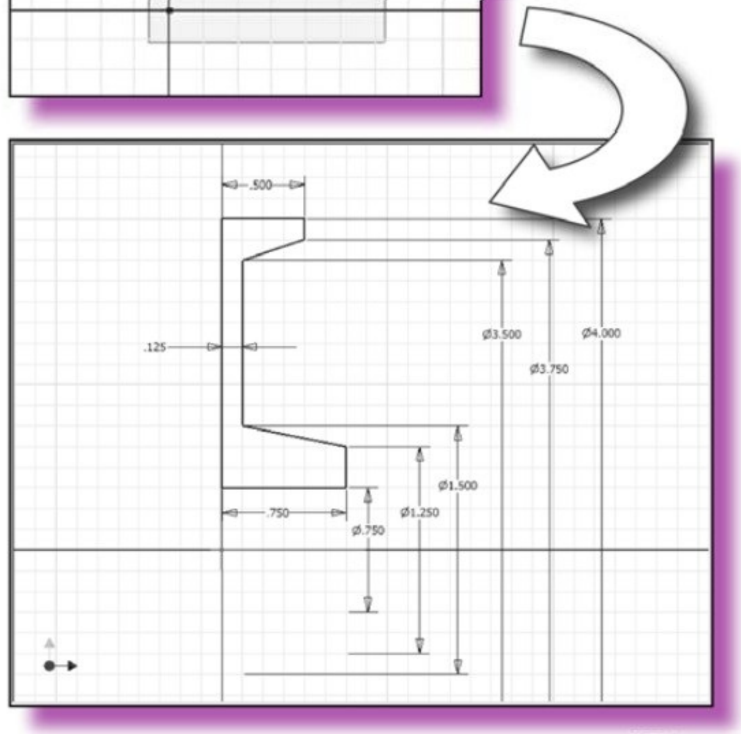
- Revolute

- Sweep

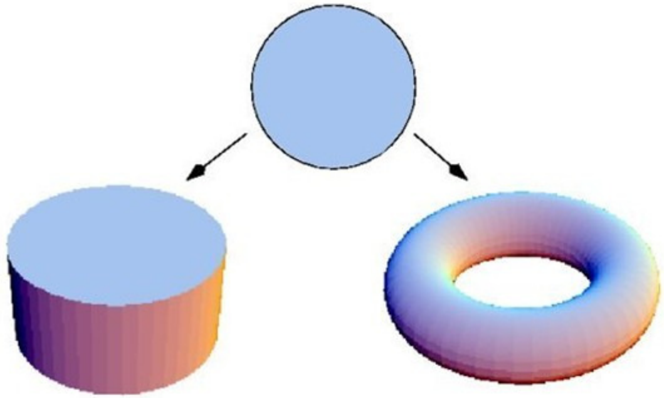
- Loft

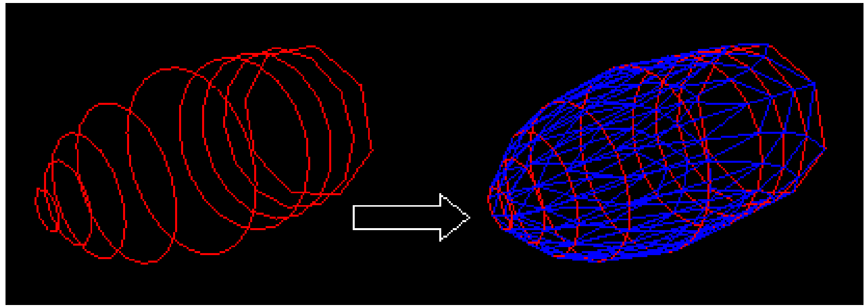


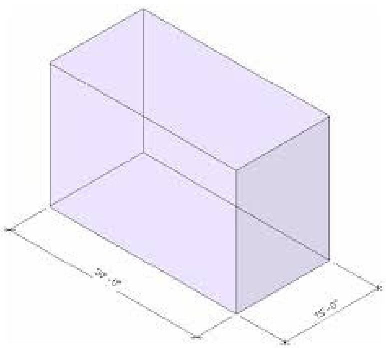


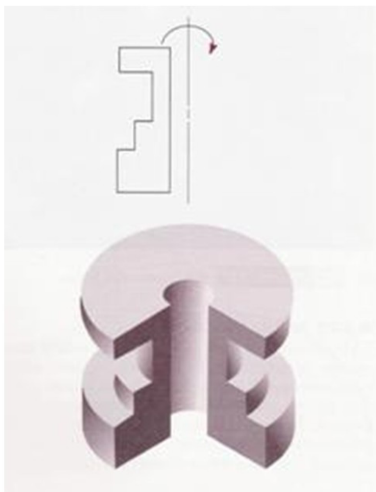


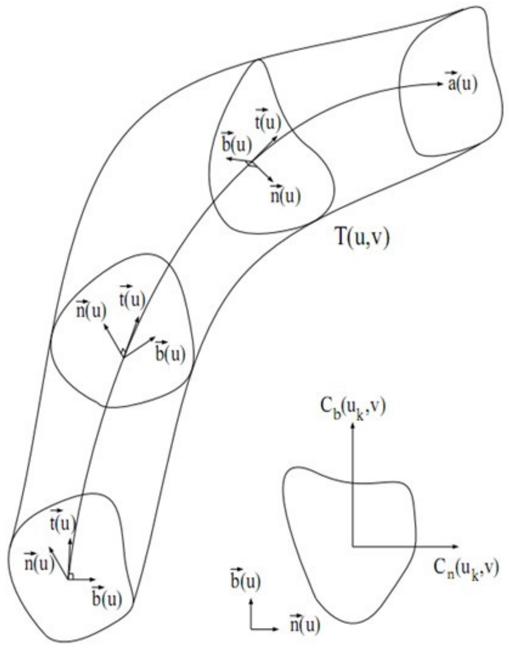
Parametric Modeling









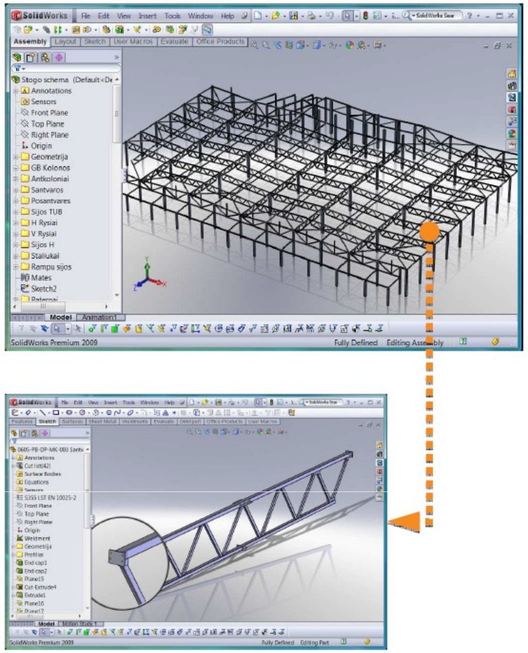


Parametric Modeling

Parametric Modeling

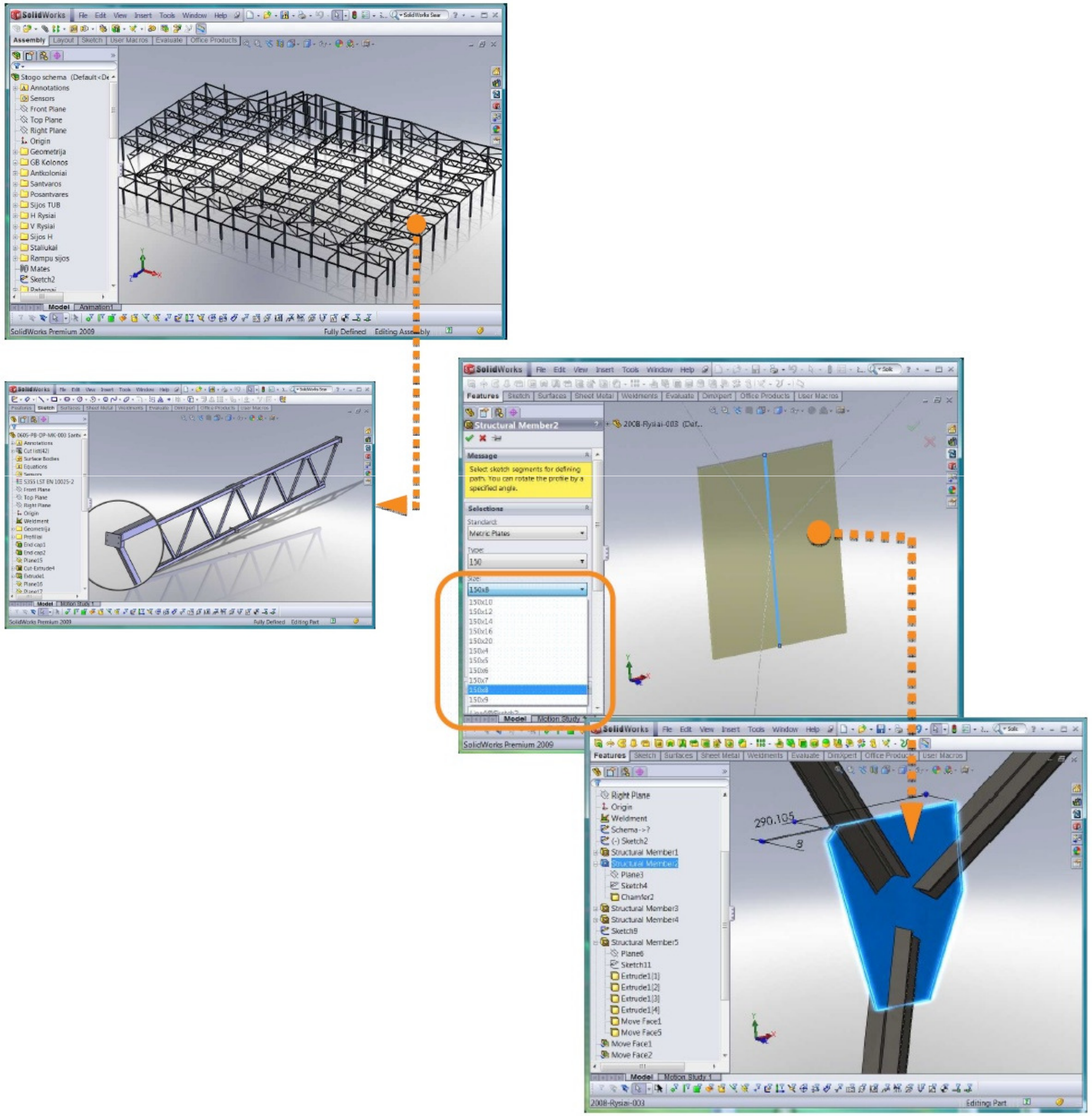
[POPOV2009]

Parametric Modeling



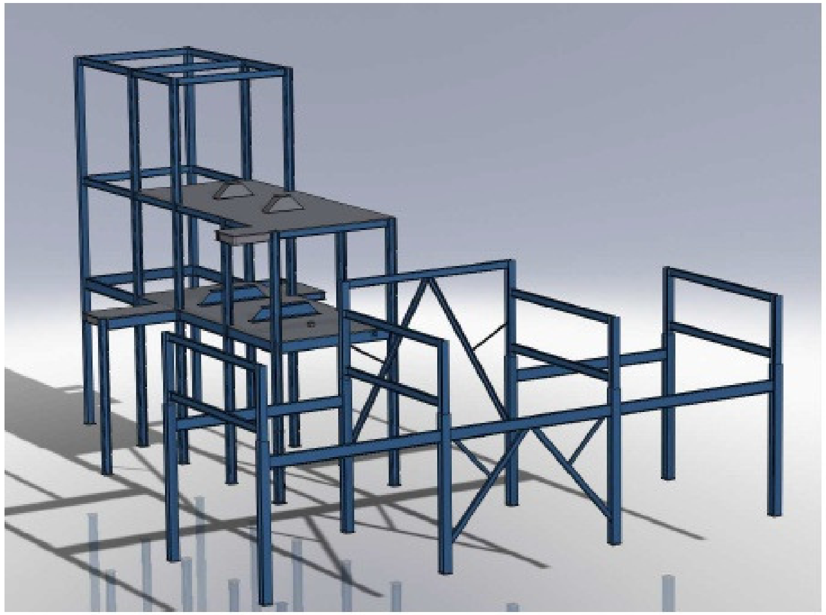
[POPOV2009]

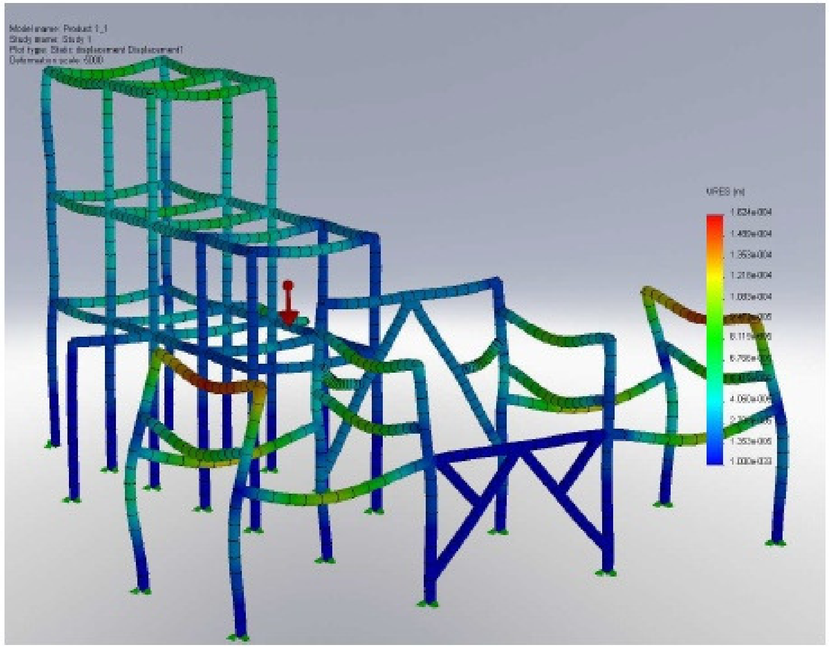
Parametric Modeling



[POPOV2009]

[POPOV2009]



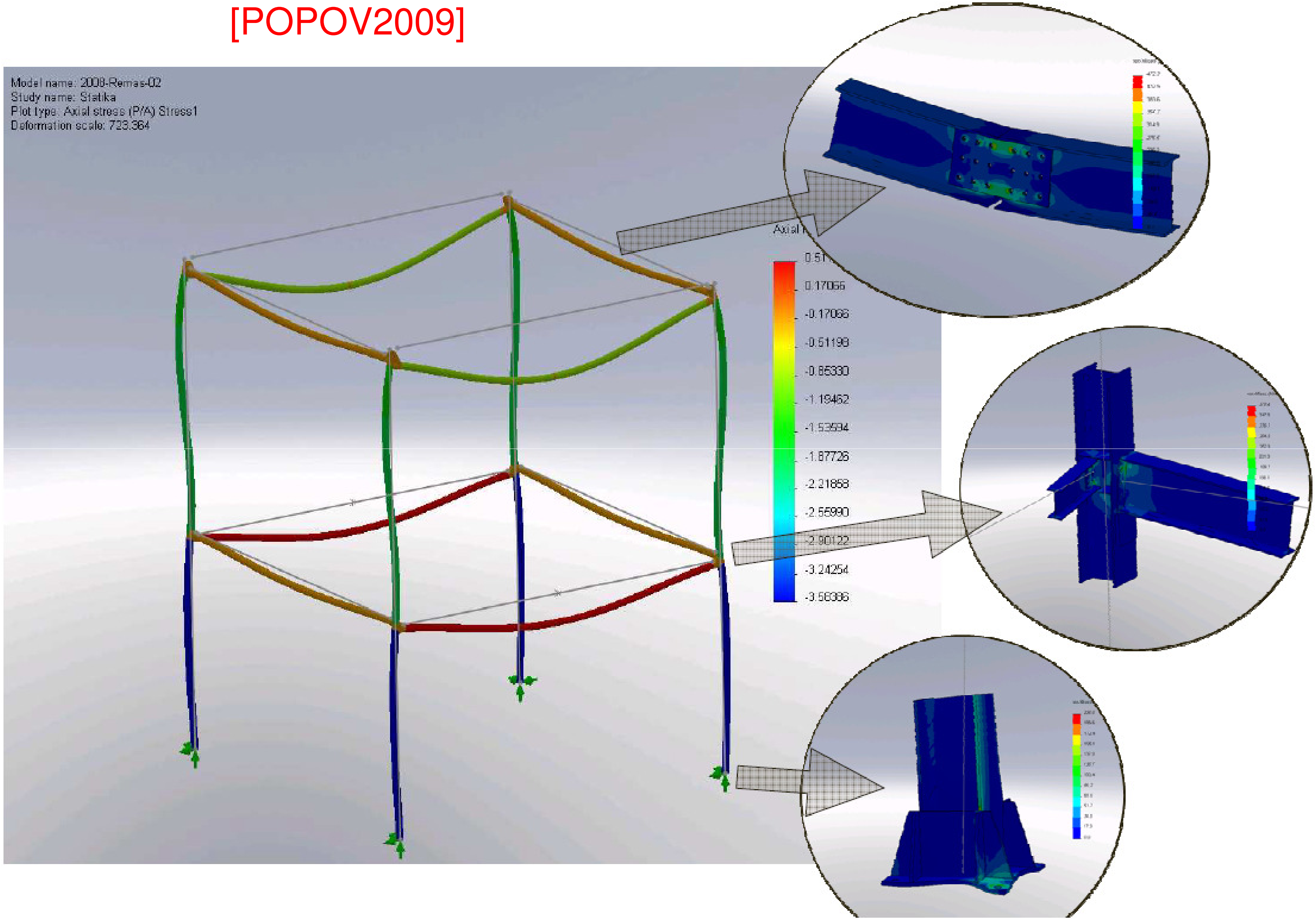


Model name: Product 1_1
Study name: Study 1
Plot type: Static displacement Displacement1
URES (m)

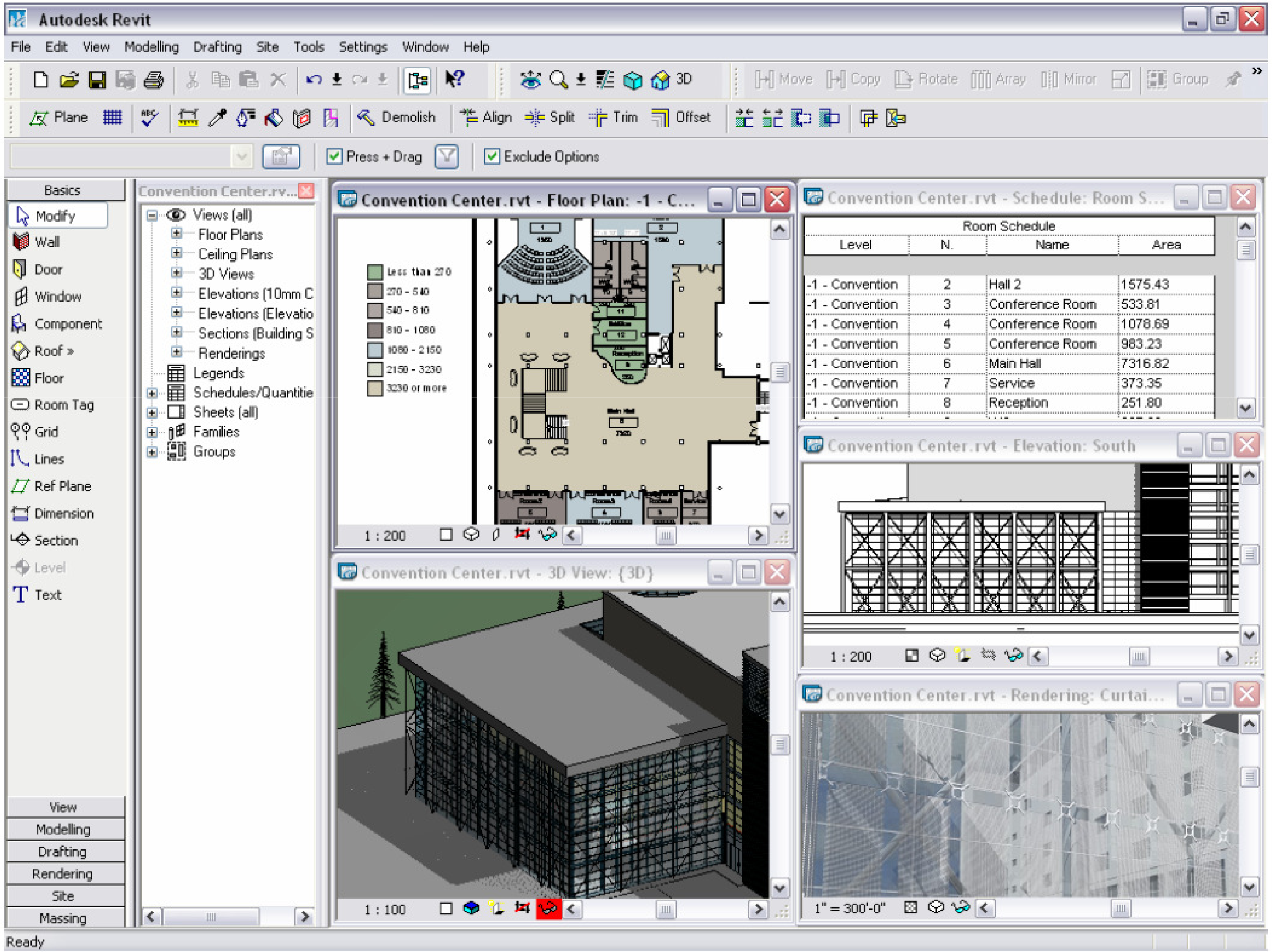
[POPOV2009]


2008-Remas-02 (00-Default<Display State-2>)
Sensors
Annotations
Front Plane
Top Plane
Right Plane
Origin
(f) 2008-Remas-01<1> (Default-1<Display Sta
2008-Sija-02<1> -> (Default<As Machined>)
Mates
LocalLPattern1
Cut-Extrude1
Cut-Extrude2
Plane1
Cut-Extrude3
Cut-Extrude4
Cut-Extrude5
Cut-Extrude6
Cut-Extrude7
Cut-Extrude8
Point2
Coordinate System1
Point4
Point6
Coordinate System2
Point8
Point10
Coordinate System4
Coordinate System5
Point12
Coordinate System6

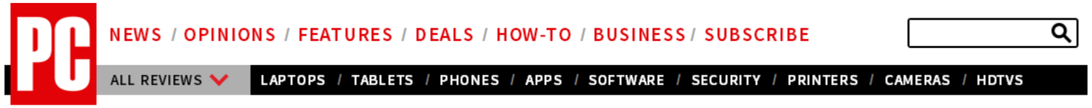
[POPOV2009]

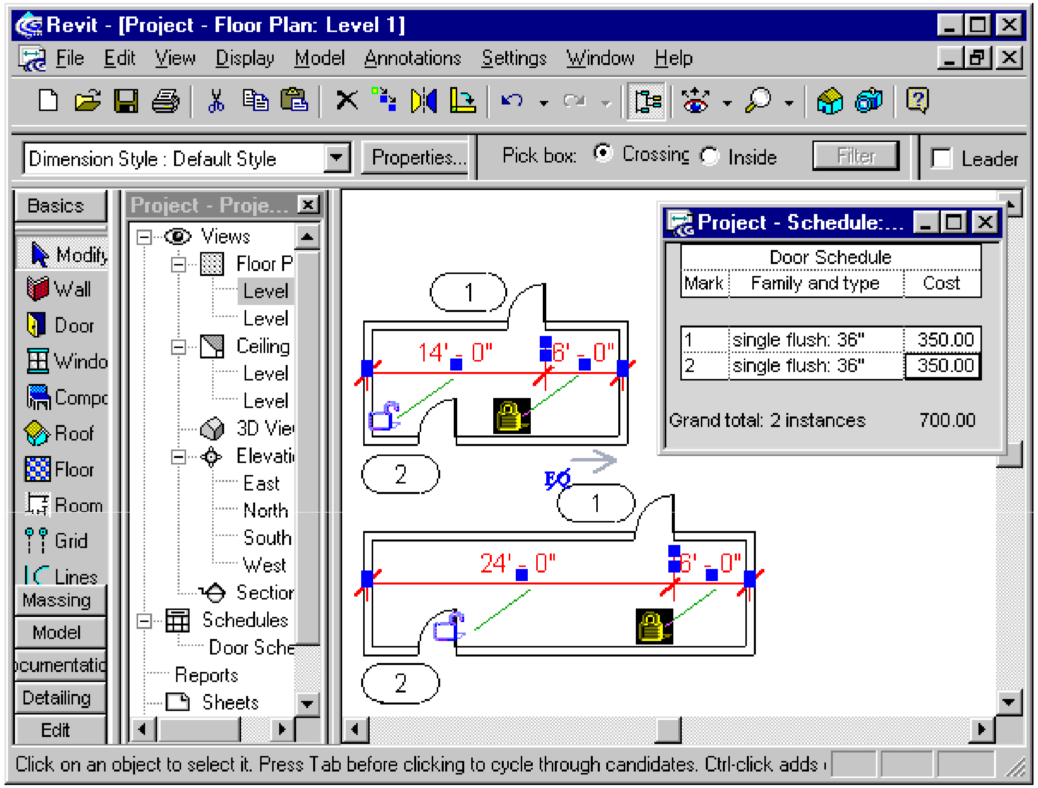

Model name: 2008-Remas-02
Study name: Statika
Plot type: Axial stress (P/A) Stress1
Deformation scale: 723.364
0.51
0.17066
-0.17066
-0.51198
-0.85330
-1.19462
-1.53594
-1.87726
-2.21858
-2.55990
-2.90122
-3.24254
-3.58386

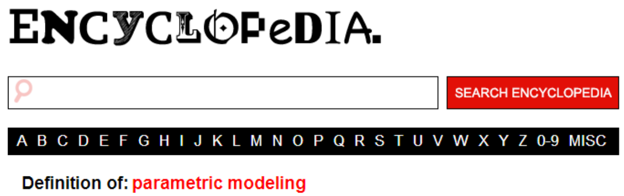
Parametric Modeling



BIM







[PCMag2014]

The door in this room has been "locked" to four feet from the right wall. When the wall is dragged to the right to make the room larger, the door maintains its relationship with the wall. This screen shot is in Autodesk Revit, the first parametric building modeler to tie together all component views and annotations parametrically for the A/E/C industry. In addition, the program maintains automatic interaction between graphic and schedule views (note door schedule at right). If either one is changed, its counterpart is updated. (Screen shot courtesy of Autodesk, Inc., www.autodesk.com)

References

[HOFFMANN1992]
Christoph M. Hoffmann 1992
Geometric and Solid Modeling
https://www.cs.purdue.edu/homes/cmh/distribution/books/geo.html

[PCMag2014]
PC Magazine 2014
Encyclopedia: Parametric Modeling
http://www.pcmag.com/encyclopedia/term/48839/parametric-modeling

[SHIH2006]
Randy Shih 2006
Parametric Modeling: The new CAD Paradigm for Mechanical Designs

[POPOV2009]
Vladimir Popov, Andrej Jarmolajev 2009
Integrated Design and Analysis Applications for Structural Steelwork and Plant Systems