4. Product Data

4.1 Geometrical Data

Requirements and Methods to Manage Geometrical Data for Engineering Applications

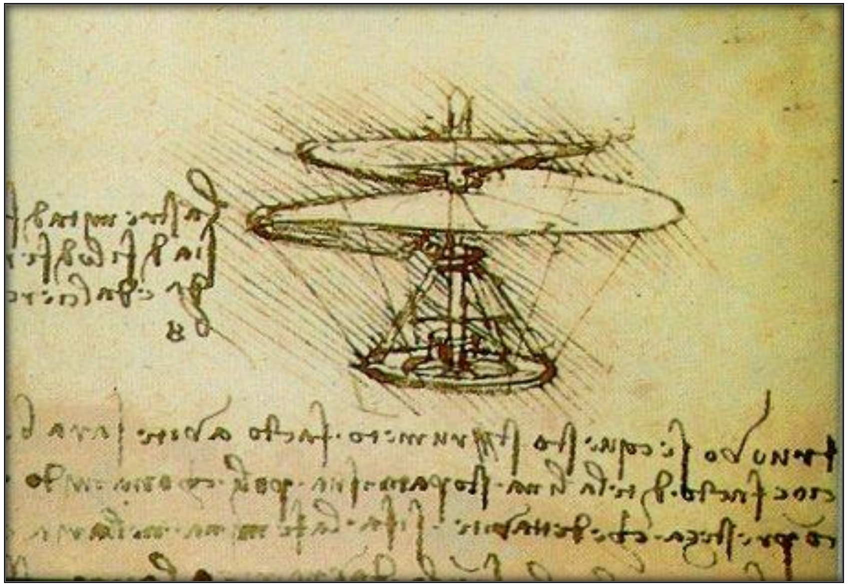
Overview

- Geometric Modeling
  - Applications
  - Historical roots of Technical Modeling
- Overview of Geometrical Models
  - Criteria
  - Wire-frame Models
  - CSG
  - Voxel/Octrees
  - Triangle Meshes (Polygon Meshes)
  - B-Rep
- The Boundary Representation-Model
  - Basics
  - Primitives and Basic Data Structures
  - Modeling Kernels and File Formats
  - B-REP Data in Databases

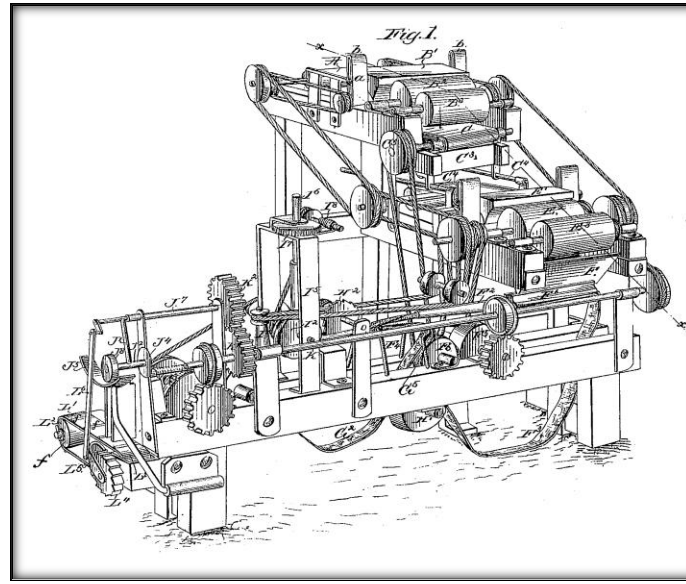
Historical Roots of Technical Modeling

- **Historical roots** date back more than 2000 years
  - Sketches and informal drawings used in ancient Egypt and Greece (Euclid, 300BC) to medieval times
  - Move from agricultural to industrial age increased importance of sharing information for technical development and documentation
  - Around the 19th century patents (protection of intellectual property) required formalization of technical
- Manual **Technical Drawing** on paper standard way for technical modeling until the 1980s
  - Formalized process with commonly used conventions for representing 3D geometries represents “visual language”
  - Projection methods (orthogonal, parallel, perspective) to map 3D geometries to 2D
  - Data representing concrete measures with special syntax as dimension values or parameters and legends
- First **Computer Aided Design** (CAD) developed in the 1960’s
  - Became industrial practice in the 1980s
  - Required digital representation of geometries

Historical Technical Drawings



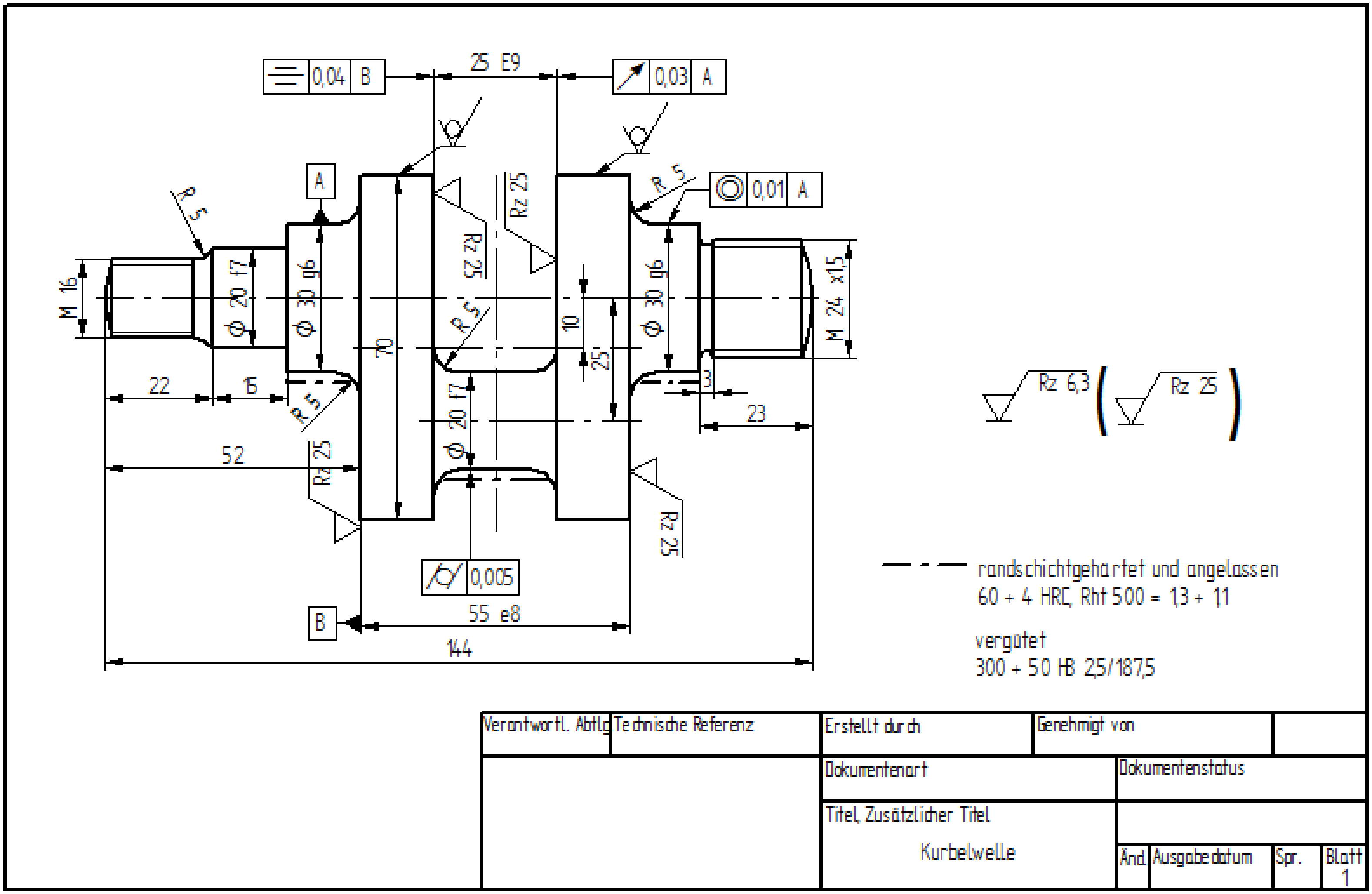
Technical drawing describing details of a helicopter by Leonardo da Vinci



Drawing of a US patent (cigarette rolling machine)from 1881

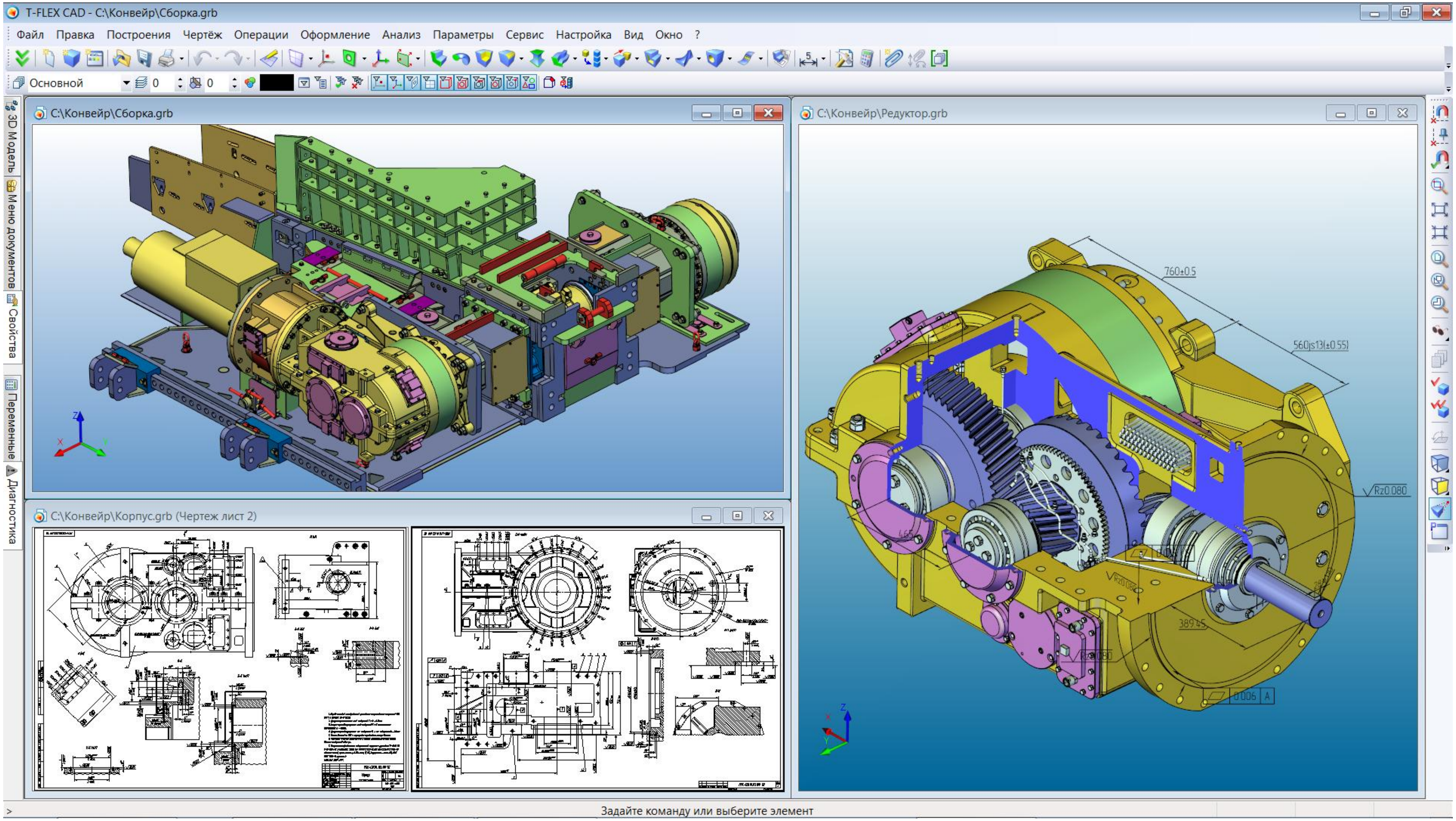
[Source: wikipedia.org]

Technical Drawings



[Source: wikipedia.org]

CAD Systems



[Source: wikipedia.org]

Geometric Modeling

Geometric modeling refers to methods and data structures suitable to represent the shape and topology of geometric objects as data for computer applications.

- In Engineering also referred to as **Solid Modeling** highlighting physical properties of solid objects
- Geometric model has central importance for product lifecycle
  - Based on specifications and requirements from
  - Result of initial design steps and step-wise refinements
  - Input for further steps in product development like FEA, simulation, mockups, production planning, manufacturing, etc.

Computer Graphics vs. CAD

- Computer Graphics
  - General term for methods to create images from data
  - Comprises geometric modeling methods (representing geometry) + rendering (creating image)
  - Geometric modeling focuses on efficiency of computations
  - Two main branches
    - Real-time rendering for fast graphics generation in interactive applications (games, virtual worlds, CAD, etc.)
    - Photo-realistic rendering for application requiring high (realistic) image quality (CGI in movies, computer arts, etc.)

- Computer Aided Design
  - Focus on formal representation of geometrical data
  - Geometric models focus on expressiveness, completeness and correctness of geometry
  - Uses methods from computer graphics (real-time rendering) for interaction
    - CAD model needs to be mapped to rendering model

Geometry vs. Topology

- Terms for different aspects of representing local and global properties of objects

Geometry: describes local features (dimensions, relations between dimensions, primitive type, etc.) of each element of an object.

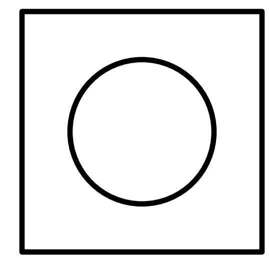
Topology: describes transformations (position in space via translation, rotation, etc.) of elements and how they are connected to form complex shapes.

Classification of Geometric Modeling

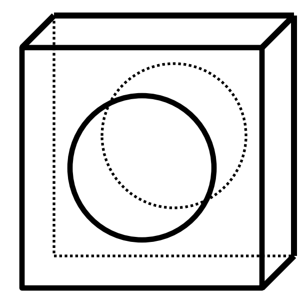
- Over time and for different applications various methods were developed, which differ regarding key criteria
  - Supported Dimensionality
  - Supported Primitives (Geometry)
  - Supported Construction (Topology)
  - Supported Level of Detail/Approximation
  - Intended Applications

Criteria: Dimensionality

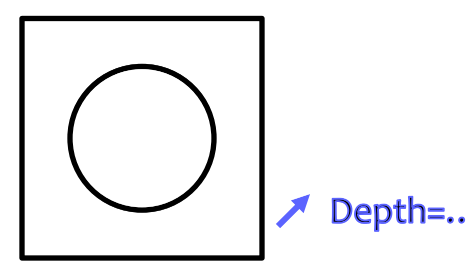
- **2D**
  - Derived from paper 2D drafting
  - Adapted in early CAD systems, today not often practiced
  - Commonly used in some applications (electronic circuit design, architecture)



- **3D**
  - Constructions represented as 3D shapes
  - Current standard in CAD

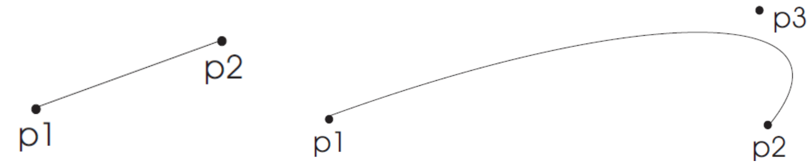


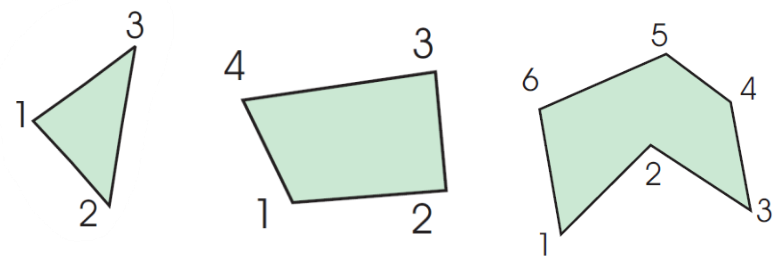
- **2½D**
  - Data represented as 2D + “3D interpretation”
  - Sweeping methods for interpretaion, e.g.
    - Extrusion (along a trajectory)
    - Rotation

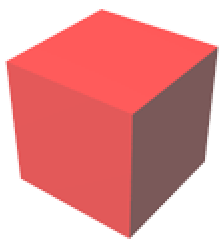


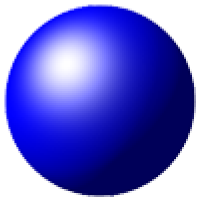
Criteria: Primitives

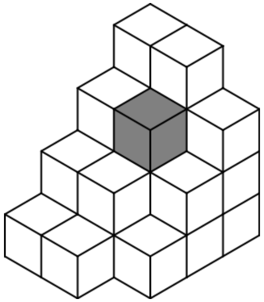
- Points and curves
  - Points, lines, line loops
  - Freeform curves like Splines and NURBS
- Polygons
  - Triangles of special importance as most simple face type (used to construct or approximate any polygon or surface)
- 3D objects
  - Basic 3D objects like prisms, spheres, cuboids, etc.
- Space partitions
  - Part of space according to a 3D grid partitioning the room





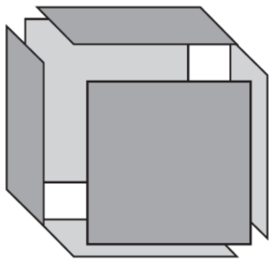




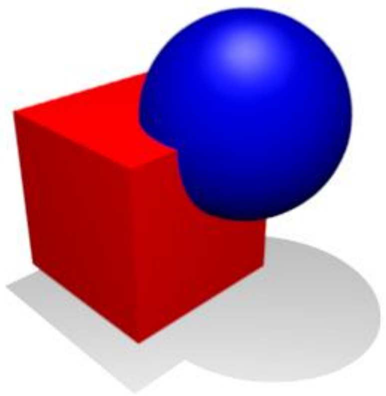


Criteria: Construction Methods

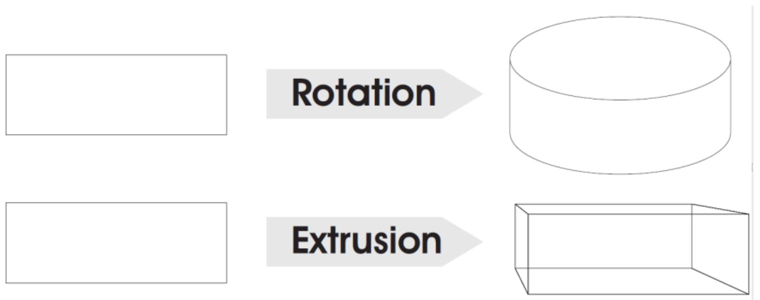
- Basic topology
  - Transformations: translation, rotation, scaling, etc.
  - Connection of vertexes, edges, faces



- Set operations on primitives
  - Union or intersection (symmetric)
  - Relative complement/subtraction (asymmetric)



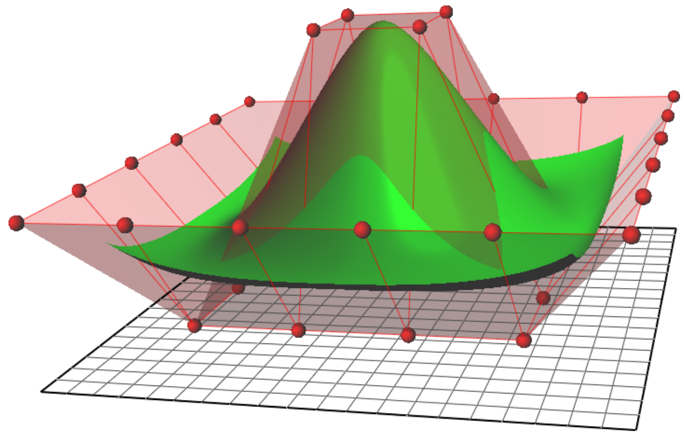
- Sweeping
  - 2½D construction methods like extrusion, rotation etc.



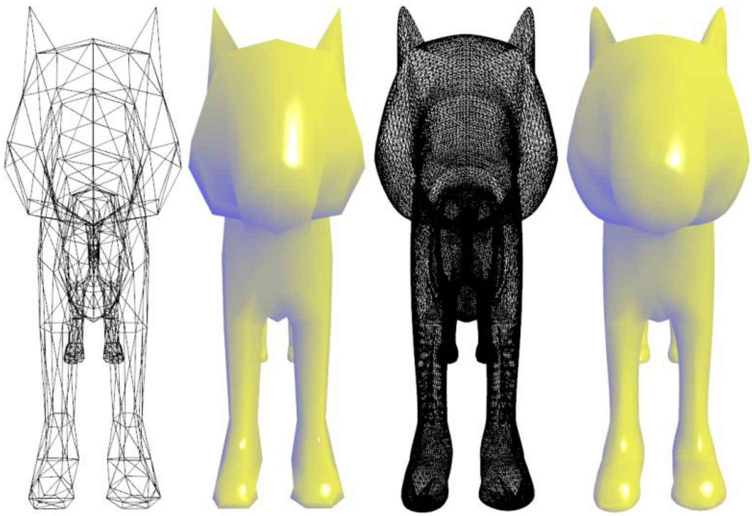
Schallehn: Data Management for Engineering Applications

Criteria: Approximation/Level of Detail

- **Exact Geometry**
  - Often can be represented applying freeform curves (2D) and freeform surfaces (3D), e.g. using NURBS
  - Support for intuitive design



- **Approximated Geometry**
  - Often implemented in terms of **tessellation**: representing a complex surface with simple polygons (e.g. triangle meshes)
  - Level of detail can be set according to requirements
  - More efficient for rendering



[Source: wikipedia.org]

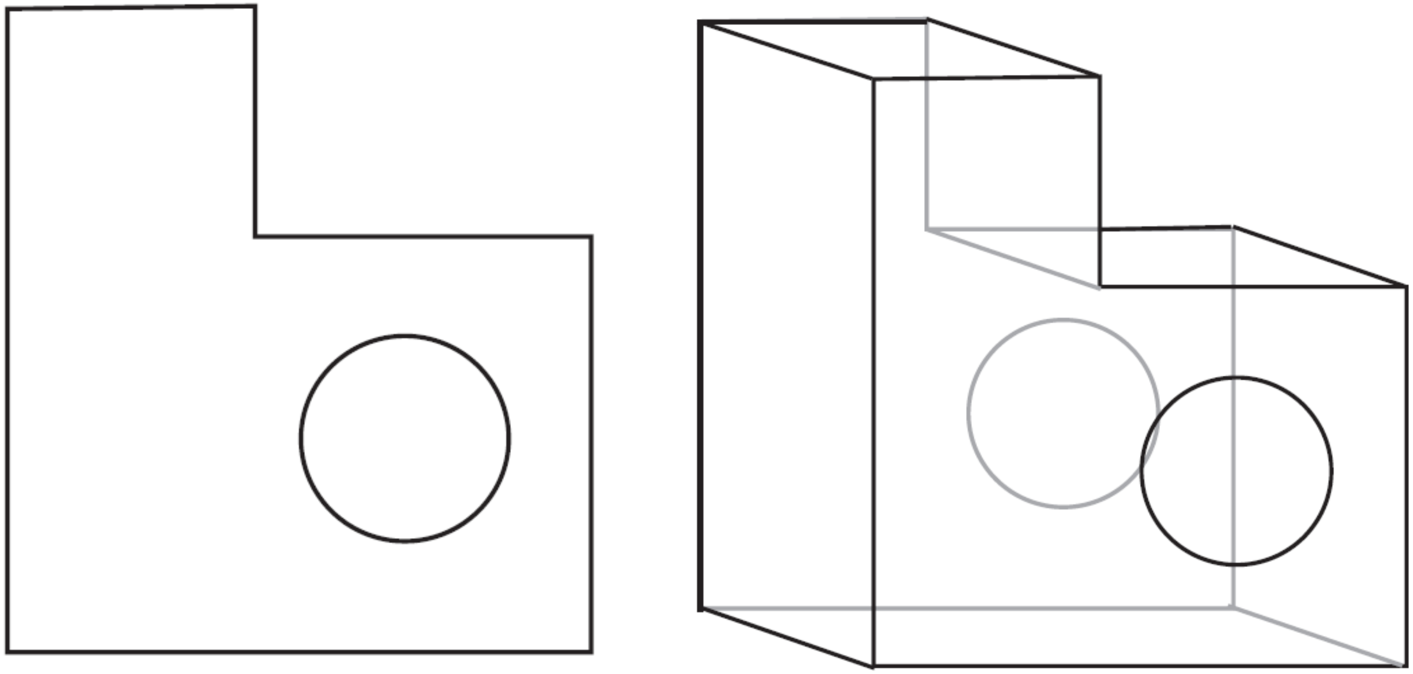
Intended Applications

- Interactive work on geometries
  - Requires intuitive methods and expressiveness
  - Examples:
    - CAD (development)
    - Game content development
    - CGI/special FX development
- Rendering
  - Requires simple structures and efficient algorithms
  - Examples
    - Games
    - CAD (interaction, display)
    - Interactive virtual worlds
- Capturing real-word geometries
  - As objects semantics are unknown, low-level representation required
  - Examples
    - 3D scanners or printers
    - X-ray computed tomography (CT)
    - Motion Capturing

Overview of Modeling Methods

- Wire-frame Models
- Constructive Solid Geometry (CSG)
- Voxel/Octrees
- Triangle Meshes (Polygon Meshes)
- B-Rep

Wire-Frame Models



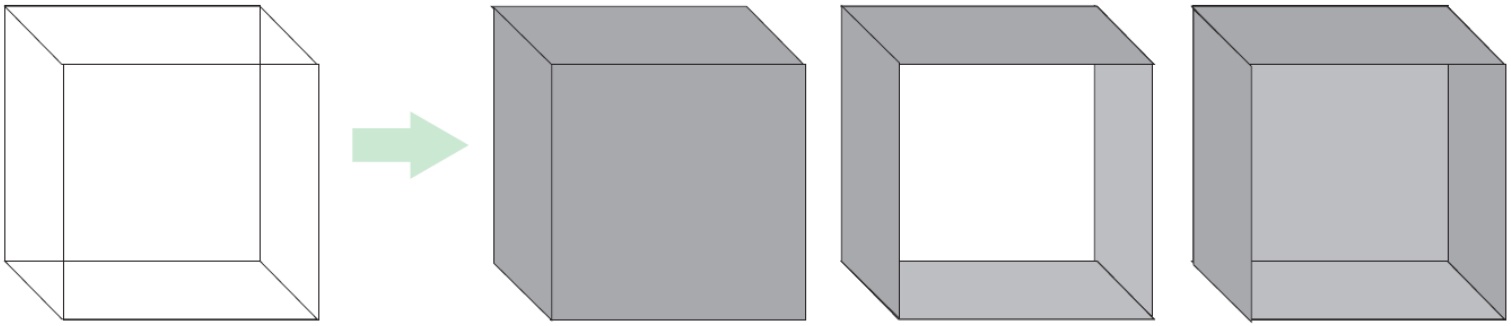
Schallehn: Data Management for Engineering Applications

Wire-Frame Characteristics

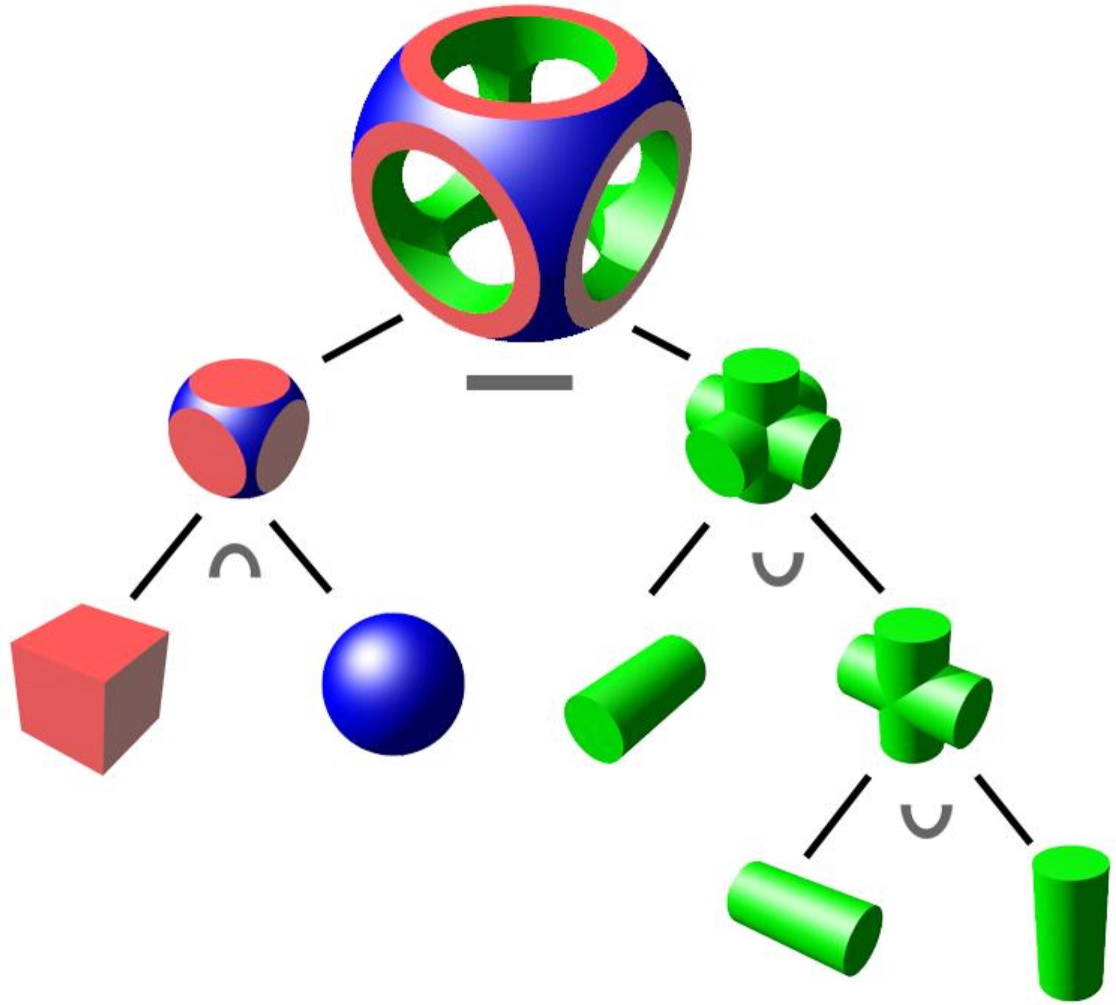
Supported Dimensionality	• 2D • 2½D • 3D
Supported Primitives	• Lines • Curves (splines, ellipses etc.)
Construction Methods	• Edges of physical objects represented by lines and curves
Level of Detail	• Precise description of edges possible • Information about surfaces and volumes lost
Intended Application	• Interactive Modeling: early CAD, simple visualizations • Rendering: early computer graphics

Wire-Frame Usage

- Rendering of 3D models applies
  - Projection methods (perspective, orthogonal, parallel)
  - Hidden edges can be removed, colored or dashed (requires information about surfaces)
- Disadvantages
  - Semantic loss (information about surfaces and volumes)
  - Leads to ambiguities in interpretation



Constructive Solid Geometry (CSG)



[Source: wikipedia.org]

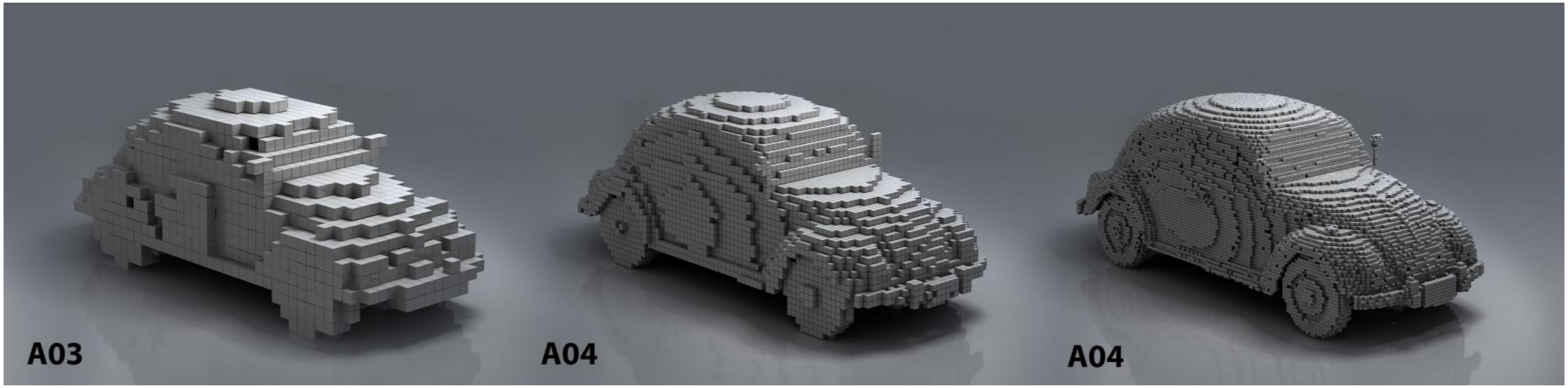
CSG Characteristics

Supported Dimensionality	• 3D
Supported Primitives	• Parametrizable 3D basic shapes: cuboids, spheres, prisms, etc. • In some approaches: parametrizable free-form shapes
Construction Methods	• Boolean set operations • Basic transformations: translation, rotation
Level of Detail	• Many geometries can represented correctly • If no free-form shapes are supported, according geometries have to be approximated
Intended Application	• Interactive Modeling: CAD, modeling for games etc.

CSG Usage

- Rich semantics allow intuitive creation of 3D models
- Disadvantages
  - Rendering and some verifications/evaluation of geometric models require complex computations
  - Construction of a shape by Boolean operators is ambiguous
  - Some geometries (e.g. free-form surfaces such as used for automotive design etc.) had to be approximated with basic primitives in early approaches
- CSG combined with other methods (e.g. B-Rep) to support interactive modeling
- Rarely used as stand-alone geometric model in CAD

Voxel-based Models



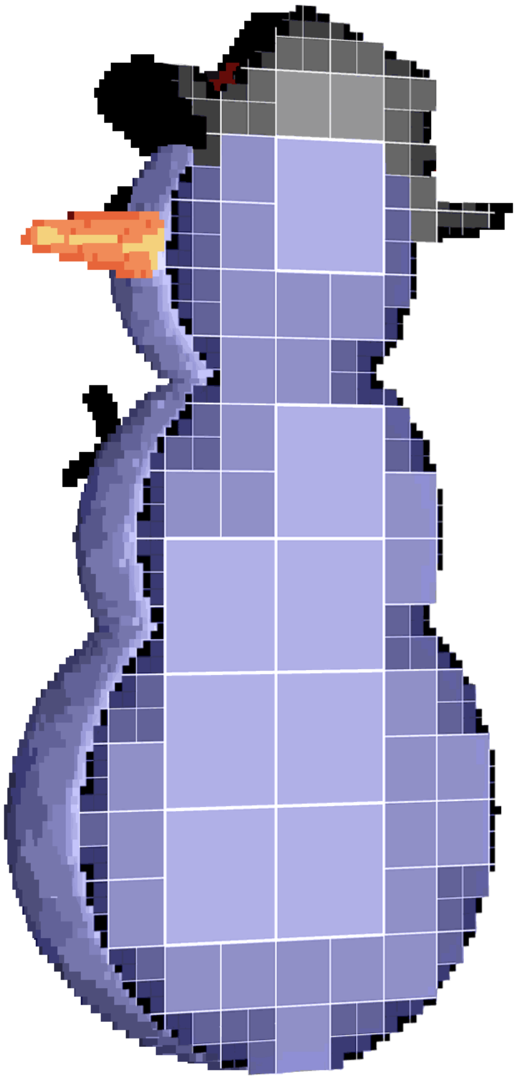
[Source: Tobias Wüstefeld, http://www.bilderzucht.de]

Voxel Characteristics

Supported Dimensionality	• 3D
Supported Primitives	• Volume elements = Voxel = “3D pixel” • Space partitioned according to grid
Construction Methods	-
Level of Detail	• Approximation according to grid properties (resolution)
Intended Application	• Capturing or creating real-world geometries: 3D scanners, 3D printers

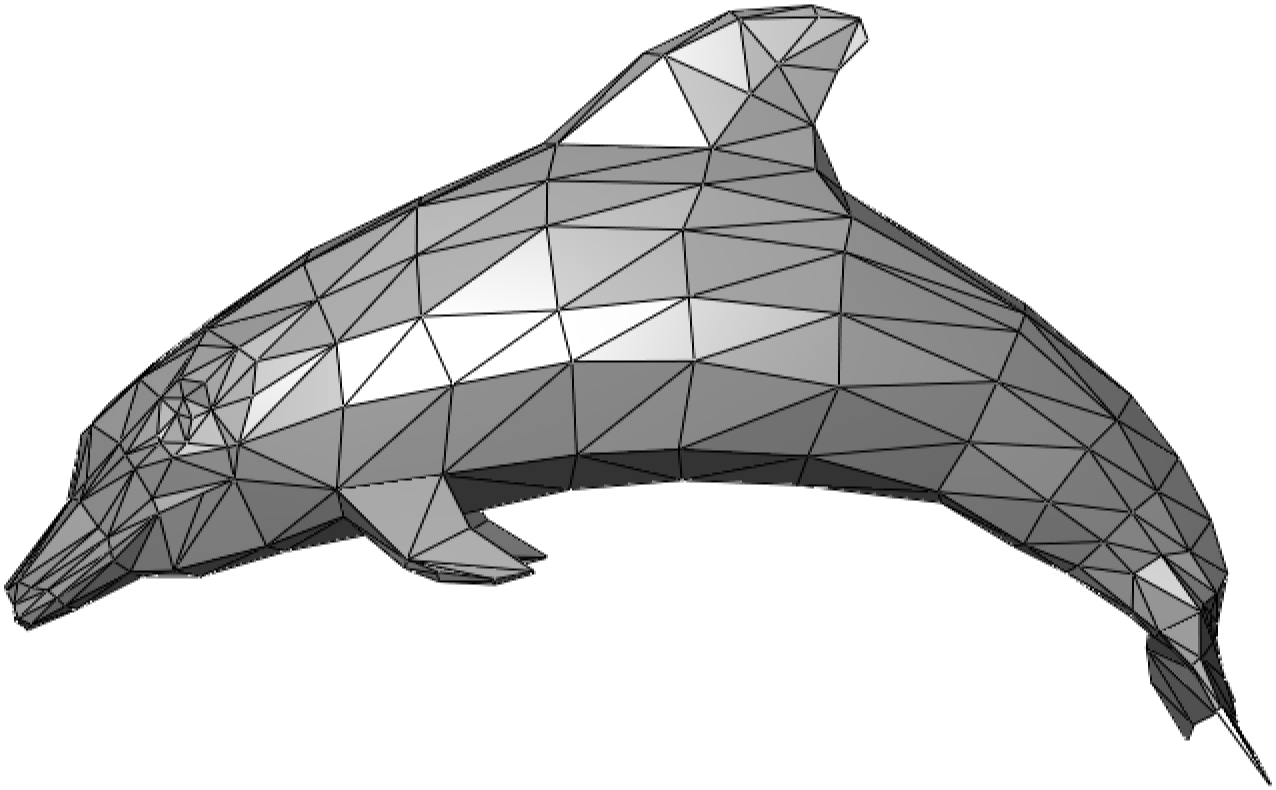
Octrees

- Problem of voxel-based 3D data: huge amounts of data for reasonable resolution
- Could be solved by compression techniques
- Alternative: Octrees
  - Partition space hierarchically
  - Starts with 8 sub-cubes (8=oct)
  - Only cubes which are not completely filled or completely empty are furthermore subdivided recursively
  - Creates (unbalanced tree)
  - 3D equivalent to 2D- Quadtree



[Source: wikipedia.org]

Triangle/Polygon Meshes



[Source: wikipedia.org]

Polygon Mesh Characteristics

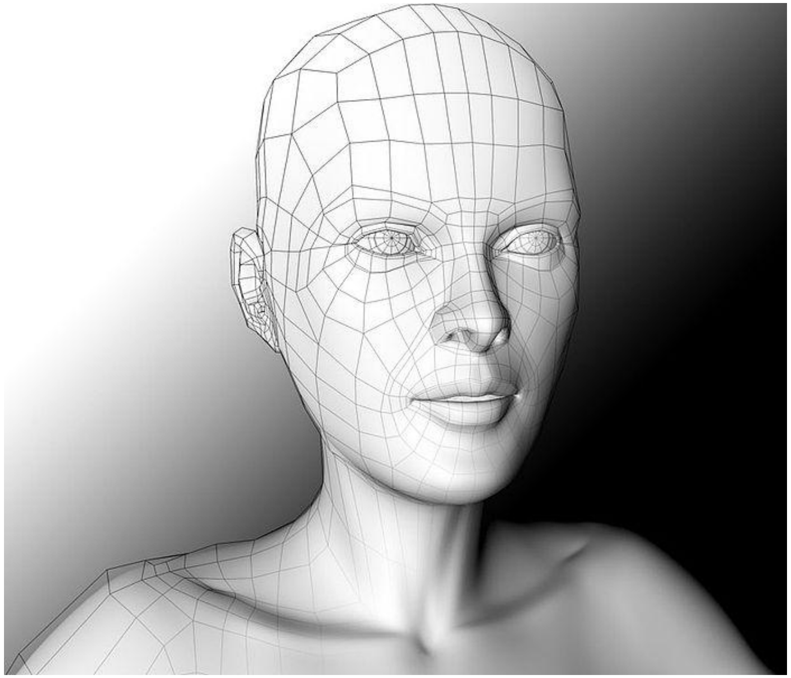
Supported Dimensionality	• 2D (less commonly used) • 3D
Supported Primitives	• Triangles • Other polygons (quads, arbitrary)
Construction Methods	• Basic topology of vertexes of polygons • Combinations like triangle strips and triangle fans for easier definition
Level of Detail	• Requires approximation of all curved surfaces (e.g. sphere, free-form) and edges • Level of detail/approximation can be controlled by number of triangles/polygons
Intended Application	• Real-time rendering: fast computation of graphics for interactive applications

Triangle Mesh Usage

- 3D real-time rendering
  - Based on geometrical projection methods
  - Addition of color, texture, lighting, etc.
- Approximation can be controlled to be below level of perception → higher computation effort
- Specialized hardware (Graphic Processing Units) work on triangle meshes
- More complex models like B-Rep, CSG, etc. are mapped to triangle meshes for rendering/visualization

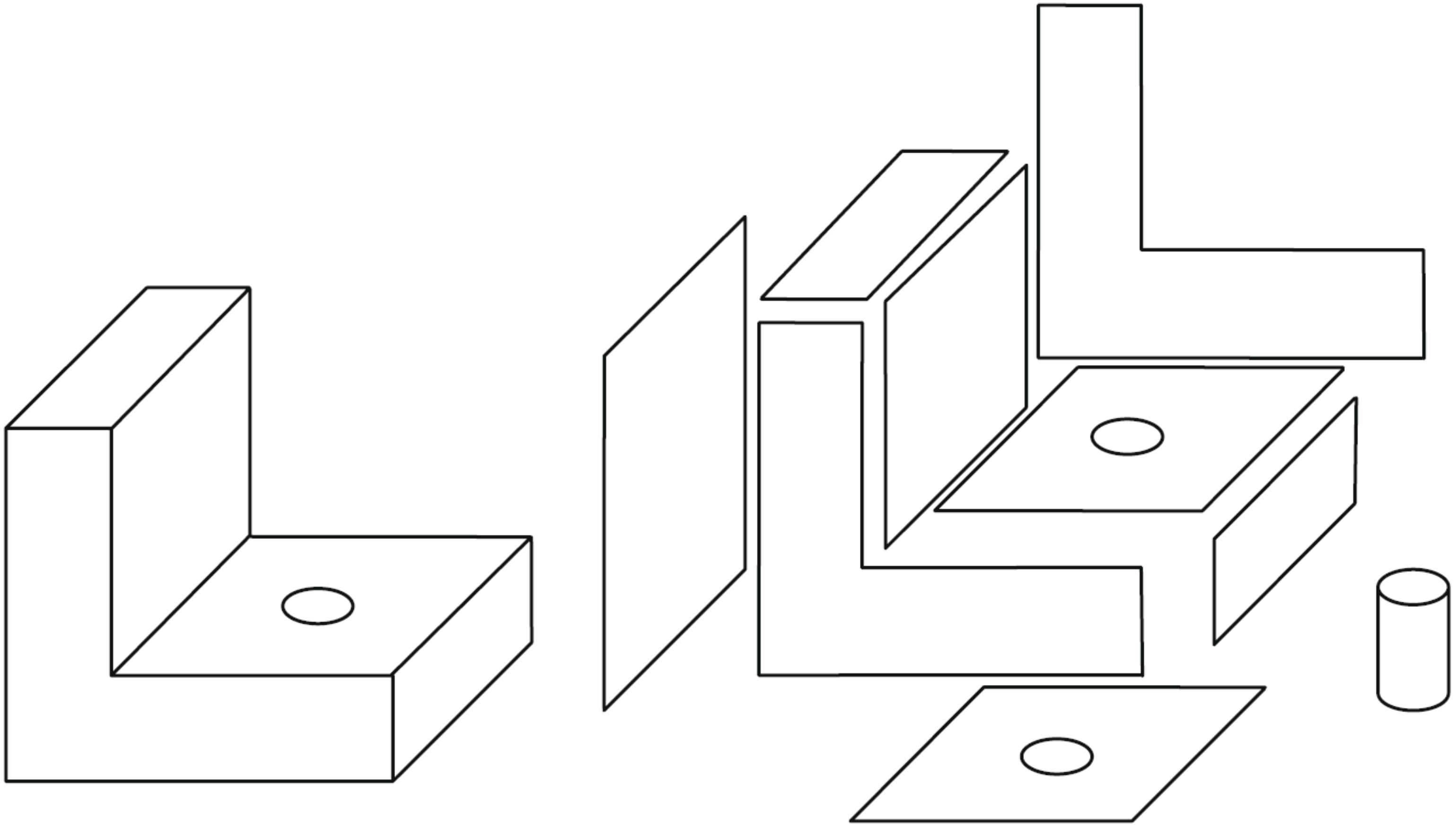
Polygon Meshes

- In general, any polygon can be used to describe/approximate surfaces
- Higher number of vertexes
  - Allows easier modeling
  - Introduces more complex computations
  - verification



[Source: wikipedia.org]

Boundary Representation (B-Rep)



Schallehn: Data Management for Engineering Applications

B-Rep Characteristics

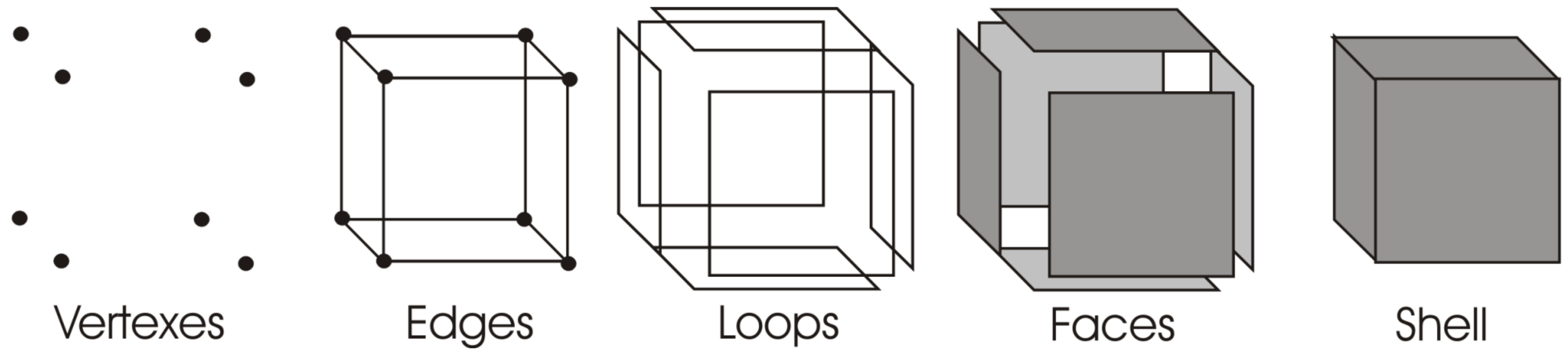
Supported Dimensionality	• 3D
Supported Primitives	• Surfaces modeled by - Vertexes - Edges as lines or diverse curves - Faces as arbitrary planar polygons or free-form surfaces
Construction Methods	• Basic topology polygons (as for polygon meshes) • Basic transformations: translation, rotation • Boolean set operations (as in CSG) • Sweeping to create 3D geometries from 2D shapes
Level of Detail	• Precise description of geometries possible • Approximations possible for lower level of detail
Intended Application	• Interactive work on geometries: CAD etc.

History of B-Rep

- Developed in the early 1970s
  - Ian Braid developed basic concepts and first prototype of a modeling kernel (ROMULUS) from CAD perspective
  - Bruce Baumgart developed basic data structures and algorithms from a computer graphics perspective
- ROMULUS became blueprint for current modeling kernels
  - Parasolid
  - ACIS
- B-Rep was extended over the years
  - Free-form curves and surfaces
  - Set operations (as in CSG)
  - Sweeping
- Because of rich semantics and intuitive modeling capabilities became de facto standard for CAD

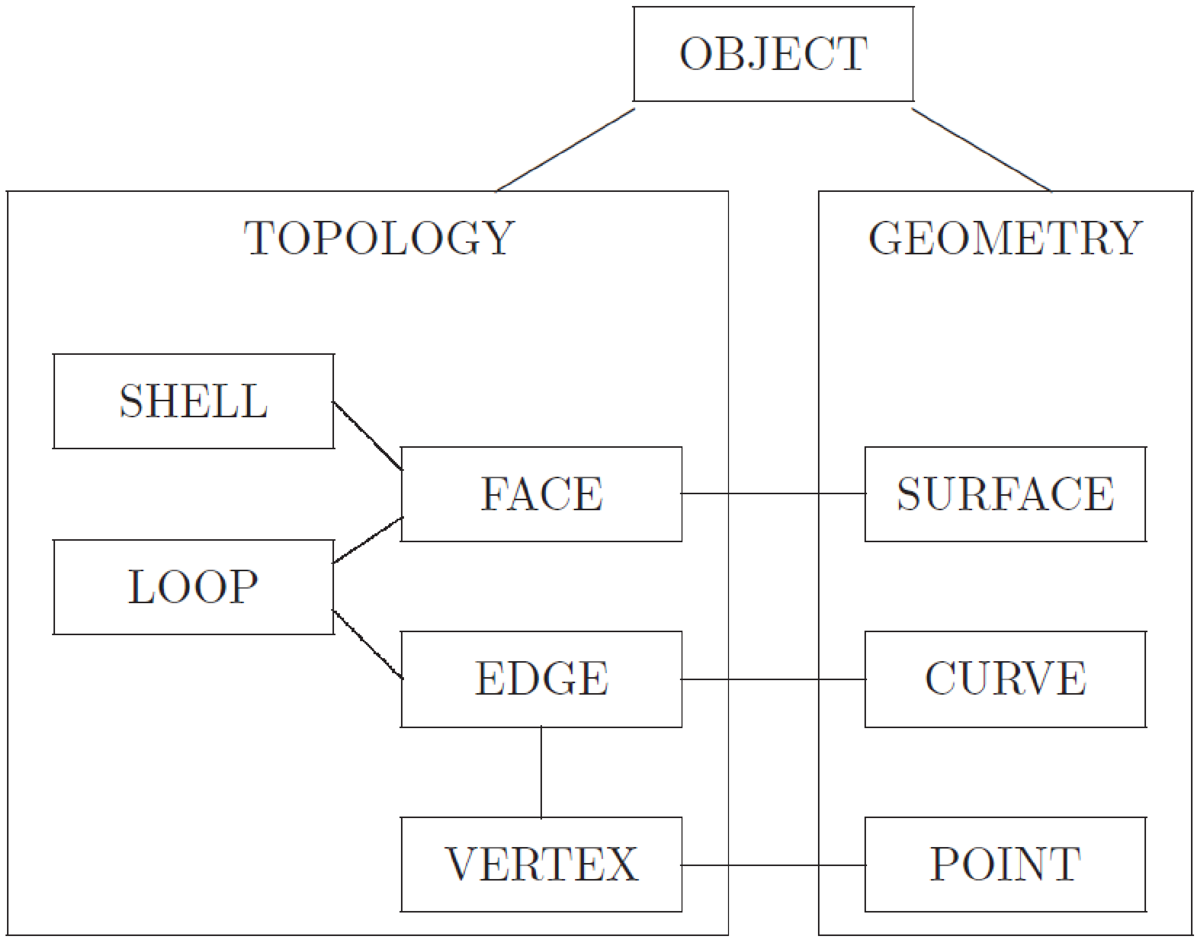
B-Rep Basic Topology

- Hierarchical definition of
  - **Vertexes** defined based on points (geometry)
  - **Edges** defined based on vertexes of lines or curves
  - **Loops** defined as sequence of closed edges
  - **Faces** defined by loops
  - **Shell** defined by enclosing faces, defines body



Schallehn: Data Management for Engineering Applications

B-Rep Topology vs. Geometry

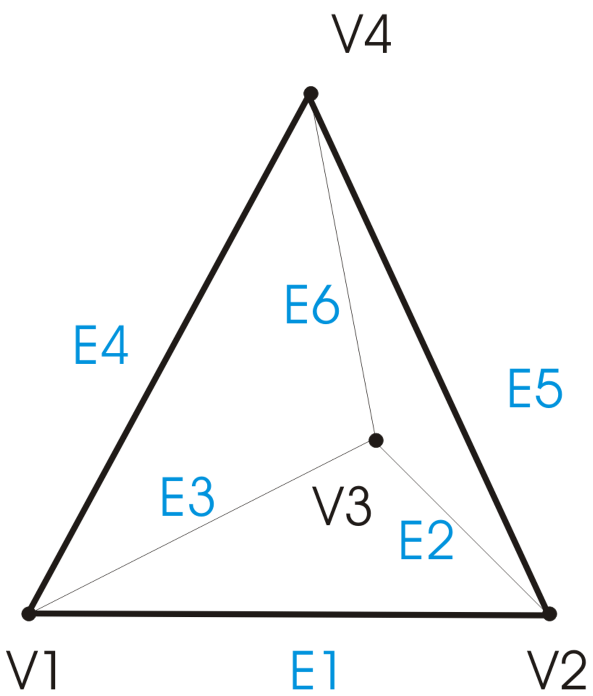


From [1]

B-Rep Data Structures

- Geometry and topological relationships can be stored in simple lists
- More efficient algorithms (validation, rendering, etc.) possible for advanced structures with some redundancies → Winged Edge (also half edge)

B-Rep Simple Data Structures



Vertex	x y z
1	
2	
3	
4	

Edge	Vertex1	Vertex2
1	1	2
2	2	3
3	3	1
4	1	4
5	2	4
6	3	4

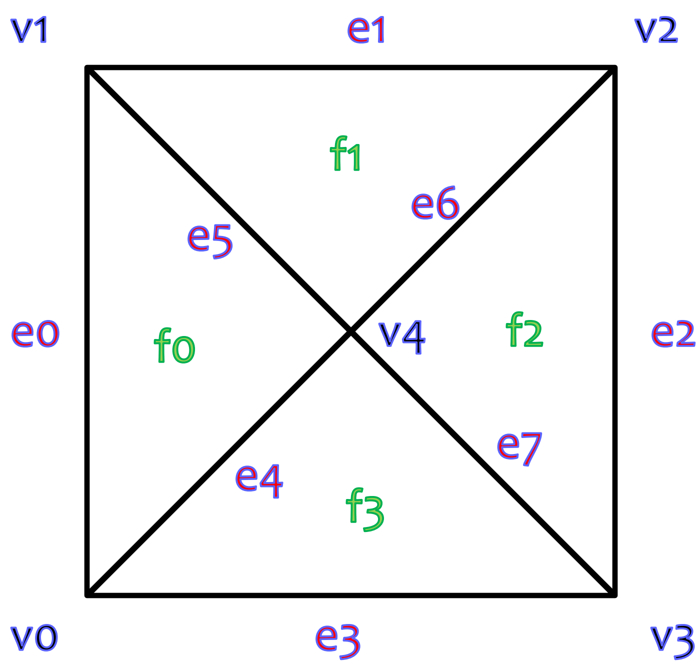
Face	Loop
1	1 2 3
2	1 4 5
3	2 5 6
4	3 4 6

Object	Shell
1	1 2 3 4

Schallehn: Data Management for Engineering Applications

Data Structure: Winged Edge /1

- Winged Edge typical data structure to represent polygon networks defined by vertexes, edges, faces
- Allows fast traversal of surface by keeping (redundant) connections of each edge to vertexes and faces, e.g.



Edge e6 is stored in EdgeList as:

Identifier:	e6			
Defining vertexes:	v2	v4		
Defined faces:	f1	f2		
Neighbor edges:	e1	e2	e5	e7

Data Structure: Winged Edge /2

- Stores for each edge
  - Defining vertexes
  - Defined faces
  - Neighbor edges
  - Actual data (e.g. curve function)
- Stores for each vertex
  - Defined edges
  - Actual data (e.g. coordinates)
- Stores for each face
  - Defining edges
  - Actual data

```
class WE_Edge {
  WE_Vertex vert1, vert2;
  WE_Face aFace, bFace;
  WE_Edge aPrev, aNext, bPrev, bNext;
            // clockwise ordering
  WE_EdgeDataObject data;
}

class WE_Vertex {
  List<WE_Edge> edges;
  WE_VertexDataObject data;
}

class WE_Face {
  List<WE_Edge> edges;
  WE_FaceDataObject data;
}
```

[Source: Winged Edge at wikipedia.org]

Validation of B-Rep Models

- Construction methods alone do not guarantee a valid geometry
- Most common problem: is the described 3D shape closed by defining surfaces?
- Simple geometries such as polyhedrons (plane edges and surfaces) can be checked e.g. by Euler-Poincaré formula:

$$V - E + F \quad = 2 * (S - R) + H \qquad \text{(with holes)}$$
$$V - E + F \quad = 2 \qquad \text{(without holes)}$$

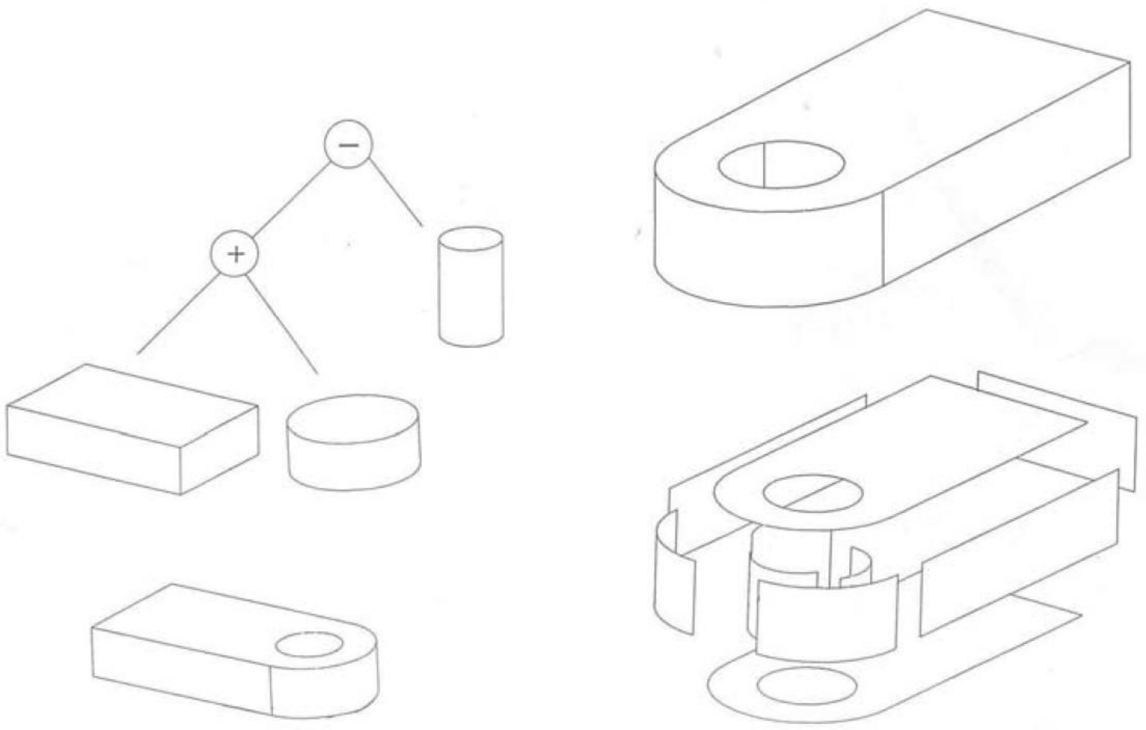
  with V = number of vertexes, E = number of edges, F = number of faces, S = number of shells, R = number of rings (holes in body), H = number of holes (in faces)
- Curves and curved surfaces may require more complex checks

B-Rep: Sweeping Methods

- **Extrusion:** extension of a 2D shape along an arbitrary vector or depth orthogonal to face definition
- **Protruding:** extending an existing 3D geometry by extrusion of a marked surface region
- **Revolve (Rotation):** space covered by revolving a 2D shape around a specified rotation axis
- **Blending:** room covered by transition of one 2D geometry to another
- **General Sweeping:** may extend 2D geometry along any arbitrary curve (free-form)

CSG in B-Rep

- Set operations as defined in CSG can be used on any set of shells created by basic topology or sweeping
- Alternative methods can be used to achieve same design



From [1]

B-Rep Modeling Kernels

- Early implementations: BUILD and Romulus
- Currently two dominating

ACIS (Alan, Charles, Ian's System)

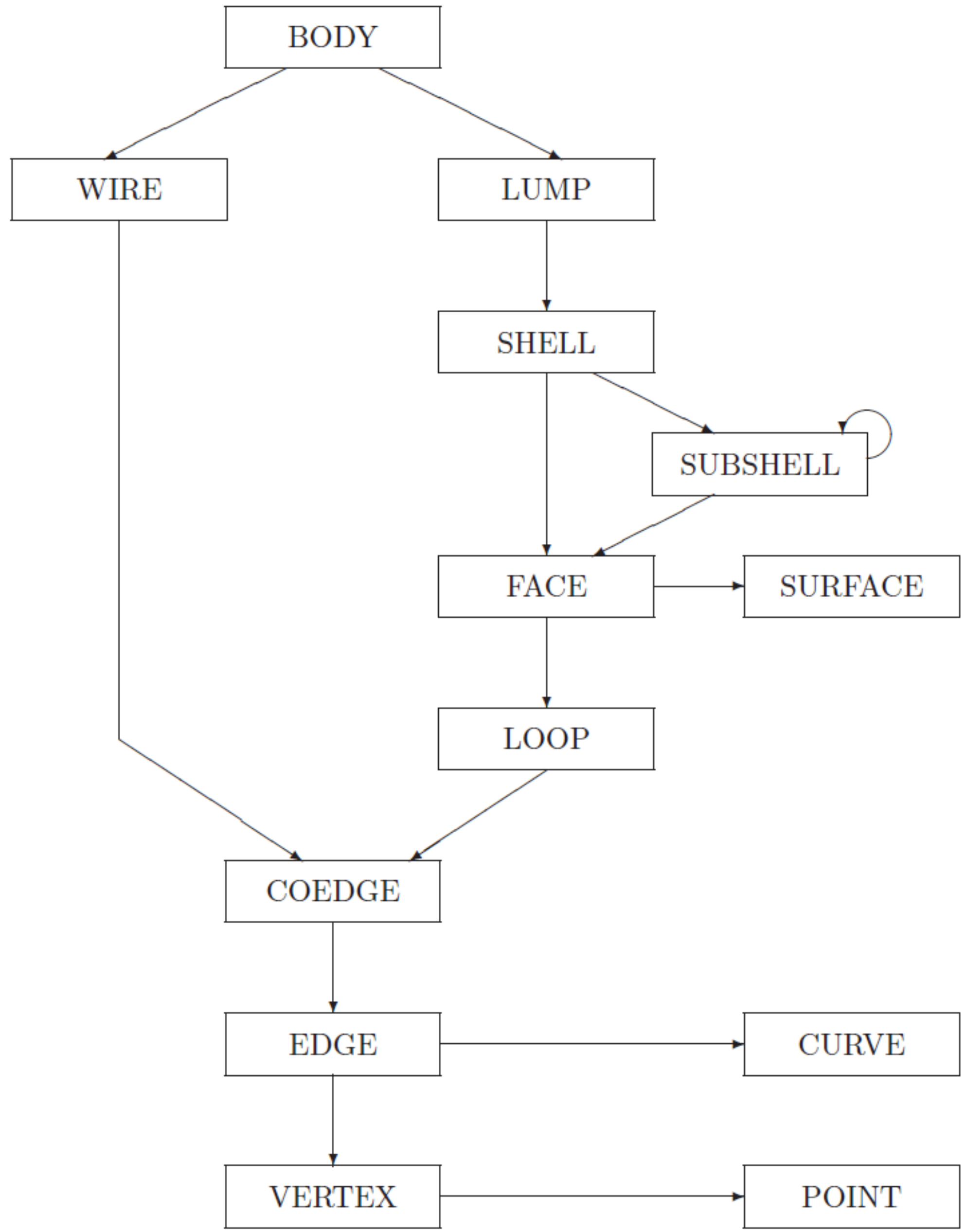
- Developed by Spatial Corp. (part of Dassault) since 1986
- Used in, e.g., AutoCAD (derived kernel), CATIA (up to V5), SolidEdge (before V5)
- Convergence Geometric Modeler (CGM) developed by same company for recent versions of CATIA

Parasolid

- Now owned by Siemens PLM
- Used in, e.g. , SolidEdge (since V5), SolidWorks, Siemens NX

ACIS Data Model

From [1]



ACIS File Formats

- ACIS defines two file formats
  - **.sab** **Binary File format**
  - **.sat** **Text File format**
- Binary file contains identical information as text, but is more compact
- Contain:
  - 3 line header
  - Core data according to defined entity types
  - Optional: update history
- Supported as exchange formats for many CAD tools (even with different kernels)

ACIS .sat File Example

```
700 0 1 0
9 Cobalt v4 16 ACIS 7.0 Unknown 24 Thu Jul 18 22:12:55 2002
25.39999999999999858 9.9999999999999547e-07 1.00000000000000036e-10
body $1 -1 $-1 $2 $-1 $3 #
rgb_color-st-attrib $-1 -1 $4 $-1 $0 0 1 0 #
lump $-1 -1 $-1 $-1 $5 $0 #
transform $-1 -1 1 0 0 0 1 0 0 0 1 0 0 0 0.03937007874015747977 no_rotate
                                                    no_reflect no_shear #
id_attribute-st-attrib $-1 -1 $-1 $1 $0 3 #
shell $-1 -1 $-1 $-1 $-1 $6 $-1 $2 #
face $7 -1 $-1 $8 $9 $5 $-1 $10 reversed single #
…
```

[Source: Paul Bourke
http://paulbourke.net/dataformats/]

Parasolid

- Similar to ACIS, defines file formats
  - .x_t (also .xt) for text data
  - .x_b (also .xb) for binary data
- “~45% CAD data worldwide is Parasolid format” [John Juckes: XT B-Rep; Making it Real]

Parasolid .x_t File Example

```
**ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz**********
*
**PARASOLID !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~0123456789***********
**PART1;MC=osf65;MC_MODEL=alpha;MC_ID=sdlosf6;OS=OSF1;OS_RELEASE=
V4.0;FRU=sdl_parasolid_test_osf64;APPL=unknown;SITE=sdl-cambridgeu.
k.;USER=davidj;FORMAT=text;GUISE=transmit;DATE=29-mar-2000;
**PART2;SCH=SCH_1200000_12006;USFLD_SIZE=0;
**PART3;
**END_OF_HEADER*****************************************************

T51 : TRANSMIT FILE created by modeller version 120000017 SCH_1200000_120060
12 1 12 0 2 0 0 0 0 1e3 1e-8 0 0 0 1 0 3 1 3 4 5 0 6 7 0     body
70 2 0 1 0 0 4 1 20 8 8 8 1 T                                 list
13 3 3 0 1 0 9 0 0 6 9                                        shell
50 4 11 0 9 0 0 0 +0 0 0 0 0 1 1 0 0                          plane
31 5 10 0 7 0 0 0 +0 0 0 0 0 1 1 0 0 1                        circle
19 6 5 0 1 0 0 3 V                                            region
16 7 6 0 ?10 0 0 5 0 0 1                                      edge
. . .
```

[Siemens PLM: Parasolid XT Format Reference]

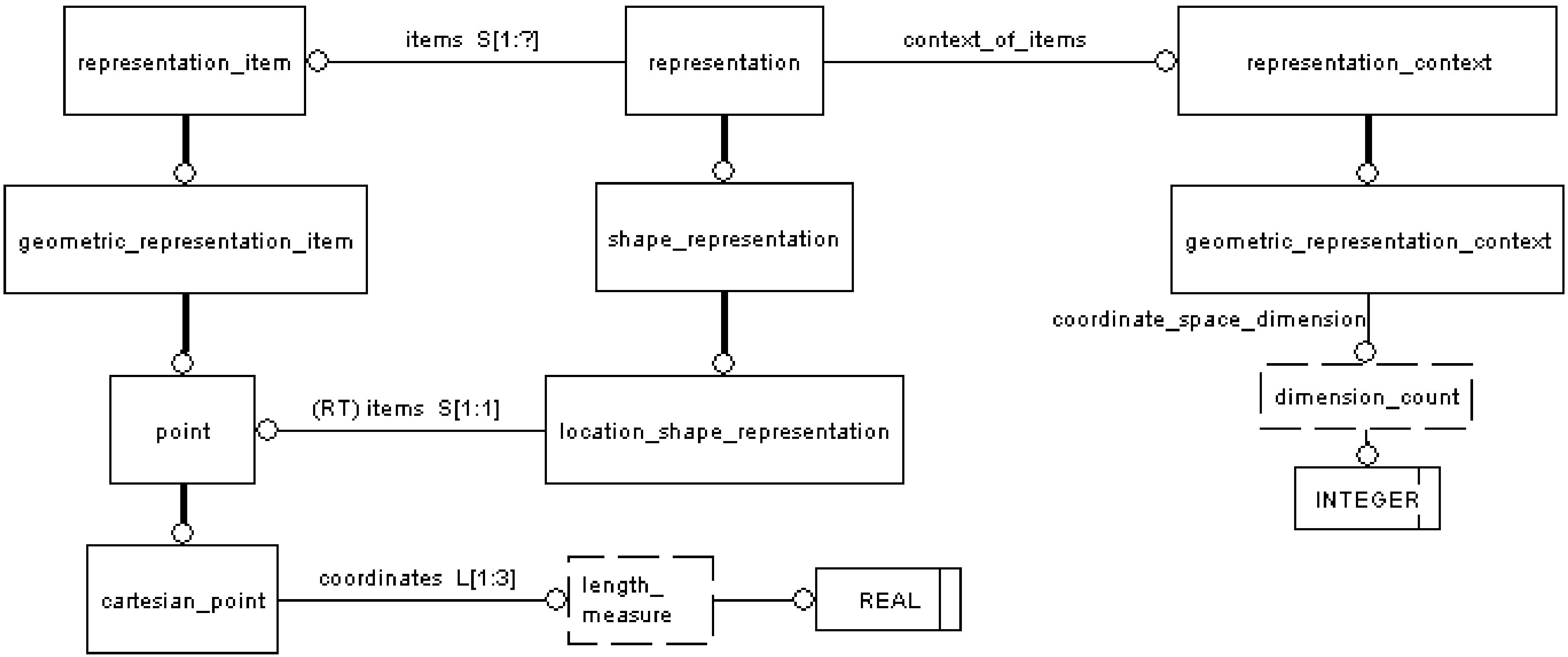
STEP

- **STandard for the Exchange of Product model data**
- Developed since 1984 by international consortium
- Standardized since 1990s as **ISO 10303**
- Contains
  - General methods for describing data and schemas
  - Definitions of generic file formats
  - Application-specific methods for engineering domains

STEP Parts relevant for Geometric Modeling

- Parts most relevant for Geometric Models:
  - 10303-1x — Description Methods, e.g.
    - 10303-11 — EXPRESS and EXPRESS-G
  - 10303-2x — Implementation Methods, e.g.
    - 10303-21 — STEP files
    - 10303-23 — C++ Language Binding
    - 10303-28 — STEP XML
  - Further 10303-XX — Integrated generic resources
    - 10303-42 — Geometric and topological representation
    - 10303-52 — Mesh-based topology
  - 10303-2XX — Application Protocols
    - **10303-203** — **Configuration controlled 3D designs of mechanical parts and assemblies**
    - **10303-214** — **Core data for automotive mechanical design processes (mostly superset of 203)**

AP214 EXPRESS-G Schema (Excerpt)



[Source: wikistep.org]

AP 214 EXPRESS Schema (Excerpt)

```
(* SCHEMA geometry_schema; *)

 ENTITY cartesian_point
   SUPERTYPE OF (ONEOF(cylindrical_point, polar_point, spherical_point))
   SUBTYPE OF (point);
    coordinates  : LIST [1:3] OF length_measure;
 END_ENTITY;
```

[Source: steptools.com]

Example AP214 .TSEP File

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION( ( '' ), ' ' );
FILE_NAME( 'pumpHousing.stp', '2004-04-13T21:07:11', ( 'Tim Olson' ), ( 'CADSoft Solutions
                                        Inc' ), ' ', 'ACIS 12.0', ' ' );
FILE_SCHEMA( ( 'automotive_design' ) );
ENDSEC;
DATA;
. . .
#3716 = POINT_STYLE( ' ', #6060, POSITIVE_LENGTH_MEASURE( 1.00000000000000E-06 ), #6061 );
#3717 = CARTESIAN_POINT( '', ( -1.10591425372267, 3.05319777988191, 0.541338582677165 ) );
#3718 = CURVE_STYLE( '', #6062, POSITIVE_LENGTH_MEASURE( 1.00000000000000E-06 ), #6063 );
#3719 = LINE( '', #6064, #6065 );
#3720 = CURVE_STYLE( '', #6066, POSITIVE_LENGTH_MEASURE( 1.00000000000000E-06 ), #6067 );
#3721 = CIRCLE( '', #6068, 1.75849340964528 );
#3722 = CURVE_STYLE( '', #6069, POSITIVE_LENGTH_MEASURE( 1.00000000000000E-06 ), #6070 );
#3723 = CIRCLE( '', #6071, 0.540114611464642 );
#3724 = SURFACE_STYLE_USAGE( .BOTH., #6072 );
#3725 = FACE_OUTER_BOUND( '', #6073, .T. );
. . .
ENDSEC;
END-ISO-10303-21;
```


[Source: Paul Bourke
http://paulbourke.net/dataformats/]

IGES

- Initial Graphics Exchange Specification
- Created in the early 1980s by National Bureau of Standards (American government organization for standardization)
- Supported by many CAD tools as exchange format
- Numeric encoding of entity types inspired by “punch cards” → despite text format, hardly human readable

IGES Example

```
IGES Start section created by Claris Graphics Translator v1.0          S      1
1H,,1H;,12Hbox+line.IGS,12Hbox+line.IGS,8HCGT v1.0,8HIGES 3.0,16,8,24,8,G      1
24,12Hbox+line.IGS,1.0,1,4HINCH,32000,1.6,13H910305.183439,,,3HIBB,3HWCPG      2
,4,0;                                                                  G      3
     124       1       1       0       0       0       0       0 0 0 0 0D      1
     124       0       0       1       0       0       0               D      2
     124       2       1       0       0       0       0       0 0 1 1 0D      3
     124       0       0       1       0       0       0               D      4
     410       3       1       1       0       0       3       0 0 2 2 0D      5
     410       0       0       1       0       0       0               D      6
     124       4       1       0       0       0       0       0 0 0 0 0D      7
     124       0       0       1       0       0       0               D      8
     124       5       1       0       0       0       0       0 0 1 1 0D      9
…
```

[Source: Paul Bourke
http://paulbourke.net/dataformats/]

Further Important Formats

- DXF – Textual AutoCAD exchange format
- DWG – Binary AutoCAD format
- VRML (Virtual Reality Markup Language)
- X3D – XML-based follow-up to VRML
- 3DXML – Exchange Format of CATIA (set of zipped XML files)
- STereoLithography (STL)
- Collada – XML-based language for Automation including geometrical data
- JT – Jupiter Tesselation (Siemens PLM – possible follow up to XT)
- . . .
- . . .
- . . .
- Many document (e.g. PDF) and pixel-based graphics formats (TIFF, PNG, etc.) supported as export formats

X3D Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN"
  "http://www.web3d.org/specifications/x3d-3.2.dtd">

<X3D profile="Interchange" version="3.2"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
     xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-
                                            3.2.xsd">
<Scene>
  <Shape>
    <IndexedFaceSet coordIndex="0 1 2">
      <Coordinate point="0 0 0 1 0 0 0.5 1 0"/>
    </IndexedFaceSet>
  </Shape>
</Scene>
</X3D>
```

3DXML Example (snippet)

```
. . .
<Faces>
   <Face strips="4 2 14 22,48 23 50 26,49 25 39 20,7 19 3 1,21 0 24
                          18,29 35 32 38,27 40 34 45,33 44 36 46,37 47 30
                          42,31 43 28 41,15 11 5 9 8 10 17 12 15 11">

         <SurfaceAttributes>
                 <Color xsi:type="RGBAColorType" red="1" green="1" blue="1"
                                  alpha="1"/>
         </SurfaceAttributes>
   </Face>
</Faces>
<Edges>
   <LineAttributes lineType="SOLID" thickness="2">
         <Color xsi:type="RGBAColorType" red="0" green="0" blue="0"
                 alpha="1"/>
   </LineAttributes>
   <Polyline vertices="0 0 0,360 0 0"/>
   <Polyline vertices="0 500 0,360 500 0"/>
   . . .
</Edges>
. . .
```

STL Example

```
solid
   :
   :
   facet normal 0.0 0.0 1.0
   outer loop
      vertex  1.0  1.0  0.0
      vertex -1.0  1.0  0.0
      vertex  0.0 -1.0  0.0
   endloop
   endfacet
   :
   :
endsolid
```

[Source: Paul Bourke
http://paulbourke.net/dataformats/]

Geometry Data in Databases

- Motivation and open Problems
- CAD data and Relational Databases (RDBMS)
  - CAD metadata
  - CAD data as Binary Large Objects (BLOBs)
  - CAD data in Database File Systems
  - CAD data as structured data
- CAD data and advanced Database Concepts
  - Object-relational Database (ORDBMS)
  - Object-oriented Databases (ODBMS)
  - XML Databases
  - Cloud Databases
  - Specialized DBMS
  - Spatial data and Geographic Information Systems
- STEP SDAI

Motivation

- General idea: use advantages of DBMS
  - Efficient access to huge data volumes
  - Multi-user support
  - Controlled consistency
- RDBMS are
  - Not often used for geometry data
  - Commonly used for PDM/PLM and other engineering applications
- ORDBMS and ODBMS theoretically provide suitable features, but have other disadvantages
- Geometry still most often stored in files due to the following problems →

Open Problems /1

- **Data too complex** for RDBMS and ORDBMS
  - Many primitives, operations, etc. require many database types (or tables)
  - Topological relationships require following connections between many entities (objects, tuples) for even simple geometries
  - Following references implemented in terms of expensive JOIN (combining data from different tables) and SELF-JOIN (combining data within tables) operations → bad performance
  - ORDBMS allow richer modeling constructs and use of references, but data is most often still stored in a fragmented way → bad performance
- **Lack of standardization** for ORDBMS and ODBMS
  - SQL:1999 and SQL:2003 describe object-relational standard, but are implemented by existing DBMS in varying ways (unlike the relational core SQL-92)
  - ODMG standard for object-oriented DBMS hardly implemented at all
  - Limited portability and reusability of developed solutions

Open Problems /2

- **Lack of acceptance** for ORDBMS and ODBMS
  - Object-relational concepts supported by many existing DBMS, but not always used (due to problems with portability, technological complexity, etc.)
  - ODBMS niche market with small companies: unstable product support
- **Data exchange problematic**
  - Exchange requires neutral physical representation
  - If DBMS is used, data has to be exported to exchange format anyway
  - Requires application-specific export filters database-to-file
- **Archiving CAD data problematic**
  - Archived data needs to be interpretable even after decades
  - DBMS of limited use for long-time archiving due to proprietary physical data structures
- ...

Storing CAD Data using RDBMS

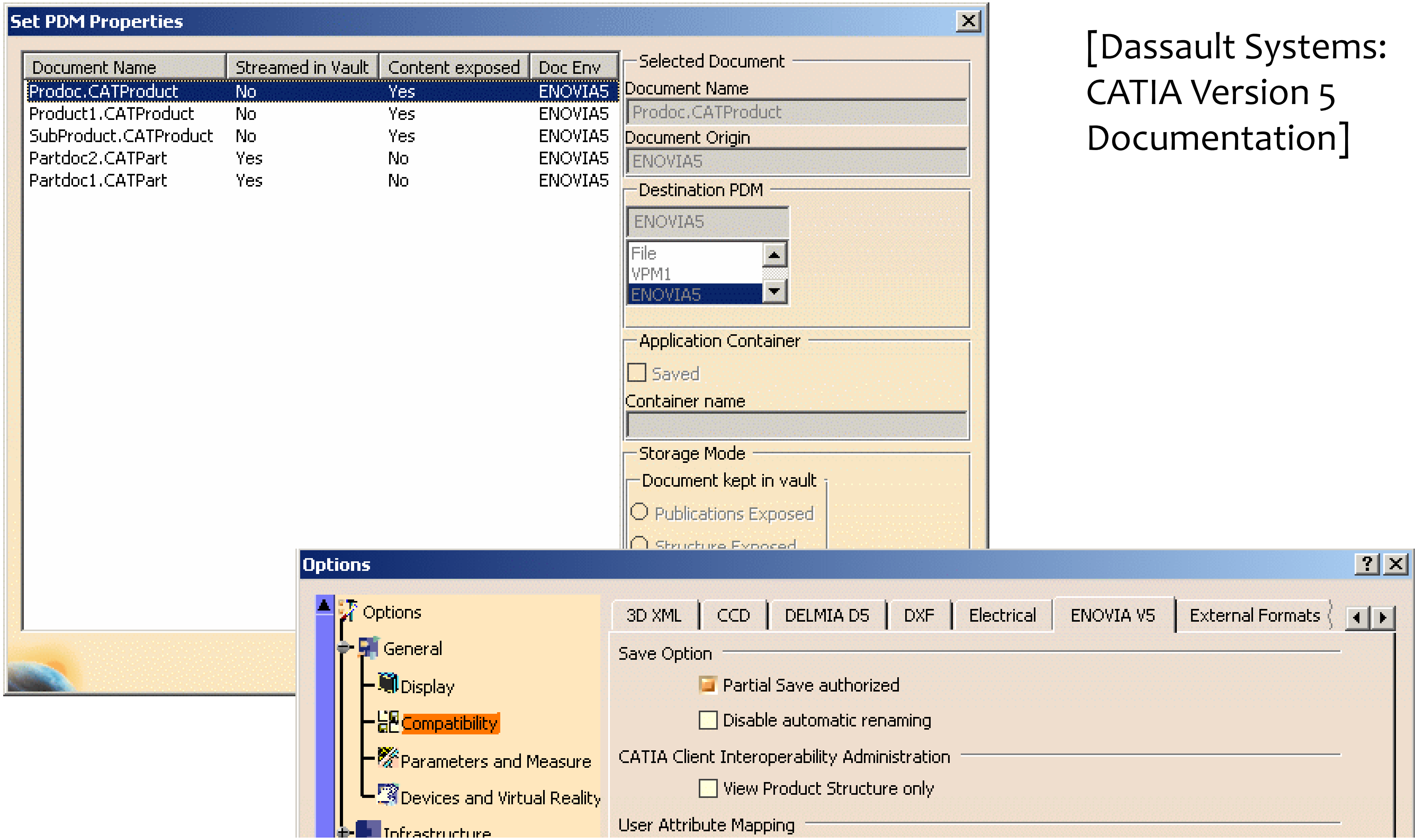
- 4 possible alternatives

  1. Store **only metadata** in DBMS
  2. Store CAD data in dedicated format as binary large objects (**BLOB**)
  3. Store CAD data (as BLOB) in database file system
  4. Store CAD data as **structured data** in RDBMS

CAD Metadata in RDBMS

- Data about CAD data typically has simpler structures
  - About design process: creator, dates, status, versions, etc.
  - Relation of partial design (part) to overall product within product structure
  - References to other engineering data
- Suitable for storage in RDBMS
- Typically task of Product Lifecycle Management (PLM) system
- CAD systems often
  - tightly integrated with PLM or
  - offer own PLM-functionality (e.g. CATIA un V5 and V6 integrates ENOVIA)

Example: Metadata in CATIA V5



[Dassault Systems: CATIA Version 5 Documentation]

CAD Data as Binary Large Objects (BLOB)

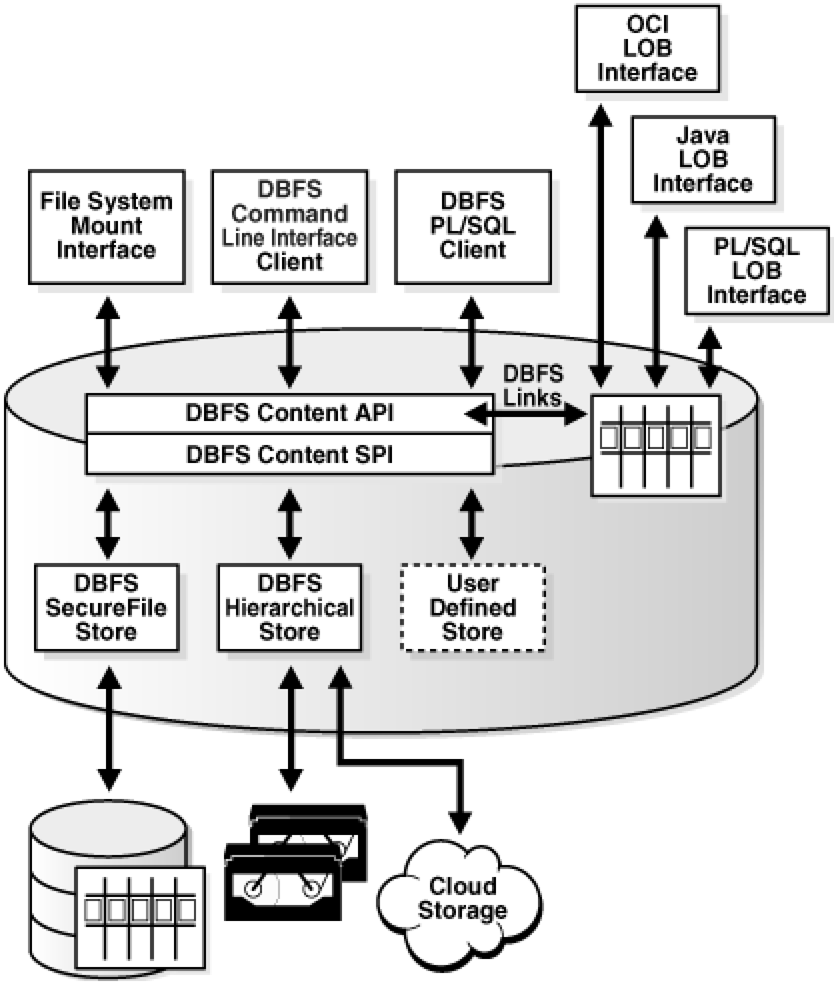
- Since SQL:1999 DBMS support arbitrary binary data
- Often used to store files in tables of database (e.g. from PLM system)
- Semantics of BLOB
  - unknown to DBMS → no functionality except for reading and writing as one value
  - Typically conforms to proprietary or standard format of used CAD system

```
CREATE TABLE construction_part (
      part_id        INT PRIMARY KEY,
      name           VARCHAR(100),
      responsible    INT FOREIGN KEY
                     REFERENCES engineer(eng_id),
      dxf_file       BLOB,
      . . .
);
```

Database File Systems

- DBMS provide storage facilities based on BLOBs that externally can be used as any file system
  - BLOB storage can be mounted as virtual file system
  - (CAD) files stored in this file system are physically stored in and controlled by the DBMS
  - Allows access via file or database interfaces
- Advantages from CAD perspective
  - Flatly structured metadata can be easily linked with complex CAD data
  - Accesses and consistency to some degree controlled by DBMS mechanisms
  - Transparent integration with file based activities
  - Advanced recovery mechanisms of DBMS can be used
  - Similar functionality as network/distributed file systems

Example: Oracle SecureFiles



[Oracle® Database SecureFiles and Large Objects Developer's Guide 11g Release 2]

CAD Data as Relational Data

- Theoretically possible to create tables from types defined in
  - Modeling kernels
  - STEP standard
- Implemented in several research prototypes and few commercial systems
- No common practice due to the disadvantages mentioned before
  - Poor performance due to complex data
  - Problems with archiving and exchange

CAD Data as Relational Data

Mech_Part

ID	FACES
cuboid	f1
cuboid	f2
pyramid	f101

FACES

ID	EDGES
f1	e1
f1	e2
...	...

EDGES

ID	VERTICES
e1	v1
e1	v2
e2	v1

VERTICES

ID	X	Y	Z
v1	0	0	0
v2	...	...	...
v3	...	...	...

```
select Mech_Part.ID,X,Y,Z
from Mech_Part,FACES,EDGES,VERTICES
where Mech_Part.FACES = FACES.ID
  and FACES.EDGES = EDGES.ID
  and EDGES.VERTICES = VERTICES.ID
  and Mech_Part.ID = "cuboid"
```

From [5]

CAD Data with ORDBMS

- Rich semantic modeling of potential benefit
  - Type system and specialization (inheritance) allows 1:1 implementation of CAD schemas (e.g. modeling kernel or STEP AP214) in SQL
  - References and nested tables (NF^2 = Non-First-Normal-Form) allow creation of complex types/objects
- No common practice due to previously mentioned disadvantages, mainly
  - Still poor performance because of fragmented storage
  - Lack of acceptance

Example: Winged Edge in SQL:2003 (excerpt)

```
. . .
CREATE TYPE we_vertex_type UNDER geometry_type (
       edges             REF(we_edge_type) MULTISET,
       coordinates       FLOAT ARRAY(3),
);

CREATE TYPE we_edge_type UNDER geometry_type (
       vertex1           REF(we_vertex_type),
       vertex2           REF(we_vertex_type),
       aface             REF(we_face_type),
       bface             REF(we_face_type),
       neighbours        REF(we_edge_type) ARRAY(4),
);

CREATE TABLE edge OF we_edge_type;
CREATE TABLE vertex OF we_vertex_type;
. . .
```

Example: NF²

```
create Mechanical_Part {
    [ ID: integer,
      NAME: string(20),
      FACES: {
        [ ID: string(4),
          EDGES: {
            [ ID: string(4),
              VERTICES: {
                [ ID: string(4),
                  LOCATION:
                    [ X: ...,
                      Y: ...,
                      Z: ...  ]
                ] }
            ] }
        ] }
    ] }
end
```

Mechanical_Part							
ID	NAME	FACES					
		ID	EDGES				
			ID	VERTICES			
				ID	LOCATION		
					X	Y	Z
...	...	...	...	...	...	...	...
5	bracket	f1	e1	v1	1	0	2
				v2	0	1	0
			e2	v3	1	2	3
				v1	1	0	2
			e3	v3	1	2	3
				v4	1	0	5
			e4	v2	0	1	0
				v4	1	0	5
		f2	e5	v5	...	...	...
				v6	...	...	...
			e6	...	...	...	...
			...	...	...	...	...
		...	...	...	...	...	...
...	...	...	...	...	...	...	...

From [5]

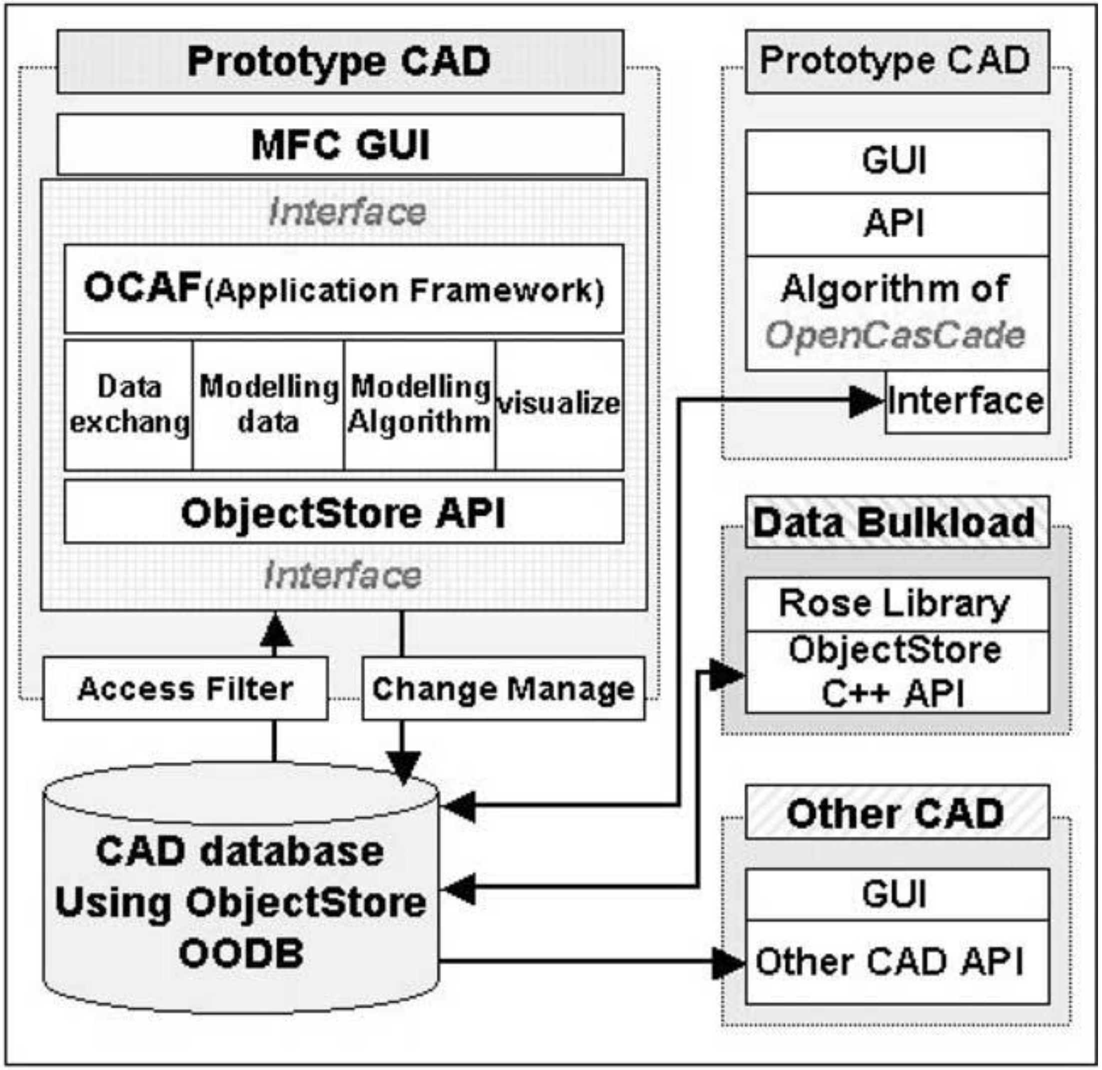
CAD Data with ODBMS

- Allow implementation of CAD schemas according to data models of C++, Java, C#, etc. (see STEP SDAI ISO 10303-22 →)
- Objects of classes can be persisted with special mechanisms
  - Schema consisting of persistence-capable classes can be created from source code (using pre- or post-processor tools)
  - Named objects (lookup of single object via unique name)
  - Object networks with entry points (named root objects)
    - Persistence by reachability: store objects along with objects “reachable” via references (transitive closure)
  - Collections (sets, lists, multi-sets) of objects
    - May provide query interface
- Provide good performance and easy development → some CAD systems (mainly in the 1990s) used ODBMS
- Today no common practice due to mentioned disadvantages, mainly lack of acceptance and standardization

ODBMS Example: ODMG Java Binding

```
…

class WE_Edge {
        WE_Vertex vert1, vert2;
        WE_Face aFace, bFace;
        WE_Edge aPrev, aNext, bPrev, bNext;
        WE_EdgeDataObject data;

        …

        public static void main(String args[]) {
                …
                Database db = odmg.newDatabase();
                …
                WE_Edge e1 = new WE_Edge(…);
                db. Bind(e1,"myEdge1");
                …
        }
}
```

CAD and ODBMS: Architecture Example



[Kim, Han: Encapsulation of geometric functions for ship structural CAD using a STEP database as native storage. Computer-Aided Design, 2003]

Special Functionality in ODBMS

- Because ODBMS rather popular in engineering some systems implemented specific functionality, e.g.
  - **Long/design transactions**: check out/check in mechanism
  - **Workspaces**: store data of one user or group in separate location during long transactions
  - **Support for versioning and variants** on data model level: create sequential (versions) and parallel (variants) manifestations of one object
  - **Nested transactions**: allow transaction within transaction to support complex design process
  - **Database file systems**: as in some RDBMS (↑)
  - …

Further DBMS Types

XML DBMS

- Allow storage of XML data (documents, document collections), i.e. useful complementary to XML CAD file formats
- Allow querying via specialized
- Similar problems regarding performance as ORDBMS

Cloud DBMS

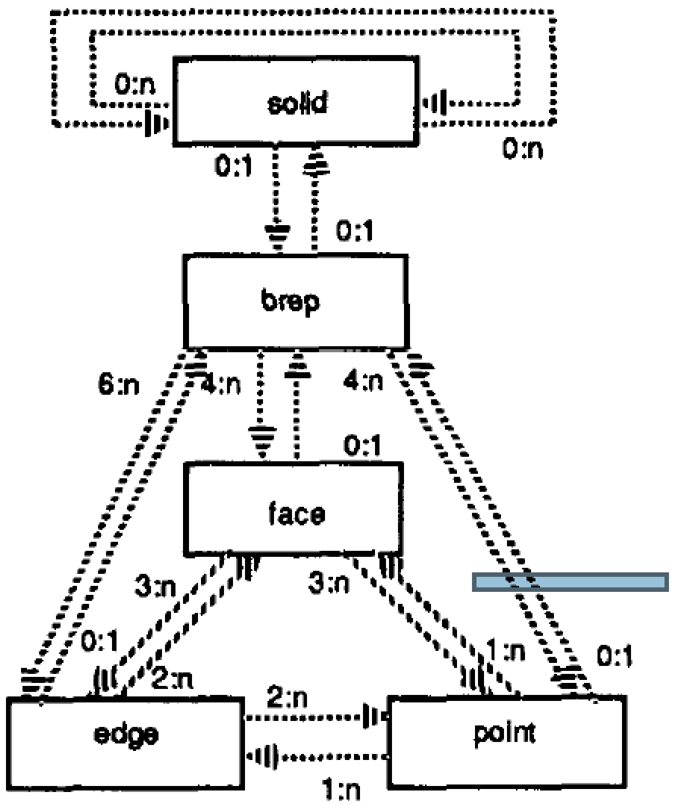
- So-called no-SQL systems for simple storage of weakly structured data of possibly huge amounts (keyword Big Data) on the Web/in the Cloud
- Requirements and usefulness for engineering applications is topic of current research

Specialized DBMS for CAD Data

- No existing system easily fulfills all requirements
- Several attempts in industrial and academic research to develop tailor-made DBMS
- Often based on concepts of object-oriented DBMS

Specialized DBMS Example: PRIMA

a) MAD schema diagram



b) atom type definitions

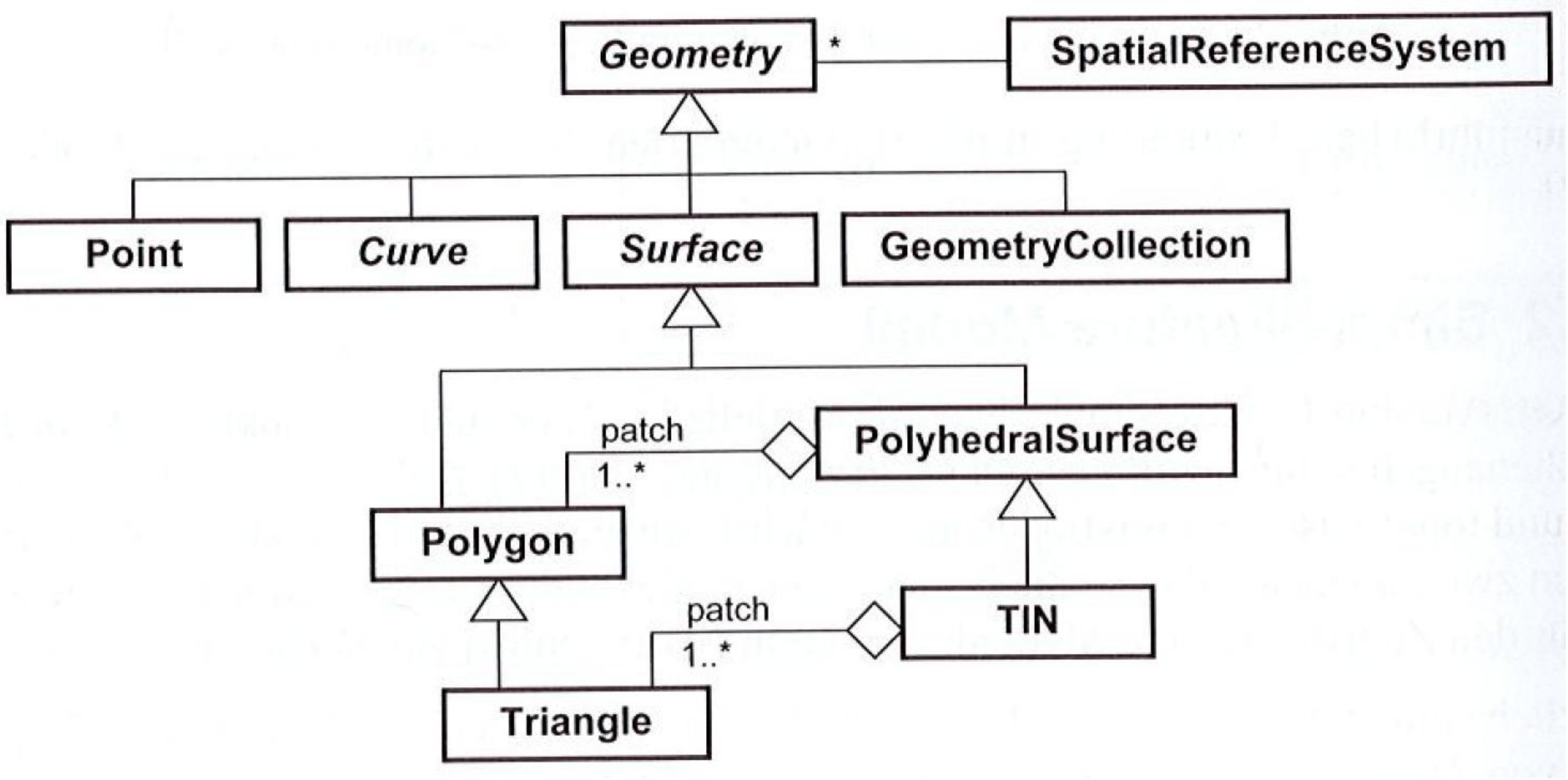
```
CREATE ATOM_TYPE solid
    ( solid_id     : IDENTIFIER,
      solid_no     : INTEGER,
      description  : CHAR_VAR,
      sub          : SET_OF (REF_TO (solid.super)),
      super        : SET_OF (REF_TO (solid.sub)) ,
      brep         : REF_TO (brep.solid)   )
      KEYS_ARE (solid_no)

CREATE ATOM_TYPE brep
    ( brep_id      : IDENTIFIER,
      brep_no      : INTEGER,
      hull         : HULL_DIM(3),
      solid        : REF_TO (solid.brep),
      faces        : SET_OF (REF_TO (face.brep)) (4,VAR),
      edges        : SET_OF (REF_TO (edge.brep)) (6,VAR),
      points       : SET_OF (REF_TO (point.brep)) (4,VAR)   )
      KEYS_ARE (brep_no)

CREATE ATOM_TYPE face
    ( face_id      : IDENTIFIER,
      square_dim   : REAL,
      border       : SET_OF (REF_TO (edge.face)) (3,VAR),
      crosspoint   : SET_OF (REF_TO (point.face)) (3,VAR),
      brep         : REF_TO (brep.faces)   )
```

[Härder, Meyer-Wegener et al. PRIMA - a DBMS Prototype Supporting Engineering Applications. VLDB 1987]

Spatial Data

- Related: geometrical data in Geographic Information Systems (GIS) is strongly supported by DBMS
  - Less complex than CAD data (few primitives, less flexible topologies)
- Standardization from GIS and DBMS community
  - Open Geospatial Consortium (OGC), e.g. Simple Feature Model for 2D
  - SQL/MM Spatial
- Numerous implementations, e.g. Oracle Spatial

Simple Feature Model (OGC)



[Brinkhoff: Geodatenbanksysteme in Theorie und Praxis. Wichmann, 2013]

Example: Oracle Spatial

```
CREATE TABLE GeoDbLand3D (
  id      INTEGER,
  name    VARCHAR(20),
  geo     SDO_GEOMETRY,
  CONSTRAINT pk_gdbland3D PRIMARY KEY(id)
);

-- Dach als 3D-Fläche einfügen:
INSERT INTO GeoDbLand3D (id,name,geo)
VALUES (51, 'Hausdach', SDO_GEOMETRY(3003, NULL, NULL,
   SDO_ELEM_INFO_ARRAY(1,1006,1, 1,1003,1, 16,2003,1, 31,1003,1),
   SDO_ORDINATE_ARRAY(
      9,9,6.5, 9,9.5,7, 12,9.5,7, 12,9,6.5, 9,9,6.5,
      10,9.15,6.65, 11,9.15,6.65, 11,9.35,6.85, 10,9.35,6.85, 10,9.15,6.65,
      9,10,6.5, 12,10,6.5, 12,9.5,7, 9,9.5,7, 9,10,6.5) ) );
-- Hauskörper als Quader mittels Eckpunktbeschreibung einfügen:
INSERT INTO GeoDbLand3D (id,name,geo)
VALUES (52, 'Hauskörper', SDO_GEOMETRY(3008,NULL,NULL,
    SDO_ELEM_INFO_ARRAY(1,1007,3), SDO_ORDINATE_ARRAY(9,9,2, 12,10,6.5) ) );
COMMIT;
```

[Brinkhoff: Geodatenbanksysteme in Theorie und Praxis. Wichmann, 2013]

STEP SDAI

- **Standard Data Access Interface ISO 10303-22** defines standard bindings to languages (C, C++, Java) for STEP data access
- Similar to an API for an RDBMS (ODBC, JDBC) or ODBMS defines basic functionality such as
  - Sessions
  - Database connectivity
  - Data dictionary
- Defines mappings of EXPRESS types to language constructs, e.
- Not specific to geometrical data → used more often for other applications

Literature / Further Readings

[1] Ali K. Kamrani, Emad Abouel Nasr: Engineering Design and Rapid Prototyping. Springer, 2010.
ISBN 978-0-387-95862-0

[2] Ian Stroud: Boundary Representation Modelling Techniques. Springer 2006.
ISBN-10: 0-387-84628-312-4

[3] M. M. M. Sarcar, K. Mallikarjuna Rao,K. Lalit Narayan : Computer Aided Design and Manufacturing
ISBN 978-8-120-33342-0

[4] ACIS Documentation
http://doc.spatial.com/index.php/Main_Page

[5] A. Kemper, M. Wallrath: An Analysis of Geometric Modeling in Database Systems. ACM Comput. Surv. 19(1): 47-91 (1987)