Introduction

- Deficiencies exist in use of conventional modeling systems for engineering design.
- Prior to 1988, systems produced only “dumb” models (no imbedded information beyond shape definition)
- Systems permitted only a single level of design abstraction (one specific geometry)

Introduction

- Early model descriptions
  - Required strictly dimensioned entities with complete dimensional specification
  - Models explicitly described shape.
  - User must ensure that model represents design intent because no intent is “embedded” within the model definition.

Introduction

- This information frequently not known during design phase
- Modeler only then functions as a documentation tool with the model created at the END of the design sequence.

Introduction

- Explicit geometric modeling systems deal with geometry on a low definition level:
  - B-Rep vertices, edges, faces
  - CSG primitives, Booleans
- Designers frequently require higher level control
  - For example: maintain constant wall thickness, provide sufficient mat’l around holes

Introduction

- Explicit geometric modelers do not capture this design intent
- Need was noted for higher level modeler interface which would allow the system to function as a true design tool.

Feature modeling interface

- higher level interface would permit specification of entities in terms of geometric and dimensional constraints
- permit definition of features directly by user, with appropriate geometric shape created within solid modeling kernel
  - solid modeling kernel: a toolkit of modeling subroutines called by software. A kernel is occasionally referred to as an “engine”
**Relationship of Feature-Based Interface to Geometric Modeler**

User Input → Feature-Based Interface → Solid Modeling kernel: (B-Rep or CSG)

**Modeling Systems**

<table>
<thead>
<tr>
<th>CAD Package</th>
<th>Manufacturer</th>
<th>Primary Format</th>
<th>Modeling Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mech. Desktop/ Inventor</td>
<td>AutoDesk</td>
<td>B-Rep</td>
<td>ACIS</td>
</tr>
<tr>
<td>MicroStation Modeler</td>
<td>Bentley</td>
<td>B-Rep</td>
<td>Parasolid</td>
</tr>
<tr>
<td>Pro/ENGINEER</td>
<td>Parametric Technologies</td>
<td>B-Rep</td>
<td>proprietary</td>
</tr>
<tr>
<td>SolidWorks</td>
<td>Dessault</td>
<td>B-Rep</td>
<td>Parasolid</td>
</tr>
<tr>
<td>IronCad</td>
<td>Ironcad</td>
<td>B-Rep</td>
<td>Parasolid/ ACIS</td>
</tr>
<tr>
<td>Unigraphics</td>
<td>EDS</td>
<td>B-Rep</td>
<td>Parasolid</td>
</tr>
<tr>
<td>Solidedge</td>
<td>EDS</td>
<td>B-Rep</td>
<td>Parasolid</td>
</tr>
</tbody>
</table>

**Feature-based modeling:**

- Geometry is defined in terms of real world “features” as opposed to abstract geometric entities.
- For example:
  - work with holes as opposed to cylinders
  - cuts and extrusions rather than blocks and wedges.

**Feature-based modeling:**

- Parts are represented in terms of higher level entities (features) that have engineering meaning.
- Models contain not just basic geometric and topological data (as all solid models will), but also higher level information.

**Feature-based modeling:**

- Features capture “design intent”
- Included information defines how the features behaves in editing
  - for example, a “through” hole
- Features store non-graphic information for use in:
  - 2d drawing creation, FEA, CNC and kinematic analysis

**Features (types)**

- Features may be divided into 3 general categories:
  - shape features
  - hard-coded features
  - reference features
Shape features

• Created through:
  – sweeping and blending (lofting) of 2D profiles
• features used to add material
  (protrusions) or remove (depressions, cuts, etc.)
• Profiles may defined as constrained sketch geometry (constraint-based systems)

Classifying Shape Features

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passages</td>
<td>subtracted volumes that intersect the pre-existing shape at both ends</td>
</tr>
<tr>
<td>Depressions</td>
<td>subtracted volumes that intersect the pre-existing shape at one end</td>
</tr>
<tr>
<td>Voids</td>
<td>subtracted volumes completely enclosed by material</td>
</tr>
<tr>
<td>Protrusions</td>
<td>added volumes that intersect the pre-existing shape at one end</td>
</tr>
<tr>
<td>Connector</td>
<td>added volumes that intersect the pre-existing shape at both ends</td>
</tr>
</tbody>
</table>

Hard-coded features

• have a characteristic topology
• have a generic geometry
• user input for specific properties such as
  – diameter, depth, entity reference (mounting face)
• examples
  – holes, slots, fillets, rounds

Reference Features

• Not geometric features
• Used for orientation and location of features within model
• Examples are:
  – datum planes
  – datum axes
• Also include defined 2D “sketch planes”
Datum plane (DTM1, etc.) and Datum axis (A-2, etc.) examples

Feature-based modeling:
- First feature created is the “base feature”.
- The base feature:
  - provides the initial reference for all subsequent features, regardless of type
  - may be reference feature(s)
  - may be a stand-alone shape feature

Relation to traditional design:
- During the design process of a component, more attention is given to form and topology than dimensional precision in the conceptual phase.
- In a feature-based which supports constraints, profiles are sketched to capture this “design intent”.

Constraint-based systems:
- Concept first developed in 1970’s
- First commercial system in 1988
- Most current professional level feature-based modelers use constraint-based techniques to help define features (through the creation of profiles)

Constraint-based systems:
- 2D profile geometry not explicitly defined, but rather driven by the constraints.
- Constraints stored internally as equations.
- Equations are solved to resolve the sketch geometry.

Constrained Sketches
- Sketches are 2D.
- They exist within an XY plane frequently referred to as the “sketch plane”.
- Within the sketcher, point locations are not constrained.
- Rather they have a temporary value.
Constrained Sketches

- Similarly, relationships between entities are not fixed.
  - (despite display, two lines may not be parallel, perpendicular, etc.)
- Point locations represent degrees of freedom of the sketch.
- We may think of them as the “unknowns” which have to be found to “solve” the sketch.

Constrained Sketches

- Each point in sketch has 2 dof
- Line segment has 4 dof
- Simple closed piecewise linear loop of “n” vertices has 2n dof
- Loop of “n” vertices has “n” segments
  - Why doesn’t loop have 4n dof?
  - Endpoint continuity means coincidence which removes 1 dof per vertex

Constrained Sketches

- The sketch shown would then have unknowns of an x and a y coordinate for each of 4 points.
- Therefore there exist 8 d.o.f (degrees of freedom) or 8 unknown values.

Constraints are applied to the sketch to “solve” for these unknowns.

- Constraints may be dimensional or geometric. Both are applied as equations.

Dimensional constraints

- Input in form of a dimension
  - e.g. linear, angular, diametric, radial
- Applied as an equation
  - For example, linear dimension between points P1 and P2:
    \[ L = (x_1 - x_2)^2 + (y_1 - y_2)^2 \]
- Dimension drives the geometry
  - e.g. vary value for L, position of endpoint changes

Geometric Constraints

- Input as a relationship between entities or applied to an entity
  - e.g. a line is parallel to another, an arc is tangent to a line
- Applied as an equation
  - For example for a horizontal line AB
    \[ Y_A = Y_B \]
- Some geometric constraints may be inferred and created by the system during entity creation.
Geometric Constraints

- Examples of geometric constraints:
  - parallelism, co-linearity, perpendicularity, equal length/angle/diameter/radius, horizontal, vertical
  - concentric arcs/circles, horizontal, vertical, parallel, tangent

Constrained sketch

- One point must be invariant (within the sketch)
  - e.g. located with respect to position on solid

Constrained sketch

- Setting invariant point may be accomplished (for example)
  - by dimensionally constraining a location to the solid,
  - by geometrically “fixing” a location
  - by geometrically forcing a line on the sketch

Constrained sketch

- Adding constraints provides information to “solve” for point positions and hence shape
- For example, geometric constraints may be applied to force lines Horizontal and Vertical

Constrained sketch

- Dimensional constraints may be applied to control feature size.

Solving constraint equations

- Set of constraint equations solved to define a solution profile.
  - Equations may be solved sequentially
    - referred to as “parametric system”
    - results of each previous solution used in next
  - Equations may be solved simultaneously
    - referred to as “variational system”
- In practice, many systems use a combination of both techniques
Dependent Constraints

• Dimensional constraints may be defined by expressions rather than values.
• In the following image, the horizontal value for the “step” is defined in terms of the overall component “length”.
• When independent constraint values are changed, the dependent constraints change.

Constraint Associativity Diagram

Feature A:

- Width
- Length
- Y_loc
- X_loc
- Contact_edge

How are features constrained?

• Shape features
  – by attachment face/plane of sketch
  – by locating invariant point of sketch
• Hard coded
  – by attachment face/plane
  – by locating feature reference (center, edge, etc.) from solid reference
• Reference features
  – by relative geometry (edges, axes, faces)
Parent/Child relationship

• dependency exists between created feature (child) and parent: e.g.
  – other features used to locate it
  – insertion features
    • mounting feature: surface or sketch plane
  – source features
    • patterns and part families
• dependency means process of editing parent feature requires considering child features of that parent

Parent/Child Relations

• Positioning defines parent/child relations
• Hard coded
  – Child of:
    • attachment face,
    • locating references
    • and feature termination constraints.

Hard-coded feature

Parent/Child Relations

• Shape feature is child of
  – sketch
  – feature termination constraints.
  • Blind distance, up to face, etc.
• Sketch is child of attachment face and locating references

Shape Feature
Feature Dependency Diagram

- Use of Feature Dependency Diagrams can help in control of parent/child relationships.
- Following image:
  - Feature C is a child of both Feature A and Feature B

Top Down Model Construction:

- Model features created in a downward cascading sequence
- Changes in a feature can effect all features later in sequence

Horizontal Structure

- Create Reference features as Base Features (Feature 1 in following figures)
- Relate created model features to the earliest possible features (preferably Base Features)
- Base features will not change, this minimizes effect of editing changes
Feature Management

• Supports feature editing processes
• Typical processes:
  – feature suppression (from display and editing)
  – control of feature construction order
  – control of feature references (parent/child)

Model creation order

• order dependent process
• parent/child relations based upon order of creation
• only able to reference previously defined features
• able to “re-order” model in feature manager