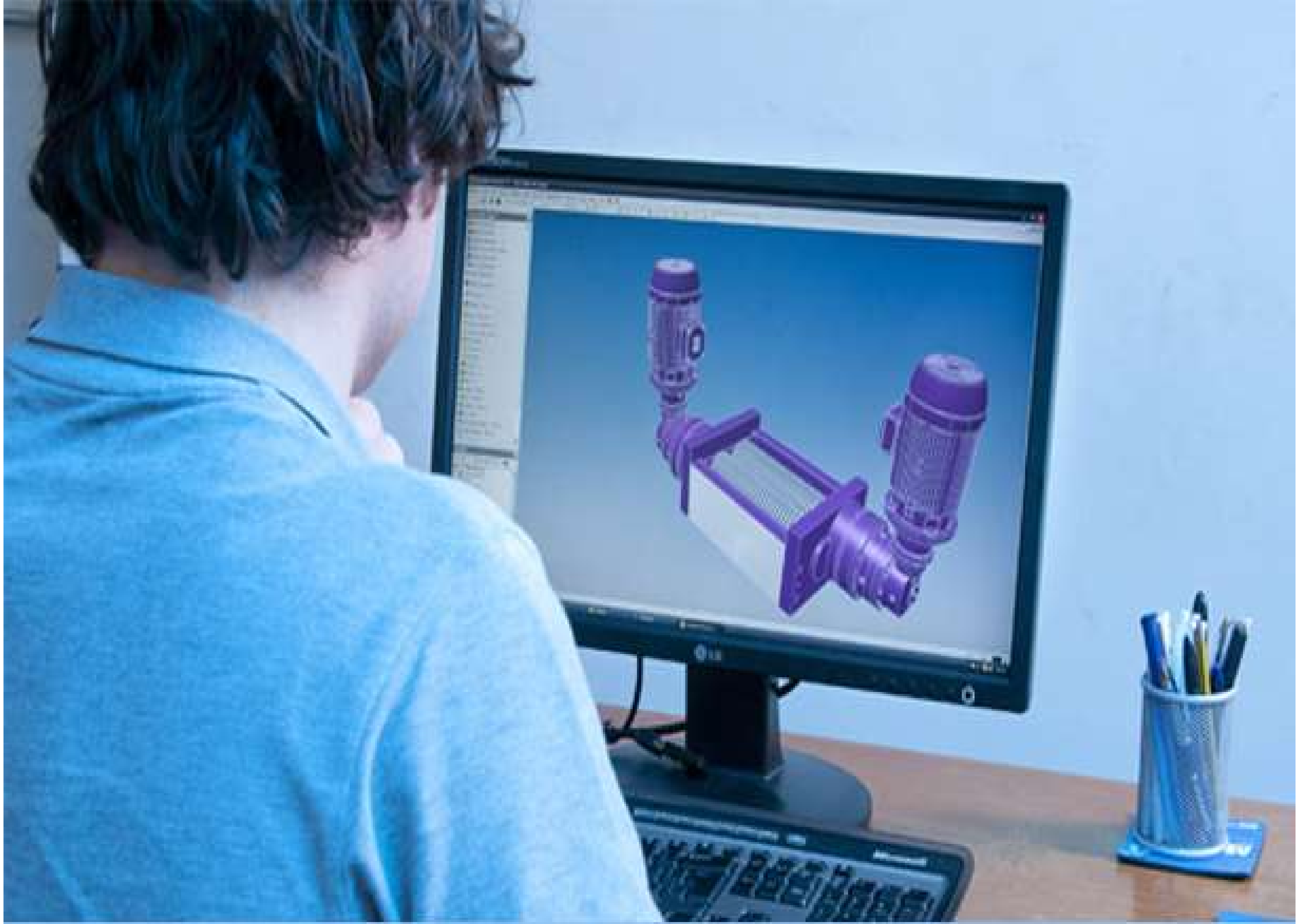


Computer-aided Design

Bryana Hudspeth

First Edition, 2012

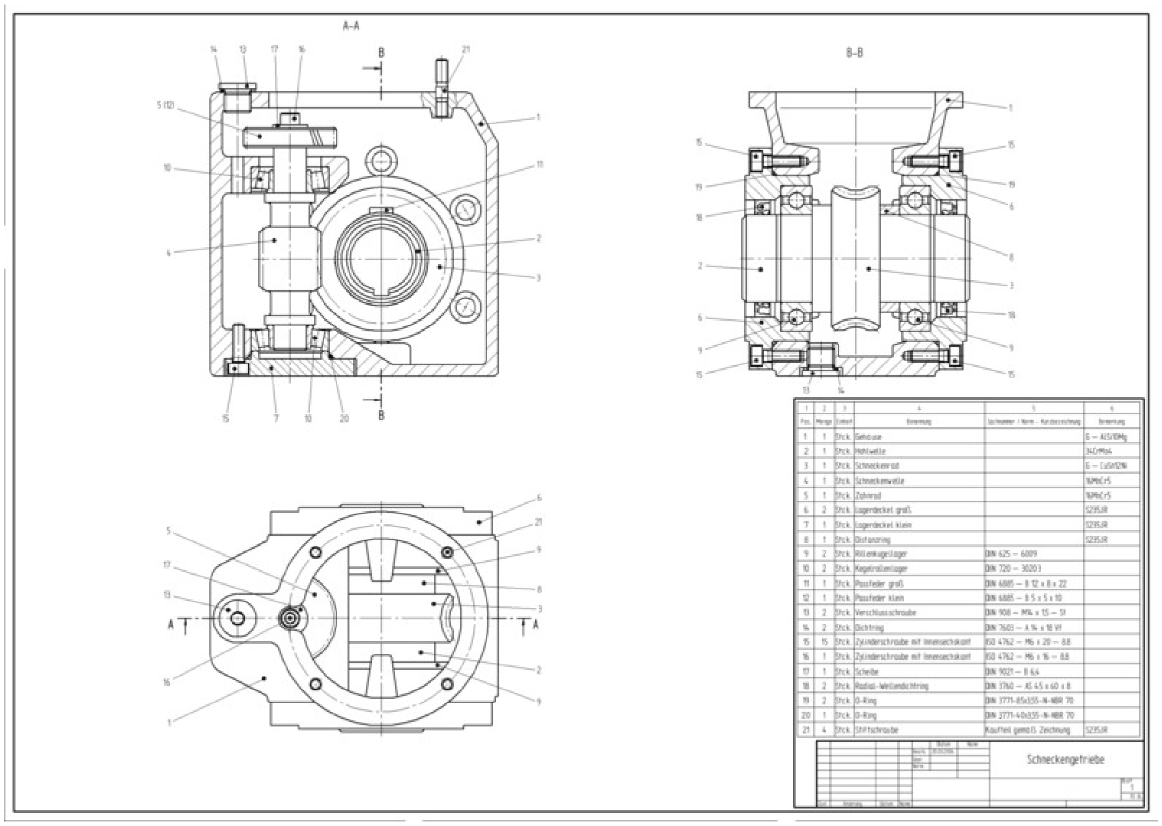
ISBN 978-81-323-3420-0

Published by:
Research World
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

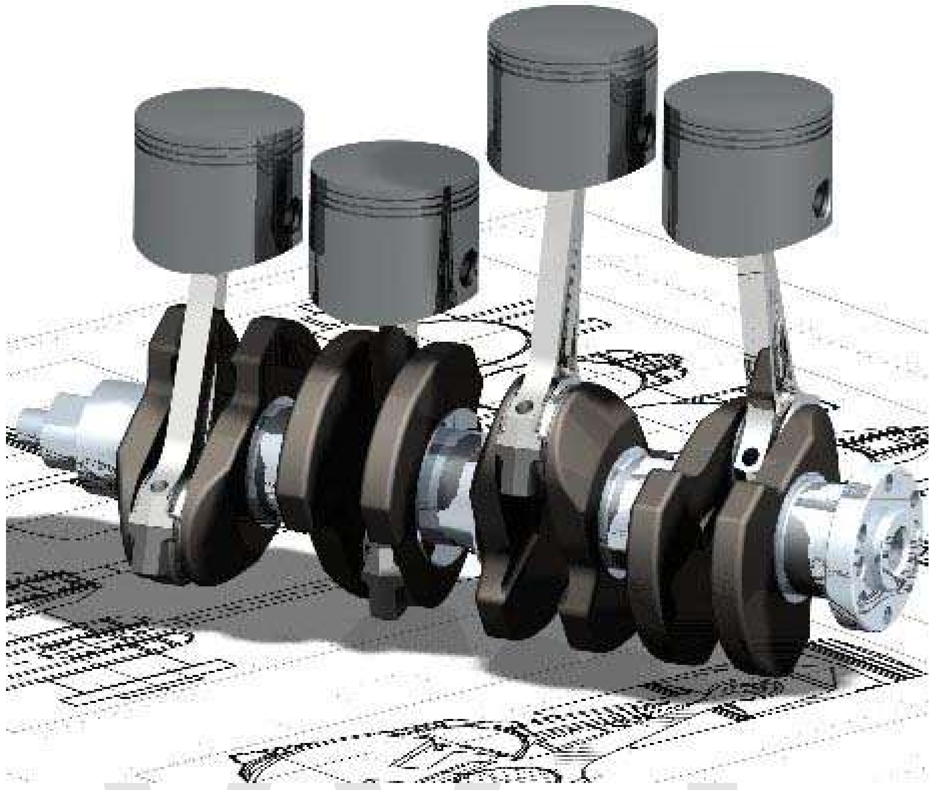
Chapter-1

Computer-aided Design



1	2	3	4	5	6
Pos.	Menge	Einheit	Benennung	Sachnummer / Norm – Kurzbezeichnung	Bemerkung
1	1	Stck.	Gehäuse		G – AlSi10Mg
2	1	Stck.	Hohlwelle		34CrMo4
3	1	Stck.	Schneckenrad		G – CuSn12Ni
4	1	Stck.	Schneckenwelle		16MnCr5
5	1	Stck.	Zahnrad		16MnCr5
6	2	Stck.	Lagerdeckel groß		S235JR
7	1	Stck.	Lagerdeckel klein		S235JR
8	1	Stck.	Distanzring		S235JR
9	2	Stck.	Rillenkugellager	DIN 625 – 6009	
10	2	Stck.	Kegelrollenlager	DIN 720 – 30203	
11	1	Stck.	Passfeder groß	DIN 6885 – B 12 x 8 x 22	
12	1	Stck.	Passfeder klein	DIN 6885 – B 5 x 5 x 10	
13	2	Stck.	Verschlussschraube	DIN 908 – M14 x 1,5 – St	
14	2	Stck.	Dichtring	DIN 7603 – A 14 x 18 Vf	
15	15	Stck.	Zylinderschraube mit Innensechskant	ISO 4762 – M6 x 20 – 8.8	
16	1	Stck.	Zylinderschraube mit Innensechskant	ISO 4762 – M6 x 16 – 8.8	
17	1	Stck.	Scheibe	DIN 9021 – B 6,4	
18	2	Stck.	Radial-Wellendichtring	DIN 3760 – AS 45 x 60 x 8	
19	2	Stck.	O-Ring	DIN 3771-85x3,55-N-NBR 70	
20	1	Stck.	O-Ring	DIN 3771-40x3,55-N-NBR 70	
21	4	Stck.	Stiftschraube	Kaufteil gemäß Zeichnung	S235JR

Example 2D CAD drawing



Example 3D CAD model

Computer-aided design (**CAD**), also known as **computer-aided design and drafting** (**CADD**), is the use of computer technology for the process of design and design-documentation. Computer Aided Drafting describes the process of drafting with a computer. CADD software, or environments, provides the user with input-tools for the purpose of streamlining design processes; drafting, documentation, and manufacturing processes. CADD output is often in the form of electronic files for print or machining operations. The development of CADD-based software is in direct correlation with the processes it seeks to economize; industry-based software (construction, manufacturing, etc.) typically uses vector-based (linear) environments whereas graphic-based software utilizes raster-based (pixelated) environments.

CADD environments often involve more than just shapes. As in the manual drafting of technical and engineering drawings, the output of CAD must convey information, such as materials, processes, dimensions, and tolerances, according to application-specific conventions.

CAD may be used to design curves and figures in two-dimensional (2D) space; or curves, surfaces, and solids in three-dimensional (3D) objects.

CAD is an important industrial art extensively used in many applications, including automotive, shipbuilding, and aerospace industries, industrial and architectural design, prosthetics, and many more. CAD is also widely used to produce computer animation for special effects in movies, advertising and technical manuals. The modern ubiquity and power of computers means that even perfume bottles and shampoo dispensers are designed using techniques unheard of by engineers of the 1960s. Because of its enormous economic importance, CAD has been a major driving force for research in computational geometry, computer graphics (both hardware and software), and discrete differential geometry.

The design of geometric models for object shapes, in particular, is occasionally called *computer-aided geometric design* (*CAGD*).

Overview

Beginning in the 1980s Computer-Aided Design programs reduced the need of draftsmen significantly, especially in small to mid-sized companies. Their affordability and ability to run on personal computers also allowed engineers to do their own drafting work, eliminating the need for entire departments. In today's world most, if not all, students in universities do not learn drafting techniques because they are not required to do so. The days of hand drawing for final drawings are almost obsolete. Universities no longer require the use of protractors and compasses to create drawings, instead there are several classes that focus on the use of CAD software such as Pro Engineer or IEAS-MS.

Current computer-aided design software packages range from 2D vector-based drafting systems to 3D solid and surface modellers. Modern CAD packages can also frequently allow rotations in three dimensions, allowing viewing of a designed object from any desired angle, even from the inside looking out. Some CAD software is capable of dynamic mathematic modeling, in which case it may be marketed as **CADD** — *computer-aided design and drafting*.

CAD is used in the design of tools and machinery and in the drafting and design of all types of buildings, from small residential types (houses) to the largest commercial and industrial structures (hospitals and factories).

CAD is mainly used for detailed engineering of 3D models and/or 2D drawings of physical components, but it is also used throughout the engineering process from conceptual design and layout of products, through strength and dynamic analysis of assemblies to definition of manufacturing methods of components. It can also be used to design objects.

CAD has become an especially important technology within the scope of computer-aided technologies, with benefits such as lower product development costs and a greatly shortened design cycle. CAD enables designers to lay out and develop work on screen, print it out and save it for future editing, saving time on their drawings.

Uses

Computer-aided design is one of the many tools used by engineers and designers and is used in many ways depending on the profession of the user and the type of software in question.

CAD is one part of the whole Digital Product Development (DPD) activity within the Product Lifecycle Management (PLM) process, and as such is used together with other tools, which are either integrated modules or stand-alone products, such as:

- Computer-aided engineering (CAE) and Finite element analysis (FEA)
- Computer-aided manufacturing (CAM) including instructions to Computer Numerical Control (CNC) machines
- Photo realistic rendering
- Document management and revision control using Product Data Management (PDM).

CAD is also used for the accurate creation of photo simulations that are often required in the preparation of Environmental Impact Reports, in which computer-aided designs of intended buildings are superimposed into photographs of existing environments to represent what that locale will be like were the proposed facilities allowed to be built. Potential blockage of view corridors and shadow studies are also frequently analyzed through the use of CAD.

Types

There are several different types of CAD . Each of these different types of CAD systems require the operator to think differently about how he or she will use them and he or she must design their virtual components in a different manner for each.

There are many producers of the lower-end 2D systems, including a number of free and open source programs. These provide an approach to the drawing process without all the fuss over scale and placement on the drawing sheet that accompanied hand drafting, since these can be adjusted as required during the creation of the final draft.

3D wireframe is basically an extension of 2D drafting (not often used today). Each line has to be manually inserted into the drawing. The final product has no mass properties associated with it and cannot have features directly added to it, such as holes. The operator approaches these in a similar fashion to the 2D systems, although many 3D systems allow using the wireframe model to make the final engineering drawing views.

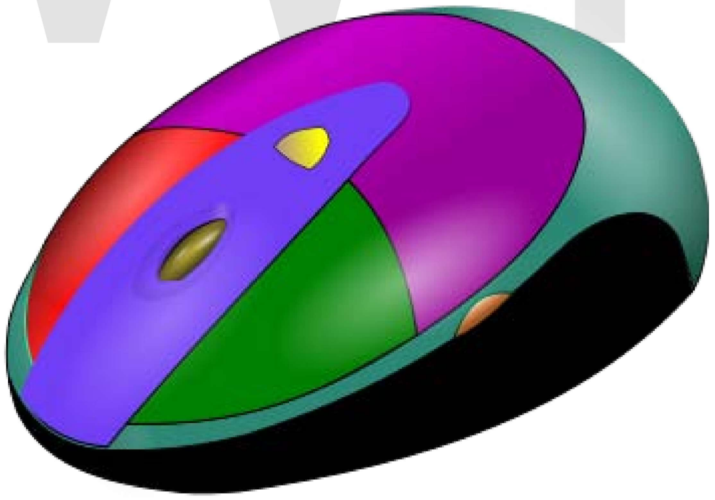
3D "dumb" solids are created in a way analogous to manipulations of real world objects (not often used today). Basic three-dimensional geometric forms (prisms, cylinders, spheres, and so on) have solid volumes added or subtracted from them, as if assembling or cutting real-world objects. Two-dimensional projected views can easily be generated from the models. Basic 3D solids don't usually include tools to easily allow motion of components, set limits to their motion, or identify interference between components.

3D parametric solid modeling require the operator to use what is referred to as "design intent". The objects and features created are adjustable. Any future modifications will be simple, difficult, or nearly impossible, depending on how the original part was created. One must think of this as being a "perfect world" representation of the component. If a feature was intended to be located from the center of the part, the operator needs to locate it from the center of the model, not, perhaps, from a more convenient edge or an arbitrary point, as he could when using "dumb" solids. Parametric solids require the operator to consider the consequences of his actions carefully.

Some software packages provide the ability to edit parametric and non-parametric geometry without the need to understand or undo the design intent history of the geometry by use of direct modeling functionality. This ability may also include the additional ability to infer the correct relationships between selected geometry (e.g., tangency, concentricity) which makes the editing process less time and labor intensive while still freeing the engineer from the burden of understanding the model's. These kind of non history based systems are called Explicit Modellers or Direct CAD Modelers.

Top end systems offer the capabilities to incorporate more organic, aesthetics and ergonomic features into designs. Freeform surface modelling is often combined with solids to allow the designer to create products that fit the human form and visual requirements as well as they interface with the machine.

Technology



A CAD model of a computer mouse.

Originally software for Computer-Aided Design systems was developed with computer languages such as Fortran, but with the advancement of object-oriented programming methods this has radically changed. Typical modern parametric feature based modeler and freeform surface systems are built around a number of key C modules with their own

APIs. A CAD system can be seen as built up from the interaction of a graphical user interface (GUI) with NURBS geometry and/or boundary representation (B-rep) data via a geometric modeling kernel. A geometry constraint engine may also be employed to manage the associative relationships between geometry, such as wireframe geometry in a sketch or components in an assembly.

Unexpected capabilities of these associative relationships have led to a new form of prototyping called digital prototyping. In contrast to physical prototypes, which entail manufacturing time in the design.

Today, CAD systems exist for all the major platforms (Windows, Linux, UNIX and Mac OS X); some packages even support multiple platforms.

Right now, no special hardware is required for most CAD software. However, some CAD systems can do graphically and computationally expensive tasks, so a good graphics card, high speed (and possibly multiple) CPUs and large amounts of RAM are recommended.

The human-machine interface is generally via a computer mouse but can also be via a pen and digitizing graphics tablet. Manipulation of the view of the model on the screen is also sometimes done with the use of a Spacemouse/SpaceBall. Some systems also support stereoscopic glasses for viewing the 3D model.

History

Designers have long used computers for their calculations. Initial developments were carried out in the 1960s within the aircraft and automotive industries in the area of 3D surface construction and NC programming, most of it independent of one another and often not publicly published until much later. Some of the mathematical description work on curves was developed in the early 1940s by Robert Issac Newton from Pawtucket, Rhode Island. Robert A. Heinlein in his 1957 novel *The Door into Summer* suggested the possibility of a robotic *Drafting Dan*. However, probably the most important work on polynomial curves and sculptured surface was done by Pierre Bezier (Renault), Paul de Casteljau (Citroen), Steven Anson Coons (MIT, Ford), James Ferguson (Boeing), Carl de Boor (GM), Birkhoff (GM) and Garibedian (GM) in the 1960s and W. Gordon (GM) and R. Riesenfeld in the 1970s.

It is argued that a turning point was the development of the SKETCHPAD system at MIT in 1963 by Ivan Sutherland (who later created a graphics technology company with Dr. David Evans). The distinctive feature of SKETCHPAD was that it allowed the designer to interact with his computer graphically: the design can be fed into the computer by drawing on a CRT monitor with a light pen. Effectively, it was a prototype of graphical user interface, an indispensable feature of modern CAD.

The first commercial applications of CAD were in large companies in the automotive and aerospace industries, as well as in electronics. Only large corporations could afford the computers capable of performing the calculations. Notable company projects were at GM

(Dr. Patrick J.Hanratty) with DAC-1 (Design Augmented by Computer) 1964; Lockheed projects; Bell GRAPHIC 1 and at Renault (Bezier) – UNISURF 1971 car body design and tooling.

One of the most influential events in the development of CAD was the founding of MCS (Manufacturing and Consulting Services Inc.) in 1971 by Dr. P. J. Hanratty, who wrote the system ADAM (Automated Drafting And Machining) but more importantly supplied code to companies such as McDonnell Douglas (Unigraphics), Computervision (CADDS), Calma, Gerber, Autotrol and Control Data.

As computers became more affordable, the application areas have gradually expanded. The development of CAD software for personal desktop computers was the impetus for almost universal application in all areas of construction.

Other key points in the 1960s and 1970s would be the foundation of CAD systems United Computing, Intergraph, IBM, Intergraph IGDS in 1974 (which led to Bentley Systems MicroStation in 1984)

CAD implementations have evolved dramatically since then. Initially, with 3D in the 1970s, it was typically limited to producing drawings similar to hand-drafted drawings. Advances in programming and computer hardware, notably solid modeling in the 1980s, have allowed more versatile applications of computers in design activities.

Key products for 1981 were the solid modelling packages -Romulus (ShapeData) and Uni-Solid (Unigraphics) based on PADL-2 and the release of the surface modeler CATIA (Dassault Systemes). Autodesk was founded 1982 by John Walker, which led to the 2D system AutoCAD. The next milestone was the release of Pro/ENGINEER in 1988, which heralded greater usage of feature-based modeling methods and parametric linking of the parameters of features. Also of importance to the development of CAD was the development of the B-rep solid modeling kernels (engines for manipulating geometrically and topologically consistent 3D objects) Parasolid (ShapeData) and ACIS (Spatial Technology Inc.) at the end of the 1980s and beginning of the 1990s, both inspired by the work of Ian Braid. This led to the release of mid-range packages such as SolidWorks in 1995, Solid Edge (then Intergraph) in 1996 and Autodesk Inventor in 1999.

Chapter-2

Building Information Modeling

Building Information Modeling (BIM) is the process of generating and managing building data during its life cycle. Typically it uses three-dimensional, real-time, dynamic building modeling software to increase productivity in building design and construction. The process produces the Building Information Model (also abbreviated BIM), which encompasses building geometry, spatial relationships, geographic information, and quantities and properties of building components.

Origins of BIM

Charles M. Eastman at Georgia Tech coined the term BIM,. This theory is based on a view that the term BIM "Building Information Model" is basically the same as "Building Product Model", which Eastman has used extensively in his book and papers since the late 1970s. ('Product model' means 'data model' or 'information model' in engineering.)

Architect and Autodesk building industry strategist Phil Bernstein, FAIA, first used the actual term BIM "building information modeling." Jerry Laiserin then helped popularize and standardize it as a common name for the digital representation of the building process as then offered primarily by Graphisoft, Bentley Systems, and Autodesk to facilitate exchange and interoperability of information in digital format. According to him and others, the first implementation of BIM was under the **Virtual Building** concept by Graphisoft's ArchiCAD, in its debut in 1987.

Definition

Building information modeling covers geometry, spatial relationships, light analysis, geographic information, quantities and properties of building components (for example manufacturers' details). BIM can be used to demonstrate the entire building life cycle, including the processes of construction and facility operation. Quantities and shared properties of materials can be extracted easily. Scopes of work can be isolated and defined. Systems, assemblies and sequences can be shown in a relative scale with the entire facility or group of facilities. Dynamic information of the building, such as sensor measurements and control signals from the building systems, can also be incorporated within BIM to support analysis of building operation and maintenance .

Under the guidance of a Virtual Design to Construction Project Manager (VDC) BIM can be seen as a companion to PLM as in the Product Lifecycle Management, since it goes beyond geometry and addresses issues such as Cost Management, Project Management and provides a way to work concurrently on most aspects of building life cycle processes.

BIM goes far beyond switching to a new software. It requires changes to the definition of traditional architectural phases and more data sharing than most architects and engineers are used to.

BIM is able to achieve such improvements by modeling representations of the actual parts and pieces being used to build a building. This is a substantial shift from the traditional computer aided drafting method of drawing with vector file-based lines that combine to represent objects.

The interoperability requirements of construction documents include the drawings, procurement details, environmental conditions, submittal processes and other specifications for building quality. It is anticipated by proponents that VDC utilizing BIM can bridge the information loss associated with handing a project from design team, to construction team and to building owner/operator, by allowing each group to add to and reference back to all information they acquire during their period of contribution to the BIM model. For example, a building owner may find evidence of a leak in his building. Rather than exploring the physical building, he may turn to his BIM and see that a water valve is located in the suspect location. He could also have in the model the specific valve size, manufacturer, part number, and any other information ever researched in the past, pending adequate computing power. Such problems were initially addressed by

Leite et al. when developing a vulnerability representation of facility contents and threats for supporting the identification of vulnerabilities in building emergencies

There have been attempts at creating a BIM for older, pre-existing facilities. They generally reference key metrics such as the Facility Condition Index (FCI). The validity of these models will need to be monitored over time, because trying to model a building constructed in, say 1927, requires numerous assumptions about design standards, building codes, construction methods, materials, etc., and therefore is far more complex than building a BIM at time of initial design.

The American Institute of Architects has further defined BIM as "a model-based technology linked with a database of project information", and this reflects the general reliance on database technology as the foundation. In the future, structured text documents such as specifications may be able to be searched and linked to regional, national, and international standards.

Managing the BIM Model guidelines

"The production of a Building Information Model (BIM) for the construction of a project involves the use of an integrated multi-disciplinary performance model to encompass the building geometry, spatial relationships, geographic information, along with quantities and properties of the building components. The Virtual Design to Construction Project Manager (VDC - also known as VDCPM) is a professional in the field of project management and delivery. The VDC is retained by a design build team on the clients' behalf from the pre-design phase through certificate of occupancy in order to develop and to track the object oriented BIM against predicted and measured performance objectives. The VDC manages the project delivery through multi-disciplinary building information models that drive analysis, schedules, take-off, and logistics. The VDC is skilled in the use of BIM as a tool to manage and assess the technology, staff, and procedural needs of a project. In short the VDC is a contemporary project managing architect who is equipped to deal with the current evolution of project delivery. The VDC acts as a conduit to bridge time tested construction knowledge to digital analysis and representation."

BIM in the UK

In the UK, CPIC, responsible for providing best practice guidance on construction production information and formed by representatives of the major UK industry institutions, has proposed a definition of Building Information Modelling for adoption throughout the UK construction industry and has invited all UK industry parties to discuss it in order to ensure an agreed starting point. The proliferation of interpretations of the term currently hampers the adoption of a working method that will drastically improve the construction industry and the quality and sustainability of the deliveries from the design and construction team to clients.

BIM in the USA

Contractors

The Associated General Contractors and contracting firms also have developed a variety of working definitions of BIM that describe it generally as "an object-oriented building development tool that utilizes 5-D modeling concepts, information technology and software interoperability to design, construct and operate a building project, as well as communicate its details." 5-D modeling concepts involve modeling not only the 3 primary spatial dimensions of X, Y, and Z; but also time as the 4th dimension and cost as the 5th.

Although the concept of BIM and relevant processes are being explored by contractors, architects and developers alike, the term itself is under debate, and it is yet to be seen whether it will win over alternatives, which include:

- Virtual Building Environment (VBE)
- Virtual Design to Construction Project Manager (VDC)

BIM is seen to be closely related to Integrated Project Delivery (IPD) where the primary motive is to bring the teams together early on the project. . A full implementation of BIM also requires the project teams to collaborate from the inception stage and formulate model sharing and ownership contract documents.

BIM is often associated with IFCs (Industry Foundation Classes) and aecXML, which are data structures for representing information used in BIM. IFCs is developed by buildingSMART (International Alliance for Interoperability). Other data structures are proprietary, and many have been developed by CAD firms that are now incorporating BIM into their software. One of the earliest examples of a nationally approved BIM standard is the AISC (American Institute of Steel Construction)-approved CIS/2 standard, a non proprietary standard with its roots in the UK.

Proponents claim that BIM offers:

1. Improved visualization
2. Improved productivity due to easy retrieval of information
3. Increased coordination of construction documents
4. Embedding and linking of vital information such as vendors for specific materials, location of details and quantities required for estimation and tendering
5. Increased speed of delivery
6. Reduced costs

In August 2004 the US National Institute of Standards and Technology (NIST) issued a report entitled "Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry" (NIST GCR 04-867 (PDF), which came to the conclusion that, as a conservative estimate, $15.8 billion is lost annually by the U.S. capital facilities industry resulting from inadequate interoperability due to "the highly fragmented nature of the industry, the industry's continued paperbased business practices, a lack of standardization, and inconsistent technology adoption among stakeholders".

BIM in France

In France, several bodies are pushing for a more integrated adoption of BIM standards, in order to improve software interoperability and cooperation among actors of the building industry. Examples are the FFB (Fédération française du bâtiment), or the French arm of buildingSMART International who are supporting IFCs.

On the other hand, software editing companies such as Vizelia were early adopters of IFCs and can now benefit from the full potential of BIM in the Green Building fast-emerging business.

Anticipated future potential

BIM is a relatively new technology in an industry typically slow to adopt change. Yet many early adopters are confident that BIM will grow to play an even more crucial role in building documentation.

BIM provides the potential for a virtual information model to be handed from Design Team (architects, surveyors, consulting engineers, and others) to Contractor and Subcontractors and then to the Owner, each adding their own additional discipline-specific knowledge and tracking of changes to the single model. The result greatly reduces the information loss that occurs when a new team takes "ownership" of the project as well as in delivering extensive information to owners of complex structures. It also prevents errors made by design team members as well as the construction team (Contractors and Subcontractors) by allowing the use of conflict detection where the computer actually informs team members about parts of the building in conflict or clashing, and through detailed computer visualization of each part in relation to the total building. As computers and software become more capable of handling more building information, this will become even more pronounced than it is in current design and construction projects. This error reduction is a great part of cost savings realized by all members of a project. Reduction in time required to complete construction directly contributes to the cost savings numbers as well. It's important to realize that this decrease can only be accomplished if the models are sufficiently developed in the Design Development phase.

The Industry Foundation Classes (IFC/ifcXML) are an open specification for Building Information Modeling and are used to share and exchange BIM in a neutral format among various software applications. Green Building XML (gbXML) is an emerging schema, a subset of the Building Information Modeling efforts, focused on green building design and operation. gbXML is used as input in several energy simulation engines. But with the development of modern computer technology, a large number of building energy simulation tools are available on the market. When choosing which simulation tool to use in a project, the user must consider the tool's accuracy and reliability, considering the building information they have at hand, which will serve as input for the tool. Yezioro, Dong and Leite developed an artificial intelligence approach towards assessing building performance simulation results and found that more detailed simulation tools have the best simulation performance in terms of heating and cooling electricity consumption within 3% of mean absolute error.

Chapter-3

CAD Data Exchange and Collaborative Product Development

CAD data exchange

CAD data exchange involves a number of software technologies and methods to translate data from one Computer-aided design system to another CAD file format. This PLM technology is required to facilitate collaborative work (CPD) between OEMs and their suppliers.

The main topic is with the translation of geometry (wireframe, surface and solid) but also of importance is other data such as attributes; metadata, assembly structure and feature data.

Methods of translation

There are basically three methods of transferring data from one CAD system to another.

- Direct CAD system export/import
- Direct 3rd party translators.
- Intermediate data exchange formats

Direct internal

Some CAD systems can directly read and/or write other CAD formats, simply by using file open and file save as options. As most CAD file formats are not open, this option is limited to either systems owned by the same company or via hacking of competitor's file format.

Direct external

There are a number of companies that specialize in CAD data translation software, providing software that can read one system and write the information in another CAD system format. These systems have their own proprietary intermediate format some of which will allow reviewing the data during translation. Some of these translators work stand-alone while others require one or both of the CAD packages installed on the translation machine as they use code (APIs) from these systems to read/write the data.

Data translation formats

A common method of translation is via an intermediary format. The sending CAD system exports out to this format and the receiving CAD system reads in this format. Some formats are independent of the CAD vendors being defined by standards organisations while others, although owned by a company, are widely used and are regarded as quasi industry standards. It is becoming increasingly common for companies owning these quasi industry standards to further the use of their formats by openly publishing these data formats.

Example formats:

- STEP – ISO 10303, a replacement for IGES and VDA-FS with the CAD specific parts:
  - STEP AP203 and AP214: Mechanical CAD systems
  - STEP AP210: CAD systems for printed circuit board
  - STEP AP212: CAD systems for electrical installation and cable harness
  - STEP-NC AP238: CAD, CAM, and CNC machining process information
- IGES
- VDA-FS
- DXF
- Parasolid XT
- JT Open
- DRG

A number of CAD data exchange methods are described by recent academic studies. The neutral modeling command (NMC) method, proposed by Zhejiang University, is an example of these methods.

Level of information detail translated.

As each CAD system has its own method of describing geometry, both mathematically and structurally, there is always some loss of information when translating data from one CAD data format to another. The intermediate file formats are also limited in what they can describe, and they can be interpreted differently by both the sending and receiving systems.

It is therefore important when transferring data between systems to identify what needs to be translated.

If only the 3D model is required for the downstream process, then only the model description needs to be transferred. However, there are levels of detail. For example: is the data wireframe, surface, or solid; is the topology (BREP) information required; must the face and edge identifications be preserved on subsequent modification; must the feature information and history be preserved between systems; and is PMI annotation to be transferred.

With product models, retaining the assembly structure may be required.

If drawings need to be translated, the wireframe geometry is normally not an issue; however text, dimensions and other annotation can be an issue, particularly fonts and formats.

No matter what data is to be translated, there is also a need to preserve attributes (such as color and layer of graphical objects) and text information stored within the files.

Sometimes, however, there is a problem caused by too much information being preserved. An example are the constraints placed on designers arising out of the design intent-history captured in parametric design systems. The receiving system must provide designers with the design freedom to modify geometry without having to understand the history of, or undo, the design tree.

Some translation methods are more successful than others at translating data between CAD systems.

MultiCAD Digital Mockups

Two CAD/CAM/CAE PLM trends have been driving CAD Data Exchange technology. One is the need for close interaction throughout today's extended multiCAD enterprises. The other is the increased reliance on digital mockups to permit visualization, design in context, simulation and analysis of large scale assemblies prior to the actual manufacture of the physical product. Ongoing advances in data exchange technology have enabled significant fulfillment of those needs.

The ability to visualize medium if not large scale assemblies was one of the early successes of these CAD translation formats. Hardware improvements and the development of lightweight formats supported larger scale assemblies. By early 2005, Siemens Power Generation unit had implemented a multi-site collaboration environment using 800 licenses of Teamcenter at 180 sites including 70 suppliers. The system uses Siemens PLM/XML as its data transport protocol and JT Open for sharing 3D models.

Current advances now allow an "Active Mockup." This technology allows design in context with simulations such as dynamic clearance analysis and automatic generation of motion envelopes. Active mockups allow the edit of components from directly within the multi-CAD assembly. Multiple level-of-detail displays support interactive performance even in huge assemblies.

CAD to CAM Data Exchange

NC programming typically requires that the geometry received from a CAD system, whether in wireframe, surface, solid or combined formats, be free from any irregularities and inconsistencies that may have occurred in the CAD phase of geometry creation. Data exchange from CAD to CAM must therefore include tools for identifying and repairing those inconsistencies. These tools are typically included in the data exchange software of each CAM solution-set.

In a true PLM environment, CAD to CAM data exchange must provide for more than the transfer of geometry. Product Manufacturing Information, whether generated by the designer for use by manufacturing, or generated by the manufacturing organization for use by design, must be a part of the data exchange system. STEP-NC was designed to carry GD&T and other PMI through CAD and CAM into a CNC.

Collaborative product development

Collaborative product development (collaborative product design) (CPD) is a business strategy, work process and collection of software applications that facilitates different organizations to work together on the development of a product. It is also known as **collaborative product definition management** (**cPDM**).

Introduction

Exactly what technology comes under this title does vary depending on who you ask; however, it usually consists of the PLM areas of: Product Data Management (PDM); Product visualization; team collaboration and conferencing tools; and supplier sourcing software. It is generally accepted as not including CAD geometry tools, but does include data translation technology.

Technologies and methods used

Clearly general collaborative software such as email and chat (instant messaging) is used within the CPD process. One important technology is application and desktop sharing, allowing one person to view what another person is doing on a remote machine. For CAD and product visualization applications an ‘appshare’ product that supports OpenGL graphics is required. Another common application is Data sharing via Web based portals.

Specific to product data

With product data an important addition is the handling of high volumes of geometry and metadata. Exactly what techniques and technology is required depends on the level of collaboration being carried out and the commonality (or lack of) the partner sites' systems.

Specific to PLM and CAx collaboration

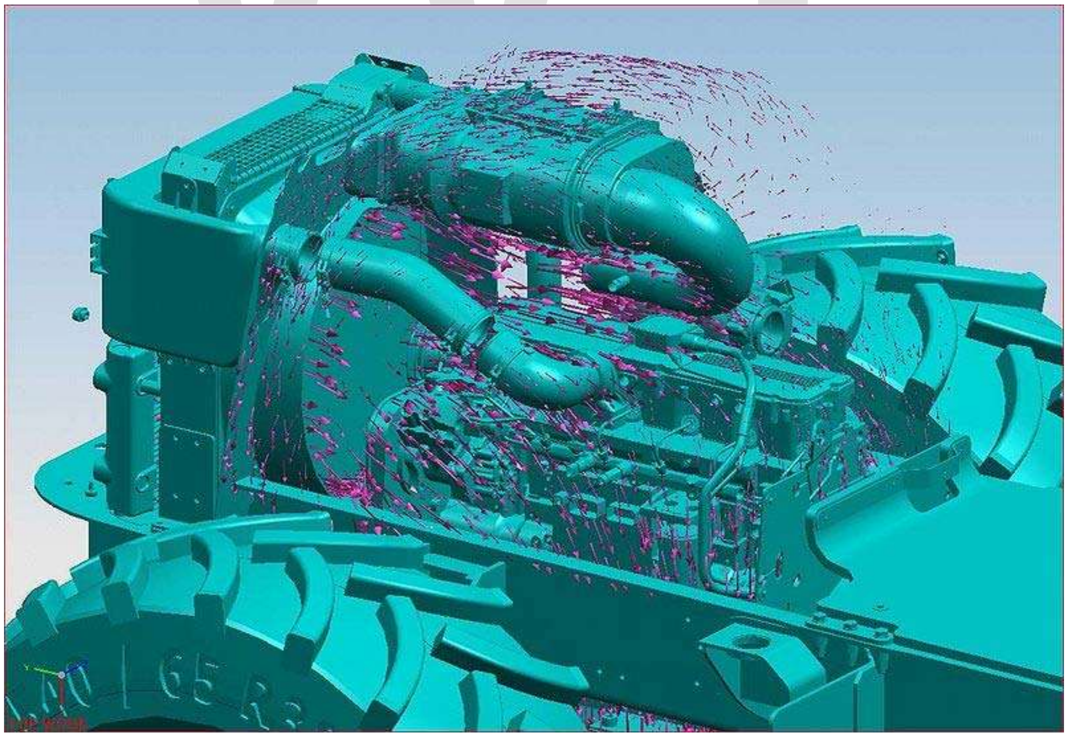
Collaboration using PLM and CAx tools requires technology to support the needs of:

1. People. Personnel of different disciplines and skill levels;
2. Organizations: Organizations throughout an enterprise or extended enterprise with different rules, processes and objectives;
3. Data: Data from different sources in different formats.

Appropriate technologies are required to support collaboration across these boundaries.

People

Effective PLM collaboration will typically require the participation of people who do not have high level CAD skills. This requires improved user interfaces including tailorable user interfaces that can be tailored to the skill level and specialty of the user.



Visualization of simulated airflow over an engine

Improved visualization capabilities, especially those that provide a meaningful view of complex information such as the results of a fluid flow analysis will leverage the value of all participants in the collaboration process. Effective collaboration requires that a participant be freed from the burden of knowing the intent history typically imbedded within and constricting the use of parametric models.

Organizations

Community collaboration requires that companies, suppliers, and customers share information in a secure environment, ensure compliance with enterprise and regulatory rules and enforce the process management rules of the community as well as the individual organizations.

Data

The most basic collaboration data need is the ability to operate in a MultiCAD environment. That is, however, only the beginning. Models from multiple CAD sources must be assembled into an active digital mockup allowing change and/or design in context.

Real-time collaborative product design

Product design is typically a highly iterative and interactive activity involving a group of designers who are geographically dispersed. A neutral modeling commands (NMC) based method is proposed to construct a real-time collaborative product design platform within heterogeneous CAD systems . Different from the visualization-based approaches, models can be constructed and modified synchronously from various sites in the proposed collaborative design environment. Based on a translation mechanism between system modeling operations (SMO) and neutral modeling commands (NMC), every operation given by a user on one site will be translated into a NMC and be sent to all the other sites through the network. When the other sites receive this command, it is converted into corresponding SMOs on the local system. In this way, the real-time collaborative product design with heterogeneous CAD systems is achieved.

Different levels of collaboration

If the collaborating parties have the same PDM and CAD systems the task usually involves the direct access and transfer of data between sites. The PDM system will have data storage at more than one site for the large graphics files, file may be copied between sites, how they are synchronized being controlled by the server(s). For the management server and metadata there are a number of options. There could be a single server that is accessed from all locations or multiple PDM servers that communicate with one another. In both cases the PDM software controls access for groups defining what data they can see and edit.

With different CAD systems the approach varies slightly depending on whether the ownership, and therefore authorship of components changes or not. If geometry only has to be viewed then a Product visualization neutral file format (e.g.JT) can be used for tasks such as viewing, markup (redlining) or multi-cad digital mock-up (DMU). It maybe that authorship does not change but components from one group needs to be placed in the assembly of another group so that they can construct their parts, so called work in context. This requires transfer of geometry from one format to another by means of a visualization format or full data translation. Between some systems there is the possibility of 'data interoperability' were geometry from one format can be associatively copied to another. If the ownership of a particular file is being transfer then full data translation is required using some form of CAD data exchange technology. For the translation process Product Data Quality (PDQ) checkers are often employed to reduce problems in transferring the work. If different PDM/EDM systems are in use then either data structures or metadata can be transferred using STEP or communication between databases can be achieved with tools based around XML data transfer.

Chapter-4

Feature Recognition

The term "**feature**" does not imply the same meaning in different engineering disciplines. This has resulted in several ambiguous definitions for feature. A feature, in computer-aided design (CAD) software, can be called a region of a part with some interesting geometric or topological patterns. This meaning can refer to all sorts of information, such as for example, shape, functional or manufacturing information. Although many types of features have been investigated, the most common type of feature is the form feature, which contains both shape information and parametric information. Examples of form features common in many shape models are round holes, slots, bosses, and pockets.

Features can also be used to represent manufacturing information of the part. Different manufacturing domains require different feature representations. Some of the properties that need to be encoded by features are assembly method, manufacturing process and tolerances. A manufacturing feature can be defined as a form feature, but not necessarily vice versa . Among manufacturing features, the ones received extensive attention are the machining features. A machining feature can be regarded as the volume swept by a cutting tool. In this sense, it is always a negative (subtracted) volume, in contrast with form features that are sometimes positive (added) volumes.

Feature data in a CAD model can be represented either as a collection of surfaces or volumetrically. Surface features are naturally used for example to describe manufacturing tolerances or locating surfaces in fixture design. volumetric features on the other hand, are used in process planning since manufacturing information (particularly in machining) is better portrayed volumetrically .

The first published work on features was for the original boundary representation modelling system, BUILD, and was performed by Lyc Kyprianou . Soon other work followed based on different solid representations. Overviews on the work on features can be found in Shah et al.; Subrahmanyam and Wozny; Salomons et al.

Feature Technology

Work on features (generally called feature technology)can be divided into two rough categories: Design-by-features and Feature recognition. In design-by-features, also known as feature-based design (FBD), feature structures are introduced directly into a model using particular operations or by sewing in shapes. On the other hand, the goal of feature recognition (FR) is to algorithmically extract higher level entities (e.g. manufacturing features) from lower level elements (e.g. surfaces, edges, etc) of a CAD model.

Design by Features

By using features to build up shape models, the design process is made more efficient, because the shape of features can be pre-defined. Features in FBD can be directly associated to manufacturing information so that these information can be retrieved in downstream applications. In this way, an overall CAD/CAM system can be fully automated, however, the idea of using manufacturing features to design a part has its own shortcomings : The features used to design the part do not necessarily represent the best way to manufacture it. It is, therefore, the designer's responsibility to evaluate all methods that can produce the part. Furthermore, manufacturing features are not the most natural way of designing a part.

Feature Recognition

The classical Kyprianou's method was aimed to encode parts for group technology (GT). The purpose of GT is to systematically classify objects based on their manufacturing method. Kyprianou's work involved classifying faces into primary and secondary groups and then identifying features according to patterns of these primary or secondary faces. A primary face is one with multiple boundaries (also called "hole-loops") or mixed concave and convex boundaries. A concave boundary is a set of concave edges, where the solid angle over the edge is more than 180. Secondary faces are all other faces. Kyprianou's work was continued and extended by Jared et al. to cover a number of important special cases where features interacted.

Automatic Feature Recognition (AFR) is regarded as an ideal solution to automate design and manufacturing processes. Successful automation of CAD and CAM systems is a vital connection in building Computer Integrated Manufacturing (CIM) systems.. This is the part of the FR research that has attracted much of the attention. Another important application of AFR is for manufacturability evaluation The AFR system should be able to interpret the design differently based on alternative features and feed back the manufacturability and cost of those interpretations to the designer.

There is a big stockpile of different AFR techniques that has been proposed for CAD/CAM integration and process planning. Han et al. provides a critical and detailed analysis of some of the existing approaches. The most common methods according to Han et al. range from graph-based algorithms to hint-based and volumetric decomposition techniques. In the graph-based feature recognition, a graph showing the topology of the part (connection of faces) is created. The graph is often attributed, for example the edges are marked as concave or convex . This graph is then analyzed to extract subsets of nodes and arcs that match with any predefined template. This is done by a variety of techniques, including graph iso-morphism algorithms.

Graph based approaches have been criticized for several shortcomings. They fail to account for manufacturability of the recognized features due to their strong reliance on topological patterns rather than geometry. The intersection of features causes an explosion in the number of possible feature patterns that spoils any attempt to formulate feature patterns. To address these difficulties, Vandenbrande and Requicha. proposed to search for "minimal indispensable portion of a feature's boundary", called hints, rather than complete feature patterns. For example, presence of two opposing planar faces is a hint for potential existence of a slot feature. Hints are not necessarily restricted to the part geometry. They can be extracted form tolerances and design attributes as well. For example, "a thread attribute may be taken as a hole hint" . This approach has been more successful in recognizing intersecting features. However, the efficiency of the approach has been argued, as there could be a huge number of traces that won't lead to valid features. Some authors have been in favor of using a hybrid of graph based and hint based FR. Other existing FR approaches are volumetric decomposition , Artificial Neural Networks, and expert systems Babic et al. briefly introduces many of them.

However, building feature recognition systems that function effectively on real industrial products has been elusive. A real product with hundreds of faces and end edges brings almost all the above approaches to a halt due to computational complexity. Furthermore, the features studied in these approaches are usually over simplified. The bulk of the feature recognition literature normally deals with 2.5D features (those made by sweeping a 2D profile along a linear axis). Graph representations, hint definitions or volume decompositions are much more difficult to define for 3D and free form features. The work done by Sundararajan is focused on free form surfaces, but again it is limited in application. Oversimplification is also evident even in the course of 2.5D features. For example, feature recognition algorithms usually assume sharp concave edges in the feature geometry. However, such edges are barely used in real design of mechanical components due to manufacturing constrains. Some of these issues such as the presence of filleted edges and free form surfaces in the model have been studied by Rahmani and Arezoo .

Commercial Feature Recognition Systems

Few commercial feature recognition systems are also available. Though feature recognition technology can be applied for various applications, commercial software have effectively adopted feature recognition technology for recreating the feature tree from imported models so that even the imported models can be edited as if it were a native solid model. Major 3D CAD modelers have Feature Recognition to convert imported 3-D models into native feature based models. CAM software and design for manufacturing software are also built using this feature recognition technology. Few

CAD/CAM software have used commercially available third-party feature recognition library, which recognizes various features from 3-D B-Rep models. Separate libraries are available for Design, Manufacturing and Sheet metal applications. Design feature recognition library can identify features such as holes of various types, split holes, hole-chains, fillets, chamfers, cut extrudes, boss extrudes, drafted extrudes, revolved cuts, revolved bosses, ribs, drafts, lofts and sweeps are identified. Manufacturing feature recognition library provides recognition of manufacturing features such as simple holes, tapered holes, counter-bore holes, counter-sunk holes, counter-drilled holes, hole-chains, hole patterns such as linear, rectangular and circular patterns, fillets, chamfers, blind pockets, through pockets, drafted pockets, filleted and chamfered pockets, simple slots, drafted slots, filleted and chamfered slots, islands in pockets and slots, machinable volumes, machinable slabs, multiple intersecting features, axi-symmetric features such as external turned profiles, internal turned profiles, turned grooves such as vee and dovetail grooves, and mill-turn features such as slots and pocket in turned profiles. Sheet metal feature recognition library extracts features from a sheet metal perspective. Various features identified through this library include walls, bends, holes, cutouts, flanged holes, flanged cutouts, notches, open hems, closed hems, teardrop hems, rolled hems (curls), jog flanges, edge flanges, contour flanges, stamps such as louver, lance, bridge, dimple, beads, embosses and ribs. Though such commercial systems can identify a variety of features listed above, further research can be driven to identify feature types that are not identified by such commercial systems. Manufacturing features such as 3-axis and 5-axis feature recognition are generally not available in such commercial systems.

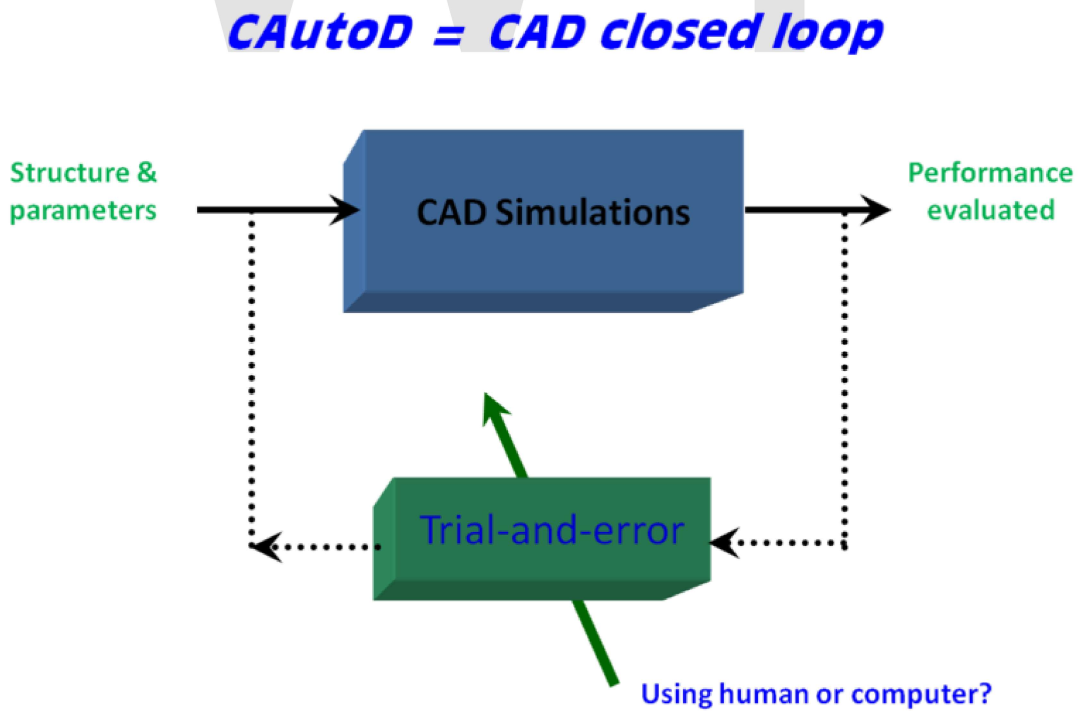
Chapter-5

Computer-automated Design and Geometric Design

Computer-automated design

Design Automation usually refers to electronic design automation. Extending Computer-Aided Design (CAD), automated design and **Computer-Automated Design (CAutoD)** are more concerned with a broader range of applications, such as automotive engineering, civil engineering , composite material design, control engineering , dynamic system identification , financial systems, industrial equipment, mechatronic systems, steel construction , structural optimisation, and the invention of novel systems. Traditional CAD simulation can be transformed to CAutoD by biologically-inspired machine learning or search techniques such as evolutionary computation, including swarm intelligence algorithms.

Guiding designs by performance improvements



Interaction in computer-automated design

To meet the ever growing demand of quality and competitiveness, iterative physical prototyping is now often replaced by 'digital prototyping' of a 'good design', which aims to meet multiple objectives such as maximised output, energy efficiency, highest speed and cost-effectiveness. The design problem concerns both finding the best design within a known range (i.e., through 'learning' or 'optimisation') and finding a new and better design beyond the existing ones (i.e., through creation and invention). This is equivalent to a search problem in an, almost certainly, multidimensional (multivariate), multi-modal space with a single (or weighted) objective or multiple objectives.

Normalized bjective function: cost vs. fitness

Using single-objective CAutoD as an example, if the objective function, either as a cost function $J \in [0, \infty)$, or inversely, as a fitness function $f \in (0, 1]$, where

$$f = \frac{J}{1+J},$$

is differentiable under practical constraints in the multidimensional space, the design problem may be solved analytically. Finding the parameter sets that result in a zero first-order derivative and that satisfy the second-order derivative conditions would reveal all local optima. Then comparing the values of the performance index of all the local optima, together with those of all boundary parameter sets, would lead to the global optimum, whose corresponding 'parameter' set will thus represent the best design. However, in practice, the optimization usually involves multiple objectgives and the matters involving derivatives are lot more complex.

Dealing with practical objectives

In practice, the objective value may be noisy or even non-numerical, and hence its gradient information may be unreliable or unavailable. This is particularly true when the problem is multi-objective. At present, many designs and refinements are mainly made through a manual trial-and-error process with the help of a CAD simulation package. Usually, such *a posteriori* learning or adjustments need to be repeated many times until a ‘satisfactory’ or ‘optimal’ design emerges.

Exhaustive search

In theory, this adjustment process can be automated by computerised search, such as exhaustive search. As this is an exponential algorithm, it may not deliver solutions in practice within a limited period of time.

Search in polynomial time

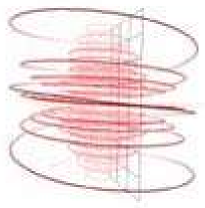
One approach to virtual engineering and automated design is evolutionary computation such as evolutionary algorithms.

Evolutionary algorithms

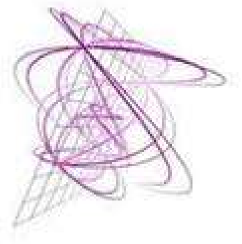
To reduce the search time, the biologically-inspired evolutionary algorithm (EA) can be used instead, which is a (non-deterministic) polynomial algorithm. The EA based multi-objective "search team" can be interfaced with an existing CAD simulation package in a batch mode. The EA encodes the design parameters (encoding being necessary if some parameters are non-numerical) to refine multiple candidates through parallel and interactive search. In the search process, 'selection' is performed using 'survival of the fittest' *a posteriori* learning. To obtain the next 'generation' of possible solutions, some parameter values are exchanged between two candidates (by an operation called 'crossover') and new values introduced (by an operation called 'mutation'). This way, the evolutionary technique makes use of past trial information in a similarly intelligent manner to the human designer.

The EA based optimal designs can start from the designer's existing design database or from an initial generation of candidate designs obtained randomly. A number of finally evolved top-performing candidates will represent several automatically optimized digital prototypes.

Geometric design



3D curves — Example 01



3D curves — Example 02

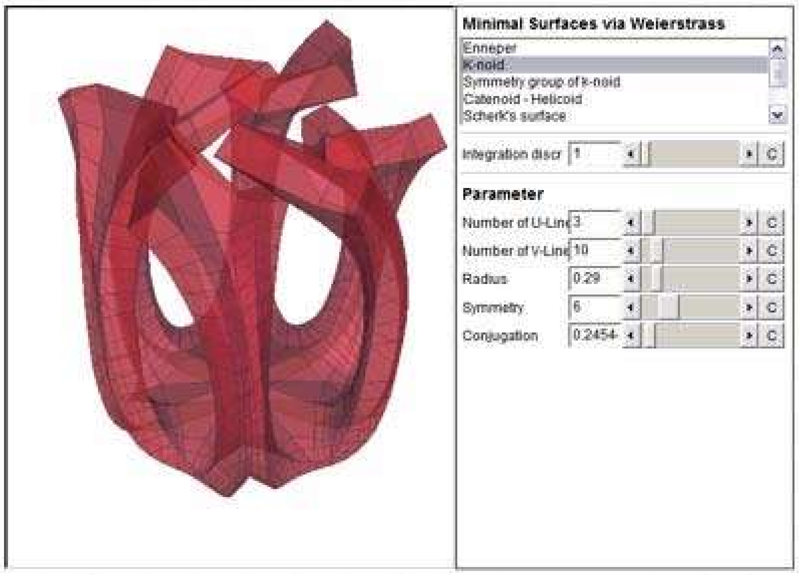
Geometric design (GD), also known as **geometric modelling**, is a branch of computational geometry. It deals with the construction and representation of free-form curves, surfaces, or volumes. Core problems are curve and surface modelling and representation. GD studies especially the construction and manipulation of curves and surfaces given by a set of points using polynomial, rational, piecewise polynomial, or piecewise rational methods. The most important instruments here are parametric curves and parametric surfaces, such as Bezier curves, spline curves and surfaces. An important non-parametric approach is the level set method.

Application areas include shipbuilding, aircraft, and automotive industries, as well as architectural design. The modern ubiquity and power of computers means that even perfume bottles and shampoo dispensers are designed using techniques unheard of by shipbuilders of 1960s.

Geometric models can be built for objects of any dimension in any geometric space. Both 2D and 3D geometric models are extensively used in computer graphics. 2D models are important in computer typography and technical drawing. 3D models are central to computer-aided design and manufacturing, and many applied technical fields such as geology and medical image processing.

Geometric models are usually distinguished from procedural and object-oriented models, which define the shape implicitly by an algorithm. They are also contrasted with digital images and volumetric models; and with implicit mathematical models such as the zero set of an arbitrary polynomial. However, the distinction is often blurred: for instance, geometric shapes can be represented by objects; a digital image can be interpreted as a collection of colored squares; and geometric shapes such as circles are defined by implicit mathematical equations. Also, the modeling of fractal objects often requires a combination of geometric and procedural techniques.

Geometric problems originating in architecture can lead to interesting research and results in geometry processing, computer-aided geometric design, and discrete differential geometry.



Integration discr 1

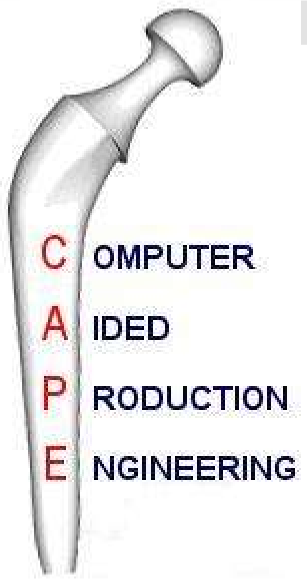
Number of U-Line 3
Number of V-Line 10
Radius 0.29
Symmetry 6
Conjugation 0.2454

K-noid based form, JavaView

Chapter-6

Computer-aided Production Engineering and Boundary Representation

Computer-aided production engineering



CAPE International Conference logo

Computer-aided production engineering (CAPE) is a relatively new and significant branch of engineering. Global manufacturing has changed the environment in which goods are produced. Meanwhile, the rapid development of electronics and communication technologies has required design and manufacturing to keep pace.

Description of CAPE

CAPE is seen as a new type of computer-aided engineering environment which will improve the productivity of manufacturing/industrial engineers. This environment would be used by engineers to design and implement future manufacturing systems and subsystems. Work is currently underway at the United States National Institute of Standards and Technology (NIST) on CAPE systems. The NIST project is aimed at advancing the development of software environments and tools for the design and engineering of manufacturing systems.

CAPE and the Future of Manufacturing

The future of manufacturing will be determined by the efficiency with which it can incorporate new technologies. The current process in engineering manufacturing systems is often ad hoc, with computerized tools being used on a limited basis. Given the costs and resources involved in the construction and operation of manufacturing systems, the engineering process must be made more efficient. New computing environments for engineering manufacturing systems could help achieve that objective.

Why is CAPE important? In much the same way that product designers need computer-aided design systems, manufacturing and industrial engineers need sophisticated computing capabilities to solve complex problems and manage the vast data associated with the design of a manufacturing system.

In order to solve these complex problems and manage design data, computerized tools must be used in the application of scientific and engineering methods to the problem of the design and implementation of manufacturing systems. Engineers must address the entire factory as a system and the interactions of that system with its surrounding environment. Components of a factory system include:

- the physical plant housing the manufacturing facility;
- the production facilities which perform the manufacturing operations;
- the technologies used in the production facility;
- the work centers/stations, machinery, equipment, tools, and materials which comprise or are used by the production facilities;
- the various support facilities;
- the relationship between the factory and its environment.

CAPE must not only be concerned with the initial design and engineering of the factory, it must also address enhancements over time. CAPE should support standard engineering methods and problem-solving techniques, automate mundane tasks, and provide reference data to support the decision-making process.

The environment should be designed to help engineers become more productive and effective in their work. This would be implemented on personal computers or engineering workstations which have been configured with appropriate peripheral devices. Engineering tool developers will have to integrate the functions and data used by a number of different disciplines, for example:

- manufacturing, industrial and plant engineering;
- materials processing and quality engineering;

- environmental engineering,
- mathematical modeling/simulation, statistical process control and computer science,
- economic and cost analysis and management science,

Many of the methods, formulas, and data associated with these technical areas currently exist only in engineering handbooks. Although some computerized tools are available, they are often very specialized, difficult to use, and do not share information or work together. Engineering tools built by different vendors must be made compatible through open systems architectures and interface standards.

What CAPE will look like

CAPE will be based upon computer systems providing an integrated set of design and engineering tools. These software tools will be used by a company's manufacturing engineers to continuously improve its production systems. They will maintain information about manufacturing resources, enhance production capabilities, and develop new facilities and systems. Engineers working on different workstations will share information through a common database.

Using CAPE, an engineering team will prepare detailed plans and working models for an entire factory in a matter of days. Alternative solutions to production problems could be quickly developed and evaluated. This would be a significant improvement over current manual methods which may require weeks or months of intensive activity.

To achieve this goal, a new set of engineering tools are needed. Examples of functions which should be supported include:

- identification of product specifications and production requirements;
- producibility analysis for products and modification of product designs to address manufacturability issues and management, scheduling and tracking of projects;
- modeling and specification of manufacturing processes and plant layout and facilities planning;
- consideration of various economic/cost tradeoffs of different manufacturing processes, systems, tools, and materials;
- analysis supporting selection of systems/vendors and procurement of manufacturing equipment and support systems;
- task and work place design;
- compliance with various regulations, specifications, and standards, and control of hazardous materials.

The tools implementing these functions must be highly automated and integrated; and will need to provide quick access to a wide range of data. This data must be maintained in a format that is accessible and usable by the engineering tools. Some examples of the information that might be contained in these electronic libraries include:

- production process models and data and generic manufacturing systems configurations;
- machinery and equipment specifications, and vendor catalogs;
- recommended methods, practices, algorithms, etc., and benchmarking data;
- typical plant/system layouts,
- cost estimation models, labor rates, other cost data and budget templates,
- time standards, industrial standards, project plans, and laws/government regulations.

These on-line libraries would allow engineers to quickly develop solutions based upon the work of others.

Another critical aspect of this engineering environment is affordability, which can best be achieved by designing an environment that can be constructed from low cost "off-the-shelf" commercial products, rather than custombuilt computer hardware and software. The basic engineering environment must be affordable. For both cost and technical reasons, it must be designed to be able to support incremental upgrades. Incremental upgrades would allow companies to add capabilities as they are needed. Commercial software products must be easy to install and integrate with other software already in use. These capabilities exist to a limited extent in some general purpose commercial software today, e.g., word processors, databases, spreadsheets.

Technical Concerns

Many technical issues must be considered in the design and development of new engineering tools for CAPE. These issues include:

- required functionality of the tools themselves;
- formalization and refinement of engineering methods;
- development of on-line technical reference libraries, and user engineering and graphics visualization;
- user engineering and graphics visualization techniques;
- system connectivity and information sharing, and integration standards for the computing environment;
- incorporation of intelligent behavior in the tools.

There are three critical elements to be addressed: creating a common manufacturing systems information model; using an engineering life cycle approach; and developing a software tool integration framework.

Resolution of these elements will help ensure that independently developed systems will be able to work together. The common information model should identify the elements of the manufacturing system and their relationships to each other; the functions or processes performed by each element; the tools, materials, and information required to perform those functions; and measures of effectiveness for the model and its component elements.

There have been many efforts over the years to develop information models for different aspects of manufacturing, but no known existing model fully meets the needs of a CAPE ernviroment. Therefore, a life cycle approach is needed to identify the different processes that a CAPE environment must support, and must define all phases of a manufacturing system or subsystem's existence. Some of the major phases which may be included in a system life cycle approach are, requirements identification; system design specification; vendor selection; system development and upgrades; installation, testing, and training; and benchmarking of production operations.

Management, coordination, and administration functions need to be performed during each phase of the life cycle. Phases may be repeated over time as a system is upgraded or re-engineered to meet changing needs or incorporate new technologies.

A software tool integration framework should specify how the tools could be independently designed and developed. The framework would define how CAPE tools would deal with common services, interact with each other and coordinate problem solving activities. Although some existing software products and standards currently address the common services issue, the problem of tool interaction remains largely unsolved. The problem of tool interaction is not limited to the domain of computer-aided manufacturing systems engineering--it is pervasive across the software industry.

CAPE's current state

An initial CAPE environment has been established from commercial off-the-shelf (COTS) software packages. This new environment is being used to demonstrate commercially available tools to perform CAPE functions, to develop a better understanding and define functional requirements for individual engineering tools and the overall environment, and to identify the integration issues which must be addressed to implement compatible environments in the future.

Several engineering demonstrations using COTS tools are under development. These demonstrations are designed to illustrate the various types of functions that must be performed in engineering a manufacturing system.

Functions supported by the current COTS environment include: system specification/diagramming, process flowcharting, information modeling, computer-aided design of products, plant layout, material flow analysis, ergonomic workplace design, mathematical modeling, statistical analysis, line balancing, manufacturing simulation, investment analysis, project management, knowledge-based system development, spreadsheets, document preparation, user interface development, document illustration, forms and database management.

Boundary representation

In solid modeling and computer-aided design, **boundary representation**—often abbreviated as **B-rep** or **BREP**—is a method for representing shapes using the limits. A solid is represented as a collection of connected surface elements, the boundary between solid and non-solid.

Overview

Boundary representation models are composed of two parts: topology and geometry (surfaces, curves and points). The main topological items are: *faces*, *edges* and *vertices*. A face is a bounded portion of a surface; an edge is a bounded piece of a curve and a vertex lies at a point. Other elements are the *shell* (a set of connected faces), the *loop* (a circuit of edges bounding a face) and *loop-edge links* (also known as *winged edge links* or *half-edges*) which are used to create the edge circuits. The edges are like the edges of a table, bounding a surface portion.

History

The basic method for BREP was developed independently in the early 1970s by both Ian Braid in Cambridge (for CAD) and Baumgart in America (for computer vision). Braid continued his work with the research solid modeller BUILD which was the forerunner of many research and commercial solid modelling systems. Braid worked on the commercial systems ROMULUS, the forerunner of Parasolid, and on ACIS. Parasolid and ACIS are the basis for many of today's commercial CAD systems.

Following Braid's work for solids, a Swedish team led by Professor Torsten Kjellberg, developed the philosophy and methods for working with hybrid models, wire-frames, sheet objects and volumetric models during the early 1980s. In Finland, Marti Mäntylä produced a solid modelling system called GWB. In the USA Eastman and Weiler were also working on Boundary Representation and in Japan Professor Kimura and his team at Tokyo University also produced their own B-rep modelling system.

Compared to the constructive solid geometry (CSG) representation, which uses only primitive objects and Boolean operations to combine them, boundary representation is more flexible and has a much richer operation set. This makes boundary representation a more appropriate choice for CAD systems. CSG was used initially by several commercial systems because it was easier to implement. The advent of reliable commercial B-rep kernel systems like Parasolid and ACIS, mentioned above, has led to widespread adoption of B-rep for CAD. As well as the Boolean operations, B-rep has extrusion (or sweeping), chamfer, blending, drafting, shelling, tweaking and other operations which make use of these.

Boundary representation is essentially a local representation connecting faces, edges and vertices. An extension of this was to group sub-elements of the shape into logical units called *geometric features*, or simply *features*. Pioneering work was done by Kyprianou in Cambridge also using the BUILD system and continued and extended by Jared and others. Features are the basis of many other developments, allowing high-level "geometric reasoning" about shape for comparison, process-planning, manufacturing, etc.

Boundary representation has also been extended to allow special, non-solid model types called **non-manifold models**. As described by Braid, normal solids found in nature have the property that, at every point on the boundary, a small enough sphere around the point is divided into two pieces, one inside and one outside the object. Non-manifold models break this rule. An important sub-class of non-manifold models are sheet objects which are used to represent thin-plate objects and integrate surface modelling into a solid modelling environment.

Standardization

In the world of data-exchange, STEP, the **Standard for the Exchange of Product Model data** also defines some data models for boundary representations. The common generic topological and geometric models are defined in ISO 10303-42 **Geometric and topological representation**. The following Application Integrated Resources (AICs) define boundary models that are constraints of the generic geometric and topological capabilities:

- ISO 10303-511 *Topologically bounded surface*, definition of an **advanced face**, that is a bounded surface where the surface is of type elementary (plane, cylindrical, conical, spherical or toroidal), or a swept surface, or b spline surface. The boundaries are defined by lines, conics, polylines, surface curves, or b spline curves
- ISO 10303-514 *Advanced boundary representation*, a solid defining a volume with possible voids that is composed by advanced faces
- ISO 10303-509 *Manifold surface*, a non intersecting area in 3D that is composed by advanced faces
- ISO 10303-521 *Manifold subsurface*, a sub-area out of a manifold surface
- ISO 10303-508 *Non-manifold surface*, any kind of advanced face arrangement
- ISO 10303-513 *Elementary boundary representation* similar to ISO 10303-514, but restricted to the elementary surfaces only
- ISO 10303-512 *Faceted boundary representation* a simplified surface model constructed by planar surfaces only

Chapter-7

Freeform Surface Modeling and DraftSight

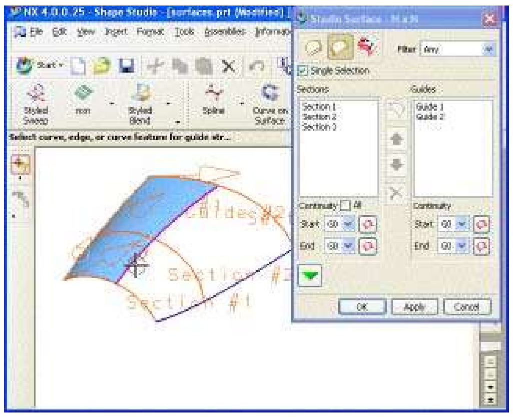
Freeform surface modelling

Freeform surface modelling is the art of engineering Freeform Surfaces with a CAD or CAID system.

Introduction

The technology has encompassed two main fields. Either creating aesthetic (Class A surfaces) that also perform a function; for example, car bodies and consumer product outer forms, or technical surfaces for components such as gas turbine blades and other fluid dynamic engineering components.

CAD software packages use two basic methods for the creation of surfaces. The first begins with construction curves (splines) from which the 3D surface is then swept (section along guide rail) or meshed (lofted) through.



NX 4.0.0.25 - Shape Studio - [surfaces.prt (Modified)]

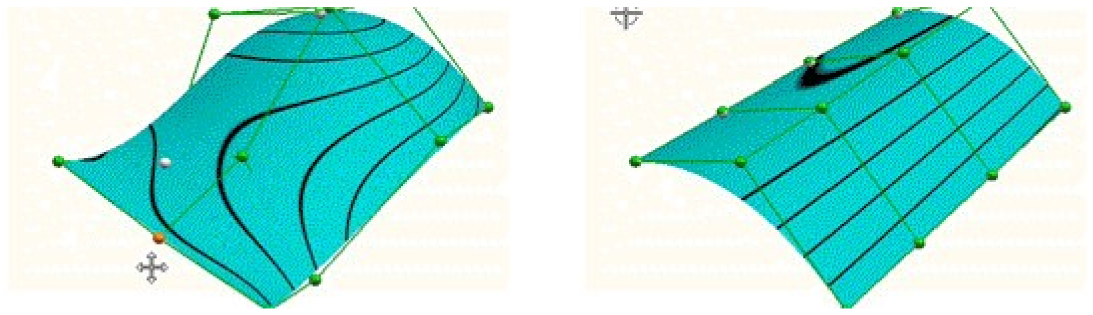
Select curves, edge, or curve feature for guide str...

Sections: Section 1, Section 2, Section 3

Guides: Guide 1, Guide 2

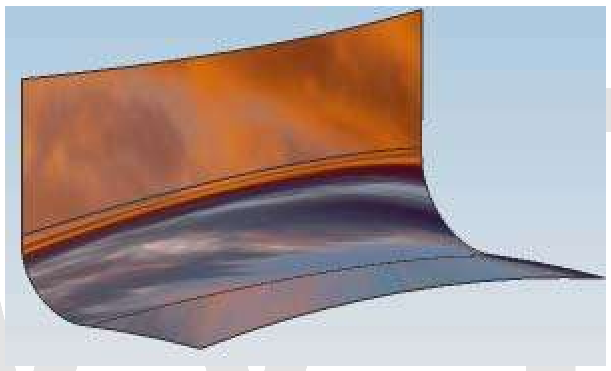
A surface being created from curves.

The second method is direct creation of the surface with manipulation of the surface poles/control points.



Surface edit by poles

From these initially created surfaces, other surfaces are constructed using either derived methods such as offset or angled extensions from surfaces; or via bridging and blending between groups of surfaces.

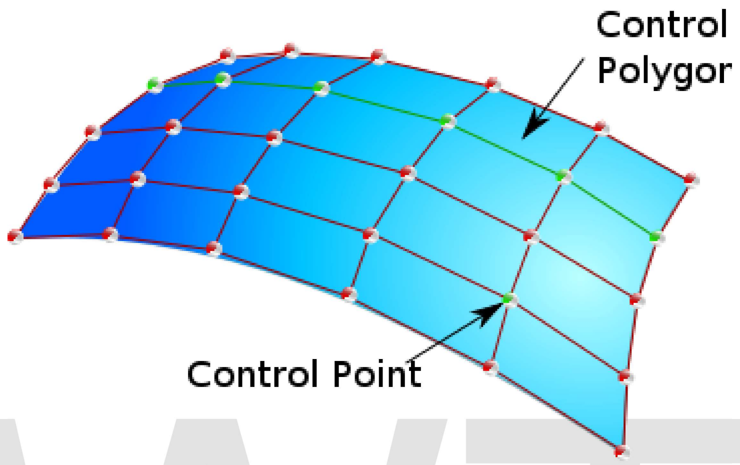


Variable smooth blend between surfaces.

Surfaces

Freeform surface, or **freeform surfacing**, is used in CAD and other computer graphics software to describe the skin of a 3D geometric element. Freeform surfaces do not have rigid radial dimensions, unlike regular surfaces such as planes, cylinders and conic surfaces. They are used to describe forms such as turbine blades, car bodies and boat hulls. Initially developed for the automotive and aerospace industries, freeform surfacing is now widely used in all engineering design disciplines from consumer goods products to ships. Most systems today use nonuniform rational B-spline (NURBS) mathematics to describe the surface forms; however, there are other methods such as Gorden surfaces or Coons surfaces .

The forms of freeform surfaces (and curves) are not stored or defined in CAD software in terms of polynomial equations, but by their poles, degree, and number of patches (segments with spline curves). The degree of a surface determines its mathematical properties, and can be seen as representing the shape by a polynomial with variables to the power of the degree value. For example, a surface with a degree of 1 would be a flat cross section surface. A surface with degree 2 would be curved in one direction, while a degree 3 surface could (but does not necessarily) change once from concave to convex curvature. Some CAD systems use the term *order* instead of *degree*. The order of a polynomial is one greater than the degree, and gives the number of coefficients rather than the greatest exponent.



Example surface pole map

The poles (sometimes known as *control points*) of a surface define its shape. The natural surface edges are defined by the positions of the first and last poles. (Note that a surface can have trimmed boundaries.) The intermediate poles act like magnets drawing the surface in their direction. The surface does not, however, go through these points. The second and third poles as well as defining shape, respectively determine the start and tangent angles and the curvature. In a single patch surface (Bézier surface), there is one more pole than the degree values of the surface. Surface patches can be merged into a single NURBS surface; at these points are knot lines. The number of knots will determine the influence of the poles on either side and how smooth the transition is. The smoothness between patches, known as *continuity*, is often referred to in terms of a *C value*:

- C0: just touching, could have a nick
- C1: tangent, but could have sudden change in curvature
- C2: the patches are curvature continuous to one another

Two more important aspects are the U and V parameters. These are values on the surface ranging from 0 to 1, used in the mathematical definition of the surface and for defining paths on the surface: for example, a trimmed boundary edge. Note that they are not proportionally spaced along the surface. A curve of constant U or constant V is known as an isoperimetric curve, or U (V) line. In CAD systems, surfaces are often displayed with their poles of constant U or constant V values connected together by lines; these are known as *control polygons*.

Modelling

When defining a form, an important factor is the continuity between surfaces - how smoothly they connect to one another.

One example of where surfacing excels is automotive body panels. Just blending two curved areas of the panel with different radii of curvature together, maintaining tangential continuity (meaning that the blended surface doesn't change direction suddenly, but smoothly) won't be enough. They need to have a continuous rate of curvature change between the two sections, or else their reflections will appear disconnected.

The continuity is defined using the terms

- G0 – position (touching)
- G1 – tangent (angle)
- G2 – curvature (radius)
- G3 – acceleration (rate of change of curvature)

To achieve a high quality NURBS or Bezier surface, degrees of 5 or greater are generally used. Depending on the product and production process, different levels of accuracy are used but tolerances usually range from 0.02 mm to .001 mm (for example, in the fairing of BIW concept surfaces to production surface). For ship building, this need not be so tight, but for precision gears and medical devices it is much finer.

History of Terms

The term lofting originally came from the shipbuilding industry where loftsmen worked on "barn loft" type structures to create the keel and bulkhead forms out of wood. This was then passed on to the aircraft then automotive industries who also required streamline shapes.

The term spline also has nautical origins coming from East Anglian dialect word for a thin long strip of wood (probably from old English and Germanic word splint).

Freeform Surface Modelling Software

- Catia
- Cobalt (Ashlar-Vellum)
- form•Z
- ICEM Surf
- Imageware
- NX (Unigraphics)
- ProEngineer
- Rhinoceros 3D
- FreeForm Modeling Plus from SensAble Technologies
- solidThinking

- Autodesk Inventor
- Solidworks
- Alias StudioTools
- FreeSHIP (FreeSHIP)
- GenesisIOD (GenesisIOD)
- OmniCAD (OmniCAD)
- ProEngineer ISDX ()
- Thinkdesign (Thinkdesign)
- MicroStation (Bentley Systems Inc)
- Shark FX (Punch!)
- **Moi** Moment of Inspiration 3D modeling for designers and artists (Moi3d)

DraftSight

DraftSight is a free 2D CAD (Computer Aided Design or Computer Aided Drafting) product for engineers, architects, designers, draftspeople, students and educators. The product was developed by Dassault Systèmes and lets users create, edit and view DWG and DXF files.

DWG files contain the binary data for CAD design and it is the drawing format for many CAD programs. DWG is a long-time abbreviation for "Drawing." A DXF, or Drawing Exchange File, is used to convert CAD files into a generic format that can be read by another CAD software product.

DraftSight competes against more than three dozen 2D or 2D/3D hybrid products on the market. General availability of DraftSight for Windows was released in February 2011 and more than 400,000 users have downloaded the product.

Languages: English, Simplified Chinese, Traditional Chinese, Czech, French, German, Italian, Spanish, Japanese, Korean, Polish, Brazilian Portuguese, Turkish and Russian.

Portability

DraftSight loads quickly and has a file size of 45 MB. The activated DraftSight application folder takes up 160 MB.

File formats

DraftSight allows users to access legacy DWG/DXF files, regardless of which CAD software was originally used to create them. According to industry estimates, there are more than three billion DWG legacy files in existence.

Platforms/operating systems

DraftSight runs on Microsoft Windows XP,Windows Vista, Windows 7 (general release), Mac OS and Linux OS (Mac and Linux are currently in Beta).

DraftSight functions

- Read and write support for DWG/DXF files
- Save DWG/DXF files back to previous versions
- Create binary or ASCII DXF files
- Attach image files (.bmp, .gif, .jpg, .jpeg, .png, .tif, & .tiff)
- Attach external reference drawings
- Print to file (.plt, .jpg, .pdf, .png, & .svg)
- Save as .wmf, .jpeg, .pdf, .png, .sld, .svg, .tif, & .stl file formats
- Multi-page PDF creation
- Publish to eDrawings or Drawings Now

Release History

- Initial Beta Release for Windows: June 2010
- Mac Beta I Release: September 2010
- General Release for Windows: February 2011
- Linux Beta Release: March 2011
- Mac Beta II Release: March 2011

Features

DraftSight was designed for professional CAD users and includes the following features:

- Cartesian coordinate system
- Command line input
- Traditional toolbars and menus
- Wheel-mouse pan & zoom
- Blocks & reference files
- Layers & layer manager
- Polygonal ViewPorts
- ViewPort locking
- Freeze, lock or turn off layers per ViewPort
- Background masks for notes
- Property manager
- View proxy objects
- Dynamic pan & zoom
- Command aliases
- Menu files
- CTB and STB print style tables
- SHX and TTF fonts

- LineStyle files
- Hatch pattern files
- Templates

Online user community

Dassault Systèmes hosts an open, on-line SwYm (See What You Mean) community for DraftSight users to ask questions, express opinions, collaborate, solve problems and share ideas. According to Machine Design magazine, engineers are increasingly turning to industry-specific social networking sites to collaborate on projects. The community also holds an abundance of training resources.

Tutorials/training

Dassault Systèmes offers free video DraftSight training resources. Additional training resources are available at no cost in the online DraftSight SwYm Community.

Education

- DraftSight is free for individual students and educators
- DraftSight Premium for Education Classroom Pack and Campus Pack includes telephone and email support, access to DraftSight APIs, network licensing and curriculum and additional online training through No-Cost Community Support.

Technical support

- No-cost community support
- DraftSight Premium Pack: Fee-based telephone and email technical support with network licensing and access to DraftSight APIs

Chapter-8

Grasshopper 3d

Grasshopper

Developer(s)	Robert McNeel & Associates
Stable release	1.0 Beta / December 11, 2010
Operating system	Windows (2000/XP/Vista/7)
Type	Visual Programming
License	Proprietary

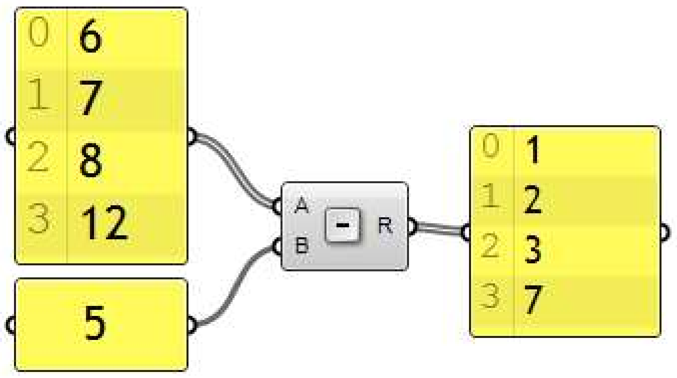
Grasshopper™ is a visual programming language developed by David Rutten at Robert McNeel & Associates. Grasshopper runs within the Rhinoceros 3D CAD application. Grasshopper is visual programming language. Programs are created by dragging components onto a canvas. The outputs to these components are then connected to the inputs of subsequent components. Grasshopper is used mainly to building generative algorithms. Many of Grasshoppers components create 3D geometry. Programs may also contain other types of algorithms including numeric, textual, audio-visual and haptic applications.

“Popular among students and professionals, McNeel Associate’s Rhino modelling tool is endemic in the architectural design world. The new Grasshopper environment provides an intuitive way to explore designs without having to learn to script."

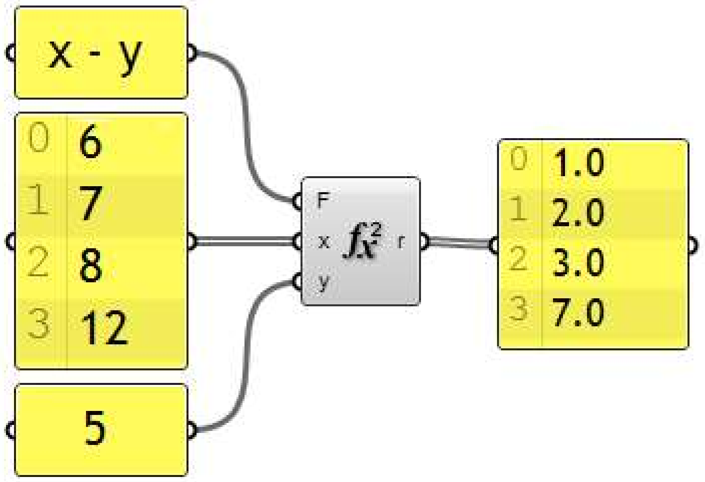
Grasshopper is currently in beta phase development and is offered as a free download without expiration date, although a licensed copy or un-expired trial of Rhinoceros 4.0 or higher is required to run the software.

Node based editor

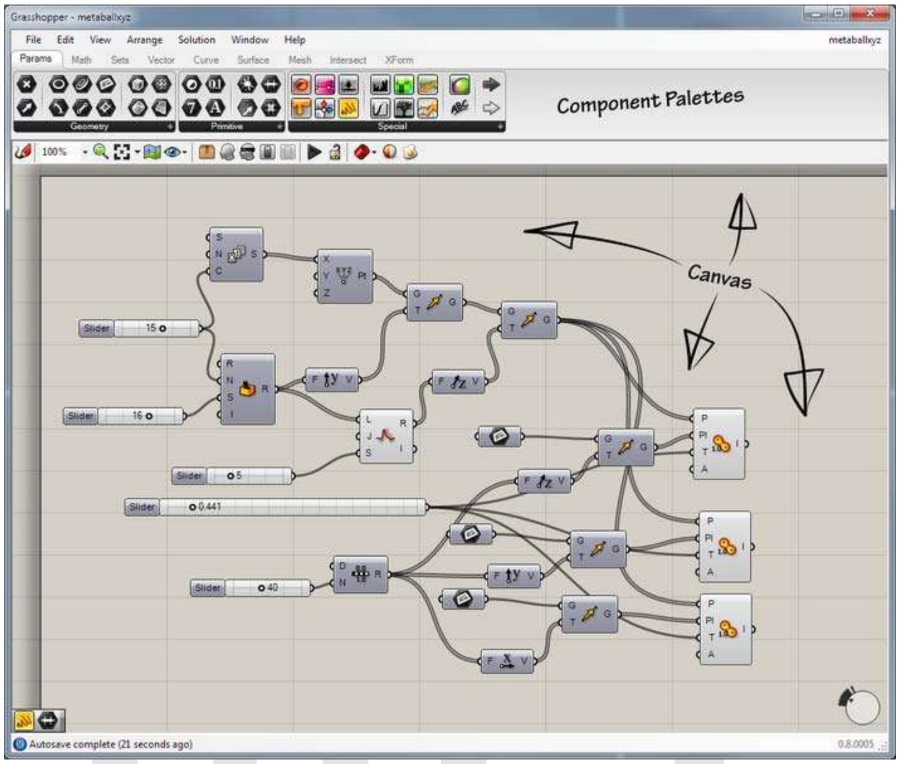
The main interface for algorithm design in Grasshopper is the node-based editor. Data is passed from component to component via connecting wires which always connect an output grip with an input grip. Data can either be defined locally as a constant, or it can be imported from the Rhino document or a file on the computer. Data is always stored in parameters, which can either be free-floating or attached to a component as input and outputs objects.



In the image above we see three free-floating parameters that are hooked up to a subtraction component. The two yellow boxes on the left both define a set of numeric constants. The top-most panel contains four integers (6, 7, 8 and 12) while the bottom-most panel contains only a single value. These floating parameters supply the subtraction component with input data, which results in four output values (6-5=1, 7-5=2, 8-5=3 and 12-5=7). The same result can be achieved using textual expressions and an evaluator component. In this fashion Grasshopper allows users to combine both visual and textual programming within the same environment.

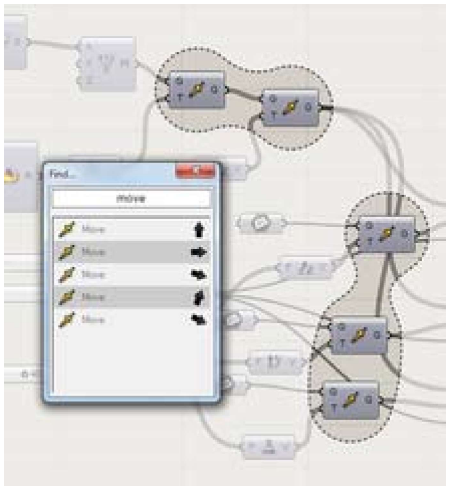
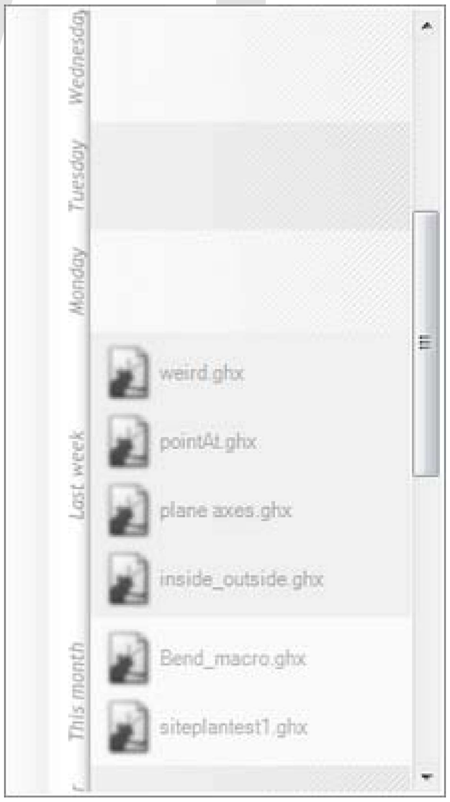


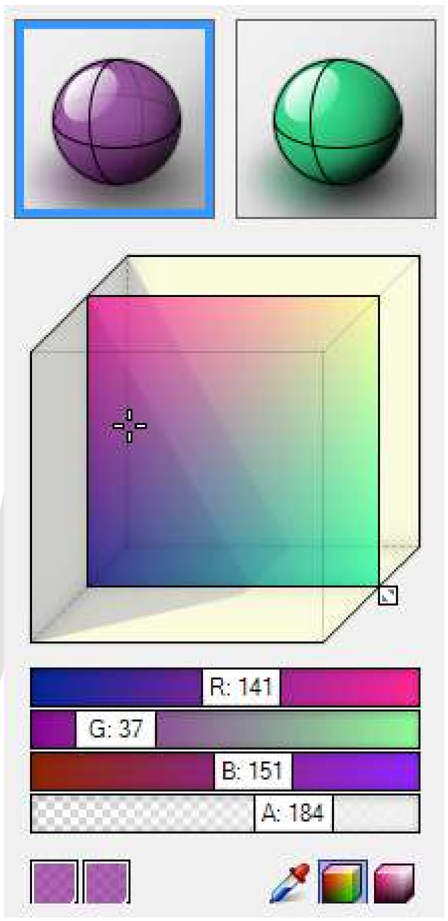
User Interface


Grasshopper - metaballxyz
File Edit View Arrange Solution Window Help
Params Math Sets Vector Curve Surface Mesh Intersect XForm
Geometry
Primitive
Special
Component Palettes
Canvas
Slider
Autosave complete (21 seconds ago)

Grasshopper features a fairly advanced GUI with a lot of features that are only rarely found in production software. It is not known however whether these elements improve or impede effective usage. The main window consist mainly of the component 'palettes' and the 'canvas', apart from standard Windows GUI elements such as the title bar, the menu and the status bar. Since Grasshopper is a plug-in to another windowed application, the layout of the main window is kept minimal. Below is a list of some of the more rare GUI elements.

GUI Element	Description	Image
MDI	The multi-document-interface menu contains small preview images of the documents in question	

Find	The Find dialog provides both textual and spatial feedback regarding the search hits. Objects are highlighted on the canvas by a Metaball outline and little arrows on the dialog point towards the location in screen space of the associated component.	
Prediction	A Markov-chain database is maintained of all the add actions of the user. This enables Grasshopper to (eventually) predict with a reasonable level of accuracy which command(s) will be called upon next. These commands are placed in an easy access toolstrip on the canvas.	
MRU	The Most-Recently-Used menu maintains not just a large collection of previously used documents, but also checks each file for availability. Files which are no longer present on the system are shown greyed out. In addition, the history of the MRU is categorized into distinct period such as "Just Now", "Today", "Thursday" and "Last Week", making it much easier to find a needed document.	 Wednesday Tuesday Monday Last week weird.ghx pointAt.ghx plane axes.ghx inside_outside.ghx This month Bend_macro.ghx siteplantest1.ghx

ZUI	Some objects drawn on the canvas adjust their display based on the zoom-level. This results in a cleaner and faster view when zoomed out and useful additional information when zoomed in.	
Color	The default color picker supports and displays transparency.	 R: 141 G: 37 B: 151 A: 184
Search	The search function which is used to locate specific components and data types uses both exact and fuzzy comparisons. If the search result list contains too few hits, Levenshtein distance comparisons are added.	

Chapter-9

ISO 13567

ISO 13567 is an international Computer-aided design (CAD) layer standard.

Standard parts

The standard is divided in three parts:

- ISO 13567-1:1998
    - Technical product documentation — Organization and naming of layers for CAD — Part 1: Overview and principles
- ISO 13567-2:1998
    - Technical product documentation — Organization and naming of layers for CAD — Part 2: Concepts, format and codes used in construction documentation
- ISO/TR 13567-3:1999
    - Technical product documentation — Organization and naming of layers for CAD — Part 3: Application of ISO 13567-1 and ISO 13567-2

Standard has been developed by Technical Committee TC 10 (Technical product documentation) Subcommittee SC 8 (Construction documentation); Refer. ICS: 01.110; 35.240.10.

Structure of layer names

CAD layer names are structured as a series of mandatory and optional fixed length fields, composed as a continuous alpha-numerical text string.

Mandatory fields

Agent responsible <clause 6.1>

(2 characters, indicating the person or organisation responsible for the layer information

```
-- manufacturer,
A- architect
A2 architect#2 on the same project
B- building surveyors
C- civil engineers
E- electrical engineers
F- facility engineers
G- GIS engineers and land surveyors
H- heating and ventilating engineers
I- interior designers
L- landscape architects
Q- quantity surveyors
S- structural engineers
T- town and country planners
W- contractors
X- sub-contractors
Y- specialist designers
```

Element <cl. 6.2>

(6 characters, indicating the functional parts of construction works or structure): follows a classification system like SfB building codes or Uniclass codes;

Presentation <cl. 6.3>

(2 characters, related to the information graphical presentation)

```
-- whole model and drawing page,
E- element graphics (Model space)
T- text (M)
H- hatching (M)
D- dimensions (M)
J- section/detail marks (M)
K- revision marks (M)
G- Grid graphic and dimension (M)
U- user red lines & construction lines (M)
B- Border (Page/Paper space)
V- text, title, notes (P)
I- tabular information, legends, schedules, tables/query (P)
-1 language#1 or pen thickness#1 or text height#1
```

Optional fields

Status <cl. 7.1>

(1 character, status of the physical part, ISO code)

```
- whole, no subdivision
N new part
E existing to remain
R to be removed
T temporary
O to be moved original position
F to be moved final position
```

Sector <cl. 7.2>

(4 characters, physical subdivision of construction work, recommended to use ISO 4157-2/3 codes)

```
---- whole project, all levels all blocks
00-- ground floor
02-- 2nd floor
-1-- basement
SA-- section AA
EA-- elevation A
EB-- elevation B
--B1 block 1
--A- zone A
01B1 1st floor block#1
-2-- 2nd basement level
01A- storey 01 zone A
```

Phase <cl. 7.3>

(1 character, time or logical subdivision of work)

```
- whole duration
P pre-design/preliminary
D design
R procurement
C construction
O post-construction
1 phase#1 (pre-design)
2 phase#2
3 phase#3 (licence design)
```

Projection <cl. 7.4>

(1 character, work multiple views differentiation, ISO code)

```
- all
0 plan
1 elevation
2 section
3 3D model
```

Scale <cl. 7.5>

(1 character, classification of layer information by the scale of final drawing, ISO code)

```
- all
A 1:1
B 1:5
C 1:10
D 1:20
E 1:50
F 1:100
G 1:200
H 1:500
I 1:1000
J 1:2000
K 1:5000
```

```
1 1:1-5
2 1:5-20
3 1:20-50
4 1:20-100
5 1:50-200
6 1:200-500
7 1:500-1000
```

Work Package <cl. 7.6>

(2 characters, materials and/or work sections)

```
RC reinforced concrete
SS structural steel
23 phase#23
```

User Defined <cl. 7.7>

(unlimited, additional subdivision or help plain description): description or subdivision.

Special characters

```
- (hyphenation) not used field
_ (underscore) relative to all possible characters
```

Applications

Resulting length

In practice, application of ISO 13567 fields can lead to short names (only mandatory fields), or much longer names (use of some or all optional fields in complex projects).

Short name samples

A-B374--T-
	agent Architect, element Roof window in SfB, presentation text
A-B374--E-
	agent Architect, element Roof window in SfB, presentation graphic element
Other

- A-374---T-
- A-24--__D-

Long name samples

A-37420-T2N01B113B23pro
	agent Architect, element Roof Window in SfB, presentation Text#2, New part, floor 01, block B1, *phase 1, projection 3D, scale 1:5(B), work package 23 and user definition "pro"
A-G25---D-R
	agent Architect, element wall in Uniclass, presentation dimensions, status Existing to be removed
Other

- A-2441__D-N01AB1
- A-37420-T2N01B113B23pro
- T-811---E-N----30F--DESCRIPTION_OF_LAYER
- E-63----E-N----30G--ELECTRICAL_EQUIPMENT
- A1645---Z-O----1-A72DESCRIPTION_OF_LAYER
- A-DUCTS-E-N02C------P281
- F-5821ABE-N-I--13C23USER
- A-144001M-----

List of layers in a standard architectural drawing

Agent is Architect, using ISO 13567 and UniClass or CI/SfB Classification System.

Generic layers

```
A-------E-: all elements
A-------T-: text
A-------H-: hatchings
A-------D-: dimensions
A-------J-: section/detail marks
A-------K-: revision marks
A-------G-: grid (graphics and dimensions)
A-------U-: user (red and construction lines)
A-------B-: border (border lines/frame and other graphics) in paper
space
A-------V-: text (title and notes) in paper space
A-------I-: tabular information (legends, schedules and tables) in
paper space
A-------E-N: new work elements
A-------E-E: elements existing to remain
A-------E-R: elements existing to be removed
A-------E1: line thickness 0,13 mm
A-------E2: line thickness 0,18 mm
A-------E3: line thickness 0,25 mm
A-------E4: line thickness 0,35 mm
A-------E5: line thickness 0,50 mm
A-------E6: line thickness 0,70 mm
A-------T1: text height 1,8 mm
A-------T2: text height 2,5 mm
A-------T3: text height 3,5 mm
A-------T4: text height 5,0 mm
A-------T5: text height 7,0 mm
A-------T6: text height 10,0 mm
```

Architectural layers using UniClass

Using UniClass Table G and Table H (exceptionally F and J tables).

```
A-F1----E-: zones (blocks, wings, floors, departments)
A-F2----E-: rooms
A-F2----T-: room numbers and text
A-F3----E-: circulation (foyers, halls, stairs, corridors, gangways)
A-F9----E-: building space analysed (areas)
```

```
A-F911--E-: usable area (ISO 9836)
A-F912--E-: circulation area (ISO 9836)
A-F913--E-: services area (ISO 9836)
A-F914--E-: structural element area (ISO 9836)
A-F919--E-: gross area (ISO 9836)
A-G-----E-: building
A-G11---E-: site clearance
A-G12---E-: ground contouring
A-G2----E-: building fabric
A-G21---E-: foundations
A-G21---E5: foundations in section
A-G22---E-: floors, slabs
A-G22---E5: floors, slabs in section
A-G23---E-: stairs (incl. balaustrades), ramps
A-G23---E5: stairs and ramps in section
A-G24---E-: roofs
A-G24---E5: roofs in section
A-G25---E-: walls
A-G25---E5: walls in section
A-G251--E-: external walls
A-G252--E-: internal walls
A-G26---E-: structural frame, columns, beams, bracing
A-G26---E5: structural frame, columns, beams, bracing in section
A-G321--E-: windows
A-G322--E-: doors
A-G33---E-: internal finishes (on floor, ceilings, walls)
A-G331--E-: floor finishes
A-G332--E-: ceilings/soffit finishes
A-G333--E-: internal wall finishes
A-G4----E-: fittings, furnitures, equipments
A-G44---E-: sanitary fittings
A-G50---E-: water supply (water pipes)
A-G501--E-: cold water
A-G502--E-: hot water
A-G51---E-: gas supply
A-G52---E-: heating/ventilation/air conditioning (HVAC) (HVAC
ductworks)
A-G53---E-: electric power (cable runs)
A-G54---E-: lighting fixtures (fittings)
A-G55---E-: communications (radio, TV, telephones, computer networks)
A-G561--E-: lifts
A-G562--E-: escalators
A-G57---E-: protection (security, fire)
A-G581--E-: removal /disposal, drainage
A-G6----E-: energy (heat, electricity) generation, storage and
conversion
A-G7----E-: external site works
A-G71---E-: surface treatment of external site (hard surfaces,
landscaping)
A-G72---E-: enclosures of external site (fence, walls)
A-G74---E-: fittings/furnitures/equipment of external site (manholes)
A-G77---E-: underground drainage in external site (drain runs)
A-H-----E-: civil engineering works (non-building)
A-H1----E-: pavements and landscaping (ground, pavements, etc.)
A-H122--E-: surfacing to pavements and hard landscaping
A-H123--E-: edgework to pavements and hard landscaping
```

```
A-H132--E-: electrical installations (mechanical, lighting, power,
communications)
A-H142--E-: fittings
A-H1422-E-: signs
A-H1424-E-: street furniture
A-H735--E-: drainage (non building)
A-JE0---E-: concrete
A-JF1---E-: block/brick work
A-JF2---E-: stone
A-JG1---E-: metal
A-JG10--E-: structural steel
A-JG2---E-: timber
A-JK3---E-: glass
A-Z001--T-: identification tags, rooms, door numbers
A-Z400--E-: special presentation
A-Z410--E-: cars
A-Z420--E-: people
A-Z430--E-: trees
A-Z900--U-: non-plotting
A-Z920--E-: external reference (Xref)
```

Note: Elements cut by section has been provvisionally indicated as A-****--E5 (reflecting the use of a wider line). An alternative -but longest- notation could be A-****--E-------2 (defining the view as section) or A-****--ES.

Architectural layers using CI/SfB

```
A-100---E-: substructure
A-110---E-: groundwork
A-160---E-: foundations
A-170---E-: pile foundations
A-210---E-: external walls
A-214---E-: external curtain walls
A-220---E-: internal walls
A-226---E-: internal framing & cladding
A-230---E-: floors
A-240---E-: stairs
A-270---E-: roofs
A-28----E-: building frames
A-280---E-: beams and columns
A-281---E-: metal columns
A-282---E-: concrete columns
A-283---E-: metal beams
A-284---E-: concrete beams
A-285---E-: timber beams
A-310---E-: external wall completions
A-314---E-: external windows
A-315---E-: external doors
A-320---E-: internal wall completions
A-324---E-: internal windows
A-325---E-: internal doors
A-330---E-: floor completions
A-340---E-: stairs
A-350---E-: ceilings
A-370---E-: roof completion's
A-374---E-: roof windows
```

```
A-410---E-: external wall finishes
A-420---E-: internal wall finishes
A-430---E-: floor finishes
A-440---E-: stair finishes
A-450---E-: ceiling finishes
A-470---E-: roof finishes
A-5-----E-: services, non electrical
A-500---E-: mechanical
A-52----E-: waste disposal, drainage
A-53----E-: water & liquid supply
A-54----E-: gas supply
A-55----E-: space cooling, refrigeration
A-56----E-: space heating
A-57----E-: ventilation
A-59----E-: parts, accessories to piped, ducted services
A-6-----E-: services, mainly electrical
A-600---E-: electrical
A-61----E-: electrical supply
A-62----E-: power supply
A-63----E-: lighting
A-630---E-: lamps
A-640---E-: communications
A-65----E-: telecommunications
A-660---E-: transports
A-661---E-: lifts
A-68----E-: security protection
A-700---E-: general fittings & furniture
A-71----E-: circulation fitting
A-72----E-: rest, work fittings
A-73----E-: kitchens, culinary fittings
A-74----E-: sanitary fittings
A-75----E-: cleaning fittings
A-76----E-: storage fittings
A-77----E-: special activity fittings
A-78----E-: loose fittings
A-900---E-: external works
A-910---E-: site information
A-920---E-: survey information
A-930---E-: land drainage/services
A-940---E-: landscaping
A-950---E-: hard surfaces
A-960---E-: utilities
A-970---E-: fences/equipment
A-980---E-: special landscaping
A-990---E-: environmental data
```

Chapter-10

Knowledge-based Engineering

Knowledge-based engineering (KBE) is a discipline with roots in computer-aided design (CAD) and knowledge-based systems but has several definitions and roles depending upon the context. An early role was support tool for a design engineer generally within the context of product design. Success of early KBE prototypes was remarkable; eventually this led to KBE being considered as the basis for generative design with many expectations for hands-off performance where there would be limited human involvement in the design process.

Overview

KBE can be defined as engineering on the basis of electronic knowledge models. Such knowledge models are the result of knowledge modeling that uses knowledge representation techniques to create the computer interpretable models. The knowledge models can be imported in and/or stored in specific engineering applications that enable engineers to specify requirements or create designs on the basis of the knowledge in such models. There are various methods available for the development of knowledge models, most of them are system dependent. An example of a system-independent language for the development machine-readable ontology databases, including support for basic engineering knowledge, is called Gellish English. An example of a CAD-specific system that can store knowledge and use it for design is the CATIA program through its KnowledgeWare module. An example of a CAD-independent, language-based KBE system with full compiler and support for runtime application deployment is General-purpose Declarative Language (GDL) from Genworks.

KBE can have a wide scope that covers the full range of activities related to Product Lifecycle Management and Multidisciplinary design optimization. KBE's scope would include design, analysis (computer-aided engineering – CAE), manufacturing, and support. In this inclusive role, KBE has to cover a large multi-disciplinary role related to many computer aided technologies (CAx).

KBE also has more general overtones. One of its roles is to bridge knowledge management and design automation. Knowledge processing is a recent advance in computing. It has played a successful role in engineering and is now undergoing modifications (to be explained). An example of KBE's role is generative mechanical design. There are others. KBE can be thought of as an advanced form of computer applications (in some forms with an extreme end-user computing flavor) that support PLM and CAx.

There are similar techniques, such as electronic design automation. AAAI provides a long list of engineering applications. some of which are within the KBE umbrella. At some point, the concept of KBE might split into several sub-categories as MCAD and ECAD are just two of many possible types of design automation.

History

KBE essentially was a complementary development to CAx and can be dated from the 1980s. CAx has been developing along with the computer after making large strides in the 1970s.

As with any bit of progress, KBE flashed on the horizon, lit the sky for a while, and then experienced a downslide. KBE had sufficient success stories that sustained it long enough into the 1990s to get attention. Some prime contributors to the hiatus of KBE were unmanageable expectations, increasing tedium associated with forming completion of results, and some notion that the architecture for KBE was not sufficiently based upon the newer technology.

KBE continued to exist in pockets. With the prevalence of object-oriented methods, systems advanced enough to allow re-implementation. This reconstruction has been on-going for several years and has been frustratingly slow. Now, with the basis for this discipline becoming more robust, it starts to get interesting again.

KBE, as implemented with ICAD can be thought of as an advanced form of computer applications (in some forms with an extreme end-user computing flavor) that support PLM and CAx.

KBE and product lifecycle management

The scope of PLM involves all the steps that exist within any industry that produces goods. KBE at this level will deal with product issues of a more generic nature than it will with CAx. Some might call this level 'assembly' in orientation. However, it's much more than that as PLM covers both the technical and the business side of a product.

KBE then needs to support the decision processes involved with configuration, trades, control, management, and a number of other areas, such as optimization.

Recently the Object Management Group released a RFP document and requested feedback.

KBE and CAX

CAx crosses many disciplinary bounds and provides a sound basis for PLM. In a sense, CAx is a form of applied science that uses most of the disciplines of engineering and their associated fields. Materials science comes to mind.

KBE's support of CAx may have some similarities with its support of PLM but, in a sense, the differences are going to be larger.

The KBE flavor at the CAx level may assume a strong behavioral flavor. Given the underlying object oriented focus, there is a natural use of entities possessing complicated attributes and fulfilling non-trivial roles. One vendor's approach provides a means via workbenches to embed attributes and methods within sub-parts (object) or within a joining of sub-parts into a part.

As an aggregate, the individual actions, that are event driven, can be fairly involved. This fact identifies one major problem, namely control of what is essentially a non-deterministic mixture. This characteristic of the decision problem will get more attention as the KBE systems subsume more levels and encompasses a broader scope of PLM.

KBE and knowledge management

KBE is related to knowledge management which has many levels itself. Some approaches to knowledge are reductionistic, as well they ought to be given the pragmatic focus of knowledge modeling. However, due to KBE dealing with aggregates that can be quite complicated both in structure and in behavior, some holistic notions (note link to complexity theory) might be apropos.

Also, given all the layers of KBE and given the fact that one part of an associated space is heavily mathematical (namely, manifold in nature), KBE is extremely interesting from the knowledge viewpoint (or one would hope).

All one has to do is note that the KBE process's goal is to produce results in the 'real world' via artifacts and to do so using techniques that are highly computational. That, in essence, is the epitome of applied science/engineering, and it could never be non-interesting.

KBE methodology

The development of KBE applications concerns the requirements to identify, capture, structure, formalize and finally implement knowledge. Many different so-called KBE platforms support only the implementation step which is not always the main bottleneck in the KBE development process. In order to limit the risk associated with the development and maintenance of KBE application there is a need to rely on an appropriate methodology for managing the knowledge and maintaining it up to date. As example of such KBE methodology the EU project MOKA "Methodology and tools

Oriented to Knowledge based Applications" propose solutions which focus on the structuration and formalization steps as well as links to the implementation.

An alternative to MOKA is to use a general methodology for developing knowledge bases for expert systems and for intranet pages. Such a methodology is described in "Knowledge Acquisition in Practice: A Step-by-step Guide" by Nick Milton

Languages for KBE

Some questions can be asked in regard to KBE implementation: can we represent knowledge in a vendor-neutral format? can the knowledge in our designs be retained for decades, long after a vendor system (such as CATIA) has disappeared?

These questions are addressed in a 2005 Aerospace COE presentation A Proposal for CATIA V6 by Walter Wilson of Lockheed Martin.

Mr. Wilson advocates using a type of programming language to define design data— operations, parameters, formulas, etc. -- instead of a proprietary file format (such as Dassault's CATIA). One's data would no longer be tied to a specific CAD system. Unlike STEP, which inevitably lags commercial CAD systems in the features it supports, programmability would allow the definition of new design features.

A logic programming language is proposed as the basis for the engineering design language because of its simplicity and extensibility. The geometric engine for the language features would be open source to give engineers control over approximation algorithms and to better guarantee long-term accessibility of the data.

Meanwhile, the commercially available General-purpose Declarative Language (GDL) from Genworks International addresses the issue of application longevity by providing a high-level declarative language kernel which is a superset of a standard dialect of the Lisp programming language (currently ANSI Common Lisp, or CL).

The GDL kernel follows a concise, pragmatic language specification representing something akin to a *de-facto* neutral format for representing KBE-style knowledge. It consists of the same Smalltalk-inspired declarative object-oriented (and object-centric) message-passing format which been a common thread among classical KBE systems for more than two decades. While CL is a multi-paradigm language (supporting procedural, object-oriented, and functional programming), the core features of KBE tend to share several aspects with Functional programming languages "under the hood" (e.g. lazy evaluation, immutable data). However there is a consensus that pure Functional programming is too esoteric for typical engineers to practice, so one of the purposes of a declarative KBE language such as GDL is to provide an "engineer-friendly" object-centric front-end on what is essentially a Functional language programming environment.

Because GDL applications are written as a strict superset of a standard Lisp dialect, only the high-level declarative surface syntax is GDL-specific. The bulk of application code is purely compliant with the underlying language standard. And because of Lisp's inherent (and unique) support for code transformation macros, even this surface syntax is subject to straightforward automated conversion among other variations of the de-facto standard. It is reasonable to expect that implementations following this approach will eventually converge on a true vendor-neutral Standard KBE language specification.

The Pacelab Suite addresses the problem that functional programming languages are still not widely accepted by potential KBE users. Engineers are usually trained in procedural and object-oriented programming languages; in quite a few software environments (Excel, MATLAB and FORTRAN) used by engineers the procedural programming style clearly dominates. Therefore the Pacelab Suite rebases the inherent technological advantages offered by a development and runtime environment like Common Lisp, on a new technological paradigm (.NET Framework) equally supporting procedural, object-oriented and functional languages.

KBE in Academia

- Knowledge-based engineering at the Norwegian University of Science and Technology (NTNU)
- Knowledge Based Engineering department at the Faculty of Aerospace Engineering of the Delft University of Technology

Implementations

The following KBE development packages are commercially available:

For CAD

- Adaptive Modeling Language from TechnoSoft Inc.
- DriveWorks A SolidWorks Certified Gold Partner
- GDL from Genworks International
- Kadviser from NIMTOTH previously edited by Kade-Tech
- KBEWorks by VisionKBE
- Knowledge Fusion from Siemens PLM Software
- Knowledgeware from Dassault Systemes
- Magix by Navitech
- Pro/ENGINEER Expert Framework from Parametric Technology Corporation
- SmartAssembly for Pro/ENGINEER from Sigmaxim Inc
- TactonWorks Interactive design automation inside SolidWorks
- YVE - Your Variant Engineer from tecneos software-engineering
- ICAD from Dassault Systemes (no longer available)
- KBMax Configurator by Citius Corporation

For General-purpose development of Web-deployed applications

- GDL from Genworks International

For analysis, design and engineering processes

- GDL from Genworks International
- Pacelab Suite by PACE Aerospace Engineering and Information Technology GmbH
- PCPACK by Tacit Connexions
- Quaestor by Qnowledge Modeling Technologies

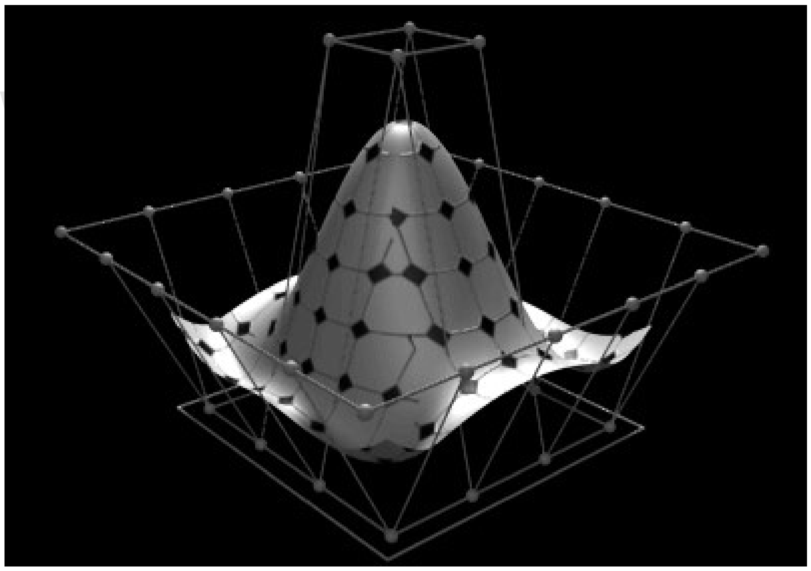
KBE futures, KBE theory

KBE, as a particular example of KS, is a multi-disciplinary framework that has more than practical considerations. Not only will KBE require successful handling of issues of the computational (Ontology, Artificial Intelligence, Entscheidungsproblem, Interactive computation, Category Theory, ...) and logic (non-monotonic issues related to the qualification, frame, and ramification problems)), it will touch upon all sciences that deal with matter, its manipulations, and the related decisions. In a sense, PLM allows us to have the world as a large laboratory for experimental co-evolution of our knowledge and our artificial co-horts. As noted in the ACM Communications, "Computers will grow to become scientists in their own right, with intuitions and computational variants of fascination and curiosity." What better framework is there to explore the "increasingly complicated mappings between the human world and the computational"?

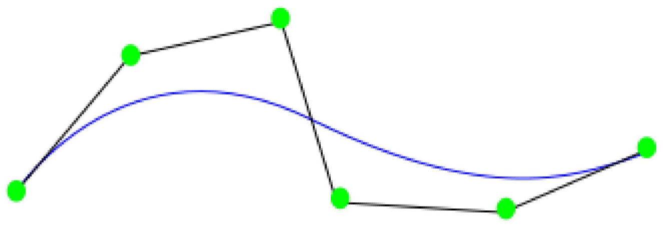
In terms of methodology and their associated means, KBE offers support via several paradigms. These range from the home-grown all the way to strategically defined and integrated tools that cover both breadth and depth. A continuing theme will be resolving the contextual definitions for KBE into a coherent discipline (or at least attempting this) and keeping a handle on managing the necessary quantitative comparisons. One issue of importance considers what limits there may be to the computational; this study requires a multi-disciplinary focus and an understanding of the quasi-empirical. Given the knowledge focus of KBE, another issue involves what limits there might be to a computational basis for knowledge and whether these are overcome with the more advanced types of human-machine interface.

Chapter-11

Non-uniform Rational B-spline



Three-dimensional NURBS surfaces can have complex, organic shapes. Control points influence the directions the surface takes. The outermost square below delineates the X/Y extents of the surface.



A NURBS curve.

Non-uniform rational basis spline (**NURBS**) is a mathematical model commonly used in computer graphics for generating and representing curves and surfaces which offers great flexibility and precision for handling both analytic and freeform shapes.

History

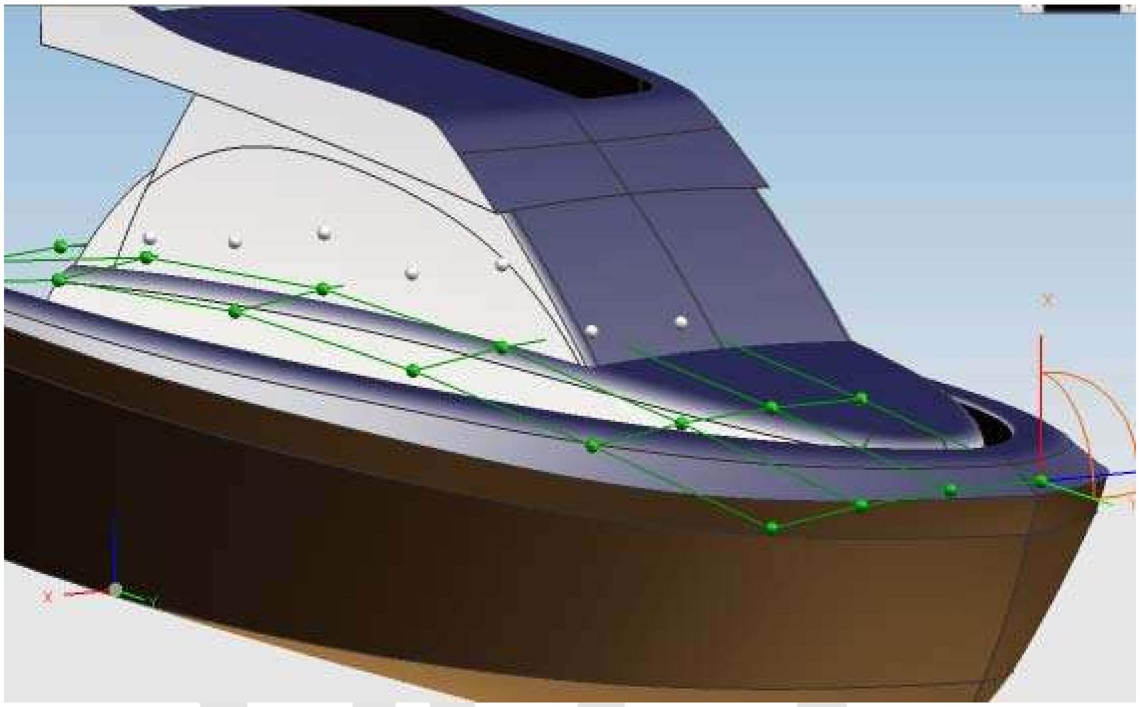
Development of NURBS began in the 1950s by engineers who were in need of a mathematically precise representation of freeform surfaces like those used for ship hulls, aerospace exterior surfaces, and car bodies, which could be exactly reproduced whenever technically needed. Prior representations of this kind of surface only existed as a single physical model created by a designer.

The pioneers of this development were Pierre Bézier who worked as an engineer at Renault, and Paul de Casteljau who worked at Citroën, both in France. Bézier worked nearly parallel to de Casteljau, neither knowing about the work of the other. But because Bézier published the results of his work, the average computer graphics user today recognizes splines — which are represented with control points lying off the curve itself — as Bézier splines, while de Casteljau's name is only known and used for the algorithms he developed to evaluate parametric surfaces. In the 1960s it became clear that non-uniform, rational B-splines are a generalization of Bézier splines, which can be regarded as uniform, non-rational B-splines.

At first NURBS were only used in the proprietary CAD packages of car companies. Later they became part of standard computer graphics packages.

Real-time, interactive rendering of NURBS curves and surfaces was first made available on Silicon Graphics workstations in 1989. In 1993, the first interactive NURBS modeller for PCs, called NöRBS, was developed by CAS Berlin, a small startup company cooperating with the Technical University of Berlin. Today most professional computer graphics applications available for desktop use offer NURBS technology, which is most often realized by integrating a NURBS engine from a specialized company.

Use



NURBS are commonly used in computer-aided design (CAD), manufacturing (CAM), and engineering (CAE) and are part of numerous industry wide used standards, such as IGES, STEP, ACIS, and PHIGS. NURBS tools are also found in various 3D modeling and animation software packages, such as form•Z, Blender, Maya, Rhino3D, Cinema 4D, Cobalt, Shark FX, and Solid Modeling Solutions. Other than this there are specialized NURBS modeling software packages such as Autodesk Alias Surface, solidThinking and ICEM Surf.

They allow representation of geometrical shapes in a compact form. They can be efficiently handled by the computer programs and yet allow for easy human interaction. NURBS surfaces are functions of two parameters mapping to a surface in three-dimensional space. The shape of the surface is determined by control points.

In general, editing NURBS curves and surfaces is highly intuitive and predictable. Control points are always either connected directly to the curve/surface, or act as if they were connected by a rubber band. Depending on the type of user interface, editing can be realized via an element's control points, which are most obvious and common for Bézier curves, or via higher level tools such as spline modeling or hierarchical editing.

A surface under construction, e.g. the hull of a motor yacht, is usually composed of several NURBS surfaces known as *patches*. These patches should be fitted together in such a way that the boundaries are invisible. This is mathematically expressed by the concept of geometric continuity.

Higher-level tools exist which benefit from the ability of NURBS to create and establish geometric continuity of different levels:

Positional continuity (G0)
: holds whenever the end positions of two curves or surfaces are coincidental. The curves or surfaces may still meet at an angle, giving rise to a sharp corner or edge and causing broken highlights.

Tangential continuity (G1)
: requires the end vectors of the curves or surfaces to be parallel, ruling out sharp edges. Because highlights falling on a tangentially continuous edge are always continuous and thus look natural, this level of continuity can often be sufficient.

Curvature continuity (G2)
: further requires the end vectors to be of the same length and rate of length change. Highlights falling on a curvature-continuous edge do not display any change, causing the two surfaces to appear as one. This can be visually recognized as "perfectly smooth". This level of continuity is very useful in the creation of models that require many bi-cubic patches composing one continuous surface.

Geometric continuity mainly refers to the shape of the resulting surface; since NURBS surfaces are functions, it is also possible to discuss the derivatives of the surface with respect to the parameters. This is known as parametric continuity. Parametric continuity of a given degree implies geometric continuity of that degree.

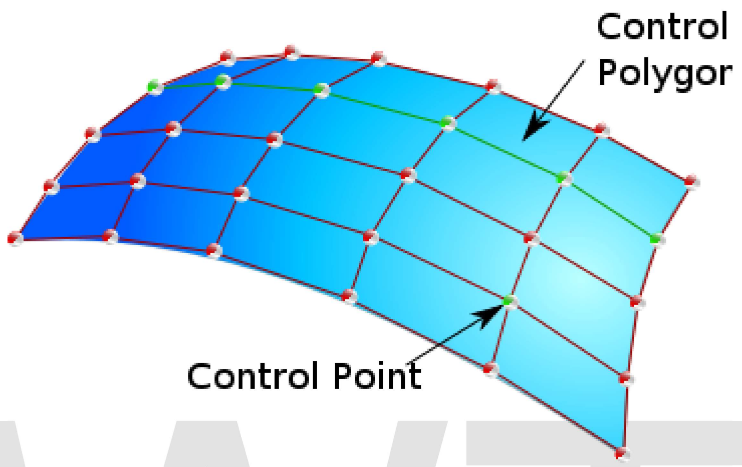
First- and second-level parametric continuity (C0 and C1) are for practical purposes identical to positional and tangential (G0 and G1) continuity. Third-level parametric continuity (C2), however, differs from curvature continuity in that its parameterization is also continuous. In practice, C2 continuity is easier to achieve if uniform B-splines are used.

The definition of the continuity 'C*n*' requires that the n^{th} derivative of the curve/surface ($d^nC(u) / du^n$) are equal at a joint. Note that the (partial) derivatives of curves and surfaces are vectors that have a direction and a magnitude. Both should be equal.

Highlights and reflections can reveal the perfect smoothing, which is otherwise practically impossible to achieve without NURBS surfaces that have at least G2 continuity. This same principle is used as one of the surface evaluation methods whereby a ray-traced or reflection-mapped image of a surface with white stripes reflecting on it will show even the smallest deviations on a surface or set of surfaces. This method is derived from car prototyping wherein surface quality is inspected by checking the quality of reflections of a neon-light ceiling on the car surface. This method is also known as "Zebra analysis".

Technical specifications

A **NURBS curve** is defined by its *order*, a set of weighted *control points*, and a *knot vector*. NURBS curves and surfaces are generalizations of both B-splines and Bézier curves and surfaces, the primary difference being the weighting of the control points which makes NURBS curves *rational* (non-rational B-splines are a special case of rational B-splines). Whereas Bézier curves evolve into only one parametric direction, usually called *s* or *u*, NURBS surfaces evolve into two parametric directions, called *s* and *t* or *u* and *v*.



By evaluating a NURBS curve at various values of the parameter, the curve can be represented in cartesian two- or three-dimensional space. Likewise, by evaluating a NURBS surface at various values of the two parameters, the surface can be represented in cartesian space.

NURBS curves and surfaces are useful for a number of reasons:

- They are invariant under affine as well as perspective transformations: operations like rotations and translations can be applied to NURBS curves and surfaces by applying them to their control points.
- They offer one common mathematical form for both standard analytical shapes (e.g., conics) and free-form shapes.
- They provide the flexibility to design a large variety of shapes.
- They reduce the memory consumption when storing shapes (compared to simpler methods).
- They can be evaluated reasonably quickly by numerically stable and accurate algorithms.

In the next sections, NURBS is discussed in one dimension (curves). It should be noted that all of it can be generalized to two or even more dimensions.

Control points

The control points determine the shape of the curve. Typically, each point of the curve is computed by taking a weighted sum of a number of control points. The weight of each point varies according to the governing parameter. For a curve of degree d, the weight of any control point is only nonzero in d+1 intervals of the parameter space. Within those intervals, the weight changes according to a polynomial function (*basis functions*) of degree d. At the boundaries of the intervals, the basis functions go smoothly to zero, the smoothness being determined by the degree of the polynomial.

As an example, the basis function of degree one is a triangle function. It rises from zero to one, then falls to zero again. While it rises, the basis function of the previous control point falls. In that way, the curve interpolates between the two points, and the resulting curve is a polygon, which is continuous, but not differentiable at the interval boundaries, or **knots**. Higher degree polynomials have correspondingly more continuous derivatives. Note that within the interval the polynomial nature of the basis functions and the linearity of the construction make the curve perfectly smooth, so it is only at the knots that discontinuity can arise.

The fact that a single control point only influences those intervals where it is active is a highly desirable property, known as **local support**. In modelling, it allows the changing of one part of a surface while keeping other parts equal.

Adding more control points allows better approximation to a given curve, although only a certain class of curves can be represented exactly with a finite number of control points. NURBS curves also feature a scalar **weight** for each control point. This allows for more control over the shape of the curve without unduly raising the number of control points. In particular, it adds conic sections like circles and ellipses to the set of curves that can be represented exactly. The term *rational* in NURBS refers to these weights.

The control points can have any dimensionality. One-dimensional points just define a scalar function of the parameter. These are typically used in image processing programs to tune the brightness and color curves. Three-dimensional control points are used abundantly in 3D modelling, where they are used in the everyday meaning of the word 'point', a location in 3D space. Multi-dimensional points might be used to control sets of time-driven values, e.g. the different positional and rotational settings of a robot arm. NURBS surfaces are just an application of this. Each control 'point' is actually a full vector of control points, defining a curve. These curves share their degree and the number of control points, and span one dimension of the parameter space. By interpolating these control vectors over the other dimension of the parameter space, a continuous set of curves is obtained, defining the surface.

The knot vector

The knot vector is a sequence of parameter values that determines where and how the control points affect the NURBS curve. The number of knots is always equal to the number of control points plus curve degree plus one. The knot vector divides the parametric space in the intervals mentioned before, usually referred to as *knot spans*. Each time the parameter value enters a new knot span, a new control point becomes active, while an old control point is discarded. It follows that the values in the knot vector should be in nondecreasing order, so (0, 0, 1, 2, 3, 3) is valid while (0, 0, 2, 1, 3, 3) is not.

Consecutive knots can have the same value. This then defines a knot span of zero length, which implies that two control points are activated at the same time (and of course two control points become deactivated). This has impact on continuity of the resulting curve or its higher derivatives; for instance, it allows to create corners in an otherwise smooth NURBS curve. A number of coinciding knots is sometimes referred to as a knot with a certain **multiplicity**. Knots with multiplicity two or three are known as double or triple knots. The multiplicity of a knot is limited to the degree of the curve; since a higher multiplicity would split the curve into disjoint parts and it would leave control points unused. For first-degree NURBS, each knot is paired with a control point.

The knot vector usually starts with a knot that has multiplicity equal to the order. This makes sense, since this activates the control points that have influence on the first knot span. Similarly, the knot vector usually ends with a knot of that multiplicity. Curves with such knot vectors start and end in a control point.

The individual knot values are not meaningful by themselves; only the ratios of the difference between the knot values matter. Hence, the knot vectors (0, 0, 1, 2, 3, 3) and (0, 0, 2, 4, 6, 6) produce the same curve. The positions of the knot values influences the mapping of parameter space to curve space. Rendering a NURBS curve is usually done by stepping with a fixed stride through the parameter range. By changing the knot span lengths, more sample points can be used in regions where the curvature is high. Another use is in situations where the parameter value has some physical significance, for instance if the parameter is time and the curve describes the motion of a robot arm. The knot span lengths then translate into velocity and acceleration, which are essential to get right to prevent damage to the robot arm or its environment. This flexibility in the mapping is what the phrase *non uniform* in NURBS refers to.

Necessary only for internal calculations, knots are usually not helpful to the users of modeling software. Therefore, many modeling applications do not make the knots editable or even visible. It's usually possible to establish reasonable knot vectors by looking at the variation in the control points. More recent versions of NURBS software (e.g., Autodesk Maya and Rhinoceros 3D) allow for interactive editing of knot positions, but this is significantly less intuitive than the editing of control points.

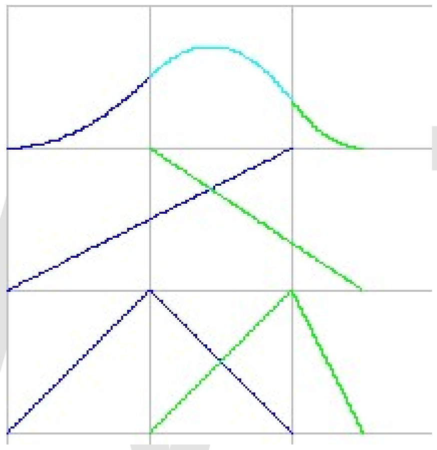
Order

The *order* of a NURBS curve defines the number of nearby control points that influence any given point on the curve. The curve is represented mathematically by a polynomial of degree one less than the order of the curve. Hence, second-order curves (which are represented by linear polynomials) are called linear curves, third-order curves are called quadratic curves, and fourth-order curves are called cubic curves. The number of control points must be greater than or equal to the order of the curve.

In practice, cubic curves are the ones most commonly used. Fifth- and sixth-order curves are sometimes useful, especially for obtaining continuous higher order derivatives, but curves of higher orders are practically never used because they lead to internal numerical problems and tend to require disproportionately large calculation times.

Construction of the basis functions

The basis functions used in NURBS curves are usually denoted as $N_{i,n}(u)$, in which i corresponds to the i-th control point, and n corresponds with the degree of the basis function. The parameter dependence is frequently left out, so we can write $N_{i,n}$. The definition of these basis functions is recursive in n. The degree-0 functions $N_{i,0}$ are piecewise constant functions. They are one on the corresponding knot span and zero everywhere else. Effectively, $N_{i,n}$ is a linear interpolation of $N_{i,n-1}$ and $N_{i+1,n-1}$. The latter two functions are non-zero for n knot spans, overlapping for $n-1$ knot spans. The function $N_{i,n}$ is computed as



From bottom to top: Linear basis functions $N_{1,1}$ (blue) and $N_{2,1}$ (green), their weight functions f and g and the resulting quadratic basis function. The knots are 0, 1, 2 and 2.5

$$N_{i,n} = f_{i,n}N_{i,n-1} + g_{i+1,n}N_{i+1,n-1}$$

f_i rises linearly from zero to one on the interval where $N_{i,n-1}$ is non-zero, while g_{i+1} falls from one to zero on the interval where $N_{i+1,n-1}$ is non-zero. As mentioned before, $N_{i,1}$ is a triangular function, nonzero over two knot spans rising from zero to one on the first, and falling to zero on the second knot span. Higher order basis functions are non-zero over corresponding more knot spans and have correspondingly higher degree. If u is the parameter, and k_i is the i-th knot, we can write the functions f and g as

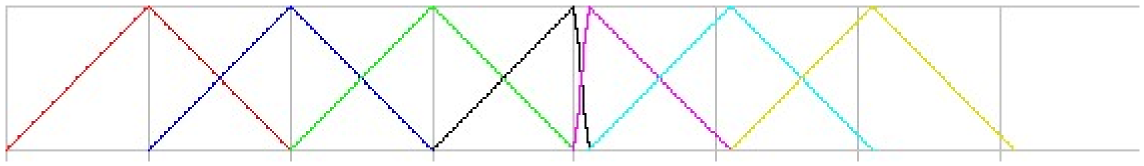
$$f_{i,n}(u) = \frac{u - k_i}{k_{i+n} - k_i}$$

and

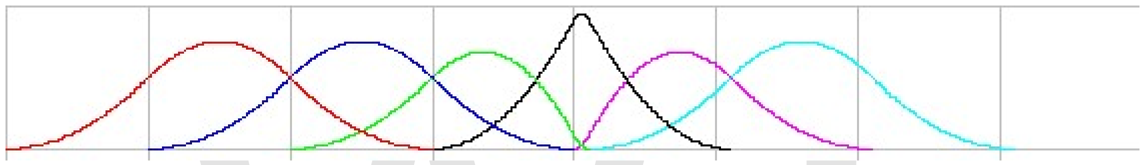
$$g_{i,n}(u) = \frac{k_{i+n} - u}{k_{i+n} - k_i}$$

The functions *f* and *g* are positive when the corresponding lower order basis functions are non-zero. By induction on n it follows that the basis functions are non-negative for all values of *n* and *u*. This makes the computation of the basis functions numerically stable.

Again by induction, it can be proved that the sum of the basis functions for a particular value of the parameter is unity. This is known as the **partition of unity** property of the basis functions.



Linear basis functions



Quadratic basis functions

The figures show the linear and the quadratic basis functions for the knots {..., 0, 1, 2, 3, 4, 4.1, 5.1, 6.1, 7.1, ...}

One knot span is considerably shorter than the others. On that knot span, the peak in the quadratic basis function is more distinct, reaching almost one. Conversely, the adjoining basis functions fall to zero more quickly. In the geometrical interpretation, this means that the curve approaches the corresponding control point closely. In case of a double knot, the length of the knot span becomes zero and the peak reaches one exactly. The basis function is no longer differentiable at that point. The curve will have a sharp corner if the neighbour control points are not collinear.

General form of a NURBS curve

Using the definitions of the basis functions $N_{i,n}$ from the previous paragraph, a NURBS curve takes the following form:

$$C(u) = \sum_{i=1}^{k} \frac{N_{i,n} w_i}{\sum_{j=1}^{k} N_{j,n} w_j} \mathbf{P}_i = \frac{\sum_{i=1}^{k} N_{i,n} w_i \mathbf{P}_i}{\sum_{i=1}^{k} N_{i,n} w_i}$$

In this, k is the number of control points $\mathbf{P}_i$ and w_i are the corresponding weights. The denominator is a normalizing factor that evaluates to one if all weights are one. This can be seen from the partition of unity property of the basis functions. It is customary to write this as

$$C(u) = \sum_{i=1}^{k} R_{i,n} \mathbf{P}_i$$

in which the functions

$$R_{i,n} = \frac{N_{i,n} w_i}{\sum_{j=1}^{k} N_{j,n} w_j}$$

are known as the *rational basis functions*.

Manipulating NURBS objects

A number of transformations can be applied to a NURBS object. For instance, if some curve is defined using a certain degree and N control points, the same curve can be expressed using the same degree and N+1 control points. In the process a number of control points change position and a knot is inserted in the knot vector. These manipulations are used extensively during interactive design. When adding a control point, the shape of the curve should stay the same, forming the starting point for further adjustments. A number of these operations are discussed below.

Knot insertion

As the term suggests, **knot insertion** inserts a knot into the knot vector. If the degree of the curve is n, then $n - 1$ control points are replaced by n new ones. The shape of the curve stays the same.

A knot can be inserted multiple times, up to the maximum multiplicity of the knot. This is sometimes referred to as **knot refinement** and can be achieved by an algorithm that is more efficient than repeated knot insertion.

Knot removal

Knot removal is the reverse of knot insertion. Its purpose is to remove knots and the associated control points in order to get a more compact representation. Obviously, this is not always possible while retaining the exact shape of the curve. In practice, a tolerance in the accuracy is used to determine whether a knot can be removed. The process is used to clean up after an interactive session in which control points may have been added manually, or after importing a curve from a different representation, where a straightforward conversion process leads to redundant control points.

Degree elevation

A NURBS curve of a particular degree can always be represented by a NURBS curve of higher degree. This is frequently used when combining separate NURBS curves, e.g.

when creating a NURBS surface interpolating between a set of NURBS curves or when unifying adjacent curves. In the process, the different curves should be brought to the same degree, usually the maximum degree of the set of curves. The process is known as **degree elevation**.

Curvature

The most important property in differential geometry is the curvature κ. It describes the local properties (edges, corners, etc.) and relations between the first and second derivative, and thus, the precise curve shape. Having determined the derivatives it is easy to compute $\kappa = \frac{|r'(t) \times r''(t)|}{|r'(t)|^3}$ or approximated as the arclength from the second derivate $\kappa = |\, r''(s_o)\,|$. The direct computation of the curvature κ with these equations is the big advantage of parameterized curves against their polygonal representations.

Example: a circle

Non-rational splines or Bézier curves may approximate a circle, but they cannot represent it exactly. Rational splines can represent any conic section, including the circle, exactly. This representation is not unique, but one possibility appears below:

x	***y***	***z***	**weight**
1	0	0	1
$\sqrt{2}/2$	$\sqrt{2}/2$	0	$\sqrt{2}/2$
0	1	0	1
$-\sqrt{2}/2$	$\sqrt{2}/2$	0	$\sqrt{2}/2$
−1	0	0	1
$-\sqrt{2}/2$	$-\sqrt{2}/2$	0	$\sqrt{2}/2$
0	−1	0	1
$\sqrt{2}/2$	$-\sqrt{2}/2$	0	$\sqrt{2}/2$
1	0	0	1

The order is three, since a circle is a quadratic curve and the spline's order is one more than the degree of its piecewise polynomial segments. The knot vector is $\{0, 0, 0, \pi/2, \pi/2, \pi, \pi, 3\pi/2, 3\pi/2, 2\pi, 2\pi, 2\pi\}$. The circle is composed of four quarter circles, tied together with double knots. Although double knots in a third order NURBS curve would normally result in loss of continuity in the first derivative, the control points are positioned in such a way that the first derivative is continuous. (In fact, the curve is infinitely differentiable everywhere, as it must be if it exactly represents a circle.)

The curve represents a circle exactly, but it is not exactly parametrized in the circle's arc length. This means, for example, that the point at *t* does not lie at (sin(*t*),cos(*t*)) (except for the start, middle and end point of each quarter circle, since the representation is symmetrical). This is obvious; the x coordinate of the circle would otherwise provide an exact rational polynomial expression for cos(*t*), which is impossible. The circle does make one full revolution as its parameter *t* goes from 0 to 2π, but this is only because the knot vector was arbitrarily chosen as multiples of π / 2.

Chapter-12

Solid Modeling

The geometry in solid modeling is fully described in 3-D space; objects can be viewed from any angle. *Modeled and ray traced in Cobalt*

Solid modeling (or **modelling**) is a consistent set of principles for mathematical and computer modeling of three dimensional solids. Solid modeling is distinguished from related areas of Geometric modeling and Computer graphics by its emphasis on physical fidelity. Together, the principles of geometric and solid modeling form the foundation of Computer-aided design and in general support the creation, exchange, visualization, animation, interrogation, and annotation of digital models of physical objects.

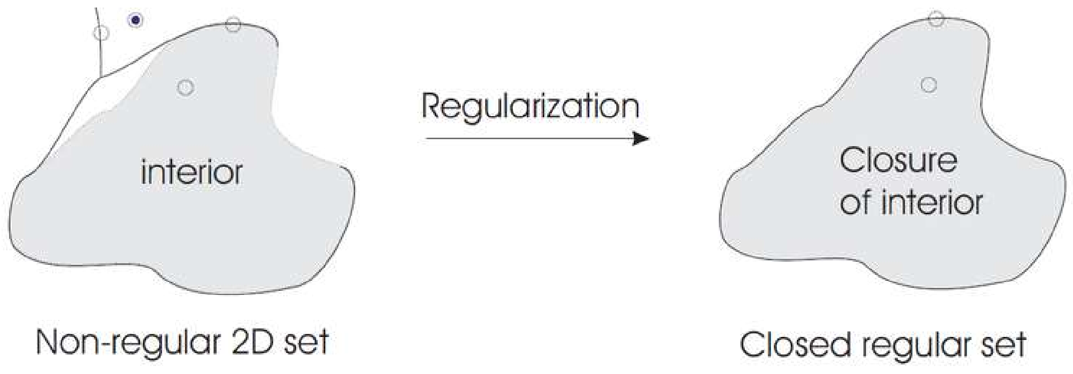
Overview

The use of solid modeling techniques allows for the automation of several difficult engineering calculations that are carried out as a part of the design process. Simulation, planning, and verification of processes such as machining and assembly were one of the main catalysts for the development of solid modeling. More recently, the range of supported manufacturing applications has been greatly expanded to include sheet metal manufacturing, injection molding, welding, pipe routing etc. Beyond traditional manufacturing, solid modeling techniques serve as the foundation for rapid prototyping, digital data archival and reverse engineering by reconstructing solids from sampled points on physical objects, mechanical analysis using finite elements, motion planning and NC path verification, kinematic and dynamic analysis of mechanisms, and so on. A central problem in all these applications is the ability to effectively represent and manipulate three dimensional geometry in a fashion that is consistent with the physical behavior of real artifacts. Solid modeling research and development has effectively addressed many of these issues, and continues to be a central focus of computer aided engineering.

Mathematical foundations

The notion of solid modeling as practiced today relies on the specific need for informational completeness in mechanical geometric modeling systems, in the sense that any computer model should support all geometric queries that may be asked of its corresponding physical object. The requirement implicitly recognizes the possibility of several computer representations of the same physical object as long as any two such representations are consistent. It is impossible to computationally verify informational completeness of a representation unless the notion of a physical object is defined in terms of computable mathematical properties and independent of any particular representation. Such reasoning led to the development of the modeling paradigm that has shaped the field of solid modeling as we know it today.

All manufactured components have finite size and well behaved boundaries, so initially the focus was on mathematically modeling rigid parts made of homogeneous isotropic material that could be added or removed. These postulated properties can be translated into properties of subsets of three dimensional Euclidean space. The two common approaches to define solidity rely on point-set topology and algebraic topology respectively. Both models specify how solids can be built from simple pieces or cells.



Regularization of a 2-d set by taking the closure of its interior

According to the continuum point set model of solidity, all the points of any $X \subset \mathbb{R}^3$ can be classified according to their neighborhoods with respect to X as interior, exterior, or boundary points. Assuming $\mathbb{R}^3$ is endowed with the typical Euclidean metric, a neighborhood of a point $p \in X$ takes the form of an open ball. For X to be considered solid every neighborhood of any $p \in X$ must be consistently three dimensional; points with lower dimensional neighborhoods indicate a lack of solidity. Dimensional homogeneity of neighborhoods is guaranteed for the class of *closed regular* sets, defined as sets equal to the closure of their interior. Any $X \subset \mathbb{R}^3$ can be turned into a closed regular set or *regularized* by taking the closure of its interior, and thus the modeling space of solids is mathematically defined to be the space of closed regular subsets of $\mathbb{R}^3$ (by the Heine-Borel theorem it is implied that all solids are compact sets). In addition, solids are required to be closed under the Boolean operations of set union, intersection, and difference (to guarantee solidity after material addition and removal). Applying the standard Boolean operations to closed regular sets may not produce a closed regular set, but this problem can be solved by regularizing the result of applying the standard Boolean operations. The regularized set operations are denoted $\cup^*$, $\cap^*$, and $-^*$.

The combinatorial characterization of a set $X \subset \mathbb{R}^3$ as a solid involves representing X as an orientable cell complex so that the cells provide finite spatial addresses for points in an otherwise innumerable continuum. The class of semi-analytic bounded subsets of Euclidean space is closed under Boolean operations (standard and regularized) and exhibits the additional property that every semi-analytic set can be stratified into a collection of disjoint cells of dimensions 0,1,2,3. A triangulation of a semi-analytic set into a collection of points, line segments, triangular faces, and tetrahedral elements is an example of a stratification that is commonly used. The combinatorial model of solidity is then summarized by saying that in addition to being semi-analytic bounded subsets, solids are three dimensional topological polyhedra, specifically three dimensional orientable manifolds with boundary. In particular this implies the Euler characteristic of the combinatorial boundary of the polyhedron is 2. The combinatorial manifold model of solidity also guarantees the boundary of a solid separates space into exactly two components as a consequence of the Jordan-Brouwer theorem, thus eliminating sets with non-manifold neighborhoods that are deemed impossible to manufacture.

The point-set and combinatorial models of solids are entirely consistent with each other, can be used interchangeably relying on continuum or combinatorial properties as needed, and can be extended to *n* dimensions. The key property that facilitates this consistency is that the class of closed regular subsets of $\mathbb{R}^n$ coincides precisely with homogeneously *n*-dimensional topological polyhedra. Therefore every *n*-dimensional solid may be unambiguously represented by its boundary and the boundary has the combinatorial structure of an *n-1*-dimensional polyhedron having homogeneously *n-1* dimensional neighborhoods.

Solid representation schemes

Based on assumed mathematical properties, any scheme of representing solids is a method for capturing information about the class of semi-analytic subsets of Euclidean space. This means all representations are different ways of organizing the same geometric and topological data in the form of a data structure. All representation schemes are organized in terms of a finite number of operations on a set of primitives. Therefore the modeling space of any particular representation is finite, and any single representation scheme may not completely suffice to represent all types of solids. For example, solids defined via combinations of regularized boolean operations cannot necessarily be represented as the sweep of a primitive moving according to a space trajectory, except in very simple cases. This forces modern geometric modeling systems to maintain several representation schemes of solids and also facilitate efficient conversion between representation schemes.

Below is a list of common techniques used to create or represent solid models. Modern modeling software may use a combination of these schemes to represent a solid.

Parameterized primitive instancing

This scheme is based on the notion of families of objects, each member of a family distinguishable from the other by a few parameters. Each object family is called a *generic primitive*, and individual objects within a family are called *primitive instances*. For example a family of bolts is a generic primitive, and a single bolt specified by a particular set of parameters is a primitive instance. The distinguishing characteristic of pure parameterized instancing schemes is the lack of means for combining instances to create new structures which represent new and more complex objects. The other main drawback of this scheme is the difficulty of writing algorithms for computing properties of represented solids. A considerable amount of family-specific information must be built into the algorithms and therefore each generic primitive must be treated as a special case, allowing no uniform overall treatment.

Spatial occupancy enumeration

This scheme is essentially a list of spatial *cells* occupied by the solid. The cells, also called voxels are cubes of a fixed size and are arranged in a fixed spatial grid (other polyhedral arrangements are also possible but cubes are the simplest). Each cell may be represented by the coordinates of a single point, such as the cell's centroid. Usually a specific scanning order is imposed and the corresponding ordered set of coordinates is called a *spatial array*. Spatial arrays are unambiguous and unique solid representations but are too verbose for use as 'master' or definitional representations. They can, however, represent coarse approximations of parts and can be used to improve the performance of geometric algorithms, especially when used in conjunction with other representations such as Constructive Solid Geometry.

Cell decomposition

This scheme follows from the combinatoric (algebraic topological) descriptions of solids detailed above. A solid can be represented by its decomposition into several cells. Spatial occupancy enumeration schemes are a particular case of cell decompositions where all the cells are cubical and lie in a regular grid. Cell decompositions provide convenient ways for computing certain topological properties of solids such as its connectedness (number of pieces) and genus (number of holes). Cell decompositions in the form of triangulations are the representations used in 3d finite elements for the numerical solution of partial differential equations. Other cell decompositions such as a Whitney regular stratification or Morse decompositions may be used for applications in robot motion planning.

Boundary representation

In this scheme a solid is represented by the cellular decomposition of its boundary. Since the boundaries of solids have the distinguishing property that they separate space into regions defined by the interior of the solid and the complementary exterior according to the Jordan-Brouwer theorem discussed above, every point in space can unambiguously be tested against the solid by testing the point against the boundary of the solid. Recall that ability to test every point in the solid provides a guarantee of solidity. Using ray casting it is possible to count the number of intersections of a cast ray against the boundary of the solid. Even number of intersections correspond to exterior points, and odd number of intersections correspond to interior points. The assumption of boundaries as manifold cell complexes forces any boundary representation to obey disjointedness of distinct primitives, i.e. there are no self intersections that cause non-manifold points. In particular, the manifoldness condition implies all pairs of vertices are disjoint, pairs of edges are either disjoint or intersect at one vertex, and pairs of faces are disjoint or intersect at a common edge. Several data structures that are combinatorial maps have been developed to store boundary representations of solids. In addition to planar faces, modern systems provide the ability to store quadrics and NURBS surfaces as a part of the boundary representation. Boundary representations have evolved into a ubiquitous representation scheme of solids in most commercial geometric modelers because of their flexibility in representing solids exhibiting a high level of geometric complexity.

Constructive Solid Geometry

Constructive Solid Geometry (CSG) connotes a family of schemes for representing rigid solids as Boolean constructions or combinations of primitives via the regularized set operations discussed above. CSG and boundary representations are currently the most important representation schemes for solids. CSG representations take the form of ordered binary trees where non-terminal nodes represent either rigid transformations (orientation preserving isometries) or regularized set operations. Terminal nodes are primitive leaves that represent closed regular sets. The semantics of CSG representations is clear. Each subtree represents a set resulting from applying the indicated transformations/regularized set operations on the set represented by the primitive leaves of the subtree. CSG representations are particularly useful for capturing design intent in the form of features corresponding to material addition or removal (bosses, holes, pockets etc). The attractive properties of CSG include conciseness, guaranteed validity of solids, computationally convenient Boolean algebraic properties, and natural control of a solid's shape in terms of high level parameters defining the solid's primitives and their positions and orientations. The relatively simple data structure and elegant recursive algorithms have further contributed to the popularity of CSG.

Sweeping

The basic notion embodied in sweeping schemes is simple. A set moving through space may trace or *sweep* out volume (a solid) that may be represented by the moving set and its trajectory. Such a representation is important in the context of applications such as detecting the material removed from a cutter as it moves along a specified trajectory, computing dynamic interference of two solids undergoing relative motion, motion planning, and even in computer graphics applications such as tracing the motions of a brush moved on a canvas. Most commercial CAD systems provide (limited) functionality for constructing swept solids mostly in the form of a two dimensional cross section moving on a space trajectory transversal to the section. However, current research has shown several approximations of three dimensional shapes moving across one parameter, and even multi-parameter motions.

Implicit representation

A very general method of defining a set of points X is to specify a predicate that can be evaluated at any point in space. In other words, X is defined *implicitly* to consist of all the points that satisfy the condition specified by the predicate. The simplest form of a predicate is the condition on the sign of a real valued function resulting in the familiar representation of sets by equalities and inequalities. For example if $f = ax + by + cz + d$ the conditions $f(p) = 0$,$f(p) > 0$, and $f(p) < 0$ represent respectively a plane and two open linear halfspaces. More complex functional primitives may be defined by boolean combinations of simpler predicates. Furthermore, the theory of R-functions allow conversions of such representations into a single function inequality for any closed semi analytic set. Such a representation can be converted to a boundary representation using polygonization algorithms, for example, the marching cubes algorithm.

Parametric and Feature based modeling

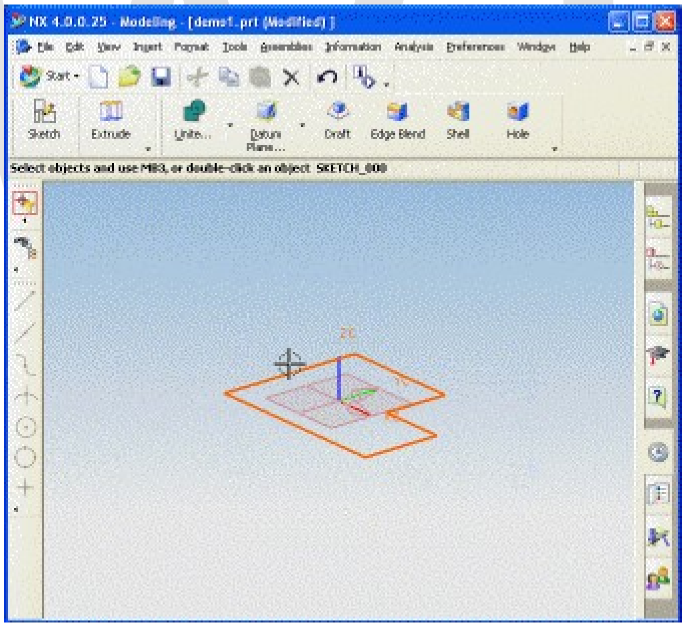
Features are defined to be parametric shapes associated with *attributes* such as intrinsic geometric parameters (length, width, depth etc), position and orientation, geometric tolerances, material properties, and references to other features. Features also provide access to related production processes and resource models. Thus, features have a semantically higher level than primitive closed regular sets. Features are generally expected to form a basis for linking CAD with downstream manufacturing applications, and also for organizing databases for design data reuse.

History of Solid Modelers

The historical development of solid modelers has to be seen in context of the whole history of CAD, the key milestones being the development of the research system BUILD followed by its commercial spin-off Romulus which went on to influence the development of Parasolid, ACIS and Solid Modeling Solutions.

Computer aided design

Solid modelers have become commonplace in engineering departments in the last ten years due to faster computers and competitive software pricing. Solid modeling software creates a virtual 3D representation of components for machine design and analysis. A typical graphical user interface includes programmable macros, keyboard shortcuts and dynamic model manipulation. The ability to dynamically re-orient the model, in real-time shaded 3-D, is emphasized and helps the designer maintain a mental 3-D image.



Select objects and use MB3, or double-click an object SKETCH_000

Creation of a solid model in NX

A solid part model generally consists of a group of features, added one at a time, until the model is complete. Engineering solid models are built mostly with sketcher-based features; 2-D sketches that are swept along a path to become 3-D. These may be cuts, or extrusions for example.

Design work on components is usually done within the context of the whole product using assembly modeling methods. An assembly model incorporates references to individual part models that comprise the product.

Another type of modeling technique is 'surfacing' (Freeform surface modeling). Here, surfaces are defined, trimmed and merged, and filled to make solid. The surfaces are usually defined with datum curves in space and a variety of complex commands. Surfacing is more difficult, but better applicable to some manufacturing techniques, like injection molding. Solid models for injection molded parts usually have both surfacing and sketcher based features.

Engineering drawings are created semi-automatically and reference the solid models.

The learning curve for these software packages is steep, but a fluent machine designer who can master these software packages is highly productive.

The modeling of solids is only the minimum requirement of a CAD system’s capabilities.

Parametric modeling uses parameters to define a model (dimensions, for example). The parameter may be modified later, and the model will update to reflect the modification. Typically, there is a relationship between parts, assemblies, and drawings. A part consists of multiple features, and an assembly consists of multiple parts. Drawings can be made from either parts or assemblies.

Example: A shaft is created by extruding a circle 100 mm. A hub is assembled to the end of the shaft. Later, the shaft is modified to be 200 mm long. When the model is updated the shaft will be 200 mm long, the hub will relocate to the end of the shaft to which it was assembled, and the engineering drawings and mass properties will reflect all changes automatically.

Examples of parameters are: dimensions used to create model features, material density, formulas to describe swept features, imported data (that describe a reference surface, for example).

Related to parameters, but slightly different are Constraints. Constraints are relationships between entities that make up a particular shape. For a window, the sides might be defined as being parallel, and of the same length.

Parametric modeling is obvious and intuitive. But for the first three decades of CAD this was not the case. Modification meant re-draw, or add a new cut or protrusion on top of old ones. Dimensions on engineering drawings were *created*, instead of *shown*.

Parametric modeling is very powerful, but requires more skill in model creation. A complicated model for an injection molded part may have a thousand features, and modifying an early feature may cause later features to fail. Skillfully created parametric models are easier to maintain and modify.

Parametric modeling also lends itself to data re-use. A whole family of capscrews can be contained in one model, for example.

Entertainment

Animation of computer generated characters is, technically, an example of parametric modeling, though few in the industry would consider it to be. Characters' skin is modeled with NURBS patches and stitched together or polygon modeled. The skin of characters is then parametrically associated to a skeleton within characters (with many characters' skins now being driven by muscle simulation systems). The skeleton of a character is rotated into poses, which parametrically drives the shape of the characters' skin for each frame to create animation.

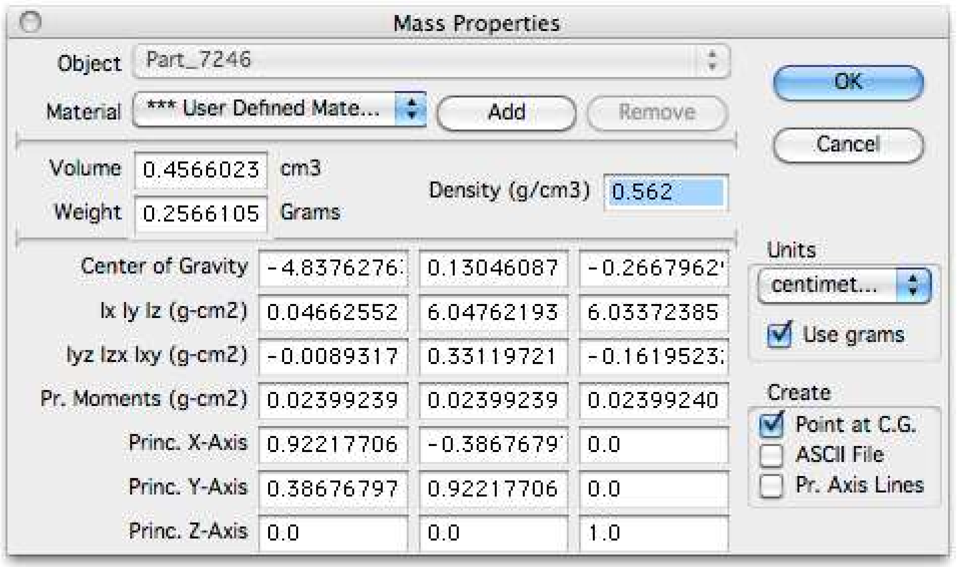
Medical solid modeling

Modern computed axial tomography and magnetic resonance imaging scanners can be used to create solid models of internal body features, so-called volume rendering. Optical 3D scanners can be used to create point clouds or polygon mesh models of external body features.

Uses of medical solid modeling;

- Visualization
- Visualization of specific body tissues (just blood vessels and tumor, for example)
- Designing prosthetics, orthotics, and other medical and dental devices (this is sometimes called mass customization)
- Creating polygon mesh models for rapid prototyping (to aid surgeons preparing for difficult surgeries, for example)
- Combining polygon mesh models with CAD solid modeling (design of hip replacement parts, for example)
- Computational analysis of complex biological processes, e.g. air flow, blood flow
- Computational simulation of new medical devices and implants *in vivo*

If the use goes beyond visualisation of the scan data, processes like image segmentation and image-based meshing will be necessary to generate an accurate and realistic geometrical description of the scan data.



Object: Part_7246

Material: *** User Defined Mate...

Volume: 0.4566023 cm3

Weight: 0.2566105 Grams

Density (g/cm3): 0.562

Center of Gravity	-4.8376276:	0.13046087	-0.2667962'
Ix Iy Iz (g-cm2)	0.04662552	6.04762193	6.03372385
Iyz Izx Ixy (g-cm2)	-0.0089317	0.33119721	-0.1619523:
Pr. Moments (g-cm2)	0.02399239	0.02399239	0.02399240
Princ. X-Axis	0.92217706	-0.3867679'	0.0
Princ. Y-Axis	0.38676797	0.92217706	0.0
Princ. Z-Axis	0.0	0.0	1.0

Units: centimet...

☑ Use grams

Create

☑ Point at C.G.

☐ ASCII File

☐ Pr. Axis Lines

Mass properties window of a model in Cobalt

Because CAD programs running on computers “understand” the true geometry comprising complex shapes, many attributes of for a 3-D solid, such as its center of gravity, volume, and mass, can be quickly calculated. For instance, the cube shown at the top measures 8.4 mm from flat to flat. Despite its many radii and the shallow pyramid on each of its six faces, its properties are readily calculated for the designer, as shown in in the screenshot at right.