Designing with Parametric CAD: Classification and comparison of construction techniques

Jami J. Shah

Dept. of Mechanical & Aero. Engineering, Arizona State University, Tempe AZ85287, USA

Key words: CAD, Parametrics, Geometric constraints

ABSTRACT

This paper gives an overview of the rapid evolution of parametrics, from somewhat slow beginnings in academic labs to mainstream commercial CAD. Although not recognized by the user community, parametric CAD is not a single technique but a set of techniques, each with its own advantages and limitations. Each is supported by different technologies. Most CAD systems allow one to mix and match different parametric approaches for creating part geometry. One objective of this article is to establish a clear terminology of parametric paradigms. An enumeration of parametric capabilities for model creation in current commercial CAD systems is reviewed. The influence of model history and constraint solving techniques on parametric modeling is also examined. Limitations exist in feature validation, unexpected model behavior due to topology changes, and solving coupled constraints at full 3D variational level. The idea of "assembly features" has not yet been exploited in commercial CAD. The discussion in this paper is a generalization of parametric capabilities of commercial systems; prototypes in academic research labs are not discussed.

1. BACKGROUND

When solid modeling based CAD was introduced, it met with resistance from designers because of difficulty of use. It was not until the introduction of parametric based CAD that this resistance began to melt away. There are three claimed benefits of parametrics:

- Automatic change propagation
- Geometry re-use
- Embedding of design/ manufacturing knowledge with geometry

F. Kimura (ed.), *Geometric Modelling*

Parametric technology is not a single technique but a set of techniques. Each technique has some capabilities and limitations. Not all of the techniques provide all of the above benefits. Also, the last benefit has yet to be exploited fully in commercial systems. Common to all parametric techniques is associativity that allows changes to propagate automatically. Parametrization may be done *a priori*, such as in design by features or modeling of part families; *a posteriori*, as in feature recognition, or *concurrently*, as in constraint based design. Most CAD systems provide many alternative methods to create and modify part geometry.

2. HISTORY OF PARAMETRIC CAD

Initial work on form features was related to automatic recognition and extraction from CAD models. Since this paper is not about feature recognition, we do not discuss this line of development. Table 1 shows three aspects of parametrics development. In the first column we have listed implementations of constraint based and feature based systems. As shown in Table 1, the concept of *design by features* was first proposed by Pratt and Wilson (1986) in a CAM-I project. Prototype feature-based modeling systems began appearing in the mid-1980s, mostly in university research labs. Among them were Dixon et al.'s system at University of Massachusetts (Cunningham and Dixon, 1988), Turner and Anderson's QTC at Purdue (Turner and Anderson, 1988), and Shah et al's ASU Features Testbed (Shah *et al.*, 1988). Commercial implementations of feature-based modeling became available around 1990, but features were only construction macros. 2D constraint based sketching were developed independently, initially to support tolerances in CAD (Hillyard & Braid 78, Light & Gossard 82). This was followed by application of constraints for positioning parts in assemblies. In the mid 1990s most major vendors supported constraint and feature-based CAD systems.

The underlying technology for parametrics is constraint solving. In the context of geometric modeling, constraints result in non-linear simultaneous algebraic equations. Work in constraint solving has been done both in conjunction with CAD and also unrelated to it. In general, a parametric problem can be described in terms of a list of variable entities, parameters describing the entities, constraints between the entities and the allowable range for each parameter. The second column in Table 1 shows the entities and constraints used in parametric systems over the years.

The third column shows the solution techniques used in parametric CAD. Simultaneous algebraic equations were solved numerically in early 2D systems. *Constraint programming* (Borning, 1979) was also developed early.

It involves using graphs; each constraint is associated with a subroutine for determining the unknown variable from the known variables.

Table 1 Parametric CAD technologies: Lines of development

	Parametric design	**Constraints**	**Solving Techniques**		
1978		2D constraints on vertex positions	Numerical (Hillyard&Braid, Light&Gossard)	Constraint Programming (Borning)	
1985	Concept of DBF (Pratt& Wilson) Academic systems; (Shah, Anderson, Dixon)	User-equations			Bipartite graph (Serrano)
	Commercial system CIMPLEX	2D constraints on points, lines and circles			
1990	Commercial System: Pro-E		Rule-based (Aldefeld, Suzuki, Shimizu)		
	Constraint based 2D sketching	3D construct. procedures		Procedural (Roller, Solano, vanEmmerik)	Constraint graph (Owen)
		3D constraints on point, line, plane			
		3D constraints on assemblies			Commercial Toolkit (DCM2D)
	Constraint based assembly modeling				Kinematic DOF methods (Kramer)
			Symbolic algebra (Buchanan/ dePennington)		
1995	Most commercial CAD support feature macros				
				Propagational (Bouma, Mäntylä)	3D DCM

Serrano & Gossard (1987) proposed *Bipartite graph* algorithms to find maximal matchings of equations to variables so that each equation is used to solve for one variable. However, these algorithms have difficulty with strongly connected components (or cycles in the graph) and resort to numerical methods (Serrano, 1991). Commercial solvers based on Bipartite Graphs, such as DCM-2D, became available in early 1990. Rule based

approaches can be divided into *degree-of-freedom analysis* and *pattern matching*. Degree-of-freedom methods are faster than pattern matching (Kramer, 1992). The pattern matching methods search for combinations of one free geometric entity, several fixed geometric entities and constraints between them which match a pattern in a rule-base (Aldefeld, 1988; Suzuki *et al.*, 1990; and Shimizu *et al.*, 1991). *Symbolic solvers* based on Buchberger's algorithm for finding Gröbner bases were developed by Davenport *et al.* (1993), and Buchanan and de Pennington (1993). Methods that act directly on geometric entities are divided into algorithmic (graph constructive) and rule-based. Algorithmic approaches operate on *constraint graphs*, either by decomposing them (Owen, 1991) or constructing them (Bouma *et al.*, 1995). Differences between geometric constraint solvers occur in the types of geometric primitives that can be handled, how multiple solutions are treated, whether over/underconstrained systems are allowed, the combinations of constraints that can be handled and the speed of solution. We will see in later sections of the paper that the lack of effective means to get the user desired solution and combining of 2D constraint solving with history based model changes create many problems for users of parametric systems today.

3. CLASSIFICATION OF CONSTRUCTION MODES

Figure 1 shows that there are basically two modes in which one can create/modify the geometry of parts and assemblies in CAD systems: traditional (non-parametric) and parametric. These are further classified into seven distinct model construction approaches, as discussed below.

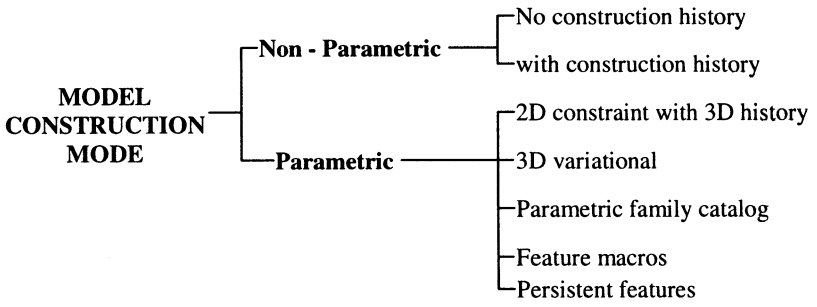


Figure 1

3.1 Non-parametric mode

Non-parametric modes may be implemented with or without construction history. In the traditional method, one creates unconstrained 2D sketches from low level entities, then sweeps or lofts them into 3D objects, and/or uses primitive solids and Booleans. Positioning of objects for Boolean operations or assembly must be done in global coordinates. This makes it tedious to construct objects. It is also difficult to make changes to the geometry, particularly if construction history is not supported. For example, if there was a groove on a hole and we want to change the hole's diameter without changing the groove depth, or to move the hole with the groove, there is no easy way to do it. Some limited editing is possible in history-based non-parametric CAD. Allowable changes are those related to primitive dimensions and positions. Some 3D construction parameters, such as sweep distance, sweep direction, and Boolean operation may also be changed. The model is rolled back to the point of change; the changed primitive is re-created with the new size or position and the rest of the history is rolled back. There is still no associativity between parameters of different primitives or entities of 2D sketches used. In pure solid models, the data stored is at a low level and cannot be directly used in knowledge based applications. The design intent is hard to encode in such models.

3.2 2D constraints with 3D history

We classify parametric methods into 5 distinct classes, shown in Figure 1. There are or two versions of constraint based modeling. The more common approach found in commercial systems involves constraints solving only at the 2D level; 3D construction parameters are edited via the construction history. In this approach, the user-defines a planar topology and specifies dimensional, geometric and algebraic constraints relating the points, lines, arcs of the 2D sketch. The sketch plane can be positioned anywhere in space, or even on one of the existing faces of the solid. If the sketch is under-constrained, the default scale on the screen is used for the unspecified dimensions. The constraint equations are solved in 2D for either point positions or parameters of the geometric entities involved. These 2D sketches are typically used as sections for sweep or lofting operations. When changes are made, the model is rolled back to the 3D operation that involves the change. In many modelers, solids resulting from 3D operations are loosely termed "features". If the change involves parameters in a 3D operation, the construction command is repeated with the new parameters. If the change involves parameters in a 2D sketch, 2D

constraints are solved again, a new sketch is generated, and then the history is replayed. This gives the user the illusion of editing at the 3D level.

Some systems use terms like "smart" or "intelligent" shapes to imply 2D shapes that can be dragged and dropped on another sketch. In case of intersections between the curves of the existing sketch and the dropped sketch, trimming is automatically done at the first intersection. In such cases, outer and inner loops of both sketches are concatenated separately. An example is shown in Figure 2. In (a) we see a sketch (lower figure) and a predefined shape. In (b) we see the predefined shape dragged and dropped on the main sketch. The result, shown in (c) is somewhat like a 2D Boolean operation. This type of capability may be regarded as a hybrid between constraint based and feature based design since it uses a pre-defined sketch from a library of shapes.

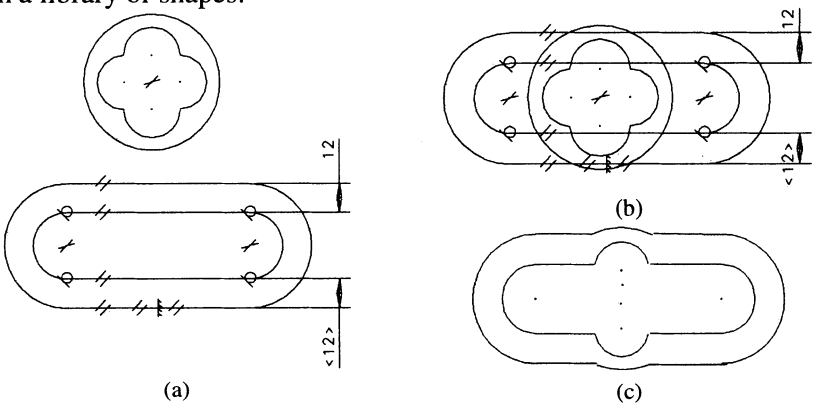


Figure 2

In complex models, users can be overwhelmed by conflicting constraints or under constrained situations. Some hints from the system are helpful. The user needs to know not only whether the sketch is under or over constrained but also which degrees of freedom are available, or which constraints are in conflict. Pure numerical solvers cannot provide this information. Symbolic solvers are useful for this type of functionality.

In 2D constraint based modeling, the construction sequence and parameterization must be carefully thought out prior to modeling because this will restrict the user to what can be edited after the model is built. Constraints cannot be applied or solved at the 3D level. The dependence on history creates many unexpected results, particularly when the reference entity, such as an edge, is split, merged or deleted. An example of such occurrence is shown in Figure 3. When the slot is shrunk away from the front edge, one cannot predict what will happen when the two edges on either side of the slot are merged. One is chamfered, the other is not. In this case the chamfer is applied to the entire edge. In addition, there are the usual

problems with constraints solving such as multiple solutions. Each modeler users its own set to offer heuristic rules. Some recent work has been done in separating constraints into active (used in solving), and passive (used in selecting between multiple solutions (Bettig 2001). For example, the solution that affects the least number of entities is preferred by some solvers. Some constraint solvers solve the equations sequentially, while others solve simultaneously. This can also result in different solutions and different capabilities.

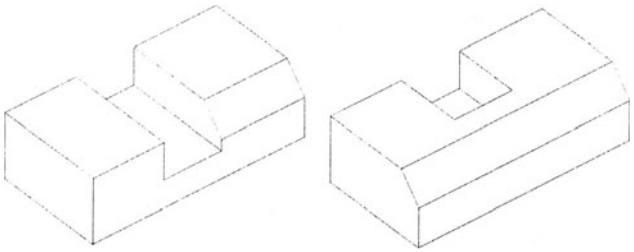


Figure 3

3.3 3D variational design

By 3D variational design we imply the application and solution of constraints on models at the 3D level. However, there is no standard approach for 3D variational design, *per se*. Part positioning in assembly design, and application of constraints between the dimensions of two parts in an assembly, are special cases of 3D constraint based design. The former problem uses fewer degrees of freedom, only those related to position of parts not to their shape or size. The latter can easily be solved procedurally.

Some implementations limit 3D constraints to only entities obtained from linear sweep. Figure 4(a) shows a slot created by linear sweep on a face of a polyhedron obtained by a separate linear sweep. No constraints were used initially. 3D constraints were applied directly to the solid: the side faces of the slot and the bottom were made parallel to the sides and bottom of the outside body (Figure 4.b). Of course, this functionality does not require 3D constraint solving because parallel/perpendicular faces can be mapped to parallel/perpendicular lines in 2D sketches and the constraints solved in 2D.

Figure 5 shows a more sophisticated 3D case. The same part of Figure 4(a) was used but an additional sweep (protrusion) was created on a different face and in a different direction than the slot (Figure 5a). The top face of the protrusion was then made parallel and coplanar with the bottom

of the slot, as shown in Figure 5.b. Even this could be implemented by embedding special procedures and without solving 3D constraints.

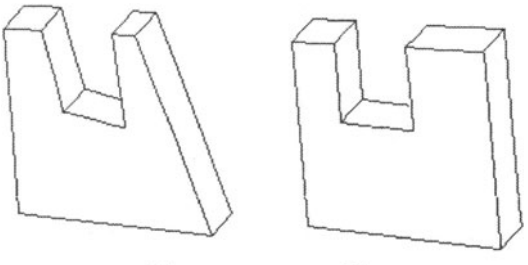


Figure 4

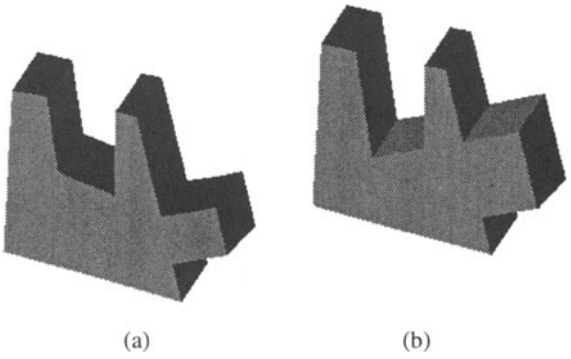


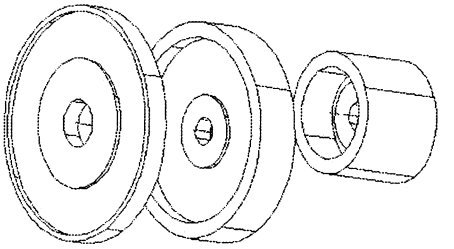
Figure 5

A truly 3D variational system needs a 3D constraint solver; going from 2D to 3D solving has proved to be challenging. Robust solvers are still not available. Thus, CAD systems that support the 3D variational approach are still not commercial. Many vendors have come to the conclusion that it is very difficult for most users to create 3D models directly with this approach. However, the real merit of this approach is in modifying solids that may have been created with or without parametrics.

3.4 Parametric family catalogs

Parametric part family catalogs allow one to pre-define a part shape driven by some key dimensions. An example part family is shown in Figure 6. Using this approach requires a two phase process. First, a parametric model is created by any of the other parametric methods and constrained. A table of standard variants is created using only driving dimensions – the rest are either calculated from these or are fixed. After a catalog is archived,

users can quickly create variants by specifying only the key dimensions or use one of the standard variants. Obviously, the domain of such modeling is very limited.



MX-1 TP-27 PC-42

Part no.	Hole dia	Hub dia.	Outer-dia	Rim-thck.	Rim-width
MX-1	2.0	4.0	8.0	0.25	1.0
MX-6	....				
TP-27	1.5	2.5	6.0	0.5	2.0
TP-45	....				
PC-42	1.0	3.0	5.0	0.40	4.5

Figure 6

3.5 Feature based design

Form features may be thought of as stereotypical shapes that may be encountered frequently in certain problem domains. Therefore, features can be predefined and stored in feature libraries. A features definition may include some or all of the following: topology, topological relations, geometry, geometric relations, dimensional parameters, algebraic relations between parameters, and parametric construction history. A definition containing both the declarative and constructive specifications would be redundant. Usually one or the other is used.

Features could be implemented as *persistent objects* or as *construction macros*. Features that are persistent must maintain all topological, geometric, and parametric conditions. For example, a through hole must always remain through. Features implemented as macros are intended only for initial geometry construction and can be modified in any way after construction. Instancing a feature onto an existing part incorporates its corresponding constraint graph and construction history into the existing model, saving design time. However, the feature does not maintain a separate identity; users can modify them in any way, as if the feature had been built piecemeal rather than as a unit. For example, a hole can be converted to a boss by changing the Boolean operation and the sweep direction. Therefore, we say

that such construction macros are not persistent and cannot be used for downstream applications, such as DFM, DFA, etc.

Unlike constraint based modeling, both types of feature based design requires considerable initial investment in creating feature libraries. Commercial systems do not support a very extensive library of pre-packaged features. This is because there are many special geometries found only on certain kinds of products. Commercial systems only provide some very general shapes, such as counter-sunk/bored holes, dovetail slots, etc. Users can use pre-packaged feature macros or define their own. The functionality of pre-packaged feature macros and vendor feature macros is now the same.

The process of defining feature macors involves creating a parametric solid by one of the methods discussed in the foregoing sections, and specifying the Boolean operation to be used for incorporating the feature into a solid model. In some systems, limits on feature parameters can be specified in terms of inequality constraints. However, these inequalities are set to numerical values and not algebraic relations involving other feature parameters. The feature may be positioned like any other primitive. In some systems one is able to create auxiliary geometry, such as reference lines or planes representing edges and faces with respect to which the feature's position will be defined all on the solid model. Once a library of features has been created, users can begin to instance them by specifying only the independent dimension and position parameters. After the features are incorporated in the solid model they are treated just like any other parametric portion of the model for the purpose of editing. For example, if the feature is created by the 2D constraint and 3D history approach; modifications can be made to any of the constraints of the 2D sketch or any of the sweep parameters.

In feature based design users are able to reduce the number of construction steps by re-using shapes that have been archived. Persistent features can be added or deleted from the model by a single operation; this is true of macros only if a single node in the construction history is involved. When a feature parameter is changed, the construction history is re-executed from the point of change. Because users can change the Boolean operation type, sweep operation parameters, and even the topology of the 2D sketch, features do not retain their original meaning. This is where commercial systems did not follow academic research systems. Additionally, CAD systems do not maintain a concurrent and separate feature model in the database. Therefore, applications that depend on features cannot be used. Another problem that contributes to this loss of functionality is related to inadvertent feature intersections with entities of the existing model, destroying the stereotypical geometry associated with the feature.

In light of the last comment, one might ask if some kind of feature validation mechanism is necessary to enforce the rules and relations archived in the feature's definition. Unlike manifold solid models that must satisfy precise topological relations (Euler-Poincare), feature models have no mathematical basis – they are whatever the creator of the feature library meant them to be. Some academic systems have implemented rule based validation (Shah, 1995).

4. COMPARISON OF CONSTRUCTION PARDIGMS

In this section we compare the six paradigms of geometry identified in this paper. Using solid models without parametrics and without history reminds us why designers considered CAD systems as detrimental to their needs. The systems would only be useful after one had already completed the design on paper and then proceeded to electronically document the design. Even in that case one cannot afford to make mistakes because making changes requires almost as much work as the initial design itself. The addition of construction history allows some limited editing, but the lack of associativity between the geometric entities prevents most changes to be made and to be propagated automatically. These methods are still available in CAD systems and can be used if the parametric methods discussed below are not justified.

3D variational modeling can be considered only a curiosity at the present time. 2D constraint modeling with 3D history does not require any customization on the part of the user-organization. However initial construction of the models is even more tedious than pure solid modeling because one must not only define the geometry but also all the relationships from scratch. Once the initial model is set up, and if it is done skillfully, making changes is quick and easy. Since all the geometric modeling and constraint capabilities of the CAD system are available for this kind of design, the domain of objects that can be model is only limited by the capabilities of the CAD system. Of course, rolling back and rolling forward using the history tree leads to unexpected results, as discussed before.

Parametric part family catalogs have limited and specialized use. They can be used for creating part variants and for standard components, such as fasteners, ball bearings, and pulleys. When catalogs can be used, part definition time is minimal. Of course, some initial effort is needed to create the catalogs in the first place.

The articulation and archival of features (persistent or macros) as stereotypical shapes allows fast definition of geometry (the system works out the details from the stored relations) and design change propagation.

Persistent features facilitate automation of some types of analysis, planning and other applications which use feature characteristics to drive their reasoning. Some problems with features are as follows. Since features are viewpoint dependent, feature mapping and recognition are needed to transform features between viewpoints. There has not been much progress in this direction so far. Also, considerable up-front investment is needed in creating feature libraries. Another problem is that the more specific a feature is, the faster and easier it makes design, change propagation, or applications but the feature becomes less versatile. In other words:

Design efficiency $\propto$ *Information Level*
Feature Versatility $\propto$ *(1/Information Level)*

Consider the connecting rods (Figure 7) that were part of a case study we conducted. 9 features were needed to fully model part A. However, Part B could not be fully modeled with those 9 features. 19 features were needed to model both A and B. 27 features were needed to model an additional group of conrods from other automotive manufacturers. Even though the functions of these parts are identical, and they even appear to be geometrically similar, the number of features proliferates. There are two ways around this problem. One is to make the features less specific and the other is to model some non-recurring portions of the part without features. The latter approach seems prudent in most cases.

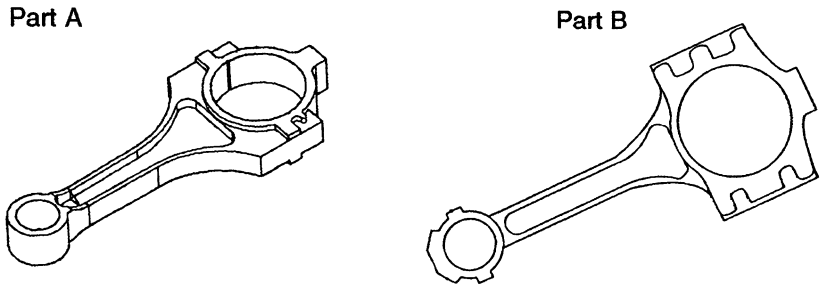


Figure 7

From the above remarks one should not get the impression that features are not versatile at all. Compared to parametric part families, features allow a greater variety of shapes to be generated. All the shapes shown in Figure 8 can be generated with a modest set of features. Because the topologies are so different, these parts cannot be modeled as one part family.

Figure 9 compares four of the six design paradigms with respect to design efficiency, change efficiency, domain and initial investment needed to begin using the method. By design efficiency we mean the time needed to create the initial design. By domain, we mean the variety of parts that can be

created. All parametric methods have high change efficiency but the design efficiency is proportional to investment. 3D variational CAD is not included in these comparisons since it is not available commercially. Also, solid modeling without history is not included since that is *passe* these days.

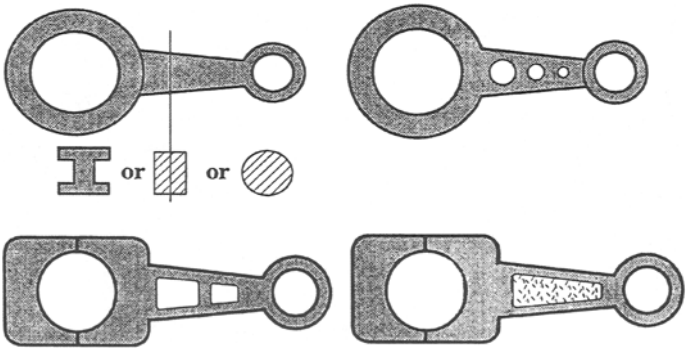


Figure 8

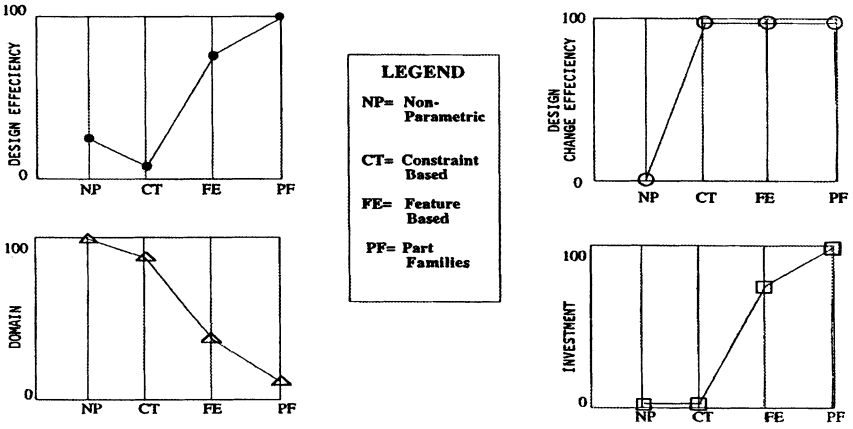


Figure 9

5. ASSEMBLY MODELING

Assembly modeling without parametrics can also be tedious because parts are positioned independently in space. Not only is it difficult to do this, it also retains no relations. Making changes to part sizes after assembly is not possible without repeating the whole process. Constraint based tools (co-axial, co-planar, etc.) are now available to align mating parts via related

geometry. This helps in both assembly definition and design change propagation. Solving for part positions in 3D is relatively easy.

The concept of features has not yet permeated assembly design. Recall that a form feature is a stereotypical shape. So what is an assembly feature? It is a stereotypical assembly "situation". For example, insertion feature or sliding contact feature. We define an *assembly feature* as an association between two form features on different parts (Shah 93). Figure 10 depicts some examples of assembly features. Assembly features encode mutual constraints on mating features' shape, dimensions, position, and orientation.

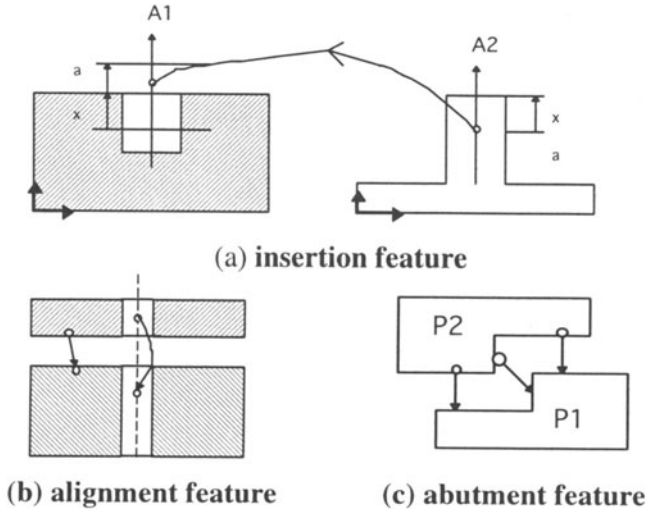


Figure 10

The information that defines an assembly feature consists of both directly specified attributes and derived parameters. Structural relations, positioning rules and constraints, and Fit are examples of directly specified, generic attributes; *Part-of* is an instance parameter; *Dof* and *Limits* are derived. Derived parameters are those that are constrained by something already in the model, while independent parameters are user choices at the time of assembly. For example, the position and orientation of a pin is constrained by the hole in which it is to be inserted, whilst the depth of insertion is user specified. The conditions for mating(Fit) need to be specified also in order to determine if the shape, size, and juxtaposition of features make a given assembly possible. Assembly features need to be parameterized and stored in libraries. During assembly design, one can instance these features and quickly apply them to the model at hand. This kind of modeling has been explored only minimally in academic circles and not all in industry. There are no assembly feature libraries, or user defined assembly features in commercial CAD systems. Assembly models are constructed using the constraint based approach not feature based.

6. CONCLUSION

After years of rapid advances in parametric CAD, recent releases seem stable as these technologies have matured. This paper gives a generalization of parametric techniques in commercial CAD systems. These capabilities are embodied in various forms and to different extents in different systems. Each system has its own terminology, often influenced by marketing hype. These claims and marketing terms are often misleading. This paper attempts to provide a generic terminology based on technical factors and tries to clarify the real differences between parametric paradigms.

From this paper one can see that parametric CAD is not a single method but a collection of construction and validation methods. Each one has its own strengths and weaknesses and hence is appropriate for particular types of design. One needs to look at the application (airfoil lofting, variant machine design, novel design, etc.); the extent of re-usable information, the level of maturity of the product; the type of artifact (door panel, turbine disk, structure, etc.); the frequency of introducing new products; the frequency of design changes. All methods are useful and one typically needs to combine several of them to achieve the desired objectives.

Some of the methods, such as 3D variational design, are not fully developed or widely available. The concept of features has not yet been extended to assemblies, yet few people understand the difference between that and constraint based assembly design which is commonly available. The usefulness of features in assembly design is up for debate.

Research in feature recognition goes further back than design by features or constraint based design. Despite the maturity of this technology and availability of a wide range of robust and efficient techniques (Shah 2001), feature recognition has not been incorporated into mainstream CAD systems. This technology has great potential in automating machining process planning but industry does not seem to have demanded this capability from CAD vendors yet.

7. REFERENCES

Aldefeld, B. , 1988 "Variation of geometries based on a geometric-reasoning method", *Computer -Aided Design*, Vol. 20, No. 3, pp. 117-126

Borning, A., 1979, "ThingLab – A Constraint-Oriented Simulation Laboratory", Technical Report SSL-79-3, XEROX Research Center, Palo Alto, California, July 1979.

Bouma, W., Fudos, I., Hoffmann, C., Cai, J., Paige, R., 1995 "Geometric constraint solver", *Computer-Aided Design*, Vol. 27, No. 6, pp. 487-501

Buchanan, S. A., and de Pennington, A., 1993, "The Constraint Definition System: A Computer Algebra Based Approach to Solving Geometric Problems", *Computer-Aided Design*, Vol. 25, No. 12.

Cunningham, J., and Dixon, J., 1988, Design with Features: The Origin of Features, *ASME Computers in Engineering Conferences*, San Francisco, July/August 1988.

Davenport, J. H., Siret, Y., and Tournier, E., 1993, *Computer Algebra; systems and algorithms for algebraic computation*, Second Edition, Academic Press, New York.

Hillyard, R. C., and Braid, I. C., 1978, Analysis of dimensions and tolerances in computer-aided mechanical design, *Computer-Aided Design* **10**(3):161–166.

Kramer, G.A. , 1992, *Solving Geometric Constraint Systems*, The MIT Press, Cambridge, MA

Light, R. and Gossard, D., 1982, "Modification of geometric models through variational geometry", *Computer-Aided Design*, Vol. 14, No. 4, pp. 209-214

Mäntylä, M., *An Introduction to Solid Modeling*, Computer Science Press, 1988.

Owen, J.C. , 1991 "Algebraic Solution for Geometry from Dimensional Constraints", *Proc. Symposium on Solid Modeling and Applications*, Austin, Texas, ACM press, New York

Pratt, M. J., and Wilson, P. R., 1986, Requirements for support of form features in a solid modeling system, final report, Technical Report R-86-ASPP-01, CAM-I, Inc., Arlington.

Roller, D., 1991, "Advanced Methods for Parametric Design", *Geometric Modelling Methods and Applications*, D. Hagen and D. Roller (Ed.'s), pp. 251-266, Springer-Verlag.

Serrano, D., 1991, "Automatic Dimensioning in Design for Manufacturing", *Proc. Symposium on Solid Modeling Foundations and CAD/CAM Applications*, Austin, Texas.

Serrano, D., and Gossard, D., 1987, "Constraint Management in Conceptual Design", *Knowledge Based Expert Systems in Engineering: Planning and Design*, D. Sriram and R. A. Adey (Ed.'s), Computational Mechanics Publications, Southhampton.

Shah J., Mäntylä, M., 1995, "Parametric & Feature based CAD/CAM: Concepts, Techniques, Applications", J. Wiley, New York.

Shah, J., Rogers, M., 1988, "Expert form feature modeling shell", *Computer aided Design,* **20**(9), 515-524.

Shah J., Rogers M., 1993, "Assembly modeling as an extension of feature based design", Research in Eng. Design, V5, 218-237.

Shah J., Anderson D., Kim Y-S, Joshi S., 2001, "A Discourse on Geometric Feature Recognition", *J. of Computing & Information Science in Eng. (JCISE)*, **V1**(1).

Shimizu, S., Inoue, K., Numao, M., 1991, "An ATMS-Based Geometric Constraint Solver for 3D CAD", *Proc. Third Int. Conf. On Tools for AI*, San Jose, CA, Nov. 1991, IEEE Computer Society Press, Los Alamitos, CA

Solano, L., and Brunet, P., 1993, " A System for Constructive Constraint-Based Modelling", *Modeling in Computer Graphics*, B. Falcidieno, T. L. Kunii, (Ed.'s) Springer-Verlag.

Suzuki, H., Hidetoshi, A. and Kimura, F., 1990, "Geometric Constraints and Reasoning for Geometrical CAD Systems", *Computers & Graphics*, Vol. 14, No. 2, pp. 211-224

Turner, G., and Anderson, D. C., 1988, An object oriented approach to interactive, feature based design for quick turnaround manufacturing, *ASME Computers in Engineering Conf.*, San Francisco, July 31–August 4, ASME Press.

van Emmerik, M., 1991, "Interactive Design of #D Models with Geometric Constraints", *The Visual Computer,* Springer-Verlag, Vol. 7, pp. 309-325.