Computer-Aided Design and Applications

http://www.cadanda.com

Feature-Based Design – An Overview

Tamer M. M. Shahin[1]

[1]UAEU University, Tamer.Shahin@uaeu.ac.ae

ABSTRACT

This paper presents an overview of the research carried out in feature-based design (FBD). Detailed definitions of features are presented and a summary on the work that has been carried out in areas such as feature taxonomies, mapping, validation, constraints in FBD, interfaces, engineering information storing schemes and the object-oriented approach is provided. A way forward is explored and a recommendation for further research is suggested in order to minimize production lead time and cut down capital costs by introducing and implementing methodologies for the automation of complex model creation.

Keywords: Feature-based design, parametric design, design by features, form feature recognition,.
DOI: 10.3722/cadaps.2008.639-653

1. INTRODUCTION

Today CAD workstations are an essential tool in the hands of a design engineer. Indeed, over the last 40 years, computers have revolutionized the way that engineering designs are created. Initially, computers were used as a drafting tool [3]. Progressively, the technological advances in computer software and hardware introduced the computer as a medium to model engineering creations in three-dimensions. "Geometric modeling" became a powerful technique in engineering, providing the ability to determine through different types of modelers the construction sequences, behavior, strengths and weaknesses, and other characteristics of most of the engineering models before they were physically constructed. The next step was FBM (Feature–Based Modeling). Features were introduced in the engineering designers' arsenal of tools that helped bring together design with manufacturing and provided a means for engineers to develop a link between CAD and CAM (Computer-Aided Manufacturing) as well as techniques to automate the procedures [30]. This automation resulted in higher production rates (efficiency) and capital cost reduction. With the aid of modern, state-of-the-art design software, the creation of a part's manufacturing code (i.e. Numerical Control-NC Code) can now be made on the fly. However, it is still a relatively new field and the future success of such computer software packages will depend on the ability of CAD systems to implement new techniques, handle more complexity, and potentially aid in the decision-making in order to ultimately make the design and representation of a product a more accurate and efficient process [41].

Figure 1 illustrates the different areas of research within FBD. This paper presents an overview of the research carried out in feature-based design (FBD). Current methods, definitions and procedures are reviewed and analyzed further so as to determine their drawbacks and to emphasize the need for development of new techniques.

2. FEATURE DEFINITION

There have been various different definitions about what exactly a feature represents. Early informal feature definitions were analyses and process planning oriented. That was because at the time most of the Computer-Aided Design (CAD) applications needed a boost so as to increase the design efficiency with the integration of other applications based on analyses and process planning. An early feature definition was "a specific geometric configuration formed on the

surface, edge or corner of a workpiece, intended to modify outward appearances or to assist in achieving a given function" [39].

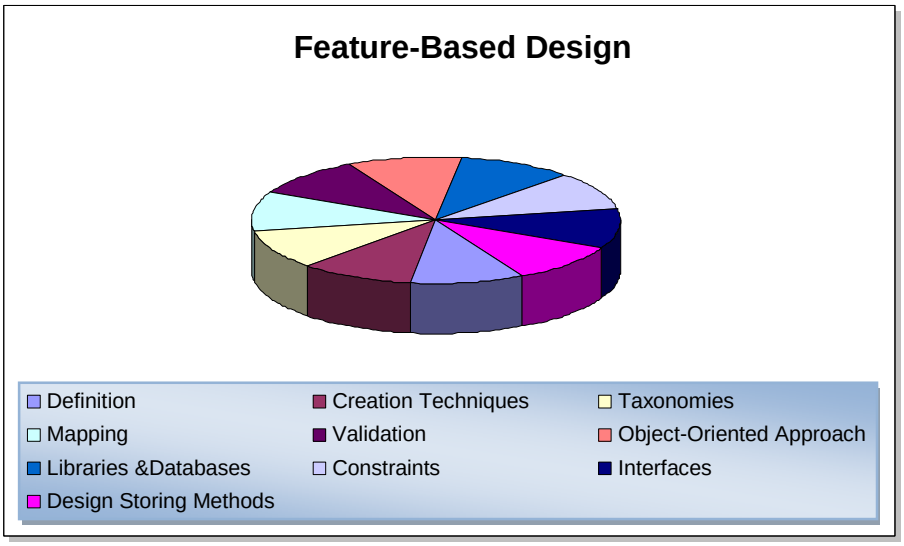


Fig. 1: Feature-based design areas.

Another early feature definition was that "a feature is a geometric form or entity that is used in reasoning in one or more design or manufacturing activities" [39].

More recent definitions have been introduced that can be used in various domains other than process planning. Such a definition is that "features encapsulate the engineering significance of portions of the geometry of a part or assembly, and, as such, are important in product design, product definition, and reasoning, for a variety of applications" [42].

Lamit defines form-features as "shapes that contain engineering knowledge such as, structure, development stage, and parameters such as dimensions and material information and can be used by the designer in order to speed up the design process" [28].

In Chu and Gadh [10] features are classified according to the number of possible tool approaches for machining. A detailed classification of machining features is given by Feng et al [16]. Some definitions (generic and non-generic based on geometric information) are presented below:

- "Features are considered as regions of parts having some manufacturing significance in the context of machining" [17].
- "Recurring patterns of information related to a part's description" [44].
- "A carrier of product information that may aid design or communication between design and manufacturing, or between other engineering tasks" [40].
- "A set of information related to a parts description or relations between parts" [21].
- "Features are protrusions or depressions; a protrusion feature is a convex face set closed by a concave boundary loop whereas a depression feature is a concave face set closed by a convex boundary loop" [49].

Further research began to see a convergence of object-oriented design/models and features. Salomons states 'features can be treated as design objects, belonging to a general class, which inherits properties from other classes'

Because of the nature of features, different information can exist and thus altering their primary role to different application contexts. Some of the different kinds of features are functional features [41] [43], assembly features [42]

[21] [43] [14], analysis features [39], tolerance features [43], technological features [42] [21], material features [42] [44] [43], precision features [42] [44] [21], mating features, abstract features [40] and physical features. Shah however places some generic criteria to fulfil the definition and properties of a feature, "A feature is mappable to a generic shape, is a physical constituent of a part, and has engineering significance and predictable properties" [50].

The use of features nowadays has been transformed to incorporate recent advances. Features are control tools that incorporate dynamic engineering knowledge to aid engineers for design, reuse and manufacturing. They can be used in various different ways and they always result in engineering model generation. It can be seen from the literature that feature definitions broadly are divided into two main categories, one for design features and the other for manufacturing features. Design features are usually defined as sets of geometric entities that represent certain shapes patterns and have certain functions or embedded information. Manufacturing features are usually portions of a workpiece that can be generated with metal removal processes. In general though, a feature is a physical constituent of a part and as a result parts are physical constituents of assemblies. Thus, any feature attribute is a characterisation of parts and their assemblies. Substantial research has gone into the means to create or recognize these features and in the next section, different methodologies for creating and using a feature will be reviewed.

3. FEATURE CREATION TECHNIQUES AND REPRESENTATION

There are three major design and manufacturing strategies/approaches whereby Feature-Based Design can be used. The first approach is called Concurrent Design. In concurrent design the design engineer creates models with respect to their manufacturing processes. As such, there is an optimisation in the production period that helps the engineers to think in advance and be able to modify the original model in case of any possible inadequacies by following a path of iterative procedures. It can be stated that this strategy is mainly process planning based and its main characteristics as stated by Duan *et al* [13] are as follows.

- Design is carried out in manufacturing mode
- The primitives that are used are mainly manufacturing features
- It is suitable for fabricating moulds on matching centers for small batches.

The second approach is called Design for Assembly (DFA). This approach has become one of the favourite in the recent years because of its simplified use. Most of the current DFA software packages are interactive and user friendly. The main target of this approach is to minimise the possible number of parts in an assembly and at the same time minimise lead time, reduce the capital and running costs. These techniques are primarily quantitative or qualitative [55] and use the cost-tolerance design methodology mainly [13].

Finally, the third is the parametric design which is the approach and focus of this paper. It groups similar products into product families helping in that way the designer to create a new or modify pre-existing solid models on the fly. Thus new, and reuse of previous designs can be achieved with a variant approach. This approach has emerged from the integration of CAD with computer-aided process planning (CAPP). Today, it is widely used by the variety of designers and software developers such as Parametric Technology Corporation (PTC), SolidWorks, Unigraphics, Aries, AutoDesk and other.

With the above applications in mind, approaches to Feature-Based Design have progressively merged into two main categories: 'Design by Features' and 'Feature Recognition' [32]. These will be discussed in more detail in the following sections, but essentially, Design by Features is a method of using design features to realize the creation of a CAD model whereby Feature Recognition aims to recognize features from an already completed model. There are two main approaches to Feature Recognition: Interactive Feature Recognition and Form Feature Recognition. Following is a summary of the main research that has been developed in this field. Zhao, Ghosh, and Link [1990] used a wireframe model to develop a methodology and algorithm of recognizing machined features by using graph theory. Kang and Nnaji [26] proposed a feature representation and classification model for an automatic process planning system. Kao and Kumara [25] proposed the super relation graph (SRG) method to recognize machining features such as slots, holes and pockets. Sheu [46] developed a computer integrated manufacturing system for rotational parts. The parametric design and feature-based solid model were used to specify the manufacturing information needed for the proposed system. Venkataraman, Sohoni, and Kulkarni [54] developed a feature recognition system for recognizing User Defined Features (UDF). The feature recognizer used a graph-based approach to represent and recognize features. Nasr and

Kamrani [2006] proposed an intelligent feature recognition methodology (IFRM) to develop a feature recognition system which has the ability to communicate with various CAD/CAM systems.

3.1 Interactive Feature Definition (Hybrid Scheme)

This technique was introduced before even FBD systems were developed (Figure-2). It was used as a means to provide feature data for process planning [3] [42] [43]. The procedure takes place in two steps. Firstly, a geometric model is created or loaded and inserted to a generic geometric modelling interface. The second step involves the interaction of the designer. The user picks entities according to the features that are needed to be specified. These can be edges, vertices, faces combined to form a manufacturing feature. In Thompson et al [52], an interactive software solution (REFAB) that helped to create feature-based models from scanned parts was proposed.

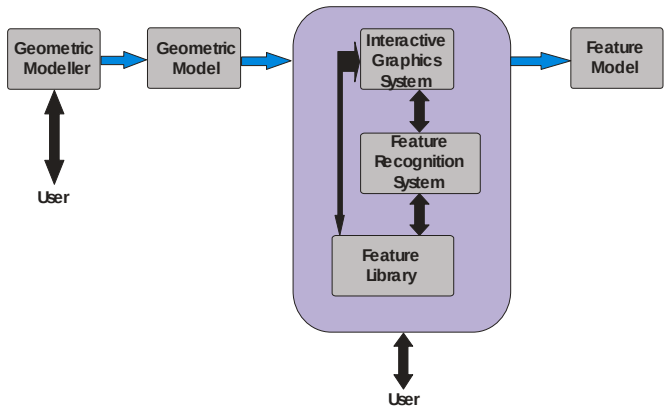


Fig. 2: Interactive feature definition (Shah, 1991).

It is also possible to have libraries with predefined features. In that way, the job of the designer can be simplified even further by having the computer to execute an analysis. The computer searches for possible matches in the database and prompts the user to continue with the feature definition by selecting any remaining entities. For example, considering Figure-3, if one chooses to define the flat bottom hole the computer will subsequently ask the user to define a cylindrical and a planar face. All the necessary calculations which ensure that the result feature is valid are made by the computer automatically.

In both Form Feature Recognition and Design by Features techniques, drawbacks exist [47]. A grouping of the two, on the other hand has been found that eliminates many of their weaknesses. This can be referred to as the "hybrid scheme" for FBD [47].

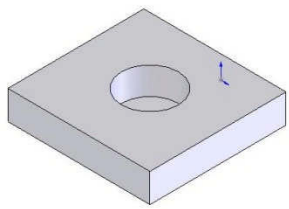


Fig. 3: A Simple flat bottom hole feature on a rectangular part.

A typical example of the hybrid scheme is the system proposed by De Martino et al [12] that enables the user to either start modelling by using features or by a predefined geometric model and create a feature-based model from it using an intermediate model as a mean between the two techniques. Guerra and Hinduja [23], examine a hybrid scheme for axisymmetric turned and milled components. Laako and Mantyla [29] present a hybrid interactive feature-modelling system named EXTDesign.

3.2 Form-Feature Recognition

Often referred to as Automatic Feature Definition (AFD) (Figure-4), this approach was mainly established to fully automate the feature creation or extraction procedure. Features are automatically recognised from an object subjected to consideration [40]. Gindy [22] defined form features as volumes enveloped by entry/exit and depth boundaries and proposed a hierarchical structure for form features classification. Based on the different categories, form features are classified into classes and sub-classes. There are many different classification schemes. Case et al. [7] implemented such a classification for CAD/CAM integration and using a similar feature classification scheme, Xu and Hinduja [57] proposed a feature recognition approach to recognize rough machining features in 2-1/2D components. A boundary-based technique for feature extraction is presented by Qamhiyah et al [36]. A multiprocessor algorithm for fast feature recognition is presented in Regli et al [38]. In Han and Requicha [24] a novel feature finder that uses hint-based feature recognition written in C++ and Lisp is presented. Barwick and Bowyer [4] propose a two-dimensional feature recognition methodology, based on Woodwark's method, that returns possible matches to the user. Similarly, De Floriani and Bruzzone [11] present a feature extraction method based on the topological information contained in a relational model of the boundary of a solid object is examined.

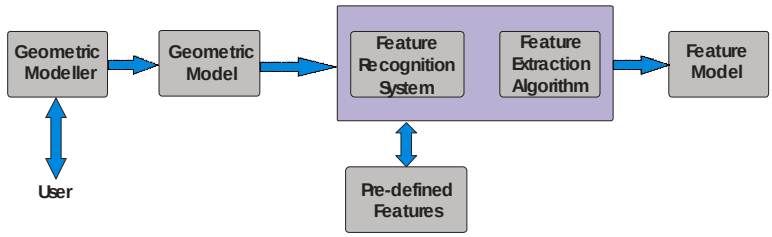


Fig. 4: Automatic feature definition (Andrews, 1999).

Shah [42] [43] proposed a distinction between two different groups of form feature recognition

a) Machining-region recognition
b) Pre-defined feature recognition (of which exists several techniques)

In the former, features are recognised by searching the model using different methods and algorithms as opposed to the latter where a feature library is used in which the software searches until it finds a feature to match to the shape or part of the geometric model. E.g in Figure-5 the computer executes a code that extracts recognised features from the geometric model. This can be achieved by using different sets of algorithms that aim to find similarities between the features that are contained within the predefined libraries and the features of the geometric model. As soon as a match between the two is achieved the features of the part are identified (i.e. a hole feature, a step feature, an open pocket and so on). Generally, in case "a" the computer will try to distinguish (depending on the method and algorithm) the characteristic morphology of the part and thus create its constituent features. On the other hand, in case "b" the computer will scan the part and try to match its characteristics with features that are predefined and stored in its database.

Although the advances in the form feature recognition method has been significant throughout the years, there are various issues that need to be considered and are highlighted in the reference literature. For example, the criteria that are used to identify a feature still depend on the application that is used, and depending on the desired output (model, NC code etc), features do not always represent the processes needed to be identified (e.g machining features). On the other hand, features often interact with each other and this has posed several problems (43a). There is still much work being carried out to develop advanced software package that can distinguish between two, or more, interacting features with success.

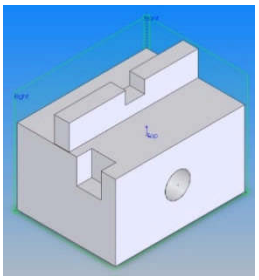


Fig. 5: Sample Part Containing Various Features.

3.3 Design by Features (DbF)

This is a rather manual process in which the user builds a feature-based (geometric) model with the aid of feature libraries. In those libraries, a variety of predefined features are stored in a form accessible by the user through a solid modelling software package. The user can specify dimensions, location parameters and various other attributes and relationships so as to position the feature in the desired location [42] [43]. This can be an advantage to the user in terms of reducing the design lead-time. On the other hand, it is possible to experience difficulties due to insufficient feature entries in the feature library [42] [49].

Shah [42] divides design by features into the following categories. These are:

- Destructive modelling with features (Figure-6)
- Synthesis by features (Figure-7)

Moreover, he mentions a third approach for the sake of completeness named "Feature databases unassociated with solid models".

Destructive modelling consists of volume subtracting operations from a base model so as to create the feature model. On the other hand, synthesis (or constructive modelling), allows the user to add or subtract volumes/features without the need of a base model. In Feru et al [17] a different view of DbF categories is presented.

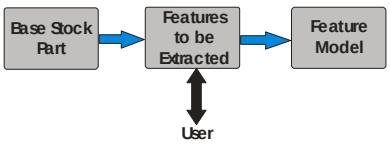


Fig. 6: Destructive modelling with features (Shah and Mantyla, 1995).

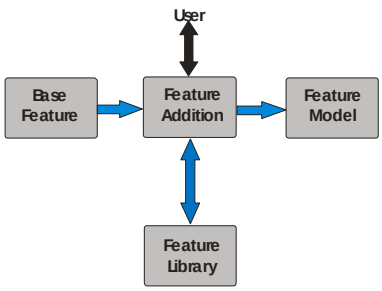
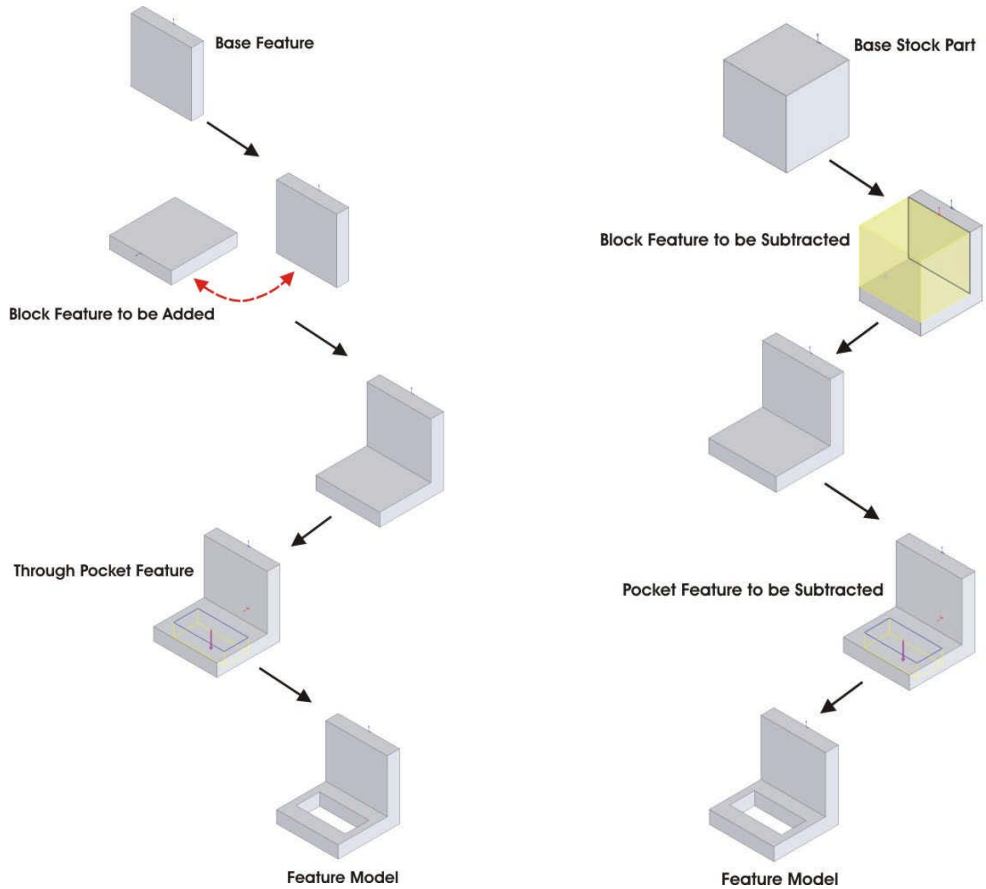


Fig. 7: Synthesis by features (Shah and Mantyla, 1995).

As an example, consider an L-bracket with a through pocket feature (Figures 8a and 8b). In the case of Synthesis by Features (Figure-8a), the user starts off by creating/selecting the base feature. The next step is to create another block feature that will be constrained to the first one so as to form the L-shaped bracket. Finally, a through pocket feature depression will be created and placed into the bracket to form the final feature-based model. However, using Destructive Modelling with Features (Figure-8b), the user starts off by declaring a base stock part. By subtracting the illustrated volume the result L-shaped bracket is formed. Finally, the next step is to subtract the pocket feature by using a cut and thus obtaining the end model.



(a) Synthesis *(b) Destructive Modeling*

Fig. 8: Feature modeling.

4. FEATURE CONCEPTS

This section summarizes the main concepts and area of current research in Feature-Based Design, including, Feature Taxonomy, feature mapping, feature validation and feature libraries and databases. As this paper is an overview in feature-based design, constraint satisfaction problems will not be examined.

4.1 Feature Taxonomy

The use of features in feature-based design has introduced some minor drawbacks. More specifically, because the number of features is not finite, the introduction of classes that consist of features that are related (families of features) had to be made. Furthermore, the relationship between the families of features must be available to the user by means

of computer software. This implies in the introduction of object-oriented programming in the basis of current feature databases.

Taxonomies are specialised categories in which features are classified. The aim is to distinguish different forms of features and make them case specific and so to simplify their use. In CAM-I [51], features at the top end are classified as rotational, non-rotational and sheet and are either volumetric or surfaces. This taxonomy was developed by John Deere. STEP (ISO 10303) (Standard for Exchange of Product Data) is also widely used as a tool for classification and data exchange. Features are distinguished in three main categories; volume, transition and pattern features [42]. Additionally, features have been classified static or kinetic Static features are further sub classified into different categories [42]. Vandenbrande and Requicha [53] also proposed a form feature taxonomy including profiles, holes, slots, grooves and pockets. The primitive features in this taxonomy are concave form features and do not include other types of form features such as a boss on a surface, which is a convex form feature. They defined these primitive features and rules and described ways to combine them into more complex shapes called composite features

To date there is not a standard way that features are classified although there have been various attempts to do so. Most of the researchers propose and develop their own techniques to classify features according to shape, state, significance, application and so on. There is a vast need for a generic feature classification technique that will allow the engineers to easily adopt it and be able to exchange in the best way possible feature information.

Shah and Mantyla [43] stated that there is a possibility to create a universal interface as a result of feature taxonomies that will be supported by different applications. Moreover, the ability to inherit properties from parent features (tree form) would be possible [40]. This could be an advantage over normal feature libraries because it enables the designer to search between related features and feature families when creating new, modifying or altering previous models maximising at the same time the efficiency of the modeller.

4.2 Feature Mapping

Often there is need to export feature-based models into different applications upon their creation. This would be impossible without a successful feature mapping scheme. Essentially, to be able to export models for further analysis such as in FEA (Finite Element Analysis) software or even to produce an NC code that will represent the original model for manufacture, feature mapping needs to take place. In reference [13] feature mapping in a proposed FSMT (Feature Solid Modelling Tool) is presented using the two already discussed downstream applications. Furthermore, in reference [40] a generic mapping shell has been proposed.

Shah and Mantyla [43] define feature mapping as:

- a “catchall phrase”
- covers various different types of transformations between feature models

Since the derivation of feature models from other feature models is desirable a lot of research is focused in the field of feature mapping or else feature transformation. There are a number of ways that mapping processes have been defined in the past. The main problem is that because of the different nature of features (i.e. manufacturing, form, design etc) a wide and complicated range of feature mapping is needed. For example, there are several ways when from a feature model one needs to convert into a machining feature model so as to develop the NC code to automate the construction process of the potential part/assembly. Duan et al propose a mapping strategy that can be applied into analysis and manufacturing. He suggests that a 3-D mesh can be generated in a parallel way with that of feature generation. Firstly a 2-D mesh is obtained on the 2-D shape of the feature and then it is swiped to produce the 3-D mesh. Likewise, any of the sweeping processes can be translated as a manufacturing operation to assist the NC programming. A more generic mapping technique named ASU has been developed and it is based between application specific feature cases [44].

It can be said that feature mapping is a general concept. This can be understood if one thinks of the infinite number of existing feature applications. For these very different applications the mapping procedure will always vary. As with feature taxonomies, feature mapping lacks from a fully specified generic procedure. It is essential for one to be able to convert any potential model into a different feature-form to follow a pre-specified generic methodology so as to maintain feature information that is necessary.

4.3 Feature Validation

The validity of form-features has to be accomplished. This is very important so as to be able to define features with success. There are certain cases in which features overlap with each other. Additionally, features often have to be deleted or modified. In that situation feature validation is necessary.

There are not any specific ways of validating a feature. Thus, there are frequently circumstances where the result operations might give a valid solid but with invalid features [43]. Salomons et al [40] state that features should be application dependent when trying to handle feature interactions. On the other hand, in references [55] [27] believe that invalid feature models arise from the application of feature operators because the corresponding changes are results of feature operations. Han and Requicha [24] propose a hint based approach for feature validation. Case and Hounsell [6] propose a validation scheme focused on the designer's intents. They introduce FRIEND (Feature-based validation Reasoning for Intent-driven Engineering Design), a mechanism that performs feature validation driven by the designer's intents. It is stated that with the aid of FRIEND, one can represent, capture, manipulate and use the designer's intents during the design process as opposed with the use of Boolean operations. When complex objects are introduced for validity checking the process turns to a composite validation task. In the majority of software packages, validation is an iterative as well as an interactive procedure in which question loops are presented to the user until feature validity is satisfied.

4.4 Features and Object-Oriented Approach

The use of object-oriented programming has increased steadily in the recent years. This has resulted from the need to develop software packages that integrate design and manufacturing activities [30] [40].

Object-oriented programming is centred to classes and objects. The inheritance between classes helps to the good representation of feature libraries. Objects can hold information as well as procedures [43]. If some of the objects hold matching variables they form a class. A class may have several subclasses in a tree-form. It is essential to mention that the latter inherits major variables or procedures from the former. Most of the object-oriented programming is done in object-oriented languages such as C++ or Visual Basic.Net.

The main characteristics of object orientation as stated by Latif and Hannam [30] are the following:

- Information hiding (encapsulation)
- Inheritance via classes
- Data abstraction (polymorphism)

4.5 Libraries as a Mean for Storing Feature Information

The need for accurate representation of real life engineering models is immense. In feature-based modelling, it is common sense that the result models are described in terms of features. In all cases of feature-based modelling (DbF, FFR and IFR), features must be included in the modelling software. In the case of automatic feature recognition, the computer via some form of interface communicates with a library of predefined features, from which chooses the best possible solution for each specific model bearing in mind topological information and possible design constraints. When designing by features, features that are available to the user have been predefined. In the majority of modern engineering applications it is possible, for one, to specify new features (user-defined). There are some feature sets in several software packages that are categorised according to specific standards. The standardisation of features was initiated by CAM-I and the United States Air Force (USAF) [40]. In general there are a number of different ways to build a feature library and another to represent features. An example of the latter is that in most cases a feature is described by a set of parameters that in some cases are constrained, a unique identity and type of representation, sets of attributes and a set of procedures for feature manipulation [12]. Shah and Rogers [44] describe a procedure for building a feature library. Further information on feature libraries can be found in references [30] [21] [33] [35].

It can be said, that feature library structure is fundamental when dealing with feature-based design. As feature models become complex so do the feature libraries. The most used programming language for the creation of such libraries is C++. That is because of it is object-oriented and facilitates the creation of feature families in tree forms. Also, it enables the storage of information and procedures. The main characteristics of the object-oriented approach are described in outlined in section 4.4.

The area of interest in feature libraries must expand. When dealing with complex multi-part assemblies that contain various user-defined features difficulties occur. It is very difficult for one to alter, for example, a specific feature and not to have problems in the product model. Of course, in such cases the need for constraint satisfaction is crucial, but in this paper we are focusing on the feature viewpoint. When interfacing with commercial packages feature libraries already predefined and stored within these programs. But because the main aim of Feature-Based Modelling is the boost of automation of engineering models, it is important to create new "state of the art" procedures that will help the accomplishment of this task.

If the designer can make use of predefined feature-based parts that can be fully adjustable to specific requirements, the modelling process will be simplified even further. The problem that arises is that for every different task (i.e. model), different sets of feature parts would be needed. But this technique is aimed for medium to large enterprises. For example, a vehicle company that makes use of feature-based design has several different departments. Every department creates specific models (i.e. engines, pumps, electronic instruments etc.). Thus the majority of parts that are used are of standard nature. Even if this is not the case, the software supported creation of feature-parts can be implemented within a "state of the art" computer software package. An implementation of such system will have a great impact in the cost minimisation and improvement of lead time.

4.6 Constraints in Feature-Based Design

Detailed analysis of constraints as a part of FBD is beyond the scope of this paper. However, it is a very important aspect of FBD and following is a brief summary of existing research in the field.

Constraints are more or less integrated with the use of features. In every different family or set of features the topological information is the same but geometric relations and parameters may vary according to the application. To be able to express this variation of relations and parameters, so as to result to a different working model, constraints are needed. On the other hand, to retain the history of a model for future reference or modification, the use of constraints and operations is fundamental. This is because the modeller must "know" the history in order to allow the user to modify the model without having to re-construct or even start from scratch (i.e. modification of a base feature in a complex model where parts are interrelated would not be feasible). Solano and Brunet, [48] in their research, state that the use of constraints can lead to the description of dependencies between objects and components within them.

There are several different types of constraints. In computer-aided design, focus of researchers on geometric constraint satisfaction is great. The constraint satisfaction description is held by producing a number of non-linear equations. Several approaches have been proposed that aim to satisfy the geometric constraint solution. The main classification can be distinguished to symbolic and numerical approaches. In the former, algebra techniques are used to determine the best possible way that equations can be solved for geometric constraint satisfaction. In the latter, the set of non-linear equations is solved numerically [1]. More information on constraint-based design, constraint satisfaction and naming schemes can be found in references [5] [2] [15] [19].

4.7 Interfacing

Modern commercial feature-based modellers, rely on the integration of several modules in order to enrich their capabilities. Multi role modellers include add-on features, such as finite element analysis modules, automation of the production of engineering drawings (i.e. two-dimensional drafting), numerical control code construction, rapid prototyping, mould creation etc.

The majority of research in the field is aimed in the creation and development of systems that can perform the task that are designed for as independent software or add-on modules to existing software packages. Examples can be found in references [13] [33] [9]. Although most of the currently available applications and systems work by using an object-oriented structure (i.e. C++) there is not a standardised way of interfacing modules. In most cases a unique way of communication has to be designed for every system. Also, compatibility issues arise when different versions of software are launched into the market.

4.8 Computerization Process

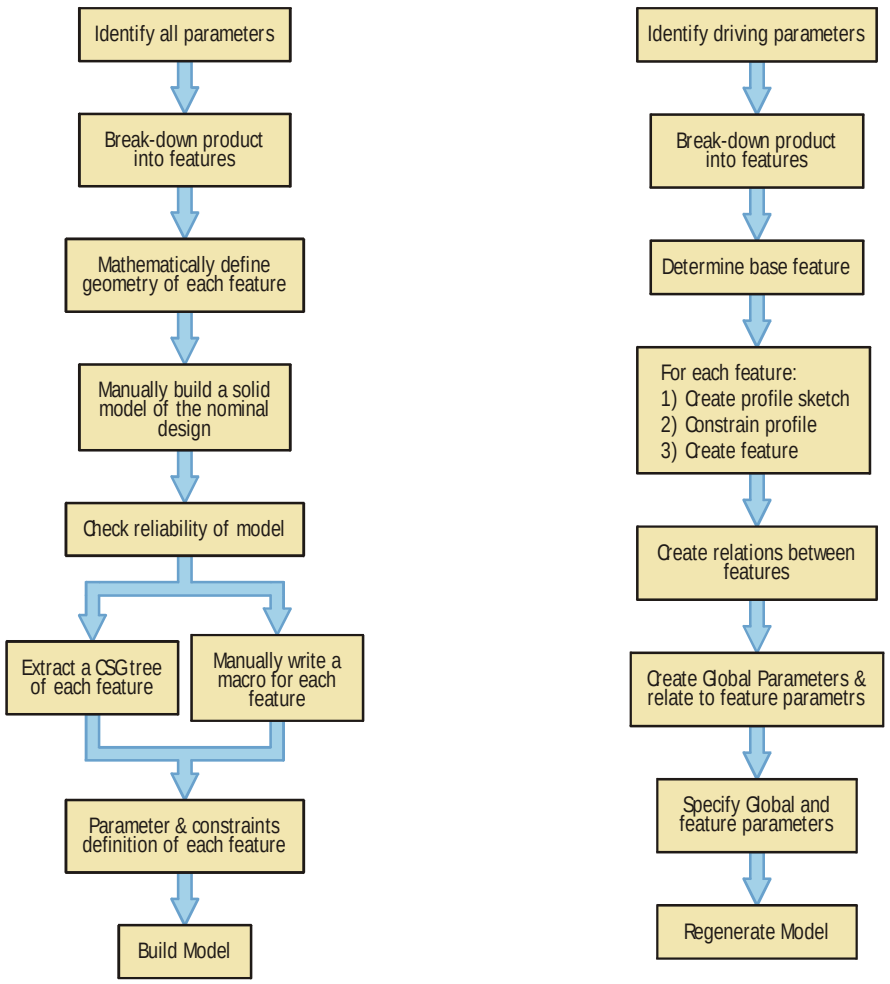
One of the main difficulties in traditional and old companies is to computerise existing paper-based drawings. Moreover, traditional models and engineering drawings created during recent years need to be converted into feature-

based. This is a time and money consuming process. It has become possible to convert computer pre-existing computer models to feature-based by using techniques such as feature-recognition described in chapter 3. Additionally, the use of old designs into the creation of new ones becomes feasible (i.e. design reuse). As a relatively new research field, further research is still needed to fully realise the potential of this area. Some preliminary work is found in [3]

5. FEATURE-BASED DESIGN REUSE

It is essential to be able to store engineering designs that can be used as a reference for the future. For centuries, engineers have relied on past designs to develop and adapt new ones. Semi-automated methods for capturing detailed designs are an attempt to incorporate the advantages of retaining a high level on design intent, whilst using techniques, such as Parametric and Variational Design and Feature Based Design, to automate the generation (or instancing) of similar designs, i.e. its variants.

The two principal techniques for the semi-automated capture of past (and the creation of new) designs, are the Generative (sometimes called Procedural) and Variant Design Methods (Figure 9).



(a) *Generative Methodology* *(b) Variant Methodology*

Fig. 9: Generative and variant feature-based design.

5.1 Generative Methodology

This method adopts a procedural technique to create a parametric model for a given design. The Generative Model is essentially a sequential list of events, or instructions, that represent the design's construction process. Real numbers, representing geometry, are replaced with variables, by editing this data structure. Other parameters, not necessarily relating to geometry can also be added. Individual instances can then be generated by declaring values for these variables and parameters, and then re-executing the procedural data structure. Shahin [45] encompasses the generative method and outlines a methodology to create a series of similar solid models from a single Generative Model, with a goal towards design optimisation. Figure 9a outlines the relevant sections of this methodology, which is categorised by the following three objectives:

Objective 1 - defines various elements of the design that are related to design intent, e.g. parameters, features and constraints. Note that the user of this system is required to manually define the geometry, constraints and relations for geometric elements and features.

Objective 2 - is concerned with the creation of a reliable model. The nominal solid model should be the best possible representation of all instances that are to be generated.

Objective 3 - describes a scheme to explicitly model each feature of the nominal model by, either writing an application-specific macro, or by extracting its representative data structure. This is then edited to include parameter definitions, constraints and relationships. Finally individual models are instanced by assigning a new set of parameter values and re-generating the model.

Clearly the process of manually identifying parameters and features that form a given design is a distinct representation of design intent. Also, having to mathematically define these features places an intent retaining emphasis upon how their related elements will react when new parameters are declared. In his research, Shahin makes use of a hybrid CSG/B-Rep data structure, as the basis of his Generative model. This is formed by creating a nominal design using a suitable CAD modelling application. The hybrid data-structure may then be extracted, if such a feature is available within the application, and edited to include variables (or parameters) in the place of numerical geometry.

5.2 Variant Methodology

Although similar in operation, the Variant approach to storing solid geometric models differs primarily in the construction of its models. Whereas the generative approach involves the often tedious operation of editing a complex data structure to enable parameterisation, the Variant Method makes use of Parametric and Variant Modelling techniques [43][20]and Feature Based Design, in particular User Designed Features to interactively draft a geometric model. It requires virtually no complex mathematical and programming operations, and is typically implemented via an efficient and familiar user-interface. Despite the difference in terminology, perhaps the most well known commercial example of this technique is the 'Parametric Modeller', Pro/ENGINEER, which was pioneered in 1990. More recently other applications vendors have adopted this technique, including Autodesk, with 'Mechanical Desktop' as an extension to AutoCAD, and 'SolidWorks'.

The process of creating a variant model is initially similar to that of the generative model. Where, to begin with, the driving design parameters and features, are identified. The majority of current modelling systems work with a 'Synthesis by Features' approach, where features are constructed in a hierarchical fashion, thereby requiring the creation of a base feature. Here, features are created by either using predefined, library features (primitives), or by generating User Defined features, which involves the creation of a 2D sketch (or profile) for each feature, which is parametrically dimensioned and constrained. (If this feature is not the base feature, then its profile must also be constrained to its parent feature, e.g. the base). This profile is then transformed, typically by parametric extrusion, to form a solid feature model. And the process is repeated for all identified features in the design. Finally, relations between features can be established. These generally govern the control of a given feature's driving parameters, and can be either on a feature to feature basis, or defined globally. In this case a set of global parameters is typically created to oversee the declaration of (subordinate) feature parameters. Variants of the model can now be instanced by modifying features and global parameters, and re-solving the models constraint set (regeneration). A further feature, that is typical in many variant design systems, is the ability to momentarily hide, or Suppress, various child features, and Resume these features when desired.

5.3 A Comparison of Generative and Variant Design Methods

These two methods are divided by a fundamental difference in their creation. The Generative Method employs a programmatic approach, whereas the Variant Method provides a more naturally, concurrent approach. However, generative models are highly customisable. This is very favourable in the case of attempting to combine a number of topologically dissimilar designs within a single model. Here the generative model can be programmed to switch between various features depending upon which individual design is required. Trying to attempt this problem with the variant method is difficult, as the variant method inherently 'varies' a given model, and cannot invoke and respond to yes/no decisions, by itself.

Variant based modelling systems are, on the whole much simpler to use than their generative counterparts. They also require less human resources to create a 'parametric', or adaptive, model for a given design. Furthermore, such systems based (even partially) on Flexible Constraint Satisfaction techniques, allow for faster model regeneration, as here only the modified and directly related features and entities are updated. However, innovative application methods have to be developed to exploit this power.

6. CONCLUSIONS

Feature-based design is now a well established research area, beginning from feature recognition for manufacturing purposes and developing into a broader method for designing by features. Most recently attempts to formalize its processes and representation have been attempted with more focus being recently employed towards design intent and manipulation of one model into multiple instances of a family of products. However it is still very much limited to rather simple assemblies. Implementation of more complex geometric constraints, a formalized mapping and taxonomy system as well as the introduction of web-based interfaces for complex mechanical assemblies are still at an early research phase and a framework combining all of these is proposed as the next step to progress this field further.

7. REFERENCES

[1] Anantha, R.; Kramer, G. A.; Crawford, R. H.: Assembly modelling by geometric constraint satisfaction, Computer-Aided Design, 28(9), 1996, 707-722.

[2] Anderl, R.; Mendgen, R.: Modelling with constraints: theoretical foundation and application, Computer-Aided Design, 28(3), 1996, 155-168.

[3] Andrews, P. T. J.; Shahin, T. M. M.; Sivaloganathan, S.: Design Reuse in a CAD Environment: Four Case Studies in Design Reuse. Computers in Industry Journal, 37(1-2), October 1999, 105-109.

[4] Barwick, S. P.; Bowyler, A.: Multidimensional set-theoretic feature recognition, Computer-Aided Design, 27(10), 1995, 731-740.

[5] Capouleas, V.; Chen, X.; Hoffmann, C. M.: Generic naming in generative, constraint-based design, Computer-Aided Design, 28(1), 1996, 17-26.

[6] Case, K.; Hounsell, M. S.: Feature modelling: A validation methodology and its evaluation, Journal of Materials Processing Technology, 107, 2000, 15-23.

[7] Case, K; Gao, J. X.; Gindy, N. N. Z.: The implementation of a feature-based component representation for CAD/CAM integration, J Engng Manufact, IMechE, 208, 1994, 71–80.

[8] Chieh-Yuan, Tsai; Alec Chang, C.: A two-stage fuzzy approach to feature-based design retrieval, Computers in Industry 56, 2005, 493–505

[9] Choi, D. S.; Lee, S. H.; Shin, B. S.; Whang, K. H.; Yoon, K. K.; Sarma, S. E.: A new rapid prototyping system using universal automated fixturing with feature-based CAD/CAM, Journal of Materials Processing Technology, 113, 2001, 285-290.

[10] Chu, C. C. P.; Gadh, R.: Feature-based approach for set-up minimisation of process design from product design, Computer-Aided Design, 28(5), 1996, 321-332.

[11] De Floriani, L.; Bruzzone, E.: Building a feature-based object description from a boundary model, Computer-Aided Design, 21(10), 1989, 602-610.

[12] De Martino, T.; Falcidieno, B.; Giannini, F.; Hassinger, S.; Ovtcharova, J.: Feature-based modelling by integrating design and recognition approaches, Computer-Aided Design, 26(8), 1994, 646-653.

[13] Duan, W.; Zhou, J.; Lai, K.: FSMT: a feature solid-modelling tool for feature-based design and manufacture, Computer-Aided Design, 25(1), 1993, 29-38.

[14] Falcidieno, B.; Giannini, F.; Porzia, C.; Spagnuolo, M.: A uniform approach to represent features in different application contexts, Computers in Industry, 19, 1992, 175-184.

[15] Feng, C. X.; Kusiak, A.: Constraint-based design of parts, Computer-Aided Design, 27(5), 1995, 343-352.

[16] Feng, C. X.; Kusiak, A.; Huang, C. C.: Cost evaluation in design with form features, Computer-Aided Design, 28(11), 1996, 879-885.
[17] Feru, F.; Cocquebert, E.; Chaouch, H.; Deneux, D.; Shoenen, R.: Feature-based modelling: state of the art and evolution, Manufacturing in the Era of Concurrent Engineering, 1992, 29-50.
[18] Fua, M. W. et al.: An approach to identify design and manufacturing features from a data exchanged part model, Computer-Aided Design, 35, 2003, 979–993.
[19] Fudos, I.; Hoffmann C. M.: Constraint-based parametric conics for CAD, Computer-Aided Design, 28(2), 1996, 91-100.
[20] Fuha, J. Y. H.; Lib. W. D.: Advances in collaborative CAD: the-state-of-the art, Computer-Aided Design, 37, 2005, 571–581.
[21] Gardan, Y.; Minich, C.: Feature-based models for CAD/CAM and their limits, Computers in Industry, 23, 1993, 3-13.
[22] Gindy, N. N. Z.: A hierarchical structure for form features, Int J Prod Res, 27, 1989, 2089–103.
[23] Guerra, A. R. O.; Hinduja, S.: Modelling turned components with non-axisymmetric features, Computer-Aided Design, 29(5), 1997, 343-359.
[24] Han, J. H.; Requicha, A. A. G.: Integration of feature-based design and feature recognition, Computer-Aided Design, 29(5), 1997, 393-403.
[25] Kao, C-Y.; Kumara, S. R. T.; Kasturi, R.: Extraction of 3D object features from cad boundary representation using super relation graph method. IEEE Transaction on Pattern Analysis and Machine Intelligence, 17(12), 1995, 1228–1233.
[26] Kang, T. S.; Nnaji, B. O.: Feature representation and classification for automatic process planning systems, Journal of Manufacturing System, 12(2), 1997, 133–145.
[27] Kim, C.; O'Grady, P. J.: A representation formalism for feature-based design, Computer-Aided Design, 28(6/7), 1996, 451-460.
[28] Lamit, L. G.: Pro/ENGINEER 2000i2, Brooks/Cole, USA, 2001.
[29] Laako, T.; Mantyla M.: Feature modelling by incremental feature recognition, Computer-Aided Design, 25(8), 1993, 479-492.
[30] Latif, M. N.; Hannam, G.: Feature-based design and the object oriented approach, Journal of Engineering Design, 7(1), 1996, 27-37.
[31] Li, W. D.; Ong, S. K.; Fuh, J. Y. H.; Wong, Y. S.; Lu, Y. Q.; Nee, A. Y. C.: Feature-based design in a collaborative and distributed environment, Computer Aided Design, 36(9), 2004, 775–97.
[32] Lin, A. C.; Lin, S. Y. Lin; Cheng, S. B.: Extraction of manufacturing features from a feature-based design model, International, Journal of Production Research, 35(12), 1997, 3249–3288.
[33] Mantripragada, R.; Kinzel, G.; Altan, T.: A computer-aided engineering system for feature-based design of box-type sheet metal parts, Journal of Materials Processing Technology, 57, 1996, 241-248.
[34] Nasr, E. S. A.; Kamrani, A. K.: A new methodology for extracting manufacturing features from CAD system, Computers & Industrial Engineering, 51, 2006, 389–415.
[35] Pedley, A. G.: The potential to exchange feature models with user defined feature libraries, Journal of Materials Processing Technology, 61, 1996, 78-84.
[36] Qamhiyah, A. Z.; Venter, R. D.; Benhabib, B.: Geometric reasoning for the extraction of form features, Computer-Aided Design, 28(11), 1996, 887-903.
[37] Raimundo, R.; da Cunha, M.; Dias, A.: A feature-based database evolution approach in the design process, Robotics and Computer Integrated Manufacturing, 18, 2002, 275–281.
[38] Regli, W. C.; Gupta, S. K.; Nau, D. S.: Towards multiprocessor feature recognition, Computer-Aided Design, 29(1), 1997, 37-51.
[39] Roller, D.: Design by features: an approach to high level manipulations, Computers in Industry, 12, 1989, 185-191.
[40] Salomons, O. W.; Van Houten, F. J. A. M.; Kals, H. J. J.: Review of research in feature-based design, Journal of Manufacturing Systems, 12(2), 113-132.
[41] Schulte, M.; Weber, C.; Stark, R.: Functional features for design in mechanical engineering, Computers in Industry, 23, 1993, 15-24.
[42] Shah, J. J.: Assessment of features technology, Computer-Aided Design, 23(5), 1991, 331-343.
[43] Shah, J. J.; Mantyla M.: Parametric and Feature-Based CAD/CAM, John Wiley & Sons, Inc., USA, 1995.
[44] Shah, J. J.; Rogers, M. T.: Expert form feature modelling shell, Computer-Aided Design, 20(9), 1988, 515-524.

[45] Shahin, T. M. M.: Automated Feature-Based Modeling for Finite Element Analyis and Optimisation, PhD Thesis, Brunel University, December, 1996
[46] Sheu, J. J.: A computer integrated manufacturing system for rotational parts, International Journal of Computer Integrated Manufacturing, 11(6), 1998, 538–547.
[47] Sheu, L. C.; Lin, J. T.: Representation scheme for defining and operating features, Computer-Aided Design, 25(6), 1993, 333-346.
[48] Solano, L.; Brunet P.: Constructive constraint-based model for parametric CAD systems, Computer-Aided Design, 26(8), 1994, 614-621.
[49] Suh, H.; Ahluwalia, R. S.: Feature modification in incremental feature generation, Computer-Aided Design, 27(8), 1995, 627-635.
[50] Shah, J. J.; Sreevalsan, P.; Billo, R.; Mathew, A.: Current status for features technology, report for task 0, Technical Report R-88-GM-01, CAM-I, Inc., Arlington, TX.
[51] Technical Report: CAM-I. Requirements for support of form features in a solid modeling system. Technical Report R-85-ASPP-01, Computer Aided Manufacturing International, Inc., Arlington, TX, 1985.
[52] Thompson, W. B.; Owen J. C.; James, H.; Stark, S. R.; Henderson, T. C.: Feature-based reverse engineering of mechanical parts, IEEE Transactions on Robotics and Automation, 15(1), 1999, 57-66.
[53] Vandenbrande, J. H.; Requicha, A. A. G.: Spatial reasoning for automatic recognition of machinable feature in solid models, IIE Trans Pattern Anal Mach Intell, 15, 1993, 1–17.
[54] Venkataraman, S.; Sohoni, M.; Kulkarni, V.: A graph-based framework for feature recognition, Proceedings of the Symposium on Solid Modeling and Applications, Ann Arbor, MI, 2001, 194–205.
[55] Wang N.; Ozsoy T. M.: A scheme to represent features, dimensions and tolerances in geometric modelling, Journal of Manufacturing Systems, 10(3), 233-240.
[56] Wu-Hon F. Leung: Program entanglement, feature interaction and the feature language extensions, Computer Networks, 51, September 2006, 480–495.
[57] Xu, X; Hinduja, S: Recognition of rough machining features in 2-1/2D components, Computer-Aided Design, 30, 1998, 503–16.
[58] Zhao, Z.; Ghosh, S. K.; Link, D.: Recognition of machined surfaces for manufacturing based on wireframe models, Journal of Materials Processing Technology, 24(1), 1990, 137–145.