A Framework for Similarity Recognition of CAD Models in Respect to PLM Optimization

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

vorgelegt von:

Leila Zehtaban
aus Kermanshah

Hauptberichter: Univ.-Prof. Hon.-Prof. Dr. Dieter Roller
Mitberichterin: Univ.-Prof. Dr. Dr.-Ing. Dr. h. c. Jivka Ovtcharova

Tag der mündlichen Prüfung: 12.11.2021

Institut für Rechnergestützte Ingenieursysteme der Universität Stuttgart
2022

Abstract

The profitable prospective of optimizing a new product design with re-using technical knowledge of already manufactured products has exceedingly attracted interests of manufacturers as well as designers. Re-using accumulated know-how has the potential to improve product quality, shorten the design lead-time and reduce costs. One of the key factors for accessing such an already existing knowledge is retrieval of similar products know how in which classification and similarity recognition are its pillars.

In CAD phase the concepts of similarity recognition and feature based design are already known to offer design efficiency, merely with focusing on optimizing the design phase. In this dissertation it is proposed to perform similarity recognition in a bigger nutshell, over a comprehensive set of data comprising engineering as well as marketing data, beyond the geometry data. Applying this approach will optimize the entire PLM encompassing the iterative engineering and design tasks. However, searching for similarity in a comprehensive knowledge base is confusing and not efficient. Unless there is a possibility of prioritizing main data and features in order to customize the result of retrieval significantly.

This dissertation utilizes the Opitz coding system as a CAM classification basis and as one of the most well-known CAM classification methods in the industry today. The novelty of this work addresses an automatic and instant conversion of a STEP file into a shape signature for CAD classification. New algorithms for STEP feature extraction and Opitz feature recognition has been developed. A rule-based system has been proposed and developed for each classified feature to construct the complete code as shape signature.

The main contribution of this work is a new systematic approach for CAD quantitative similarity comparison and retrieval. A new query interface has been proposed and developed in this dissertation for searching of similar prioritized 3D parts based on their neutral file format such as STEP format. The proposed approach assigns individual priorities (0-100%) for each local geometrical feature of a CAD model.

The key advantage of using this method is that the CAD designer has access to comprehensive information on the designed part beyond geometrical information; such as manufacturability or cost. Thus, the design iterations inter and between different design disciplines are reduced. In addition, this method ascertains retrieval of local similarity as well as global similarity retrieval. With using this approach, the shape signature is extended to a product signature including integrated data.

The result of using the proposed method has been evaluated in comparison to the other recent methods. This is assessed by applying a well-known benchmark Engineering Shape Benchmark (ESB), for CAD part similarity retrieval. The proposed framework for similar-

ity recognition of CAD models has proven to be a novel method for quantitative similarity recognition for local features. It offers a new perspective on optimization of new product development by considering the upstream product development methodologies in the design procedure.

Zusammenfassung

Die Entwicklung neuer Produkte auf Basis des technischen Wissens bereits existierender Produkte verspricht eine Steigerung der Effizienz in der Produktentwicklung und ist von daher für Hersteller wie auch Entwickler gleichermaßen interessant. Die Wiederverwendung bereits existierenden Know-hows verspricht höherwertige Produkte und kürzere Entwicklungszeiten bei geringeren Kosten.
Inwiefern auf bereits vorhandenes Wissen zugegriffen werden kann, hängt davon ab, inwiefern die Ähnlichkeit zwischen Produkten erkannt wird und damit letztlich von der Klassifizierung und dem Erkennen dieser Ähnlichkeiten. In der CAD Phase gibt es bereits Möglichkeiten durch die Erkennung von Ähnlichkeiten die Entwicklung zu optimieren, indem man schon in der Entwicklungsphase Optimierungsmöglichkeiten nutzt.
In dieser Arbeit soll die Erkennung von Ähnlichkeiten in einem umfassenderen Sinn betrachtet werden und nicht nur geometrische Daten, sondern auch technische Aspekte, wie auch marktrelevante Daten berücksichtigt werden. Mit diesem Ansatz werden alle Aufgaben der Konstruktion und Entwicklung während des Lebenszyklus eines Produktes erfasst. Die Suche nach Ähnlichkeiten in einer umfassenden Wissensdatenbank ist allerdings verwirrend und nicht effizient, wenn es keine Möglichkeit gibt, bei der Suche Prioritäten zu setzen und die Suche hinsichtlich der wesentlichen Daten und Aspekte zu optimieren. Der Schwerpunkt dieser Arbeit besteht in einem neuen systematischen Ansatz auf die Wissensdatenbank in der CAD Phase zuzugreifen und der quantitativen Suche nach Ähnlichkeiten.
In dieser Arbeit wird ein neues Abfrage Interface, das die Suche nach ähnlichen 3D Komponenten, die in einem neutralen Format, etwa dem STEP Format, vorliegen, vorgestellt und entwickelt.
Der wesentliche Vorteil dieser Methode ist, dass der CAD Entwickler nicht nur Zugriff hat auf die geometrischen Daten der Komponente hat, sondern auch auf andere relevante Daten, wie etwa die Produktionskosten. Die Methode erlaubt nicht nur Abfragen bzgl. der Ähnlichkeiten ganzer Komponenten, sondern auch Abfragen bzgl. der Ähnlichkeiten von Teilkomponenten.
Bei diesem Ansatz wird die Formsignatur zu einer Produktsignatur erweitert, die auch die dazugehörigen Daten mitumfasst. In dieser Arbeit wir das Opitz Codierungs System, das heutzutage in der Industrie weit verbreitetste CAM Klassifizierungssystem, verwendet. Die CAM Klassifizierung wird hier eingesetzt zur Klassifizierung des CAD Models und dient dem Technologie Transfer. Der Beitrag dieser Arbeit besteht in der automatischen und simultanen Konvertierung der STEP Datei in eine Formsignatur für die CAD Klassifikation.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

The amount of various manufactured products has been continuously growing in the last decades. However, it is a well-known fact that only a small percentage of new products are real innovations. Based on a research made by A.D. Little, Inc. [1]: "Up to 80% of the work done in an engineering department is identical or very similar to work done previously". In another research Ullman [2] claims more than 75% of the engineering design activities contains reuse of previous design knowledge to address a new design problem. Even though most of the companies carefully maintain their product know-how, they take hardly advantage from a systematically reuse of such data. The main benefit of utilizing the existing know-how is the reduction of design time in new product development. The engineering design is a multidisciplinary task and thus an iterative process, a fact that bares major importance. This iteration refers to transdisciplinary and interdisciplinary design optimization. It is common to frequently revise an optimized design of a discipline in order to satisfy the other disciplines. These iterations lead to a time-consuming and an inefficient design process. Thus it is highly significant to have a framework which systematically provides access to know-how of the existing products in the design phase of new product development, in order to reuse the design knowledge. Furthermore, to entirely utilize the existing know-how, it is important to have only one reference for multidisciplinary data of a product. That would provide an optimized solution for the reduction of interdisciplinary design iterations. The reuse method addresses three main concepts: (1) classification, (2) similarity recognition and (3) similarity retrieval. Classification consists in assigning objects to a certain class. In the context of 3D geometry, it refers to selecting appropriate method (representation) to include all features of an object. Similarity recognition and retrieval are aimed at the recognition and retrieval of data for objects

which share specified commonality with a given query. In data classification, the main focus is to set a representation which could contain multidisciplinary data. Most of the product data are presented in a logical manner either by numbers or by alphanumerical codes. However, the existing methods for 3D shape representation are mainly graphs or statistics. Recently, there have been conducted a large number of studies in the field of 3D classification, not only regarding new product development and CAD design, but also computer vision. This is a highly interesting research field due to the significant amount of 3D data produced, uploaded and searched in the virtual worlds such as Google (Google Street View [3] and Google Earth [4]) and computer games as Kinect [5]. Eurographics conference organizes an annual contest on 3D Shape classification and retrieval to evaluate the effectiveness of proposed algorithms [6][7][8][9]. In the field of new product design, the 3D classification refers to two leading concepts of feature recognition and feature based design which have already been known to offer CAD design efficiency [10][11][12][13]. Feature recognition refers to the identification and topological grouping of entities into major functionally features such as holes. The feature based design process captures the relevant design information and stores it to be reused afterward.

1.2 Problem Statement and the Proposed Approach

The process of finding similar shapes (typically from the same object class) given a query representation is referred to 3D shape retrieval. This is applied to measure the similarity between two 2D [14]. Representation is divided into two main groups:

Global representation:

Which refers to the global information extracted from a shape; similarity is assessed by alignment of two presentations.

Local representation:

In which a shape is represented as a collection of much local information. Such a representation should be invariant to rotation, scale and pose of a shape.

Fig. 1.1: The global similarity compares the posture of the animals (cat and lion) whereas the local representation highlights the similarities of local parts such as head or tail (cat and cat)

Fig. 1.1 indicates the differences between local and global similarity. While a cat and a lion have a similar pose, they belong however to different classes. The recent research studies focus more on which features have to be considered for comparison and on how to present features than on how to perform the comparison process itself [13]. That being said, sometimes the choice of feature representation determines the method of comparison.

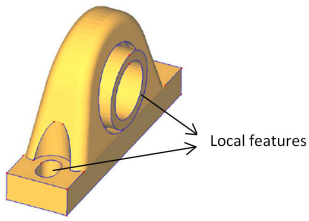


Fig. 1.2: Two local features of CAD model

Although the state-of-the-art of global shape descriptors goes back decades and feature recognition has included already some pioneering researches, there are still research gaps to be fulfilled. Firstly, the existing methods have to do mostly with the previously mentioned challenge of having a unified product representation for all product data as well as geometrical and topological data. Most of the 3D shape representations are graphical or statistical based and incapable of being extended for the other product data. In addition, the search of local similarity in graph-based methods with continuous function is not possible. Within the Product Lifecycle Management (PLM) field, there have been researches focusing on the product data, as a concept [15] or integrated in one presentation such as SysML [16]. However, such models are mainly used for the illustration of product data and unfortunately are incapable of adopting similarity retrieval algorithms and specifically quantitative similarity retrieval algorithms. Secondly, most of the existing similarity searching methods focus merely on 3D design optimization and only a few research studies are available for local similarity retrieval in the field of product development [17][18][19][20]. With the exception of Johnson and Hebert [21] work, the topic of local features, is however fairly new in the computer visual field. The new concepts of "visual words" proposed by Lavoué [22] and the extension of "bag of words" found in the research work of Tabia et al. [23] are amongst the most recent methodologies that deal with and recognize the local features. This dissertation proposes a framework for classification, similarity recognition and similarity retrieval covering a comprehensive range of product data for local and global form features. This framework encompasses not only engineering and design data but marketing data as well and classifies the geometrical data in the same data presentation model besides other product data in PLM. The proposed framework applies an upstream application of CAM features for the classification of CAD features, see Fig. 1.3.

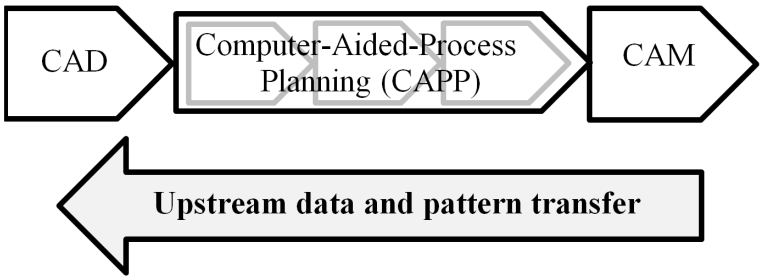


Fig. 1.3: Technology transfer in integrated design and manufacturing engineering

By upstreaming this information and applying it systematically in the detailed design phase, the number of design iterations reduces dramatically. Based on Chakrabarti [24] there are three models of engineering design synthesis: (1) function-based, (2) grammar-based and (3) analogy-based. The analogy-based design is further divided into three categories: (1) analogy-based design, (2) biological inspired design and (3) case-based reasoning design. The proposed methodology in this dissertation addresses the case-based reasoning design. I have chosen the case-based reasoning because it contains multidisciplinary data, thus being an proper option to classify multidisciplinary PLM data. Such data refers to the various constraints such as geometrical, logical and dimensional, having the potential to extend to more constraints.

This dissertation, presents a novel framework to resolve these challenges, including:

- An automatic feature representation of CAD models
- A variable similarity retrieval system for local and global features

With the exception of these two main solutions, some other methods that are required for the generation of a shape representation or shape index, are proposed. These methods are:

- A pose standardization and an automatic rule based system for STEP feature recognition
- A graph-based algorithm for Opitz feature extraction
- A quantitative (local or global) similarity retrieval system
- A text-based Opitz feature recognition and code generation
- A visual representation (CAD model) as well as the text-based description of the similarity retrieval.

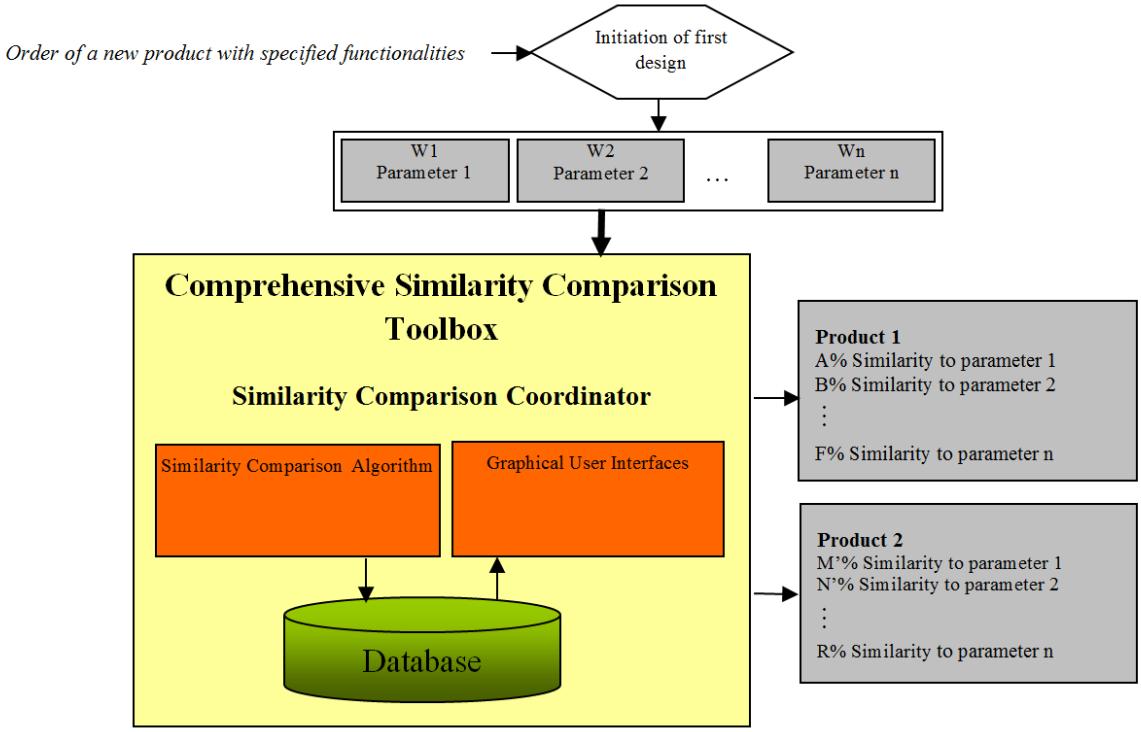


Fig. 1.4: Perspective of the desired solution

The proposed methodology has been benchmarked and compared with the other most recent methods, in order to validate its efficiency and feasibility. This was achieved by applying a well-known benchmark Engineering Shape Benchmark (ESB), for CAD part similarity retrieval.

1.3 Research Objectives

This dissertation presents a systematic approach for developing a variable similarity retrieval methodology for desired CAD comparison and retrieval. The proposed approach assigns individual priorities (0-100%) for each local geometrical feature of a CAD model, see Fig. 1.4. This is reached by an automatic and instant conversion of a STEP file into a shape signature for the classification of geometrical information. The method proposed in this research is expected to enhance PLM as follows:

- Cost reduction in PLM by realization of a CAM classification method to be used as a CAD classification method. The aim of this method is the similarity retrieval in the design phase of product development.
- Cost reduction in PLM and specifically in CAD by reducing design iteration as well as engineering tasks. This is done by providing access to comprehensive informa-

tion (geometrical and non-geometrical) of previously designed similar parts in the first stage of design. By applying this approach, the decision making and expertise interchange in the design phase are optimized.

1.4 Organization of Dissertation

This dissertation comprises seven chapters. Chapter two presents existing similarity retrieval methods and their application. These methods include the feature recognition of 3D CAD models, shape representation and distance function. In addition, an introduction to and the review of group technology approaches as well as methods of automatic feature recognition are explained and discussed. The proposed methodology in this dissertation is introduced and explained in detail, starting with chapter three. This chapter explains the architectural design of the proposed approach, the feature recognition, the feature extraction as well as the rules applied. Chapter four describes the proposed similarity retrieval and the proposed quantitative similarity retrieval method. Chapter five introduces the details of the prototype developed and implemented in this research for automatic feature recognition. Chapter six presents the results of applying the proposed approach using Engineering Shape Benchmark (ESB) and evaluates every module of the proposed approach. In addition, it discusses how the proposed approach improves PLM. Chapter seven concludes this work and presents ideas for the future work.

Chapter 2

Review on Similarity Recognition of CAD Models

Comparison of the geometrical 3D CAD models is an intricate investigation. Most of the methods apply an injective (one-to-one) function or principle to transform a 3D shape to a computable model, a so called shape signature. A shape signature is the final result of a feature representation method. A shape signature is defined as the encapsulated data of a solid model, including all of the major geometrical and topological features of an object. It is supposed to have an understandable and computable format for automatic computation and comparison by computer, such as an image, graph, vector or an ordered string of numbers or alphabet. Thus, the outcome of a similarity assessment between the two 3D shapes would result in a comparison of their shape signatures. Hence, a similarity assessment between two 3D shapes includes two main steps: First, computing a shape signature and second, comparing the shape signatures by a suitable distance function. In respect to classification of the methods, Cardone et al. 2003 [25], Gao et al. 2006 [26], Liverani and Ceruti 2010 [27] use almost the same classification for shape signature methods while the classification of Iyer et al. 2005 [28] is slightly different. Both categorization models are basically the same; a minor difference refers to including the multi resolutional method into the first categorization model. In addition, the feature-based category has been properly divided into global and manufacturing features in the first categorization.

2.1 CAD Feature Recognition Methods

The word “feature” is an ambiguous and dependent denotation which mainly alters its meaning based on the application. This might be the reason why most of the discussed approaches have considered different features according to their method of feature extraction and their intentions. Consequently, the concept of feature recognition relies on the specific engineering principle. In general, feature recognition/extraction refers to finding the most compact and informative set of features to be applied in data storage or data processing. Since the influential work of Kyprianou 1980 [29], feature recognition has been the subject of much research. Nonetheless, feature in the concept of CAD essentially refers to two domains, design feature and manufacturing feature. Design features defined by CAD systems are low level entities such as vertices, edges, faces, etc. However, features which are applied to give information about a part's manufacturing are high level entities like holes, slots, pockets, etc. Additionally, manufacturing features usually include process planning information such as a selection of manufacturing processes, a work piece size and auxiliary tools, set-up planning, selection, determination and grouping of elementary machining operations, selection of machining systems, operation sequencing and optimization of operation sequences, selection of cutting tools, determination of cutting parameters and conditions, tool path determination, selection of quality inspection methods, cost analysis, optimization of process plan elements and Computer Numerical Control (CNC) code generation and verification [30].

In the context of CAD system, design features are referred to in three types of applications. Design features are based on form feature concept but differ in methods of form feature determination. The mentioned applications include:

- Design-by-feature
- Automated Feature Recognition (AFR)
- Interactive form feature definition

In design-by-features, there is a required form feature library categorized on part manufacturing needs. AFR focuses on finding an algorithm to classify form features without any interference of the manufacturing engineer, while applying part representation methods. In an interactive form feature definition, a user selects a form feature set as well as recognition parameters and the system automatically recognizes the features.
In the following sections a comprehensive combination of both major categorizations is concisely discussed [31][32].

2.1.1 Invariant-Based Methods

These approaches use invariants or descriptors of the 3D shape such as volume, surface area, aspect ratio, higher order moments or moment invariants as signatures [28]. In the following, four methods which belong to the category of invariant-based will be briefly discussed. These methods include: RTS-invariants, moments and relevance feedback, non-dimensional and scale-independent features, elementary-shape-based features and active learning.

2.1.1.1 RTS-Invariants

In this method, solid objects given in a standard digital representation like the IGES file format are converted into a surface triangular mesh representation [33]. Afterward, the triangular mesh representation is converted into a voxel model representation using a flood filling method. For the shapes represented by voxel model, geometrical moments are calculated and used to normalize the object into a canonical form. Shape features are computed by calculating variant volumetric invariants. They are called RTS-invariants because these features are invariant against rotation, translation and scaling. The following RTS-invariants can be calculated: second-order 3D moments invariants, spherical-kernel moments invariants, axis aligned bounding box and centroid of the object, and the surface area of the objects. In the first step of the similarity measurement, feature vectors are used to compute a set of best candidate objects. On this set a voxel-by-voxel comparison is performed as the second step of the similarity measurement. This step allows a detailed comparison between voxel model representations of objects based on the template matching.

2.1.1.2 Moments and Relevance Feedback

Moments are used as shape features of 3D models and relevance feedback as an iterative and interactive method to improve the performance retrieval. For the models given in Virtual Reality Modeling Language (VRML) file format, the geometrical moments are calculated and approximated up to the third order and used to normalize the object in a canonical form. For the normalized objects, moments are approximated again (up to forth-to-seventh-order is sufficient) and used as a feature vector of objects [34].

In the first step of the similarity measurement, a set of the best candidates is presented to the user by computing the Euclidean distance between feature vectors of the primary object and objects from the database. After that, the user has the possibility

to influence the future search results by applying the method of relevance feedback. The user can mark a subset of presented results as relevant or as irrelevant. Based on these markings, which capture the user-perceived similarity between objects, the distance measure can be adapted and a new search result is calculated. The adaption of the distance measure is based on Support Vector Machine (SVM) learning algorithms and can be repeated until the user is satisfied with the search results.

2.1.1.3 Non-Dimensional and Scale-Independent Features

Corney et al. [35] used various non-dimensional and scale-independent features as signature for 3D CAD models in an internet search engine. Most of these features are computed using object characteristics such as volume, surface area and convex hull of objects. For example, the features such as crinkliness, compactness, and hull crumbliness are calculated as follows. Crinkliness is defined as the surface area of the model divided by the surface area of the sphere having the same volume as the model. Compactness is defined as a ratio of the volume squared over the cube of the surface area and used as a non-dimensional feature. Hull crumbliness is defined as a ratio of an object's surface area to the surface area of its convex hull. Hull packaging is defined as the percent of the convex hull volume not occupied by the original object. Hull compactness is defined as ratio of the convex hull's surface area cubed over the volume of the convex hull squared. Further features being used are: the ratio of the longest edge to the shortest edge of the bounding box, the number of the holes of the object and the number of the facets of the object. The user can specify combination of features which are used in the similarity search and the tolerance values of these features [36][37].

2.1.1.4 Elementary-Shape-Based Features and Active Learning

The method from Zhang and Chen [38] describes that features such as volume, surface area, moments and Fourier transform coefficients can be well extracted from a mesh representation and be considered as the signature of an object. The inspiration of this method is to compute features for elementary shapes such as triangles and tetrahedrons in advance and sum up the feature values of the elementary shapes in order to get the feature value of the whole object. Annotation of the object was used as a method to improve the performance retrieval. The hidden annotation has to be performed as a learning stage before a database can be used for the similarity search. By using an active learning method, the system determines the sample objects to the annotator. The sample objects are selected so that annotation of the object can provide the maximum information or knowledge gain to the system. Using this method reduces the number of training samples

by selecting the most informative ones to the annotator. Settles [39] has a comprehensive review on active learning concept and methods.

2.1.1.5 Evaluation of Invariant-Based Methods

All invariant-based methods have the advantage of being robust to the small changes in shape. The disadvantage of these methods refers to the improbable partial matching. RTS-invariants method has a disadvantage because it requires a huge amount of storage for every object and its different models. In addition, the voxelization of models is a time and memory consuming process [40][33]. In the method using the relevance feedback, not only is the geometrical similarity being computed between the objects, but also the user-perceived similarity can be incorporated in the similarity search process. Hence, the retrieval performance is being improved by retrieving more objects than the user has in mind. The approach of Corney et al. [35] can be useful as a coarse filter in huge databases. However, to perform a finer comparison between objects when the sets of retrieved objects are large, combination of this method with further methods might be necessary. Zhang and Chen [38] used the method of hidden annotation and active learning to improve the retrieval performance of the system. In practice, an annotation of large databases can hardly be performed because of the manual effort. Besides, with applying partial annotation, it is difficult to decide how much annotation is sufficient for a specific database [39]. The disadvantage of the method presented by Zhang and Chen [38] lies on the requirement of an explicit routine to compute a feature value for elementary shape. Nevertheless, it is difficult to develop explicit routines to compute the high order moments for triangles and tetrahedrons [31].

2.1.2 Harmonic-Based Methods

These methods use a set of harmonic functions of a shape as the signature. Spherical or Fourier functions are usually used to decompose a discrete 3D model into an approximate sum of its (first n) harmonies components [28]. The four methods of this category will be discussed in the following sections: Ray-based SH-descriptor, Rotation-invariant SH-descriptor, Layered depth sphere-based SH-descriptor; and Concrete radialized spherical projection descriptor.

2.1.2.1 Ray-Based SH-Descriptor (Spherical Harmonics)

In the method from Vranić [41][42] and Ruggeri et al. [43], 3D models represented by polygon meshes are normalized to achieve invariance against rotation, translation and scaling. For that purpose a Principal Component Analysis (PCA) is applied to a discrete set of points, as well as the union of all polygons of the mesh with infinitely number of points. After normalization, the 3D models are characterized by defining a function on a sphere which measures the extension of an object in different directions.

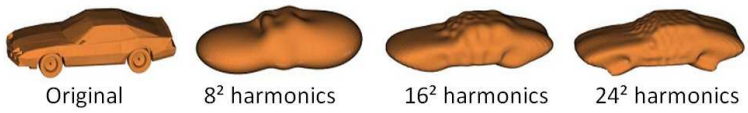


Fig. 2.1: Multi-resolution representation used to derive feature vectors from Fourier coefficients for spherical harmonics [42]

For each direction a ray is cast from the center of mass in order to compute the last point of the intersection with the polygonal mesh which is used as a sample of the function. After sampling, the function Fast Fourier Transformation (FFT) is performed to obtain the Fourier coefficients to be applied as the feature vector. Fig. 2.1 represents reconstruction of the different levels of a primary object when using three different spherical harmonic coefficients.

2.1.2.2 Rotation Invariant SH-Descriptor

Kazhdan and Funkhouser [44] in description of the Princeton shape benchmark claim that the methods using PCA are unstable referring to the multiplicity of Eigenvalues and its sensitivity to outliers.

As a solution, a new method [45][46] to compute the harmonic shape representation was proposed. In this method, the model polygon is rasterized into a $64 \times 64 \times 64$ voxel grid. The voxel grid is decomposed into 32 functions on concentric spheres by restricting the voxel grid to spheres with radii 1 to 32. By decomposing each of these functions as a sum of its first 16 harmonic components, analogous to a Fourier decomposition into different frequencies and defining the signature of each spherical function as a list of these 16 norms and combining the different signatures, a 32×16 signature for 3D model is obtained. In order to compare two harmonic presentations, the Euclidean distance between them should be computed. An example of the explained method is shown in Fig. 2.2.

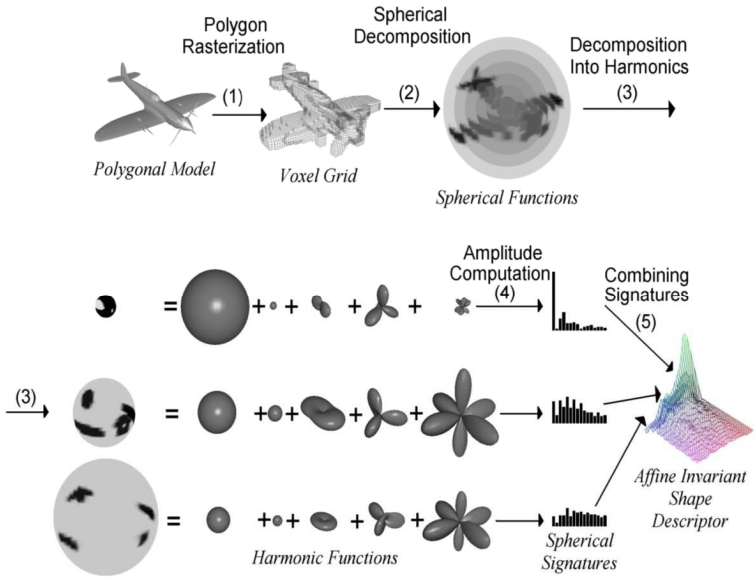


Fig. 2.2: Computing the Harmonic Shape Representation [44]

2.1.2.3 Layered Depth Spheres (LDS)-Based SH-Descriptor

Vranić [41], described a further harmonics-based method which captures information about the internal structure of objects. The shape descriptor is extracted from a triangle mesh representation of the objects. Invariance against translation and scaling is achieved using Continuous PCA (CPCA). 3D model is decomposed into a family of functions on the sphere restricting function values by lower and upper bounds which describes a bounded area of the model. Using the ray cast method for rays emanating from the origin in many directions, all points of intersection with the polygonal mesh are computed, see Fig. 2.3.

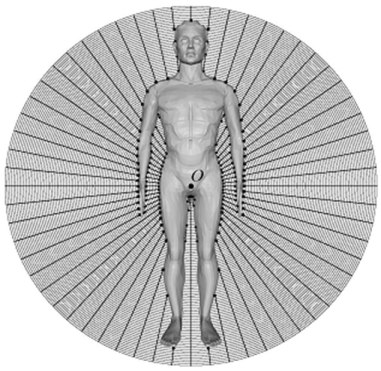


Fig. 2.3: Concept of Layered depth Sphere with an example in 2D [41]

For intersection points the closest sphere and a set of corresponding value of the function on that sphere is determined. If two intersection points of the same ray belong to the same sphere then the larger distance determines the function value. On each sampled function on the sphere FFT is performed to obtain a set of coefficients. The PCA method can be

performed during the normalization step or the properties of spherical harmonics can be used to achieve rotation invariance.

2.1.2.4 Concrete Radialized Spherical Projection Descriptor (CRSP)

In the method from Papadakis et al. [47], a shape descriptor is extracted from a triangle mesh representation of 3D models. In this method, scaling and axial flipping invariance is achieved referring the properties of spherical harmonics. Rotation and translation invariance is achieved by applying CPCA and PCA on the model's Normal (NPCA). This algorithm, results in two versions of an object and, therefore, two descriptors for an object. For each version of the object, a set of functions on spheres is defined sampled by casting rays from the origin of the object.

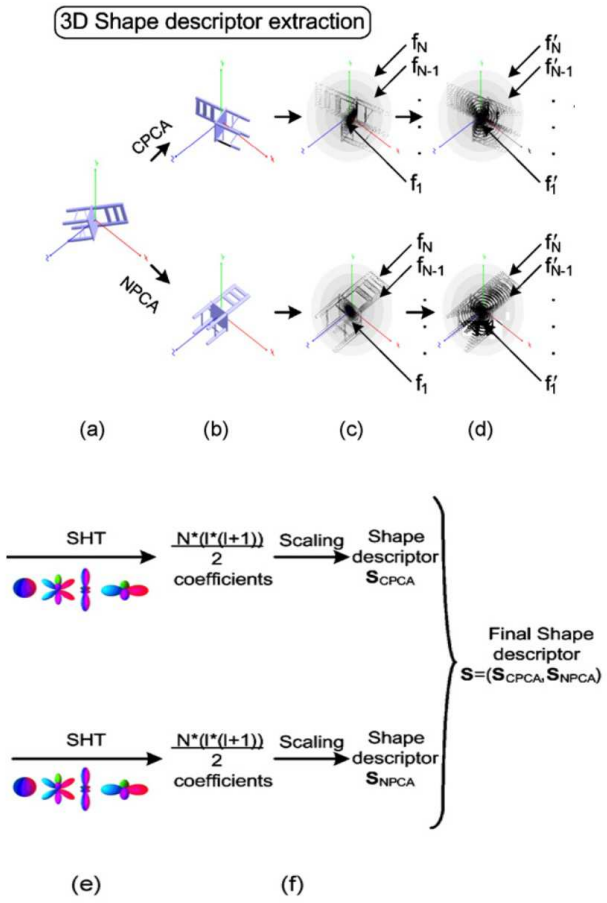


Fig. 2.4: The stage of the shape matching using CPCA and NPCA [47]

A function on a sphere represents the intersectional points of the models surface with rays and also all points in the direction of each ray that are closer to the origin than the furthest intersection point. For every function, Short-time Fourier Transform (SFT) is performed to obtain the Fourier coefficients. Scaling invariance of the descriptor is

achieved using the properties of the spherical harmonics. Fig. 2.4 illustrates the stages of the shape decomposition and matching as well as obtaining the shape descriptor/signature respectively.

2.1.2.5 Evaluation of Harmonics-Based Methods

All harmonics-based methods have advantages which feature extraction and similarity measurements are efficiently performed. Drawbacks of these methods are as follows. First, specific details of shape cannot be captured, and second; partial matching is not possible in these methods [28]. Kazhdan and Funkhouser used a coarse voxel grid to achieve robustness against small changes of shape. However, coarse voxel grid causes the loss of many details [44]. Voxelization also affects the efficiency of feature extraction. The ray-based method allows an embedded multi resolution representation of the descriptor. This means that a descriptor contains all descriptors which have lower dimension [45]. Unlike the ray-based method, LDS-based method captures information about the internal structure of objects by defining several functions on spheres instead of only one. The CRPS method improves the invariance properties of the descriptor by applying two normalization methods, CPCA and NPCA. Thus, the retrieval performance of the descriptor is improved. Although this process increases the complexity of the descriptor, since for each object two descriptors are extracted [47][48].

2.1.3 Graph-Based Methods

A graph-based method develops a graph, based on the encoded shape, geometry or feature of a 3D model. In some methods, the subgraph isomorphism is used in order to match B-Rep graphs, or to match eigenvalues of a model signature graph which is constructed from the B-Rep graph [49], see Fig. 2.5.

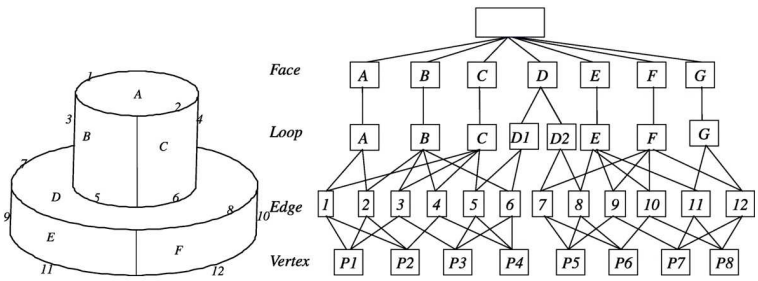


Fig. 2.5: A part and its B-Rep data structure [49]

In graph-based approaches subgraph isomorphism is used in order to match B-Rep graphs, or to match eigenvalues of a model signature graph which is constructed from the B-Rep

graph. Four different methods belonging to the graph category are briefly explained in the following sections. These four methods include: Model signature graphs, Attributed graphs, Reeb graphs and Skeletal graphs with parameter-controlled thinning [28].

McWherter et al. [50][51] and few other researcher [52][53], developed Model Signature Graph (MSG) for similarity measurement between 3D CAD models. MSGs are labeled, undirected graphs, which are generated using the Boundary Representation (B-Rep) on CAD models. The boundary representation consists of a set of edges and a set of faces. For the definition of MSGs every face of the model is represented as a graph vertex and every edge in the B-Rep is represented as a graph edge. Labels of edges and vertices contain attributes of faces and edges in B-Rep, such as topological identifier, underlying geometrical representation, etc. For every MSG the eigenvalue spectrum and Invariant Topology Vector (ITV) are extracted and used to perform similarity comparison. ITV contains graph invariants, such as vertex and edge counts, maximum, minimum, mode degrees graph diameter, etc., see Fig. 2.6.

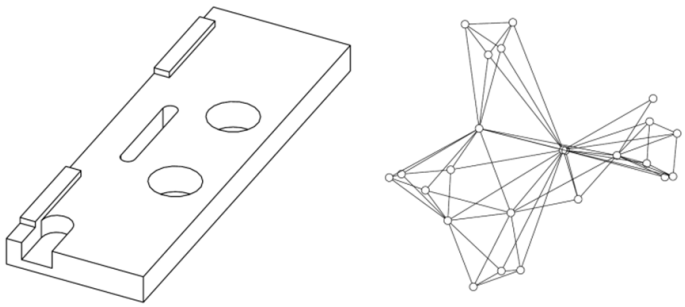


Fig. 2.6: A model and its transformation into a Model Signature Graph [51]

The spectrum of a graph is sorted as the eigenvalues of its adjacency matrix, which holds information related to the graph structure. In addition, the eigenvalues of the graphs can be used to partition the graph into two or more subgraphs in order to compare substructure of the graphs.

2.1.3.1 Attributed Graphs

In the approach of El-Mehalawi and Miller [49][54], attributed graphs are used as the signature of 3D CAD models, which are extracted from STEP files of these models. Attributed graphs are quite similar to MSGs, hence, graph nodes describe the faces of CAD components and graph edges describe the edges of CAD components. In addition, the node attribute corresponds to the surface attribute, such as type of surface and direction of the normal. Edge attributes correspond to the edge attributes in B-Rep, such as type

of edge, the normal direction and length of the edge. For the retrieval process, abstract information is extracted from attributed graphs and used in the first step of the retrieval process. The abstract data is the total number of nodes, number of nodes representing plan, cylindrical and conical surfaces. These data are used as an index, where the set of graphs candidate similar to the query graph can be calculated very quickly. In addition, a more accurate comparison is applied for the set of candidate graphs. To finalize the method and to complete the retrieval procedure, inexact graph matching algorithm based on an integer programming model is applied. This algorithm has a polynomial computational complexity. You and Tsai [55] used an attributed graph for a B-Rep to preserve the geometrical and topological data from STEP. Local feature correspondence is evaluated by identifying the size of the common subgraph from the graph descriptor. An innovate structure called Independent Maximal Clique (IMC) detection as well as simulated annealing algorithm promise to solve the problem of graph-matching.

2.1.3.2 Skeletal Graphs with Thinning

Sundar et al. [56] used skeletal graphs as signature of 3D models for similarity measurement as Fig. 2.7 presents.

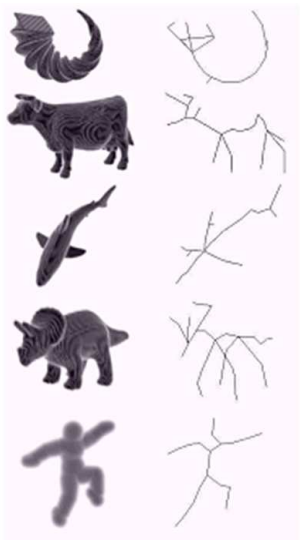


Fig. 2.7: Skeletal graphs based on the thinning algorithm [56]

To extract the skeletal graph of an object, the respective 3D model ought to be converted into a voxel model. In the next step, for the skeletonization process, a parameter-controlled thinning algorithm is used to calculate a subset of voxels. In this thinning method, the thickness of the skeleton is determined by the parameter given the user. Hence, a family of the different voxel sets can be calculated, each one is thinner than its parent. The thinness

parameter classifies the importance of the voxels for the boundary coverage by comparing the distance transform of the voxel with its 26 neighbors. After skeletonization the Minimum Spanning Tree (MST) algorithm is applied in order to generate an undirected acyclic graph out of unconnected skeletal points. For every node in the graph Topological Signature Vector (TSV) is defined which holds information related to the node underlying subgraph's structure. TSV contains the eigenvalues of the subgraph's adjacency matrix and is used as an index to fast determination of a set of best candidate graphs. On the set of candidate graphs a graph matching algorithm is performed by reformulating the problem of largest isomorphic subgraph as the problem of finding the "maximum cardinality, minimum weight matching" in a bipartite graph. To preserve the hierarchical structure of the graph a greedy form of the above bipartite formulation is combined with a recursive depth search.

2.1.3.3 Skeletal Graphs with Parameter-Based Thinning

In the method from Iyer et al. [57] and explained or continued in [58][59], skeletal graphs and feature vectors jointly present the signature of 3D models. 3D models are normalized into a canonical form and converted into voxel model. In the skeletonization process, iterative thinning algorithm is applied by deleting border points satisfying conditions of topology preservation. On the generated skeleton, the skeleton-marching algorithm is performed to identify the basic entities and construct the skeletal graph. Basic skeletal entities are vertex, edge and loop. For the definition of feature vectors the following shape descriptors are extracted from the voxel model: moments, geometry parameters such as volume and surface area, voxelization parameters such as voxel size, and graph parameters such as number of loops edges and nodes. For the similarity measurements the Euclidean distance of the feature vectors, as well as the distance between skeletal graphs are calculated. For the graph matching, a decision-tree based algorithm developed by Messmer and Bunke [60], is applied. In another research, a statistical-based method to generate shape signatures from the properties of the skeleton oposed to graph-based has been employed. In this algorithm all graphs in a database are indexed in the form of a decision tree using the different permutations of the adjacency matrix as described in Fig. 2.8. The space requirements are exponential, but the search requirement is sub-polynomial in the number of query graph nodes. The advantage of this method refers to a faster construction of the shape signatures and thereby the increased speed of matching [61][62].

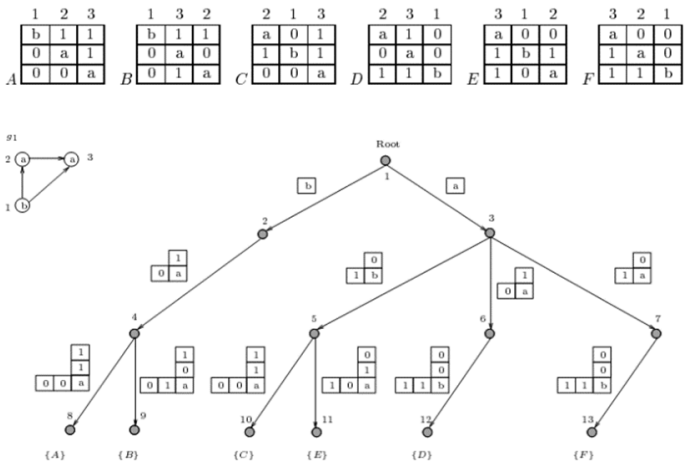


Fig. 2.8: Decision tree for the related (above) adjacency matrix [60]

2.1.3.4 Evaluation of Graph-Based Methods

The advantage of all graph-based methods is the description of the topology of 3D models which is an important shape feature. In addition, representation of 3D models as topological graphs which facilitate the abstraction of these models at different levels of detail and description of local geometry at each node [28]. Other advantage of the graph-based methods is the possibility of partial matching between 3D models (except for MSG method). MSGs are efficient despite the large and complex graphs in the database. This method is considered insufficient for fine discrimination between 3D models, referring to the disability of capturing all properties of the adjacency matrix by the eigenvalues [50][53][54]. Although skeletal graphs with using simplification of 3D, are stable to small changes in shape, but the simplification causes a loss of information affecting the discrimination power of the method [57]. The advantage of multi resolutional reeb graphs refers to the fact that geodesic distance as a continuous function is invariant against rotation and translation. The exponential computational complexity of this method has been avoided by applying the coarse-to-fine strategy in the retrieval process [52][63].

2.1.4 Object Recognition-Based Methods

Ruiz-Correa et al. [64] introduced the spin images as a signature. Spherical Spin Images (SSI) are signatures associated with the vertices of a polygonal mesh of a given resolution that approximates the surface of an object. SSI are points of the unit sphere in which corresponding points are found by computing the angle between two SSIs, see Fig. 2.9.

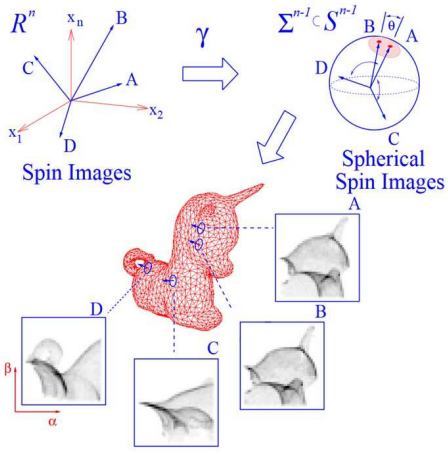


Fig. 2.9: Spherical spin images for four oriented points (3D points plus normal vectors) A, B, C and D on the polygonal mesh of a model [64]

2.1.5 Histogram-Based Methods

Using shape functions to create a shape distribution of random sampling of points [65][66], a well-known method of this category is Shape Distribution Graph (SDG). To obtain SDG, fixed numbers of points are considered on a part surface and the Euclidean distance for each pair of points is calculated. Then a graph is drawn based on distances and number of pair points as Fig. 2.10 presents. The graph demonstrates typical trends according to the overall shape of the part. Thus, similarity of two graphs reveals similarity of two 3D models respectively.

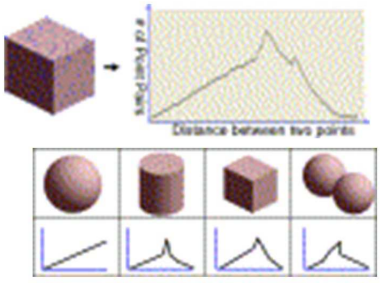


Fig. 2.10: Shape Distribution Graph [65]

The main challenge of such graphs is the fact that the SDG graph for complicated parts have the tendency to look like a normal graph and does not highlight the individual feature of a part. Generally, this method is a proper choice when the selected 3D models are primitive shapes or non-complex shapes which have an identical SDG. The advantage of this method is its comparatively simple calculation. Another method measures geometric properties of a 3D model and produces a shape signature for a 3D model by means of

a probability distribution sampled from a shape function. Similar to the motivation for the other methods for producing shape signature, this method attempts to transform an arbitrary 3D model into a parametrized function that can be easily compared with others [66].

2.1.6 Manufacturing Feature-Based Methods

In such methods, the manufacturing features are recognized from a CAD model. It is observed that mostly the similarity of shape may also be reflected in the similarity of the manufacturing process. However, this is not a general rule. In this regard, Group Technology (GT) conventionally has been applied for categorization of parts which have similar manufacturing and/or manufacturing processes. GT introduces the classification of similar parts into groups in order to reduce time in production. For the classification, one should have a detailed coding scheme for describing manufacturing features. Opitz coding system, DCLASS, MICLASS and MM3 are among the most well-known methods of GT schemes. Each of these methods offers various rules in order to capture manufacturing features of a part. The result of feature recognition is embedded in an alphanumerical string. Started in 18th century, many manufacturing companies either used the common GT codes or developed their own code such as MM3 for Toyota. One of the basic ideas for GT was introduced by Gombinski 1963, applied for Britsch & Co. company, one of the earliest companies which applied GT. Hendeson et al. 1988 used GT for similarity detection in concept of Agile Manufacturing. Belonging to this category, Model Dependency Graph (MDG) approach for a 3D CAD model is used to compare machining parts [25]. Since this research applies a manufacturing feature-based method, an extensive literature review on GT is given in section 2.1.5.

2.1.7 Automatic Feature Recognition Methods

Automated Feature Recognition (AFR) encompasses searching through some part representation aiming to find information which defines a form feature type. All methods of this approach have a unique aim to develop an algorithm capable of identifying features, without interfering with the design or manufacturing engineer. Interactive form feature is a system in which designer picks a set of features and defines recognition factors for those features. This system applies those commands to perform recognition either in a CAD system or in a compatible system. As the focus of this research is automatic feature recognition, the next section discusses the methods of AFR in details. The field of automatic feature recognition has been extensively investigated by researchers. The most

well-known methods in this domain for extracting geometrical features are classified in the following groups [30]:

1. Syntactic pattern recognition methods
2. Rule-based methods
3. Graph-based methods
4. Logic rules and expert system
5. Convex-hull volumetric decomposition (volume decomposition)
6. Cell-based volumetric decomposition
7. Hint-based methods
8. Hybrid methods

In the following sections a detailed overview of the above-mentioned methods are presented.

2.1.7.1 Syntactic Pattern Recognition

In this group of methods, features are presented as strings of geometric primitives. Some rules to define a particular pattern are applied as a set of grammar. A parser uses a grammar to analyze the input sentences. If the syntax analysis (string) is approved, then the description is classified in a corresponding class of forms (patterns). In the other words, string matching is performed for distinguishing predefined features. This method was introduced by Kyprianou [29] and is the earliest method of AFR. This method has been commonly applied for 2D feature recognition. When using it for 3D, it has to be translated to 2D in advance. Furthermore, the method shows good results for rotational features such as holes and curved components, see Fig. 2.11.

Another approach for syntactic pattern recognition was presented by Jain and Kumar [67]. This approach accepts wire frame part representation imported from AutoCAD **.dxf* file. A 3D model in wire frame representation is deciphered into a 2D vertices-edges graph. This approach has been considered for prismatic parts. Besides, it is able to recognize several form features such as a hole, step and slot. In this content, a hole is considered as a basic feature, consequently all other features are derived from it; i.e. steps are holes without two faces and slots are holes without one face. Fig. 2.12 illustrates the methodology with strings of primitives which are matched later with the patterns in a knowledge base.

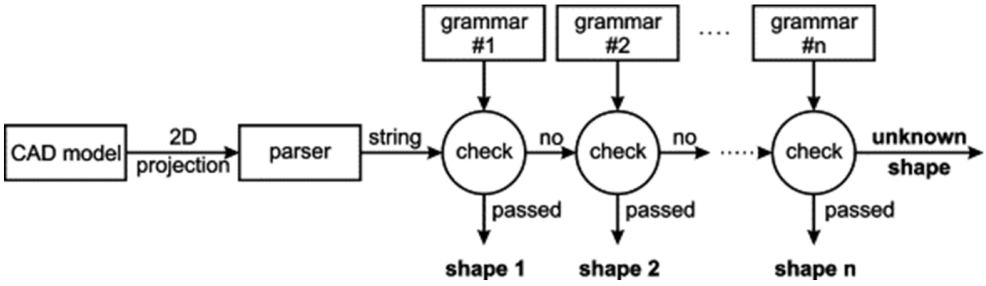


Fig. 2.11: Syntactic pattern recognition convey [30]

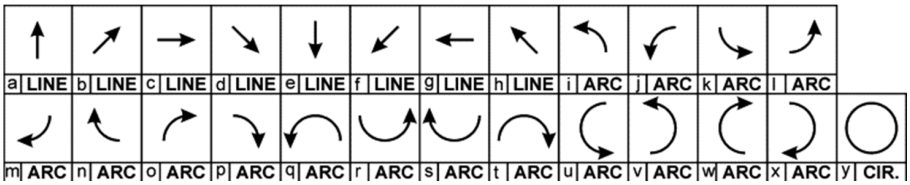


Fig. 2.12: Shape primitives of PRIZKAPP approach [67]

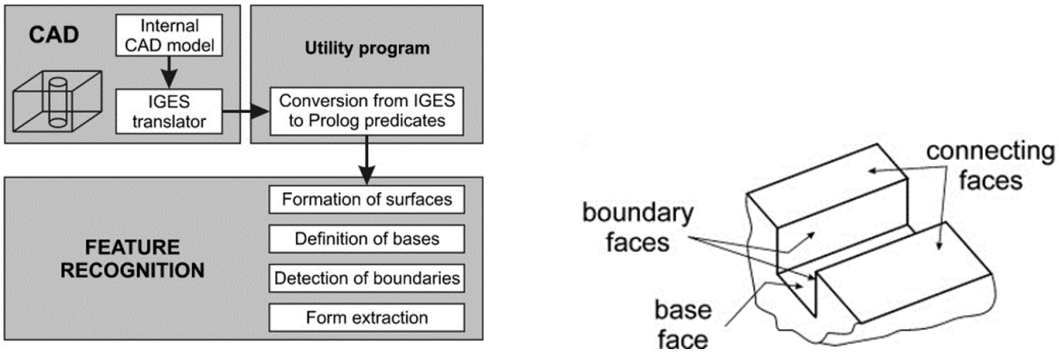
2.1.7.2 Rule-Based Methods

This approach was firstly introduced by Henderson and Anderson [68]. A pattern for AFR is provided by a set of production rules;

- IF C1, C2, C3... Cn THEN A

Each Cx was defined from features and provide a pattern. Only if the first conditions C1, C2, C3 which are provided by a pattern are satisfied, then the corresponding part representation is recognized as a form feature A. The approaches in this group are commonly based on the following stages. A 3D solid model of a part is converted into IGES format and then into Prolog facts, see Fig. 2.13(a,b). Face extraction and base faces are recognized and determined as well as boundary faces. A base face is a feature face which has a concave adjacency with at least one feature face. In these approaches, the number and the type of boundary faces for a form feature matching (except for holes) are the main principles. The features that system may recognize are pocket, slot, blind slot, step, corner step, hole, blind hole, countersink hole. In order to recognize faces, additional conditions are defined.

Sadaiah et al. [69] has applied logic rules for AFR as well. They apply Constructive Solid Geometry (CSG) graph representation for geometric feature extraction. First, the *.txt file in SolidWorks application protocol interface is browsed with an additional program. Then the extracted form is matched with patterns in an Oracle database. The focus of their research was to recognize various prismatic form parts however their approach did not fully succeed in recognition of other forms such as intersecting forms. The rule-based



(a) CAD/CAPP interface model (b) Definition of the faces

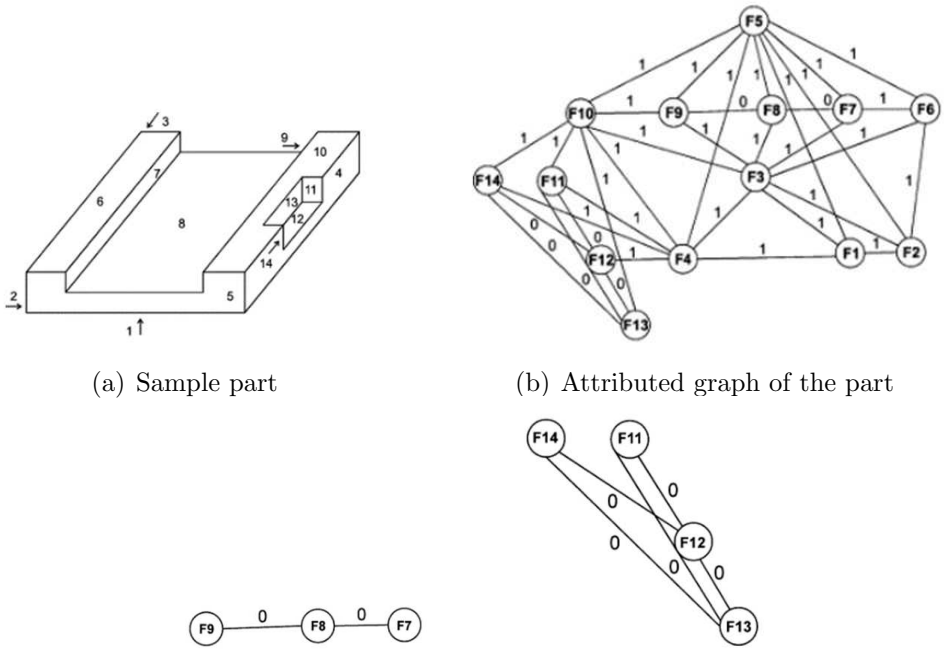
Fig. 2.13: (a,b) A CAD/CAPP interface model and Definition of the faces of a form consist of [30]

method has proven to be robust and more capable of recognizing various forms than the syntactic method. However, unclear predefined representation and rules for every conceivable feature may cause the system to be inflexible or encumbered.

2.1.7.3 Graph-Based Approach

The graph-based approaches mainly utilize the B-Rep representation of a part as a face edge or vertex edge graph for extracting features. Such a graph inherits the topological and geometrical information of the part. Graph isomorphism or graph partitioning is applied afterward to obtain features. One of the first approaches developed by Joshi and Chang in 1988 [70] applied an Attributed Adjacency Graph (AAG). The proposed system applies B-Rep model of a part designed in a solid modeler is employed as an input of the system. If a node has a concave adjacency relation every arc takes attribute '0' and '1' if they have convex adjacency relation in AAG, depicted in Fig. 2.14a and Fig. 2.14b. Form features represent subgraphs of part AAG and form feature recognition becomes a process of finding such subgraphs that can be matched with the patterns from the database. Logic rules analyze the subgraphs. This procedure which is called subgraph isomorphism is a time consuming and computationally demanding method. Another approach in this regard is graph partitioning [71]. In this method the AAG nodes with all adjacent faces convex (attribute '1'), are parsed. Fig. 2.14. AAG has two major weaknesses. The first is the restriction of applying the method only for negative, polyhedral objects (polyhedral shaped intrusions, without curved faces). The second weakness is the infeasibility of extraction of boundary faces, because only basic faces can be extracted.

The shortcomings of the AAG may be considerably lowered by using the concept of Multi-Attributed Adjacency Graph (MAAG) which allocates the attributes with more precisely



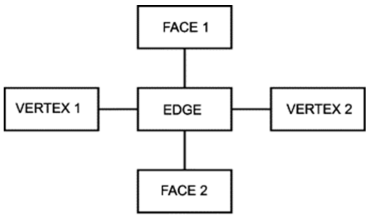
(a) Sample part (b) Attributed graph of the part

(c) Subgraph (d) Subgraph

Fig. 2.14: Attributed Adjacency Graph forming [30]

descriptive adjacency relations. For example, if the plane and curved face make a convex angle of 270, then the assigned attribute is '2'. When a matrix is applied to present MAAG, the new system is called Multi-Attributed Adjacency Matrix (MAAM). The feature recognition is achieved by adjacency matrix schemes defined in advance for every basic form. There are many systems which applied MAAG. Between them an interesting system introduced by Venuvinod and Wong [72] in which the extracted elements form CAD model using a so-called EWEDS. Enhanced Winged-Edge Data Structure (EWEDS) is an advanced model of Winged-Edge Data Structure (WEDS) [30] developed by Baumgart (1974), see Fig. 2.15. In WEDS information of an object's faces, edges and vertices are presented in an unambiguous way.

Most of the graph-based AFR systems suffer from interacting features problem. Marefat and Kashyap proposed to solve this problem by restoring the missing arcs between the nodes in part's graph representation [30][73]. This is suggested for the edges between the faces which disappear with the interaction of two or more features. Nevertheless, this approach often led to a wrong set of missing arcs, due to the uncertainty reason [74]. This reason refers to the increase of uncertainty and non-uniqueness of the patterns associated with the topology and geometry of features when features interact. Two universal techniques for handling uncertainties and finding the exact set of missing arcs have often been applied in graph-based AFR systems [75], the Dempster-Shafer theory [74] and Bayesian probabilistic rules [76][30].

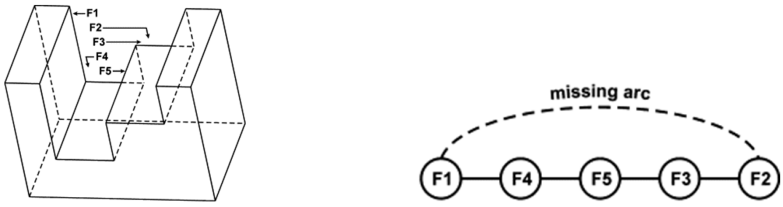


(a) WEDS

1. edge (edge number, vstart, vend, fcw, fccw, ANGLE ncw, pcw, nccw, pccw)
2. vertex (vertex number, FIRST EDGE LIST, coordinates)
3. face (face number, first ecige, TYPE OF FACE, PARAMETRIC DATA LIST LIST)

(b) EWEDS

Fig. 2.15: (a) WEDS; (b) EWEDS (Prolog facts) with additional information (upper case letters) [30]

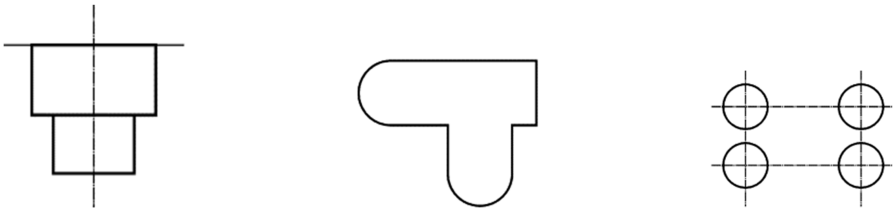


(a) Sample part (b) Part's graph representation

Fig. 2.16: Missing arc in intersecting features [30]

Apart from face-edge graph methods, there are other approaches for the graph-based structure. For example, a feature extraction technique based on loop-adjacency hypergraph. This approach proposes to acquire generalized properties feature with planar faces only. However, such a narrow range of applications is the major disadvantage of this technique. The system presented by Huang and Yip-Hoi [77] is focused on so-called "high-level" features such as "stepped", "compound" and "array" features, see Fig. 2.17. A feature relationship graph is used to organize primitive features in user-specific high-level feature patterns. In such way, at least these categories of interacting features may be recognized, but it requires extensive user intervention, reducing the level of automation of such feature recognition system.

Di Stefano et al. [78] proposed a system which is based on a face adjacency multi-graph, precisely attributed to capture all of the properties that are important for the part's manufacturability. Attributes such as nodes and arcs of the graph are arranged in order to obtain a unique representation, so-called "intermediate model" in addition to a wide



(a) Stepped features (profile combination)

(b) Compound features (path combination)

(c) Array features (instance combination)

Fig. 2.17: High-level features [77]

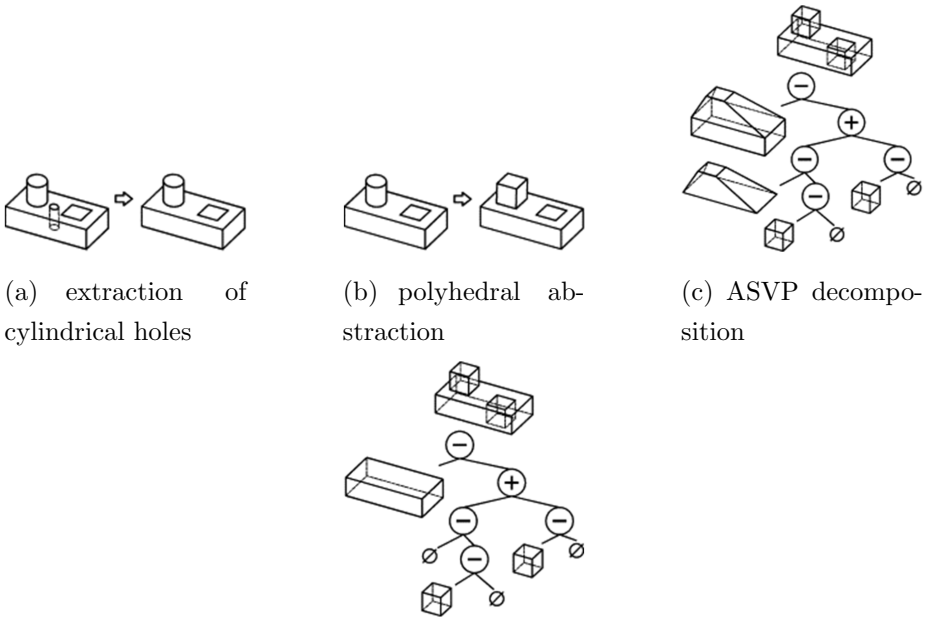
range of data needed for engineering oriented semantic recognition. However, the human intervention for the system validation in this approach is the weakness of the method despite its ability to capture a large quantity of procedural knowledge, directly from the geometric model.

All methods of the graph-based category require extensive pre-processing in order to construct representation for each part and each primitive and in most examples of their applications only polyhedral parts are treated. Even when they are capable of successful recognition, there is no guarantee that the recognized feature is manufacturable; i.e. if they are applicable in all modules of Computer Aided Process Planning (CAPP). However, the main problem which graph-based methods could not effectively solve refers to the problem of interacting features. This drawback can be partially lowered by using various techniques of geometric reasoning. The other way is to enrich the feature library with as many interacting features as possible, treating them as singular features. This approach requires a lot of computational time for searching and pattern matching and does not give a universal and complete solution of the problem, if the requested feature does not exist in the feature library. From all these reasons graph-based approaches caused larger investigation of alternative methods, such as volumetric and hint-based, to deal with interacting features [30].

2.1.7.4 Convex Volumetric Decomposition

Convex hull decomposition is an approach based on decomposing the input model into a set of intermediate volumes and manipulating the volumes in order to produce features. Kyprianou [29] gave the original idea for the convex hull approach formulized by Woo 1982 [79]. A polyhedron convex hull is determined, circumscribed around a part. The difference in volume between a part and its convex hull is defined as an Alternating Sum of Volumes (ASV). Kim 1992 [80] provided the method for convergence, initiating remedial partitioning procedure, ASV with partitioning (ASVP) to provide an effective algorithm

for the method implementation. Despite the successful application for polyhedral parts, complex convex hull computation for the curved objects could be complicated. Additionally, the difficulty of converting the set of alternating volumes into meaningful constituents of shape of the part and, further, into machining volumes has been stayed unsolved, see Fig. 2.18. Kim proposed an approach to solve this problem in several steps including (a) extraction of cylindrical holes; (b) polyhedral abstraction; (c) ASVP decomposition; (d) form feature decomposition [80][81][30].



(a) extraction of cylindrical holes

(b) polyhedral abstraction

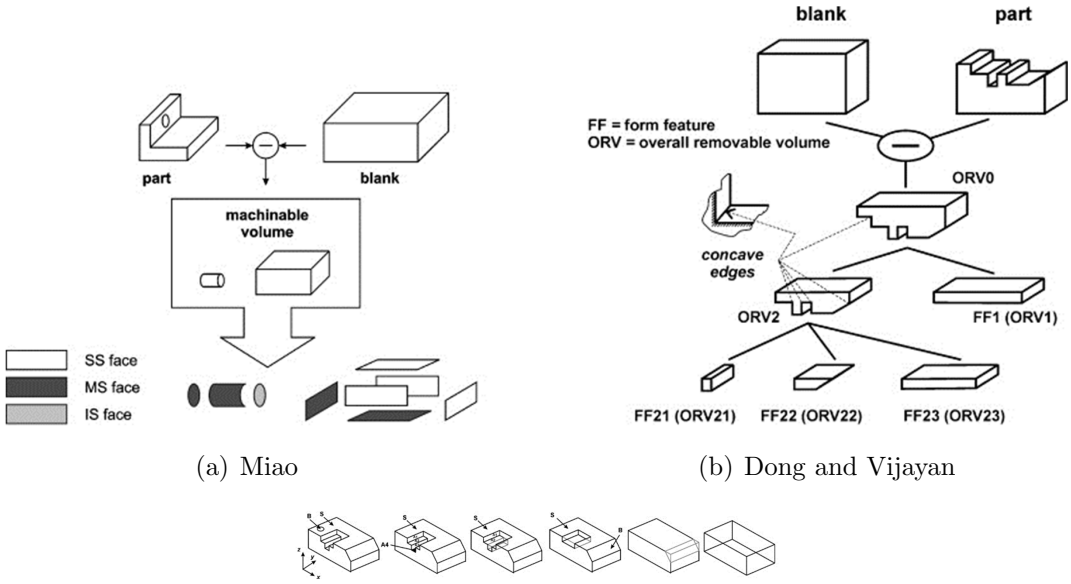
(c) ASVP decomposition

(d) form feature decomposition

Fig. 2.18: Kim's approach [80][30]

Kim's approach presented good results in various parts of manufacturing domains [80]. Though this approach is limited to the polyhedral features and cylindrical features which interact in principal directions, with constant-radius variations.

One of the examples of ASVP implementation has been shown in [82]. The proposed system has some limitations concerning forms production that can be managed only in 2.5- and 3-axis machining centers. Extraction of geometric information (primitives) is performed using an external approach, neutral STEP or IGES data file is exported from CAD model of the part, with B-rep. Faces in ASVP derive different attributes whether they are part of the stock (SS), finished part (MS) or they emerge in some intermediate stage of manufacturing (IS). A general set of forms, issued as a unique combination of SS, MS and IS is then defined as a generation attribute of the feature. Independent forms are directly recognized through pattern matching, while interacting forms have first been parsed along the concave edge loops. This process is illustrated in Fig. 2.19.



(a) Miao

(b) Dong and Vijayan

(c) Nagaraj and Gurumoorth

Fig. 2.19: AFR systems [78]

A similar approach has been developed by Dong and Vijayan. The system extracts features from a CAD model using an original technique called "blank surface - concave edge". The overall removable volume refers to the total material that has to be removed from the blank to produce a finished part. And a set of elementary manufacturing features refers to parts of volume that are removed in a single tool path. In this context, the overall removable volume is represented as a set of elementary manufacturing features. There are numerous alternatives for segmenting the overall removable volume into the machine volumes - the optimal one is selected using the mathematical optimization algorithms. The extraction of geometric features and formation of the part representation for AFR is performed through graphical comparison of the part and its blank. The pattern matching process is based on if-then rules. The illustration of the current system is given in Fig. 2.19(b). In another research, a similar method called "backward-growing approach" was introduced aiming for increasing the effectiveness of intersecting form features. This approach was implemented by Nagaraj and Gurumoorthy [83], using predefined manufacturing features. The cavity volumes, regarding the most distant outer surface of the part, are defined and, in an iterative process filled with predefined manufacturing primitives (cuboid, wedge, cylinder, etc.). These primitives are then used for CSG tree formation, in whose structure they can further be reorganized to better suit the selected blank's dimensions. The significance of this approach is to envisaging of the preformed blanks. The illustration of such system is given in Fig. 2.19(c) [84][85][30].

2.1.7.5 Cell-Based Volumetric Decomposition

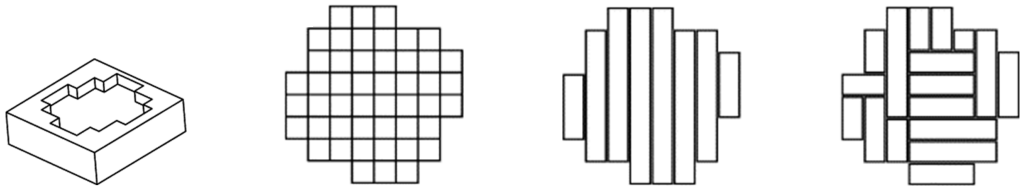
All instances of cell-based decomposition approach share a similar basic algorithm which is comprised of three steps [30]:

Step (1) the overall removable volume is identified as a difference set between the blank and the finished part;

Step (2) this volume is then decomposed to unit volumes by using the extended boundary faces as cutting planes (cell decomposition);

Step (3) all unit volumes that have common faces or coplanar faces are merged to get maximum cells that can be removed in a single tool path (cell composition).

The main drawback of this method is that even for a simple part the number of cells may be very huge which results to a large number of possible feature interpretations. This problem is addressed as "the global effect of local geometry". Sakurai in his former work proposed all multiple interpretations to be generated and then recognized through graph-pattern matching. The focus of this system was parts with planar faces and only a limited number of cases of convex curved faces. Clearly, a large number of possible combinations of cells (up to n!) led to an enormous time complexity. However, some of the interpretations were unsound from a machining point of view, see Fig. 2.20 [30].



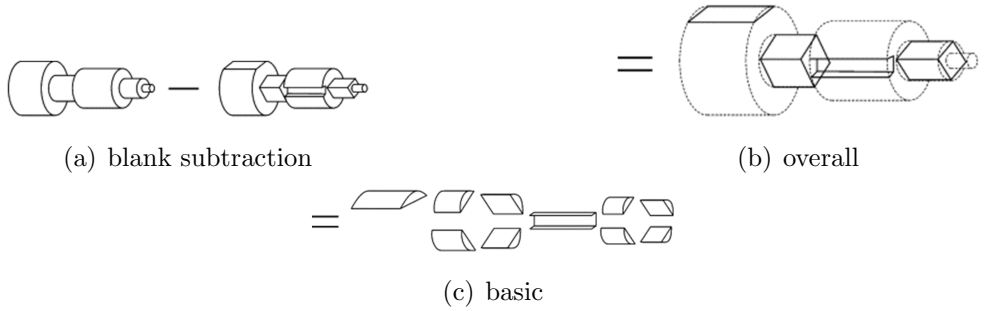
(a) part (b) cell decomposition (c) a reasonable composition result (d) an awkward composition result

Fig. 2.20: Multiple interpretations of features in cell decomposition/composition process [30]

In another work, Sakurai and Dave [86] proposed one more algorithm for concave intermediate and final volumes to extend the application of cellular method to models with all types of curved faces and reducing, ignoring "complex and awkward" features. In the latest research, Woo and Sakurai present the development of an algorithm for scalability of complex parts in order to reduce computational exhaustion and improve applicability

of the cell-based approach [30]. Tseng and Joshi [87] developed a cell-based decomposition system. The geometrical information in (B-rep) as **.txt* file in ACIS solid-modeler application protocol interface are considered as system input [88].

The feature volumes are generated by sweeping a boundary face in direction of an adjacent boundary face. The predefined type of machining operation such as rotational (turning) or prismatic (milling) decides for the direction. In addition, the applied methodology is different in one stage for rotational and prismatic part. For prismatic structures an AAG representation of the cells is made and pattern matching performed. For rotational structures as shown in Fig. 2.21, syntactic pattern recognition is performed.



(a) blank subtraction (b) overall

(c) basic

Fig. 2.21: Cell decomposition method illustration [87]

Bronsvoort and Noort [89], from Delft University of Technology has introduced a broader perspective as AFR for cell-based decomposition. He introduced a multiple-view feature modeling system for integral product development with a practical solution (System for Product modeling by Interaction on Form Features, SPIFF modeling system). They define a semantic cellular model (non-manifold geometry) which integrates the contributions from all features in a feature model. Thus, geometry is presented as a set of volumetric cells of arbitrary shape that do not geometrically overlap, and lie completely inside or completely outside the shape of a feature. Each cell comprises information on every feature that overlap with it, and each cell face contains information on the feature faces that overlap with it.

Cellular representation is an alternative to classic B-rep models. Such defined model, using geometric reasoning based on constraint solving, can be converted in specific feature models to be used in different product development phases: conceptual, assembly, part detail and manufacturing planning, and, within the latter, feature recognition [30].

2.1.7.6 Hint-Based Approach

The primary focus of developing this approach was to solve the problem of arbitrary interactions of form features as well as presenting a combination of logic approach and

delta-volume approach or graph-based approach. Heuristic, geometrical and topological information about the envisioned part are employed as the hints of presence of a certain form feature. Integrated Incremental Feature Finder (IF2) method from this category has the ability to combine design-by-features and automatic feature recognition approach into one [90]. The latest version of IF2 offers to recognize only the manufacturing features as well as consulting the tool database. This system has been designed to optimize sequencing process in the whole CAPP system. The proposed system in [90] centers the form feature recognition in orthographic and isometric projections of a part, ignoring the use of hidden lines. However, such a system cannot be considered as an input from automated visual inspection systems which AFR system is aimed for. In other research, McCormack and Ibrahim [91] employed Multi-Attributed Adjacency Matrix (MAAM) to extract geometrical information from neutral formats; IGES, STEP and **.dxf*. Their approach uses hints instead of pattern matching to extract features. The concept of "light rays" was used in another interesting research by Sommerville et al. [92]. In this method, the light rays originating from the observers' eyes are considered as solid model hints. However, this technique has many disadvantages including large number of hints and duplicates.

2.1.7.7 Hybrid Approach

Other researches [93][94] proposed an approach for combining the conventional graph-based recognition with hint-based recognition. Virtual links, i.e. the edges which are not included in B-Rep, are applied for Extended Attributed Adjacency Graph (EAAG). Ye et al. [95] applied a new approach for AFR system defined by Extended Attributed Face-Edge Graphs (EAFEG) to recognize isolated and interacting undercut specific features. This approach uses several heuristic rules for hint generation. Sundararajan and Wright [96] also developed a hybrid system as a combination of graph-based and convex-hull approach. Such a system helps in improving information about feature adjacency as well as prismatic parts and to some extent for free-form features. However, this method has a problem in recognition of fillet as a free form feature. Besides, there are restrictions regarding the direction of machining. AFR is the first and the most important step in the process of translation of CAD information into some instructions appropriate for manufacturing. The advantages of AFR over the other feature based designs are substantial. This advantage regards to saving time and human interaction while it guarantees the functionality of the desired part.

2.1.8 Introduction to Group Technology

The concept of Group Technology first was introduced by S.P. Mitrofanov who began his work in 1950s in the U.S.S.R. He published a book entitled Scientific Principles of Group Technology which was translated into English in 1966 [97]; however, it was published in the original language much earlier. Mitrofanov (1971) won the Lenin prize for his outstanding achievement in GT. He identified 3 major problems to be solved by GT: First; eliminating any unwanted variations in active technological process. Second, improving processes to a level where they would be applicable to large batch and mass production techniques and third, introducing high-speed, easily adjusted equipment Mitrofanov's work was followed up by A. P. Sokolovskiy subsequently. A second major researcher was Herwart Opitz at late 1950s and beginning of 1960s, who came up with the conclusion that although the designs and functions of the parts might be different, there are many similarities in the parts being manufactured (based on his research on the entire German machine tool industry). His work successfully resulted with the creation of the Opitz classification and coding system widely used in German industry. Third research was started from the standardization at Serck Audco Valves, a valve manufacturing company, in 1959. They developed an eight-digit coding for a unique identification for each element in the production process as a GT system. The result was 20% reduction in work pieces and purchased parts as the effect of part standardization Another attempt followed in England was done by E.G. Brisch, who developed a concept for code classification. His work later was employed in E.G. Brisch and Partnes Ltd. Furthermore, Forges et Alteliers de Construction Electrique de Jeument of France (1959) followed Brisch's coding and standardization successfully. In more recent decades, different applications of GT have been implemented and investigated. Kaperthi and Suresh used a neural network to identify features from bit-mapped 2D drawings to feed a part in Opitz code (1991). Nadir et al. (1993) and Ames (1991) developed a system which generates codes from a 3D data source. Barton and Love [98] developed a system which automatically encodes an engineering drawing to GT. As mentioned before, one of the main objectives of Group Technology is design standardization. Group Technology applies predetermined selection of characteristics of products including design engineering and functional characteristics as well as physical characteristic. As a result of the standardization, searching and retrieval systems can be improved and optimized. The significant impacts of using GT are related into its application in design and manufacturing. In industry, GT is relevant to agile manufacturing, variant process planning, manufacturing cell layout and manufacturing technology systems design and manufacturability evaluation [99]. There are different methodologies to generate GT code, and the most important ones include [100]: Opitz classification system (University of Aachen in Germany), Brisch System (Brisch-Birn Inc.), CODE (Manufacturing Data System, Inc.), CUTPLAN (Met-

cut Associates), DCLASS (Brigham Young University), MultiClass (OIR: Organization for Industrial Research), hierarchical or decision-tree coding structure, and Part Analog System (Lovelace, Lawrence & Co., Inc.). In each of the mentioned approaches, the basic idea is to capture critical design and manufacturing features of a part and place them in an alphanumeric string, or the GT code, that is assigned to that part. Based on Groover [100], the potential benefits of using group technology includes the following:

1. GT promotes standardization of tooling, fixturing, and setups;
2. material handling is reduced because parts are moved within a machine cell rather than the entire factory;
3. production scheduling is simplified;
4. manufacturing lead time is reduced;
5. work-in-process is reduced;
6. process planning is simpler;
7. worker satisfaction usually improves working in a cell;
8. higher quality work is accomplished.

Despite the main application of GT in manufacturing, GT has highly affected the design process as well. The major benefit of using GT for design is design standardization. GT generates an alphanumeric string based on the individual characteristics of a part such as geometrical shape, dimensional accuracy, material and production quality. Using GT concept in design, through the predetermined collection of characteristics, the designer has a number of alternatives and choices to initiate the new design. In this way, the designer is inspired for an innovative new design or to pursue the existing successful designs. Thus the retrieval of earlier successful design knowledge becomes an uncomplicated assignment.

There are two main significance of applying GT as a shape signature or part presentation which are as follows:

- Interactive design rather than isolated design: it is essential to consider the design process as a first step in the product development chain which is ended by manufacturing. A great geometrical design which cannot be manufactured is useless. Design is not an independent effort, but an interactive procedure. An ideal design is manufacturable at a low price while maintaining a high level of quality. Applying GT as a

manufacturing-based grouping methodology for producing design signature can fulfill the gap between Computer-Aided Design and Computer-Aided Manufacturing, see Fig. 2.22. In addition, entering GT at the early stage of the geometrical design gives the designer an improved overview of the steps that a design goes through until production.

- Product signature instead of shape signature: this is the second factor for choosing GT refers to the flexible nature of GT. The format of GT as an alphanumerical string has the capability to be extended with extra digits to contain additional product data. This characteristic promises to advance one step beyond having merely shape information.

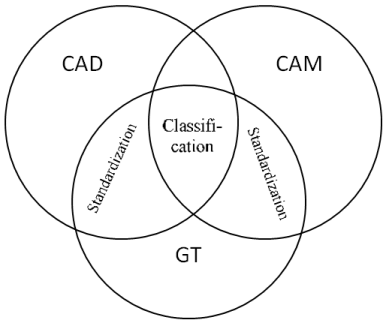


Fig. 2.22: Classification as a key association between CAD and CAM

In the next sections, two approaches for GT, DCLASS and Opitz coding system are described.

2.1.8.1 DCLASS

DCLASS (Design and Classification Information System) was developed by Professor Allen and his team from Brigham Young University, USA [101]. The structure of the DCLASS code consists of 5 basic sections which are described as follows, see Fig. 2.23.

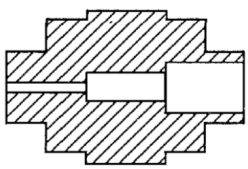


Fig. 2.23: CODE: B32144A6 [102]

The first section consists of three digits which describe the overall shape of the part. The fourth digit describes the part's form features. The fifth digit describes the part's size. The sixth digit describes the precision. And the last section consists of two digits describing the part's material. Only the first and seventh digits may take on alphabetic characters [102]. This example presents a part with no form features, of medium-small size, and made of stainless steel.

DCLASS code is an 8-digit alphanumeric coding system and its principle to characterize a part or model includes the following elements:

- the part's basic shape
- the part's special features (holes, slots, fillets, etc.)
- the part's size
- the precision
- the material type, form, and condition

2.1.8.2 Opitz Coding System

Between different methods of part classification systems in GT, Opitz coding system is one of the most well-known and widely applied approaches. Initially, Opitz coding system recognizes manufacturing features and lists them in a predefined order of digits in an Opitz code. Around 300 geometrical and topological single features are recognized as individual Opitz features to be settled in digits. Moreover; with different possibilities of combinations of single features in a code, a minimum of 30,000 part features could be recognized. An Opitz code is comprised of 14 digits divided by three categories [103]. The first category, from the 1^{st} to the 5^{th} digit, is called "form code" dedicated for design attributes. The second category, from the 6^{th} to the 10^{th} digit, is supplementary code used for manufacturing attributes. And the last category, from the 11^{th} to the 14^{th} digits, are secondary code for production operation type and sequence, see Fig. 2.24.

Opitz has 5 main digits in which each digit can have a number from 0-9. It means all together, there are 50 single features recognized in this system. However, several combinations of these features provide $10 \times 10 \times 10 \times 10 \times 10$ variations for a single part which can be identified by Opitz code. Opitz coding system provides a basic framework for understanding of the classification and coding process. Furthermore, it can be applied to machined parts, non-machined parts and purchased parts as it considers both design and manufacturing information. In the context of PLM, the variant applications of Opitz coding system can be found in [104]:

- Design: variety reduction, recognition of repeat or similar parts.
- Standards: standard components easily identified, uniformity of characteristics.
- Production planning: use of repeat, grouping parts requiring same machines, use of standard times.

- Production Control: suitability for data processing.
- Production: parts family manufacture.
- Equipment: adapting the machine tool to the work pieces required.

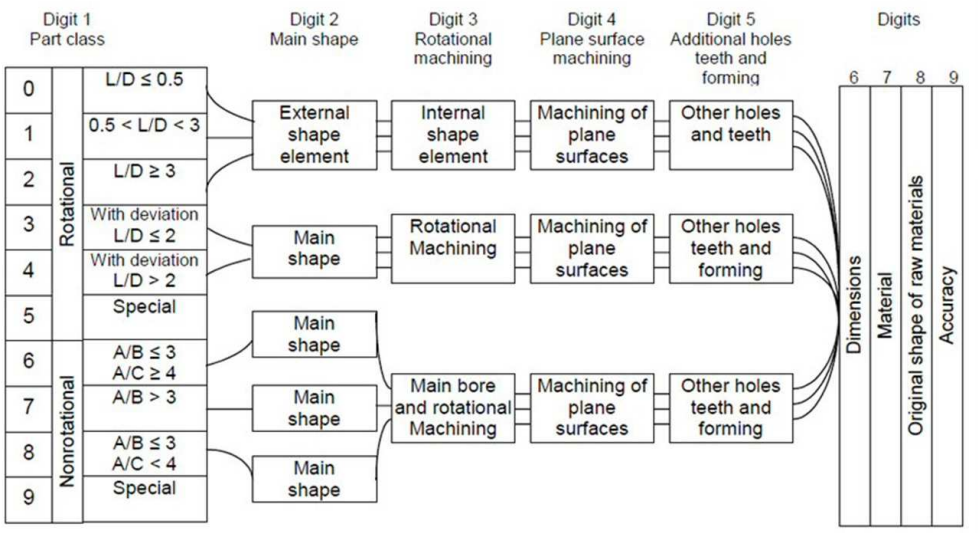


Fig. 2.24: Opitz coding system [103]

In the first category of Opitz code, and correspondingly in the form code, the first digit is the decisive digit which distinguishes between rotational and non-rotational parts [103]. Furthermore, this digit calculates a dimensional ratio to evaluate the geometry of the shape. For rotational parts the code uses the length (L) and the diameter (D) of the components in decreasing order of magnitude (A, B and C). The second digit stands for external shapes and relevant forms, and these features are recognized as stepped, conical or straight contours. Threads and grooves are also important. The third digit is for internal shapes, the features are solid, bored, straight or bored in a stepped diameter, threads and grooves are integral part. The fourth digit is for the surface plane machining, such as internal or external curved surfaces, slots and splines. And finally the fifth digit is for auxiliary holes and gear teeth. In the category of supplementary code, there are four digits in which the first one is for diameter or length of the work piece, the second one is for material used, the third one is for the raw materials like round bar, sheet metal, casting or tubing and the fourth digit is for the accuracy of the work piece.

In Fig. 2.25, the first digit divides the parts in two groups, rotational and non-rotational forms. There are more calculations and consequently categorization which decide for other features such as flat, cubic and long components. In this sense, each subsequent digit is qualified by the preceding digits. However in an object-oriented sense, each subsequent

digit inherits the properties of the previous digits. Opitz coding system as a method of GT benefits from the advantages of classification in engineering design. These advantages are the reduction of similar parts and drafting errors, easy retrieval of similar parts, as well as having an overview of parts expenses and manufacturing limitations. Moreover, it profits from its representation form as a code and having the capability of code extension increases the level of information customization. Additionally in original Opitz coding system some of the digits are preserved for the special features. Opitz coding system is a public domain and non-proprietary method which has been widely applied in industry in comparison to the other methods of GT such as KK3, DCLASS and MICLASS [104]. Some of these codes such as KK3 contain more complete information in comparison to Opitz which is merely focused on the manufacturing aspects. Few disadvantages have been mentioned for GT coding system [105] regarding the coding generation process, connection between data and code as well as a generalized overall part description. To overcome these challenges in the current research, AFR method has been used to generate GT code automatically. Furthermore, in the database all information about a part is saved together in the database to facilitate the retrieval of information about a part including CAD model, extra engineering notes as well as the Opitz code. However, the last challenge refers to classification aspect of GT code. This problem could be reduced by utilizing all digits of Opitz code; i.e. 13 digits or adding extra digits to maximize the variations in a dynamic model. In addition, GT codes including Opitz coding system are applied merely for part classification not assemblies.

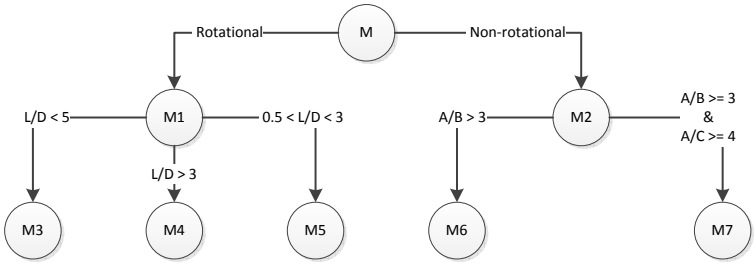


Fig. 2.25: Monocode structure for a part of the first digit of Opitz code in the code

2.1.9 CAD Model Types

In Computer-Aided Design, part representation is one of the main concepts which contain information and knowledge about a part. This information may include shape, features and dimension of a part which coexist in an explicit balance to meet physical and functional requirements [106]. Such information may be applied throughout the entire product lifecycle phases, such as in engineering design, manufacturing, marketing and

maintenance. In an ideal scheme for part presentation, this information might include the following information: Administrative (part identification, part structure), Design and Analysis (idealized models), Basic Shape (geometric, topological), Augmenting Physical Characteristics (dimensions and tolerances, intrinsic properties), Processing Information and Presentation Information [107]. The focus of the current research has been concentrated on the Basic Shape category, so the usage of "part representation" term implies merely to this category. In traditional design, the engineering drawing is considered as part representation. Nevertheless, the contemporary CAD environments provide an infrastructure for storage and representation of drawings in an electronic form. Modern computer technology has been modified from 2D drafting systems to complicated solid editors, consequently the data proved to exist in many various formats. In this regard, having a shared data format highly increases the cooperation, communication and ease of the process development among various environment users. In the early years of CAD technology, software systems were developed with an employment of translators that transform data to support the variety of environments. Such translators were successful in some ways, but as more vendors appeared in the market it became harder to provide support for all of them. Therefore, neutral data formats were introduced as well as appropriate translators for them. Some of these translators were addressed to the specific industries and others were accepted as standards by general authorized organizations. Among the most well-known neutral formats is STandard for Exchange of Product model data (STEP), Initial Graphics Exchange Specifications (IGES), Drawing Exchange Format (DXF) have gained more popularity among user communities. Based on Quintana et al. [108] the distinguished shape representation format applied in CAD software is divided in 3 major groups: Native CAD, Neutral, Lightweight format group. Fig. 2.26 presents the most famous formats of each category.

2.1.9.1 CAD Format

This format is preserved by the company proprietor regarding the registered regulations of the company. Consequently, format specification and documentation are not completely published. As a result there is a trend of replacing native CAD format to better support product information management as well as the global market place. However, native CAD format is still widely used by many companies. Main disadvantages of native CAD representations is listed as the following [109]:

- Software proprietary dependency, problem in collaborative situation
- Software is subjected to obsolescence (CATIA V4-V5-V6, etc.)

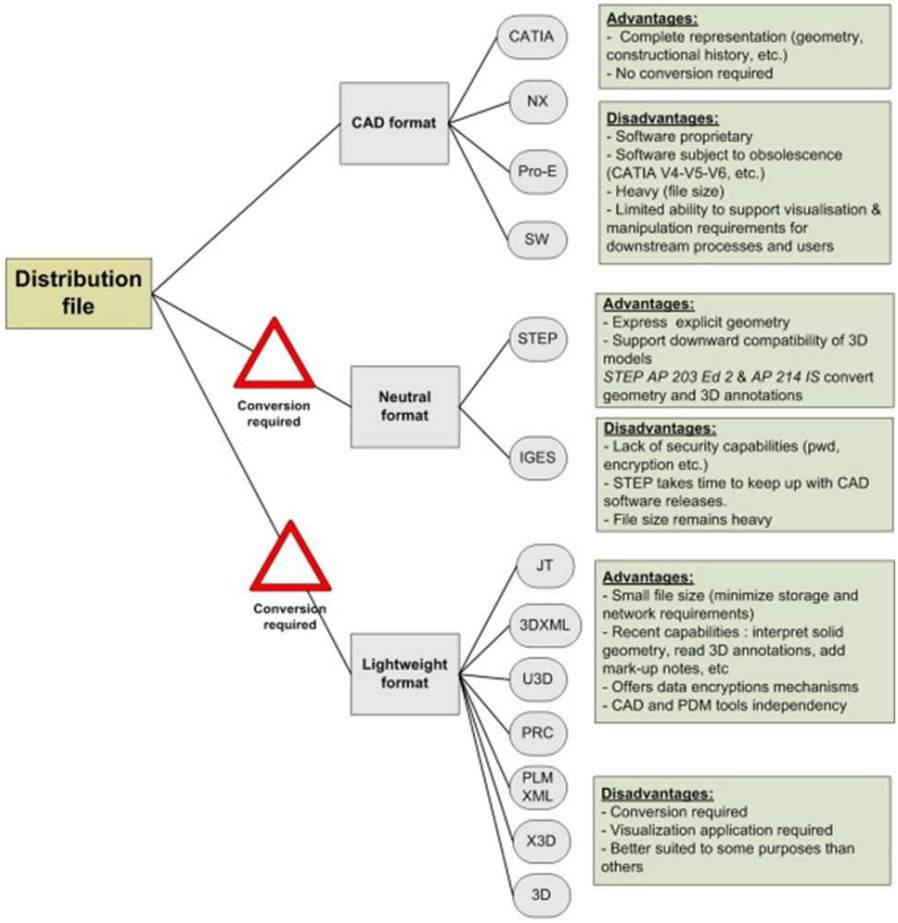


Fig. 2.26: Distribution file analyzed format [108]

- Current CAD models have big file size. This is a domain dependent drawback: there is a restriction for information transmission using internet to geographically distributed users.
- The user's limited abilities to support visualization and manipulation requirements (CAD systems might not be affordable for the entire development stream in the downstream processes).

Despite all the mentioned disadvantages for native CAD representation, There is a an important advantage of using native CAD format which is the ability to preserve specific feature of the engineering data while keeping comprehensive object information to be applied later. Among the main representation of CAD format are DWG, CATIA and SolidWorks.

2.1.9.2 Lightweight Representation

Lightweight representation methods support product data through the different stages of product lifecycle. The major characteristic of lightweight representation is a reduced file size via compression techniques, platform/application independence, open source and support for the protection of sensitive information. In addition, they can read and display 3D annotations. However, this group of formats suffer from the lack of comprehensive information as in the traditional CAD format. As mentioned the small file size which eases storage and communication is a main advantage of these methods. In addition, there are recent capabilities in software such as solid geometry interpretation, 3D annotations and mark-up notes support, etc. Lightweight representation methods offer data encryption as well as being CAD independent. However, the main failure of this category is losing information in shape representation. Furthermore, a conversion is required.

2.1.9.3 Neutral Representation

Neutral formats are based on international standard and are capable of expressing robust geometry representations [108]. Using neutral representation promises a precise, informative communications and collaboration among all of the applicants of data even in manufacturing phase. It is specially an important type of representation considering that companies are actively searching for effective approaches to carry, control, distribute and maintain the shape definition throughout the part lifecycle. Neutral format promises to improve such property. As an advantage of neutral representation, the capability to retain explicit geometry and support the downstream compatibility of 3D models is among the most significant aspects. However, the lack of security capabilities such as password and encryption in addition to heavy file size are addressed as drawbacks of using neutral representation. The major neutral representation methods are Standard for the Exchange of Product model data (STEP) and Initial Graphics Exchange Specification (IGES), Parasolid, ACIS, VDAFS, STL, VRML. In the following section STEP file format is discussed in more details.

2.1.9.4 Initial Graphics Exchange Specification (IGES)

IGES is a neutral format for defining the product representation data by means of solid modeling with B-Rep structure. It addresses a wide range of application domains such as electrical, plant design and mechanical applications. IGES includes a format by which the user can transfer the data amongst different CAD systems. To perform such data exchange, IGES requires two levels of processing: a pre-processor that takes some CAD

data as described in the system specific format and converts it into IGES format; and a post-processor that converts data from IGES back to some CAD-system format [110]. IGES is comprised of entities that are used to digital representation and communication of product definition data. These entities are stored in a domain-independent manner that enhances the product representation exchange capability across the different CAD-systems. It is concluded that the fundamental information unit in an IGES is the entity. A product description includes two general entities; geometrical and non-geometrical entities. Geometrical entities present the definition of the physical shape including points, curves, surfaces, solids and relations which are collection of likewise structured entities. Non-geometrical entities deal with viewing perspective of a drawing. This includes annotations and dimensions of drawing such as view, drawing, text, notation, witness lines and leaders [111]. The IGES-file consists of 80 column lines that are grouped into 5 or 6 sections and each section has its own code. In the following section, IGES file sections are briefly defined [112].

Flag section

An optional section at the beginning to indicate whether the file is of binary format or of the compressed ASCII format.

Start section

The start section is supposed to provide a prologue readable by a human to include some engineering information.

Global section

The required Global Section contains information describing the pre-processor and information needed by post-processors to handle the file.

IGES data can persist in either an ASCII or a binary format. The code identifying a section is placed in the 73rd column of each line. In columns 74-80 the sequence number of every line within a section is stored. This sequence number is a seven digit number starting from 1, sequentially increasing by 1 with leading space or leading zero fill in accordance to the line number. Columns 1-72 store the data specific to the entity.

Directory entry section

The goal of this section is to provide an index for a file as well as storing the attribute data for every entry. The order of the records within the current section is arbitrary. Some of the fields in the Directory Entry may contain either an attribute value or a pointer to an entity containing one or more such values.

Parameter data section

Parameter data associated with each entry is saved in this section. It is formatted in a free way in which the first field indicates the entry type number. The entity type number and a parameter delimiter precede Index 1 of each entity in the exchange file. The file formatted in a freeway part of a parameter line ends in Column 64. Column 65 comprises of a space character. Furthermore columns 66 through 72 on all parameter lines contain the sequence number of the first line in the Directory Entry of this entity. Column 73 of all lines in the Parameter Data Section contains the letter P and Columns 74 through 80 contain the sequence number.

Terminate section

Terminate section includes one line to signal the end of file. In addition, it comprises of minor information like number of lines inside each section.

IGES Drawbacks

IGES is essentially a neutral format that enables the product data representation exchange; therefore it has a clear advantage over native translators. Even though it is overall applied and employed in the various industries, IGES has several drawbacks. Although its 80-column files are very verbose, there might still be complications to understand them for the reason of their complex construction. That's why if there is a syntax error in IGES-file recognized, it will be hard to trace its position to be corrected. Moreover, errors may appear due to the changes made to the file [113]. CAD vendors usually develop their individual IGES due to the absence of any conformance requirements imposed by IGES. This leads to inconsistency of IGES files among vendors in addition to incomplete translation and loss of information. IGES offers only data related to drawing or 3D modeling data ignoring other information about engineering, process plan or manufacturing of a part. This is a major disadvantage specially for the current research as the focus is on a having a comprehensive data structure. Based on [113], IGES has the following drawbacks:

- Lack of a formal information model
- Problem of incomplete exchange due to the individual 'flavors' added by vendors
- No support for PLM data
- Ambiguity of 80-column format which is not easily understandable for human (in case of error)

2.1.9.5 Standard for Exchange of Product Model Data (STEP)

Amongst the different neutral files available in CAD software, STEP is the most popular one. STEP is intended for product data exchange, while IGES is more for geometry data exchange. The description of product data for mechanical parts has been standardized by ISO10303 for the computer interpretable representation and exchange of product manufacturing information. The official title is automation systems and integration-product data representation and exchange, informally called STEP. The most tangible benefit of using STEP is the ability to exchange design data as solid models and assemblies of solid models. In fact, using STEP file gives an opportunity to combine data representation together with product lifecycle data including different phases of design, production, utilization, support. Obviously, such a data exchange is done among different computer systems and even in diverse geographical locations. Supporting of such a distributed cooperation requires a consistent data representation for exchange of information. STEP consists of a series of distinct parts including description methods, integrated resources, application protocols, abstract test suites, implementation methods and conformance testing. STEP uses EXPRESS, a formal data specification language applied for representation of product information. Application Protocols (APs) define one of the parts of STEP that belongs to the Integrated Resources series. These APs use the low-level information of current series in form of combinations and configurations to represent a particular data model of an engineering or technical application. Some of its applications related to the context of this research are as following:

- AP203: Configuration controlled 3D design of mechanical parts and assemblies
- AP214: Core data for automotive mechanical design processes
- AP224: Mechanical product definition for process plans using machining features

The current research has applied AP203 to develop its structure. Thus, this application is discussed in more details in the following sections.

STEP AP203

Currently, the most widely used AP is AP203 which is designed for representing 3D geometry and configuration management information [110]. It is named “Configuration Controlled 3D Designs of Mechanical Parts and Assemblies” (ISO 10303-203). Below a scope and conformance classes of AP203 is presented to give a better overview of this application protocol [114].

AP203 Scope main clauses:

- Five types of shape representations of a part that include wireframe and surface without topology, wireframe geometry with topology, manifold surfaces with topology, faceted boundary representation, and boundary representation
- Products that are mechanical parts and assemblies
- Product definition data and configuration control data pertaining to the design phase of a product's development
- The change of a design and data related to the documentation of the change process
- Identification of government, industry, company or other specifications for design, process, surface finish, and materials which are specified by a designer as being applicable to the design of the product
- Data that are necessary for the tracking of a design's release
- Data that is used in, or results from, the analysis or test of a design which is used as evidence for consideration of a change to a design

AP203, Edition 1 has 12 Conformance Classes:

- cc 1a, b: Configuration controlled-design information without shape (cc 1a is a specified "product identification" subset of cc 1b)
- cc 2a, b: cc 1a, b and 3D geometrically bounded wireframe and/or surface models
- cc 3a, b: cc 1a, b and 3D wireframe models with topology
- cc 4a, b: cc 1a, b and manifold surface models with topology
- cc 5a, b: cc 1a, b and faceted B-Rep
- cc 6a, b: cc 1a, b and advanced B-Rep

It can be seen that Conformance Classes of AP203 includes such information as 3D shape description (bounded wireframe, surface, B-Rep). By default AP203 stores 3D data as a B-Rep format. Root element 'Solid' contains the complete definition of the 3D model geometry and topology. The outer extent of this solid is defined by a closed shell. Closed shell consists of faces, which are defined by advanced face. Then every face is represented by outer loops and inner loops which are, from their side, defined edge loops. An edge loop

consists of oriented edges. Oriented edges in turn consist of edge curves which are represented by vertex points and edge geometry (vector direction, start point, etc.). Current geometric data can be used on later stages, for feature extraction as an example [115].

STEP Application Domains

Within STEP specification there are Application Suites which address specific industrial domains and operations. Among these domains the following suites are mentioned: the Ship building suites, the Electromechanical Suite, the Process Plant Suite, the System Engineering Suite, the Engineering Analysis Core Model, Product Lifecycle Support and the Manufacturing Suite are provided which address to general application domains. The following Suites, in contrast to a single Application Protocol, employ a series of Application Protocols [114]. The share of STEP awareness continues to grow that leads to the gain of its industrial acceptance in such spheres as automotive industry, defense industry, aerospace and ship building industry. The overall amount of STEP-based applications keeps on rising for the last years [110]. In the following some of the main Production implementations of STEP are listed.

CSTAR

C-17 STEP Transfer and Retrieval. Went through production in 1995 at McDonnell Douglas (Boeing) having AP203 cc1.

AEROSTEP/PowerSTEP (Boeing)

Went through production in 1995 with Rolls Royce (Catia/CADDS5 - AP203 cc6) Went through production in 1996 with General Electric and Pratt & Whitney (Catia/UG - AP203 cc6). In 1997 entered into agreement with Rolls Royce, General Electric, and Pratt & Whitney to exchange data using STEP AP203 to support digital preassembly verification for the 777 and 767-400 aircrafts.

General Motors STEP Translation Center

Went through production in 1996 to test and validate surface and solid model data exchange. Extensive STEP/IGES comparison analysis. CATI/UG translation services with GM Powertrain, Delphi/Delco Electronics, and Delphi Automotive divisions.

Lockheed Martin

Tactical Aircraft Systems. Went through production in 1998 with the use of CATIA STEP AP203 translators for data exchange on the F-16, JSF, F-22, KTX-2, and F-2 aircraft Programs. In 1999, Lockheed Martin-Tactical Aircraft Systems (LM-TAS),

undertook the Virtual Product Development Initiative for Finite Element Analysis (VPDI-FEA) using AP209 DIS.

NASA

The policy that STEP Translators are required to be available at all NASA sites stated.

Closure for Native CAD Representation

As presented, the native CAD representation appeared as bulky and proprietary shape formats; their specifications are closed that contradicts with the ideas of effective information exchange. On the other hand as lightweight representations has been kept aside for this research as it implies data loss. Furthermore, merely a precise geometrical data can be used for feature extraction. So the choice fell on the neutral format group.
Two market leaders of neutral format are IGES and STEP. IGES tendency is to present only geometry information. As the scope of this research is on product information rather than only geometry, STEP was selected for reasons of evident strengths and overcoming mentioned issues. In this regard, STEP ISO 10303-AP203 cc5a (or cc6a) appeared as the most suitable protocol for shape data description and exchange for the further development of this research. In the next chapters the details of this development will be presented and discussed.

2.1.10 Distance Function

Distance function defines a distance/similarity between each pair of components of a group. If a group consists of shape signatures in design concept, the distance function would be the unique distance between two shape signatures. A distance function in general should benefit from the following specifications:

Positivity:

Distance between CAD models is positive $\sigma(S(x), S(y)) \geq 0$

Identity:

Self-similarity $\sigma(S(x), S(y)) = 0 \Longleftrightarrow x = y$

Symmetry:

$\sigma(S(x), S(y)) = \sigma(S(y), S(x))$

Triangle Inequality:

$\sigma(S(x), S(y)) + \sigma(S(y), S(z)) \geq \sigma(S(x), S(z))$

Invariance:

Independent of representation, Invariant to the transformation

Shape signatures have various types and natures such as feature-based, spatial functions and shape histograms as Fig. 2.27 presents. Thus the distance function should be correctly selected to fit the signature type.

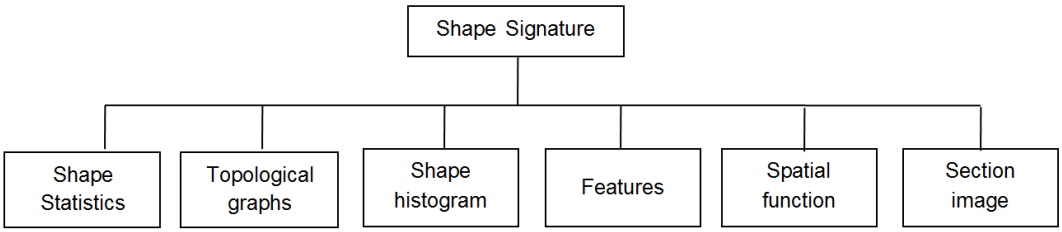


Fig. 2.27: Various types of shape signature

Most of the mentioned methods benefit from their individual distance function. Apart from these specific methods, in general the methods of distance function are divided into two groups of Euclidean distance and non-Euclidean methods. Based on [116] an Euclidean space has some number of real-valued dimensions and dense points. Euclidean distance is based on the location of points in such a space. A Non-Euclidean distance is based on properties of point but not their location in a space. Two examples of non-Euclidean distances are presented in the following including Cosine distance and Minimum Edit distance. The well-known example of Euclidean distance d(X, Y) is the square root of the squares of the differences between X and Y in each dimension.

$$if \ \{X = (x_1, x_2, \ldots, x_n) \,\&\, Y = (y_1, y_2, \ldots, y_n)\}$$

$$d(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots (x_n - y_n)^2} \tag{2.1}$$

2.1.10.1 Manhattan Distance

Manhattan Distance is an Euclidean distance introduced by Hermann Minkowski and is defined as the following. The distance between two points measured along axes at right angles [117]. In a plane with P1 as (x_1, y_1) and P2 at (x_2, y_2), it is:

$$(p_1, p_2) = |x_1, x_2| + |y_1, y_2| \tag{2.2}$$

$$d_{MK} = \sqrt[p]{\sum_{i=0}^{d} |p_i, p_i|^p} \tag{2.3}$$

For any two vectors, the Minkowski distance is calculated as shown in Eqn. 2.3, after choosing an appropriate 'p'. This distance function calculates the distance between each two elements within a vector to the p$^{\text{th}}$ power, and the sum of those distances is then raised to the $1/p^{th}$ power. It usually results in a real number X between 0 and positive infinity ($0 \leq X \leq \infty$). It is also the generalization of the following distance function, the Euclidean distance [118]. Manhattan distance is often used in the integrated circuits where wires only run parallel to the X or Y axis [119].

2.1.10.2 Cosine Coefficient

The cosine coefficient computes the cosine of the angle between two vectors. The nominator is the scalar product between the two vectors in question, while the denominator is the product of the norms of the two vectors. Deviating from the previously introduced functions, this function results in a real number X between -1 and 1 ($-1 \leq X \leq 1$). The closer the value is to 1 or -1, the more similar the vectors are, and a value moving closer to 0 indicates a growing dissimilarity between both vectors.

$$s_{Cos} = \frac{\sum_{i=1}^{d} P_i, Q_i}{\sqrt{\sum_{i=1}^{d} {P_i}^2} \sqrt{\sum_{i=1}^{d} {Q_i}^2}} \tag{2.4}$$

In this research, the shape signature applied is based on the features obtained by using the Opitz coding system. Features are presented in an alphanumerical string form. To compute the distance function between two Opitz codes, the cosine coefficient (similarity) method has been applied, Eqn. 2.4 based on Cha [120]. The following chapter describe the applied method thoroughly.

2.1.10.3 Minimum Edit Distance (MED)

As depicted in the work by Jones and Pevzner [118], the Minimum Edit Distance (MED) is used to assess the similarity of one string (v) to another (w). The distance between (v) and (w) is measured by the amount of editing operations which would take to transform (v) into (w) or vice versa.

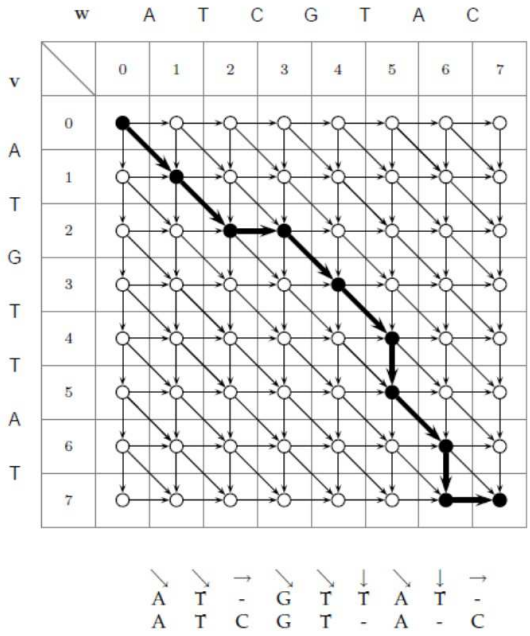


Fig. 2.28: MED and Alignment Grid Example [118]

The permitted operations in this case are inserts, deletes (called indels since they are usually assigned the same weight), and substitutions. The most popular implementations of this distance function usually feature an $n \times m$ matrix where n is the number of letters in v and m is the number of letters in w. An example of one such matrix can be found in Fig. 2.28. Depending on the operation itself, one moves along the matrix adding up the costs of the operations until the bottom right cell is reached. The path with the minimum cost is then computed and returned as the sequence of operations needed. The cost for that path represents the MED.

Chapter 3

The Proposed Approach

3.1 Architectural Design of the Proposed Approach

The proposed approach consists of five distinct modules which interrelate. This includes a GUI module, a feature recognition module, a similarity-retrieval module, a CAD model reader module and a repository interface module.

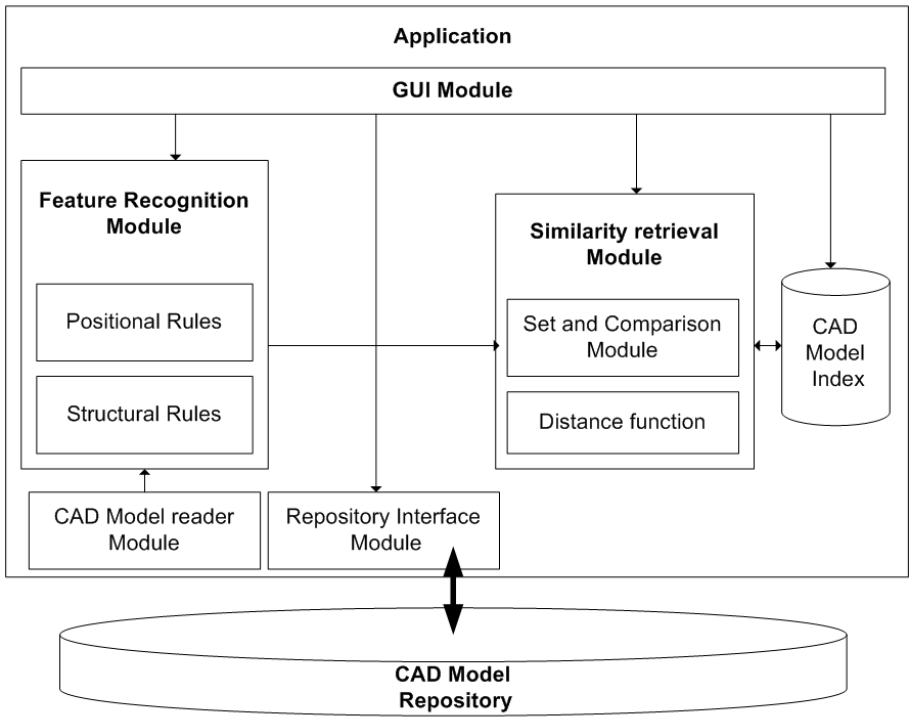


Fig. 3.1: The application architecture diagram

CAD Model Repository and GUI module interrelate with the core. The external module

which connects the system with user is the interface presented by GUI module and the interior model to host the data is the repository module, the fifth unit of the system. Additionally, a CAD model repository which is internal to the system and belongs to the system core. The system architecture is depicted in Fig. 3.1. The CAD model reader module is responsible for reading CAD model files, and translating them into a set of objects that the application can deal with. This set of objects is later applied by the feature recognition module. The feature recognition module uses these objects to identify and recognize features relevant to the required model signature (Opitz code). The resulting signature is then visible through the application's GUI and can also be used as an input to the similarity retrieval module. The similarity retrieval module will compare the given signature to the internal CAD model index while exposing the query parameters to the user via the GUI. Once the parameters are set, user can initiate the search function. The result is a list of CAD models retrieved from the CAD model index which includes the requested parameters. The repository interface module is used to update the CAD model index so that it conforms to the latest version of the CAD model repository. The separation of storage mediums (local and external) was chosen since it provides greater portability and flexibility regarding different types of repositories. Furthermore, it is more efficient for query execution. The feature extraction applies a Rule Based System (RBS) as a method of Automatic Feature Recognition (AFR) for constructing the Opitz code. The feature library was constructed according to the Opitz coding system and consequently uses Opitz features to fulfill the requirements of RBS. The classification core includes two distinct modules, a feature extraction module to extract features from STEP and a feature recognition module to construct Opitz features with RBS assistance and to generate the Opitz code. During this process of analysis, the incoming STEP data should be parsed for the purpose of Opitz code feature extraction with further assignment to the Opitz Classification System. These two main phases include recognition of the 3D-shape information (feature recognition) and constructing the code (feature extraction).These two modules are required to acquire Opitz features from a CAD file with STEP format as a neutral CAD file format. Based on the discussion in section 2.1.6, the STEP file format has been selected as the input format of the system. If the input of the system, i.e. the CAD file format is not a STEP file or if the form data is available, the first and second stages are not used and the third stage which is a RBS for Opitz code generation is directly applied. In the following, Fig. 3.2 presents the proposed framework with the details of the core system and its module interactions.

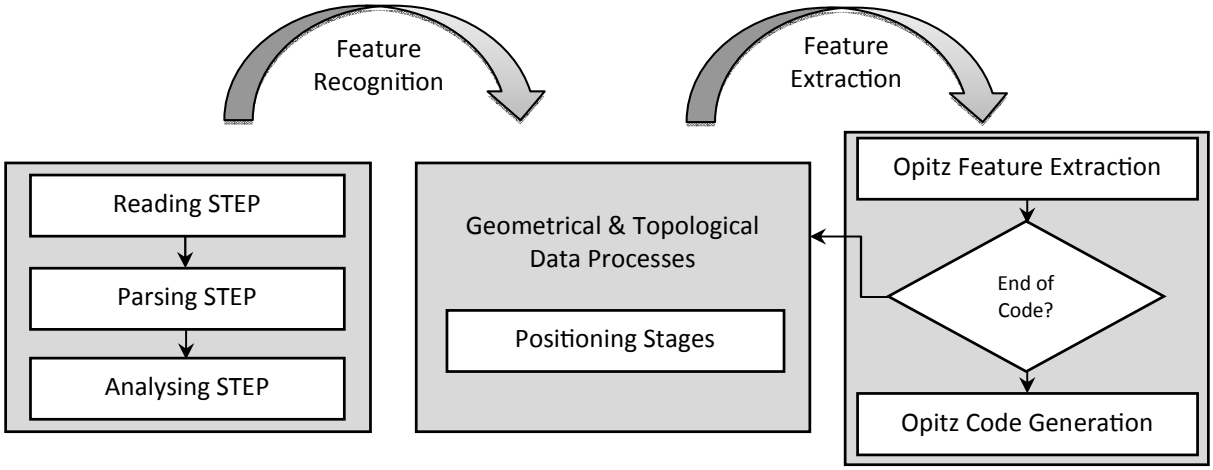


Fig. 3.2: The proposed rule based system framework

3.2 Opitz Feature Recognition

STEP defines a readable text encoding format of the STEP neutral file. It is comprised of two parts: the HEADER segment which contains the information related to STEP converter version and the 3D modeling CAD software. The second part is the DATA segment which includes a geometric entity definition and data elements. The geometry entity has been divided into many entity instances in which each entity has its own unique ID number or reference number (#N, N is a positive integer) or a pointer. The entity instances can be mutually nested and called constituted the geometry entity or the characteristic of entities. The same entity instances can also be called repeatedly by several instances. There are many kinds of data elements in the STEP file, and some examples follow in the next section. "CLOSED_SHELL" means one or more planes "ADVANCED_FACE" set, the plane defines a three dimensional geometric space. "ADVANCED_FACE" is a type of surface, and form a certain geometry related to the surface, the boundary and vertex. It has a topological character which describes a set of inner loops placed within one outer loop on the same surface. Also "ADVANCED_FACE" must contain one of the surface geometry entities: "CYLINDRICAL_SURFACE", "CONICAL_SURFACE", or PLANE, each of them includes "AXIS2_PLACEMENT_3D" entity that represents one normal (start point plus direction) to surface. "FACE_BOUND" is to compose a closed boundary plane. "EDGE_LOOP" expresses a closed orbit with starting point and end point in coincidence, it constitutes "FACE_BOUND". "ORIENTED_EDGE" is a border which composed of geometry boundary "EDGE_CURVE", such as a line, a section of arcs; the difference is that it has orientation. "EDGE_CURVE" means the boundary of geometry outline, it has no orientation. "VERTEX_POINT" means the vertex of geometry shape. "CARTESIAN_POINT" is the point in Cartesian coordinate, the behind of the

brackets is the coordinate values in the Cartesian coordinate system [121]. As presented in Fig. 3.3, the process of parsing the STEP file for extracting the geometry includes different phases, started by reading STEP file. Every STEP file includes lines presenting predefined geometrical entities. Examples of these entities are:

FACE_OUTER_BOUND (“NONE”, #101, T.)

EDGE_CURVE (“NONE”, #94, #96, #170, T.)

The reference symbol (#) addresses some other entities for forming a graph of all entities to represent a complete shape description model. As Fig. 3.3 demonstrates, the STEP-file “CLOSED_SHELL” contains a root entity that should be first identified in order to start the feature extraction process. It has a set of links to “ADVANCED_FACE” entities.

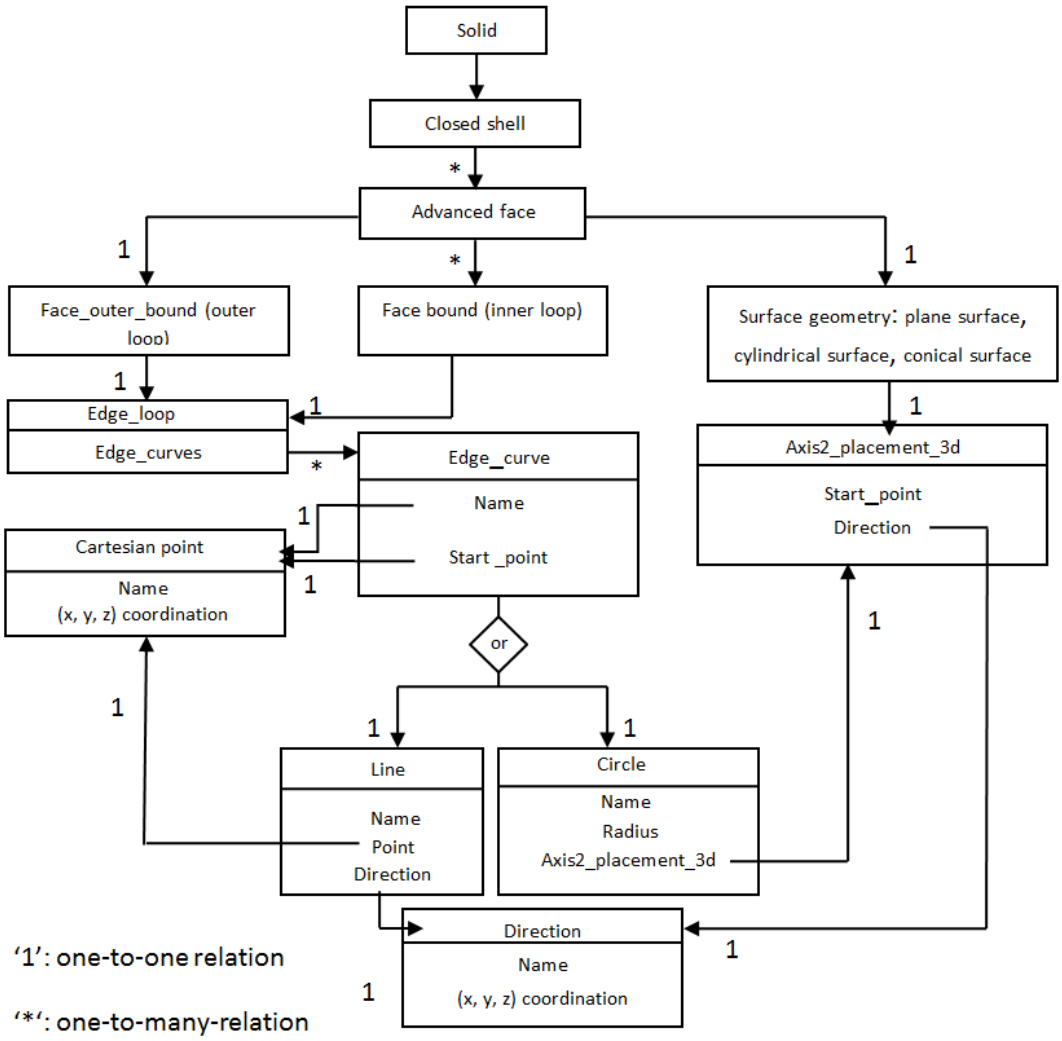


Fig. 3.3: STEP (AP203) structure based on [122][114][121]

The STEP file provides a sufficient and complete 3D geometry description model. For ex-

ample, the “EDGE_LOOP” entity consists of “EDGE_CURVEs” entity which is formed by “VERTEX_POINTs” and VECTORs entities. In order to start feature extraction process, the “CLOSED_SHELL” entity is a root entity which has to be identified primarily. It has a set of links to “ADVANCED_FACE” entities. “ADVANCED_FACE” has topological information to describe a set of inner loops placed within one outer loop on the identical surface.

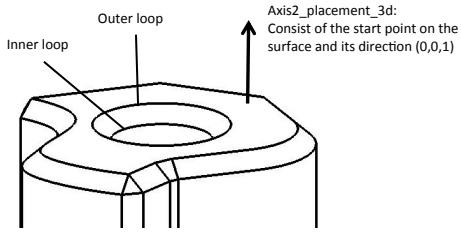


Fig. 3.4: Advanced face example

In addition, the “ADVANCED_FACE” ought to contain one of the surface geometry entities including the “CYLINDRICAL_SURFACE”, “CONICAL_SURFACE” or PLANE. Each of them includes an “AXIS2_PLACEMENT_3D” entity that represents one normal (start point plus direction) to the surface as seen in Fig. 3.4. The Cartesian point and the direction to demonstrate 3D space is presented in Fig. 3.5.

- DIRECTION(”NONE”,(0.000000000000000000,1.000000000000000000,0.000000000000000000))
- CARTESIAN_POINT(”NONE”,(0.000000000000000000,0.000000000000000000,0.000000000000000000))

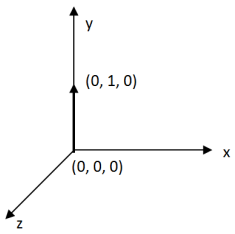


Fig. 3.5: Cartesian point and Direction (normal to surface XZ) example

Every “FACE_OUTER_BOUND” and “FACE_BOUND” has a one-to-one relation with an “EDGE_LOOP” entity. The latter one represents a set of adjacent “EDGE_CURVES” entities forming shape boundaries on a surface, see Fig. 3.6. An “EDGE_CURVE” entity always includes start point, end point and its edge geometry. When the edge geometry is a LINE entity, a direction and a start point of a vector is inscribed within this entity. And when the edge geometry is a CIRCLE entity, an “AXIS2_PLACEMENT_3D” (with a vector starting from the point on the top middle of an arc) entity and a radius of the arc is inscribed within this entity. The directions in both last cases help to decide an orientation

of a particular edge that is useful for edge curves relationship calculations (e.g. angles between lines).

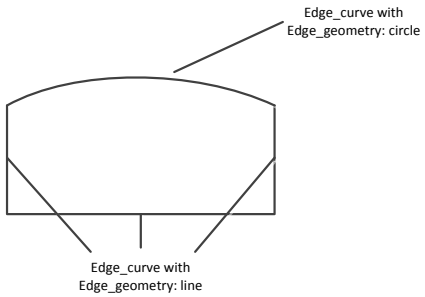


Fig. 3.6: Four edge curves forming one edge loop, a reflection of a shape

As was described earlier in Chapter 2, the input of the system is a CAD file with STEP file format. The geometrical as well as the topological data of a mechanical part is stored in its STEP file. In order to extract this data, STEP file is analysed and the resulting information is parsed for the next step which is feature recognition. The feature recognition from STEP has been tailored according to the next phase of the methodology which is Opitz feature extraction. In this regard and based on the Opitz classification, forms are divided into two main categories including rotational and non-rotational forms. Therefore, the STEP file is parsed and analyzed accordingly. In the next section, the applied methodology of STEP feature recognition in respect to the individual feature recognition of Opitz coding system is described in detail.

3.2.1 Opitz Feature Recognition of Non-Rotational Components

The main form in a non-rotational component describes the rough shape of the machined work pieces. There are three main categorizations for the non-rotational forms in which each category has its own sub-classifications. These three categorizations are the flat part, long part and cubic part. Small deviations and gaps are not included in the assessment of the main form.

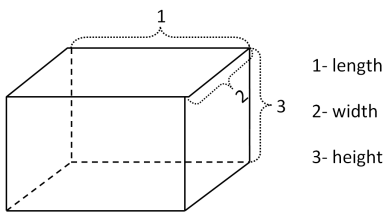


Fig. 3.7: Shape measures

To relate a non-rotational part to a predefined component class of the Opitz code, three measures should be considered: part length, part width and part height as shown in Fig. 3.7. The result of deviation of $(length/width)$ and $(length/height)$ determines if a part belongs to the category of flat, long or cubic based on the following calculations:

a) $$\left.\begin{array}{c} \frac{length}{width} \leq 3 \\ \& \\ \frac{length}{height} \geq 4 \end{array}\right\} flat\ part$$

b) $$\left.\begin{array}{c} \frac{length}{width} \leq 3 \\ \& \\ \frac{length}{height} < 4 \end{array}\right\} long\ part$$

c) $$\left.\frac{length}{width} > 3\right\} cubic\ part$$

The forms are divided into two main categories of rotational and non-rotational. After the main categorization, the feature recognition is started. The rules for feature extraction are classified into general rules for each grouping and specific rules for each individual feature. Each grouping includes different sets of features, for example, see Fig. 3.8 which depicts the main categorization of Opitz coding system. The general rules, are repeated for each individual feature extraction from a group followed by specific rule(s). Such distinguishing is based on the hierarchical nature of the Opitz code.

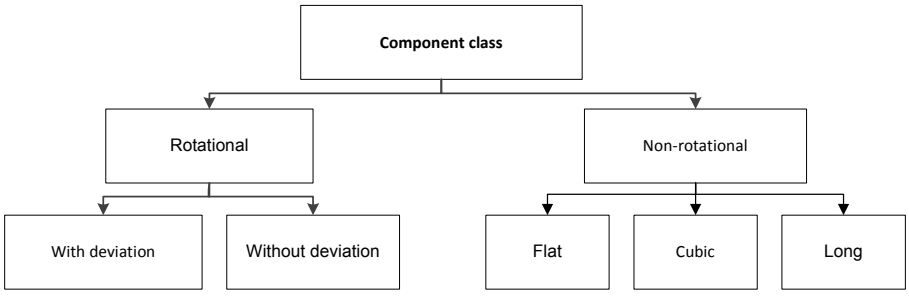


Fig. 3.8: The main classification of Opitz coding system

Starting with the non-rotational classification, the following sections describe the components varieties of flat, cubic and long components and their feature recognition in details. Each of these components is comprised of extended features; in which their association is presented in a diagram at the beginning of each section. In addition, two types of rules

including the general rules which are repeated in each feature extraction of dependent features as well as the specific rules to extract each individual feature are explained. At the end of this section, some of the features which are common between the non-rotation components are discussed and analyzed as well (section: Recognition of principal bore and rotational surface machining).

3.2.1.1 Opitz Non-Rotational Flat Components

In this section, the details of recognizing some of the key features of flat components in the Opitz coding system from STEP file are explained [115]. These features are presented in Fig. 3.9. As it has been mentioned earlier, the Opitz coding system has a hierarchical structure and that's the reason for defining general rules for each main category of Opitz features.

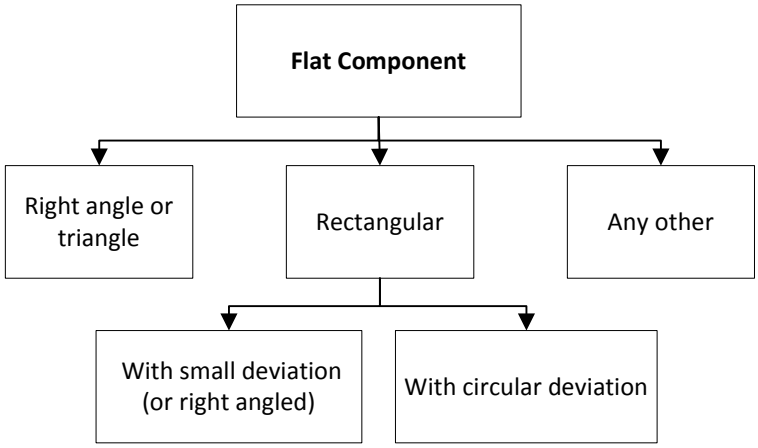


Fig. 3.9: Main features recognized from STEP for flat components in Opitz

The general rules for non-rotational flat components consist of three rules which are repeated in every feature of this category, as depicted in Fig. 3.10. These rules are as following:

General rules for positioning of flat components:

- Bottom surface should be found which is plane (not cylindrical or conical surface), parallel to Y-plane and has the minimal Y coordinate.
- Area of the bottom surface should be bigger than the Y-plane cross-section of each cylinder orthogonal to this plane.
- All adjacent surfaces to an outer loop of the bottom plane must be orthogonal to this plane as well.

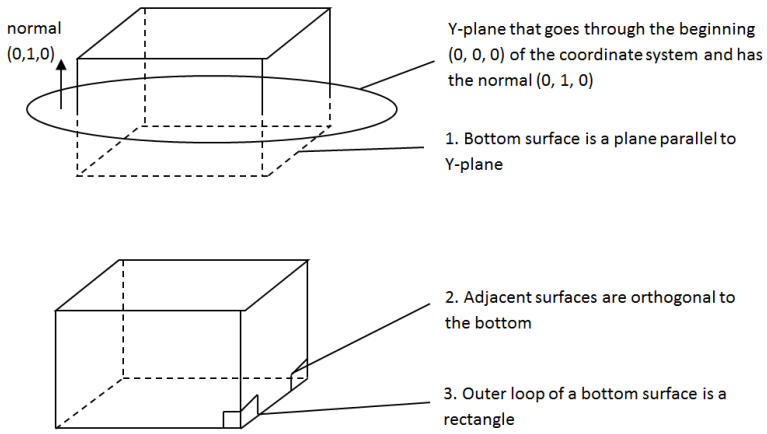


Fig. 3.10: Main aspects of feature extraction of the rectangular flat component

Specific rule for rectangular form feature

- Outer loop of the bottom plane must be a rectangle, see Fig. 3.10

Specific rule for right angle or triangular flat form feature

- Outer loop of the bottom plane must be a triangle, see Fig. 3.11.

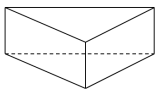


Fig. 3.11: Example of triangular flat component

Specific rule for angular flat form feature

- Outer loop of the bottom plane should have more than 4 edges, with the same angle between them, see Fig. 3.12.

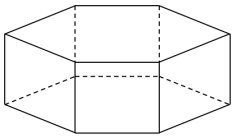


Fig. 3.12: Example of angular flat component

Rectangular flat with deviations form feature

- This feature is divided into two subsections: Rectangular flat with circular deviations form feature and rectangular flat with right angle and deviation form feature.

Specific rule for rectangular flat with circular deviations form feature

- Outer loop of the bottom plane should consist of linear edges and one circular edge, see Fig. 3.13.

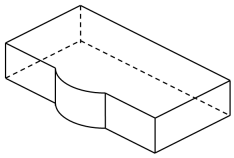


Fig. 3.13: Example of rectangular with circular deviation flat component

Specific rule for rectangular flat with right angled deviation form feature

- Outer loop of the bottom plane should consist of linear edges that form a shape with small deviations, see Fig. 3.14.

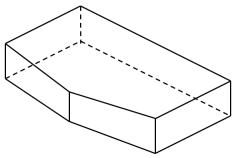


Fig. 3.14: Example of rectangular component with small frontal deviation

All other variation of the flat components and deviations are classified under another category.

3.2.1.2 Opitz Non-Rotational Long Components

Similar to the flat components, the general rules have been defined for the long components as well as follows, see Fig. 3.15.

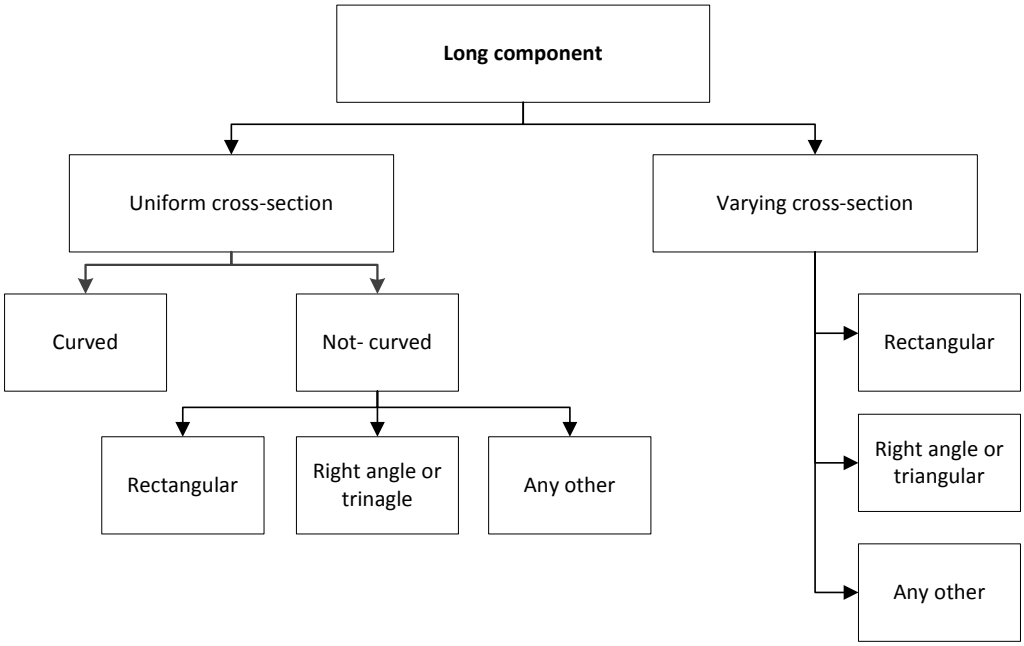


Fig. 3.15: Main features extracted from STEP for long components in Opitz

General rules for long components:

- Front and back surfaces should be found which are plane (not cylindrical or conical surface) and parallel to Z-plane having the maximal and minimal z-coordinates within current shape respectively, see Fig. 3.16.
- Outer loop of the front plane equals to the one of the back plane: for each vertex of the front plane's outer loop there should be a vertex on the outer loop of the back plane having the same x, y coordinates, see Fig. 3.16.
- Shape axis should be straight and have a direction (0, 0, +/-1). This condition is met when all adjacent surfaces to front or back plane's outer loop are plane.

Specific rule for forms with rectangular (uniform) cross-section and curved

- Outer loop of the front and back plane must be rectangular.

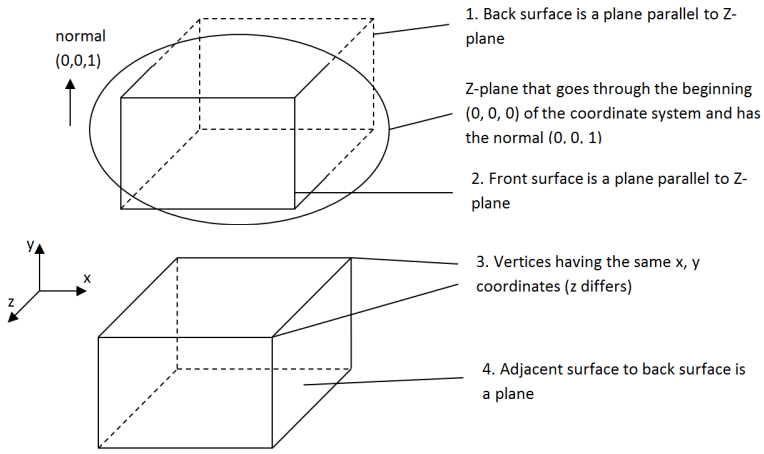


Fig. 3.16: Example of the rectangular long component with uniform cross-section

Specific rule for right angle or triangular long form feature

- Outer loop of the front and back plane must be triangular, see Fig. 3.17.

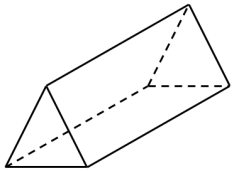


Fig. 3.17: Example of the triangular long component with uniform cross-section

Specific rule for any uniform cross-section other than the mentioned forms

Any uniform cross-section form features other than rectangular and triangular long form features pursue the specific rule as follows:

- Outer loop of the front and back plane must not be triangular and rectangular, see Fig. 3.18.

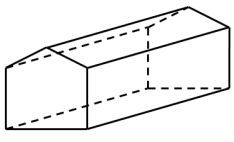


Fig. 3.18: Example of the long component with uniform cross-section other than 0 and 1 (not rectangular and not triangular)

Uniform cross-section and curved

In this section, rectangular, angular or any other cross-section long components with curved shape axis are analyzed, see Fig. 3.19. For this category, the general rules are defined as follows:

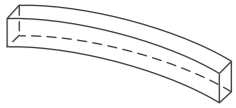


Fig. 3.19: Example of the long component with curved shape axis

General and specific rules:

- Front and back surfaces should be found which are plane, i.e. not cylindrical or conical surface and parallel to Z-plane. In addition, they have the maximal and minimal z-coordinates within current shape respectively.
- Shape axis should be curved.
- Outer loop of the front and back planes should be rectangular, angular or have other cross-sections.

Varying cross-section

General rules for extraction of a long component with varying cross-section are as follows:

- Front and back surfaces should be found which are plane (not cylindrical or conical surface) and parallel to Z-plane having the maximal and minimal z-coordinates within the current shape respectively.
- Shape axis should be straight. This condition is met when all adjacent surfaces to front or back plane's outer loop are plane.
- Outer loop of the front plane is not equal to the one of the back plane: not for each vertex of the front plane's outer loop there should be a vertex on the outer loop of the back plane having the same x, y coordinates.

Specific rule for rectangular

- Outer loop of the front and back plane must be rectangular, see Fig. 3.20.

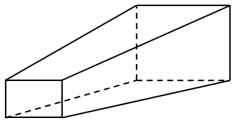


Fig. 3.20: Example of the rectangular long component with varying cross-section

Specific rule for right angle or triangular

- Outer loop of the front and back plane must be triangular, see Fig. 3.21.

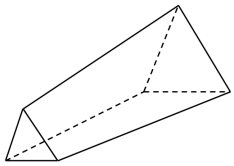


Fig. 3.21: Example of the triangular long component with varying cross-section

Specific rule for any varying cross-section (not rectangular and not triangular)

- Outer loop of the front and back plane must not be triangular and rectangular, see Fig. 3.22.

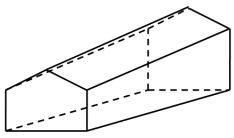


Fig. 3.22: Example of the long component with varying cross-section (not rectangular and not triangular)

3.2.1.3 Opitz Non-Rotational Cubic Components

Cubic forms are classified as in box-like component and block-like components, see Fig. 3.23.

General rules for extracting cubic components:

- Bottom surface should be found which is a plane (not a cylindrical or conical surface) and parallel to Y-plane.

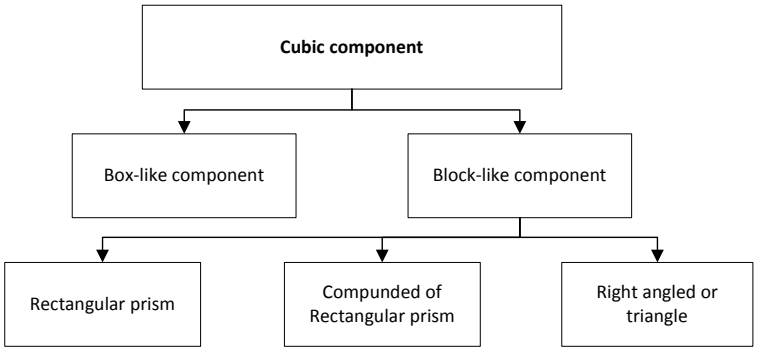


Fig. 3.23: Main features extracted from STEP for cubic components in Opitz

- All adjacent surfaces to an outer loop of the bottom plane must be orthogonal to this plane as well.
- Outer loop of the bottom plane should have more than 5 linear edges, with an angle of 90 degrees between any adjacent pair of edges as depicted in Fig. 3.25(b).

Specific rule for box-like compounded of rectangular prisms cubic component

- One inner loop within the bottom plane should exist, having the same shape as the outer loop, see Fig. 3.24.

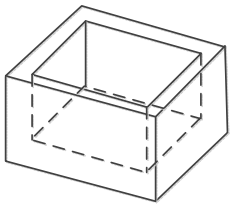


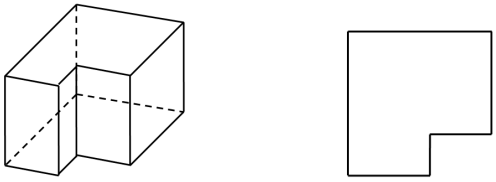
Fig. 3.24: Box-like cubic component compounded of rectangular prisms

Specific rule for compounded of rectangular prisms cubic component

- There is no specific rule for this feature. When applying the general rules for the cubic components, the feature is extracted, see Fig. 3.25.

Rectangular prism cubic component is identified using the same rules as for the Rectangular Flat component. Right angled or triangular cubic component can be

identified using the same criteria as for right angled or triangular flat components. Up to here, all components without additional features were considered. In the following sections, different forms of features combinations including number of principal bores are analyzed and discussed. These features are common features for flat, cubic and long component categories.



(a) Rectangular prism (b) Bottom plane

Fig. 3.25: (a) Cubic component compounded of rectangular prisms, (b) its bottom plane

3.2.1.4 Recognition of Principal Bore and Rotational Surface Machining

In this section, rules to identify bore-features for the (0, 1, 0) direction are presented, see Fig. 3.26. However, they are valid for other directions (1, 0, 0), (0, 0, 1) as well.

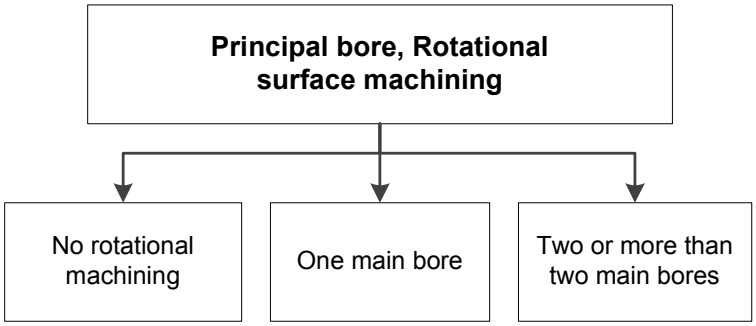


Fig. 3.26: Extension of principal bore

General rule:

- Bottom surface should be found which is a plane (not a cylindrical or conical surface) and parallel to Y-plane.

No rotational machining

- Bottom surface should have no inner loops.

One main bore

- Bottom surface should have one inner loop which is a circle.
- Adjacent surface (that is a cylinder) to this inner loop should be orthogonal to the surface of the current inner loop, see Fig. 3.27.

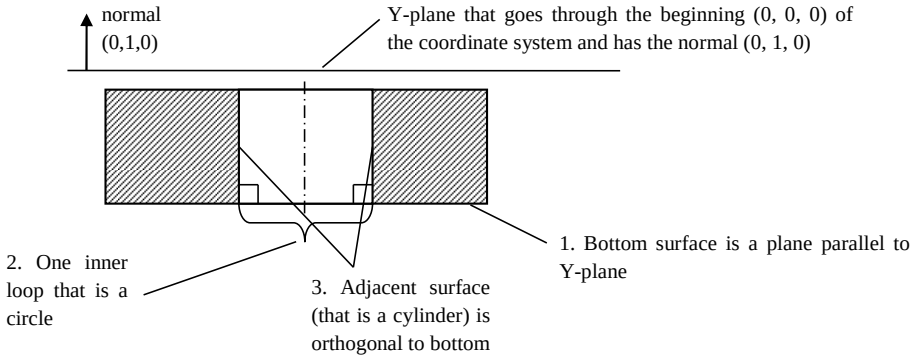


Fig. 3.27: Cross section by Z-plane through a part with one main bore

Two (or more than two) main bores

- Bottom surface should have two (or more) inner loops which are circle.
- Adjacent surface (which is a cylinder) to each inner loop should be orthogonal to the surface of current inner loop.

3.2.1.5 Recognition of Plane Surface Machining

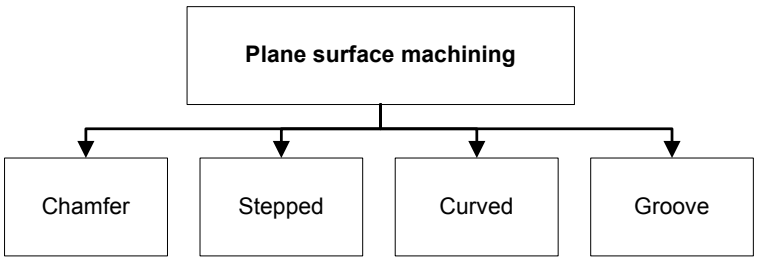


Fig. 3.28: Extensions of plane surface machining

Chamfers

For a given part, chamfers are exposed whether the following statements are met:

- Top surface should be found which is plane (not cylindrical or conical surface) and parallel to Y-plane, see Fig. 3.29.
- All adjacent surfaces to the top plane should have the same angle between the normal of current surface and the Y-oriented normal (0, 1, 0).

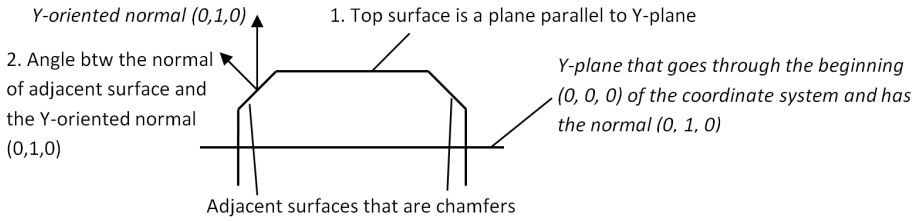


Fig. 3.29: Cross section by Z-plane through a part with chamfers

Stepped plane surface

- To check whether a part has a stepped plane surface machining, only one condition should be evaluated: the total amount of plane surfaces that are parallel to the Y-plane must be more than 2 and there must be no grooves for the current detail, see Fig. 3.30.

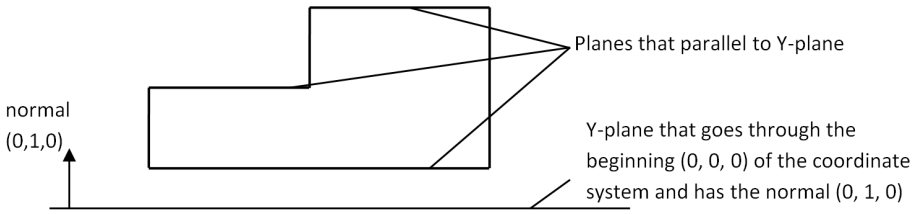


Fig. 3.30: Cross section by Z-plane through a part with stepped top machining

For non-rotational parts there are 3 Opitz code groups of plane surface machining: plane surface, stepped plane surface, stepped surface vertically inclined and/or opposed. These groups differ in the methods of machining, having the result of the same part shape. It means for feature recognition of these three groups the same rules are applied.

Curved surface

- The curved face machining is recognized if the following conditions are met. A cylindrical surface with normal vector that lies within Y-plane should be found (i.e. normal = (*,0,*)). Additionally, a bottom plane is identified (bottom plane

can be within non rotational parts only), when a positive result is concluded, see Fig. 3.31.

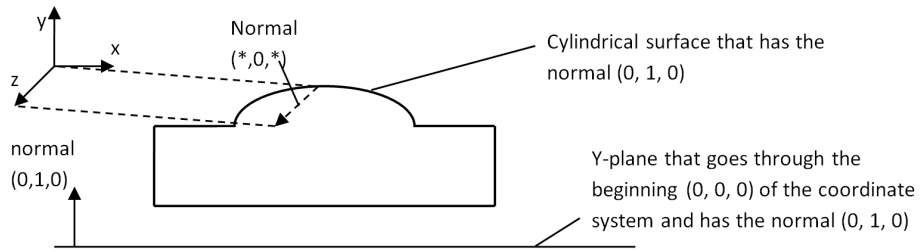


Fig. 3.31: Cross section by Z-plane through a part with the curved top machining

Groove and/or slot

For a given part groove and/or slot are exposed when the following conditions are met:

- Top surface should be found which is a plane (not a cylindrical or conical surface) and parallel to Y-plane, see Fig. 3.32.
- The found top surface should have one or more inner loops that are not circles.

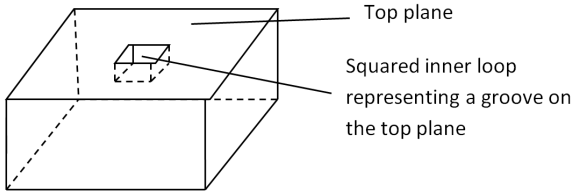


Fig. 3.32: Groove or slot identification

A surface with no plane surface machining is a shape that has the following characteristics:

- Top surface should be found which is a plane (not a cylindrical or conical surface) and parallel to Y-plane
- No chamfer is recognized for the found top surface
- Current shape has no stepped plane surfaces
- Current shape has no curved machining
- Has no grooves or slots

In the above sections, the main Opitz features and their dedicated rules and conditions for recognizing them from STEP file were presented and explained. In the following sections, the main form features of the Opitz rotational form features are presented and explained.

3.2.2 Opitz Feature Recognition of Rotational Components

Similar to the classification of non-rotational parts into cubic, flat and long components, the rotational parts are also divided into three categories. This categorization is based on the result of the deviation of the length to the diameter of a form, as seen in Fig. 3.33.

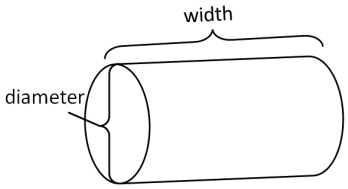


Fig. 3.33: Rotational part shape measures

According to the Opitz code specification:

a) $\frac{length}{diameter} \leq 0.5 \quad \Rightarrow \quad 1^{\text{st}}\ digit\ of\ Opitz\ code = 0$

b) $\frac{length}{diameter} < 3 \quad \Rightarrow \quad 1^{\text{st}}\ digit\ of\ Opitz\ code = 1$

c) $\frac{length}{diameter} \geq 3 \quad \Rightarrow \quad 1^{\text{st}}\ digit\ of\ Opitz\ code = 2$

In the following sections, all positioning rules to extract Opitz features from a STEP file are presented and explained [115].

3.2.2.1 External Shape and External Shape Elements

In this section the analyzed rules to identify cylinders are presented for the (0, 0, 1) direction. However, they are applicable for the other directions including (1, 0, 0), (0, 1, 0) as well.

The general rule for the rotational components is defined as following with only one step, see Fig. 3.34.

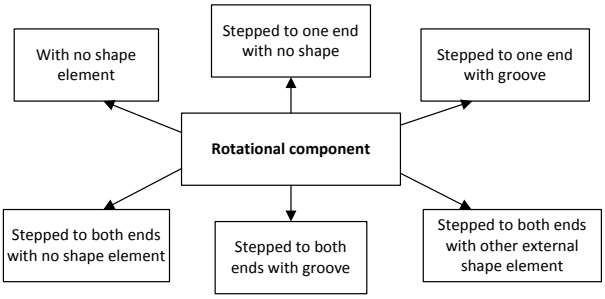


Fig. 3.34: The main features of rotational components

General rule

- Front and back surfaces should be found which are on a plane (not a cylindrical or conical surface) and parallel to the Z-plane having the maximal and minimal z-coordinates within the current shape respectively.

Rotational component with no shape elements

A rotational component with no shape element is identified after consideration of the following aspects as depicted in Fig. 3.35.

- Only one cylindrical surface should be identified which is orthogonal to the plane of the back surface.

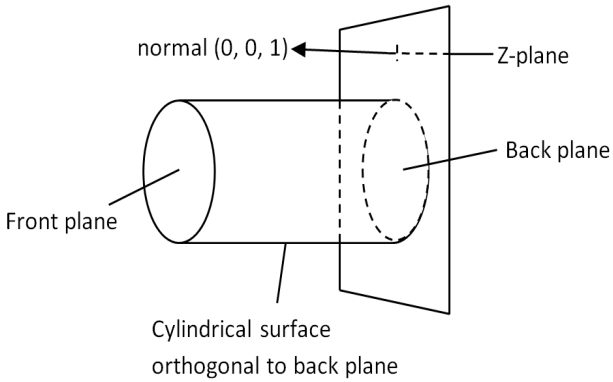


Fig. 3.35: Rotational part with no shape elements

Rotational component stepped to one end with no shape elements

- Two cylindrical surfaces should be identified that are orthogonal to the plane of the back surface, see Fig. 3.36.

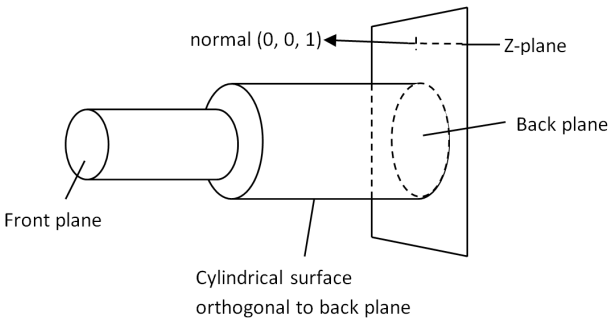


Fig. 3.36: Rotational part stepped to one end with no shape elements

Rotational component stepped to one end (or smooth) with a groove (slot)

- Two (or one for a smooth part) cylindrical surfaces identified that are orthogonal to the plane of the back surface, see Fig. 3.37.
- Grooves count = 1

(a) Rotational part

(b) stepped one end

Fig. 3.37: (a) Rotational part with a groove smooth, (b) stepped to one end

Rotational component stepped to both ends with no shape element

- Three cylindrical surfaces should be identified that are orthogonal to the plane of the back surface, see Fig. 3.38.

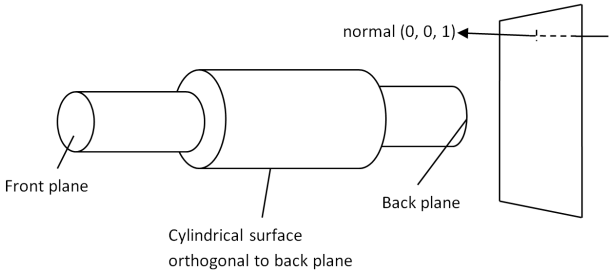


Fig. 3.38: Rotational part stepped to both ends with no shape element

Rotational component stepped to both ends with grooves (or slots)

- Three cylindrical surfaces should be identified that are orthogonal to the plane of the back surface, see Fig. 3.39.
- Number of the grooves should be equal to 1 or 2 (information about groove identification is in section 3.2.1.4.

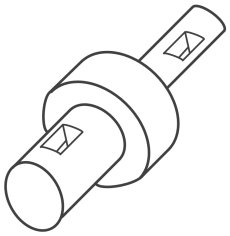


Fig. 3.39: Rotational part stepped to both ends with 2 grooves

Rotational component with other external shape elements

In this part, the rotational components with shape elements and more than 10 functional diameters are considered.

- More than 10 cylindrical surfaces are identified which are orthogonal to the plane of the back surface

Internal Shape and Internal Shape Elements

- At least one cylindrical surfaces is identified which is orthogonal to the plane of the back surface.
- There is no inner loops on the front face and on the back face.

Smooth or stepped to one end component with internal no shape element

- General rules of rotational component with no shape elements, see Fig. 3.40.
- One or two cylindrical surfaces are identified that are orthogonal to the plane of the back surface.
- There is 1 circled inner loop on the front face that equals to the one on the back face.

(a) Rotational smooth

(b) Stepped to one end

Fig. 3.40: Rotational smooth (a) and stepped to one end (b) part with internal no shape element

Stepped to both ends component with internal no shape element

- Three cylindrical surfaces are identified that are orthogonal to the plane of the back surface, see Fig. 3.41.
- There is one circled inner loop on the front face that equals to the one on the back face.

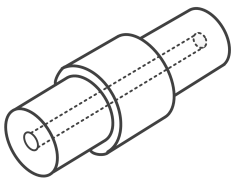


Fig. 3.41: Rotational stepped to both ends part with internal no shape element

3.3 Opitz Feature Extraction

The Opitz table of features has an ideal logic base to build a decision tree and the code. The features being used within this module were based on the Opitz coding system. The available features covered some of the first, second, third, and fourth digits. This module covers the analysis of allocation of a number to a digit in Opitz code (0-9). In order to give an overview of the Opitz code, the following tree diagrams were constructed to indicate how features are identified via a rule-based system. Fig. 3.42 presents the decision tree to

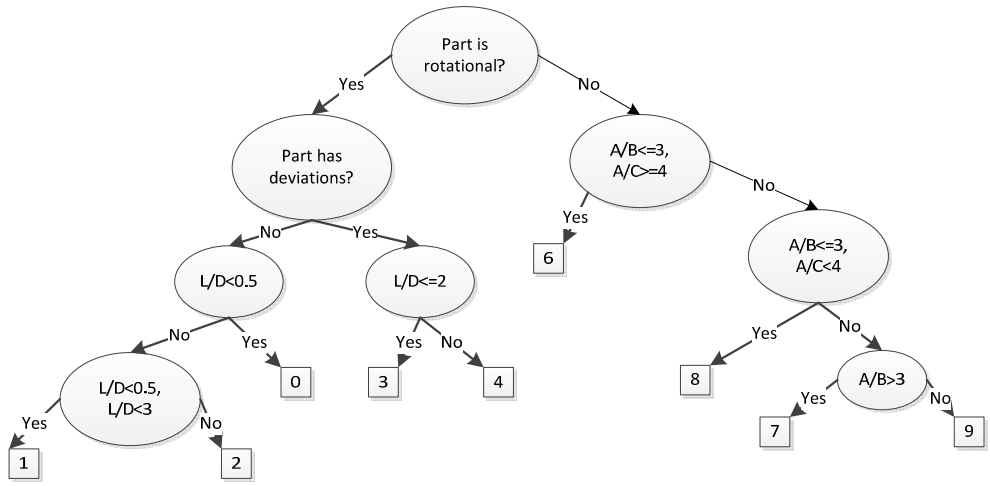


Fig. 3.42: Opitz Code Decision Tree for Digit 1

determine the first digit of the Opitz code. According to the Opitz specification [103], the "L" refers to the length of the part. The "D" refers to its diameter. The "A", "B", and "C" refer to the length, width, and height of the part respectively.

The second digit varies according to the value of the first digit of the Opitz code. This is due to its hierarchical nature. There are multiple subtrees for the second digit that depend on the value of the first digit. These subtrees are arranged accordingly [123].

- when the first digit is less than 3
- when the first digit is greater than 2 but less than 5
- when the first digit is equal to 6
- when the first digit is equal to 7
- and when the first digit is equal to 8

In the first case, the external shape and the external shape elements are described. In the other cases, the overall shape of the part is described.

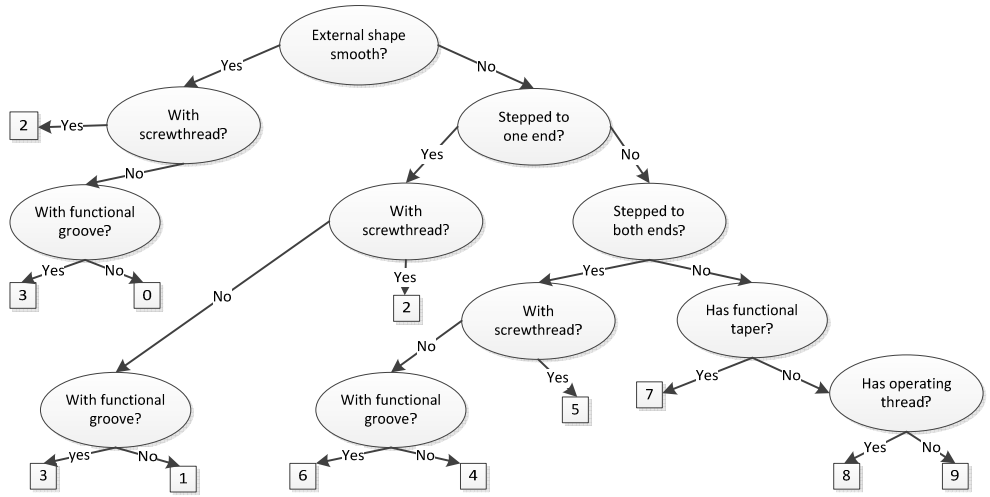


Fig. 3.43: Opitz Code Decision Tree for Digit 2 (1st digit $<$ 3)

For digit 2 (1st digit $<$ 3)

The above diagram, see Fig. 3.43, shows the decision tree to determine the second digit of the Opitz code when the first digit is less than 3. Of this tree, the screw thread, taper and operating thread inquiries needed to be designed, and later implemented. The rule evaluation takes place in a forward-chaining manner, first inquiring about the shape's smoothness, then based on the result, moves to the appropriate subtree, and so forth.

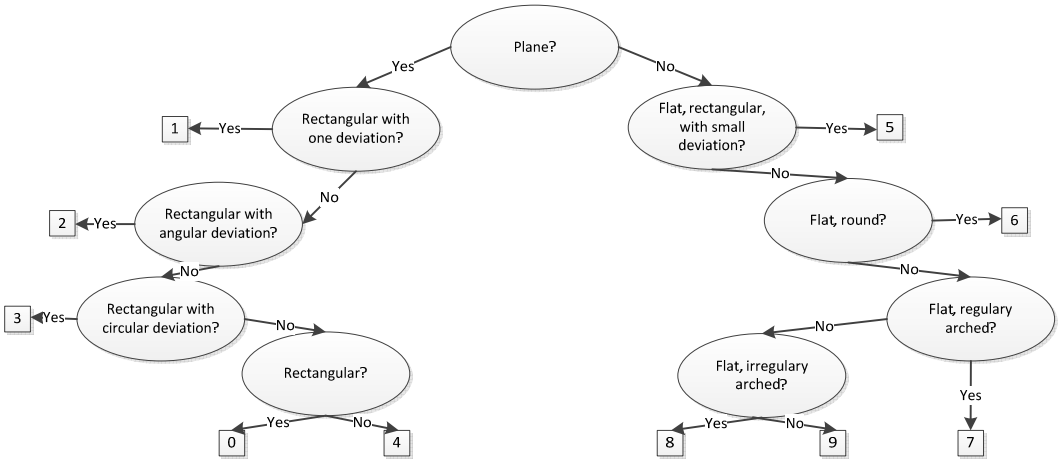


Fig. 3.44: Opitz Code Decision Tree for Digit 2 (1st digit = 6)

For digit 2 (1st digit = 6)

The above diagram, see Fig. 3.44, shows the decision tree to determine the second digit of the Opitz code when the first digit is 6 which means for flat components.

For digit 5 (2 < 1st digit < 5)

The diagram below, see Fig. 3.45 shows the decision tree to determine the second digit of the Opitz code when the first digit is greater than 2, but less than 5. This entire tree had to be designed and implemented.

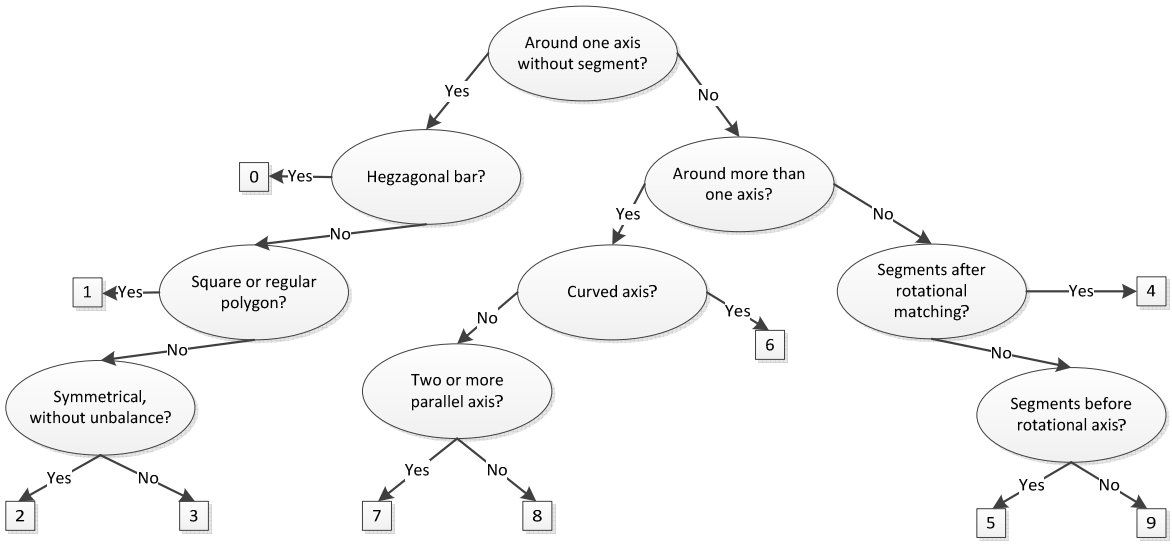


Fig. 3.45: Opitz Code Decision Tree for Digit 2 (2 < 1st digit < 5)

For digit 3 (1st digit < 3)

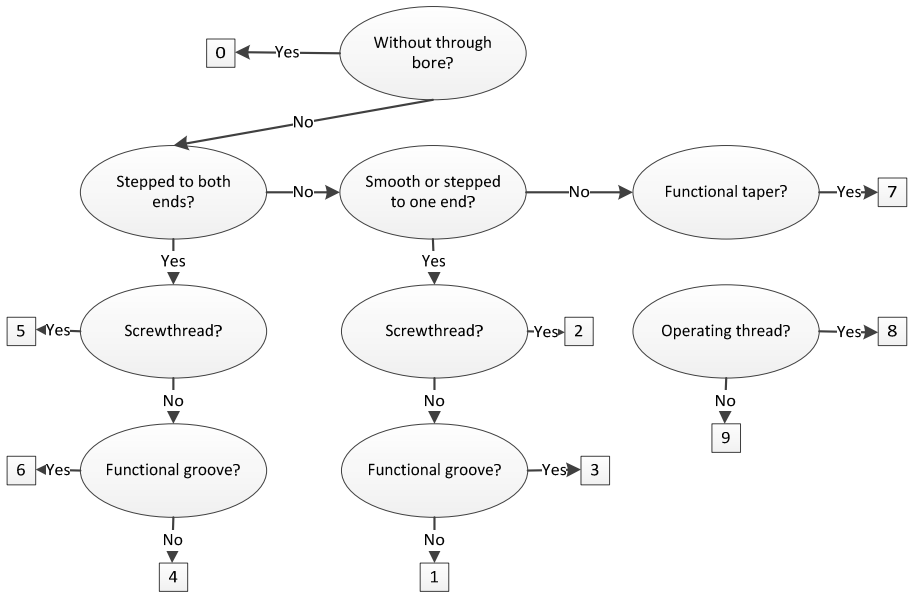


Fig. 3.46: Opitz Code Decision Tree for Digit 3 (1st digit < 3)

The above diagram, see Fig. 3.46 shows the decision tree to determine the third

digit of the Opitz code when the first digit is less than 3 (i.e. for rotational parts without deviations).

For digit 2 (1st digit = 8)

The following diagram, see Fig. 3.47 shows the decision tree to determine the second digit of the Opitz code when the first digit is 8 (i.e. for the cubic components). Just like the second digit, the third digit varies according to the value of the first digit. This is again due to the hierarchical nature of the Opitz code. It is, however, not as diverse as the second digit. The third digit describes the internal shape for rotational parts without deviations, the rotational machining for rotational parts with deviations, and the principal bore properties and rotational surface machining for non-rotational parts.

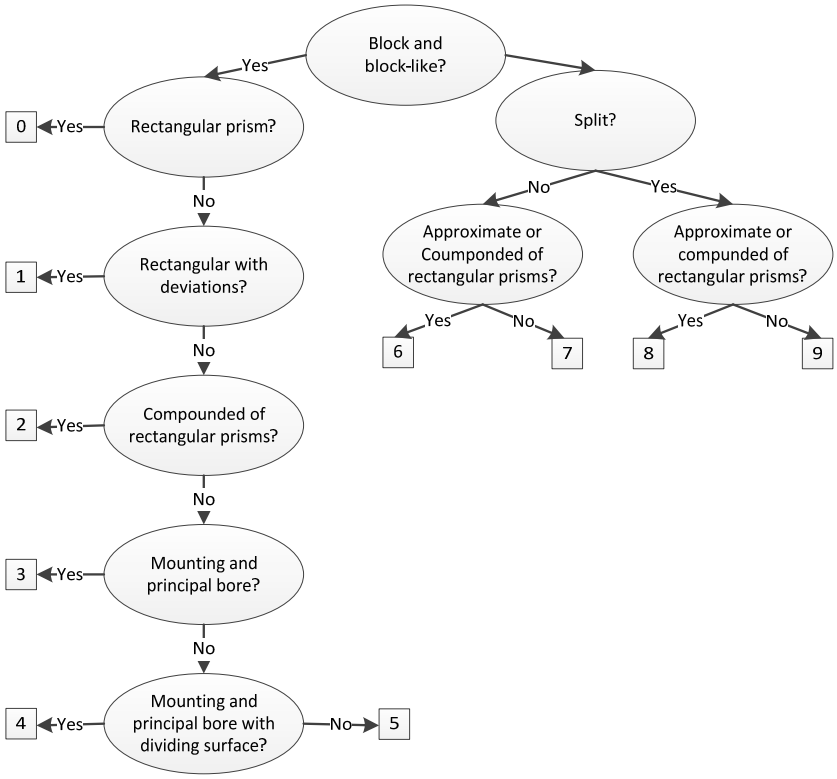


Fig. 3.47: Opitz Code Decision Tree for Digit 2 (1st digit = 8)

For digit 3 (2 < 1st digit< 5)

Fig. 3.48 shows the decision tree to determine the third digit of the Opitz code when the first digit is less than 5 but greater than 2 (i.e. for rotational parts with deviations). It is important to mention when there is a similarity between features; rules can be reused in different classification. As an example, screw thread detection is the same when applied externally as well as internally. Operational threads are the same as well.

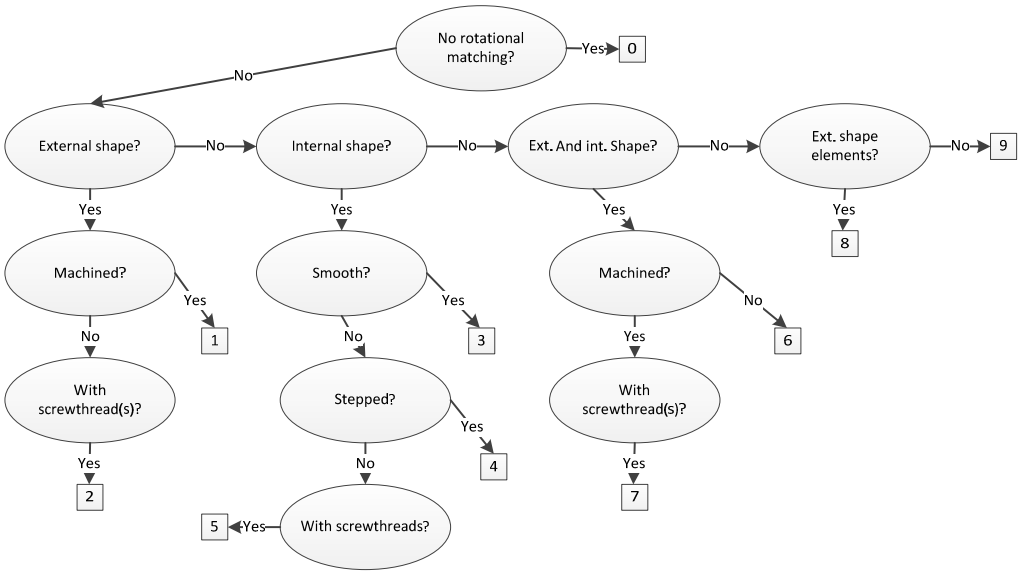


Fig. 3.48: Opitz Code Decision Tree for Digit 3 (2 < 1st digit < 5)

For digit 5 (1st digit< 3)

Fig. 3.49 depicts the decision tree for the fifth digit of the Opitz code. This case, however, is only when the first digit is less than 3 and it refers to the rotational parts without deviations.

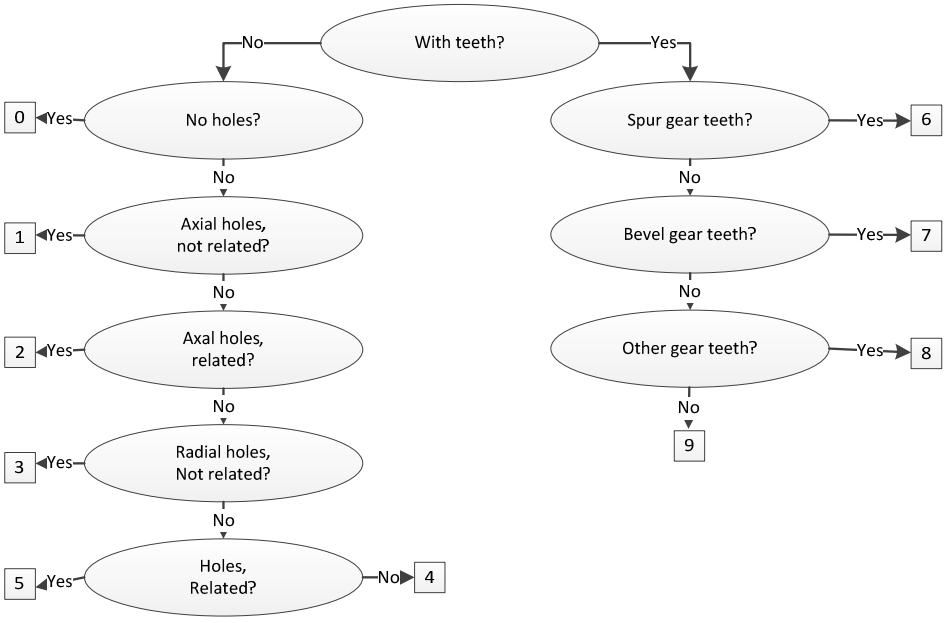


Fig. 3.49: Opitz Code Decision Tree for Digit 5 (1st digit < 3)

For digit 3 (5 < 1st digit< 9)

Fig. 3.50 shows the decision tree to determine the third digit of the Opitz code when the first digit is less than 9 but greater than 5 (i.e. for non-rotational parts).

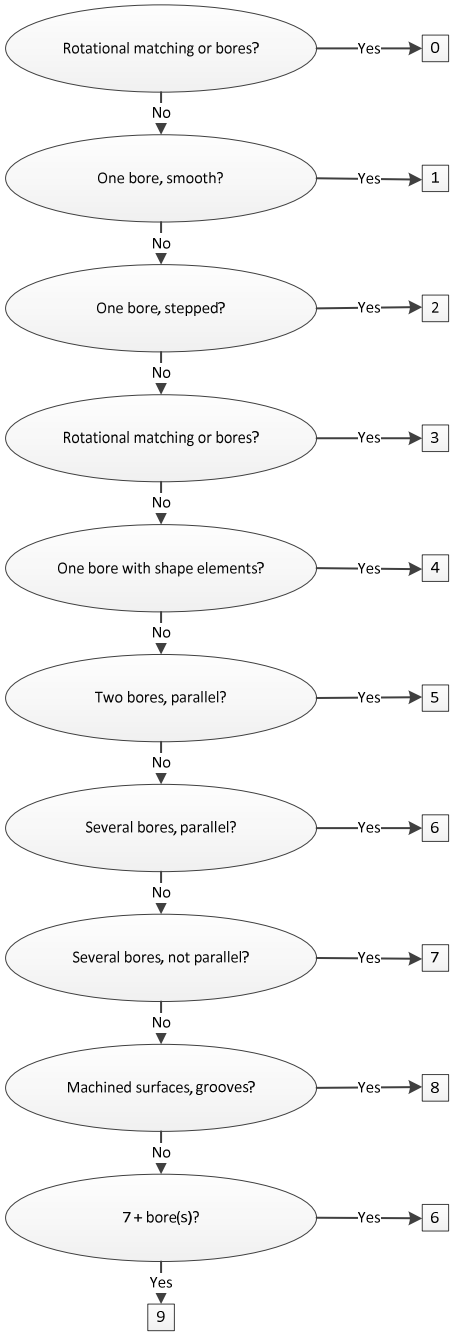


Fig. 3.50: Opitz Code Decision Tree for Digit 3 (5 < 1st digit < 9)

For digit 5 (2 < 1st digit < 5)

Fig. 3.51 depicts the Opitz decision tree for the fifth digit, when the first digit is less than 5, but greater than 2 (i.e. rotational parts with deviations).

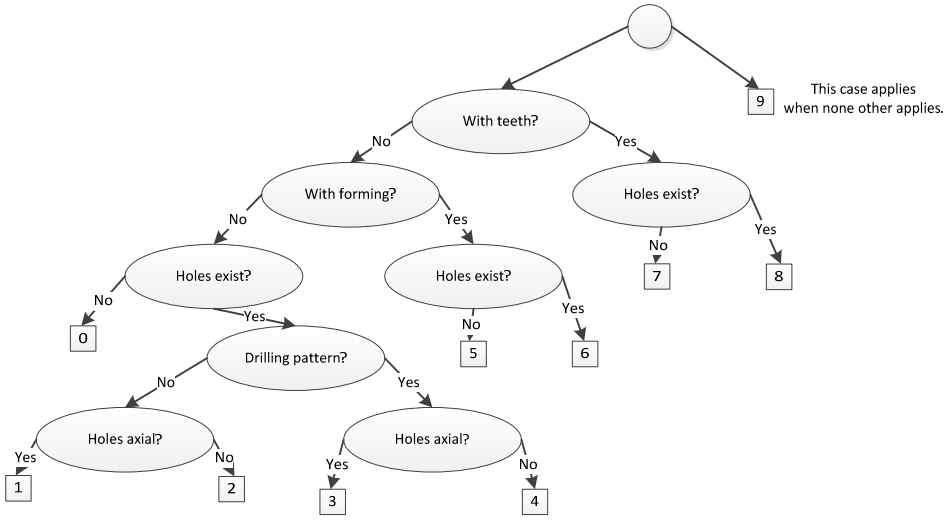


Fig. 3.51: Opitz Code Decision Tree for Digit 5 (2 < 1st digit < 5)

For digit 5 (5 < 1st digit < 9)

Fig. 3.52 depicts the Opitz decision tree for the fifth digit, when the first digit is less than 9, but greater than 5. This feature refers to non-rotational parts.

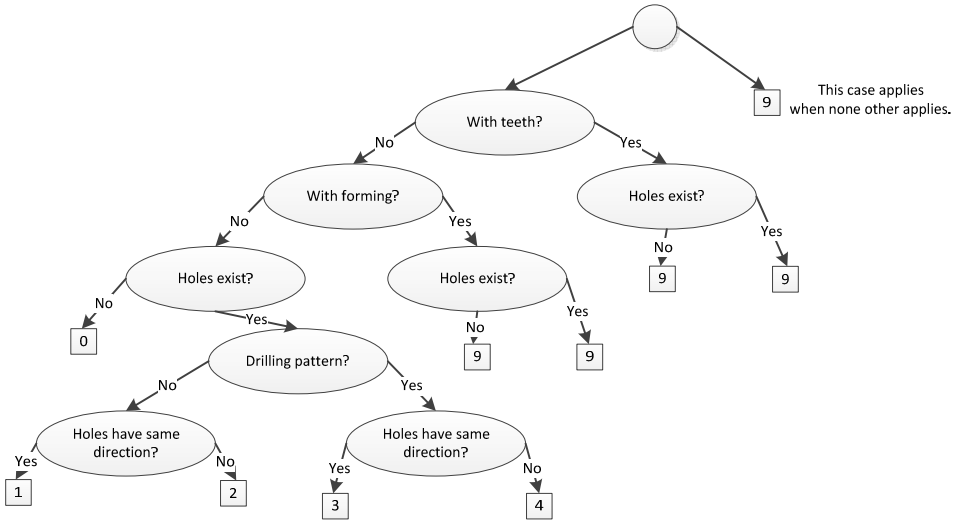


Fig. 3.52: Opitz Code Decision Tree for Digit 5 (5 < 1st digit < 9)

For digit 2 (1st digit = 7)

The below diagram, see Fig. 3.53 shows the decision tree to determine the second digit of the Opitz code when the first digit is 7 which means for long components.

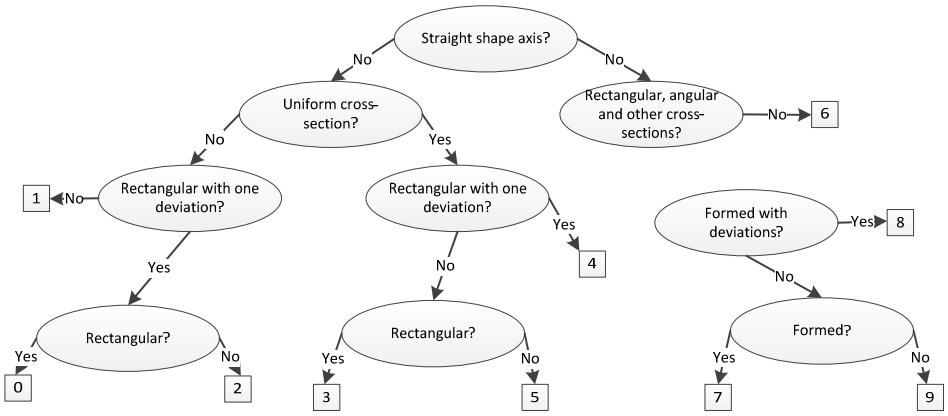


Fig. 3.53: Opitz Code Decision Tree for Digit 2 (1st digit = 7)

For digit 4

The next diagram, see Fig. 3.54, shows the decision tree to determine the forth digit of the Opitz code.

Opitz code tree for supplementary digits

The supplementary digits of the Opitz code (digits 6 through 9) is presented in Appendix A, see Fig. A.8. These digits provide possibilities to add manufacturing perspectives to the models. Extra digits are added to the model to include more characteristics of a form in the coding system. Such a capability empowers the coding system for forms with the higher complexity. Digits 7 and 8 provide information about the source material from which the part will be manufactured, as well as its initial form. The first six digits can be classified based on the input from the STEP file, however, the last three digits are different.

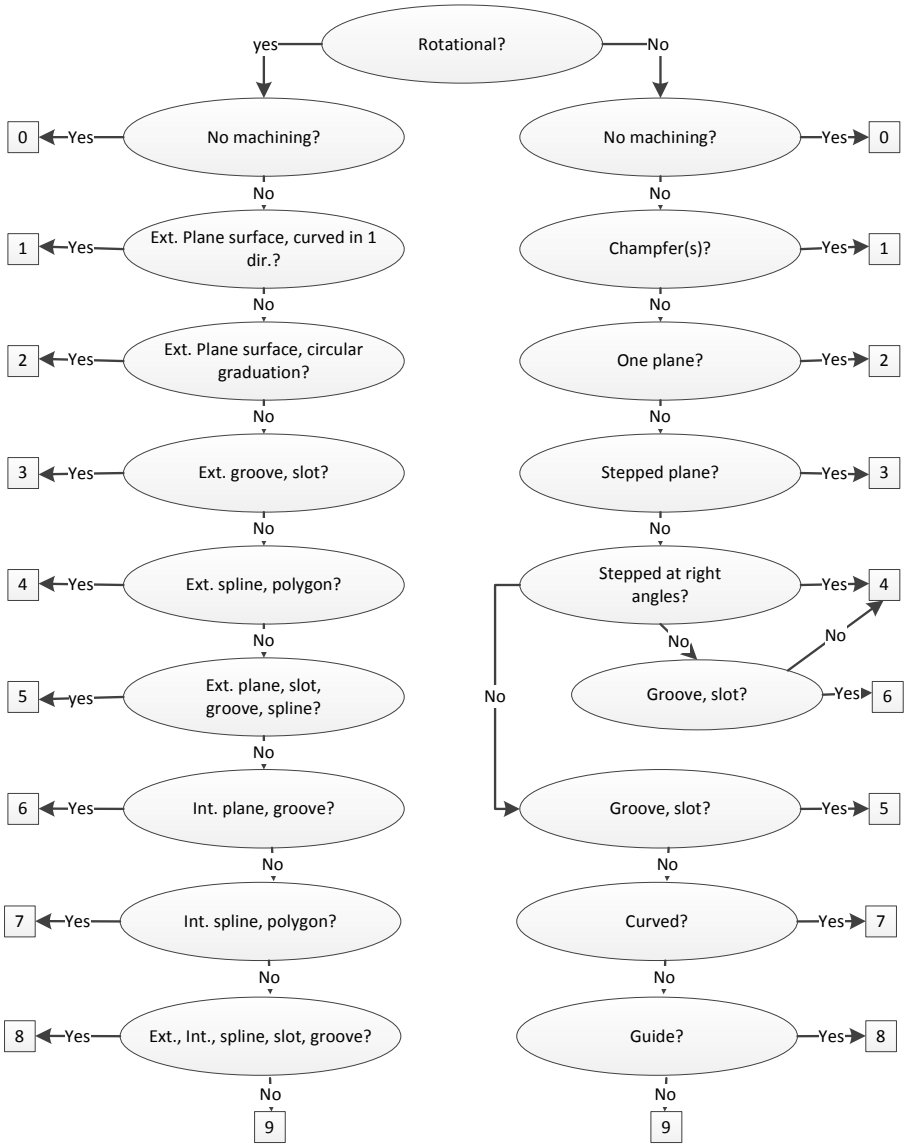


Fig. 3.54: Opitz Code Decision Tree for Digit 4

There is no information in the STEP file regarding material, or even accuracy for that matter. So, it was concluded that they should be entered directly by the user. As for the last digit, which pertains to accuracy, a novel approach was employed. The recognition process must be made flexible enough to accommodate the user, and sometimes, user errors. A standard level of flexibility will be set for each feature (or digit), all the while giving the user the option to override that level. If this level is overridden, the digit will be considered inaccurate. This "flexibility level" will be discussed in more details in the implementation sub-phase.

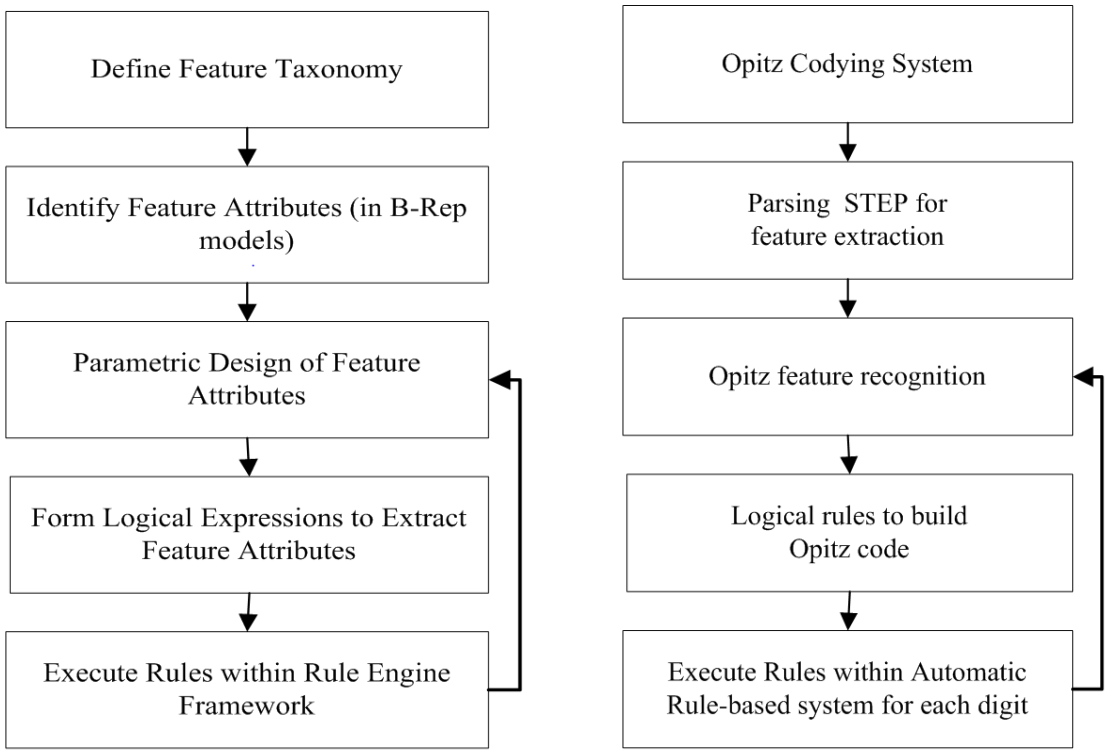
3.4 The Proposed Automatic Rule Based System for Feature Recognition

This is the second phase of the proposed methodology. All the mentioned rules including rotational and non-rotational are combined and structured to define a set of rules which are applicable for implementing the code as well. The rules are defined for STEP file format as input and tailored for construction of the Opitz features and consequently the Opitz code. The rules greatly take the advantage of hierarchic structure of the Opitz coding system. The applied rule based system in this dissertation, is based on the Van der Velden et al. [124] model to construct the automatic rule based system which automatically extracts engineering features, see Fig. 3.55. The input is neutral STEP models to be applied in downstream processes including, but not limited to, analysis (CAE systems) and manufacturing process planning (CAM systems). Four main steps have been defined for this method which have been implemented in the proposed approach as well.

The model consists of five steps, each step in Van der Velden is comparable with a step in the proposed method. In the following, the detail of this comparison is explained.

Step 1. Define Feature Taxonomy Which originally refers to the fundamental characteristics of a given feature type. In the other words, it defines a feature type for any geometric model. In the proposed method, there is a standard definition for features respect to the Opitz coding system. The Opitz codying system is an extensive part classification based on the manufacturing grouping.

Step 2. Identify Feature Attributes This step was basically defined to identify the minimum set of the required information from B-Rep to present a feature. In comparison, in the other method, the STEP file is parsed and the require information is extracted. Also based on the comprehensible definition of each feature in Opitz code,



(a) Van der Velden method (b) The proposed approach

Fig. 3.55: Comparison of Van der Velden method with the proposed approach

it is important to define the quality of extraction of this rules and its circumstances has to be defined.

Step 3. Parametric Design of Feature Attributes This phase identifies parameters of feature attributes that are uniquely separated from other entities in the model data. Consequently the attributes are defined such as face surface type, edge attribute (concave/convex/tangent), angle between faces, distance between entities, face area, etc.

In the proposed method rules were defined to extract the feature attributes. For instance, Panel Face attributes are identified using the following criteria: "a planar face surrounded on all sides of the outer boundary by faces that are concave and normal to the face surface", i.e.: 1) surfaceType = plane; 2) edgeAttribute = concave for all adjacentFaces on main wire; 3) faceAngle = 90 degrees (± tolerance) for all AdjacentFaces on main wire; 4) faceArea > minPanelArea.

Step 4. Form Logical Expressions to Extract Feature Attributes In this step the translation of the parametrized rules into the forms which can be interpreted by the interface engine is completed. To accomplish this task, the logical set of expressions presented in the previous step is applied.

Similarly, in the proposed method, the rules are presented and transformed into the logical expression. For instance the following rule based on [124]:

Step 5. Execute Rules within Rule Engine Framework The outputs of the inference engine is the list of entities that satisfy criteria contained within the rule for a particular feature attribute which are list of unique identifiers of the model entities.

```
1  get list of all faces with surfaceType = plane (List1);
2    for each face in List1
3      if faceArea > minPanelFaceArea then
4          get list of all adjacent faces on outer wire (
               List2);
5          get list of all adjacent faces on outer wire
               that are normal and concave (List3);
6          if lengths of (List2) and (List3) are NOT equal
               then
7            remove current face from List1;
8          end if
9      end if
10   end for
11 return (List1);
```

Listing 3.1: The proposed method for transforming into logical expression

Considering all conditions explained in the mentioned steps, the following sets of rules were realized as seen in Fig. 3.56, Fig. 3.57 and Fig. 3.58, published by Zehtaban and Roller [125].

The rule based system is performed several times in order to complete all the digits of Opitz code correctly.

3.5 Hierarchical Representation of the Developed Rules

Fig. 3.59 and Fig. 3.60 present a comprehensive design of rules for extracting Opitz features. Fig. 3.59 presents the rule set R2 which pertain to the operations with plane surfaces parallel to Y-plane. Fig. 3.60 displays the rule set R3 as well as the sequences to reach and identify are the operations with plane surfaces parallel to Y-plane.

R1. Part dimension measures extraction (length, width, height)

R1.1 Flat non-rotational part identified $\left(\frac{length}{width} \leq 3\right)$ and $\left(\frac{length}{height} \geq 4\right)$
R1.2 Cubic non-rotational part identified $\left(\frac{length}{width} \leq 3\right)$ and $\left(\frac{length}{height} < 4\right)$
R1.3 Long non-rotational part identified $\left(\frac{length}{width} > 3\right)$
R1.4 Rotational part (1^{st} digit of Opitz code = 0): $\left(\frac{length}{diameter} \leq 0.5\right)$
R1.5 Rotational part (1^{st} digit of Opitz code = 1): $\left(0.5 < \frac{length}{diameter} < 3\right)$
R1.6 Rotational part (1^{st} digit of Opitz code = 2): $\left(\frac{length}{diameter} \geq 3\right)$

Fig. 3.56: R1- The main sets of rules

R2. Operations with plane surfaces parallel to Y-plane

R2 rule set can be presented as a tree depicted in Fig. 3.59. All rules within current tree are hierarchy dependent, starting from the root node going to its child nodes. For example, to identify a ring machining on some non-rotational part, a chain of rules should be satisfied. $R_2 \rightarrow R_{2.1} \rightarrow R_{2.1.2} \rightarrow R_{2.1.2.3} \rightarrow R_{2.1.2.3.1}$ within given tree, see Fig. 3.59.

Fig. 3.57: R2- The main sets of rules

R3. Operations with plane surfaces parallel to Z-plane

R3 rule set can also be presented as a tree illustrated in Fig. 3.60. All rules within this tree are hierarchy dependent, starting from the root node going to its child nodes. For instance, to identify a long non-rotational part that has a rectangle as a cross section should satisfy the following chain of rules $R_3 \rightarrow R_{3.1} \rightarrow R_{3.1.1} \rightarrow R_{3.1.1.1}$ within given tree, see Fig. 3.60.

Fig. 3.58: R3- The main sets of rules

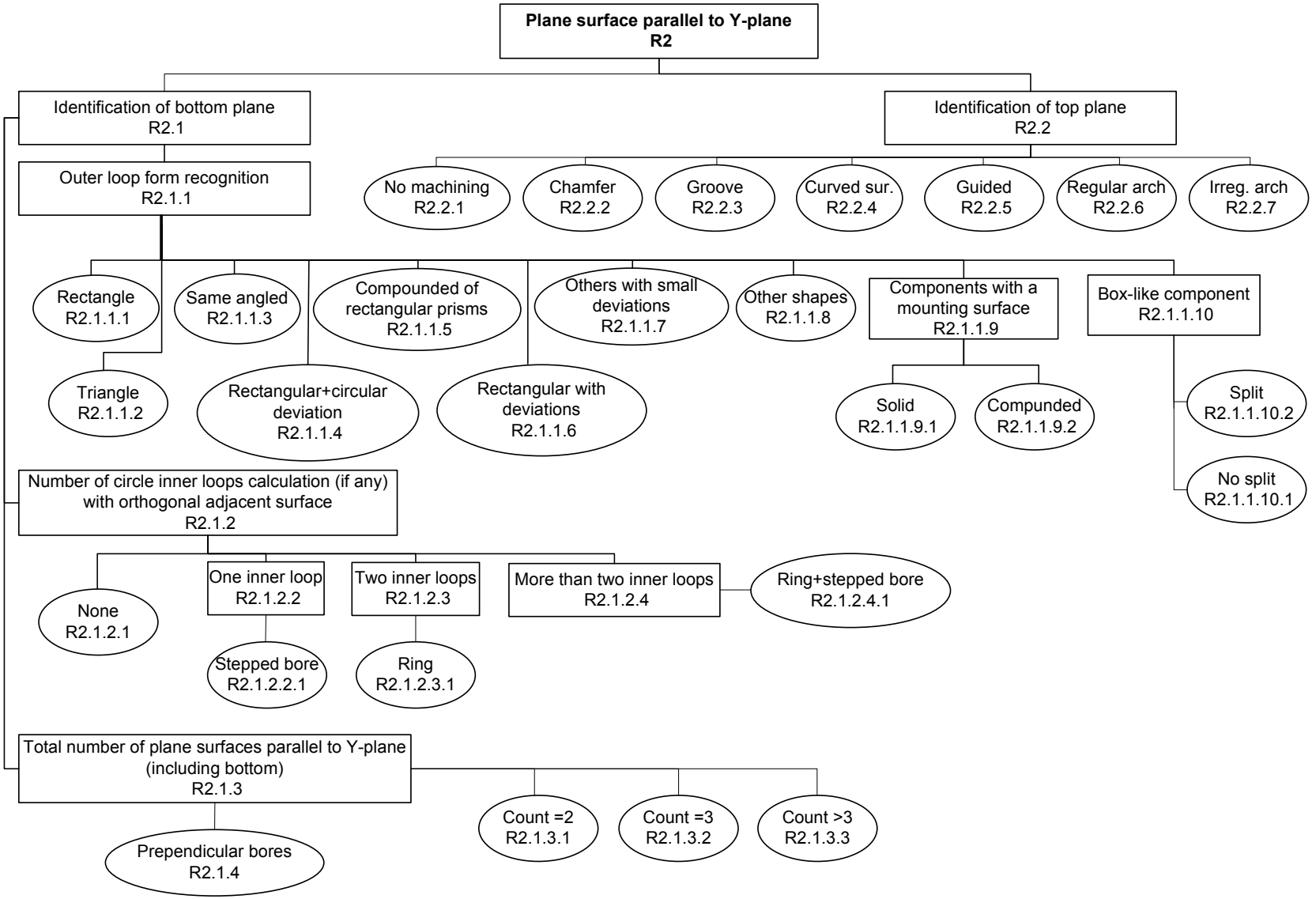


Fig. 3.59: Rules of feature recognition for surfaces parallel to Y-plane

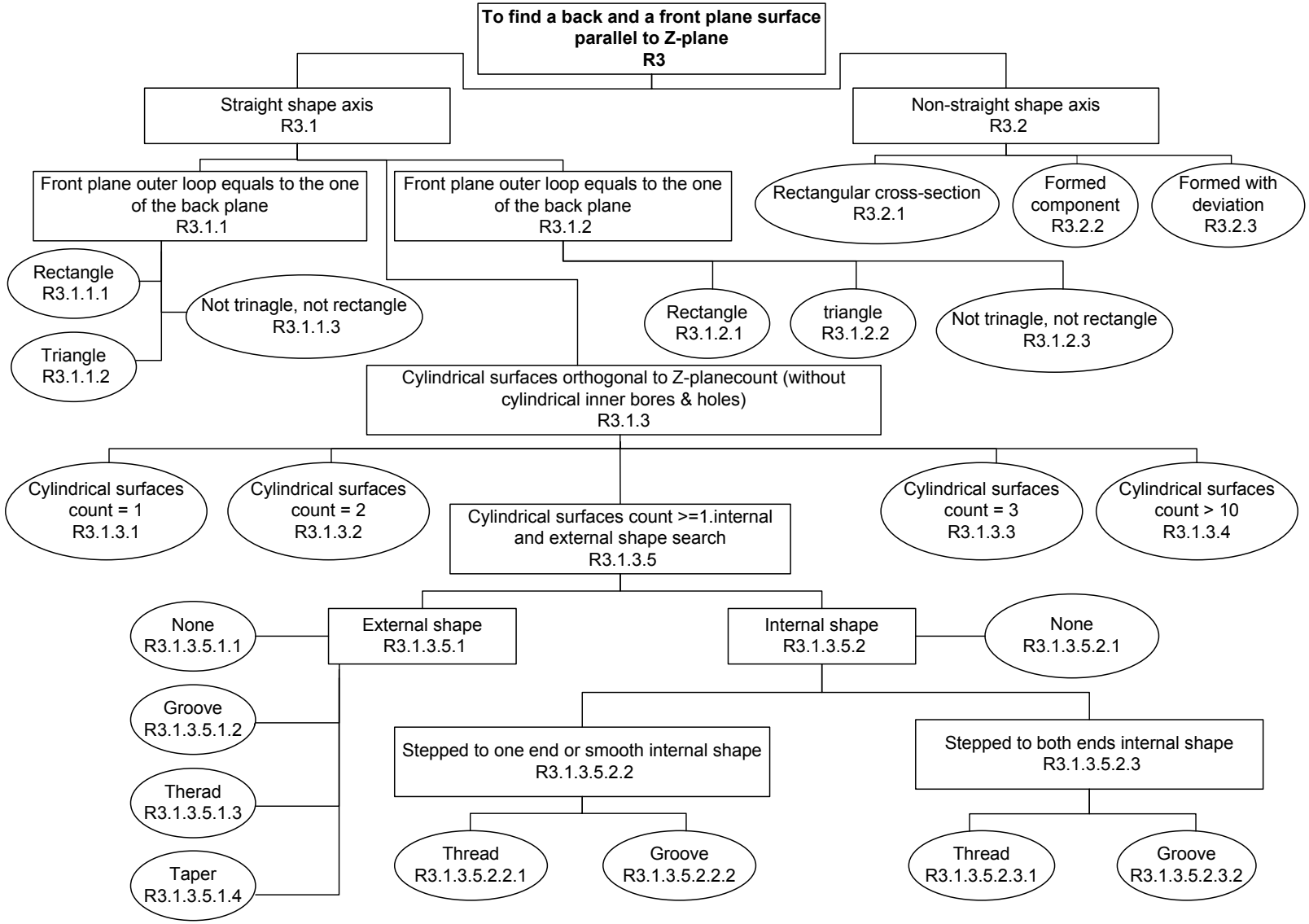


Fig. 3.60: Rules of feature recognition for surfaces parallel to Z-plane

Chapter 4

Similarity Retrieval and Distance Function

4.1 The Proposed Similarity Retrieval Method

One of the advantages of using the Opitz coding system is its numerical nature. Thus, it is possible to apply one of the distance functions in order to calculate the similarity of code vectors. This implicitly means the generated Opitz code would be used as a partial signature. The cosine similarity is used mainly due to the fact that it always provides a number between -1 and 1. Also, it can be adjusted to give a number between 0 and 1 indicating a degree of similarity. This is because each number and its additive inverse have the same magnitude, but move in opposite directions. In this context, measuring the direction of the code vectors is not important, but their magnitude is. Therefore, the negative solution space of the cosine similarity function can be omitted. By slightly modifying the output of the function, it can yield values between 0 and 100% indicating a level of similarity between two code vectors. This makes it more user-friendly. Another benefit from using the cosine similarity is that a weight vector may be used in order to emphasize or stress certain digits of the code vector, indicating their importance. This "weight vector" would be multiplied by each code vector, and then they would be submitted to the cosine similarity function, which will then output a similarity value in accordance with the applied weights.

However, it is not possible to apply the distance function directly to any code vectors. Due to the hierarchical structure of some sections of the Opitz code (form features), comparison rules have to be defined. To clarify, an Opitz code with a first digit of 1 cannot be compared with a code that has 8 for a first digit. The reason behind is by

interpreting of the Opitz code, the ending four digits of two codes do not necessarily refer to the same features. For example, in the previously described case, when the first digit is 1, the third digit describes the internal shape, while when the first digit is 8, the third digit describes principal bores and rotational surface machining. Hence, the following ranges have been defined:

- first_digit ≤ 3
- (first_digit ≤ 5) & (first_digit > 3)
- (first_digit ≤ 9) & (first_digit > 5)

Only if the first digit of a code falls within one of the previously described ranges, the rest of digits are comparable.

4.2 Active Database Design for Similarity Recognition

The "flowing behavior" of the paired similarity comparison and the "responding environment" are two major issues to be considered in the design of a database for a similarity searching application. This concept has been entirely discussed and published by Zehtaban and Roller [126].

Flowing behavior

Paired similarity searching in a database has a flowing behavior. It means, in a database of similar models, the criteria for similarity detection gradually changes after a number of similarity comparisons occurs.

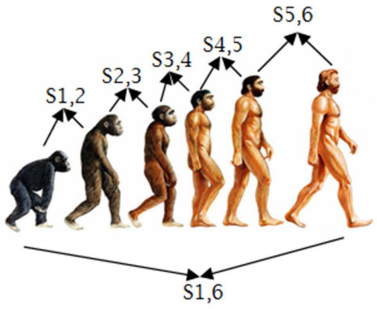


Fig. 4.1: Flowing behavior of paired similarity comparison: every side pairs have a high similarity although the first and the last images are not very similar

For example, S1,2 in Fig. 4.1 presents the percentage of similarity between image 1 and 2, and S2,3 indicates the percentage of similarity between image 2 and 3 and so on. It is noticeably observed that S1,2 and S2,3 indicate roughly the same percentage however, S1,6 refers to very different percentage and to a far less percentage number. In the other words, if the pair similarity comparison is continued in a domain including variant objects, after a number of objects, there might be just a trivial similarity between the first object and the last object. A continuous similarity comparison may cause isolation from the correct direction of searching.

Responding environment

Every new successful design has to be saved in the database as well as being immediately available for the subsequent similarity searching and retrievals. Thus the searching algorithm of the database should be flexible and classified to adopt the new member in the right category in the database.

4.3 Proposed Methods for a Quantitative Retrieval

With applying Opitz code as the shape signature, similarity comparison between two 3D shapes is concluded to one-by-one digits comparison of two Opitz codes. Since each digit is dedicated to a specific feature of the solid model, it is possible to prioritize some features in the similarity comparison process. Therefore, two possibilities are considered for the similarity retrieval in the database. First possibility is the similarity retrieval with an equal weight or priority for each digit and the second possibility is dedicated to the similarity retrieval with prioritized digits. These two methods are explained in detail in the next sections.

4.3.1 First Possibility: Similarity Retrieval with an Equivalent Priority for All Digits

When all digits in Opitz code have the same priority, there is a possibility that the similarity criteria changes gradually, in reference to the “flowing behavior” of the similarity searching, as discussed before. To prevent this challenge, in our proposed method, the database is divided into clusters in which each cluster has a header, see Fig. 4.2. Based on some characteristics, each entity (the information package of an object) belonging to the same cluster have similar attributes. The header is selected as the most referred-to entity in the previous searches in its cluster. Each header is a representative of its cluster. In every similarity searching each entity gets a score if it has been referred to and the one

which has a higher score will be placed as the header of its cluster. In the next search, if another entity holds a higher score, it will be replaced with the current header [127]. To give the retrieval an improved focus, it is possible to perform the quantitative similarity searching; it means the user can choose a similarity percentage for the retrieved results. Since a five digit Opitz code is considered here, the user can choose similarity percentage such as 20%, 40%, 60%, 80% and 100% for the retrieval. In general, the value of each digit for similarity comparison is calculated by Eqn. 4.1.

$$Value\ of\ each\ digit = \left(\frac{100}{number\ \ of\ digits} \times 100\%\right) \tag{4.1}$$

The total number of the similarity between two Opitz codes, considering the order and the number of the similar digits, is calculated by Sim(n,i) when n is the number of total digits in the code and i is the number of similar digits, Eqn. 4.2. In other words, two similar Opitz codes have one of the similarity models among all the possible models calculated by Eqn. 4.2.

$$Sim_{(n,i)} = \sum_{i=1}^{n} \binom{n}{i} \qquad Where\ \ i \leq n \tag{4.2}$$

The searching process in this approach; i.e. the similarity retrieval with an equivalent priority for all digits, is performed in two stages to achieve the possible best result for the similarity searching process, see Fig. 4.2. These two stages are horizontal searching and vertical searching.

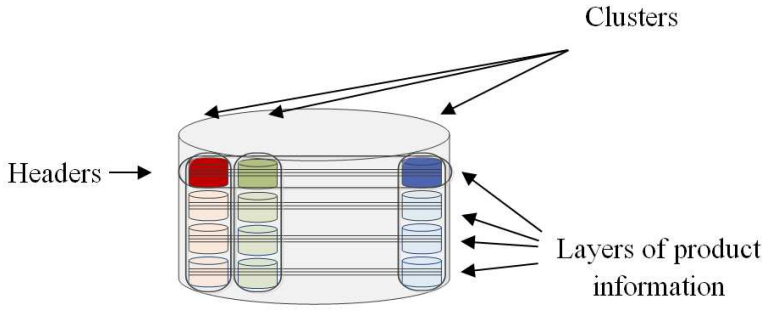


Fig. 4.2: Classification and layers of data in the database

Horizontal Searching This is an individual search in the headers. If a header presents an acceptable result of similarity, the header will be selected to continue for the

vertical search. Otherwise the current header will be ignored and the next header will be examined.

Vertical Searching Contains a search in the clusters. The second phase of similarity comparison will be performed in the clusters of the designated headers.

The workflow of the complete search, including the two mentioned search stages, is presented in Fig. 4.3. It includes 6 steps to retrieve five similar models illustrated in the following diagram in Fig. 4.3.

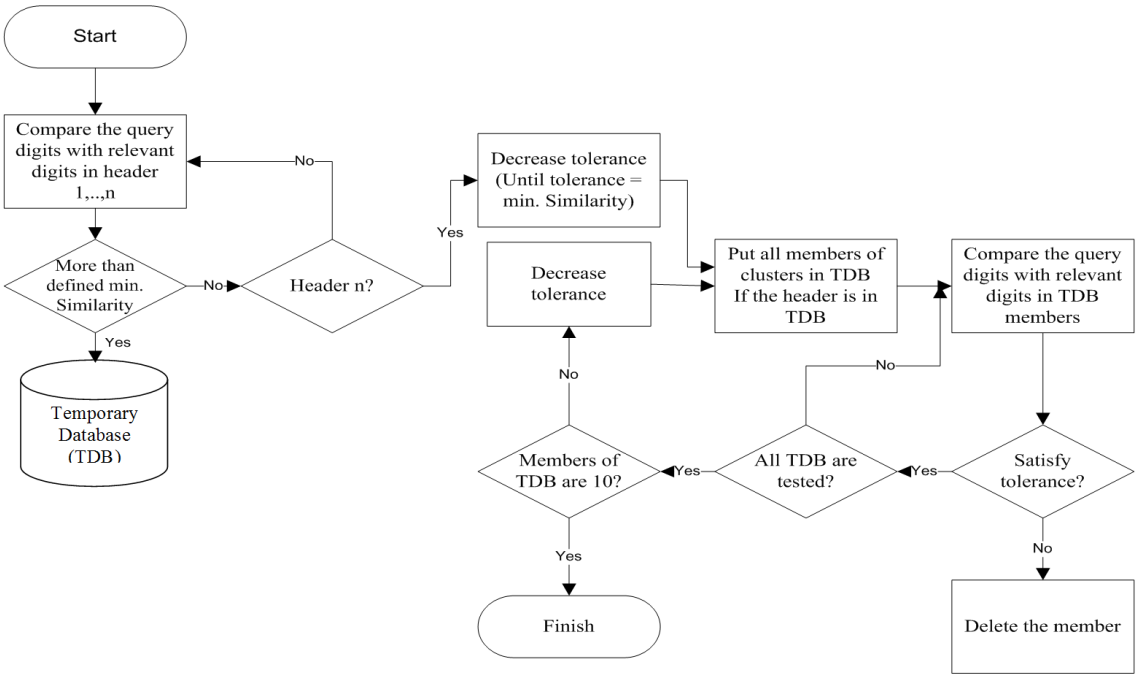


Fig. 4.3: Workflow of similarity algorithm [127]

Step 1: Comparing the query code with the code in header 1.

Step 2: If similarity comparison between the two codes is less than the minimum similarity defined by user, then the current header is ignored and the next header will be compared. Otherwise the current header is saved in the temporary database.

Step 3: After all headers are checked, the work on the temporary database is started. A tolerance domain (minimum similarity<tolerance<100) will be established to find the most similar models. The tolerance number is an additional value to the requested similarity percentage.

Step 4: The process is finished with the decrease of the tolerance, until only a few designs are left in the temporary database. In presented workflow in Fig. 4.3, 10 designs were desired and set, however the number of the retrieved designs can be defined by user.

4.3.2 Second Possibility: Similarity Retrieval with Prioritized Digits

For such a condition an interactive communication with the database has been considered. The user determines and chooses the digits with priority. In this model, only the Opitz codes containing an identical value for the prioritized digits are retrieved and any other similarity is ignored. After the user sets the priorities, in an active system, the most similar designs are located in the headers and are retrieved. This method benefits from a flexible attitude in the "responding environment" of similarity search. As mentioned earlier, two similar Opitz codes with equivalent priorities for digits may have one of the singular forms of similarity, calculated by Eqn. 4.1. This equation is extended to Eqn. 4.3, when the digits have priority. In this equation, n is the number of total digits in the Opitz code and x is the number of prioritized digits.

$$Sim_{(n-x,i)} = \sum_{i=1}^{n-x} \binom{n-x}{i} \qquad Where \;\; i \leq n-x \tag{4.3}$$

The number of possibilities to prioritize the digits, without considering the order of the digits, i.e. $P_{(n,i)}$, is calculated according to the Eqn. 4.4, when n is the number of the Opitz code and i is the number of prioritized digits. If the order of the digits is considered, the number of possibilities to prioritize the digits is calculated by the Eqn. 4.2 where n is the number of Opitz code and i is the number of the prioritized digits. Clearly, the number of the retrieved models or the quality of the results depends on the number of the models in the database.

$$Sim_{(n,i)} = \sum_{i=1}^{n} i \tag{4.4}$$

4.4 Cosine Coefficient Accuracy

This section is mainly concerned with the effectiveness of the proposed similarity evaluation method and whether it yields accurate, comprehensive, and satisfying results. The measure used to evaluate such a function is the F-Score (Eqn. 4.5) according to [128]. The F-Score is the harmonic mean of the precision and recall, given in Eqn. 4.6 and Eqn. 4.7 respectively. Precision measures the amount of the correctly retrieved results out of all the retrieved results, while recall measures the amount of correctly retrieved results out of all relevant cases to this query, Eqn. 4.5.

$$F - Score = \frac{2 \times precision \times recall}{precision + recall} \tag{4.5}$$

$$Precision = \frac{|r|}{|R|} \tag{4.6}$$

$$Recall = \frac{|r|}{|A|} \tag{4.7}$$

The “r” variable in the previous equations denotes the set of correctly retrieved documents. The “R” variable indicates the set of all retrieved documents, while the “A” variable indicates the set of all relevant documents. The F-Score typically results in a value between 0 and 1, 0 being the worst precision and recall, and 1 being the best.

Chapter 5

Prototype Development and Implementation

5.1 Pre-implementation

This section addresses analyzing of the possible scenarios in pre-implementation phase of the prototype. In this regard, different phases including requirement phase, specification phase and the implementation phase are discussed and presented. Between all the available models of software development, the incremental build model was chosen. The main reasons behind was flexibility and simplicity. The incremental model is iterative in its builds, which offers possibility for improvements with each build. This is not offered by the waterfall model. It also does not focus so strongly on risk analysis, like the spiral model. And since the requirements are clearly defined from the beginning and the result is a completely functioning program, the rapid prototyping model becomes risky to use as the prototyping itself may take up the entire project's duration. Based on the model proposed by Schach [129], the requirements are gathered during the requirements phase and then verified. This phase is important as once this phase is finished, it is no longer possible to be changed or extended. Subsequently the specification phase is defined, where the requirements are further examined and elaborated into a full software specification. For the same reason, the software specification requires verification. In the architectural design phase, a very high-level design of the product is established based on the specification formulated before. Since this design is linked more to the specification, and less to the actual design processes, it is treated the same (non-iterative) and requires verification. The build-activities phase comes next, where each set of specifications are designed in detail, implemented, integrated, and tested. This phase repeats for every build that incorporates a new set of specifications. Finally, the product enters the maintenance phase,

and lives there until retirement. In this regard, the next sections, discuss the requirements, specification and architectural design phases, and the reasons behind several of the design decisions that were encountered [123].

5.1.1 The Requirements Phase

In this phase, the requirements were gathered using the available techniques and sources. The available sources in this case included personal experiences of working in the electro-motor design department of Robert Bosch GmbH [130], few online industrial sources, and a large amount of literature regarding this domain. The primary technique used to gather requirements was interviews, although, in the cases pertaining to the industrial sources, interview capacity was limited. The interviews were then reduced to an online interaction using a forum as an interaction medium. Nonetheless, the following rough requirements were gathered:

- Models should be recognized automatically based on their features.
- It should be possible to compare the recognized models to the existing models registered within a model-base.
- The underlying mechanics should be represented using a user-friendly interface.
- The effectiveness and efficiency of this approach should be evaluated and validated.
- It could be interesting to have an alternative, easier to use method to recognize similar models, while providing users with various options that will aid them in specifying their search query parameters.

The mentioned requirements were translated into the formal software specifications within the second phase, the detailed design specification.

5.1.2 The Detailed Design Specifications

The previously defined requirements are broken down and re-formulated into a software specification in this phase. The first, second, and the last points are the most relevant to building the project's core functionality. Thus, they can be represented within this use-case diagram in Fig. 5.1. The use-case diagram depicts the target user for this application, namely a product designer who uses CAD software. The target user is assumed to have little or no previous domain experience with product design. The user should be able to

browse through the repository of models that already exist. The user should also be able to input a certain model, and the application should analyze it and present the user with the model's recognized features. Finally, the user should be able to compare a model, or a set of features, to the existing models within the repository and retrieve the most similar models.

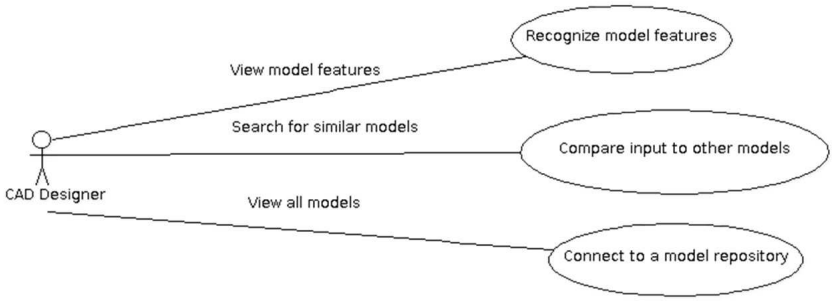


Fig. 5.1: Core Functionality Use Case Diagram

Based on the previous use-cases, three main scenarios can be identified:

- the CAD designer attempts to view an input model's features,
- the CAD designer attempts to compare an input model or a selection of features to existing models,
- the CAD designer views the existing models within the model repository. These scenarios depict the main forms of interaction with the application. These will be discussed in detail below.

Scenario #1

The CAD designer attempts to view an input model's features

Primary Actors: CAD designer

Preconditions: The model repository is not empty

1. The CAD designer launches the application.
2. The CAD designer submits an input model to the application.
3. The application prints out the model's relevant features.

Scenario #2

The CAD designer attempts to compare an input model or a selection of features to the existing models

Primary Actors: CAD designer

Preconditions: The model repository is not empty

The model submitted by the user is valid and interpretable

1. The CAD designer launches the application.
2. The CAD designer supplies the input.
   a) The CAD designer submits an input model.
   b) The CAD designer submits a set of features.
3. The CAD designer sets the query parameters.
4. The CAD designer initiates the "Search" function.
5. The application retrieves similar existing models.

The previous scenario offers the user two options when it comes to inputting data into the application. Option 2a indicates the user supplies a model from which the features are extracted via the same core functionality described in the first scenario. Option 2b indicates the user supplies a set of features from which a model signature would be extracted. The signature will then be used to compare different models. Should the user proceed with option 2b, the precondition need not apply.

Scenario #3

The CAD designer views the existing models within the model repository.

Primary Actors: CAD Designer

Preconditions: The model repository should exist (but it shouldn't necessarily be full)

1. The CAD designer starts the application.
2. The CAD designer navigates to the repository view.
3. The application outputs a list of models (if any) with the option to view each model's details.

From the previous use-cases and scenarios, it is shown that some elements are missing. These are the elements that would give the application a rudimentary user-friendly design and functionality. These missing elements include:

- adding, editing, and deleting existing models in the model repository,
- connecting to a repository,

- setting query parameters (this functionality will be integrated within its sister core functionality, i.e. Compare input to other models),
- and viewing query results.

For the aforementioned points, a new use-case diagram is constructed showing the additional functionality the application should offer, see Fig. 5.2. Based on the defined use-case diagram in Fig. 5.2, a new set of scenarios is now available. These scenarios will encompass all model-related operations and depict the user's interaction with the model repository as well as the search results. At this point, the search results are considered the same as a model repository, but filtered.

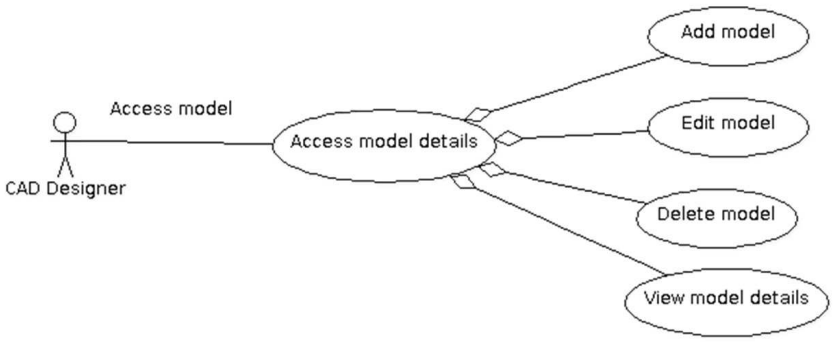


Fig. 5.2: Additional Functionality Use-case Diagram

Scenario #4

The CAD designer views a model's details

Primary Actors: CAD Designer

Preconditions: The required model exists within the repository.

If the required model is from a search result set, a search must have been run beforehand.

1. The CAD designer starts the application.
2. The CAD designer selects a model to view.
   a) From the repository
      a1) The CAD designer navigates to the model repository view.
      a2) The CAD designer selects a model to edit.
   b) From the search results
      b1) The CAD designer selects a model from the search results.
3. The application outputs the existing model information.

This scenario applies whether the CAD designer selects a model from the repository (case "a"), or if they select a model from the search results (case "b").

Scenario #5

The CAD designer adds a model to the model repository

Primary Actors: CAD Designer

Preconditions: A suitable storage medium exists to store new models (repository).

1. The CAD designer starts the application.
2. The CAD designer navigates to the model addition form.
3. The CAD designer fills in the model information.
   a) The CAD designer fills in the name, description, and other basic information.
   b) The application recognizes the model's features for storage.
4. The application saves the data to the repository.

Scenario #6

The CAD designer edits an existing model

Primary Actors: CAD Designer

Preconditions: The required model already exists in the model repository. The required model is accessible.

1. The CAD designer starts the application.
2. The CAD designer navigates to the model repository view.
3. The CAD designer selects a model to edit.
4. The application outputs the existing model information.
5. The CAD designer modifies the existing information.
6. The application saves the changes to the model repository.

Scenario #7

The CAD designer deletes an existing model

Primary Actors: CAD Designer

Preconditions: The required model already exists in the model repository. The required model is accessible.

Post-conditions: The required model is no longer available within the repository.

1. The CAD designer starts the application.
2. The CAD designer navigates to the model repository view.
3. The CAD designer selects a model to delete.
4. The application outputs the existing model information.
5. The CAD designer chooses to delete the model.
6. The application deletes the model from the model repository.

Based on the previous analysis, a list of functional requirements is obtained. These functional requirements will constitute the essential parts of the system that a user needs to perform the required functions. They also provide a user-friendly interface for the user. This consists of the following:

- the application should allow a user to store models in a model repository,
- the application should allow a user to perform basic CRUD (Create, Read, Update, and Delete) operations on the repository,
- the application should allow a user to use a model as a search parameter for a model query,
- the application should allow a user to modify the search parameters according to their desire,
- the application should allow a user to perform a feature recognition process on a model,
- the features should be represented using the Opitz coding system,
- the application should use the identified features as parameters for the model queries,
- and the application should give a user an alternative method to feature recognition that does not require the user to have a preexisting model [123].

5.1.3 The Architectural Design Phase

As it has been presented in chapter 3.1, the application consists of five basic modules, each of which interacts with other specific modules. There are also two storage modules, the CAD model index (which is internal to the system), and the CAD model repository (which is external to the system). The five basic modules include a GUI module, a feature

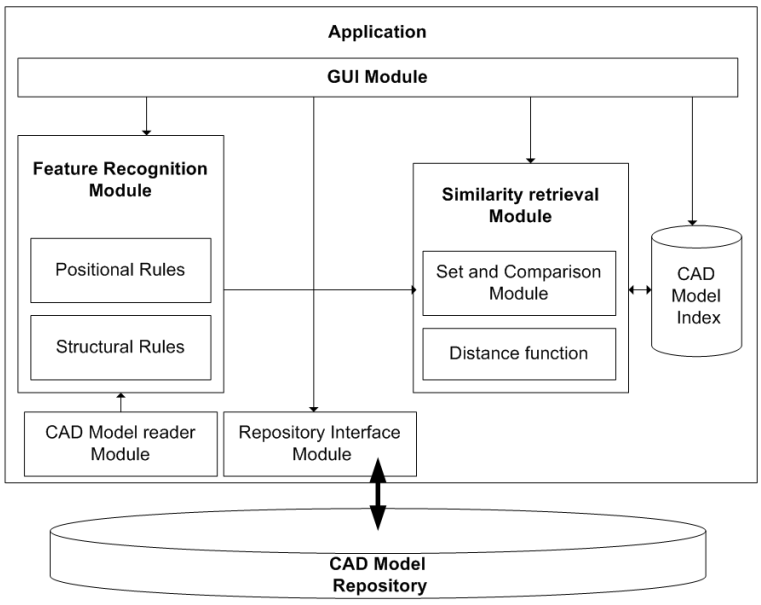


Fig. 5.3: Application Architecture Diagram

recognition module, a similarity-retrieval module, a CAD model reader module, and a repository interface module, see Fig. 5.3.

The CAD model reader module is responsible for reading CAD model files, and translating them into a set of objects that the application can deal with. This set of objects is later passed to the feature recognition module. The feature recognition module then uses these objects to identify and recognize features relevant to the required model signature. The resulting signature is then visible through the application's GUI. The resulting signature can also be used as an input to the similarity retrieval module. The similarity-retrieval module will compare the given signature to the internal CAD model index, all the while exposing the query parameters to the user via the GUI. Once the parameters are set, the user can initiate the search function and a list of models to which the parameters apply will be retrieved from the CAD model index. This approach to CAD model similarity is based on the work done by Zehtaban and Roller [126]. The repository interface module is used to update the CAD model index so that it conforms to the latest version of the CAD model repository. The separation of storage mediums (local and external) was chosen since it provides greater portability, flexibility regarding different types of repositories, and more efficiency when it comes to query execution [123][131].

CAD Model Repository

The CAD model repository is assumed to be any database on a remote system that contains information about models. The application should be able to interface with it in order to retrieve model data. The retrieved model data would then be used

to populate the CAD model index within the application so that the user would always have a local copy which would be accessible at all times. It is also assumed that the basic structure of this database would be as simple as possible.

This structure is presented in the ERD (Entity Relationship Diagram) in Fig. 5.4. From the extracted ERD and the physical description, the simplest structure for a CAD model repository is shown. These are the most basic pieces of information that need to be saved in order to retain meaningful information about a model.

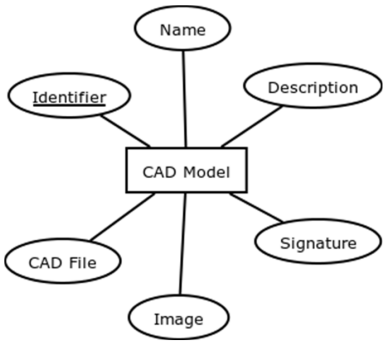


Fig. 5.4: CAD Model Repository ERD

This information also allows a saved model to be used in the similarity and retrieval functions. “CAD Model repository ERD”, see Fig. 5.4, consists of six fields in total. The “Identifier” field is a unique surrogate key assigned to each model in order to distinguish it from other existing models. “Name”, “Description”, “CAD File” and “Image” fields store the model’s name, a short description, the path to its CAD file, and the path to its image file respectively. The “Signature” field is used to store the model’s features that have been identified upon insertion. While it may be possible that the remote system contain more fields, these fields are considered mandatory for the application to function. The physical design is given as follows: CAD_MODEL (IDENTIFIER, NAME, DESCRIPTION, FILE_PATH, IMG_PATH, SIGNATURE).

CAD Model Index

The CAD model index is a local storage medium that stores relevant model details for the search and retrieval operation. The reason behind opting for a local copy of the data instead of directly using the remote system is mainly to increase efficiency. Had the remote system been used, the whole table would have had to be retrieved so that it may be filtered by signature. This becomes clearer on going into the details of the similarity-retrieval module. the advantage is that the designer can now access the similarity-retrieval functions without having to connect to a remote system. This local file contains the minimum amount of information necessary to perform

the functions the application needs to do. So, it conforms to the schema defined in the previous subsection. It should contain six fields in total. The identifier, name, description, signature, CAD file, and image file fields.

Repository Interface

The repository interface should allow the application to synchronize its local CAD model index with any type of remote CAD model repository. The most common Database Management Systems (DBMSs) should be accommodated. This includes Microsoft SQL Server, Oracle's MySQL, and PostgreSQL. The basic structure of this module is given in the class diagram in Fig. 5.5.

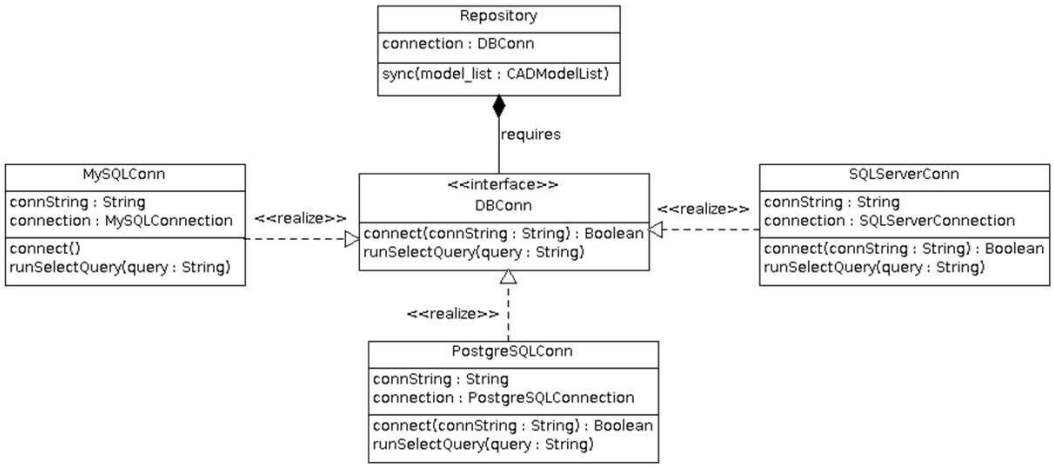


Fig. 5.5: Repository Interface Class Diagram

The repository object is what the application uses to connect to a database of any of the aforementioned types. It uses any of the three "Conn" objects to connect to a certain DBMS depending on the user's wishes. For example, if the user would choose to connect to a MySQL database server, a MySQLConn object will be instantiated, and used to retrieve all the models in that database. The repository object would then use the result to synchronize the existing local file. The synchronization process will be strictly one-way, from the repository to the application. New models will be imported from the repository, older models will be deleted. User-defined models will not be exported or affected. This is best defined in an organization's operational policy.

5.2 Prototype Development

For the realization of this research a software prototype has been designed and implemented based on the book by Schach [129]. The incremental model consists of 6 main phases: the requirements phase, the specification phase, the architectural design phase, the build-activities phase which includes detailed design, implementation, integration, testing, and delivery to the client, the maintenance phase, and the retirement phase. The requirement phase, the specification phase and the architectural phase of the current project have been deeply explained in section 5.1.

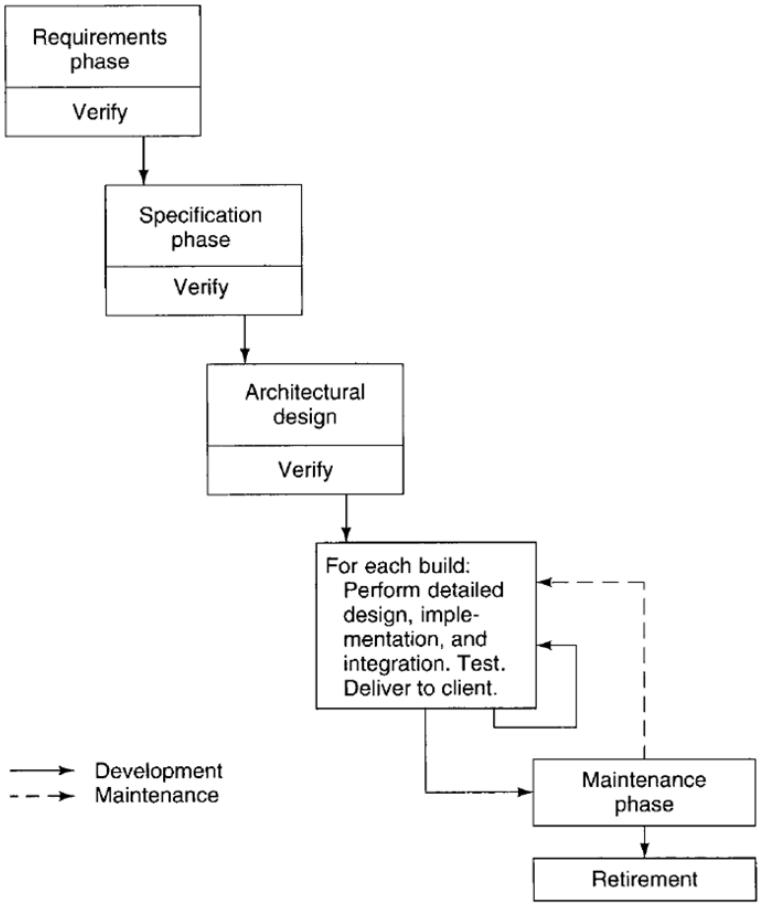


Fig. 5.6: Incremental Model by Schach [129]

The developed software has been defined in a prototype scale thus the maintenance and retirement phases have not yet been reached. In the next sections the concrete implementation of the modules which were presented earlier are presented. These modules include the feature recognition module, the similarity retrieval module, and the GUI module. Additionally, the CAD model index module is also presented to illustrate some of the implementation decisions that were faced.

5.2.1 CAD Model Index

As presented earlier in Fig. 5.3, the CAD model index is a local storage medium that contains a copy of all the relevant information regarding the CAD files within the remote repository. The type of storage selected was a local **.xml* file, Listing 5.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<model_base>
  <model>
    <identifier> 1 </identifier>
    <part_name> Countersunk Gear </part_name>
    <part_file_path>
      countersunk_awasher_no__4_0.stp
    </part_file_path>
    <part_gt_code> 140070500 </part_gt_code>
    <part_description> test. </part_description>
    <part_image_path>
      10-07062007-112834L.gif
    </part_image_path>
  </model>
</model_base>
```

Listing 5.1: An excerpt from the local file

This is mainly due to XML's portability. There might arise a situation where the user would want to export their local index into a database or import from a database. XML offers a well-defined, highly portable, and highly extensible option to perform these operations. In addition, it is adequately supported in java. For this purpose, the schema in Listing 5.2 was constructed. An excerpt from the local file is also given below.

```
<xs:schema attributeFormDefault= "unqualified" elementFormDefault = "qualified"
xmlns:xs= "http://www.w3.org/2001/XMLSchema">
  <xs:element name= "model_base">
    <xs:complexType>
      <xs:sequence>
        <xs:element name= "model">
          <xs:complexType>
            <xs:sequence>
              <xs:element type= "xs:byte" name= "ident ifier"/>
              <xs:element type= "xs:string" name= "part_name"/>
              <xs:element type= "xs:string" name= "part_file_path"/>
              <xs:element type= "xs: int" name= "part_gt_ code"/>
              <xs:element type= "xs:string" name= "part_description"/>
              <xs:element type= "xs:string" name= "part_image_path"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    < /xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 5.2: CAD Model Index XML Schema

As previously discussed in the detailed design chapter, and presented below, there are six fields. An identifier, a "name" field, a field for the path to the STEP file, a field to store the part's features (Opitz code), a description field, and a field for the path to the part's image. Using the interface, the user can add, delete, and modify entries within this file. These entries, however, will not be synchronized with the remote repository. The remote repository is used only for calling for new models.

5.2.2 Feature Recognition

Opitz code as a numerical specifier does not truly support flexibility. To overcome this problem, the focus in this research is to increase the flexibility level through other ways. For example, additional feature recognition techniques were implemented to provide the functionality of calculating the fifth digit of the Opitz form code, as well as the Opitz supplementary code. "flexibility level" or "thresholds" are exposed to the user in order to determine the last digit of the supplementary code. They were originally intended to provide some sort of flexibility to the recognition process. The reason behind this is rule-based systems are essentially rigid. They cannot perform well if they encounter a case that is not included in the provided rules. There is also no room to maneuver. For example, if a part has teeth, but they are too widely spaced, they could be considered external deviations of the part. This, unfortunately, would lead to the part being classified incorrectly. Now assume that the user only has such parts in the repositories (local and remote), and attempts to classify a new part and search for similar ones. Since the stored parts were classified correctly, and the new part is incorrectly classified, the user will not retrieve any relevant search results. The "Teeth Sensitivity" threshold remedies this situation. It allows the user to vary the detection sensitivity for teeth. The lower it is, the more likely sparse teeth formations are detected. A high value for this threshold means that more densely grouped teeth formations are detected. Thresholds, however, should not be abused. A very low value for the aforementioned threshold will result in detecting teeth where there are none. And a very high value for the aforementioned threshold will result in not being able to detect any teeth formations at all. It was previously stated that the thresholds will be used to calculate the last digit of the supplementary code, which is related to the accuracy of the Opitz code for the part in question. Given multiple tests and trials with various parts, a set of general thresholds has been found that can be applied to most cases. These values have been set as defaults within the application. It was decided that changing these values would be treated as a reduction in the accuracy of the code.

Function	Description
hasExternalScrewThread	Checks for the existence of a screw thread on the outer boundary of the part.
teethExist	Checks for the existence of teeth on the outer boundary of the part. Uses a threshold value.
getRotationCore	Gets the core cylindrical object around which the part rotates.
HolesHaveSameOrientation	Checks whether the part's holes are oriented similarly (i.e. pointing in the same direction). This function uses a threshold value to discern orientation.
isHoleRadial	Checks whether a hole is radial. This basically means that the hole is not parallel to the rotation axis, rather, perpendicular to it.
AreHolesAxial	Checks if all existing holes in the model are axial. This basically means that the hole is not perpendicular to the rotation axis, rather, parallel to it.
TeethParalleltoRotationAxis	Checks if the existing teeth are parallel to the part's rotation axis. This function uses two threshold values; one for comparing the orientation of the teeth, and the other for classifying all the teeth as parallel or not.
HasDeviations	Checks if a rotational part has deviations by checking if the objects on its edges are not composed of lines. In other words, checking if the rotational plane is cylindrical or not. This function uses a threshold value to determine how "cylindrical" a plane is.
getInnerCirclesCountForOnePlane	The modification allows the storage of "Hole" objects as they are discovered to be used later.
ClosedShell	The modification locates and stores the part's rotation axis orientation and rotation plane (if the part is rotational) for later use.

Table 5.1: Examples of functions in ClosedShell class

Based on which thresholds were altered, a level of accuracy can be determined for the second, third, fourth, and fifth digits of the Opitz code. The accuracy of these digits and the formal definition for Opitz supplementary code form, determines or calculates the values for the ninth digit to be used as a feature when performing queries [123].

5.2.3 Similarity Retrieval

Here, the main function used in the similarity retrieval module is explained. The flow is shown in Fig. 5.7. First, the CAD model index is loaded into memory. Next, the part signature is supplied by the user using either of the two available recognition methods (from STEP file, or via the wizard). Then, the signature is compared to a model from the model index. If the cosine similarity between two signatures is greater than or equal to the supplied threshold, the part from the model index is saved separately. This process continues until all the models in the model index have been checked. Once this process is complete, all saved models (those which had a higher or equal similarity than the threshold) are output to the user, see Fig. 5.7.

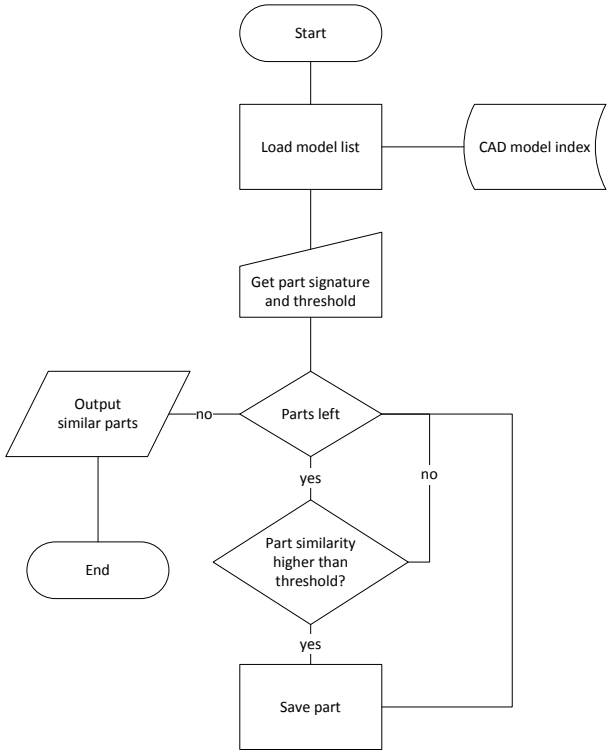


Fig. 5.7: Similarity recognition function flowchart

5.2.4 Graphical User Interface

The purpose of this section is to showcase some of the main GUI of the prototype. Furthermore, a simple walk-through is presented.

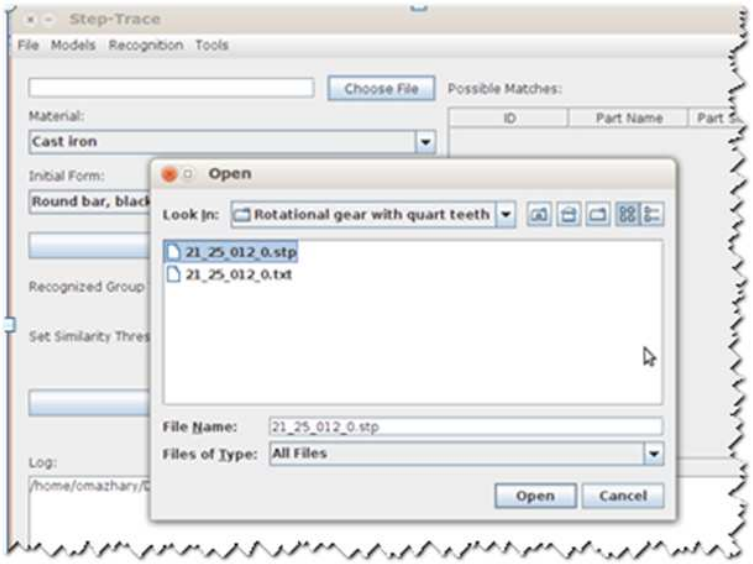


Fig. 5.8: STEP File Selection

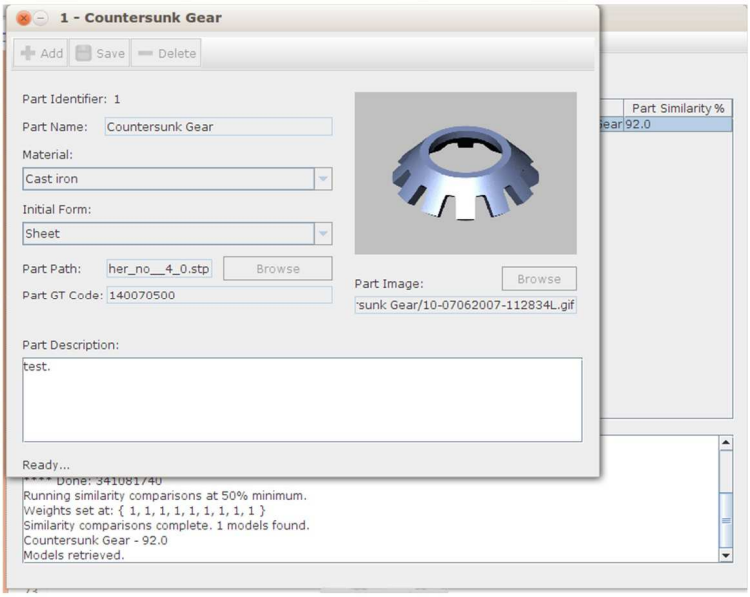


Fig. 5.9: Resulting CAD model details in the main screen

"Choose File" button, see Fig. 5.8 allows the user to navigate to and select a local STEP file. Then, the user has two combo boxes: one for different materials, and one for the initial form of the selected material. Once the user has chosen both options, they can

```
**** Start: Rotational gear with quart teeth\21_25_012_0.stp
Rotational shape
More than 3 external cylinders
External shape... Machined
Holes found: 2
Teeth found, holes found.
Rotational: using diameter as 'measure' at a value of
27.496164321899414
**** Done: 341081740
Running similarity comparisons at 50% minimum.
Weights set at: { 1, 1, 1, 1, 1, 1, 1, 1, 1 } Similarity comparisons complete.
1 model found. Countersunk Gear - 92.0
Models retrieved.
```

Fig. 5.10: The main sets

then run the recognition process by clicking the button "Run Recognition". The Opitz code for the selected STEP file is calculated, and output in the corresponding text field. Consequently, the user can set a similarity threshold of his choice, and search for similar models using the "Search for Similar Models" button. Search results will be displayed in the tabular pane on the right. Each process performed in the main screen outputs a small diagnostic, and the output is visible in the log text field. An existing feature is that the user can save the log output for the future reference.

By clicking on the "Choose File" button, an STEP file in its local directory is navigated. Once the user opens the selected file, they are redirected back to the main screen. The next step is for the user to run the recognition and search processes. Note that the model index contains only one file as shown previously in the CAD model index section. Once the search is run, the model will be retrieved. In Fig. 5.10, the excerpt of the log output belong to Fig. 5.9 is highlighted and presented separately.

Using a STEP file is not always an option to start the prototype with. If only description of a part is available, the user has the possibility of using the recognition wizard. This wizard is launched from the main screen and proceeds by asking the user several questions regarding the features, see Fig. 5.11. The answers of these questions will form the corresponding Opitz code. The questions are classified in 7 main classification:

- Part class: determining rotational or non-rotational main form. In case of non-rotational part, the user is asked to describe the external surface of the part in three categories including flat, long or cubic form. Additional, an approximate size for length, width and height is requested.
- External shape: these questions include a rough description of the external shape

of the part for example, box and box-like components, not split, etc. The second question refers to the existence of any additional shape details. In this category, the user is asked if his answers are completely accurate or not.

- Internal shape: refers to the number of part's principal bore and/or its rotational surface machining and their positioning to each other, i.e. parallel or other than parallel.
- Plane surface machining: the parts plane surface machining could be selected as with/ without guide surface, curved stepped, etc. Here, the user is asked about the accuracy of his answers as well.
- Auxiliary hole(s) and gear teeth: refers to existence of these features and by positive answers the user is asked further about details of the features such as drilling patterns.
- Material and initial forms: in order to increase the accuracy of the generated code with more characteristics rather than geometry features, these category has been designed.

When the user clicks the "Finish" button, the features are redirected back to the main screen, and the result of the wizard process, i.e. the corresponding Opitz Code is presented.

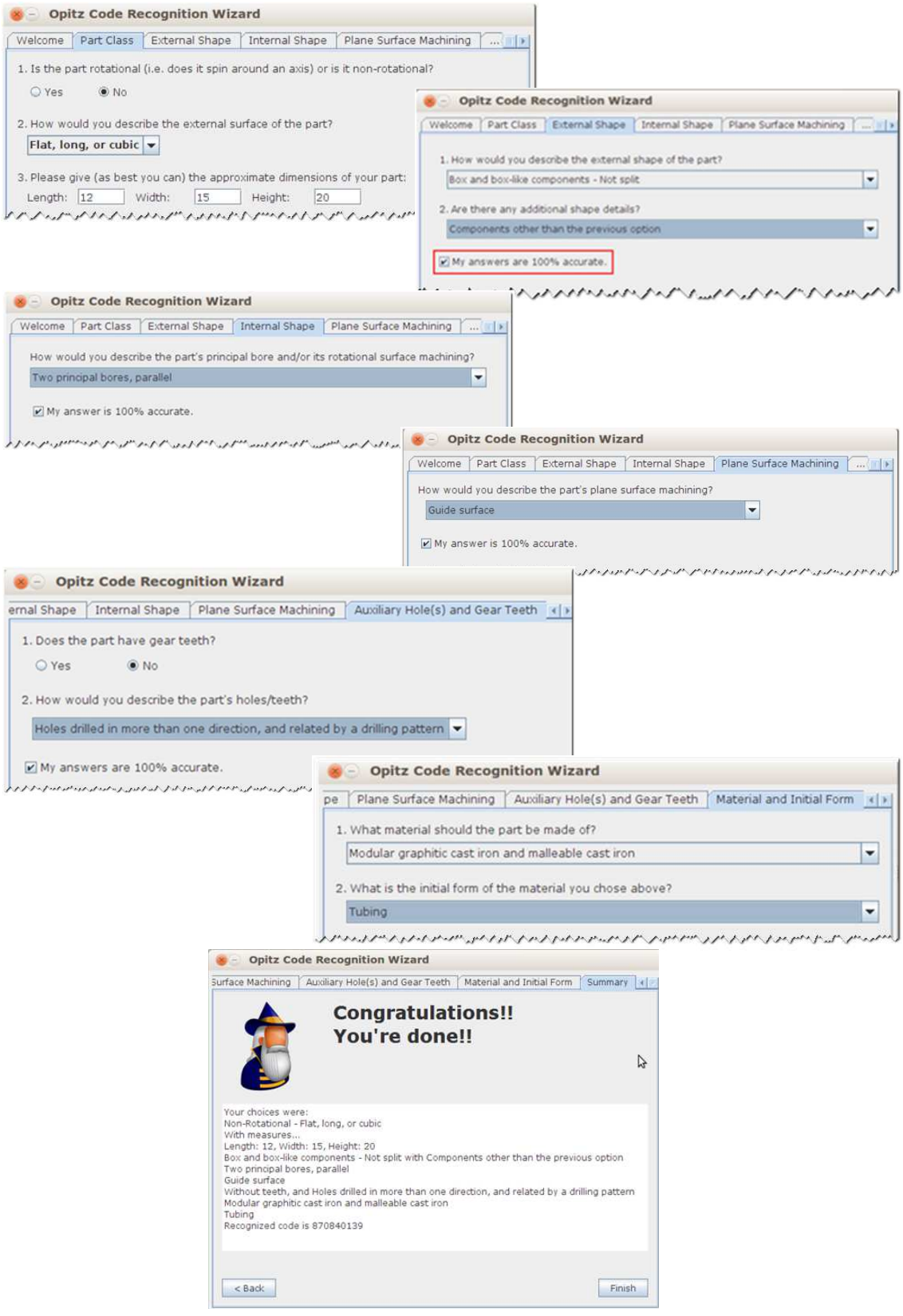


Fig. 5.11: The wizard steps

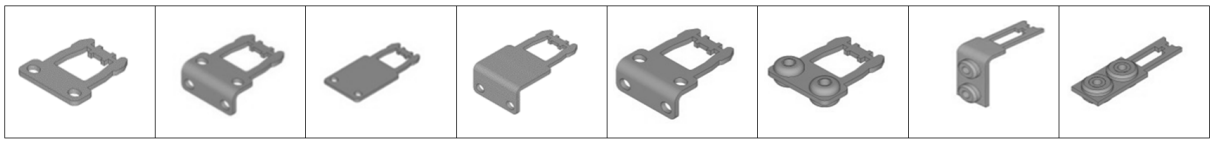
Chapter 6

Results and Evaluation

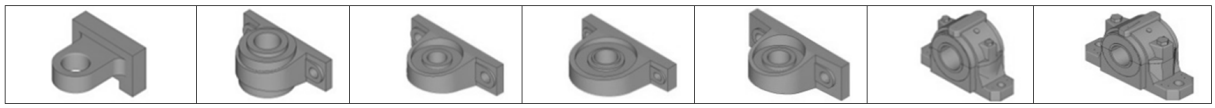
This section discusses the benchmarking of the proposed method using Engineering Shape Benchmark (ESB). Then evaluation of the proposed method including feature recognition module, similarity retrieval and GUI module is discussed. As each module has different evaluation methods, after a detailed description of the evaluation process, the results will be examined and interpreted. In the next section the validation of each module in comparison with the other method is discussed. The final section investigates the effects of the proposed method in improving Product Lifecycle Management (PLM).

6.1 Engineering Shape Benchmark (ESB)

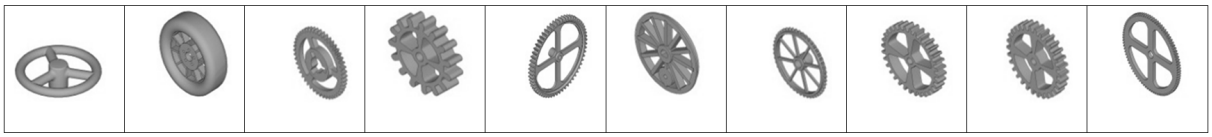
In regards to different shape representations in the mechanical engineering domain for shape based matching and retrieval, there is a limited number of standard datasets for the mechanical domain to be applied to benchmark various shape representations. Among the more commonly used evaluation methods for shape benchmarking are the Princeton Shape Benchmark (PSB) [132] and the National Design Repository (NDR) which belongs to Drexel University [133]. In this dissertation, one of the most extensive datasets for shape benchmarking, the Engineering Shape Benchmark (ESB), developed at Purdue University, was used because of the advantages discussed by Jayanti et al. [37]. ESB includes 867 models in 3D form in three main super-classes comprising Flat-thin wall components (107 models), Rectangular-cubic prism (281 models) and Solids of revolution (479 models). Within each super-class, models are, additionally, classified into groups of similar shapes. Fig. 6.1(a),(b),(c) present some examples of models varieties and classification in three super-classes of ESB including Flat-Thin Wall components, Rectangular-Cubic Prism and Solid of Revolution super-classes.



(a) Contact switches class from Flat-Thin Wall components super-class



(b) Bearing Blocks class from Rectangular-Cubic Prism super-class



(c) Spoked Wheels class from Solid of Revolution super-class

Fig. 6.1: Examples of ESB super-class clusters

To evaluate the effectiveness of the algorithm, the conventional precision-recall calculation and plot have been used for the test. Fig. 6.2, presents retrieval results using Opitz codes for ESB. Furthermore, Fig. 6.1(a) indicates the efficiency of applying the Opitz coding system classification on the ESB database by precision-recall curve. In comparison, the precision-recall curve has been used for some of the most recognized shape representation methods based on [26]. These distinguished methods include: Light Field Descriptor, 2.5D Spherical Harmonics, 2D Shape Histogram, 3D Spherical Harmonics, Convex Hull Histogram, Solid Angle Histogram, 3D Shape Distribution, Surface Area and Volume, Crinkliness and Compactness, Geometric Ratios, Moment Invariants as well as Principal Moments. The tests are run with the default threshold levels, and only take into account the form features recognized from the model file. This means the supplementary codes (i.e. digits 6, 7, 8, and 9) are not taken into account. This is mainly due to the fact that the 6^{th} digit represents the dimensions of the model, which are not hard to interpret. Digits 7 and 8 are supplied by the user. Finally, the 9^{th} digit depends on adjusting the threshold levels which were not adjusted. Furthermore, the correctness of the automatic classified results has been compared with the manual feature classification.
Taking into account the rigidity of rule-based systems, it is found that the resulting values are acceptable for considering the application capable of correct classification with regard to the Opitz code.

Regarding the size and dimension, it has to be mentioned that the form code (the first 5 digits) does not highlight a part's size. The 6^{th} digit deals with a limited perspective of the part size, where it only takes into account a single dimension. This dimension can be either the diameter (if the part is rotational), or the longest edge (if the part is non-rotational).

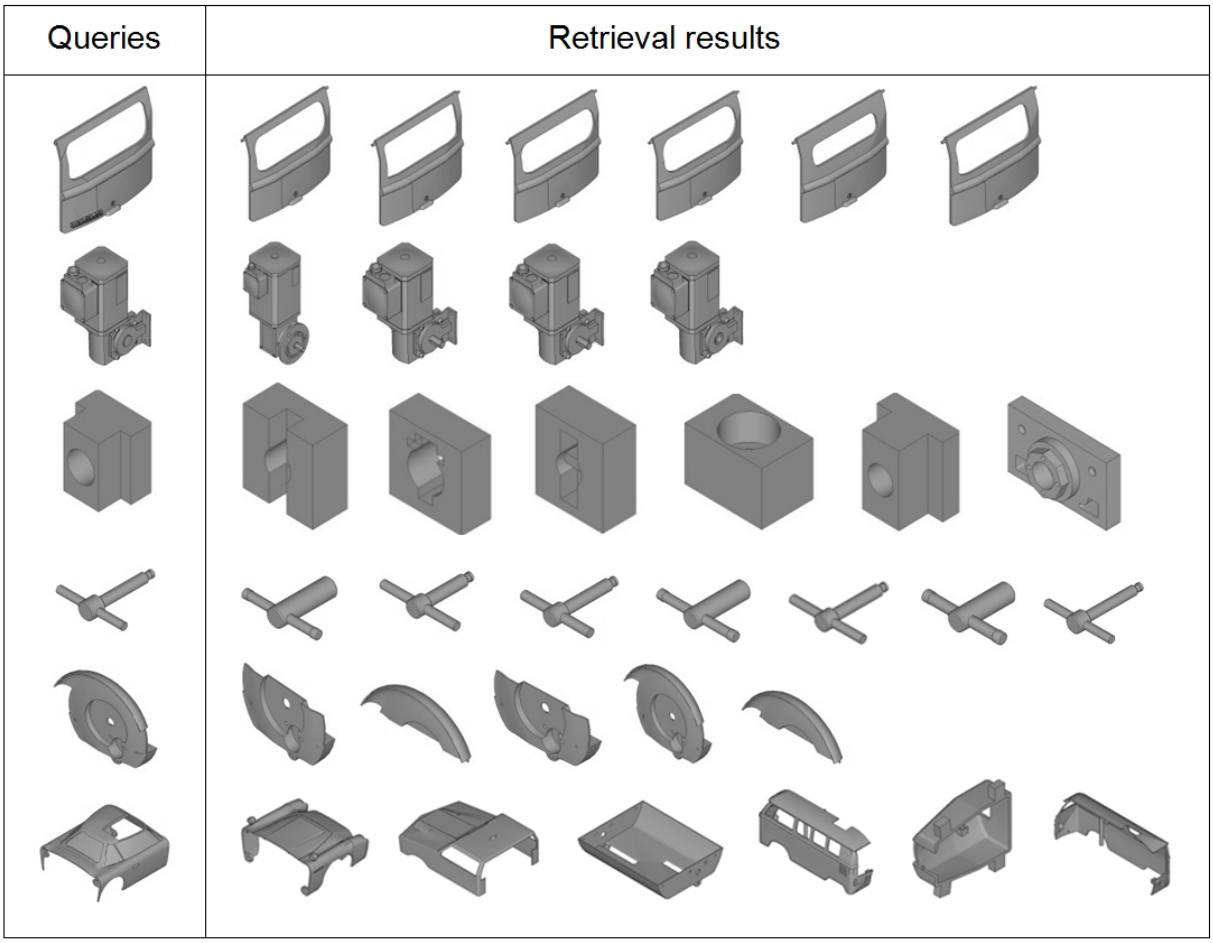


Fig. 6.2: Retrieval results using the Opitz classification system

However, this limitation can be solved by extending the Opitz code beyond its original 9 slots, and make use of the additional 4 slots to provide more descriptive measures. The comparison mechanism is flexible enough to allow the consideration of extra digits.

6.2 Evaluation of the Proposed Approach

6.2.1 Feature Recognition Module

The evaluation method applied by Frankel et al. [134] is used for this research. Two basic measures were used to evaluate the performance of the feature recognition approach; accuracy per feature, and accuracy per entire classification. By applying those concepts to this approach, accuracy per feature becomes the accuracy per digit of the Opitz code. The accuracy per entire classification becomes the accuracy per total Opitz code. Thus, an aggregation can be introduced; average accuracy per digit of the Opitz code. The formulas for these measures are shown below, where APD presents accuracy per digit; ND stands

for number of instances a digit was classified correctly, TT for total number of test, NC for number of instances the entire code was classified correctly and APC for accuracy per code.

$$Average\ APD = \frac{ND}{TT} \tag{6.1}$$

$$Average\ APD = \frac{\sum Accuracy\ per\ Digit}{Number\ of\ Digits} \tag{6.2}$$

$$Average\ APD = \frac{NC}{TT} \tag{6.3}$$

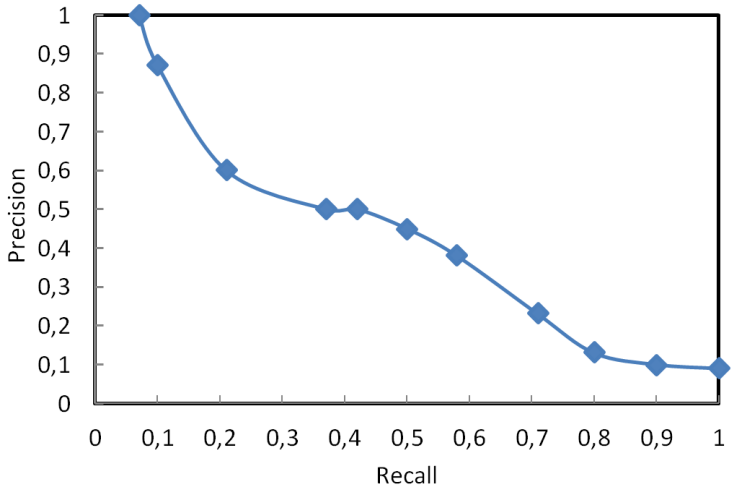
These measures are highly dependent on the test set used to evaluate the system.

6.2.2 Similarity Retrieval Module

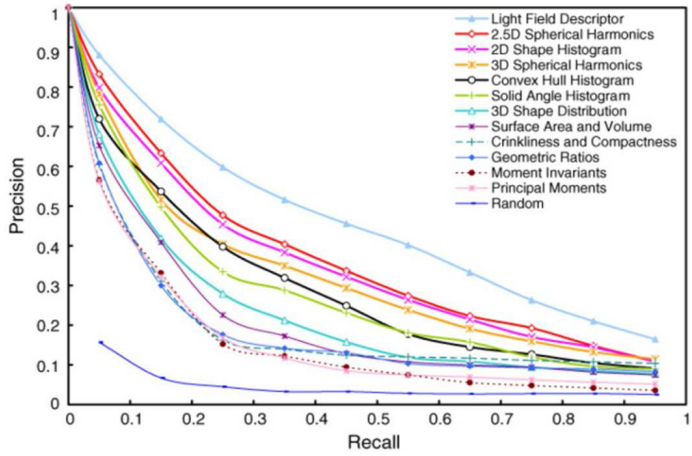
All the queries were performed using a similarity threshold of 60%. The results are as follows:

- Average Precision = 0,450
- Average Recall = 0,613
- Average F-score = 0.517

The moderate precision indicates that the application was able to retrieve the correct results, albeit not as specifically as one would prefer. That is highly dependent on the query itself, as well as the models currently used as a search space. The higher recall value indicates that among the models retrieved, a high percentage thereof were actually relevant to the query. Finally, an F-score of 0.517 indicates a moderate effectiveness of this approach as a search and retrieval technique. One of the issues observed with using the Opitz code to classify objects is the fact that it does not reflect all of the information illustrated in the part-model. For example, when classifying teeth (in the 5^{th} digit), the classification for triangular teeth is the same as that for elliptical teeth so long as both are parallel to the rotational axis. They will both have the same code, but they are different part-models which result in different parts/products and may eventually be used in completely different ways. This is, however, not a failing of the application, rather one of the Opitz code itself, which does not go into such rigorous detail. The previous results also illustrate the aforementioned issue regarding using the Opitz code. Since it



(a) The proposed method [131]



(b) The well-known methods [37]

Fig. 6.3: Comparison of Precision-recall curve representations

is a group technology code, it inherently groups models that are not necessarily 100% similar. As each enumeration of the code signifies a "group" of models, it cannot be treated as a unique identifier, thereby broadening the spectrum of the retrieved results. This is the main reason for the low precision value encountered during the tests. However, it is compensated by the high recall value, which indicates that a vast majority of (if not all) relevant models have been retrieved.

6.2.3 GUI Module

After the Opitz code of a part is known, either generated by the system through the wizard, or the designer has already set a basic design, several scenarios were defined to administrate the functions of the GUI. These scenarios encompass all model-related

operations and depict the user's interaction with the model repository as well as the search results. At this point, the search results are considered the same as a model repository, but filtered. A menu bar should be available at the top to give the user access to more advanced features, such as calibrating the weight settings, adjusting the flexibility levels for the recognition process, and connecting to a repository. It should also allow access to the local storage medium, so that the user can browse the existing models and view their details. The user has the possibility to set the similarity threshold of his choice in the main window, or predefined weights for the comparison process which can be set in the "Advanced Settings" screen, see Fig. 6.4.

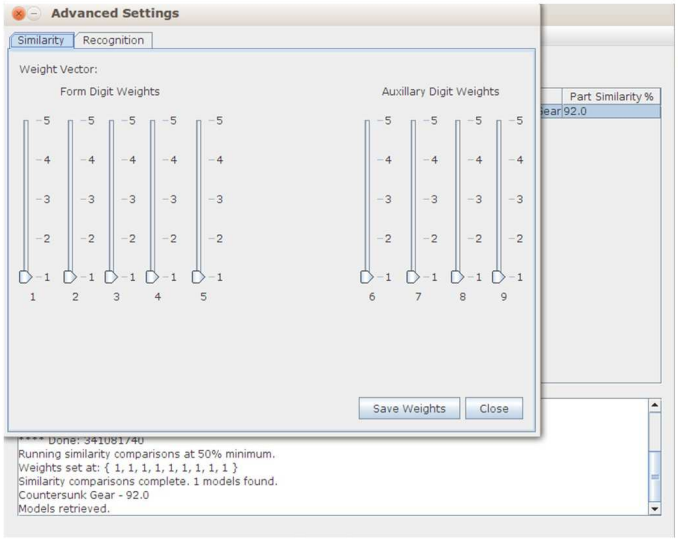


Fig. 6.4: Advanced settings for weights

The excerpt of the log output for the CAD model presented in section 5.2.1.

Using a STEP file is not always an option. For these cases, a user can apply the recognition wizard. This wizard is launched from the main screen and proceeds by asking the user questions regarding the features which will be translated and converted into an Opitz code [131].

6.3 Validation of the Proposed Approach

In this section the proposed method is validated in comparison with a number of very similar approaches to check whether or not the selected method and the proposed approach are appropriate for the objective and purpose of this research. The experimental part of this research has essentially been divided into three sections; feature recognition module, similarity retrieval module and GUI module. In this regard, in the following sections, two modules of feature recognition and similarity retrieval are validated.

6.3.1 Feature Recognition Module

The proposed method using a rule-based structure is comparable with two main groups of approaches. This comparison will help in proving of the effectiveness and efficiency of the presented approach in comparison to other similar approaches as well as providing grounds to compare both the Opitz code to a similar group technology code. The first group of approaches apply rule-based methods, as seen in the work of Liverani and Ceruti [27] as well as in [135]. The other comparison will be against the second group of methods [136], which has applied group technology as a classification method in a neural network and fuzzy environment.

Another research used Prolog rules to determine the DCLASS code of a part or product [102]. CODER, as an application, was constructed to perform this operation. CODER consists of three modules: a solid model converter, a geometry interpreter, and a part coder. The solid model converter prepares the given CAD file for processing (i.e. pre-processing) and then passes it along to the geometry interpreter. A significant part of preprocessing consists of transforming the model into predicates that are interpretable by Prolog. The geometry interpreter then takes over and uses this predicate calculus representation of the model to find and identify all mid- and low-level features that exist within the model. These may consist of axis sets, protrusions, depressions, and edge types. All the feature information is then stored into a "description list". The DCLASS code is generated through applying the description lists by the part coder.
The approach used in the current research is similar in its reliance on rule-based systems to generate the group technology code, but different in how it is applied. To generate feature information in the current approach, the application simply runs through a decision tree instead of using predicates to generate feature information. Thus any possible error in performing a complicated technique is avoided. The first applicable path is selected and followed to its end in a forward chaining manner. Furthermore, the memory utilization usually consumed by storing predicates and conclusions of such predicates is reduced.

The presented method, as seen in the evaluation section, is somewhat rigid, as a result of utilizing the fixed template of Opitz code. In this regard, in the current dissertation, additional functions are added to facilitate suppleness by applying threshold and flexibility. Such functions have been thoroughly discussed in section 5.2.2. Though the overall code accuracy is not high, the accuracy per feature as well as the average accuracy per feature measures are promising.

Concerning employment of the DCLASS code in the work done by Henderson and Musti [102] vs. the Opitz code applied in this research, it is found that the Opitz code provides much more detail. By simply comparing the length of the codes, where the DCLASS has 8 digits as opposed to a 9-digit Opitz code with 4 additional customizable digits, it is calculated that the DCLASS can have about 676,000,000 possible enumerations, as opposed to the 4.56976×10^{14} possible enumerations given by the Opitz code. In this project, the more details are available, the better the similarity-retrieval functionality will work. In addition, the DCLASS code does not place enough emphasis on form features, which are only assigned one digit. The Opitz code allocates four digits to the description of form features, which is crucial in filtering search results.

Liverani and Ceruti [27] have presented a semi-automatic similarity retrieval search method as well. They have applied a string based component description similar to group technology. 2D parts are easily accepted by the system and converted to the GT code. However, for 3D parts the encoding procedure is completely integrated in the modeling CAD interface and the code is calculated incrementally feature by feature. In this way, the similarity assessment is interactive, i.e. the designer may use either the offered model or to change it to suit the further application. The encoding procedure is integrated in the modeling CAD interface and the code is calculated incrementally feature by feature. This research has considered a collaborating encoding environment for geometrical parts as well. In addition, it has applied Opitz code as a well-known method of GT. The reason behind is to shorten the time consumed for PLM because it is currently applied in manufacturing. The method applied by Liverani and Ceruti or any other similar methods which develops theoretically a new GT, may perform well in geometrical design optimization, but they do not address industrial application, so this will not be considered for PLM optimization. For the system validation, a second group of approaches have been considered in which they apply intelligent methods such as artificial neural networks or fuzzy methods. Well-known methods of this group was perfectly analyzed and reviewed in [136] which use artificial intelligent methods to classify part models into groups based on common features. Similar to Liverani and Ceruti, in both approaches, new GT code have been designed to classify different parts. As a usual process, the network is first initialized and a new pattern is compared to the various groups and elements in the system and is assigned to the most similar features.

In another work [137] a new solution is offered for the classification challenge to part family formation. As an approach, a fuzzy environment has been applied to optimize part family formation in a GT application. The proposed classification method assigns a symbolic value to a numeric number in a simple but unrealistic way assuming a sharply defined boundary. This research utilizes a fuzzy set to define fuzzy features and to integrate fuzzy features into the code structure.
Artificial Neural Networks (ANN) are flexible when it comes to classifying models and provide good results in feature recognition. Nevertheless, they rely on the existence of a large set of training data. The training data is used to teach the ANN how to classify and assign part models to different groups. The drawback to using a machine-learning method in the context of this project is the large number of models that have to be available in order to perform both the training and testing activities. In addition, the approach proposed in [137] only considers geometrical and topological features. It does not take into account material or accuracy. The use of ANNs also limits the possibility for expansion. For example, if it was decided that additional information would be needed to describe a model, ANN would have to be discarded, re-designed and rebuilt, then it would have to be re-trained and re-tested. On the other hand, the Opitz code can simply be expanded by adding several more digits. And given the encapsulation of the functions constructed, only minor changes would need to be made in order to incorporate the new criteria.

6.3.2 Similarity Retrieval Module

The approach presented in this dissertation relies on the Opitz code to provide a model signature for model comparison. Other approaches use statistics or graphs as a signature in order to determine similarity. This section compares the presented approach in this dissertation with the one proposed by Jiantao et al. [138] as well as the method from El-Mehalawi and Miller [49][54]. The first approach uses 2D slice similarity measurements to establish a model signature, then compares them using a very basic distance function. The second uses a topological graph and attempts to compare the graph representations of different models in order to determine similarity. The differences will be discussed, as well as the various evaluation aspects used. As was mentioned in Chapter 2, slicing is one of the common methods to acquire a shape signature. The signature consists of several section such as cross-sections or longitudinal sections. The sections are then used to produce a rudimentary set of measures which can be used as a shape signature. While this approach reduces the effort required to acquire model information, it does not provide enough information regarding relevant features or general features. Particularly, there is no place for presenting CAM features such as material or PLM features such as cost. In

addition, due to the sampling that occurs within this approach, it is very likely that an indentation or a curvature will be overlooked, and thus the relevant model is ignored in the retrieval process. However, this method does have advantages which are primarily the simplicity of the distance function, as opposed to the cosine similarity and its implementation, as well as the entire feature recognition module. The only relevant measure used to evaluate the effectiveness of the queries is the accuracy of the top 5 and top 10 results, which can be likened to the precision calculated in the previous section. By that measure, it is found that the average precision is less than the accuracy of the system in question. This is mainly attributed to the large number of results returned by the approach developed here.

In the approach by El-Mehalawi and Miller [49][54], a topological graph is applied for shape representation and attempts are made to compare the graph representations of different models in order to determine the similarity comparison. In this method, a topological graph from STEP file is created where nodes and the edges correspond to the connecting edge curves between surfaces. Each node may have multiple properties. However, this approach limits itself only to the geometrical properties. Yet again, similar to the Jiantao et al. [138] method, this makes it very hard to establish any manufacturing properties or features. The largest subgraph between models is searched and is considered indicative of the similarity between two models (i.e. they share the most common topological and geometrical features). Furthermore, this approach is computationally expensive. This is valid especially for the complex models which consist of a large number of surfaces and connections.

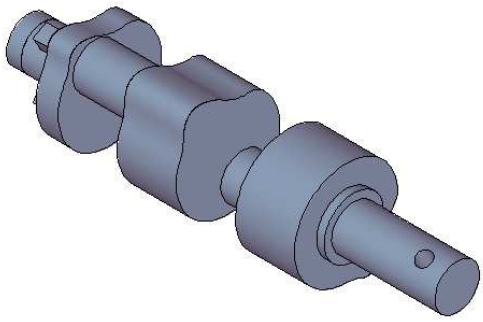
Such a drawback is avoided in the presented application by purely comparing the Opitz code vectors. However, due to the limitations of the Opitz code approach, models that are not completely similar may be considered similar. Nonetheless, as Opitz coding system is a manufacturing method of classification, it automatically inherits the main manufacturing features, as well as being a flexible method regarding the additional PLM features as a flexible structure for representation of independent string of features.

6.4 Challenging Opitz Coding System with Assemblies

The objective of this section is to present and analyze the reliability of the developed prototype as well as Opitz code in classification of complex parts or assemblies. Such complex parts may be produced by Rapid Prototyping machines or dyes. Six complex models have been selected from ESB for this evaluation. The potential of the presented methodology proposed in this dissertation as well as ability of Opitz coding system in coding of complex parts are evaluated. For the evaluation of Opitz coding system, two criteria have been considered, corresponding vertical and horizontal deficiency. The horizontal deficiency refers to the lack of extra digits to define a geometry. The vertical deficiency discusses the shortage of an extra classification in one digits of an Opitz code. Considering that each digit can obtain numbers from 0-9 corresponding to 10 status of features. In the following sections, a picture of the model, its derived Opitz code as well as Opitz text-based classification is presented. Subsequently, an evaluation of Opitz classification for the model is presented.

Model 1: Crank Shaft

The first model is a crank shaft that consists of multiple rotational components. Ideally, if they all had the same axis, the prototype would be able to recognize it. However, since the axis is slightly displaced after each cylinder, the program assumes each of them is a separate cylinder. Therefore it is disable to find a single axis to identify the component as rotational, and since all its sub-components are rotational as well, it cannot be classified as non-rotational. In regards to its Opitz code, the first digit of the code addresses whether the model is rotational or not as well as its size if it is a rotational part. Considering the open upper-bounds of that digit, there is no vertical deficiency here. The second digit describes the model's external shape. Regarding this selected number for this model, it does not accurately describe this model, since there are only 7 cylindrical objects, hence 7 diameters. This is an example of a vertical deficiency. The third digit is accurate in its description. The fourth is accurate as well, since there is no machining on any of the surfaces within the model, as well as the fifth. However, there should be a digit for multiple rotational components within a single model, which would help with this model. This is an example of a horizontal deficiency.



(a) crankshaft

1st digit: 2

Component class: Rotational components $\frac{L}{D} \geq 3$

2nd digit: 9

External shape, external shape elements: more than 10 functional diameters

3rd digit: 0

Internal shape, internal shape elements: without through bore blind hole

4th digit: 0

Plane surface machining: no surface machining

5th digit: 0

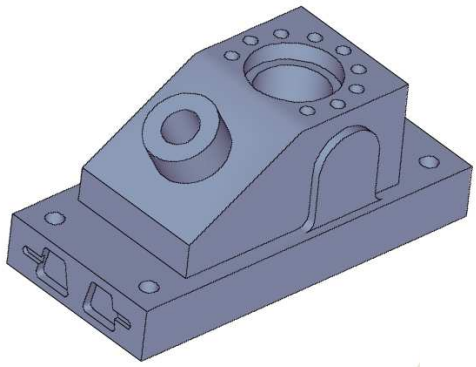
Auxiliary hole(s) and gear teeth: No auxiliary holes

(b) description

Fig. 6.5: Crankshaft and its corresponding Opitz code

Model 2: Miscellaneous

This model suffers from a similar problem to the first model. As its sub-components have different axes, therefore the shape cannot be reduced to a simple non-rotational component. For this model, there are mistakes within the classification; however, these can be remedied by the existing digits within the Opitz code. The fact that the digits of the Opitz code for this particular case have an "Other" option (usually number 9), makes it extremely difficult to run into vertical deficiencies. Also, there is no horizontal deficiencies and the existing digits are able to describe the shape of the model entirely.



(a) Miscellaneous

1st digit: 6

Component class: non-rotational component, flat component $\frac{A}{B} \leq 3$, $\frac{A}{C} \geq 4$

2nd digit: 1

External shape, external shape elements: plane, rectangular with one deviation (right angle or triangular)

3rd digit: 4

Principal bore, rotational surface machining: two principal bores, parallel

4th digit: 9

Plane surface machining: Other than guide surface, groove, slot or stepped plane surface

5th digit: 1

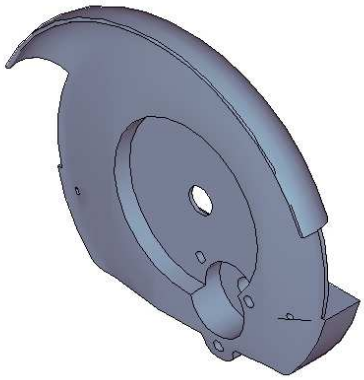
Auxiliary hole(s) and gear teeth: No gear teeth, no forming, holes drilled in one direction only

(b) description

Fig. 6.6: Miscellaneous form and its generated Opitz code

Model 3: Fender Gearbox

The third model encompasses the rotational axes however, it still consists of multiple sub-components that are a mix between rotational and non-rotational components. Since the first step is to discern whether the model is either rotational or non-rotational, it throws an exception because it is unable to reconcile that a model can consist of several sub-components. For the second digit, the closest possible description is the one given, however, it is not correct. Since the shape is not completely round, nor does it suffer casting, welding, or forming deviations. This is an example of a vertical deficiency. Horizontally, however, there are no deficiencies, as the five digits of the Opitz code adequately describe the shape. There are no other shape elements that are not described.



(a) Fender gearbox

1st digit: 6

Component class: Non-rotational component, flat component $\frac{A}{B} \leq 3$, $\frac{A}{C} \geq 4$

2nd digit: 6

External shape, external shape elements: Flat components, round or of any shape other than rectangular or right angled with small deviations due to casting, welding, forming

3rd digit: 3

Principal bore, rotational surface machining: one principal bore with shape elements

4th digit: 7

Plane surface machining: Curved surface

5th digit: 6

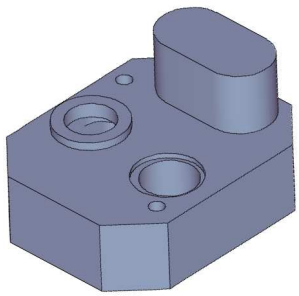
Auxiliary hole(s) and gear teeth: Forming, no gear teeth Formed with auxiliary hole(s)

(b) description

Fig. 6.7: Fender gearbox and its generated Opitz code

Model 4: Machined Block

In the forth model, the first digit is identified correctly as a non-rotational component. However, the rest of the digits are incorrectly classified, due to the large number of sub-components in the model. The program assumes the largest sub-component to be the base, and starts looking for a defined set of features in the other sub-components. Since the sub-components can be classified as components themselves, it does not make that distinction. There is a vertical deficiency with the first digit, in that the model is not cubic, however, there is no number for an oddly structured model that is not uniform.
This causes the second digit to be lacking as well, and so on (since the last 4 digits are



(a) Machined block

1st digit: 8

Component class: Non-rotational component Cubic component $\frac{A}{B} \leq 3$, $\frac{A}{C} < 4$

2nd digit: 0

Overall Shape: Rectangular

3rd digit: 4

Principal bore, rotational surface machining: Two principal bores, parallel

4th digit: 9

Plane surface machining: Other than guide surface, groove, slot, curved or stepped plane surface

5th digit: 1

Auxiliary hole(s) and gear teeth: No gear teeth, no forming Holes drilled in one direction

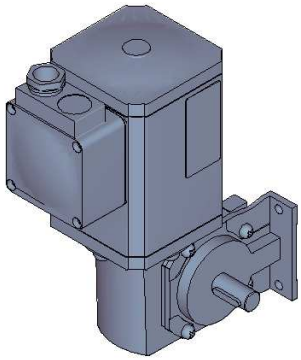
(b) description

Fig. 6.8: Machined block and its generated Opitz code

built on the first). An entire class of models is not represented here because they are not uniformly shaped. This is an example of both horizontal and vertical deficiencies, the second causing the first.

Model 5: Groschopp

The fifth and sixth models suffer from the exact same problem as the fourth model.
This model suffers from the same deficiencies of the previous model. Since this model is



(a) Groschopp

1st digit: 8

Component class: Non-rotational component Cubic component $\frac{A}{B} \leq 3$, $\frac{A}{C} < 4$

2nd digit: 5

Overall Shape: Block and block-like component Coponents other than 0 to 4

3rd digit: 0

Principal bore, rotational surface machining: No rotational machining or bore(s)

4th digit: 9

Plane surface machining: Other than guide surface, groove, slot, curved or stepped plane surface

5th digit: 6

Auxiliary hole(s) and gear teeth: Forming, no gear teeth Formed, with auxiliary holes

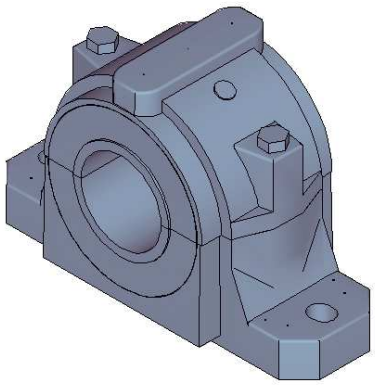
(b) description

Fig. 6.9: Groschopp and its generated Opitz code

not uniform in its structure, it cannot be classified by the Opitz code's first digit (vertical deficiency), which leads to a complete misclassification of the remaining digits (horizontal deficiency).

Model 6: Bearing Block

Similarly to the previous two models, this model is not cubic, due to its non-uniform shape (vertical deficiency in the first digit) which is not recognizable by the Opitz code. This, in turn leads to a horizontal deficiency caused by the fact that each number in the first digit of the code leads to a complete different set of digits, meanings for the remaining 4 digits.



(a) bearing Block

1st digit: 8

Component class: Non-rotational component Cubic component $\frac{A}{B} \leq 3$, $\frac{A}{C} < 4$

2nd digit: 3

Overall Shape: Block and block-like component or component with mounting or locating surface and Principal bore

3rd digit: 1

Principal bore, rotational surface machining: One principal bore, smooth

4th digit: 6

Plane surface machining: Groove and / or slot and stepped plane surfaces at right angles, inclined and/ or opposite

5th digit: 6

Auxiliary hole(s) and gear teeth: Forming, no gear teeth Formed, with auxiliary holes

(b) description

Fig. 6.10: Bearing block and its generated Opitz code

Conclusion:

In conclusion, while the Opitz code performs well for classifying simple components, it does take into account components that can be further broken down into sub-components. This results only one level of feature investigation, making the program incapable of digging deeper.

6.5 How the Proposed Approach Improves PLM?

Stark [139], defines product lifecycle management as a management process that mainly focuses on an organization's products, from the very moment they are conceived as ideas, to their final stage of retirement. PLM system is a system that keeps track of an organization's products throughout their lifecycles. Or in another words, a shared platform for different enterprise applications at each stage of Product Life Cycle (PLC) in or across enterprises [140].

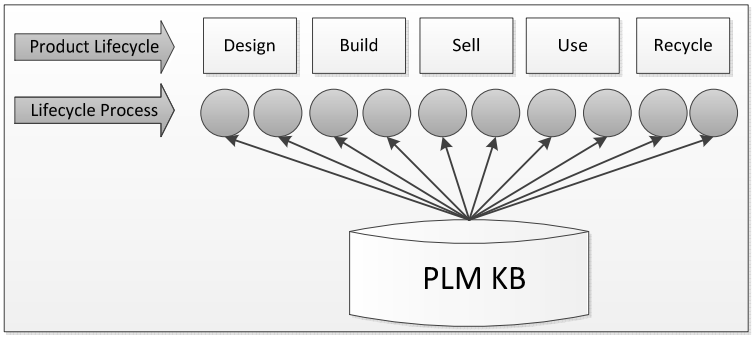


Fig. 6.11: PLM Stages as given in [141]

As seen in Fig. 6.11, PLM is started with a product's conceptualization and design, where pre-existing product information might conceivably be used as inspiration for the new products. Or rather in the product development and manufacturing whereby cloud computing is being used as an infrastructure network for automated manufacturing [142]. While PLM system provides an integrated and holistic view on data across the full product lifecycle, it raises other challenges in terms of [17]:

1. Deriving relevant knowledge from the integrated information in order to support engineering and manufacturing. processes

2. Enabling cross-discipline collaborations between actors involved in the product lifecycle using approaches for collaborative engineering.

According to Srinivasan [18] considering the requisite to improve PLM models and data integration, it is the perfect time to start integrating product data from various stages in the product lifecycle due to the convergence of three important developments:

1. The maturity of the current product data and metadata models, as well as the maturity and standardization of the current business and engineering processes that are involved with product lifecycle management. Based on [143], choosing a maturity model assists to analyze the current state and the progress made over a period of time concerning processes or structures.
2. The emergence of a new -at the time- technology dubbed the Service Oriented Architecture (SOA) which facilitates the information sharing.
3. The existence of middle ware that is flexible enough to handle the implementation of such a system.

These reasons will facilitate the improvement of existing PLM systems, and allow them to more effectively and efficiently harness the required metadata.

National Institute of Standards and Technology (NIST) proposed a product exchange framework which is called Lifecycle Information Framework and Technology (LIFT). In the very recent research of Hedberg et al. [15], LIFT was introduced as a conceptual framework for lifecycle information management and the integration of emerging and existing technologies, to form the basis of a research agenda for dynamic information modeling in support of digital-data curation and reuse in manufacturing. The framework is made up of the following layers:

1. Product Lifecycle Data
2. Data Certification and Traceability
3. Data-driven Applications

The product lifecycle data layer encompasses all the information in a product's lifecycle, from design data till customer and product support data. The data certification and traceability layer is in the place to support trust and validity throughout the product lifecycle. In addition it provides some form of procedural tracking to be able to map specific actions to events or people who carried them out. And finally, the data-driven applications layer will provide information to the applications that require specific product data in order to draw conclusions and/or conduct analyses. The complete structure can be viewed in Fig. 6.12.

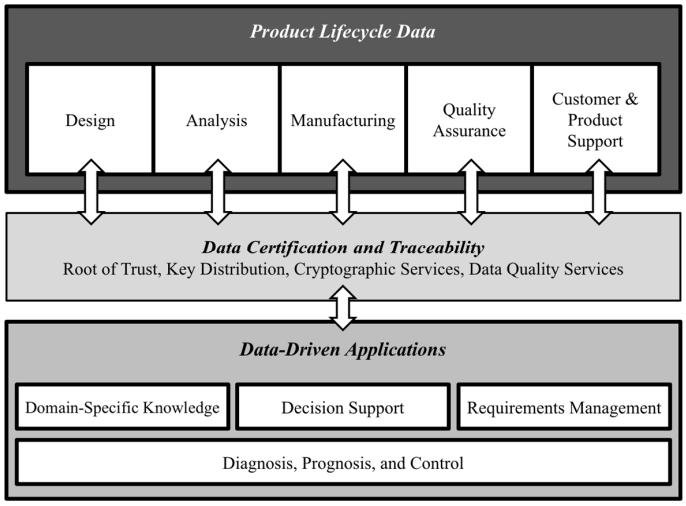


Fig. 6.12: The LIFT framework [15]

It is presented in Fig. 6.13, that the current state of the existing standards offers no single standard for the full coverage of PLM processes and functions.

This has been described in [17] as well as in [144] comprehensively. Although some standards come close, there are still unsolved research problems in order to achieve the total coverage. The other problem refers to the extent to which standard applies to the PLM cycle. As Fig. 6.13 displays, the X-axis represents the various stages of the product lifecycle while the Y-axis represents the various areas each standard can be applied.

In addition, none of these standards support the comprehensive product search by design (i.e. using design specifications or models) to relate the existing models to each other. Which -considering the PLM lifecycle- would make the design process much easier.

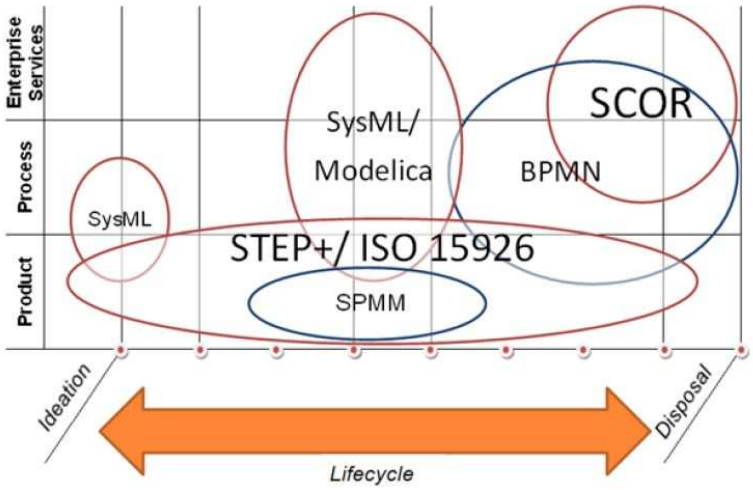


Fig. 6.13: Current state of Standard Coverage [144][145]

As a conclusion, this research improves PLM through the following points:

- Cost reduction in PLM by realization of a CAM classification method to be used as a CAD classification method. This method is used for the aim of similarity retrieval in the design phase of product development.
- Cost reduction in PLM and specifically in CAD by the reduction of design iteration as well as engineering tasks. This is done through providing access to the comprehensive information (geometrical and non-geometrical) of previously designed similar parts in the first stage of design. By applying this approach, the decision making and expertise interchange in design phase are optimized.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This dissertation defines a framework for the optimization of time and cost in PLM through an optimized 3D shape retrieval and similarity recognition focusing on the first stage of CAD design. The main research contributions are:

1. Developing a new query interface for searching of similar 3D parts based on their STEP file and their Opitz code. The main advantage of using this method is that the CAD designer has access to the comprehensive information on the designed part beyond geometrical information; such as manufacturability or cost.

2. New algorithms for STEP feature extraction and Opitz feature recognition have been developed. For the Opitz feature recognition phase, a new pose standardization algorithm for each Opitz feature (each digit in Opitz code has been considered as a feature) has been proposed and developed.

3. A rule-based system has been proposed and developed for each Opitz feature to construct the complete code. The rule-based system comprises of three main categories; R1, R2 and R3. R1 category comprises of the geometrical rules mainly based on the first digit of Opitz coding system for rotational as well as the non-rotational parts. R2 category presents operations with plane surfaces parallel to Y-plan, and R3 category has been developed for the operations with plane surfaces parallel to Z-plane.

4. A new quantitative similarity retrieval algorithm has been proposed and developed. This method ascertains local similarity retrieval as well as partial similarity retrieval. As Opitz code can include wide range of product data, it is possible to prioritize

them as well. Using this approach, the shape signature is extended to a product signature including integrated data.

5. The benchmarking of result of the proposed method with well-known shape signatures presented an effective performance. The result would be even more accurate when including more product data beyond the geometrical data.

7.1.1 Query Interface

The current prototype employs the STEP file format which has been saved as .obj or .stl . In addition, the CAD models which are retrieved are presented as a picture. In the future, the prototype can have direct access to the CAD software and it could be possible to interact with the retrieved CAD model for the further operations in CAD software.
The "code recognition wizard" feature in the prototype was designed almost for generation of the geometrical code since the utilized benchmark (Engineering Shape Benchmark) data was focused merely on CAD parts and only one question regarding the material. In a real case, this section should be expanded with more relevant questions to classify other attributes of a part as well.
In addition, by adding extra attributes to the code for a specific part series, the section of "Similarity Weight Vectors" for "Auxiliary Digit Weights" should be adjusted as well. As was discussed earlier, the quality of the retrieval for a query directly depends on a major factor, which is the number of auxiliary digits. When there is more information about a part, the result of retrieval would be more precise.

7.1.2 Similarity Comparison

There is still other information about a part which could be classified and adopted in the product signature (here the Opitz code), such as color and texture. Furthermore, as seen in Chapter 6, the more attributes which are added to a code, the more accurate the result of similarity would be. The idea from the beginning of this dissertation was having a comprehensive reference, which includes almost all information about a part. Having such a wide-ranging reference would remedy the weaknesses of Opitz code. As was discussed in Chapter 6, the geometry classification by Opitz code sometimes fails in the uniqueness aspect as two different parts may have an identical geometrical reference. Essentially this is not considered as a problem in the group technology and it is referred to a natural characteristic of the group technology for the grouping of similar parts. However, in CAD design and by means of similarity retrieval, uniqueness plays an important role. By adding extra attributes to an Opitz code, the ambiguity would reduce

greatly and directly.
Apart from the geometrical aspect, the characteristics which each digit refers to, could additionally be defined. Although there are already two digits dedicated to the individual remarks (elements) of a part in the Opitz code, extra digits could be added as well. For example, if there are some important characteristics for a part's series, they could be specified as well and digits could be allocated to them.

To conclude, the proposed framework for 3D model searching has proven to be flexible method for similarity recognition. And it offers a new perspective on product development which has potential for development in many interesting directions. Shape similarity assessment based grouping of parts has a limitation when applied to the sheet metal domain because parts that are dissimilar in shape can sometime be produced using a common press-brake setup but parts with a similar shape cannot necessarily be produced on the same press-brake setup.

7.2 Future Work

Future product development should consider the downstream design factors in the upstream design procedure. Product development ought to contemplate the methods applied in different phases of product development. In fact, overview analysis of different phases are required to recognize the redundant methods and plan to avoid such steps instead of repetition and re-inventing the wheel. Considering the rather limited access to the updated methods and process used in industry, Opitz code as a pattern of CAM classification has been nominated for this research. However, the proposed framework can be adopted to any CAM classification method with the same methodology for the feature recognition and extraction. In this research, similarity comparison and identification of parts has been considered in which multiple parts may constitute an assembly, see Fig. 7.1. Adding extra methodologies for classification of assemblies and modules to the developed prototype makes it an interesting and powerful tool for the universal similarity retrieval. In this case, the user can choose the structure level of the similarity retrieval.

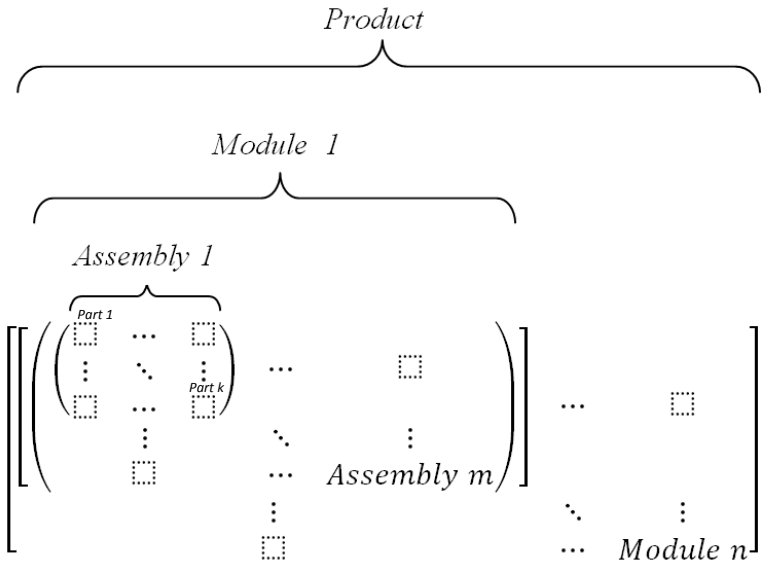


Fig. 7.1: Schematic representation of the product data layers matrix

List of Figures

Bibliography

[1] P. Bilello. What does a successful PLM implementation look like? In *PLM Innovation Conference*, 2012.

[2] D.G. Ullman. *The Mechanical Design Process*, volume 2. McGraw-Hill New York, 1992.

[3] Google Inc. Google Street View.

[4] Google Inc. Google Earth.

[5] A. Al-Nuaimi, M. Piccolrovazzi, S. Gedikli, E. Steinbach, and G. Schroth. Indoor location retrieval using shape matching of kinectfusion scans to large-scale indoor point clouds. In *Proceedings of Eurographics Workshop on 3D Object Retrieval*, pages 31–38, 2015.

[6] B. Li, Y. Lu, C. Li, A. Godil, T. Schreck, M. Aono, M. Burtscher, H. Fu, T. Furuya, H. Johan, et al. Shrec14 track: Extended large scale sketch-based 3D shape retrieval. In *Proceedings of Eurographics Workshop on 3D object retrieval*, pages 121–130, 2014.

[7] M. Savva, F. Yu, H. Su, A. Kanezaki, T. Furuya, R. Ohbuchi, Z. Zhou, R. Yu, S. Bai, X. Bai, M. Aono, et al. Shrec17 track: Large-scale 3D shape retrieval from shapenet core55. In *Proceedings of Eurographics Workshop on 3D Object Retrieval*, pages 39–50, 2017.

[8] A. Giachetti, F. Farina, F. Fornasa, A. Tatsuma, C. Sanada, M. Aono, S. Biasotti, A. Cerri, and S. Choi. Shrec15 track: Retrieval of non-rigid (textured) shapes using low quality 3D models. In *Proceedings of Eurographics Workshop on 3D Object Retrieval*, page 4, 2015.

[9] A. Cerri, S. Biasotti, M. Abdelrahman, J. Angulo, K. Berger, L. Chevallier, M. El-Melegy, A. Farag, F. Lefebvre, A. Giachetti, et al. Shrec13 track: Retrieval on textured 3D models. In *Proceedings of Eurographics Workshop on 3D Object Retrieval*, pages 73–80, 2013.

[10] B. Venu and V.R. Komma. Step-based feature recognition from solid models having non-planar surfaces. *International Journal of Computer Integrated Manufacturing*, 30(10):1011–1028, 2017.

[11] R. Roj and H.-B. Woyand. An examination of engineering parts in large CAD-databases in order to create adjacency matrices and build clusters. In *19th International Conference on Intelligent Engineering Systems (INES)*, pages 97–102. IEEE, 2015.

[12] K. Sfikas. *Retrieval of 3-dimensional rigid and non-rigid objects*. PhD thesis, National and Kapodistrian University of Athens, 2014.

[13] J. Knopp. *Large-scale Classification and Retrieval of 3D Shapes*. PhD thesis, Arenberg Doctoral School, Belgium, 2015.

[14] J. Kim. *Region Detection and Matching for Object Recognition*. PhD thesis, The University of Texas, 2013.

[15] T. Hedberg, A.B. Feeney, M. Helu, and J.A. Camelio. Toward a lifecycle information framework and technology in manufacturing. *Journal of Computing and Information Science in Engineering*, 17(2), 2017.

[16] D. Dori. *Model-based systems engineering with OPM and SysML*. Springer, 2016.

[17] S. El Kadiri and D. Kiritsis. Ontologies in the context of product lifecycle management: state of the art literature review. *International Journal of Production Research*, 53(18):5657–5668, 2015.

[18] V. Srinivasan. An integration framework for product lifecycle management. *Computer-Aided Design*, 43(5):464–478, 2011.

[19] H. Grabowski, R. Anderl, and M.J. Pratt. *Advanced Modelling for CAD/CAM systems*, volume 7. Springer Science & Business Media, 1991.

[20] J. Ovtcharova and U. Jasnoch. Featured-based design and consistency management in CAD applications: a unified approach. *Advances in Engineering Software*, 20(2-3):65–73, 1994.

[21] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[22] G. Lavoué. Combination of bag-of-words descriptors for robust partial shape retrieval. *The Visual Computer*, 28(9):931–942, 2012.

[23] H. Tabia, H. Laga, D. Picard, and P.-H. Gosselin. Covariance descriptors for 3D shape matching and retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4185–4192, 2014.

[24] A. Chakrabarti. *Engineering design synthesis: understanding, approaches and tools.* Springer Science & Business Media, 2002.

[25] A. Cardone, S.K. Gupta, and M. Karnik. A survey of shape similarity assessment algorithms for product design and manufacturing applications. *Journal of Computing and Information Science in Engineering*, 3(2):109–118, 2003.

[26] W. Gao, S.M. Gao, Y.S. Liu, J. Bai, and B.K. Hu. Multiresolutional similarity assessment and retrieval of solid models based on DBMS. *Computer-Aided Design*, 38(9):985–1001, 2006.

[27] A. Liverani and A. Ceruti. Interactive GT code management for mechanical part similarity search and cost prediction. *Computer-Aided Design & Application*, 7(1):1–15, 2010.

[28] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani. Shape based searching for product lifecycle applications. *Computer-Aided Design*, 37(13):1435–1446, 2005.

[29] L.K. Kyprianou. *Shape classification in computer-aided design.* PhD thesis, University of Cambridge, 1980.

[30] B. Babic, N. Nesic, and Z. Miljkovic. A review of automated feature recognition with rule-based pattern recognition rule-based pattern recognition. *Computers in Industry*, 59(4):321–337, 2008.

[31] L. Zehtaban and D. Roller. Review of shape-based similarity algorithms and design retrieval methods for computer-aided design and manufacturing. In *Proceedings of the First International Conference on Advances in Information Mining and Management*, 2011.

[32] L. Zehtaban and D. Roller. Application of similarity comparison methods in product design procedure (3D shape searching techniques). In *Proceedings of Advanced Design Concepts and Practice Workshop*, 2011.

[33] G. Cybenko, A. Bhasin, and K.D. Cohen. Pattern recognition of 3D CAD objects: Towards an electronic yellow pages of mechanical parts. *International Journal of Smart Engineering System Design*, 1(1):1–13, 1997.

[34] S.M. Yoon, M. Scherer, T. Schreck, and A. Kuijper. Sketch-based 3D model retrieval using diffusion tensor fields of suggestive contours. In *Proceedings of the 18th ACM International Conference on Multimedia*, pages 193–200, 2010.

[35] J. Corney, H. Rea, D. Clark, J. Pritchard, M. Breaks, and R. MacLeod. Coarse filters for shape matching. *IEEE Computer Graphics and Applications*, 22(3):65–74, 2002.

[36] R. Sung, J.M. Ritchie, H.J. Rea, and J. Corney. Automated design knowledge capture and representation in single-user CAD environments. *Journal of Engineering Design*, 22(7):487–503, 2011.

[37] S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani. Developing an engineering shape benchmark for CAD models. *Computer-Aided Design*, 38(9):939–953, 2006.

[38] C. Zhang and T. Chen. Active learning for information retrieval: Using 3D models as an example. *Carnegie Mellon Technical Report: AMP01-04*, 2001.

[39] B. Settles. Active learning literature survey. Technical report, 2009.

[40] J.P.M. De Sá. *Pattern recognition: concepts, methods and applications*. Springer Science & Business Media, 2001.

[41] D.V. Vranić. *3D model retrieval*. PhD thesis, University of Leipzig, 2004.

[42] D.V. Vranić, D. Saupe, and J. Richter. Tools for 3D-object retrieval: Karhunen-loeve transform and spherical harmonics. In *IEEE Fourth Workshop on Multimedia Signal Processing*, pages 293–298, 2001.

[43] M.R. Ruggeri, D.V. Vranić, and D. Saupe. Shape similarity search for surfel-based models. In *Computer Vision and Graphics*, pages 131–140. Springer, 2006.

[44] M. Kazhdan and T. Funkhouser. Harmonic 3D shape matching. In *ACM SIGGRAPH 2002 conference abstracts and applications*, page 191, 2002.

[45] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*, volume 6, pages 156–164, 2003.

[46] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3D models. *ACM Transactions on Graphics (TOG)*, 22(1):83–105, 2003.

[47] P. Papadakis, I. Pratikakis, S. Perantonis, and T. Theoharis. Efficient 3D shape matching and retrieval using a concrete radialized spherical projection representation. *Pattern Recognition*, 40(9):2437–2452, 2007.

[48] K. Viswanathan, S. Chowdhury, and Z. Siddique. Shape comparison of 3D models based on features and parameters. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 43277, pages 647–657, 2008.

[49] M. El-Mehalawi and R.A. Miller. A database system of mechanical components based on geometric and topological similarity. Part I: Representation. *Computer-Aided Design*, 35(1):83–94, 2003.

[50] D. McWherter, M. Peabody, W.C. Regli, and A. Shokoufandeh. Transformation invariant shape similarity comparison of solid models. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 80241, pages 303–312. American Society of Mechanical Engineers, 2001.

[51] D. McWherter, M. Peabody, W.C. Regli, and A. Shokoufandeh. Solid model databases: techniques and empirical results. *Journal of Computing and Information Science in Engineering*, 1(4):300–310, 2001.

[52] W.C. Regli, A. Shokoufandeh, and D. Bespalov. Multi-scale segmentation and partial matching 3D models, 2011. US 8266079 B2.

[53] M. Peabody and W.C. Regli. Clustering techniques for databases of CAD models. Technical report, Department of Mathematical and Computer Science, Drexel University, Technical Report DU-MCD-01-0, 2001.

[54] M. El-Mehalawi and R.A. Miller. A database system of mechanical components based on geometric and topological similarity. part II: indexing, retrieval, matching and similarity assessment. *Computer-Aided Design*, 35(1):95–105, 2003.

[55] C.-F. You and Y.-L. Tsai. 3D solid model retrieval for engineering reuse based on local feature correspondence. *The International Journal of Advanced Manufacturing Technology*, 46(5):649–661, 2010.

[56] H. Sundar, D. Silver, N. Gagvani, and S. Dickson. Skeleton based shape matching and retrieval. In *Proceedings of Shape Modeling and Application*, pages 130–139, 2003.

[57] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani. A multi-scale hierarchical 3D shape representation for similar shape retrieval. In *Proceedings of the TMCE*, 2004.

[58] S. Jayanti, Y. Kalyanaraman, and K. Ramani. Shape-based clustering for 3D CAD objects: A comparative study of effectiveness. *Computer-Aided Design*, 41(12):999–1007, 2009.

[59] C.-F. You, Y.-L. Tsai, and K.-Y. Liu. Representation and similarity assessment in case-based process planning and die design for manufacturing automotive panels. *The International Journal of Advanced Manufacturing Technology*, 51(1):297–310, 2010.

[60] B. T. Messmer and H. Bunke. Subgraph isomorphism in polynomial time. Technical report, 1995.

[61] M. Ramanathan. Matching of shapes bound by freeform curves. *Computer-Aided Design and Applications*, 9(2):133–146, 2012.

[62] M. Weber, M. Liwicki, and A. Dengel. Indexing with well-founded total order for faster subgraph isomorphism detection. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 185–194. 2011.

[63] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 203–212, 2001.

[64] S. Ruiz-Correa, L. Shapiro, and M. Melia. A new signature-based method for efficient 3-D object recognition. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, 2001.

[65] T. Hong, K. Lee, and S. Kim. Similarity comparison of mechanical parts to reuse existing designs. *Computer-Aided Design*, 38(9):973–984, 2006.

[66] Z.Q. Chen, K.S. Zou, W.H. Ip, and C.Y. Chan. 3D model retrieval based on fuzzy weighted shape distributions. *Advanced Materials Research*, 201-203:1678–1681, 2011.

[67] P.K. Jain and S. Kumar. Automatic feature extraction in PRIZCAPP. *International Journal of Computer Integrated Manufacturing*, 11(6):500–512, 1998.

[68] M.R. Henderson and D.C. Anderson. Computer recognition and extraction of form features: a CAD/CAM link. *Computers in Industry*, 5(4):329–339, 1984.

[69] M. Sadaiah, D.R. Yadav, P.V. Mohanram, and P. Radhakrishnan. A generative computer-aided process planning system for prismatic components. *The International Journal of Advanced Manufacturing Technology*, 20:709–719, 2002.

[70] S. Joshi and T.-C. Chang. Graph-based heuristics for recognition of machined features from a 3D solid model. *Computer-Aided Design*, 20(2):58–66, 1988.

[71] O. Owodunni and S. Hinduja. Evaluation of existing and new feature recognition algorithms: Part 1: Theory and implementation. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 216(6):839–851, 2002.

[72] P.K. Venuvinod and S.Y. Wong. A graph-based expert system approach to geometric feature recognition. *Journal of Intelligent Manufacturing*, 6(3):155–162, 1995.

[73] M. Marefat and R.L. Kashyap. Geometric reasoning for recognition of three-dimensional object features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):949–965, 1990.

[74] Q. Ji and M.M. Marefat. A dempster-shafer approach for recognizing machine features from CAD models. *Pattern Recognition*, 36(6):1355–1368, 2003.

[75] H. Parvaz and M.J. Nategh. A multi-TAD automatic machining feature recognition framework using hybrid approach. In *National Conference on Mechanical Engineering of Iran*, 2014.

[76] Q. Ji and M.M Marefat. Bayesian approach for extracting and identifying features. *Computer-Aided Design*, 27(6):435–454, 1995.

[77] Z. Huang and D. Yip-Hoi. High-level feature recognition using feature relationship graphs. *Computer-Aided Design*, 34(8):561–582, 2002.

[78] P. Di Stefano, F. Bianconi, and L. Di Angelo. An approach for feature semantics recognition in geometric models. *Computer-Aided Design*, 36(10):993–1009, 2004.

[79] T.C. Woo. Feature extraction by volume decomposition. In *Proceedings of Conference on CAD/CAM Technology in Mechanical Engineering*, pages 76–94, 1982.

[80] Y.S. Kim. Recognition of form features using convex decomposition. *Computer-Aided Design*, 24(9):461–476, 1992.

[81] A. Eftekharian and M.I. Campbell. Convex decomposition of 3D solid models for automated manufacturing process planning applications. In *International Design*

Engineering Technical Conferences and Computers and Information in Engineering Conference, volume 45011, pages 727–735. American Society of Mechanical Engineers, 2012.

[82] H.K. Miao, N. Sridharan, and J.J. Shah. CAD-CAM integration using machining features. *International Journal of Computer Integrated Manufacturing*, 15(4):296–318, 2002.

[83] H.S. Nagaraj and B. Gurumoorthy. Automatic extraction of machining primitives with respect to preformed stock for process planning. *Journal of Manufacturing Systems*, 20(3):210–222, 2001.

[84] S. Wan, Y. Huang, Q. Wang, L. Chen, and Y. Sun. A new approach to generic design feature recognition by detecting the hint of topology variation. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 45011, pages 297–306. American Society of Mechanical Engineers, 2012.

[85] H. Parvaz and M.J. Nategh. A multi-TAD framework for recognizing machining features using hint based recognition algorithm. In *Advanced Materials Research*, volume 445, pages 905–910. Trans Tech Publ, 2012.

[86] H. Sakurai and P. Dave. Volume decomposition and feature recognition, part II: curved objects. *Computer-Aided Design*, 28(6):519–537, 1996.

[87] Y.-J. Tseng and S.B. Joshi. Recognition of interacting rotational and prismatic machining features from 3-D mill-turn parts. *International Journal of Production Research*, 36(11):3147–3165, 1998.

[88] J. Corney, C. Hayes, V. Sundararajan, and P. Wright. The CAD/CAM interface: A 25-year retrospective. *Journal of Computing and Information Science in Engineering, ASME*, 5(3):188–197, 2005.

[89] W.F. Bronsvoort and A. Noort. Multiple-view feature modeling for integral product development. *Computer-Aided Design*, 36(10):929–946, 2004.

[90] S. Meeran, J. Taib, and M. Afzal. Recognizing features from engineering drawings without using hidden lines: a framework to link feature recognition and inspection systems. *International Journal of Production Research*, 41(3):465–495, 2003.

[91] A.D. McCormack and R.N. Ibrahim. Process planning using adjacency-based feature extraction. *International Journal of Advanced Manufacturing Technology*, 20(11):817–823, 2002.

[92] M.G.L. Sommerville, D. Clark, and J.R. Corney. Viewer-centered geometric feature recognition. *Journal of Intelligent Manufacturing*, 12(4):359–375, 2001.

[93] S. Gao and J.J. Shah. Automatic recognition of interacting machining features based on minimal condition subgraph. *Computer-Aided Design*, 30(9):727–739, 1998.

[94] Y.G. Li, Y.F. Ding, W.P. Mou, and H. Guo. Feature recognition technology for aircraft structural parts based on a holistic attribute adjacency graph. In *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, volume 224, pages 271–278, 2009.

[95] X.G. Ye, J.Y.H. Fuh, and K.S. Lee. A hybrid method for recognition of undercut features from moulded parts. *Computer-Aided Design*, 33(14):1023–1034, 2001.

[96] V. Sundararajan and P.K. Wright. Volumetric feature recognition for machining components with freeform surfaces. *Computer-Aided Design*, 36(1):11–25, 2004.

[97] S.P. Mitrofanov. *Scientific Principles of Group Technology: (Nauchnye Osnovy Gruppovoĭ Tekhnologii)*. National Lending Library for Science and Technology, 1966.

[98] J. Barton and D. Love. Retrieving designs from a sketch using an automated GT coding and classification system. *Production Planning & Control*, 16(8):763–773, 2005.

[99] C.R. Alavala. *CAD/CAM: Concepts and Applications*. PHI Learning Private Limited, New Delhi, 2008.

[100] M.P. Groover. *Automation, production systems, and computer-integrated manufacturing, Third Edition*. Pearson Education India, 2008.

[101] D. Allen and P. Smith. *Part Classification and Coding Monograph, No. 3*. Brigham Young University, CAM Software Laboratory, 1982.

[102] M.R. Henderson and S. Musti. Automated group technology part coding from a three-dimensional CAD database. *Journal of Manufacturing Science and Engineering*, 110:278–287, 1988.

[103] H. Opitz. *Verschlüsselungsrichtlinien und Definitionen zum werkstückbeschreibenden Klassifizierungssystem*. Girardet, Essen, 1966.

[104] R.L. Timings and S.P. Wilkinson. *Manufacturing technology, Vol. II., second edition*. Pearson Education Limited, Harlow, 2000.

[105] J.A. Holton. The coding process and its challenges. *The Grounded Theory Review*, 9(1), 2010.

[106] V. Quintana, L. Rivest, R. Pellerin, F. Venne, and F. Kheddouci. Will model-based definition replace engineering drawings throughout the product lifecycle? A global perspective from aerospace industry. *Computers in Industry*, 61(5):497–508, 2010.

[107] E. Reid, D.A. Harrod Jr., W.B. Gruttke, et al. The Initial Graphics Exchange Specification (IGES) version 6.0, baseline 1/98. 2001.

[108] V. Quintana, L. Rivest, R. Pellerin, and F. Kheddouci. Re-engineering the engineering change management process for a drawing-less environment. *Computers in Industry*, 63(1):79–90, 2012.

[109] L. Ding, A. Ball, J. Matthews, C. McMahon, and M. Patel. Product representation in lightweight formats for product lifecycle management (PLM). In *4th international conference on digital enterprise technology*, page 47. Citeseer, 2007.

[110] E.A. Nasr and A. Kamrani. *Rapid Prototyping: Theory and Practice*, volume 6, chapter IGES Standard Protocol for Feature Recognition CAD System, pages 25–62. Springer, 2006.

[111] E.A. Nasr and A.K. Kamrani. *Computer Based Design and Manufacturing: An Information-Based Approach*. Springer Science & Business Media, 2006.

[112] A. Al-Ahmari, E. Abouel Nasr, and O. Abdulhameed. *Computer-Aided Inspection Planning: Theory and Practice (Advanced and Additive Manufacturing Series)*, chapter Data Transfer in CAD/CAM Systems. CRC Press, 2016.

[113] M. Bhandarkar, B. Downie, M. Hardwick, and R. Nagi. Migrating from IGES to STEP: one to one translation of IGES drawing to STEP drafting data. *Computers in Industry*, 41(3):261–277, 2000.

[114] *STEP Application Handbook ISO 10303*. SCRA, 5300 International Boulevard, North Charleston, SC 29418, version 3 edition, June 2006.

[115] S. Moskalenko. Modeling of an automatic CAD-based feature recognition and retrieval system for group technology application. Master's thesis, Universität Stuttgart, Institute of Computer-aided Product Development Systems, 2014.

[116] Dominik R., P. Gawrysiak, M. Kryszkiewicz, and H. Rybiński. *Machine Intelligence and Big Data in Industry*. Springer, 2016.

[117] P. Black. DADS: The on-line dictionary of algorithms and data structures. Access: 2017-06-18.

[118] N.C. Jones and P.A. Pevzner. *An Introduction to Bioinformatics Algorithms*. MIT press, 2004.

[119] S. Pandit and S. Gupta. A comparative study on distance measuring approaches for clustering. *International Journal of Research in Computer Science*, 2(1):29–31, 2011.

[120] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

[121] J. Yang, Y. Tian, S. He, J. Wang, Q. Zhou, X. Zhang, Q. Ji, X. Yan, and J.C. Hung. Research on converting CAD model to MCNP model based on STEP file. In *2013 International Joint Conference on Awareness Science and Technology & Ubi-Media Computing (iCAST 2013 & UMEDIA 2013)*, pages 541–547. IEEE, 2013.

[122] T. Yifei, L. Dongbo, L. Changbo, and Y. Minjian. A feature-extraction-based process-planning system. *International Journal of Advanced Manufacturing Technology*, 38(11):1192–1200, 2008.

[123] O. Elazhary. Validation and optimization of a rule-based feature recognition method for CAD design retrieval. Master thesis, Univerität Stuttgart, Institute of Computer-aided Product Development Systems, 2015.

[124] C. Van der Velden, H.-L. Zhang, X. Yu, T. Jones, I. Fieldhouse, and C. Bil. Extracting engineering features from B-Rep geometric models. In *Proceedings of 27th International Congress of the Aeronautical Sciences (ICAS)*, 2010.

[125] L. Zehtaban and D. Roller. Automated rule-based system for Opitz feature recognition and code generation from STEP. *Computer-Aided Design and Applications*, 13(3):309–319, Dec. 2015.

[126] L. Zehtaban and D. Roller. Beyond similarity comparison: Intelligent data retrieval for CAD/CAM designs. *Computer-Aided Design and Applications*, 10(5):789–802, 2013.

[127] S. Opletal, D. Roller, and L. Zehtaban. An architecture for a knowledge augmented integral view across multiple information resources. In *Proceedings of Third International Conference on Advanced Engineering Computing and Applications in Sciences*, pages 130–135, 2009.

[128] D.A. Grossman and O. Frieder. *Information retrieval: Algorithms and heuristics*, volume 15. Springer Science & Business Media, 2004.

[129] S.R. Schach. *Object-oriented and classical software engineering*. Boston: McGraw-Hill Higher Education, 2002.

[130] L. Zehtaban. Development of a coordinator for a computer aided design tool for designing electrical machines and drives. Master thesis, Brunel University, Department of Electronic and Computer Engineering, 2005.

[131] L. Zehtaban, O. Elazhary, and D. Roller. A framework for similarity recognition of CAD models. *Journal of Computational Design and Engineering*, 3(3):274–285, 2016.

[132] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *Proceedings of the International Conference on Shape Modeling Applications*, pages 167–178, 2004.

[133] D. Bespalov, C.Y. Ip, W.C. Regli, and J. Shaffer. Benchmarking CAD search techniques. In *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, pages 275–286, 2005.

[134] J. Frankel, M. Wester, and S. King. Articulatory feature recognition using dynamic bayesian networks. *Computer Speech & Language*, 21(4):620–640, 2007.

[135] A. Liverani and A. Ceruti. Real-time geometric similarity retrieval through fast string matching. In *Proceedings of Congreso International Conjunto XXI Ingegraf XVII ADM*, 2009.

[136] B. Agard and M. Barajas. The use of fuzzy logic in product family development: literature review and opportunities. *Journal of Intelligent Manufacturing*, 23(5):1445–1462, 2012.

[137] T. Ghosh, S. Sengupta, B. Doloi, and P.K. Dan. Al-based techniques in cellular manufacturing systems: a chronological survey and analysis. *International Journal of Industrial and Systems Engineering*, 17(4):449–476, 2014.

[138] P. Jiantao, L. Yi, X. Guyu, Z. Hongbin, L. Weibin, and Y. Uehara. 3D model retrieval based on 2D slice similarity measurements. In *Proceedings of 2nd International Symposium on 3D Data Processing, Visualization and Transmission*, pages 95–101. IEEE, 2004.

[139] J. Stark. *Product Lifecycle Management (Volume 1) 21st Century Paradigm for Product Realisation*. Springer International Publishing, Third edition, 2015.

[140] Y. Liao, M. Lezoche, H. Panetto, N. Boudjlida, and E.R. Loures. Semantic annotation for knowledge explicitation in a product lifecycle management context: A survey. *Computers in Industry*, 71:24–34, 2015.

[141] F. Ameri and D. Dutta. Product lifecycle management: closing the knowledge loops. *Computer-Aided Design and Applications*, 2(5):577–590, 2005.

[142] X. Xu. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1):75–86, 2012.

[143] T. Stentzel, M. Niknam, and J. Ovtcharova. Comparison framework for PLM maturity models. In *IFIP International Conference on Product Lifecycle Management*, pages 355–364. Springer, Berlin, Heidelberg, 2014.

[144] P. Witherell, K.C. Morris, A. Narayanan, J.H. Lee, S. Rachuri, and S. Kumara. Sustainability through lifecycle synthesis of material information. In *Re-engineering Manufacturing for Sustainability*, pages 571–576. Springer, 2013.

[145] S. Rachuri, S. Foufou, and S. Kemmerer. *Analysis of Standards for Lifecycle Management of Systems for US Army–a preliminary investigation*. US Department of Commerce, National Institute of Standards and Technology, 2006.

Appendix A

Classification System to Describe Workpieces

Geometerical Code

1st Digit — Component Class

0	Rotational Components	L/D ≤ 0.5
1		0.5<L/D<3
2		L/D ≥ 3

2nd Digit — External Shape, internal shape elements

0	Smooth, no shape elements		
1	Stepped to one End	No shape elements	
2		With screthread	
3		With functional groove	
4	Stepped to both ends (Multiple increase)	No shape elements	
5		With screthread	
6		With functional groove	
7	Functional taper		
8	Operating thread		
9	Others (>10 functional diameters)		

3rd Digit — External Shape, internal shape elements

0	Without through bore, blind hole	
1	Smooth or Stepped to one End	No shape elements
2		With screwthread
3		With functional groove
4	Stepped to both Ends (Multiple increase)	No shape elements
5		With screwthread
6		With functional groove
7	Functional taper	
8	Operating thread	
9	Others (>10 functional diameters)	

4th Digit — Plane Surface Machining

0	No surface machining
1	External plane surface and/orsurface curved in one direction
2	External plane surface related to one another by graduation around a circle
3	External groove and/or slot
4	External spline and/ or polygon
5	External plane surface and/ or slot and/ or groove, spline
6	Internal plane surface and/or groove
7	Internal spline and/ or polygon
8	External and Internal splines and / or slot and /or groove
9	Others

5th Digit — Auxiliary Hole(s) and Gear Teeth

0	No gear teeth	No auxiliary hole(s)
1		Axial hole(s) not related by a drilling pattern
2		Axial hole(s) not related by a drilling pattern
3		Radial hole(s) not related by a drilling pattern
4		Holes axial and/ or radial and/ or in other directions, not related
5		Holes axial and/ or radial and/ or in other directions related by drilling pattern
6	With gear teeth	Spur gear teeth
7		Bevel gear teeth
8		Other gear teeth
9		Others

Fig. A.1: Opitz table when first digit is 0 or 1 or 2

Geometerical Code

1st Digit — Component Class

	Rotational Components	
3		L/D <=2 With deviation
4		L/D > 2 With deviation

2nd Digit — Overall Shape

0		Hegzagonal bar
1	Around one axis no segments	Square or other regular polygonal section
2		Symmetrical cross-section producing no unbalance
3		Cross-section other than 0 to 2
4		Segments after rotational machining
5		Segments before rotational machining
6	Around more than one axis	Rotational components with curved axis
7		Rotational components with two or more parallel axes
8		Rotational components with intersecting axes
9		Others

3rd Digit — Rotational Machining

0		No rotational machining
1	External shape	machined
2		With scrwthread(s)
3		Smooth
4		Stepped towards one or both ends (multiple increases)
5		With scrwthreads
6		machined
7		screwthread(s)
8		External shape elements
9		Others

4th Digit — Plane Surface Machining

0	No surface machining
1	External plane surface and /or surface curved in one direction
2	External plane surface related to one another by graduation around a circle
3	External groove and / or slot
4	External spline and / or polygon
5	External plane surface and/or slot and/or groove, spline
6	Internal plane surface and / or groove
7	Internal plane surface and / or polygon
8	External and internal spline and/ or slot and/ or groove
9	Others

5th Digit — Auxiliary Hole(s), Gear teeth, Forming

0		No auxiliary holes, gear teeth and forming
1	No forming, No gear teeth	Axial hole(s) not related by drilling pattern
2		Holes axial and / or radial and/ or in other directions, not related
3	drilling pattern	Axial holes
4		Holes axial and/ or radial and/ or in other directions
5	Forming, No gear teeth	Formed, no auxiliary holes
6		Formed, with auxiliary holes
7		Gear teeth, no auxiliary holes
8		Gear teeth, with auxiliary holes
9		Others

Fig. A.2: Opitz table when first digit is 3 or 4

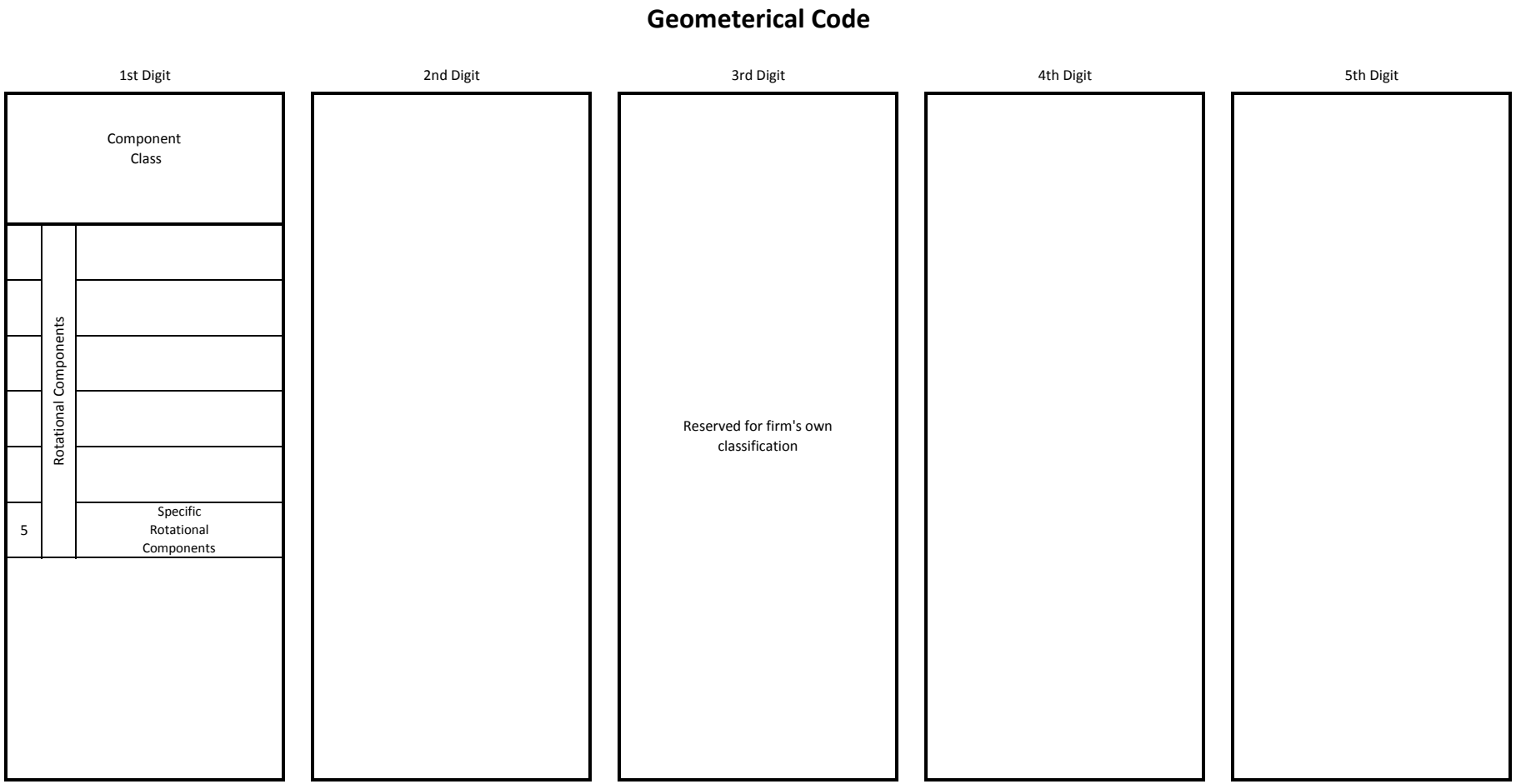


Fig. A.3: Opitz table when first digit is 5

Geometerical Code

1st Digit		2nd Digit			3rd Digit		4th Digit		5th Digit		
Component Class		Overall Shape			Principal bore, rotational surface machining		Plane surface machining		Auxiliary hole(s) forming, gear teeth		
		0	Plane	Rectangular	0	No rotational machining or bore(s)	0	No surface machining	0	No auxiliary holes, gear teeth and forming	
		1	Plane	Rectangular, with one deviation (right angle or triangular)	1	One principal bore, smooth	1	Functional chamfers (e.g. welding prep.)	1	No gear teet, no forming	Holes drilled in one direction only
		2	Plane	Rectangular with angular deviation	2	One principal bore stepped to one or both ends	2	One plane surface	2	No gear teet, no forming	Holes drilled in more than one direction
		3	Plane	Rectangular with circular deviation	3	One principal bore with shape elements	3	Stepped plane surfaces	3	No gear teet, no forming, Drilling pattern	Holes drilled in one direction only
		4	Plane	Any flat shape other than 0 to 3	4	Two principal bores, parallel	4	Stepped plane surfaces at right angles, inclined and/ or opposite	4	No gear teet, no forming, Drilling pattern	Holes drilled in more than one direction
		5	Flat components, rectangular or right angled with small deviation due to casting, welding and forming		5	Several principal bores, parallel	5	Groove and/ or slot	5	Forming, no gear teeth	Formed, no auxiliary holes
6	Flat Components A/B ≤ 3, A/C ≤ 4	6	Flat components, round or of any shape othan than position 5		6	Several principal bores, other than parallel	6	Groove and/ or slot and 4	6	Forming, no gear teeth	Formed, with auxiliary holes
		7	Flat components regularly arched or dished		7	machined annular surfaces, annular grooves	7	Curved surface	7	Gear teeth, no auxiliary hole(s)	
		8	Flat components irregularly arched or dished		8	7 + principal bore(s)	8	Guide surface	8	Gera teeth, with auxiliary hole(s)	
		9	Others		9	Others	9	Others	9	Others	

Fig. A.4: Opitz table when first digit is 6

Geometerical Code

1st Digit: Component Class			2nd Digit: Overall Shape			3rd Digit: Principal bore, rotational surface machining	4th Digit: Plane surface machining	5th Digit: Auxiliary hole(s) forming, gear teeth		
			0	Shape Axis- Straight	Uniform Cross-Section: Rectangular	0: No rotational machining or bore(s)	0: No surface machining	0	No auxiliary holes, gear teeth and forming	
			1	Shape Axis- Straight	Uniform Cross-Section: Rectangular with one deviation(right angle or triangle)	1: One principal bore, smooth	1: Functional chamfers (e.g. welding prep.)	1	No gear teet, no forming	Holes drilled in one direction only
			2	Shape Axis- Straight	Uniform Cross-Section: Any cross-section other than 0 and 1	2: One principal bore stepped to one or both ends	2: One plane surface	2	No gear teet, no forming	Holes drilled in more than one direction
			3	Shape Axis- Straight	Varying Cross-Section: Rectangular	3: One principal bore with shape elements	3: Stepped plane surfaces	3	No gear teet, no forming / Drilling pattern	Holes drilled in one direction only
			4	Shape Axis- Straight	Varying Cross-Section: Rectangular with one deviation (right angle or triangle)	4: Two principal bores, parallel	4: Stepped plane surfaces at right angles, inclined and/ or opposite	4	No gear teet, no forming / Drilling pattern	Holes drilled in more than one direction
			5	Shape Axis- Straight	Varying Cross-Section: Any cross-section other than 3 and 4	5: Several principal bores, parallel	5: Groove and/ or slot	5	Forming, no gear teeth	Formed, no auxiliary holes
6	Non-rotational Components		6	shape Axis Curved (bent)	Rectangular, angular and other cross-sections	6: Several principal bores, other than parallel	6: Groove and/ or slot and 4	6	Forming, no gear teeth	Formed, with auxiliary holes
7	Non-rotational Components	Long components A/B > 3	7	shape Axis Curved (bent)	Formed components	7: machined annular surfaces, annular grooves	7: Curved surface	7	Gear teeth, no auxiliary hole(s)	
			8	shape Axis Curved (bent)	Formed component with deviations in the main axis	8: 7 + principal bore(s)	8: Guide surface	8	Gera teeth, with auxiliary hole(s)	
			9	shape Axis Curved (bent)	Others	9: Others	9: Others	9	Others	

Fig. A.5: Opitz table when first digit is 7

Geometerical Code

1st Digit: Component Class		
6	Non-rotational Components	
7		
8		Cubic components A/B ≤ 3, A/C<4

2nd Digit: Overall Shape			
0	Block and Block-like Components		Rectangular prism
1			Rectangular with deviations (right angle or triangular)
2			Componded of rectangular prisms
3			Componets with mounting or locating surface and principal bore
4			Componets with mounting or locating surface with dividing surface
5			Components other than 6
6	Box and Box-like Components	Not split	Approximate or compounded of rectangular prisms
7			Components other than 6
8		Split	Approximate or compounded of rectangular prisms
9			Others

3rd Digit: Principal bore, rotational surface machining	
0	No rotational machining or bore(s)
1	One principal bore, smooth
2	One principal bore stepped to one or both ends
3	One principal bore with shape elements
4	Two principal bores, parallel
5	Several principal bores, parallel
6	Several principal bores, other than parallel
7	machined annular surfaces, annular grooves
8	7 + principal bore(s)
9	Others

4th Digit: Plane surface machining	
0	No surface machining
1	Functional chamfers (e.g. welding prep.)
2	One plane surface
3	Stepped plane surfaces
4	Stepped plane surfaces at right angles, inclined and/ or opposite
5	Groove and/ or slot
6	Groove and/ or slot and 4
7	Curved surface
8	Guide surface
9	Others

5th Digit: Auxiliary hole(s) forming, gear teeth			
0	No auxiliary holes, gear teeth and forming		
1	No gear teet, no forming		Holes drilled in one direction only
2			Holes drilled in more than one direction
3		Drilling pattern	Holes drilled in one direction only
4			Holes drilled in more than one direction
5	Forming, no gear teeth		Formed, no auxiliary holes
6			Formed, with auxiliary holes
7	Gear teeth, no auxiliary hole(s)		
8	Gera teeth, with auxiliary hole(s)		
9	Others		

Fig. A.6: Opitz table when first digit is 8

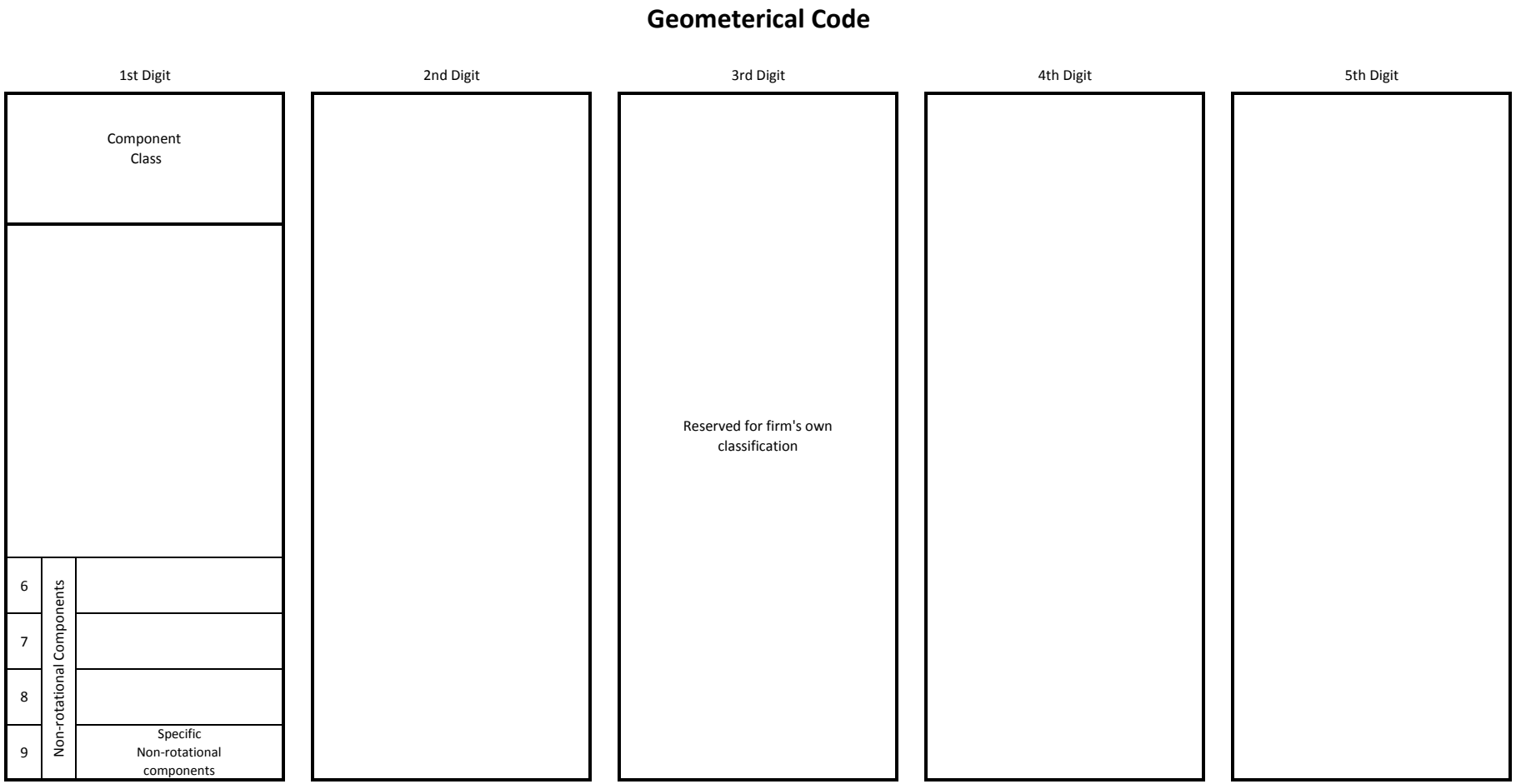


Fig. A.7: Opitz table when first digit is 9

Supplementary Codes

1st Digit

DIAMETER 'D' or EDGE LENGTH 'A'		
0	MM's ≤ 20	Inches ≤ .8
1	> 20 ≤50	> 0.8 ≤ 2.0
2	> 50 ≤ 100	> 2.0 ≤ 4.0
3	> 100 ≤ 160	> 4.0 ≤ 6.5
4	> 160 ≤ 250	> 6.5 ≤ 10.0
5	> 250 ≤ 400	> 10.0 ≤ 16.0
6	> 400 ≤ 600	> 16.0 ≤ 25.0
7	> 600 ≤ 1000	> 25.0 ≤ 40.0
8	> 1000 ≤ 2000	> 40 ≤ 80.0
9	> 2000	> 80.0

2nd Digit

Material	
0	Cast iron
1	ar graphitic castiron and malleable cas
2	Steel ≤ 26.5 tonf/in² not heat treated
3	Steel> 26.5 tonf /in² heat treatable low carbon and case hardening steel, not heat treated
4	Steels 2 and 3 heat treated
5	Alloy steel (not heat treated)
6	Alloy steel heat treated
7	Non-ferrous metal
8	Light alloy
9	Other material

3rd Digit

Initial Form	
0	Round bar, block
1	Round bar, bright drwan
2	Bar-triangular, square, hexagonal, others
3	Tubing
4	Angle, U-, T-, and similar sections
5	Sheet
6	plate and slabs
7	Cast or forged components
8	Welded assembly
9	Pre-machined components

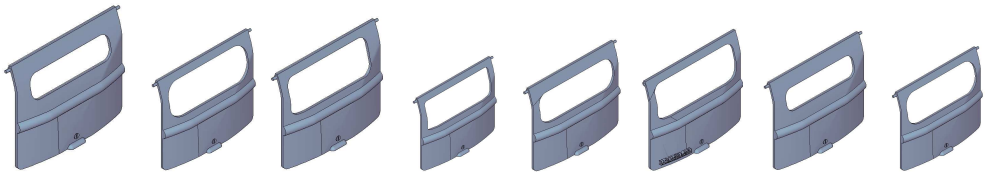
4th Digit

Accuracy in Coding Digit	
0	No accuracy specified
1	2
2	3
3	4
4	5
5	2 and 3
6	2 and 4
7	2 and 5
8	3 and 4
9	(2 + 3 + 4 + 5)

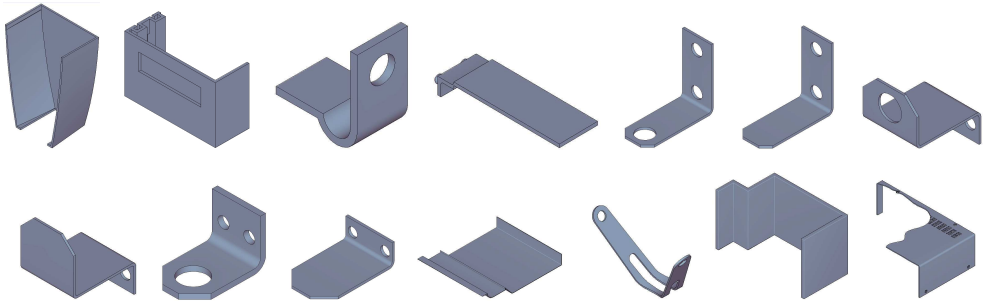
Fig. A.8: Opitz Supplementary Codes

Appendix B

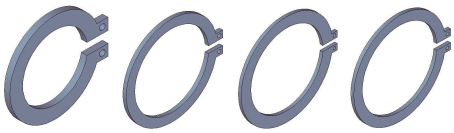
Engineering Shape Benchmark (ESB)



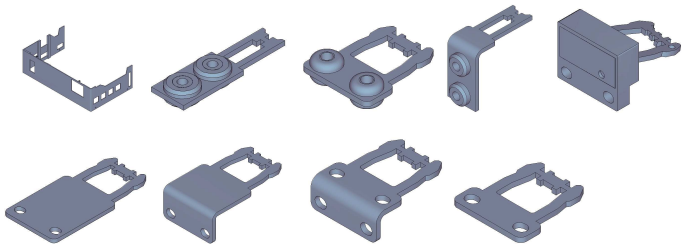
Flat-Thin Wallcomponents $\longmapsto$ *Back Doors*



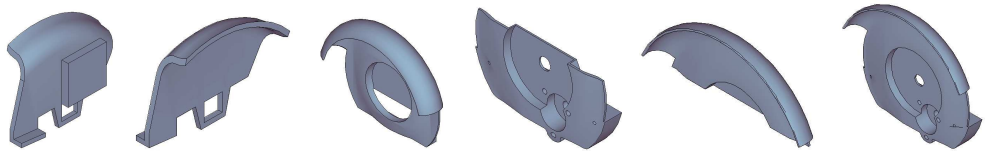
Flat-Thin Wallcomponents $\longmapsto$ *Bracket like Parts*



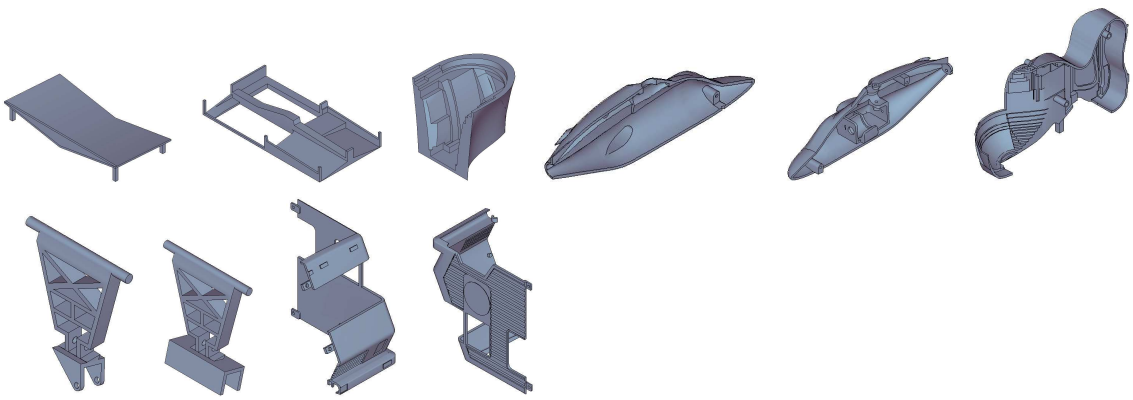
Flat-Thin Wallcomponents $\longmapsto$ *Clips*



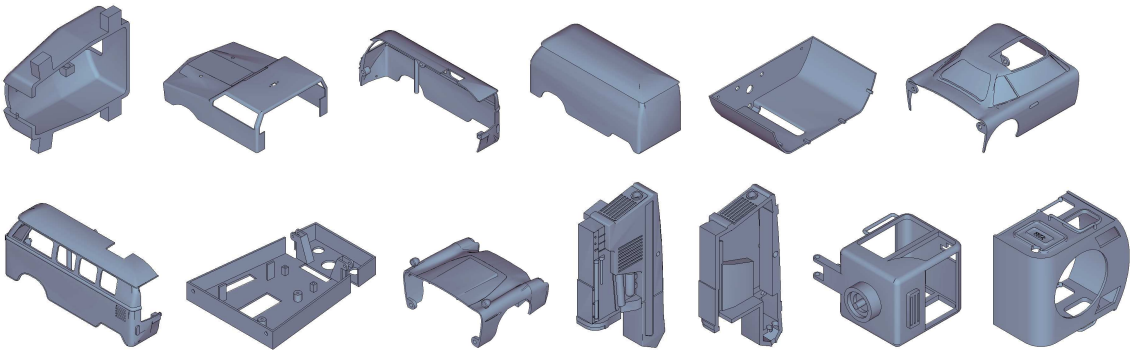
Flat-Thin Wallcomponents ⟼ Contact Switches



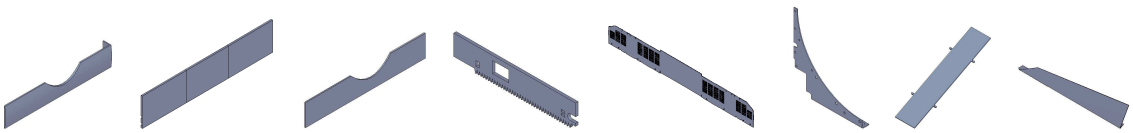
Flat-Thin Wallcomponents ⟼ Curved Housings



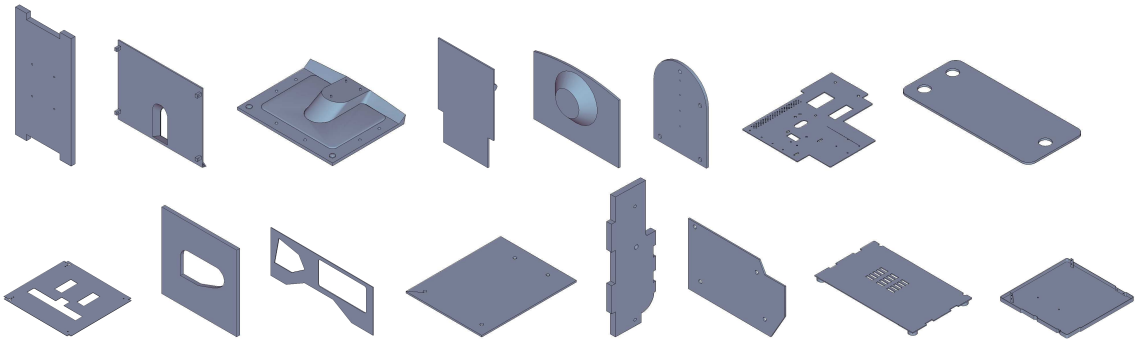
Flat-Thin Wallcomponents ⟼ Miscellaneous



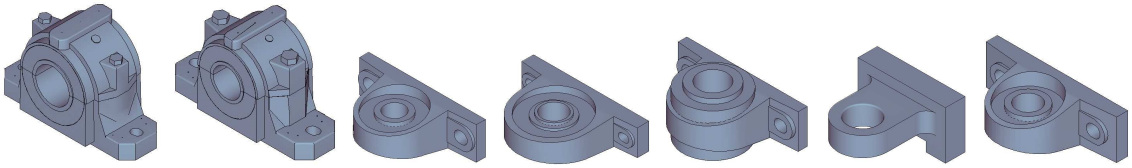
Flat-Thin Wallcomponents ⟼ Rectangular Housings



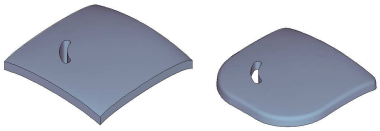
Flat-Thin Wallcomponents ⟼ Slender Thin Plates



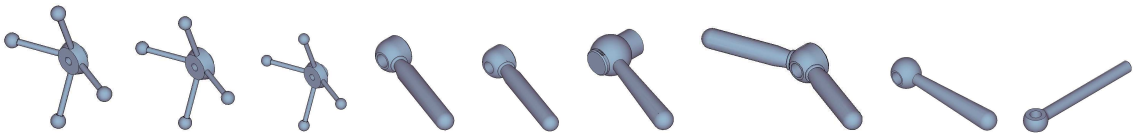
Flat-Thin Wallcomponents $\longmapsto$ *Thin Plates*



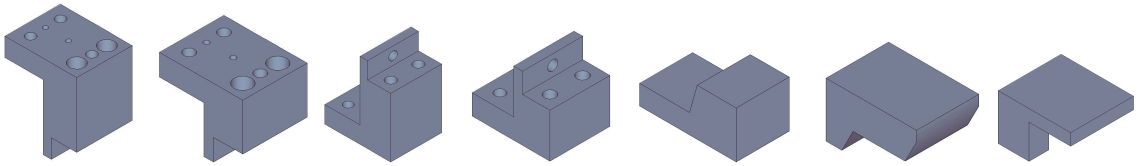
Rectangular-Cubic Prism $\longmapsto$ *Bearing Blocks*



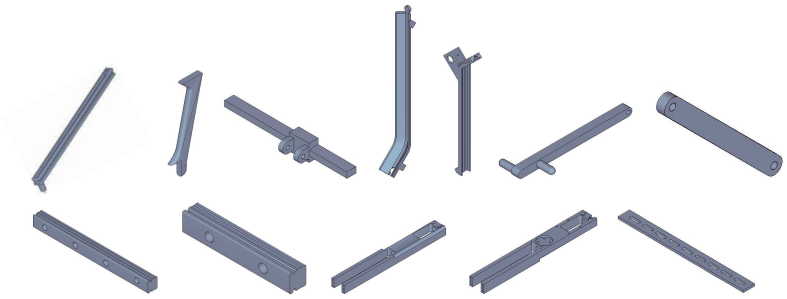
Rectangular-Cubic Prism $\longmapsto$ *Contoured Surfaces*



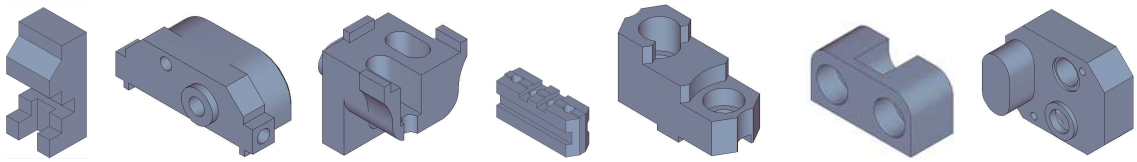
Rectangular-Cubic Prism $\longmapsto$ *Handles*



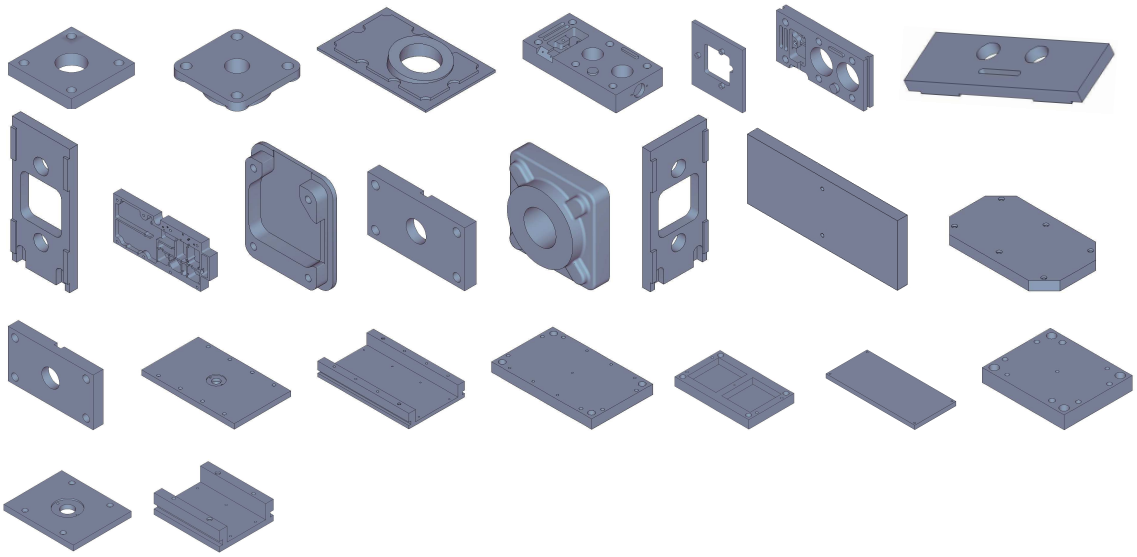
Rectangular-Cubic Prism $\longmapsto$ *L Blocks*



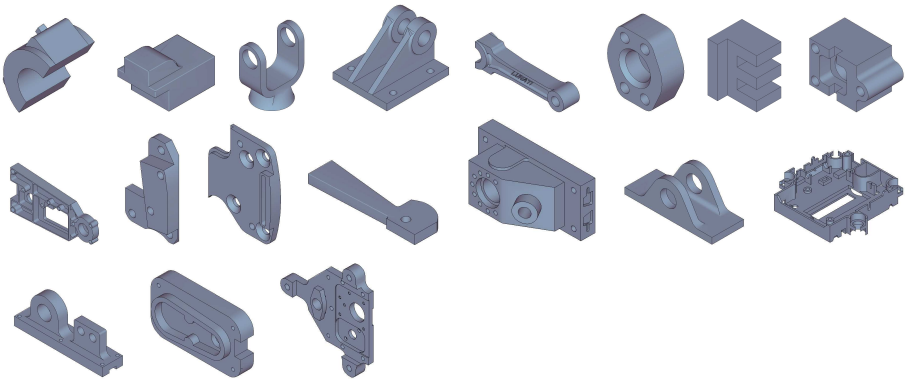
Rectangular-Cubic Prism $\longmapsto$ *Long Machine Elements*



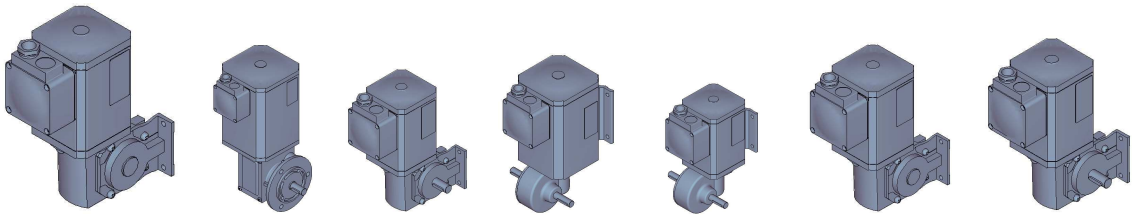
Rectangular-Cubic Prism ⟼ Machined Blocks



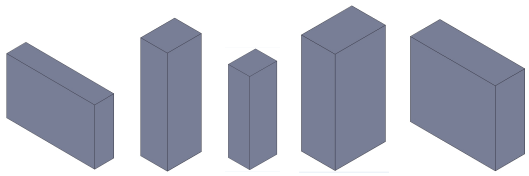
Rectangular-Cubic Prism ⟼ Machined Plates



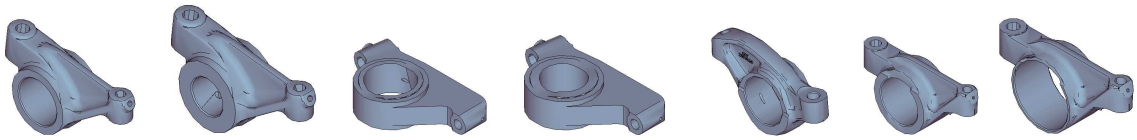
Rectangular-Cubic Prism ⟼ Miscellaneous



Rectangular-Cubic Prism ⟼ Motor Bodies

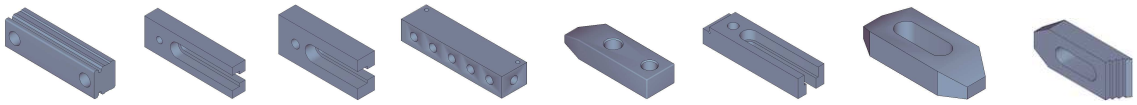


Rectangular-Cubic Prism $\longmapsto$ *Prismatic Stock*

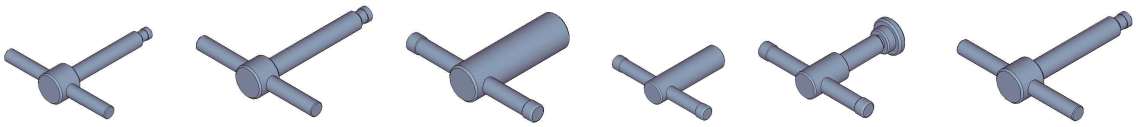


Rectangular-Cubic Prism $\longmapsto$ *Rocker Arms*

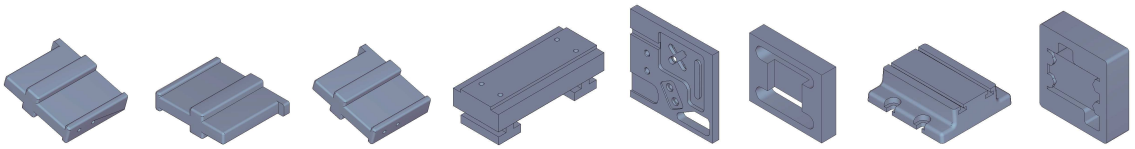
Rectangular-Cubic Prism $\longmapsto$ *Slender Links*



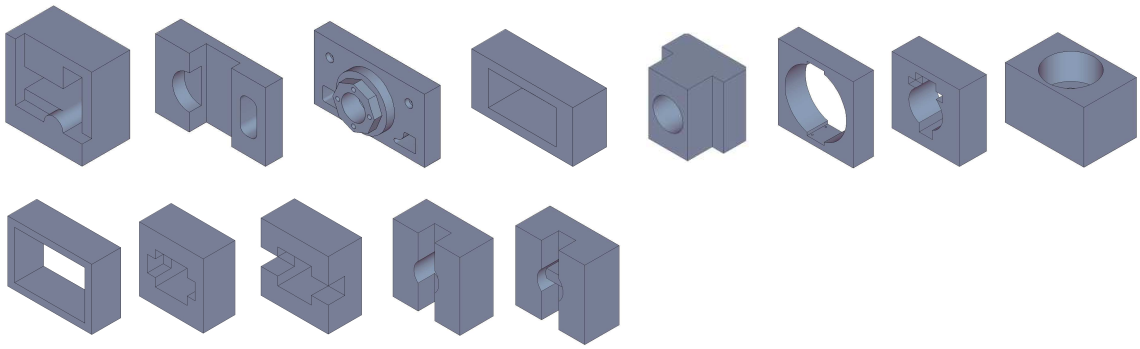
Rectangular-Cubic Prism $\longmapsto$ *Small Machined Blocks*



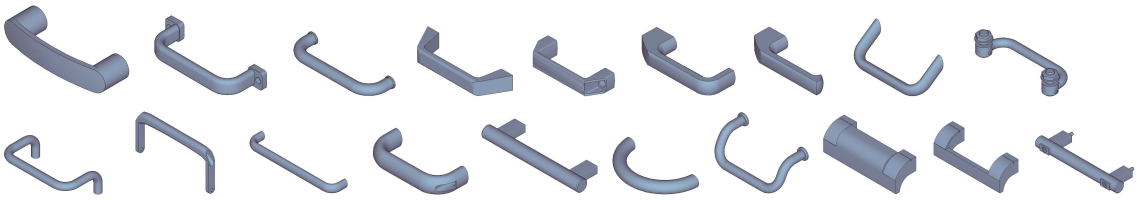
Rectangular-Cubic Prism $\longmapsto$ *T shaped parts*



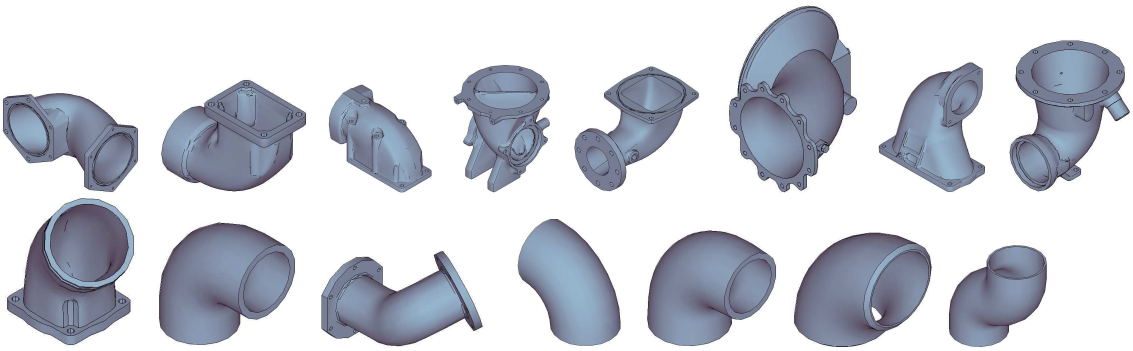
Rectangular-Cubic Prism $\longmapsto$ *Thick Plates*



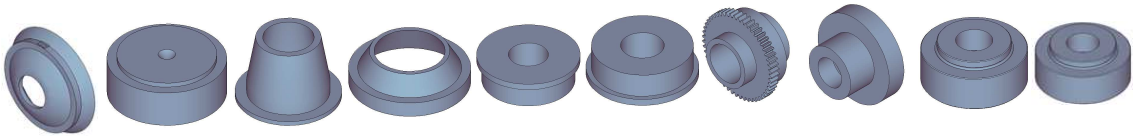
Rectangular-Cubic Prism $\longmapsto$ *Thick Slotted plates*



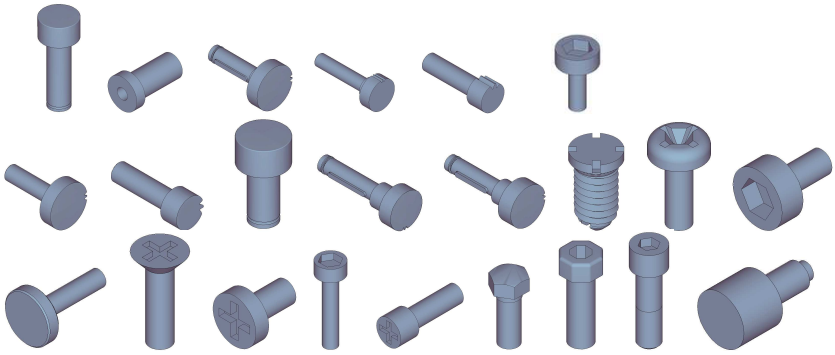
Rectangular-Cubic Prism ⟼ U shaped parts



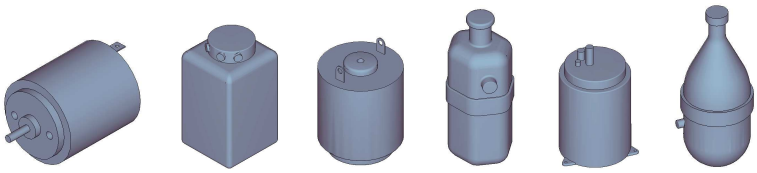
Solid Of Revolution ⟼ 90 degree elbows



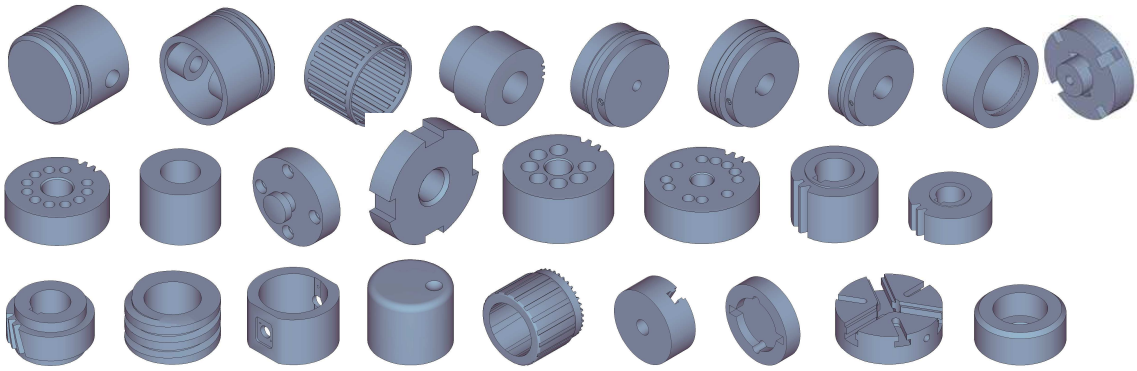
Solid Of Revolution ⟼ Bearing Like Parts



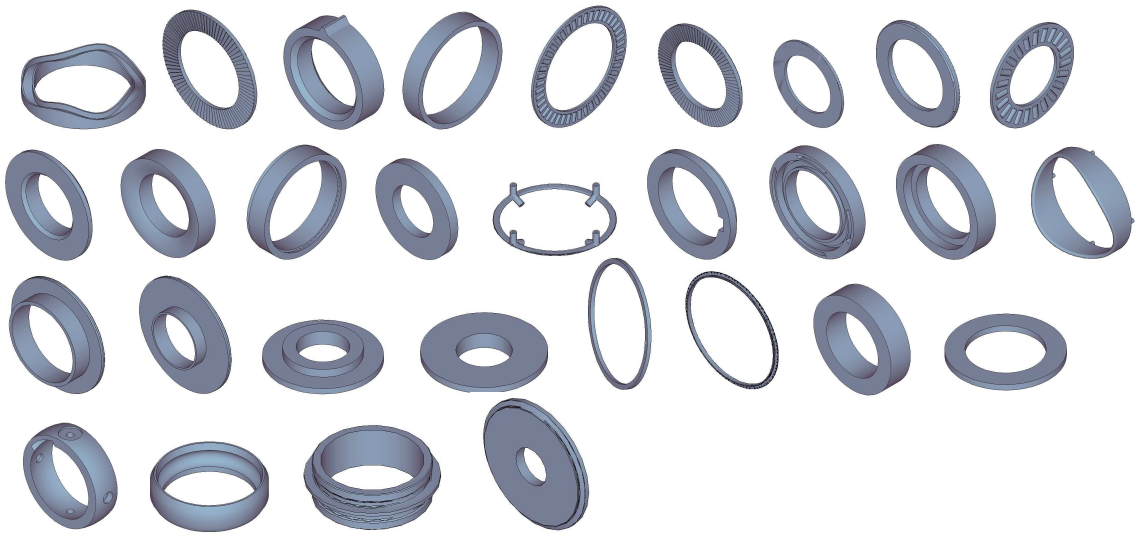
Solid Of Revolution ⟼ Bolt Like Parts



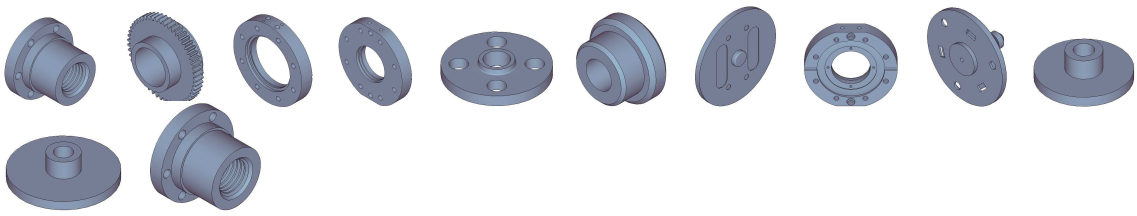
Solid Of Revolution ⟼ Container Like Parts



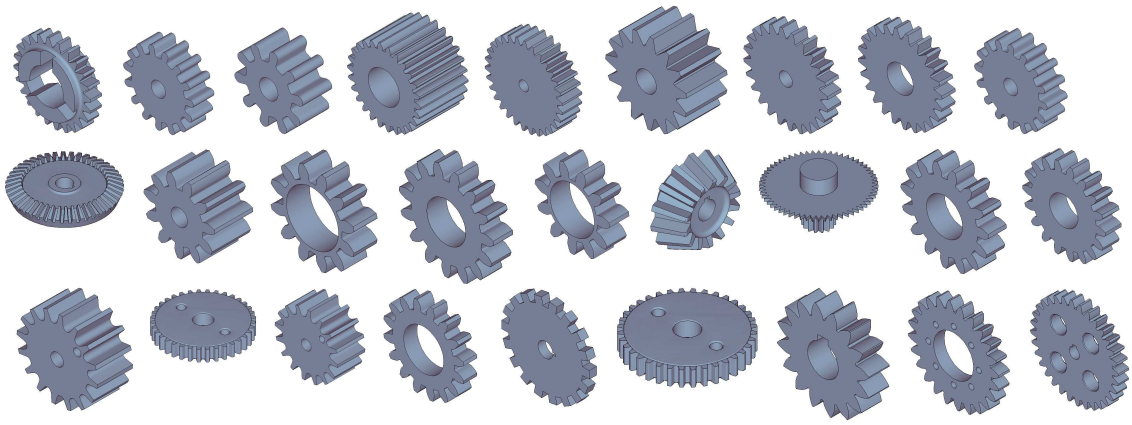
Solid Of Revolution $\longmapsto$ *Cylindrical Parts*



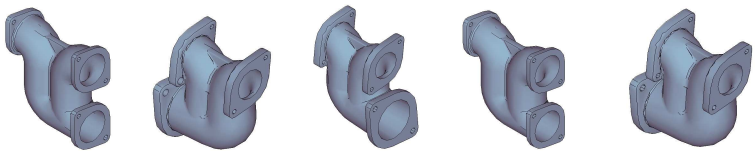
Solid Of Revolution $\longmapsto$ *Discs*



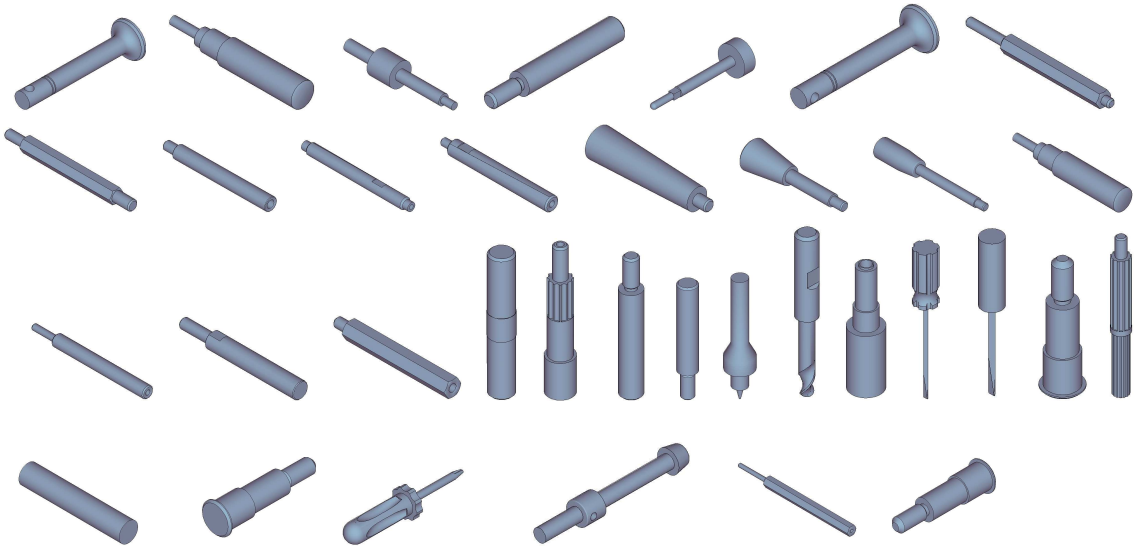
Solid Of Revolution $\longmapsto$ *Flange Like Parts*



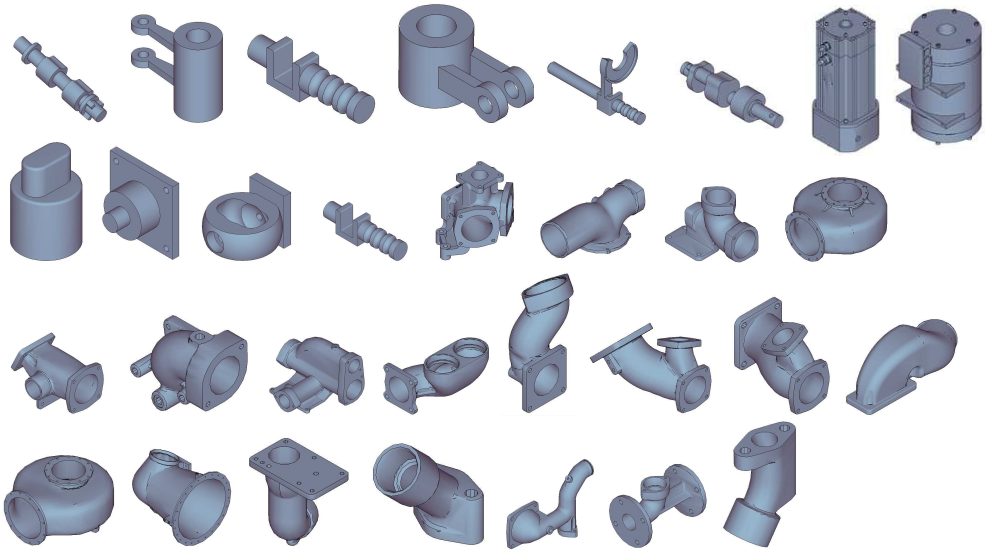
Solid Of Revolution $\longmapsto$ *Gear like Parts*



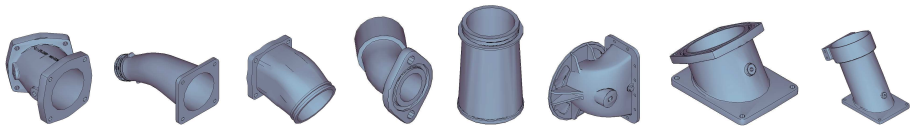
Solid Of Revolution $\longmapsto$ *Intersecting Pipes*



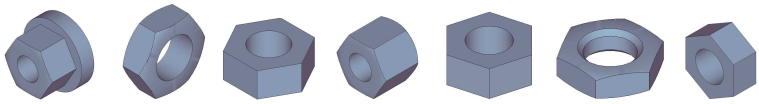
Solid Of Revolution $\longmapsto$ *Long Pins*



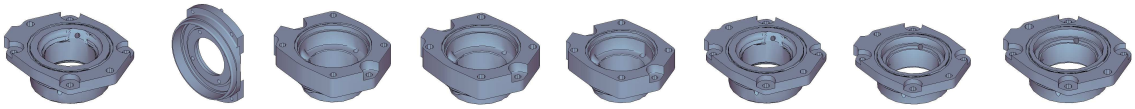
Solid Of Revolution $\longmapsto$ *Miscellaneous*



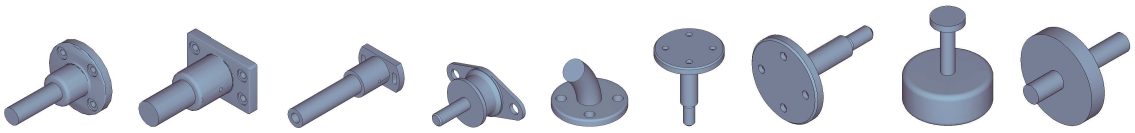
Solid Of Revolution $\longmapsto$ *Non-90 degree elbows*



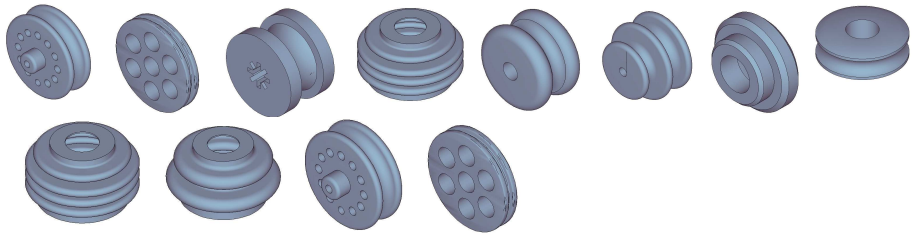
Solid Of Revolution $\longmapsto$ *Nuts*



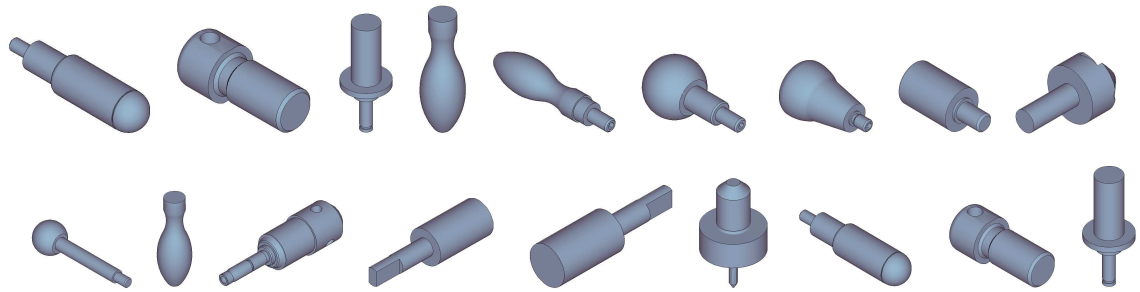
Solid Of Revolution $\longmapsto$ *Oil Pans*



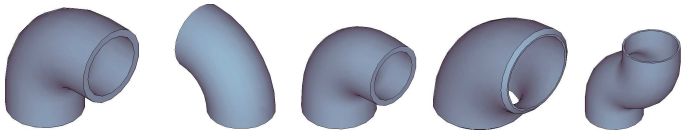
Solid Of Revolution $\longmapsto$ *Posts*



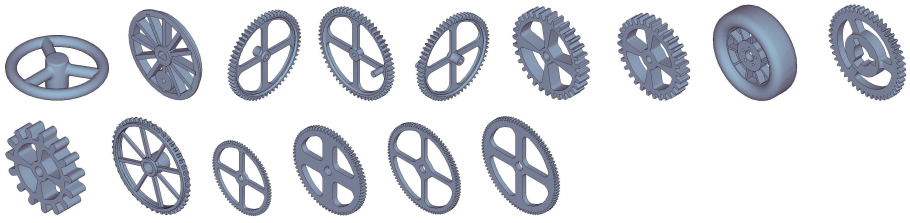
Solid Of Revolution $\longmapsto$ *Pulley Like Parts*



Solid Of Revolution ⟼ Round Change At End



Solid Of Revolution ⟼ Simple Pipes



Solid Of Revolution ⟼ Spoked Wheels