UNIVERSITY OF CINCINNATI

DATE: December 5, 2002

I, Sushilendra Deshpande, hereby submit this as part of the requirements for the degree of:

Master of Science
in:
Mechanical Engineering

It is entitled:
A Sheet Metal Feature Extraction and Design Advisory System

Approved by:
Dr. Jay H. Kim
Dr. Sam Anand
Dr. David F. Thompson
FEATURE EXTRACTION AND INTRA-FEATURE DESIGN ADVISOR FOR SHEET METAL PARTS

A thesis submitted to the

Division of Research and Advanced Studies of the University of Cincinnati

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in the Department of Mechanical, Industrial and Nuclear Engineering of the College of Engineering

2003

by

Sushilendra A Deshpande

Bachelor of Engineering (B.E.) University of Mumbai, Mumbai, India, 1999

Committee Chair: Dr. Sam Anand
ABSTRACT

The design of sheet metal parts and the dies used to manufacture them is an art perfected through decades of experience. Reduction of costs is also increasingly important and lack of this experience is a handicap in designing a part with the lowest achievable cost. Parts are frequently redesigned after they are deemed as expensive or infeasible to manufacture. Products can be brought to market faster if the designer follows certain recommended guidelines without violating any design rules.

The present work aims at developing a feature extraction module and intra-feature design advisor for reducing infeasible designs, costs and production cycle times. The proposed design advisory system will aid the designer right at the design stage with useful design and manufacturability recommendations that have been gathered through decades of experience. These rules are incorporated here and implemented for SolidWorks 2000, which has a separate sheet metal modeling module. The implementation has been done in Visual Basic using the OLE Interface provided by SolidWorks API.
I thank my thesis advisor Dr. Sam Anand for his invaluable guidance and advice throughout my graduate studies. I also wish to thank him for allowing me to work in a healthy research environment that has given me a well-rounded educational experience.

I would also sincerely like to thank the committee members, Dr. Jay Kim, who is my academic advisor, Dr. David Thompson for their time and valuable suggestions.

Thanks are due to my friends in the CAM lab and my project mate Shashikiran Hegde, with whom I shared many interesting and rewarding discussions during both algorithm and software development stages of the project.

I finally thank the University of Cincinnati for providing me the opportunity to pursue my dreams, do research and for providing financial assistance.
Contents

Abstract

Acknowledgements

Contents

List of Figures and Tables.................................................................viii

1 Introduction ..................................................................................... 1
  1.1 Background and Need for Research ............................................. 1
  1.2 Objectives of Research .............................................................. 2
  1.3 Outline of Thesis ...................................................................... 2

2 Literature Review ........................................................................... 3
  2.1 Feature Extraction / Feature Based Design .................................... 3
  2.2 Cost Estimation of Sheet metal parts ........................................... 6
  2.3 Process Planning ...................................................................... 6
  2.4 Design And Manufacturability Analysis ....................................... 8

3 Design Rules .................................................................................. 10
  Intra-feature Vs. Inter-feature .......................................................... 11
  3.1 Intra-feature rules ................................................................. 12
    3.1.1 Holes ............................................................................... 12
    3.1.2 Notches .......................................................................... 13
    3.1.3 Lanced lugs ...................................................................... 13
    3.1.4 Triangular tabs or slots ...................................................... 13
    3.1.5 Beads ............................................................................ 14
    3.1.6 Bridges ......................................................................... 15
    3.1.7 Curls ............................................................................. 15
    3.1.8 Bends ............................................................................. 15
    3.1.9 Tapped Holes ................................................................. 16
  3.2 Inter-feature Rules ................................................................... 16

4 Feature Extraction and Design Advisor for Intra-Feature Rules........... 21
  4.1 Internal Feature Extraction ....................................................... 22
  4.2 General Methodology of feature extraction .................................. 23
  4.3 Circular Feature .................................................................... 24
    4.3.1 Feature Extraction ......................................................... 24
    4.3.2 Feature Information ....................................................... 24

v
# Table of Contents

4.15.2 Feature Extraction ................................................................. 51
4.15.3 Feature Information ............................................................. 51
4.15.4 Methodology for calculating feature parameters: .................... 53
4.15.5 Design Rules for validating the feature parameters: ................. 54
4.15.6 Example showing feature extraction and design validation for the part: 54

4.16 Custom Feature ........................................................................ 55
4.16.1 Feature Pattern String .......................................................... 55
4.16.2 Feature Extraction ................................................................. 55
4.16.3 Feature Information ............................................................... 55
4.16.4 Methodology for calculating feature parameters: ..................... 56
4.16.5 Design Rules for validating the feature parameters: ................. 56
4.16.6 Example of feature extraction and design validation for the feature: 57

4.17 Data structures used ................................................................. 58
4.17.1 StartPointArray, EndPointArray and Edge Orientation process .... 58
4.17.2 NewEdgeLocationArray ......................................................... 63

4.18 Incremental Feature Filling Technique ........................................... 65

4.19 External Features Classification Algorithm .................................... 73

4.20 Overall Feature Extraction Module in Detail ................................... 75

4.21 Extracting of Bend Features from the CAD model ......................... 76
4.21.1 Base-Bends ........................................................................ 76
4.21.2 Edge Bends ...................................................................... 76
4.21.3 Sketched Bends ................................................................. 76
4.21.4 Miter Bends .................................................................... 76
4.21.5 Design Advisor for Bend Features ......................................... 77

4.22 Getting the Total Bend Length of the part .................................... 78

4.23 Other Information about the sheet metal part .................................. 79

4.24 Inter-Feature rules ................................................................... 81

5 Conclusions and Directions for future research ................................ 84
5.1 Conclusions ............................................................................. 84
5.1.1 Comprehensive collection of rules ........................................... 84
5.1.2 Reduced Cycle Time for bringing products to market .................. 84
5.1.3 Customizability .................................................................. 85
5.1.4 Defect-Free Cost-Effective Method ......................................... 85
5.1.5 Scope for integrating with other modules .................................. 85

5.2 Directions for Future Research .................................................. 86

References .................................................................................... 88

Appendix ......................................................................................... 90
List of Figures

Figure 2.1 Schema of feature reasoning process ................................................................. 3
Figure 2.2 An integrated CAE system with deformation analysis and knowledge based system for optimal design of sheet metal parts ................................................................. 4
Figure 2.3 BendCad System ................................................................................................. 7
Figure 2.4 Overview of Promod-S ...................................................................................... 8
Figure 2.5 Assumption: Constant thickness of the part ....................................................... 8
Figure 2.6 Voronoi Diagram for a sample part ..................................................................... 9
Figure 3.1 Diameter/Thickness design rule[14] .................................................................. 12
Figure 3.2 Length/Width design rule[14] ......................................................................... 13
Figure 3.3. Lanced Lugs[14] ............................................................................................. 13
Figure 3.4 Triangular Tabs[14] .......................................................................................... 14
Figure 3.5 Stock Radius[14] ............................................................................................ 14
Figure 3.6 Stock Relief Angle[14] ..................................................................................... 14
Figure 3.7 Direction of rolling[15] ..................................................................................... 15
Figure 3.8 Tapped Holes[14] ............................................................................................ 16
Figure 3.9 Distance between two holes[14] ................................................................. 17
Figure 3.10 Distance of the hole from the edge[16] ......................................................... 17
Figure 3.11 Distance of the hole from another feature[14] ............................................. 17
Figure 3.12 Distance of the hole from the edge[14] ......................................................... 17
Figure 3.13 Distance between adjacent tabs or slots[14] ............................................. 18
Figure 3.14 Distance of the hole from the edge[14] ......................................................... 18
Figure 3.15 Spacing for extruded holes[14] ..................................................................... 18
Figure 3.16. Spacing for threaded holes[14] ..................................................................... 19
Figure 3.17. Aligned Holes on opposite bends[14] ....................................................... 20
Figure 4.1 Schematic View of Feature Extraction and Design Advisory System [13] .... 22
Figure 4.2 Standard Library for Internal Stamped Features ............................................ 23
Figure 4.3 Circular Feature Sketch .................................................................................. 24
Figure 4.4 Feature Extraction: Circular Feature .............................................................. 25
Figure 4.5 Rectangle Feature Sketch ................................................................................. 26
Figure 4.6 Feature Extraction: Rectangular Feature ......................................................... 27
Figure 4.7 Single-D Feature Sketch .................................................................................. 28
Figure 4.8 Feature Extraction: Single-D Feature ............................................................. 29
Figure 4.9 Double-D Feature Sketch ................................................................................ 30
Figure 4.10 Feature Extraction: Double-D Feature ............................................................ 32
Figure 4.11 Oblong Feature Sketch ................................................................................... 33
Figure 4.12 Feature Extraction: Oblong Feature ............................................................... 34
Figure 4.13 Flowchart for Internal Stamped Feature Classification .................................. 35
Figure 4.14. Standard External Features .......................................................................... 36
Figure 4.15 Figure showing the convexity between edges ............................................ 37
Figure 4.16 Figure showing the feature pattern values for the edges based on the convexity .................................................................................................................. 38
Figure 4.17 Figure showing feature pattern methodology where one of the edges is an arc ................................................................. 38
Figure 4.18 Figure showing a snap shot of a sample sheet metal part with external features and its feature string pattern .......................... 39
Figure 4.19 Arc Notch Feature Sketch ......................................................... 40
Figure 4.20 Arc Notch Feature Sketch Details .................................................. 41
Figure 4.21 Arc Notch Feature Extraction ........................................................... 42
Figure 4.22: Figure showing the feature pattern for a V-Notch feature ............... 43
Figure 4.23 Figure showing geometry of V-Notch feature for case(a) ................. 44
Figure 4.24 Figure showing geometry of V-Notch feature for case(b) ............... 44
Figure 4.25 V-Notch Feature Extraction ............................................................ 46
Figure 4.26 Feature Pattern Straight U-Notch Feature ....................................... 47
Figure 4.27 Straight U-Notch Feature Details for case(a) .................................. 48
Figure 4.28 Straight U-Notch Feature Details for case(b) .................................. 48
Figure 4.28 Straight U-Notch Feature Extraction ............................................. 50
Figure 4.29 Arc U-Notch Feature Pattern .......................................................... 51
Figure 4.30 Arc U-Notch Feature Sketch Details for case(a) .............................. 52
Figure 4.31 Arc U-Notch Feature Sketch Details for case(b) .............................. 52
Figure 4.32 Arc U-Notch Feature Extraction ..................................................... 54
Figure 4.33 Custom Feature Pattern ................................................................. 55
Figure 4.34 Custom Feature Sketch Details ...................................................... 56
Figure 4.35 Custom Feature Extraction ............................................................ 57
Figure 4.36 Orientation of edges - 1 ................................................................. 58
Figure 4.37 Orientation of edges - 2 ................................................................. 59
Figure 4.38 Orientation of edges - 3 ................................................................. 60
Figure 4.39 Orientation of edges - 4 ................................................................. 60
Figure 4.40 Orientation of edges - 5 ................................................................. 60
Figure 4.41 Re-Orienting the edges - 1 ............................................................ 61
Figure 4.42 Re-Orienting the edges - 2 ............................................................ 62
Figure 4.43 Re-Orienting the edges - 3 ............................................................ 62
Figure 4.44 Re-Orienting the edges - 3 ............................................................ 63
Figure 4.45 Feature String Pattern for a sample sheet metal part ....................... 64
Figure 4.46 Incremental Feature Filling Explained for a sample part - 1 .............. 65
Figure 4.46 Incremental Feature Filling Explained for a sample part - 2 .............. 68
Figure 4.47 Incremental Feature Filling Explained for a sample part - 3 .............. 68
Figure 4.48 Incremental Feature Filling Explained for a sample part - 4 .............. 69
Figure 4.49 Incremental Feature Filling Explained for another part - 1 ............... 70
Figure 4.50 Incremental Feature Filling Explained for another part - 2 ............... 71
Figure 4.51 Incremental Feature Filling Explained for another part - 3 ............... 71
Figure 4.52 Incremental Feature Filling Explained for another part - 4 ............... 71
Figure 4.53 Incremental Feature Filling Explained for another part - 5 ............... 72
Figure 4.54 Incremental Feature Filling: Part Reconstructed ............................ 73
Figure 4.55 Flowchart for External Features Classification ............................... 74
Figure 4.56 Flowchart for External Feature Extraction ...................................... 75
Figure 4.57 Bend Feature Extraction ............................................................... 77
Figure 4.58 Total Bend Length of the Sheet metal part ...................................... 78
Figure 4.59 Extraction of Base Face and bounding box Details .......................... 80
1 Introduction

In today’s market several products are made with sheet metal components. These products can be simple components requiring simple fabrication steps while others may involve very complicated processes requiring several steps to fabricate. For the complicated ones, the time to market from conceptualization stage can be significantly shortened at various levels from design to manufacturing by understanding the product, its features and implications (functional) of the features [4].

It is widely accepted that about 70 percent of the final product cost is determined during the design stage [1]. Systems that automatically provide inputs to designers during the initial design process on manufacturing guidelines that lower costs and reduce cycle times have proven highly effective in achieving their goals. This chapter discusses the need for the current research and the outline of the thesis.

1.1 Background and Need for Research

The sheet metal industry is largely an empirical, experience-oriented industry. The human involvement necessary in the design and process planning of sheet metal products is abundantly high when compared to the advances made in other areas of manufacturing. The design of complex parts is based heavily on human experience and some parts might be prohibitively expensive to manufacture as initially designed due to deviation from recommended practices.

To prevent modification attempts of the initial design from becoming a trial and error process, some knowledge-based assistance is necessary. If the traditional knowledge gained from decades of experience is integrated into a solid modeler, it will provide valuable inputs to the designer. The designer can then attempt changes while adhering to the functional requirements of the design for achieving easier manufacture and reduced costs at the same time.

Though there have been attempts ([2], [3], [4]) to create a solution for these problems, most of them have been design rule checkers using a few design rules. Also, these
systems only checked for violations of certain constraints and did not suggest recommended practices where appropriate. A comprehensive design advisory system has not been attempted yet. The comprehensive approach should successfully encompass the design rules, standard manufacturing practices and design recommendations across all sheet metal processes and also investigate the interaction between these elements in the creation of a final sheet metal part.

1.2 Objectives of Research

This research aims at building a comprehensive feature extraction module and design advisory system to address the lack of manufacturing experience among designers and advice them on recommendations that would help reduce costs and ease manufacture where appropriate. Another aspect of this system is to verify if any design rules have been violated. These design rules are based on the failure of the metal due to its mechanical limitations of stress and strain or process capability limitations of the process generally used.

1.3 Outline of Thesis

The introduction is followed in Chapter 2 by a literature review on other work done of feature extraction and design advisory systems for sheet metal parts. Chapter 3 describes the development of the manufacturability rules, which have been collected from various literatures available on sheet metal design such as die design handbooks. Techniques for applying these design rules in a solid modeling system are also presented. This is followed by the methodology and implementation details of feature extractor and design advisor systems. Finally, the conclusions and directions for future work are described in Chapter 6.
2 Literature Review

There have been various attempts at developing feature based sheet metal modelers, design checkers and various sheet metal design and manufacturing process automations. Standard CAD packages used in the industry such as SolidWorks, Pro-Engineer come equipped with a separate sheet metal design module that aids specifically in designing sheet metal parts. The following section describes the existing research in sheet metal design and manufacturability analysis of parts.

2.1 Feature Extraction / Feature Based Design

Nnaji et al. [4], have developed a set of principles for extracting features from sheet metal parts. In order to make the system generalized, the system they have developed requires B-Rep information about the sheet metal part in IGES neutral file format. The B-Rep information stored in IGES contains only geometric information and lacks topological information. The feature reasoning logic is shown in the figure 2.1 below. The B-Rep information is first checked for uniqueness of information, all the face normals are calculated and bounding loops and internal loops for all the faces are extracted and calculated. Once modification of boundary information is done, edge-grouping process is performed. The contour face is obtained, which is assumed to be the base face. Then all the internal loops present in this face are found.

![Figure 2.1 Schema of feature reasoning process](image)

Figure 2.1 Schema of feature reasoning process
The part is then checked for its flattenability. All the other faces are flattened relative to this face by applying geometric rotation and translation transformations. Entity groups are found from the B-Rep information and based on the connectivity information and feature entities are formed. Feature extraction is done by looking for the existence of a corresponding associative entity for every feature entity. Once features are extracted, they are stored and represented in a graph-based format using a 3-D face oriented data structure. Pattern matching is done for the feature graphs with the feature pattern primitives that are stored in the database by checking for isomorphism and then feature classification is done.

Mantripragada et al. [5], have developed a feature based design system (refer figure 2.2 below), which acts like an interactive design tool which can be used to alert the designers to potential productions problems, defects and failures, and to provide them with information that can be used to explore alternative design, evaluate trade-offs and arrive at optimal designs for the given process conditions.

Figure 2.2 An integrated CAE system with deformation analysis and knowledge based system for optimal design of sheet metal parts.
The part is modeled using Pro-Engineer using its library of sheet metal features. Feature extraction is done using an external application program using Pro-Develop, which is a library of C functions, which provide a supported interface to Pro-Engineer and direct access to Pro-Engineer database. The authors have developed a system module for Analysis, which enables the user to perform mechanics and formability based analyses to predict stresses, strains, failures and defects for forming operations. The other modules developed are knowledge-based design support system module and CAD system modules. Design Support module contains information about tooling and process conditions in the early of design and knowledge to establish a systematic methodology for the design of box type sheet metal parts. The CAD module is developed on top of the existing modeler in Pro-engineer.

Jagirdar et. al. [7], have proposed a feature recognition methodology for recognizing shearing operations for 2-D sheet metal components created by a wire-frame model. The methodology initially processes geometric data for identifying different identity groups. The entity groups are classified into a raw material feature set, boundary feature set and an inside feature set. Manufacturing features are identified based on that proposed classification system for shearing operations and using different feature sets.

Lentz et al. [6], outline a hole extraction methodology for sheet metal components. Holes are grouped into two types on the basis of the number of faces that they emanate from. Type-1 hole is a hole, which emanates from a single face whereas in a type-2 hole, boundary of the hole is incident on more than one face. Topological and geometric properties of holes are described in order to point the way towards development of methodologies for their extraction. The authors use a Modified Feature Adjacency Hypergraph (MFAH), which is a modified version of the Attribute Adjacency Graph. Graph-based rules were developed to extract each of the two types of holes that they have described. The cross section of the hole need not be necessarily convex. The algorithm provides linear time complexity.
2.2 Cost Estimation of Sheet metal parts

Donovan et al. [1], [2] and [3] have created a computer design system for sheet metal parts manufactured using turret press and press brake operations. The CAD modeler interface used for modeling sheet metal parts used is Pro-Engineer. All the feature information is obtained through a Neutral text file “.inf” in which Pro-Engineer stores all the information related to sheet metal features in the order in which the features were created. Information gathered about the features includes size and shape of the sheet metal part, size and shape of the contour and the number of bends needed for design. Using this information as input, cost of sheet metal part is estimated. For estimating the cost, they have taken into account the press rate cost, material cost and the tool cost.

2.3 Process Planning

Wang C [8], et al. have developed a system for design and production of sheet metal parts. The system consists of two systems, the design system and the planning system. The design system called BendCad, which is a design-with features system, manages the relationships among the multiple representations of sheet-metal parts. The system uses multiple representations of the part during different manufacturing stages. The final assembly of the sheet metal product is represented as the connectivity relationship between 3-D parts.

The planning system consists of an operations planner, a tooling system, a grasping system, a robot motion planner and an open architecture controller. The planner generates the possible bend sequences and asks the sub-systems to evaluate the manufacturability costs. The planner uses a heuristic search method that uses the heuristic estimate of the cost between the current state and the goal state.

The BendCad system (refer figure 2.3) has various pre-defined features and these can either be directly generated using this system or are reasoned using the BendCad geometric kernel. The features in this system suggest precedence rules for the tool selection, grasping and motion strategies. These precedence rules form the knowledge base of the system and can be used as precedence heuristics or precedence constraints.
based on the certainty factors of these rules. Features also suggest special tools, workpiece grasping and fine motion strategies. These are used as constraints, which are related to the bending sequence.

Duflou J. R. [9] has developed a directed graph based approach for the reducing problem complexity for bend sequence verification of sheet metal parts. The basic problem is to identify a feasible bending sequence, according to geometric constraints, and to select the optimum solution based on ergonomical considerations. The process of evaluating feasible bending sequences for a part is computationally intensive and would require a study of \(2^n\).n.n! operations. Bending verification implies identification of a suitable gauging edge, interference checking for positioning and collision checking between machine and workpiece. The author suggests a redundancy elimination check. Also the use of a directed graph eliminates the number of cases to be considered. Pre-processing of the part geometry further reduces the problem size. This includes a backwards-unfolding check that identifies the bends that can be performed last. These and a few other set of heuristics described reduces the problem size.

![Figure 2.3 BendCad System](image-url)
2.4 Design And Manufacturability Analysis

Yeh, et al. [10] describe a feature based product modeler, Promod-S, which includes a rule-based design advisor among several other modules. The schematic is shown in Figure 2.4.

![Figure 2.4 Overview of Promod-S](image)

Methodologies for both the identification and correction of design violations are presented. The ProMod-S system considers only one side of the sheet metal part and reduces many three-dimensional problems to two-dimensional ones by assuming constant thickness across the part as shown in Figure 2.5.

![Figure 2.5 Assumption: Constant thickness of the part.](image)
Inter-feature violations are also addressed in this work. Voronoi diagrams, shown in Figure 2.6 are used to solve proximity problems are used in this system. The Voronoi diagrams for the feature and the surface that encloses it are generated and blended to obtain a medial axis transform (MAT). MAT greatly reduces the complexity of the search for design rule violations and avoids a combinatorial explosion.

Figure 2.6 Voronoi Diagram for a sample part

Radhakrishnan, et al. [11] further advances the medial axis transform technique by analyzing the design rules and derives the common characteristics of these rules. This aids in checking for violation of design rules for user-defined features as well. The rule-checking problem is modeled as a proximity problem of a multiply connected polygon and proceeds one face at a time. Each face is modeled as a multiply connected polygon. The medial axis approach reduces the search space considerably while checking for design violations in complex sheet metal parts. It also reveals plausible displacing directions for features that violate design rules to correct these violations. However, they do not consider rules for formed features and deep drawn parts.
3 Design Rules

Adequate design advice cannot be given without consideration of the tools available to manufacturing. Though any known design rule may not be violated, it may not be produced at a reasonable cost without the proper tools. Determining the best way of producing a part between product engineering, actual design of the required press dies and process planning requires knowledge about manufacturability of the part. The rules developed by such study are called design-for-manufacture rules and this is discussed here in the context of sheet metal parts [1].

This section describes the rules for sheet metal component design. Most companies have their best practice knowledge base for manufacturing to reduce infeasible designs, costs and production cycle times. However, this knowledge is treated as a trade secret and is not readily available. As a result, as Radhakrishnan, et al. [11] rightly point out, compiling comprehensive design rules for manufacture rules is a challenging task involving scavenging literature extensively, synthesizing the knowledge base in bits and pieces. Research carried out by S. Mallikarjun[12] identifies quite a few of the design rules. The work carried out by Shashikiran Hegde[13] attempts to compile a comprehensive design rule database across all operations for the domain of sheet metal parts.

This research helps in detecting violations leading to material failures across processes involved in sheet metal manufacture. This is a very important pre-cursor to costing of the part since a rule violation may imply that special tooling is required and the costs tend to accelerate. This means that two similar looking parts can widely vary in cost if any design rule is violated. If the designer costs the part with a design rule violation, only to find it very high, he/she might not know how to correct the wrong design unless suitable design recommendations are suggested.
Intra-feature Vs. Inter-feature

The design recommendations for sheet metal components may be broadly classified as intra-feature rules and inter-feature rules.

Intra-feature rules are those rules that are applied to the set of geometric elements that constitute a single feature. The early design rule checking systems were mostly based on intra-feature rules. The solid modelers available now can handle very complex solids and represent standard features as parametric primitives. These primitives have standard properties and can easily incorporate intra-feature design rules as constraints between their various parameters. This implementation would be simple and efficient.

Inter-feature rules are those rules that involve relationships between elements of more than one feature, which may or may not be of the same kind. These rules need search algorithms to identify the conditions under which the rule applies in their topology. Techniques have to be developed specific to each inter-feature design guideline that are easy to implement and also very efficient. Since this involves a combination of two or more features, it can soon lead to a combinatorial explosion if not searched for every possible combination. Ways to reduce the possible search space have to be devised.

The design advisory system developed in the current research includes both intra- and inter-feature type of rules. Solid modeler primitive based rules are used to search for the necessary feature combinations and detect possible violations of these rules.

The features that are basic to sheet metal parts are stamped features and formed features. Stamped features are categorized as internal stamped features and external stamped features. Internal stamped features include features like round hole, square hole, rectangular hole, obround hole etc. External stamped features includes different types of notches The most widely formed features are bends, lances, bridges, hems, seams and flanges.

In the next section, some of the design rules are discussed. The implementation details are discussed in the subsequent chapters. This work provides a firm foundation for a
knowledge base repository to build upon and interface with tool mapping, process planning and costing engines.

3.1 **Intra-feature rules.**

Some examples of intra-feature rules are explained below.

3.1.1 **Holes**

1. The diameter of punched holes should be greater than the thickness of the work piece and not less than 2.4892 mm (0.098 in) as shown in Figure 3.1. This is because the punch diameter becomes too small to bear the shear force required to punch the hole over a small area, often leading to failure. These small diameter holes are punched using the fine-blanking process.

![Figure 3.1 Diameter/Thickness design rule](image)

2. Oblong holes: Oblong holes are generally incorporated as adjustment slots. As a general rule, the length to width ratio of an adjustment slot should be no more than a 5 to 1 ratio, with a maximum of 2 to 1 permissible.

3. Holes on faces having surface angles greater than 30° should be avoided as far as possible.
3.1.2 Notches

1. Notches should extend inside the stock edge at least 1.5 times the thickness but not less than 0.508 mm.
2. The minimum widths of tabs and slots should be 1.5T or 0.5 mm (0.020 in). Their length should be a maximum of 5 times their width.
3. For long narrow projections the minimum width of narrow sections should be 1.5 times the thickness. The rules are shown in Figure 3.2.

3.1.3 Lanced lugs

1. Lanced lugs should have tapered sides if no clearance is provided.

3.1.4 Triangular tabs or slots

1. Triangular tabs or slots should have minimum end radii of one thickness and should form included angles of 60° or more as shown in Figure 3.4.
2. Tapers for blanks should be recessed at least one thickness from the edge of the part.
3. Components requiring round ends should have a radius greater than or equal to 0.75W unless otherwise stated as shown in Figure 3.5. The radius is arrived at here based on the minimum angle that is recommended as acceptable.

4. 0.5W may be used if a relief angle 10° or greater at the point of tangency with the part edge is also used as shown in Figure 3.6

Following rules are applicable only for forming operations in sheet metal parts.

3.1.5 Beads

1. The maximum height for beads is 2 times the thickness.
2. The minimum inner radius for beads is equal to the thickness of the piece
3.1.6 Bridges

1. The maximum ratio of lance length to bridge height is 4:1.

3.1.7 Curls

1. The inner diameter for curls should be 2.5 times to 8 times the metal thickness.

3.1.8 Bends

1. The width along the bend axis should be greater than or equal to 3 times the thickness.
2. The bend radius (rb)

\[ r_b \geq 1.6 \text{mm or } r_b \geq t \text{ (thickness of workpiece)} , \text{ whichever is greater.} \]

3. Metal that is rolled shows grains along the direction in which the stock was drawn though the mill rolls. Bends should be at right angles to the grains or as close as possible to avoid breakage of the blank as shown in Figure 3.7. These factors must be considered while producing the blank and sometimes the most economical layout of the strip will have to be abandoned in order to avoid trouble in later forming operations. If the bends are at right angles, the blank should be diagonal to the grain.

![Figure 3.7 Direction of rolling][15]
3.1.9 Tapped Holes

1. A design guideline for the minor thread diameter (tap-drill size) as shown in figure 3.8, is that it should not exceed twice the stock thickness for steel and brass and 1.5 times the stock thickness for Aluminum, copper and zinc.

![Figure 3.8 Tapped Holes](image)

3.2 Inter-feature Rules

1. The spacing between the holes should be at least 2 times the stock thickness but not less than 1.5 mm (0.060 in) and preferably 3 times to provide additional strength to the die, by allowing a greater wall thickness as shown in Figure 3.9.

![Figure 3.9 Distance between two holes](image)

2. The minimum distance from the edge of the hole to the edge of any other feature should be greater than the stock thickness and not less than 0.8 mm (0.030 in). It is recommended that this space be at least 1.5 to 2 times the thickness of the work-piece to the hole.
3. The lowest edge of the hole should be at least 1.5 times stock thickness plus radii of the bend. Distortion of the hole will occur if the two features are any closer. If this placement is absolutely necessary, then a stress discontinuity should be provided during forming to prevent the distortion of the hole by a non-functional hole, slot or tab as shown in Figure 3.11.

![Figure 3.11 Distance of the hole from another feature](image)

4. Slots that are parallel to the bend should be a minimum of 4 times the stock thickness ($T$) from the bend tangent line as shown in Figure 3.12.

![Figure 3.12 Distance of the hole from the edge](image)

5. Sharp corners, which are stress concentration points, should be avoided wherever possible. Sharp external corners of punches and dies tend to break down prematurely, causing more pull-down, larger burrs, or rougher edges of the blanked part in the area of the corner. Similarly, sharp interior corners of punches and dies are a stress-concentration point and can lead to cracking and failure from heat treatment or in use.
A design guideline is to allow a minimum corner radius of two times the stock thickness and never less than 0.16 in. Sharp corners produced wherever two edges produced by separate shearing, slotting or blanking operations intersect at approximately 90° or less may have to be rounded by some secondary operation to achieve corners.

6. Adjacent tabs or slots should be spaced a minimum of 2T or 0.8 mm (0.030 in) apart as shown in Figure 3.13.

![Figure 3.13 Distance between adjacent tabs or slots][14]

7. Internal slots should be at least 1.5T or 0.8 mm (0.030 in) from the edge of the stock as illustrated in Figure 3.14.

![Figure 3.14 Distance of the hole from the edge][14]

8. Extruded holes should have a minimum spacing of 6T between their edges and should be at a minimum of 4T from the blank edge. Depth of the extrusion should be a maximum of 30 percent of its outer diameter as shown in Figure 3.15.

![Figure 3.15 Spacing for extruded holes][14]
9. Threaded screw or bolt holes should be at least 1.5 times the screw diameter from the centerline of the hole to the edge of the part as shown in Figure 3.16.

![Figure 3.16. Spacing for threaded holes](image)

10. Aligned hole in opposite bends: It is often required to include two aligned holes in opposite legs of a U-bent part for holding a shaft or for some other purpose. It is difficult to form such a part from a pre-pierced blank and have the holes aligned precisely. Several alternatives can be considered as shown in Figure 3.17: (1) Pierce or drill holes after forming, this provides excellent alignment though more expensive. (2) Use broad tolerances on the holes, or make one a slot, allowing for misalignment if the part function permits. (3) Include a pilot hole in the bottom of the U bend. This hole is located over a pin in the pad of the forming die that will position the blank consistently. (4) Stock of close thickness can be used for true alignment. Though the material is more costly, savings resulting from not having to perform a second operation may offset the extra cost.
Figure 3.17. Aligned Holes on opposite bends[14]
4 Feature Extraction and Design Advisor for Intra-Feature Rules

The concept of features has been developed as a means of bridging the gap between the low-level geometric data representations and the high level information desired by CAD users. A feature is thus a geometric form or entity used in reasoning about the geometry of a part by one or more design or manufacturing activities.

Sheet metal features can be categorized into stamped features and formed features. In the current work done in feature extraction, stamped features are classified into Internal Stamped Features and External Stamped features depending on if they lie on the outer loop of the flattened sheet metal part or if they form one of the inner loops. Formed features are features like bends, hems, curls etc. Only bends have been considered in the current research.

Detailed analysis of each of these categories of features is presented in the following sections of the chapter. Along with the feature extraction, the parts are checked for manufacturability. These design rules are classified into Intra-Feature Design Rules and Inter-Feature Design Rules. Detail implementation of Intra-Feature Design is carried out along with feature extraction in the current scope of research.

The following figure 4.1 shows the overview of the system developed for feature extraction and design advisor.
4.1 Internal Feature Extraction

Following are the different types of Internal Stamped Features that are extracted using the feature extraction module:

1. Circle Feature
2. Single-D Feature
3. Rectangle Feature
4. Double-D Feature
5. Oblong Feature
6. Custom Feature

The following figures represent the different types of internal stamped features that are listed above. Each of the above features presented an internal loop found in the base face of the flat pattern feature of the sheet metal part.

![Diagram of internal stamped features]

Rectangle Feature  Circle Feature  Single-D Feature  Double-D Feature  Oblong Feature

Figure 4.2 Standard Library for Internal Stamped Features

4.2 General Methodology of feature extraction

The sheet metal part is first flattened by unsuppressing the flat pattern feature. Next each of the loops except the outermost loop of the base face is extracted. Each of these loops is then the input to the InternalFeatureExtraction Algorithm. They are classified into one of standard features listed above or else they are classified as Custom Features.

Detailed analysis and treatment of every standard feature is presented in the next sections of the chapter.
4.3 Circular Feature

![Circular Feature Sketch](image)

**Figure 4.3 Circular Feature Sketch**

4.3.1 Feature Extraction: Following rules are followed for circular feature extraction.

1. Check if the loop has only 1 edge.
2. Check if that edge is a circular arc
3. Check if the circular arc is a closed one. This is done by taking the difference of the end parameter of the edge and the start parameter of the edge. If the difference is exactly equal to \(2\pi\), then the arc is a circle.

4.3.2 Feature Information: The following feature information is stored.

1. Diameter of the circle: \(D\)
2. Co-ordinates of the center of the circle: \(C\)

4.3.3 Design Rules: The following design rules are validated for the extracted feature.

1. Check: \(D/2 > t\) or \(2.4892 \text{ mm}\), whichever is larger.
Figure 4.4 Feature Extraction: Circular Feature

An example of Circular Feature extraction is shown in figure 4.4 above.
4.4  Rectangle Feature

**Figure 4.5 Rectangle Feature Sketch**

4.4.1 Feature Extraction: Following rules are followed for rectangle feature extraction.

1. Check if the loop has 4 edges
2. Check if all the edges are straight lines
3. Take the pair-wise dot product of adjacent sides. This is done by forming the unit vectors along each of the edges and then taking the dot product of the two vectors. Check that all the dot product values are equal to zero.

4.4.2 Feature Information: The following feature information is stored for the feature.

1. Length of the feature $L$
2. Width of the feature $W$
3. Center of the rectangle $C$. It is calculated as follows:

$$C = (S_1 + E_1 + S_2 + E_2 + S_3 + E_3 + S_4 + E_4) / 8$$
where, \( S_i, E_i \) represent the start and end point of the first edge and similarly for the other edges. This method works for any rectangle even if the axes are not parallel to the edges of the rectangle.

4.4.3 Design Rules: The following design rules are validated for the feature.

1. Check \( L > 1.5 \times \text{Sheet Metal Thickness} \)
2. Check \( W > 1.5 \times \text{Sheet Metal Thickness} \)
3. Check \( L < 5 \times W \)

Figure 4.6 Feature Extraction: Rectangular Feature

An example of Rectangle Feature extraction is shown in figure 4.6 above.
4.5 Single-D Feature

![Figure 4.7 Single-D Feature Sketch](image)

4.5.1 Feature Extraction: Following rules are followed for Single-D feature extraction.

1. Check if the loop has two edges
2. Check if one of the edges in the loop is a straight line and the second edge is an arc.

4.5.2 Feature Information Stored: The following information is stored for the feature.

1. Length of the feature $L = $ Diameter of the arc
2. Angle of the arc $\theta$
3. Width of the feature $W = R \left(1 + \cos \left(180 - \theta \right)/2 \right)$
4. Center of the feature, $C$, which is the center of the arc.

4.5.3 Design Rules: The following design rules are validated for the feature.

1. Check $L > 1.5 \times $ Sheet metal thickness
2. Check $W > 1.5 \times $ Sheet metal thickness
3. Check $L < 5 \times W$
Fig 4.8 Feature Extraction: Single-D Feature

An example of Single-D Feature extraction is shown in figure 4.8 above.
4.6 Double-D Feature

Figure 4.9 Double-D Feature Sketch

4.6.1 Feature Extraction: Following Rules are followed for Double-D feature extraction

1. Check if the loop has four edges
2. Check if there are two straight lines and two circular edges
3. Check that the two straight lines are opposite to each other and similarly for the two arcs, i.e. the two straight lines and arcs are alternating.
4. Check if the two straight lines are parallel to each other. This is done by forming unit vectors along the two straight lines and then checking to see if the magnitude of cross product of the two vectors is zero.
5. Check if the angle of arcs, for both of arcs are equal in magnitude.
6. If each of the angles is 180°, i.e, the arcs are semi-circles, then the feature is a oblong feature, which is a specific case of a Double-D feature. It is categorized as a separate standard feature and treated separately.
4.6.2 Feature Information: Following information parameters are stored for this feature.

1. Calculation of Length of the feature: $L$
2. Calculation of Width of the feature: $W$
3. Center of the Feature $C$. It is calculated as follows:

$$C = \frac{(S_1 + E_1 + S_2 + E_2 + S_3 + E_3 + S_4 + E_4)}{8}$$

Where, $S_i$ and $E_i$ are the start and end points of Edge 1 and similarly for the other edges. This method works even if the feature does not have its edges aligned with the principal axes.

$L_1$ = Length of the straight edge

$R$ = Radius of the arc

$\Theta$ = Angle of the arc

$L_2 = R \times (1 - \cos (\Theta / 2))$

$L = L_1 + 2 \times L_2$

$W = 2 \times R \times \sin (\Theta / 2)$

4.6.3 Design Advisor: The following rules are used to validate this feature parameters.

1. Check if $L > 1.5 \times \text{Sheet Metal Thickness}$
2. Check if $W > 1.5 \times \text{Sheet Metal Thickness}$
3. Check if $L < 5 \times W$
Figure 4.10 Feature Extraction: Double-D Feature

An example of Double-D Feature extraction is shown in figure 4.10 above.
4.7 Oblong Feature

Figure 4.11 Oblong Feature Sketch

4.7.1 Feature Extraction:

The Steps are same as those described in the Feature Extraction for Double-D Feature

4.7.2 Feature Information: Following feature parameters are calculated and stored.

1. Calculation of Length of the feature: \( L \)
2. Calculation of Width of the feature: \( W \)
3. Center of the Feature \( C \). It is calculated as follows:

\[
C = (S_1 + E_1 + S_2 + E_2 + S_3 + E_3 + S_4 + E_4) / 8
\]

where, \( S_i \) and \( E_i \) are the start and end points of edge 1 and similarly for the other edges. This works even if the feature does not have edges aligned with the principal axes.
\[ L_1 = \text{Length of the straight edge} \]

\[ R = \text{Radius of the arc} \]

\[ W = 2 \times R \]

\[ L = L_1 + 2 \times R \]

4.7.3 Design Rules: The feature is validated against the following rules.

1. Check if \( L > 1.5 \times \text{Sheet Metal Thickness} \)
2. Check if \( W > 1.5 \times \text{Sheet Metal Thickness} \)
3. Check if \( L < 5 \times W \)

Figure 4.12 Feature Extraction: Oblong Feature

An example of Oblong Feature extraction is shown in figure 4.4 above.
4.8 Algorithm for Classifying Internal Stamped Features

The following flowchart (figure 4.13) describes the overall internal feature extraction steps.

**INPUT : EDGE LOOP**

- **CHECK NUMBER OF EDGES IN THE LOOP**
  - **TWO**
  - **FOUR**
  - **ONE**
    - **CIRCLE FEATURE**
    - **OTHER**

- **CHECK IF ONE EDGE IS STRAIGHT AND OTHER EDGE IS CURVED**
  - **YES**
    - **FEATURE IS A SINGLE-D FEATURE**
  - **NO**
    - **OTHER**

- **CHECK IF THERE ARE FOUR STRAIGHT EDGES OF TWO STRAIGHT LINES AND TWO ARCS**
  - **TWO EDGES AND TWO ARCS**
    - **YES**
      - **FOUR STRAIGHT EDGES**
      - **YES**
    - **NO**
      - **OTHER**

- **CHECK IF THE FOUR EDGES ARE ADJACENTLY PERPENDICULAR**
  - **YES**
    - **RECTANGLE FEATURE**
  - **NO**
    - **OTHER**

- **CHECK IF OPPOSITE EDGES ARE OF THE SAME TYPE**
  - **YES**
    - **OTHER**
  - **NO**
    - **OTHER**

- **CHECK IF THE TWO STRAIGHT LINES ARE PARALLEL TO EACH OTHER**
  - **YES**
    - **OTHER**
  - **NO**
    - **OTHER**

- **CHECK IF THE LENGTH OF THE STRAIGHT LINES IS THE SAME**
  - **YES**
    - **OTHER**
  - **NO**
    - **OTHER**

- **CHECK IF THE ARCS ARE EQUAL IN ANGLE**
  - **YES**
    - **OTHER**
  - **NO**
    - **OTHER**

- **CHECK IF THE ANGLE IS EQUAL TO 180 DEGREES**
  - **YES**
    - **OBLONG FEATURE**
  - **NO**
    - **DOUBLE-D FEATURE**

- **CUSTOM FEATURE**

**Figure 4.13 Flowchart for Internal Stamped Feature Classification**
4.9  External Feature Extraction

List of standard features extracted as external features:

1. Arc – Notch Feature
2. V – Notch Feature
3. Straight U – Notch Feature
4. Arc U – Notch Feature
5. Custom Feature

These features are extracted based on the convexity or concavity between successive edges. The following figures represent each of these features:

![Figure 4.14. Standard External Features](image)

**Custom Feature** is any sequence of edges, which form a continuous non-convex pair.
4.10 Feature String Pattern Methodology

Based on the connectivity between the adjacent edges, i.e. whether a particular vertex is convex or concave, a value of 0 or 1 is assigned to the second edge in the pair of edges considered. The entire loop is traversed in the counter clockwise direction and all the edges are assigned a value depending on the connectivity between that edge and the earlier edge. The following figures explain the methodology of assigning a value of 1 or 0 to second edge in the pair.

Terminology:

- "\( E_6 \) --- Going into the plane of the paper (Inward normal)
- "\( E_1 \) --- Coming out of the plane of the paper (Outward normal)

**Figure 4.15** Figure showing the convexity between edges

The cross product of the two edges \( E_1 \) and \( E_2 \) will result in a normal, which is directed out of the plane of the paper. In this case, the second edge in the pair \( E_2 \) is assigned a value of 1. In order to develop an algorithm for developing the feature string the following technique is used. The face normal for the surface in the outward direction can be calculated and then the dot product of the face normal and the normal vector corresponding to the cross product of vectors along \( E_1 \) and \( E_2 \) are taken. This dot product will be positive if the normal vector is along the direction of the face normal, and negative if the normal vector is in the opposite direction to the face normal. So depending on when the value is positive or negative, a value of 0 or 1 is assigned to the corresponding edge. The feature pattern for the above figure will be as follows:
Figure 4.16 Figure showing the feature pattern values for the edges based on the convexity

So the Feature string is: 1-1-0-1-1

If one of the edges is an arc, it is approximated as two straight edges, by two chords, one joining the start point and mid point of the chord. The rest of the methodology is same as the case discussed for straight edges. An example of a part with a circular edge is shown below.

Figure 4.17 Figure showing feature pattern methodology where one of the edges is an arc

The feature string for the above figure is: 1-1-0-1-1.
One more example of a part and its feature string is presented in Figure 4.18.

Figure 4.18 Figure showing a snap shot of a sample sheet metal part with external features and its feature string pattern

The detailed information about this figure is shown in the form of table below

<table>
<thead>
<tr>
<th>Counter</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature Pattern Array</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Feature Pattern String in this case is thus: 1-1-0-1-1-0-1-1-0-0-1-1-0-0-0-1
4.11 Detailed Description of the Standard Features

The following sections describe each of the standard features and the following details about the features:

- Feature pattern string
- Feature Extraction algorithm
- Feature information stored and methodology for calculating feature parameters
- Design rules validated for the feature
- A Snap of a sheet metal part from SolidWorks showing feature extraction and design validation of the part.

4.12 Arc Notch Feature

![Arc Notch Feature Sketch](image)

Figure 4.19 Arc Notch Feature Sketch

4.12.1 Feature Pattern String

\((E_1 - E_2 - E_3): 1--0-1\)

4.12.2 Feature Extraction

1. Search the feature pattern of the loop for the feature pattern string \(1-0-1\)
2. Check if the second and the third element in the pattern string belong to the same edge and that edge is a circular arc.
3. Check that the edge corresponding to the first element in the pattern string is a straight line.
4.12.3 Feature Information

Information stored about the feature is:

1. Co-ordinates of the Start-Point of the notch (S)
2. Co-ordinates of the Mid-Point of the notch (M)
3. Co-ordinates of the End-Point of the notch (E)
4. Width of the Notch (W)
5. Depth of the Notch (D)

Figure 4.20 Arc Notch Feature Sketch Details

4.12.4 Methodology for calculating feature parameters:

\[ S = \text{Start Vertex of edge } E_1 \]

\[ M = \text{Mid Vertex of edge } E_2 \]

\[ E = \text{End Vertex of edge } E_3 \]
Calculations for Depth and Width of the notch are as follows:

\[ D = R (1 - \cos \left( \frac{\theta}{2} \right)), \] where \( \theta \) is the angle of the arc.

\[ W = 2 R \sin \left( \frac{\theta}{2} \right) \]

4.12.5 Design rules for validating the feature:

1. Check \( W < 1.5 \times \text{swSheetMetalThickness} \text{ or } 0.5 \text{ mm} \) whichever is more
2. Check \( L < 1.5 \times \text{swSheetMetalThickness} \text{ or } 0.508 \text{ mm} \) whichever is more
3. Check \( L < 5 \times W \)

4.12.6 Example part showing feature extraction and design validation for the feature.

Figure 4.21 Arc Notch Feature Extraction
4.13 V – Notch Feature

Figure 4.22: Figure showing the feature pattern for a V-Notch feature

4.13.1 Feature Pattern String:

\((E_1 - E_2 - E_3): 1--0-1\)

4.13.2 Feature Extraction Algorithm

1. Search the feature pattern of the loop for the feature pattern string \(1-0-1\)
2. Check if all the edges corresponding to the three feature pattern string elements are straight lines.

4.13.3 Feature Information

1. Information stored about the feature is:
2. Co-ordinates of the Start-Point of the notch (S)
3. Co-ordinates of the Mid-Point of the notch (M)
4. Co-ordinates of the End-Point of the notch (E)
5. Width of the Notch (W)
6. Depth of the Notch (D)

We consider two cases here:
Case (a): $D_1 < D_2$

Figure 4.23 Figure showing geometry of V-Notch feature for case(a)

Case (b): $D_1 > D_2$

Figure 4.24 Figure showing geometry of V-Notch feature for case(b)
4.13.4 Methodology for calculating feature information:

\[ S = \text{Start Vertex of edge } E_1 \]

\[ M = \text{Mid Vertex of edge } E_2 \]

\[ E = \text{End Vertex of edge } E_3 \]

Calculations for Depth and Width of the notch:

\[ \hat{e}_1 = \text{Unit Vector along edge } E_1 \]

\[ \hat{e}_2 = \text{Unit Vector along edge } E_2 \]

\[ \hat{e}_3 = \text{Unit Vector along edge } E_3 \]

\[ \cos \theta_1 = \text{dot product} (\hat{e}_1, \hat{e}_2) \]

\[ \cos \theta_2 = \text{dot product} (\hat{e}_2, \hat{e}_3) \]

\[ W_1 = E_1 \cos \theta_1 \]

\[ D_1 = E_1 \sin \theta_1 \]

\[ W_2 = E_3 \cos \theta_2 \]

\[ D_2 = E_3 \sin \theta_2 \]

If \( D_1 < D_2 \) then

\[ W_3 = (D_2 - D_1) / D_2 \times W_2 \]

Else

\[ W_3 = (D_1 - D_2) / D_1 \times W_1 \]
\[ D = \min (D_1, D_2) \]

\[ W = W_1 + W_2 - W_3 \]

4.13.5 Design Rules for validating the feature:

1. Check \( W < 1.5 \times \text{swSheetMetalThickness} \)
2. Check \( L < 1.5 \times \text{swSheetMetalThickness} \)
3. Check \( L < 5 \times W \)

4.13.6 An example showing the feature extraction and design validation for this feature:

![Figure 4.25 V-Notch Feature Extraction](image)

Figure 4.25 V-Notch Feature Extraction
4.14 Straight U – Notch Feature

4.14.1 Feature Pattern String

\((E_1 – E_2 – E_3 – E_4): 1\text{-}0\text{-}0\text{-}1\)

4.14.2 Feature Extraction

1. Search the feature pattern of the loop for the feature pattern string 1-0-0-1
2. Check if all the four edges corresponding to the feature pattern string elements are straight lines.

4.14.3 Feature Information

1. Information stored about the feature is:
2. Co-ordinates of the Start-Point of the notch \((S)\)
3. Co-ordinates of the Mid-Point of the notch \((M)\)
4. Co-ordinates of the End-Point of the notch \((E)\)
5. Width of the Notch \((W)\)
6. Depth of the Notch \((D)\)
We consider two cases here:

Case (a): \( D_1 < D_2 \)

Figure 4.27 Straight U-Notch Feature Details for case(a)

Case (b): \( D_1 > D_2 \)

Figure 4.28 Straight U-Notch Feature Details for case(b)
4.14.4 Methodology for calculating feature parameters:

\[ S = \text{Start Vertex of edge } E_1 \]

\[ M = \text{Mid Vertex of edge } E_2 \]

\[ E = \text{End Vertex of edge } E_3 \]

Calculations for Depth and Width of the notch:

\[ \hat{e}_1 = \text{Unit Vector along edge } E_1 \]

\[ \hat{e}_2 = \text{Unit Vector along edge } b \]

\[ \hat{e}_3 = \text{Unit Vector along edge } E_3 \]

\[ \cos \theta_1 = \text{dot product (} \hat{e}_1 , \hat{e}_2 \text{)} \]

\[ \cos \theta_2 = \text{dot product (} \hat{e}_2 , \hat{e}_3 \text{)} \]

\[ W_1 = E_1 \cos \theta_1 \]

\[ D_1 = E_1 \sin \theta_1 \]

\[ W_2 = E_3 \cos \theta_2 \]

\[ D_2 = E_3 \sin \theta_2 \]

If \( (D_1 < D_2) \) then

\[ W_4 = (D_2 - D_1) / D_2 \cdot W_2 \]

Else

\[ W_4 = (D_1 - D_2) / D_1 \cdot W_1 \]
\[ D = \min (D_1, D_2) \]

\[ W = W_1 + W_2 + W_3 - W_4 \]

4.14.5 Design rules for validating the feature parameters:

1. Check \( W < 1.5 \times \text{swSheetMetalThickness} \)
2. Check \( L < 1.5 \times \text{swSheetMetalThickness} \)
3. Check \( L < 5 \times W \)

4.14.6 Example showing feature extraction and design validation for the feature.

![Figure 4.28 Straight U-Notch Feature Extraction](image)
4.15 Arc U – Notch Feature

![Figure 4.29 Arc U-Notch Feature Pattern](image)

4.15.1 Feature Pattern String

\((E_1 - E_2 - E_3 - E_4 - E_5): 1-0-0-0-1\)

4.15.2 Feature Extraction

1. Search the feature pattern of the loop for the feature pattern string \(1-0-0-0-1\)
2. Check if the second and the third element in the pattern string belong to the same edge and that edge is a circular arc.
3. Check that the edges corresponding to the first, fourth and fifth elements in the pattern string are straight lines.

4.15.3 Feature Information

Information stored about the feature is:

1. Co-ordinates of the Start-Point of the notch (S)
2. Co-ordinates of the Mid-Point of the notch (M)
3. Co-ordinates of the End-Point of the notch (E)
4. Width of the Notch (W)
5. Depth of the Notch (D)
We consider two cases here:

Case (a): $D_1 < D_2$

Figure 4.30 Arc U-Notch Feature Sketch Details for case (a)

Case (b): $D_1 > D_2$

Figure 4.31 Arc U-Notch Feature Sketch Details for case (b)
4.15.4 Methodology for calculating feature parameters:

\[ S = \text{Start Vertex of edge } E_1 \]

\[ M = \text{Mid Vertex of arc} \]

\[ E = \text{End Vertex of edge } E_3 \]

Calculations for Depth and Width of the notch:

\[ \hat{e}_1 = \text{Unit Vector along edge } E_1 \]

\[ \hat{e}_2 = \text{Unit Vector along the constructed edge } E_2, \text{ which is formed by joining the start and end points of the arc.} \]

\[ \hat{e}_3 = \text{Unit Vector along edge } E_3 \]

\[ \cos \theta_1 = \text{dot product } (\hat{e}_1, \hat{e}_2) \]

\[ \cos \theta_2 = \text{dot product } (\hat{e}_2, \hat{e}_3) \]

\[ W_1 = E_1 \cos \theta_1 \]

\[ D_1 = E_1 \sin \theta_1 \]

\[ W_2 = E_3 \cos \theta_2 \]

\[ D_2 = E_3 \sin \theta_2 \]

\[ D_3 = R (1 - \cos (\theta/2)), \text{ where } \theta \text{ is the angle of the arc.} \]

If \( D_1 < D_2 \) then

\[ W_4 = (D_2 - D_1) / D_2 \times W_2 \]

Else
\[ W_4 = \frac{(D_1 - D_2)}{D_1} \times W_1 \]

\[ D = \min(D_1, D_2) + D_3 \]

\[ W = W_1 + W_2 + W_3 - W_4 \]

4.15.5 Design Rules for validating the feature parameters:

1. Check \( W < 1.5 \times \text{swSheetMetalThickness} \)
2. Check \( L < 1.5 \times \text{swSheetMetalThickness} \)
3. Check \( L < 5 \times W \)
4. Check \( \theta > 60 \text{ degrees} \)
5. Check if \( R > \text{Sheet metal thickness} \)
6. Check if the straight edges are tangent to the arc. This is only a recommendation and not a design violation.

4.15.6 Example showing feature extraction and design validation for the part:

![Figure 4.32 Arc U-Notch Feature Extraction](image-url)
4.16 Custom Feature

![Figure 4.33 Custom Feature Pattern](image)

**Figure 4.33 Custom Feature Pattern**

4.16.1 Feature Pattern String

\[(E_1 - E_2 - E_3 - E_4 - E_5): 1-(0^*)-1\]

4.16.2 Feature Extraction

1. Search the feature pattern of the loop for the feature pattern string \(1-(0^*)-1\) where the number of zeros is not corresponding to any of the standard features and the feature string does not correspond to any of the other standard external features.

4.16.3 Feature Information

Information stored about the feature is:

1. Co-ordinates of the Start-Point of the notch (S)
2. Co-ordinates of the Mid-Point of the notch (M)
3. Co-ordinates of the End-Point of the notch (E)
4. Width of the Notch (W)
5. Depth of the Notch (D)
4.16.4 Methodology for calculating feature parameters:

\( S = \) Start Vertex of edge \( E_1 \)

\( E = \) End Vertex of edge \( E_3 \)

Calculations for Depth and Width of the notch:

\( W = \) Length of the straight line joining \( S \) and \( E \)

For calculating depth, the following approximation is used:

The distance from every vertex in the feature other than \( S \) and \( E \) to the line joining \( S \) and \( E \) is calculated and the maximum distance among all these values is considered to be the depth of the feature

4.16.5 Design Rules for validating the feature parameters:

1. Check \( W < 1.5 \times sw\text{SheetMetalThickness} \) or \( 0.5 \text{ mm} \) whichever is more
2. Check \( L < 1.5 \times sw\text{SheetMetalThickness} \) or \( 0.508 \text{ mm} \) whichever is more
3. Check \( L < 5 \times W \)
4.16.6 Example of feature extraction and design validation for the feature:

![Custom Feature Extraction](image)

Figure 4.35 Custom Feature Extraction
4.17 Data structures used

4.17.1 StartPointArray, EndPointArray and Edge Orientation process

In forming the featurePatternArray, the orientation of the edges is very important because all the edges are scanned in counter clockwise direction when the featurePatternArray is formed.

By default when any particular edge loop is considered, if all the edges are scanned the orientation of the edges, i.e the start and the end points are such that they follow the counter clockwise convention. An example is shown in the figure 4.36:

![Figure 4.36 Orientation of edges - 1](image)

In the above case, all the edges are oriented in the counter clockwise direction. Hence, the order of vertices will be as follows:

\[E_1: \text{Start Vertex 1, End Vertex 2}\]

\[E_2: \text{Start Vertex 2, End Vertex 3}\]

\[E_3: \text{Start Vertex 3, End Vertex 4}\]
$E_4$: Start Vertex 4, End Vertex 1

However, there is an exception to this convention. This exception is explained as follows:

The orientation of the edges very much depends on at what stage during modeling the edge has been created. The following example illustrates this point.

![Figure 4.37 Orientation of edges - 2](image)

**Figure 4.37 Orientation of edges - 2**

There are two ways in which this model, which is an extruded sheet metal part can be created. In the first method, the entire 2–D sketch shown above is first drawn and then the entire sketch is extruded in a direction normal to the plane of the paper. In that case, the orientation of all the edges remains correct.

However, if the user models the same part in the way as shown below, then that causes a discrepancy in the orientation of all the edges. The explanation is as follows:

Consider that initially the part shown below is sketched. The sketch is then extruded as a sheet metal part.
Now, the part is again edited to include the arc shown in the above figure as an extruded cut. This is shown below as follows:

So when the circle is sketched, the orientation of the circle will be as shown, in order to be consistent with the counter clockwise convention of orientation. Then after the circle is cut-extruded through the thickness of the sheet metal part, the resulting sheet metal part will be similar to the first part shown but with a slight difference in the orientation which is shown below.
The arc part of the circle which is labeled as $E_4$ retains its original orientation, which causes a reversal of the orientation of that edge relative to the other edges. This causes a problem when the edges are scanned and assumed that the order of the vertices is counter clockwise orientation. In this case for the edge $E_4$, Start Vertex : 4 and End Vertex : 5 are in clockwise orientation compared to the rest of the edges in the loop. To overcome this problem, the orientation of the edges is checked and then an array is maintained. The edges are realigned and the correct order is maintained in an array called the startpoint array and endpoint array corresponding to every edge. The size of this array is the same as the number of edges in the loop.

The module which checks the orientation of the edges is generalized to take into consideration all the cases. This is explained as below:

Consider two adjacent edges, The two edges can be either straight lines or arcs. The methodology remains the same for both straightlines and arcs.

$E$ and $S$ denote the end and start points of the edge and the sub script denotes the order of the edge: 1 being the first edge and 2 being the second edge.

There are four possibilities to be considered:

Case 1:

\[ \text{S}_2 \quad \text{E}_2 \quad \quad \text{S}_1 \quad \text{E}_1 \]

\[ \text{Figure 4.41 Re-Orienting the edges - 1} \]

In this case, both the edges are aligned in the correct direction, because when travelled from the first edge to the second edge, the orientation obtained is counter clockwise. Assuming that the two edges are having index $k$ and $k+1$ then the vertices stored in the array are:
\(\text{StartPointArray}(k) = S_1\)

\(\text{EndPointArray}(k) = E_1\)

\(\text{StartPointArray}(k+1) = S_2\)

\(\text{EndPointArray}(k+1) = E_2\)

Case 2:

\[\begin{array}{cc}
\text{E}_2 & \text{S}_2 \\
\text{S}_1 & \text{E}_1 \\
\end{array}\]

\textbf{Figure 4.42 Re-Orienting the edges - 2}

In this case, the orientation of the first edge is in the clockwise direction. Therefore, the values which are stored in the \textit{StartPointArray} and \textit{EndPointArray} are:

\(\text{StartPointArray}(k) = E_1\)

\(\text{EndPointArray}(k) = S_1\)

\(\text{StartPointArray}(k+1) = S_2\)

\(\text{EndPointArray}(k+1) = E_2\)

Similarly, there are two other cases to be considered as shown below:

Case 3:

\[\begin{array}{cc}
\text{S}_2 & \text{E}_2 \\
\text{E}_1 & \text{S}_1 \\
\end{array}\]

\textbf{Figure 4.43 Re-Orienting the edges - 3}

Here the alignment of edge \(k\) is counter-clockwise, but the Edge \(k+1\) is aligned in the clockwise direction. The values of \textit{StartPointArray} and \textit{EndPointArray} are:
Case 4:

\[
\text{StartPointArray}(k) = S_1 \\
\text{EndPointArray}(k) = E_1 \\
\text{StartPointArray}(k+1) = E_2 \\
\text{EndPointArray}(k+1) = S_2
\]

In this case, alignment of both the edges is counter clockwise. The values dumped in the StartPointArray and EndPointArray are:

\[
\text{StartPointArray}(k) = E_1 \\
\text{EndPointArray}(k) = S_1 \\
\text{StartPointArray}(k+1) = E_2 \\
\text{EndPointArray}(k+1) = S_2
\]

4.17.2 NewEdgeLocationArray

This is the key array used in case of nested feature extraction. Before the start of the Feature Classification, all elements of this array are initialized to –1. When a feature is found, start and end index of that feature are obtained. After classifying the feature, in the next pass before feature extraction, this feature has to be filled with a straight line connecting the startpoint of the first edge and the endpoint of the last edge in the feature. To make a note of this, in the array, all the elements corresponding to the edges of the feature are marked with the index of the first edge of that feature. At the end of one
iteration of classification of Features the part is redrawn. Features that are extracted in the current iteration are being filled using this array as reference to find the edges that are a part of one feature. All edges for which the value of the NewEdgeArray element corresponding to index have a value other than –1 form a feature. The value of index of edge stored in every element of a feature distinguishes one feature from another. This using this array, it is possible to successfully redraw the sheet metal part with all the features found in the earlier pass being filled.

An example demonstrating the use of NewEdgeLocation Array is explained below in Figure 4.45.

![Figure 4.45 Feature String Pattern for a sample sheet metal part.](image-url)
The detailed information about this figure is shown in the form of table below

<table>
<thead>
<tr>
<th>Counter</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>Feature Pattern Array</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>New Feature Array</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>5</td>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>-1</td>
<td>-1</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>-1</td>
</tr>
</tbody>
</table>

4.18 Incremental Feature Filling Technique

Incremental feature filling technique is used to extract feature information from nested features present on sheet metal parts. It involves filling up of features extracted in the current iteration of the feature extraction algorithm. The model is then regenerated with straight lines filling up the already extracted features. This is an iterative process and continues till no more features are present on the new generated model. The example below demonstrates this principle in detail.

![Figure 4.46 Incremental Feature Filling Explained for a sample part - 1.](image)

Consider the following example to demonstrate how incremental feature filling is used to detect the presence of nested features. Although the part shown above does not contain any nested feature, it is considered in order to explain how this technique is used. The
labels inside the periphery of the edges represent the index of the edges starting from 0 to 6.

Following is the information already known about the part before the feature extraction actually begins. The table below is used as a reference for the explanation below.

<table>
<thead>
<tr>
<th>Counter</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Feature Pattern Array</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New Feature Array</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Start Point Array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End Point Array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

New Feature Array is used to keep track of those edges, which form a part of some feature. Initially all the elements of the array are assigned to –1.

Now when the feature extraction module begins, a feature is found corresponding to the feature string 1-0-1 which represents a V-notch feature. The edges corresponding to the V-notch feature are 3 and 4 from the figure 4.46. This is when changes are made to the New Feature Array. For the edges found to be a part of the V-Notch, elements corresponding to the index of the edges are changed, i.e. in this case, index corresponding to the edges 3 and 4 are 2 and 3 (refer table above). So the elements in New Feature Array corresponding to these indices are assigned the value of the first index (referring to table this value is 2), i.e.

\[ \text{NewFeatureArray}(2) = 2 \]

\[ \text{NewFeatureArray}(3) = 2 \]

There is only one feature present in the part. Hence the New Feature Array after feature extraction looks like this.

<table>
<thead>
<tr>
<th>Counter</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Feature Array</td>
<td>-1</td>
<td>-1</td>
<td>2</td>
<td>2</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Before the system proceeds to construct a new part with filled features, first the New Feature Array is scanned. If all the elements of this array are still –1, it implies that no features were found in the earlier pass.

The construction of the new model of sheet metal part with features found in the current pass is described as follows.

The complete table is shown again for reference:

<table>
<thead>
<tr>
<th>Counter</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Feature Pattern Array</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New Feature Array</td>
<td>-1</td>
<td>-1</td>
<td>2</td>
<td>2</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Start Point Array</td>
<td>Start Point Co-ordinates of all the edges in the array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End Point Array</td>
<td>End Point Co-ordinates of all the edges in the array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

First create a new SolidWorks document and insert a new sketch by initializing the appropriate sketch plane. Then, starting from counter = 0, check to see the value of New Feature Array corresponding to this index. In this case, the value is –1, hence this edge is not a part of any feature found in the earlier pass, and hence this edge is constructed as it is. The type of edge (arc, straight line or circle) is known from the data structure used to store the edges and all the information required to draw this edge is already known. This edge is thus constructed in the new sketch. Similarly for counter = 1, the edge 2 is constructed. For counter = 2, the value of New Feature Array is 2. This implies that the edge corresponding to counter 2, which is edge 3 is a part of a feature. In order to find how many edges are parts of this feature, the New Array is scanned for all elements, which have a value equal to 2, which in this case is only upto counter 3. The value of the first counter is stored as firstIndex (value=2 from table) and counter 3 is stored as lastIndex (value = 3 from table). Thus all the edges corresponding to the indices between firstIndex and lastIndex form a part of this feature. These edges are not drawn in the new constructed part. Instead, the start point of the edge corresponding to the firstIndex and the endpoint of the edge corresponding to the lastIndex are joined together to form a straight line, which is then drawn in the newly constructed model. This is demonstrated in the figure below.
Construction of new model with edges 3 and 4 being replaced with an edge joining the end points, thus filling the feature.

For counter = 4 to 6, all the elements of New Feature Array are –1, hence all the other edges are constructed as they are. Hence the completely drawn sketch for the new model is as shown.

When the feature is filled, the new sketch may result in collinearities. But when this sketch is extruded as a sheet metal part, the CAD takes care of all such inconsistencies in
the sketch, merges 2, (3,4) and 5 edges in to one straight line and then extrudes the model as another sheet metal part, but with the V-Notch feature completely filled.

This new sheet metal part again forms input to the feature extraction module, we can call this as the second pass of the feature extraction module. For this part, the same steps are repeated as was done for the earlier part. They are outlined below.

![Figure 4.48 Incremental Feature Filling Explained for a sample part - 4.](image)

The edges are re-labeled in the above figure. The table for this part is shown below.

<table>
<thead>
<tr>
<th>Counter</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Feature Pattern Array</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New Feature Array</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Start Point Array</td>
<td>Start Point Co-ordinates of all the edges in the array</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End Point Array</td>
<td>End Point Co-ordinates of all the edges in the array</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Thus, the above table clearly shows that there are no features in the part, because there are no 0’s in the feature pattern array. Thus, the feature extraction module does not recognize any features in this pass. All the elements of New Feature Pattern array remain unchanged and equal to –1. Hence no new part is constructed and the feature extraction module ends.
Following example describes how this technique is used to extract a truly nested feature in multiple passes.

![Figure 4.49 Incremental Feature Filling Explained for another part - 1](image)

<table>
<thead>
<tr>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
</tr>
<tr>
<td>Feature Pattern Array</td>
</tr>
<tr>
<td>New Feature Array</td>
</tr>
<tr>
<td>Start Point Array</td>
</tr>
<tr>
<td>End Point Array</td>
</tr>
</tbody>
</table>

Note that the edge 5 occurs twice in the Edge array, because of the technique used to split an arc into two straight lines to find the feature pattern string.

The features extracted in the first pass are:

1. **V-Notch Feature: Edges 3-4**
2. **Arc Notch Feature: Edge 5**
3. **V-Notch Feature: Edges 5-6**

The modified New Feature Array is shown below after the first iteration.

<table>
<thead>
<tr>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Feature Array</td>
</tr>
</tbody>
</table>
These features are shown in the figure below.

Figure 4.50 Incremental Feature Filling Explained for another part - 2

After all these features are extracted, the new model constructed will be constructed by filling the features found in the current pass. This is shown as below.

Figure 4.51 Incremental Feature Filling Explained for another part - 3

The new part reconstructed will be as shown in Figure 4.52

Figure 4.52 Incremental Feature Filling Explained for another part - 4
The information for this part is shown in the table below:

<table>
<thead>
<tr>
<th>Counter</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Feature Pattern Array</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New Feature Array</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Start Point Array</td>
<td>Start Point Co-ordinates of all the edges in the array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End Point Array</td>
<td>End Point Co-ordinates of all the edges in the array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The feature extracted from the above model is:

1. **Straight U-Notch Feature: Edges 3-4-5-6**

This part forms the input to the second pass of the External Feature Extraction Module. The feature extracted in this pass is a Straight U-notch feature. The most important thing in this case is that none of these edges, which form the part of this new feature, existed in the original model. All these edges were formed by filling up the features that were found in the earlier model. Thus, this feature could not have been extracted in the earlier pass. Hence, a step-by-step analysis of features with feature filling technique helps in extraction of nested features.

![Figure 4.53 Incremental Feature Filling Explained for another part - 5](image)

Figure 4.53 Incremental Feature Filling Explained for another part - 5

The new constructed part and its details are shown below.
The edges are re-labeled in the above figure. The table for this part is shown below.

<table>
<thead>
<tr>
<th>Counter</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Feature Pattern Array</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>New Feature Array</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Start Point Array</td>
<td>Start Point Co-ordinates of all the edges in the array</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End Point Array</td>
<td>End Point Co-ordinates of all the edges in the array</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Thus, the above table clearly shows that there are no features in the part, because there are no 0’s in the feature pattern array. Hence the feature extraction module does not recognize any features in this pass. All the elements of New Feature Pattern array remain unchanged and equal to –1. Hence no new part is constructed and the feature extraction module ends.

4.19 External Features Classification Algorithm

The following flowchart represents the methodology for extracting external stamped features from the part.
INPUT:
FEATURE PATTERN ARRAY

START FROM THE CURRENT INDEX OF THE ARRAY
SEARCH FOR A ZERO IN THE ARRAY
STORE THE INDEX BEFORE ZERO AS FIRST INDEX
CONTINUE SCANNING THE ARRAY TILL A NON ZERO IS FOUND
LAST INDEX IS THE INDEX OF THE ONE VALUE AFTER THE LAST ZERO VALUE IN THE CHAIN
CALCULATE ENDINDEX - FIRSTINDEX + 1

EDGES CORRESPONDING TO FIRST INDEX ELEMENT AND LAST INDEX -1 ELEMENTS ARE THE SAME ARC
ARC NOTCH FEATURE

EDGES CORRESPONDING TO FIRST INDEX ELEMENT TO LAST INDEX -1 ELEMENTS ARE ALL STRAIGHT EDGES
V- NOTCH FEATURE

STRAIGHT U- NOTCH FEATURE

EDGES CORRESPONDING TO THE FIRST INDEX ELEMENT AND LAST INDEX -1 ELEMENTS ARE STRAIGHT EDGES AND THE EDGE CORRESPONDING TO FIRSTINDEX + 1 AND FIRSTINDEX + 2 IS THE SAME ARC
ARC U-NOTCH FEATURE

CUSTOM FEATURE

IS ARRAY INDEX LESS THAN ARRAY SIZE

EXIT

Figure 4.55 Flowchart for External Features Classification
4.20 Overall Feature Extraction Module in Detail

Figure 4.56 Flowchart for External Feature Extraction.
4.21 Extracting of Bend Features from the CAD model

Bend information from any bend feature is the Bend Radius and the angle of bend. This information can be obtained for each bend by traversing through the feature manager and obtaining access to the bend feature and then accessing the feature definition of the bend.

The Bend features can be classified as:

4.21.1 Base-Bends

These types of bends are created automatically by SolidWorks. When the starting profile for the Sheet metal part is an open profile which contains many sharp corners, and when this profile is extruded as a sheet metal, SolidWorks transforms the sharp edges into bends by introducing bend faces in the region.

4.21.2 Edge Bends

These types of bends occur when an Edge Flange is created.

4.21.3 Sketched Bends

This bend is created when an already existing face is to be bent from a line in the interior of the face. The line is first sketched on the face and the fixed face is selected and this results in the other half of the face being bent at the specified angle about the sketched bend line.

4.21.4 Miter Bends

These bends are created when an entire profile is to be bent, say a flange is to be created to an entire open profile. In this case, first the profile is sketched perpendicular to one of the edges in the profile and then the bend is propagated to the entire profile. This results in the number of bends equal to the number of edges in the profile.
In case of bends, all the information about the feature required is represented in the exact same form in SolidWorks. This information can be extracted from the SolidWorks API.

A handle to each of the bend features is obtained through the feature manager, which gives the bend radius and bend angle information for each bend.

4.21.5 Design Advisor for Bend Features

Bend Radius must be less than sheet metal thickness or 1.6 mm whichever is greater

A sample sheet metal part showing the extraction of the bend feature and design recommendation is shown in Figure 4.49.

Figure 4.57 Bend Feature Extraction
4.22 Getting the Total Bend Length of the part

The total bend length of the part is a useful input in the cost estimation of the sheet metal part. The methodology for calculating the total bend length of the part is as follows:

1. Get access to the underlying bend lines in the sheet metal part. Using the SolidWorks API functions obtain access to all the bend features and all the underlying bend lines. The next step is to calculate the length of the underlying bend line for every bend feature and then adding up the lengths to obtain the total bend length of the part.

Figure 4.58 shows a sample sheet metal part for which the total bend length is calculated.

![Figure 4.58 Total Bend Length of the Sheet metal part](image.png)
4.23 Other Information about the sheet metal part

Among the other information extracted from the sheet metal part is the information about the flat pattern face and its bounding box.

Details extracted are:

1. Area of the flat pattern face
2. Perimeter of the flat pattern face
3. Bounding box dimensions
4. Bounding box area
5. Bounding box perimeter

The area and perimeter information about the flat pattern face is extracted using the class structure, which is developed for representing a FACE object. The area is calculated using the API function provided by SolidWorks, which directly gives the area of the face. For calculating the perimeter of the face, perimeter of all the loops in the face is calculated which in turn is obtained by summing the length of the edges in that loop. Bounding box is directly obtained using API function, which gives the smallest enclosing box with sides parallel to the X, Y and Z-axes.

Figure 4.59 shows the extraction of base face details and bounding box details.
Figure 4.59 Extraction of Base Face and bounding box Details
4.24 Inter-Feature rules

Extensive research on Inter-Feature rules has been done by Shashikiran Hegde [13]. It is quoted here on a brief scale only for the sake of completeness.

Inter-feature rule checking involves checking the various design constraints between features. These constraints involve distance and location considerations. The module developed checks for the various distance constraints between features that might lead to design violations.

The following features are defined as standard features for the inter-feature rule checking module.

1. Holes (Round, Oblong, Single-D, Double-D and custom shapes)
2. Bends
3. Slots
4. Extruded Holes
5. External Edges
6. Forms.

The methodology used for inter-feature rule checking is as follows.

1. **Profile Offsetting:** - The profile for each feature, which consists of edges, is offset by a ‘safe distance’. These distances are defined for each type of feature based on the various design rules collected. This offsetting generates a closed offset profile. The inner features in a part are offset outwards (towards the boundary) and the outer profile of the part is offset inwards. The offset profiles define a safe region for each feature. If there are any design violations, these offset profiles will intersect.

2. **Intersection Checking:** - The second part of the module checks the offset profiles for interference. A recursive spatial sub-division technique, similar to a quadtree is used to check for intersection. This method is used to avoid the combinatorial explosion
that might occur while checking each edge in a profile against all edges in other profiles for intersection.

The following table gives us the safe design distances to be followed during a sheet metal part design, between various features. The distances are usually specified in terms of the thickness of the part t.

<table>
<thead>
<tr>
<th></th>
<th>Holes</th>
<th>Bend</th>
<th>Slot</th>
<th>Extruded Holes</th>
<th>External Edge</th>
<th>Forms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hole</td>
<td>2 t or 0.060,</td>
<td>1.5 t + r_b</td>
<td>2 t or 0.060,</td>
<td>3 t ~ 4 t</td>
<td>2 t or 0.060,</td>
<td>3 t + r_f = 4.5t~5t</td>
</tr>
<tr>
<td></td>
<td>whichever is greater</td>
<td>= 3 t</td>
<td>whichever is greater</td>
<td></td>
<td>whichever is greater</td>
<td></td>
</tr>
<tr>
<td>Bend</td>
<td></td>
<td></td>
<td>4 t if the slot is parallel to the bend</td>
<td></td>
<td></td>
<td>4t + r_f1 + r_f2 = 6t</td>
</tr>
<tr>
<td>Slot</td>
<td></td>
<td>2 t or 0.060,</td>
<td>3 t ~ 4 t</td>
<td>2 t or 0.060,</td>
<td>3 t + r_f = 4t~5t</td>
<td></td>
</tr>
<tr>
<td></td>
<td>whichever is greater</td>
<td></td>
<td></td>
<td>whichever is greater</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extruded Holes</td>
<td></td>
<td>6 t</td>
<td>4 t</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Edge</td>
<td></td>
<td></td>
<td>4 t + r_f1 = 5t</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Forms</td>
<td></td>
<td></td>
<td></td>
<td>8t + r_f1 + r_f2 = 10t</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Based on the above table, the profile of each feature is offset by a certain distances. The various offset distances d for each profile are listed below.

1. Holes: \( d = t \) or 0.030 inches, whichever is greater.
2. Bends: \( d = 3 \) t or 3 * 0.030 inches, whichever is greater.
3. Slots: \( d = t \) or 0.030 inches, whichever is greater.
4. Extruded Holes: \( d = 3t \) or \( 3 \times 0.030 \) inches, whichever is greater.
5. External Edges: \( d = t \) or 0.030 inches, whichever is greater.
6. Forms: \( d = 5t \) or \( 5 \times 0.030 \) inches, whichever is greater.

The offset distances are so defined such that, if distance considerations between features are not satisfied, the offset profiles will intersect. These offset distances are based on the rules collected in the literature and can be easily modified according to the company standards and design practices followed.
5 Conclusions and Directions for future research

This chapter presents the conclusions of this research and suggests some directions for future research.

5.1 Conclusions

5.1.1 Comprehensive collection of rules

The design advisory system incorporates a comprehensive repository of design rules and recommendations. The collection of rules has been very extensive and includes all areas of sheet metal design with both Intra-Feature and Inter-Feature rules. These rules form the Knowledge Base for the Feature Extraction and Design Advisor systems built in SolidWorks. They have the capability to extract some of the features from modeled sheet metal parts and check for design violation and also suggest design recommendations.

5.1.2 Reduced Cycle Time for bringing products to market

Without such a DFM system, the designer would not have knowledge of these rules resulting in slower cycle times for bringing products to market. The enterprise priorities to deliver new products in response to competition are hampered if these needless bottlenecks of manufacturing intelligence information are not removed. The proposed design advisory system provides an extensive framework for successfully overcoming all these problems. The framework is scalable to incorporate an unlimited content of design and manufacturability advice and does not limit the designer to mere checking of rule violations.
5.1.3 Customizability

The recommended practices for each process and the process capabilities are a very significant factor for the successful manufacture of defect free parts. This information can be adapted to the specific job shop depending on the type of machines that are used.

5.1.4 Defect-Free Cost-Effective Method

This means that during the initial design of the part, the designer can have all the information required for manufacture of the product right at the design workstation computer and reliably consult the design advisor for lessening the iterative process of design. Thus, the new framework provides for the application of the proposed design advisory system along with other manufacturing knowledge to be a true advisor to the designer ensuring that once a drawing is sent to the shop for manufacture, it will be produced immediately in a cost-effective defect free way.

5.1.5 Scope for integrating with other modules

The present system also leaves a number of threads for the other modules, which can be developed and incorporated into the existing system.

1. Nested Feature Extraction module gives the convex hull of the outer shape of the sheet metal part as the final output, which can be used as an initial estimate for the Blank Layout Module.
2. Feature Information stored in the output file could provide useful information for the Process-Planning Module.
3. Information about each of the internal stamped feature and bend feature is also stored in the output file. The dimensions of the standard features and the perimeter of the non-standard feature are stored in the output file. This information can provide useful input to the Cost Estimation Module.
5.2 Directions for Future Research

This framework and feature extraction and design advisory system should be a first step for developing process planning system and costing system of sheet metal parts. The relative costs of two parts should be easier to determine by applying the recommendations and checking for violations to arrive at a manufacturing score for each part. This is a very promising path to investigate. Feature information provided by the Feature extractor system could serve as input to both the systems and heuristics could be developed to get the process plan for the system. Also once the part layout and process planning is done, cost estimation system can be used to give an estimate of the sheet metal part in the design stage itself.

Die design is also a very creative field with several possible designs for the same sheet metal part. The design recommendations for die design would translate into manufacturability rules for the sheet metal part and several rules in the current design advisor are a result of this association.

As proposed in the framework, an inter-process approach can be taken for manufacturability of sheet metal parts. An example of this is bending sequence determination where a part might be infeasible due to the chain of operations required. Thus, if process plan generation is combined into the solid modeler, then a part of fewer features might be more expensive compared to a part with greater number of features due to the number of operations required in the interprocess sequence. An array of holes can be punched cheaply in a single blanking and punching operation compared to a part with a few precisely located mounting hole for bent parts that would need a blanking, bending
and punching with alignment operation. This approach holds a lot of promise for cost reduction strategies and is a direction for future work.
References


Appendix

1. Description of modules developed in the implementation

1.1. Wrapper Classes for B-Rep Data

1.1.1. CVertex

This is a wrapper class for Vertex Object defined in SolidWorks. It provides interfaces to the underlying properties, which are not easily accessible using the Vertex Object directly from SolidWorks. Like other wrapper classes, CVertex does not contain a Vertex Object defined in SolidWorks because; there are some entities like Circle, the center, which is not represented as a Vertex Object in SolidWorks.

1.1.2. CFace

This class is a wrapper class for the SolidWorks Face Object. This class has access to all the underlying information such as the underlying loops, edges, curves and vertices. It also contains the Face Object as a member and it is used to set all the other members of the class.

1.1.3. CLoop

This class is a wrapper class for the SolidWorks Loop Object. This class has handles to underlying information such as edges and vertices of the Loop. It also contains the SolidWorks Loop Object as a member variable, which sets all the member values of the class.

1.1.4. IEdge

This is an interface, which captures the general characteristics of any curve irrespective of its type such as Straight Line, Arc or Circle. These general parameters are the type of curve, length of the curve, the SolidWorks Edge Object, the Start and End Parameters,
and if the curve is closed or open. These general properties of any type of curve are captured in this Interface. Then there are classes which implement this interface like C StraightLine, CCircle and CArc, and add and set the other properties of the object such as Start Vertex, End Vertex for a Straight Line; Start Vertex, End Vertex, Radius and Center for an Arc and Radius and Center for a Circle.

1.1.5. C StraightLine

This class implements the IEdge class. Other properties of a straight line such as Start Vertex and End Vertex are set in this class. Since Visual Basic does not support implementation inheritance and only supports interface inheritance, inclusion is used. An object of class IEdge is included as a member of the class and all the common properties of the Edge are set to the member variables of the object of the class IEdge.

1.1.6. CCircle

This class implements the IEdge class. Other properties of a circle such as Center and Radius are set in this class. Since Visual Basic does not support implementation inheritance and only supports interface inheritance, inclusion is used. An object of class IEdge is included as a member of the class and all the common properties of the Edge are set to the member variables of the object of class IEdge.

1.1.7. CArc

This class implements the IEdge class. Other properties of an arc such as Start Point, End Point, Center and Radius are set in this Class. Since Visual Basic does not support implementation inheritance and only supports interface inheritance, inclusion is used. An object of class IEdge is included as a member of the class and all the common properties of the Edge are set to the member variables of the object mIEdge.
1.2. Stamped Feature Class Modules

1.2.1. Base Class Module: IFeature

This is a dummy interface that is created so that both internal features and external features can be implemented as classes from this base class.

1.2.2. Internal Stamped Feature Class Modules

1.2.2.1. CCircleFeature

This class implements the IFeature Interface. This is used to store feature information about an Internal Circular-Stamped Feature.

1.2.2.2. CRectangleFeature

This class implements the IFeature Interface. This is used to store feature information about an Internal Rectangle-Stamped Feature.

1.2.2.3. CSingleDFeature

This class implements the IFeature Interface. This is used to store feature information about an Internal Single-D Stamped Feature.

1.2.2.4. CDoubleDFeature

This class implements the IFeature Interface. This is used to store feature information about an Internal Double-D Stamped Feature.

1.2.2.5. COblongFeature

This class implements the IFeature Interface. This is used to store feature information about an Oblong Stamped Feature.
1.2.2.6. CCustomFeature

This class implements the IFeature Interface. This is used to store feature information about an Internal Custom Stamped Feature.

1.2.3. External Stamped Feature Class Modules

1.2.3.1. CArcNotchFeature

This class implements the IFeature Interface. This is used to store feature information about CArcNotch Stamped Feature.

1.2.3.2. CVNotchFeature

This class implements the IFeature Interface. This is used to store feature information about CVNotch Stamped Feature.

1.2.3.3. CStraightUNotchFeature

This class implements the IFeature Interface. This is used to store feature information about CStraightUNotch Stamped Feature.

1.2.3.4. CArcUNotchFeature

This class implements the IFeature Interface. This is used to store feature information about CStrightUNotch Stamped Feature.

1.2.3.5. CExternalLoopCustomFeature

This class implements the IFeature Interface. This is used to store feature information about External Loop Custom Stamped Feature.
1.3. Main Code Modules

1.3.1. MainModule

1.3.1.1. MainProcedure

This procedure calls all the individual modules, which extract all the information from the Sheet metal part.

1.3.2. FileOutputModule

1.3.2.1. WriteToFile

This procedure takes in a string as argument and writes it to the file, which is initialized using the procedure InitializeFileWriter, which is described below.

1.3.2.2. InitializeFileWriter

This procedure is used to write all information related to the features in the part to a text file. It creates a text file named FeatureFile.txt and opens a text stream to the file for writing.

1.3.3. InitializeSolidWorksModule

1.3.3.1. InitializeSolidWorks

In this function, all the start up work is done. A handle to the active SolidWorks Object is obtained through OLE interface provided by VB. Then the feature Flat-Pattern is accessed and the base face and face normal are obtained.
1.3.4. FlatPatternModule

1.3.4.1. GetFlatPatternData

This function calls the two private functions declared in this module that are explained below and gets the data from them. It then writes the data to the text file using the WriteToFile procedure explained above.

1.3.4.2. BaseFaceData_ToString

This function gets dimensional data about the base face such as the area and perimeter and returns the information in the form of a string.

1.3.4.3. BoundingBoxData_ToString

This function calculates the dimensional data about the bounding box for the base face and finds the area, perimeter and the length and width of the smallest rectangle which surrounds the part, whose sides are parallel to two of the major axes.

1.3.5. InnerStampedFeaturesModule

1.3.5.1. ExtractInnerStampedFeatures

This procedure first extracts every loop from the base-face except the outermost loop of the face. Then for every loop it calls the function GetInternalStampedFeatureType to determine the type of feature. Then it writes information about the feature to the text file using the WriteToFile method. It calls the function DesignAdvisorInternalFeatures_ToString which checks the design rules on the part and gives recommendations in the form of a string, which this procedure writes to the text file. These steps are repeated for every loop present in the base-face except the outermost loop.
1.3.5.2. FeatureInformation_ToString

This function first determines the type of feature from the argument passed to it, then it creates an object of that feature type and calls the To_String member function of that class which gives all the information stored about that feature.

1.3.5.3. DesignAdvisorInternalFeatures_ToString

This function checks all the design rules depending on the type of the feature and returns all the information and design recommendations in the form of a string to the function or procedure that calls this function.

1.3.5.4. GetInternalStampedFeatureType

Depending on the rules followed for classifying the Internal Stamped Features, this function tries to classify each of the loops into one of the standard features defined in the library and categorizes it as a Custom Feature if it does not fit into one of these standard types.

1.3.6. OuterStampedFeaturesModule

1.3.6.1. ExtractOuterLoopFeatures

This function first extracts the outermost loop from the base-face. Then it calls FormFeaturePatternArray, FormEdgePatternArray and FormEndPointArrays to initialize the respective arrays. Then it initializes the NewEdgePatternArray with all values to –1. It then calls ClassifyOuterLoopFeatures, where the features are extracted and classified and the NewEdgePatternArray is also updated. It then scans the NewEdgePatternArray and returns a value false if all the elements of the array are –1, which means that there are no more features in the new part which could be constructed. It returns a value true otherwise.
1.3.6.2. CreateNewEdgeArray

This procedure first initializes a new SolidWorks part, then makes that part as the active part. It then finds the plane of the part to draw in and initializes a sketch on that plane. Then it scans the NewEdgeLocationArray and depending on its value calls either DrawEdge to draw the edge from Loop1 as it is or DrawEdge2a to draw an edge corresponding to incremental feature filling for the particular feature. It thus generates a new part where all the features found in the earlier part are completely filled with straight lines.

1.3.6.3. DrawEdge2a

This procedure is used to draw an edge in the new part, which corresponds to filling of the feature extracted in the earlier pass. The startEdgeIndex, endEdgeIndex and the startPointArray and endPointArray, which are passed to the procedure as parameters are used to find the start point of the feature and the end point of the feature, which are connected by a straight line. These two points are used to draw the straight line in the new part.

1.3.6.4. DrawEdge

This procedure is used to draw an edge as it is in a new sketch. This is used when a new part is created using Incremental Feature-Filling technique. This procedure first determines the plane that the edge was currently in and then draws the new edge in the corresponding plane.

1.3.6.5. DisplayNewEdgeLocationArray

This procedure is just used to display the NewEdgeLocationArray, which is passed to it as an array in the form of a string.
1.3.6.6. FormEndPointArrays

This procedure sets the values for the startPointArray and endPointArray, which are passed to it as parameters. For every pair of consecutive edges in the Loop, which is passed as parameter to the procedure, it calls GetEndPointsOfEdges module and thus the entire array of start points and end points is set.

1.3.6.7. FormFeaturePatternArray

This function forms the FeaturePatternArray. For every pair of edges in the Loop, which is passed to the procedure, it calls the procedure FormFeaturePattern, which sets the feature pattern values for the corresponding edges. Thus it forms the entire Feature pattern array, which is used by the ClassifyOuterLoopFeatures module for classifying the external features.

1.3.6.8. FormEdgePatternArray

This procedure sets the value of the EdgePatternArray, which is used as input for the ClassifyOuterLoopFeatures module.

1.3.6.9. SetNewEdgeLocationArray

This procedure takes the start and the end indices and changes the NewEdgeLocationArray elements having indices in that range to the value, which is also passed as a parameter to the procedure.

1.3.6.10. ClassifyOuterLoopFeatures

This procedure scans the featurePatternArray for the presence of features using the rules for external feature extraction and then classifies the extracted features into one of the standard external stamped features defined in the library. It also sets all the properties of the extracted feature by calling the SetFeature for that particular Feature Class. Then it
dumps all the information about that feature to the text output file. Then it checks the feature for design rules and also dumps all the information provided by design advisor to the text file. Also it modifies and updates the NewEdgeLocationArray so that the new part constructed by incremental feature filling can be performed.

1.3.6.11. FormFeaturePattern

This procedure first calls the procedure GetEndPointsOfEdges, which aligns the end points of the edges such that the orientation of both the edges is counter clockwise. Then depending on the types of edges it forms the feature pattern. It calls the function FormVector to form the vectors along the straight lines or the approximated chords in case of arcs. It then updates the featurePatternArray, which is passed to the procedure as a parameter.

1.3.6.12. GetEndPointsOfEdges

This procedure realigns the start points and end points of the edges depending on how the orientation of the edges is. This is done because some times the edges in the loop are aligned in a clockwise sense with respect to the other edges in the loop. Since the orientation of edges is very important for evaluating the feature string, the realignment of edges is necessary. This function just takes the two edge objects as inputs and finds the correct aligned start points and end points of the two edges and dumps the values in the variant variables, which are passed as parameters to this procedure.

1.3.6.13. DisplayFeaturePatternArray

This procedure displays the FeaturePatternArray, which is passed as a parameter to the procedure as a string.
1.3.7. GeneralModule

1.3.7.1. HeaderFormat_ToString

This function just formats the string as a Header. This is done before writing the result to
the output file so that the text stands out among others to identify a header in the text file.

1.3.7.2. FormatToMM

This function formats the variant passed to it as parameter by converting it from meters
to millimeters and formatting it to FIXED type. It returns the formatted value a string.

1.3.7.3. GetCurveType

This function takes the edge Object as a parameter and determines the type of edge, i.e.
whether it is a straight line, arc or a circle. Global Constants have been declared for each
type of edge as integers i.e swStraightLine, swArc, swCircle, which is the value returned
by this function for the corresponding type of edge.

1.3.7.4. CheckCrossProduct

This function gets two edges as parameters and returns a boolean variable indicating
whether vectors along the two edges are parallel or not. It creates vectors along the two
edges and then gets the value of the cross-product of the vectors using the Math Utility
provided by SolidWorks API functions. Then it determines the value of the magnitude of
the cross-product and if the value is 0, i.e if the two vectors are parallel, returns a value
true. Otherwise, it returns false.

1.3.7.5. CheckDotProduct

This function gets two edges as parameters and returns a boolean variable indicating
whether vectors along the two edges are perpendicular or not. It creates vectors along the
two edges and then gets the value of the dot-product of the vectors using the Math Utility provided by SolidWorks API functions. Then it determines the value of the dot product and if the value is 0, i.e if the two vectors are perpendicular, returns a value true. Otherwise, it returns false.

1.3.7.6. AngleOfArc

This function returns the value of arc angle in radians by taking the difference between its startParam and endParam values.

1.3.7.7. CompareVertices

This function compares two vertex objects, which are passed to it as parameters and returns a value true if both vertices are same and false otherwise. This function is required as a special function because of the numerical approximations, which arise when the values of co-ordinates are stored. A tolerance of 0.000001 is set for the difference between the corresponding co-ordinates of the vertices and if the limit is satisfied, the function returns true.

1.3.7.8. GetAngleBetweenVectors

This function calculates the angle between two vectors where the direction values of the two vectors are passed as arrays of size 3 to the function. It calculates the angle between the two vectors by dividing the dot product of the two vectors by the magnitude of the two vectors and then taking the inverse cosine of the value obtained.

1.3.7.9. FormVector

This function forms the vector along the line joining the point1 and point2, which are passed as variant parameters to the function. This function forms a vector Object and returns the value to the calling function.
1.3.8. BendFeaturesModule

1.3.8.1. ExtractBendFeatures

This procedure scans all the features stored in SolidWorks in the Feature Manager and extracts all the bend features. All the information about a bend feature is already stored in SolidWorks, hence this information just has to be extracted. Once these features are extracted, the procedure checks these bend features with the design rules. The procedure then dumps all the feature information and the design advise into the text output file using WriteToFile procedure. It then calls the procedure GetTotalBendLength to obtain the total bend length of the part.

1.3.8.2. GetTotalBendLength

This procedure accesses a feature called “Bend-Lines1” which gives access to the profile feature which has information about all the bend lines stored as an array of the start and end point co-ordinates of the lines. From the end point information, the length of each of the bend lines is calculated and then they are added to get the total bend length of the part.

1.3.9. swConst

This module stores all the constant variable declarations used by SolidWorks. Some of the new constant variables, which have been declared and used in the code have also been declared over in this module.