International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:11, No:4, 2017

Towards a Complete Automation Feature Recognition System for Sheet Metal Manufacturing

Bahaa Eltahawy, Mikko Ylihärsilä, Reino Virrankoski, Esko Petäjä

Abstract**—**Sheet metal processing is automated, but the step from product models to the production machine control still requires human intervention. This may cause time consuming bottlenecks in the production process and increase the risk of human errors. In this paper we present a system, which automatically recognizes features from the CAD-model of the sheet metal product. By using these features, the system produces a complete model of the particular sheet metal product. Then the model is used as an input for the sheet metal processing machine. Currently the system is implemented, capable to recognize more than 11 of the most common sheet metal structural features, and the procedure is fully automated. This provides remarkable savings in the production time, and protects against the human errors. This paper presents the developed system architecture, applied algorithms and system software implementation and testing.

Keywords**—**Feature recognition, automation, sheet metal manufacturing, CAM, CAD

I. Introduction

SHEET metal products vary with their design complexity from very simple ones as with simple cuts or holes, to complicated ones which include nested features (feature means topography or structure in this context) that host other features in other directions [1]. This in turn puts a stress to the design phase. Previous research showed that the design phase can consume up to 70% [2] of the total production cycle. Moreover, at least 50 to 60% of the time consumed by design is actually lost with other tasks related to the design and handling processes rather than the design work itself. It is believed that the design phase and any required modifications are the most costly ones of the production cycle [3], and thus many assisting tools have been suggested to fix for the lost time and effort within this phase.

Computer-aided Drafting (CAD) and Computer-aided Manufacturing (CAM) are typical examples of these tools [4]. CAD software products are used to optimize design and to reduce the design phase time. Many tools exist under this category to suit companies of different sizes, for instance AutoCAD, TurboCAD, CATIA, Solidworks, Inventor, etcetera [5] [6]. Some CAD software would work with different 2d designs, and then they get combined and converted to 3d models during the manufacturing phase, while other products have the ability to work with 3d from the beginning. Minor differences exist between the different suites, and they all are known as being processor, memory and graphics intensive applications that require high specifications. Still in real situation there can be a significant time lost due to lagging, processing, converting, and related file tasks. The other category is CAM which is used to assist the manufacturing process and its related tasks, e.g. planning, storage, tooling, etc. Normally, the work flow starts with CAD, then CAM, and ends with implementation, which compared to previous tasks takes the least time.

Many issues arise within the design phase and need to be considered. Firstly, it is assumed by default that designers know the properties of the products they are working with, sheet metal specifications, and machinery limitation. Another important task for designers to do is to document the design and its features to be ready for manufacturing. These features include but are not limited to thickness, diameter, direction, position, and angle. However, for the bending factor, also known as k-factor, it is usually not known at the design phase since it depends of the bending tools that will be used to for bending. Documentation is one of the most time-consuming processes, because of the amount of details it may include. Moreover, any further modifications that are required due to any limitation will turn the process back to the design phase to start over, and then to documentation after. A very important factor to pay attention to as well is the individual experience and the design's dependability on the designer's style himself. Designers know more details than they document about their own work, and different individuals have different design approaches. This in turn may cause problems with complicated designs if they would be handled by others. Also any missed details in documentation may cause manufacturing errors and render parts waste material. All these reasons indicate the need of sophisticated feature recognition. Feature recognition systems would provide the required assistance during the design phase, since they perform documentation and guidance for the control of the sheet metal processing machine, while including the pre-defined specifications. Many feature recognition systems already exist in different companies that work in the sheet metal industry, trying to simplify the design phase to its basics to speed up the design process. Human intervention within the recognition phase still cannot be avoided, and machines are unable to perform all feature-based tasks completely. Moreover, any shortage or required modification during the final stages would return designs for further processing. In this work with Prima Power we are completing the feature recognition shortages in the existing system. We target to increase the automation level, to

Bahaa Eltahawy and Reino Virrankoski are with the University of Vaasa, Department of Computer Science, P.O. Box 700, FI-65101, Vaasa, Finland (e-mail: {bahaa.eltahawy, reino.virrankoski}@uva.fi).
Mikko Ylihärsilä and Esko Petäjä are with the Prima Power Oy, Metallitie 4, FI-62200, Kauhava, Finland (e-mail: {mikko.yliharsila, esko.petaja}@primapower.com).

reduce the human error, processing and tooling time. One key target is to improve the production process so that the separation between design and manufacturing phases becomes clear. We are doing so by introducing a system that can recognize all typical features with all possible variations and factors for the various CAD-models, thus the manufacturing phase goes totally independent of the design phase. In turn this will save the time required for documentation, tooling and other models' handling tasks. This speeds up the design phase by at least 50% and creates a better independent feature recognition system.

In this paper we introduce our solution for sheet metal feature recognition from the model. Hereby it includes the system, the applied feature types, challenges and limitations we faced during our work, and the ways how we solved them. We introduce the main feature recognition rules and implementation schemes, and finally we provide some study cases with different levels of complexity.

II. The System

The system consists of following entities: CAD system, feature recognition system, and design and testing library. Due to its robustness and flexibility to provide a wide range of functions to utilize, our CAD solution is Siemens PLM NX® (https://www.plm.automation.siemens.com/) modeler [7] [8]. This suite allows us to perform the CAD and CAM functions, and it also provides tooling, handling, programming, processing, cutting, and many other properties that are beneficial to our implementation. The other part of the system is Microsoft Visual Studio® which is an Integrated Development Environment (IDE) (https://www.visualstudio.com/) that we use to program our recognition system, i.e. Feature Recognition Engine©® (FRE) [9]. The third part of the system is the intensive library that we have created to study all considered features, the testing library has been created using SolidWorks® (http://www.solidworks.com/). Finally, the system output is the implementation sheet with the found features, their specifications and the required tooling, thus making the imported design ready for the manufacturing stage.

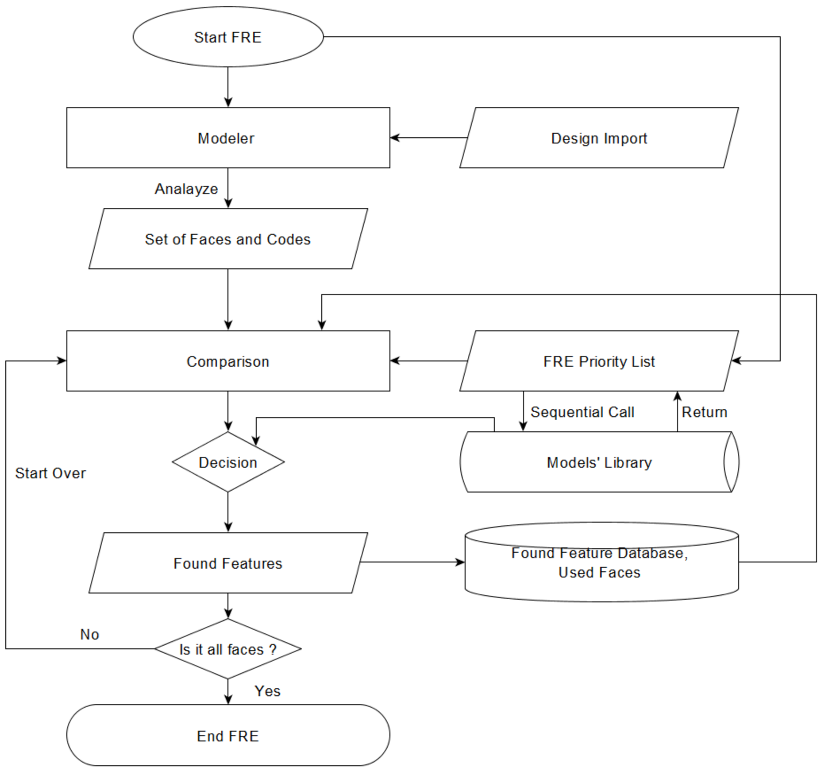


Fig. 1 The flowchart of the developed Feature Recognition System (FRE)

Feature recognition flowchart is presented in Fig. 1. Here FRE system upon importing a certain CAD-design launches the modeler system, to visualize the design and to open it in memory. FRE system will analyze the CAD-design and convert it to a set of faces with a set of codes. Recognition is done by performing a thorough comparison with the pre-defined feature models that are stored in the model library. To avoid conflicts, certain features are given higher priority than others, so that features do not get recognized mistakenly as the most common ones. For instance, louver feature without prioritization will be recognized as a set of bent features. For that reason testing will be done for features with the higher priority at first. FRE therefore starts by knowing which feature it should test at first by consulting the priority list, and then after it performs a call to its model library. Here, FRE will perform all the required comparisons. Matching is done between the input design's found features and the feature models stored in the model library. Once matching is successful, extra tests for validation are performed to ensure a correct identification of a feature. If a feature is found, FRE then will save all the found faces of the found feature with their characteristics in the Found Feature Database (FFD). If matching is unsuccessful, FRE brings the next feature to study from the priority list, then repeats the process to study if such feature exists within the model upon study or not. The process is repeated as many times as needed to recognize all features in the processed model.

III. Types of Features

In the context of sheet metal processing, the word "feature" means topography or structure. As presented in Fig. 2, features [10], [11], [3] exist due to the different methods using within manufacturing, for example with cut, draw, bent, and out of stretching. Many features would exist due to such variations, but in our work we rather are focusing on features that take an offset shape out of the sheet metal plate. Namely, we concern holes, extruded holes, cuts, flanged holes, louvers, lanced bridges, dimples, embosses, all forms of beads, and all forms of bends, and finally nested features [12]. The following 11 features are illustrated in Fig. 2.

1. Hole: It is merely a round hole within the sheet metal. A hole can take different shapes depending on the usage purpose; it can be blank hole as in Fig. 2 A, with fillets as in Fig. 2 C, with chamfers as in Fig. 2 B, or even conical or fully rounded fillet as in Fig. 2 B.
2. Extruded hole: This one is an offset from the metal surface, it normally is of 0.5 cm height, but this can change according to the purpose. Extruded holes share the same variations with simple holes; they can have fillets or chamfers, also they can be cylindrical as in Fig. 2 F or conical as in Fig. 2 E.
3. Cut: They are holes that can take any shape depending on the design. For example they can take rounded rectangular form as in Figs. 2 G-H. Both holes and cuts are the same concept, but what makes difference between them is the ratio and dimensions that form the cut itself.
4. Flanged holes: Similarly as extruded holes and cuts. A flanged hole is an offset from the metal surface that can take any other shape rather than the cylindrical one, as in Fig. 2 I.
5. Louvers: A louver is a window that is protected from 3 sides and open from the forth to allow for air passage, it is manufactured as a punch feature. A louver takes normally the shape presented in Fig. 2 J shape, but it can also take the spherical shape as in Fig. 2 K.
6. Lanced Bridges: It is another punch feature, where the punching tool will form a bracket-shaped "]" form that is offset of the sheet plate. Normally bridges take the form in Fig. 2 L.
7. Dimples: A dimple is a stretched offset feature made to the sheet plate, taking the shape of counter sink of a hole. Dimples can be rectangular, rounded-rectangular shape, or they can go non-uniform taking random shapes, these are presented in Figs. 2 M-O.
8. Embosses: Emboss is the same idea as dimple, but rather it is spherical, as shown in Fig. 2 P.
9. Beads: A bead is formed by pinching a metal piece placed between dies, thus the plate takes the shape of the given metal piece. Beads can take several forms as in Figs. 2 Q-S. Also in these figures beads are straight-shaped form, while they also can take a circular form.
10. Bends: A simple bend is presented in Figs. 2 T, V-X. Bends can also be nested which means a bend following a bend, as in Fig. 2 U.
11. Nested features: They can include features that host other features. Fig. 2 Y is an example of nested features. In this figure, the sheet metal plate hosts a dimple feature, that hosts a lance bridge and louver features as well, moreover the hosted bridge feature itself hosts an extruded hole feature on its upper plane.

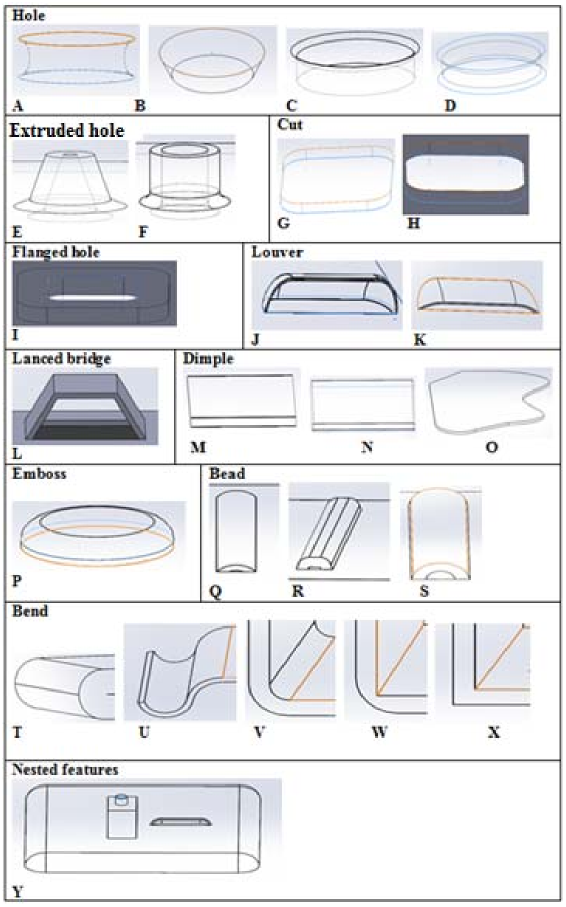


Fig. 2 The set of considered features in this work

IV. Difficulties

The main difficulties in feature recognition are related to designs' inconsistency and common similarities. Firstly, designs have many variations, which need to be considered during modeling. This in turn increases the complexity of the modeling scenario, also the algorithm becomes computationally heavy to execute once the number of combinations increases drastically when all variations must be computed. For example, a very simple hole can be merely a hole, a hole with upper fillet, a hole with lower fillet, a hole with double fillets, a hole with upper chamber, a hole with lower chamfer, a hole with double chamfers, a hole with fillet and chamfer, a conical hole, or a fully rounded fillet-shaped hole. Using coding convention a hole can change its state from 00 to 11, where a digit represents existence of a pattern. Considering that rounding can be either fillet or chamfer gets the result doubled, and by taking into account 2 special cases, a simple hole can take 9 different forms at the end. Similarly, extruded holes can go from 0000 to 1111 with one form, which means $2x2^4$ -1 = 31 different forms at the end if rounding forms are not mixed. With the mixing case however we should consider 3^4 = 81 different variations. For the purpose of complete feature recognition implementation, we have considered all cases that could be available/ applicable. Table I shows the number of variations considering only fillets as rounding pattern. It can be seen from that table that if rounding patterns and other special cases are mixed, then the total number of cases will be at least doubled, i.e. around 1200 recognition cases at least.

The other challenge we have encountered is the similarity that some models share. For example, a basic simple hole can have only one transition, which is a cylinder, a rounded one. A bend as well can only have one rounded curve in a cylindrical form. This in the beginning could cause a lot of confusion, as the system would mix between holes and bends. The same case exactly occurs with other features, for example the louver model in its simplest form has only 3 transitions, in the same time one of the many variations of the bridge model could take a wrong short-cut and come to end-state only after 3 transitions. This will simply show a bridge as louver mistakenly. This can be solved by creating many rules and validators to force the system to go only in a certain direction when searching for a specific feature. Similarities errors are shown in Fig. 3.

TABLE I
NUMBER OF VARIATIONS FOR SELECTED FEATURES

Simple hole	4
Extruded hole	16
Louver	4 + 4 = 8
Lance bridge	256
Dimple	16 x 16 = 256
Bend	4
Bead	16 + (16 x 2) + (16 + 2) = 80 for straight beads, similarly with rounded beads
Emboss	16
Total	636 **simple** cases

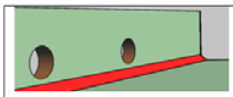
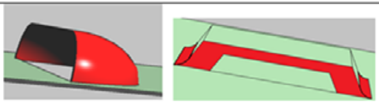
	**Bend and Simple hole recognition error**	**Here, the logic for bend and simple hole is: Wall face >>Cylinder >>Wall face**
	**Louver and Lance bridge recognition error**	**Here, the logic for louver and bridge is: Wall face >>Cylinder >>Planar face >> Cylinder >>Wall face**

Fig. 3 Features with common logical recognition errors

V. Feature Set-Up

In order to make successful recognition, the following factors have been considered:

1. The main surface: The recognition algorithm will at first perform a thorough analysis to test all surfaces, to find the surface that will be treated as "the main surface". This surface is found by considering that the surface that has the most number of features hosted within, to be the main surface. This main surface will be connected to the feature nesting as shown hereafter.
2. Surface type: Distinction between surfaces has been done. Surfaces are classified to planar, cylindrical, conical, spherical and torus surfaces. If a surface does not follow the common rule for classification, it is tagged merely as a "surface".
3. Convexity: Many features can get recognized mistakenly if the searching algorithm does not include the convexity of the body it is searching for. Features can exist inwards or outwards, and also the directionality with the main surface will play a factor here. When considering the angle of search and the orientation, we could refine the search to a minor number, so that after the right validators the correct feature will be recognized. The algorithm can search for following convexity types: convex, concave, or when it is not sharp then it will be smooth convex, smooth concave, or when it is beyond certain angles then it is knife convex or knife concave, or when inflecting in the other direction then it searches for inflection.
4. Directionality: For simplicity recognition has been designed to recognize features in only direction, which is upwards. For efficiency the algorithm will run till the end state, then it will perform surface switching, in which the upper face will be considered the bottom one and the bottom face will be considered the upper face. At this stage, the algorithm will perform a thorough scan to find features that could not get recognized in the earlier phase. Finally the output sheet will specify the location of the found feature and its exact direction.
5. Prioritization of features: It is very important to run the algorithm inherently on several stages instead of

performing a one-shot run. Features share many common characteristics, and this might lead to either recognizing a wrong feature, duplication of results, or recognition failure. For instance, bends, louvers, beads, lance bridges or other features may start with cylindrical face in the form of a fillet, and without prioritization the wrong feature will get recognized. As illustrated in Fig. 3, a louver will have a higher priority than a bridge, and also a hole will have a higher priority than a bent, which in fact would have the lowest priority among all other features.

6. Nesting: Features still can exist on other surfaces rather the main one and its opposite direction replica. In other words the features can host other features within one of their faces. After finishing with the main features the algorithm will switch to features individually then treats them as sequence of sub-surfaces to avoid the loss of nested features. Recognition will run on surfaces individually. If other features are found then it will compare them to the main found ones to determine if it is a new feature or just a replica of an already found feature. Nested features will get included within the output, which will include all their characteristics exactly as in the case of main ones.
7. Tolerance: To optimally find all features under study, different tolerance mappings have been considered. It is known from real implementation that different features have different sizes that they exist within. Some features can accept ±1 mm while others can go to 0.001 mm. For that we specified different tolerance mappings according to the feature under study and the tolerance value it can accept. Mathematical explanation is at [13].
8. Forks: To find all faces within a certain feature, the algorithm needs to leave the main route in favor of searching in other directions. For that reason we have included a function called "Fork" that would perform path separation. Normally recognition goes in a certain direction; this will work with features that do not experience similarity from two opposite sides while being in the same face. Other features like louver or lanced bridge require the recognition system to get out of its recognition direction at a certain point to find other surfaces. After finding those side surfaces, the recognition algorithm returns back to its main direction with the found information to be included for successful feature recognition. The return is governed by another function called "Join". Join function combines paths that got separated before, to direct them to the main drawn route again. Unlikely forks suffer from being considered as a minor route compared to the main route, which means that if two features with the same priority would exist then forks will mostly get ignored. As illustrated in Fig. 3, a bridge got recognized as a louver, which is wrong. This was corrected by applying forks in the most minor tracks rather than main ones. For example with the case of louvers and bridges, forks are only considered for fillets, not main bodies.
9. Line and vertex recognition: These are special cases. Typically the recognition algorithm searches for surfaces and bodies. However, in some certain cases this is not applicable. In those cases one needs to search for a line, for example, and then treat it as a body. In some other type of cases there is even no line but a point, vertex, connecting between two surfaces that do not share any common bodies but this single point. Link and vertex recognition is required when dealing with bends and with many of the special cases of dimples.
10. Non-uniform patterns: As mentioned in pact 2, all surface types are uniform, except the ones we explicitly refer as "Surface". The surfaces can take any form and they will not abide with any rules. The problem with surface bodies that the recognition can find one of its faces but not both of them, because as mentioned none of the normal rules can be applied. However, we can tweak this shortage only by considering bodies' thickness. If this is constant despite the body type, the other face will be found. This is shown later when the rules are presented in the next sections.
11. Design constraints: Even though we are targeting to create a system that can deal with all possible cases, few shortages still exist. These include sharpness and angles, and for that recommended ranges are specified during design, so that the recognition system fully functions.

VI. Rules

The recognition algorithm principally works as presented in Fig. 4. The algorithm starts by searching for neighboring faces that have the type upon search, then it checks a set of rules to make sure if this face is the right one or if it is one of many faces sharing the same characteristics. If the result indicates that it is the right face then it will proceed to the next face, and if the result does not indicate that it is the right face then it will not be included for this feature. Faces can include as many validators as needed to distinguish them from other faces. However, not all validators are applicable to all types of faces. Rules that have been used are built by using validators like parallelism, perpendicularity, concentricity, thickness, fully round, or face repeatedly.

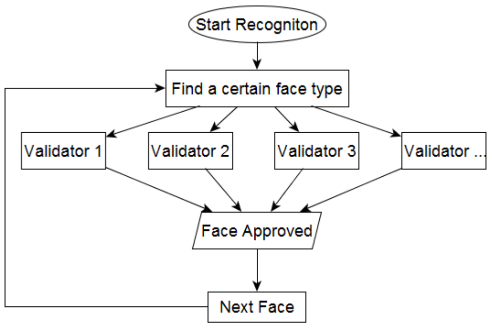


Fig. 4 The Feature Recognition algorithm

1. Parallelism: This will check if surfaces have a parallel relation. For plane it means that the normal (perpendicular) lines from the plane center are rather

parallel. For round shapes like cylinder or cone, parallelism is measured about the concentric lines that form those shapes. For surfaces like chamfers or fillets which are complex shapes because of having two radii, parallelism is measured about the central lines that form the rounding radii lines that directly form that shape, as shown in Fig. 5.

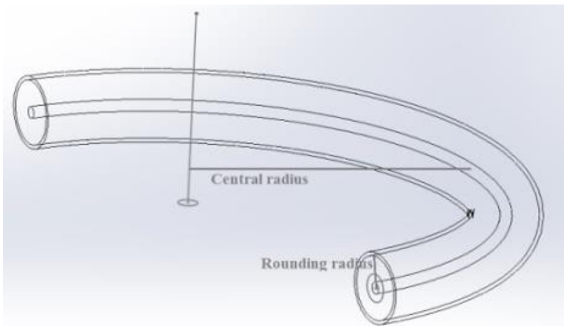


Fig. 5 For tours surfaces, parallelism is measured about the central radius

2. Perpendicularity: This rule is similar to parallelism rule, rather it measures if surfaces have 90 degrees in between. Also perpendicularity unlike parallelism is applicable in different dimensions, so it does not need to be on the same plane.
3. Concentricity: When bodies take one of the rounded forms, this rule does find the adjacent or correspondent faces by examining if they have the same center. By convention surfaces have to be parallel at first.
4. Thickness: Applying parallelism and perpendicularity rules will limit the number of found surfaces. However, when non-uniformity exists, parallelism and perpendicularity rules might be unable to deal with it, and hence we need the rule of thickness. The thickness rule takes a surface as a reference, then runs a test on all other surfaces, till it finds a surface with a match, i.e. a surface with constant thickness from the surface under consideration (reference surface).. This is done by taking into account that the formed body does have a constant thickness across its route. This rule plays a vital role with surfaces tagged as “Surface”, which do not have a direction and cannot be included under the normal categories.
5. Surface completion: When dealing with rounded bodies, it is imperative to know if the body is fully rounded or if it just takes a part of a rounded shape. This rule will examine that. The rule is useful to remove the confusion between rounding that any feature can have, e.g. fillets, chamfers, and features that may include a rounding pattern, e.g. holes.
6. Surface repeatedly: For the matter of recognizing bodies that are off-path, the route needs to split into two different routes. Recognition might continue with two or more routes until it reaches the end-state, or it may need to return to the main path again. For the latter case, this rule will test if the face it came across had been examined before i.e. if faces are identical, or the found face had not been examined before, i.e. a new face to examine.

VII. Modeling and Characterization Approach

Feature Modeling aims characterization of all considerable variations a certain feature can take. In Fig. 2 for instance, a simple extruded hole can have 16 possible variations if there are no special cases. To consider these variations, different steps have been taken to characterize the different models and their associated variations. As presented in Fig. 6, and explained below, feature modeling is done as follows:

1. Characterization: In this phase we consider only one feature at a time. Feature characterization has been simplified at first by means of using finite-state machines approach to represent all possible transitions a single feature can take. Details are beyond the scope of this work.
2. Implementation: After characterizing the feature logically in phase 1, phase 2 comes in place where the feature characterization’s logic is implemented by means of programming. This will be used in practice later for feature recognition.
3. Identification and Précising: Upon implementing the logic, we need to test it to make sure the feature under study is well recognized. We start with Identification, where we run the program against the feature in its simplest form. For example, if we are modeling Extruded Hole feature, then identification will be done against the model which has no fillets, chamfers, or any special forms. Upon successfulness in identifying the feature in its very basic form, we perform précising by running our program against models with different variations of the same feature. Upon recognition of the feature in its different variations, we proceed to phase 4. Otherwise, we return to phase 1 to modify the characterization logic, and then to phase 2 to modify the implementation.
4. Testing with other features: As mentioned previously, some features have similarities, which result in very close characterization logic. For instance, Louver feature in its simplest form has the same characterization logic as one of the variations of the Bridge feature. In the previous phases, we were considering only the feature under study. In this phase, we consider running the logic and the implementable program of this feature against models hosting other features. The purpose here is to double-check that the characterization logic works only with the feature under consideration, and to avoid any false identification of other features. By now, fully independent feature characterization is achieved.

For the matter of characterizing the different features, the previous four phases have been repeated to model and every feature individually. Now we have set of recognition algorithms and programs, in which each can recognize a certain feature successfully.

5. Nesting: In this phase, we aim identification of the nested structures. This is done by running the program against

the sub-structures of the main structure under study. For instance, a feature can host another feature in one of its faces, so after recognizing the main feature at first, we consider running the recognition algorithm on the feature's different faces themselves to search for any nested features within. To visualize this, in Fig. 2 Y, for instance, the main structure is a model hosting a Dimple feature, in which the Dimple feature's structure hosts Louver and Bridge features as sub-features, in which the Bridge feature's structure as well hosts an Extruded Hole feature as a third-level feature. Here we could recognize all features and the sub-features by running the recognition algorithm over the different structures. That is what we mean by finding nested features. Again, if required, then we may modify the characterization logic at phase 1 if the recognition algorithm would mix between the different features and their hosting ones.

Previous phases have been performed using standard testing models we created earlier for the purpose of testing only. In reality, this is not the case. Thus, further testing has been done with real manufacturing parts extracted from various customers, so that tolerance and flexibility could be adjusted properly to match with real-world parts for effectiveness.

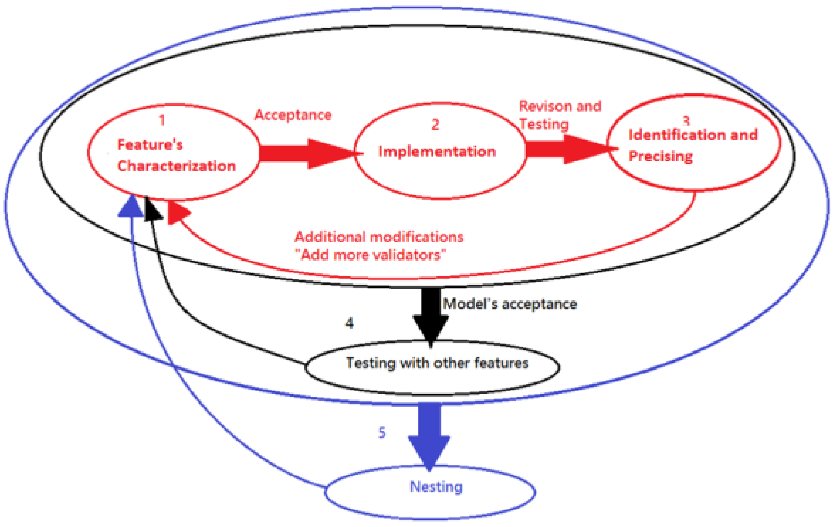

1
Feature's Characterization
Acceptance
2
Implementation
Revison and Testing
3
Identification and Precising
Additional modifications "Add more validators"
Model's acceptance
4
Testing with other features
5
Nesting

Fig. 6 Feature's modeling and characterization process

VIII. Case Study: A Louver Model

We are here introducing the work that has been done to model louver model. The recognition algorithm is given in Fig. 7, and details are as following:

1. Analyzing surfaces: A lover has mainly 3 bodies with 7 to 9 faces according to the design. As well it can have up to extra 6 faces if it includes fillets, torus or sphere shaped. The surface in between can be a planar or conical surface. If fillets exist, then their main body will take the cylindrical shape, and minor bodies will be either torus or complex surfaces entitled "surface".
2. Analyzing relations between surfaces:

a. Main cylindrical body: Main outer and inner cylindrical faces are parallel, concentric, perpendicular to the main face, have the same thickness, and are not fully rounded.

b. Fillets: Main surfaces are parallel, concentric, perpendicular to the main face, have the same thickness, and are not fully rounded. Main surfaces however are not concentric with the main outer and inner cylindrical faces of the main cylindrical body. For minor surfaces, if they are torus-shaped, then they will be parallel, parallel to the main face, concentric and have the same thickness. If sphere-shaped, then they are only concentric and have the same thickness. If surface-shaped, then only the thickness rule is applicable. All fillets are not fully rounded.

c. Planar faces: They are parallel to each other, and perpendicular to the main face.

3. Analysis of convexity:

a. To find the main body, the surface is "concave"

b. To find the main fillet, the surface is "smooth concave".

c. If a fillet is present, then to find the main body, the convexity rule searches for a surface with convexity type "inflection".

d. All other bodies are convex, except if there would be a fillet, then there would be inflection.

e. Reaching the end-state would be either convex or smooth convex if there is a fillet.

4. Technical results: Upon successful recognition, the system could recognize louver feature in a sheet of multiple features. It also gave details to its parameters, including exact position from the center, length, width, height, and thickness. Different bodies were tagged with their radius, face identity number, bending factor, corner rule, and the material for implementation. For the existing curvatures, the system indicates the type of convexity with the type of curve, e.g. line, circle, or non-uniform shape.

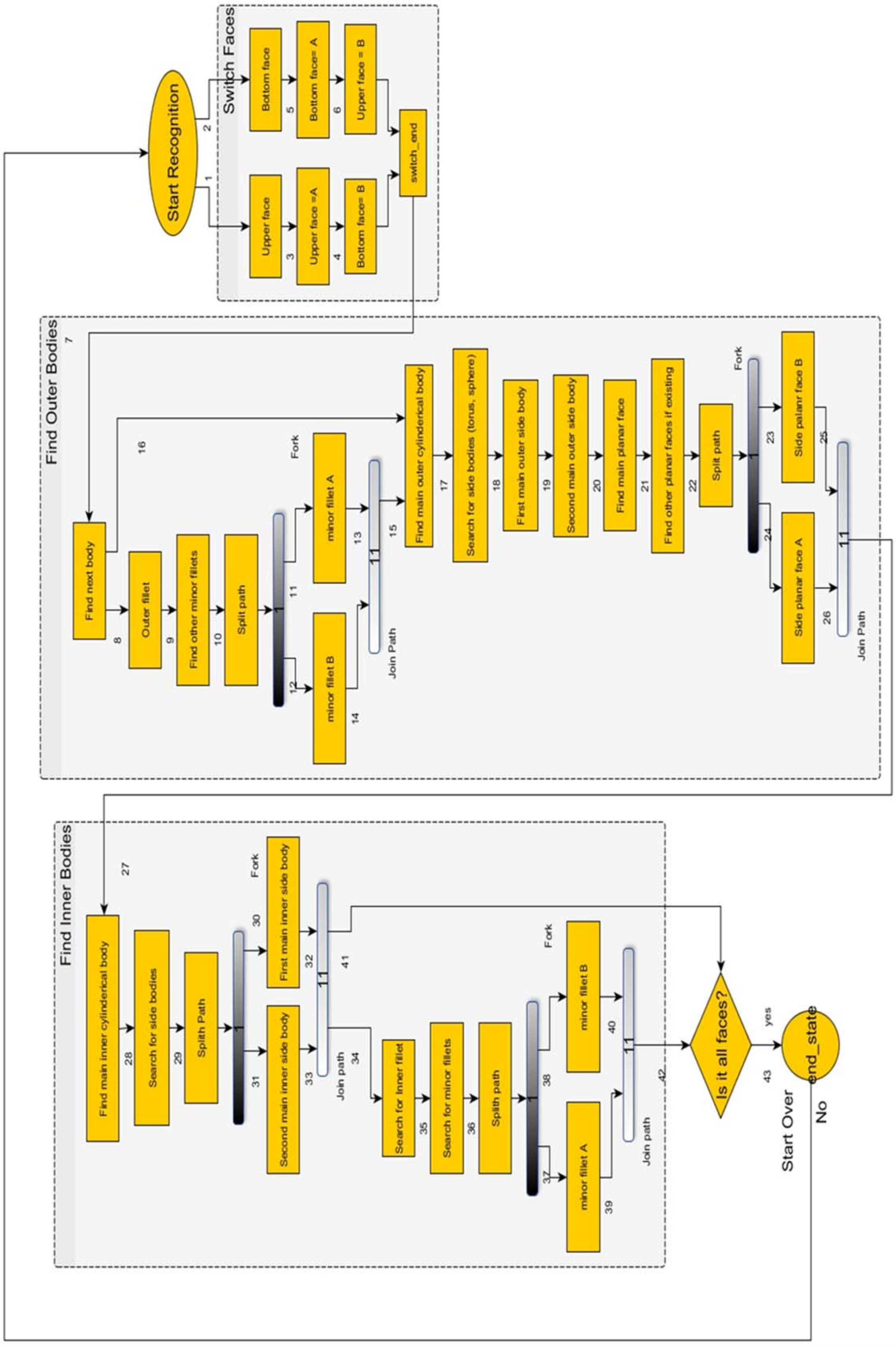
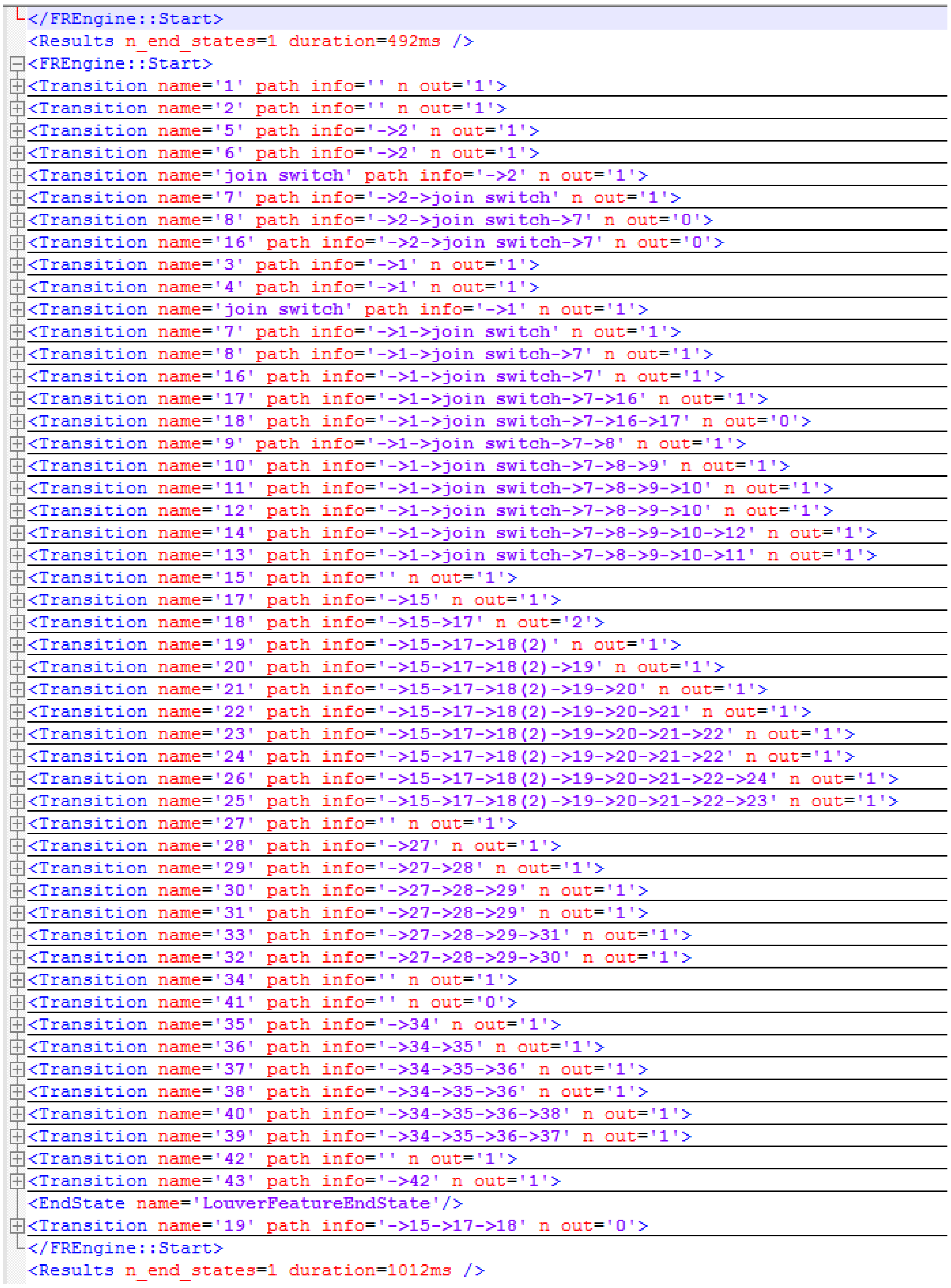


Fig. 7 Louver feature recognition algorithm

5. Technical details: In Fig. 8, the system writes down the found features and all possible combinations that might occur during the recognition phase. The system could make all required comparisons with the defined models, and already tried different paths for recognition from other different features, and successfully it could recognize the louver feature within 492 ms.



```
</FREngine::Start>
<Results n_end_states=1 duration=492ms />
<FREngine::Start>
<Transition name='1' path info='' n out='1'>
<Transition name='2' path info='' n out='1'>
<Transition name='5' path info='->2' n out='1'>
<Transition name='6' path info='->2' n out='1'>
<Transition name='join switch' path info='->2' n out='1'>
<Transition name='7' path info='->2->join switch' n out='1'>
<Transition name='8' path info='->2->join switch->7' n out='0'>
<Transition name='16' path info='->2->join switch->7' n out='0'>
<Transition name='3' path info='->1' n out='1'>
<Transition name='4' path info='->1' n out='1'>
<Transition name='join switch' path info='->1' n out='1'>
<Transition name='7' path info='->1->join switch' n out='1'>
<Transition name='8' path info='->1->join switch->7' n out='1'>
<Transition name='16' path info='->1->join switch->7' n out='1'>
<Transition name='17' path info='->1->join switch->7->16' n out='1'>
<Transition name='18' path info='->1->join switch->7->16->17' n out='0'>
<Transition name='9' path info='->1->join switch->7->8' n out='1'>
<Transition name='10' path info='->1->join switch->7->8->9' n out='1'>
<Transition name='11' path info='->1->join switch->7->8->9->10' n out='1'>
<Transition name='12' path info='->1->join switch->7->8->9->10' n out='1'>
<Transition name='14' path info='->1->join switch->7->8->9->10->12' n out='1'>
<Transition name='13' path info='->1->join switch->7->8->9->10->11' n out='1'>
<Transition name='15' path info='' n out='1'>
<Transition name='17' path info='->15' n out='1'>
<Transition name='18' path info='->15->17' n out='2'>
<Transition name='19' path info='->15->17->18(2)' n out='1'>
<Transition name='20' path info='->15->17->18(2)->19' n out='1'>
<Transition name='21' path info='->15->17->18(2)->19->20' n out='1'>
<Transition name='22' path info='->15->17->18(2)->19->20->21' n out='1'>
<Transition name='23' path info='->15->17->18(2)->19->20->21->22' n out='1'>
<Transition name='24' path info='->15->17->18(2)->19->20->21->22' n out='1'>
<Transition name='26' path info='->15->17->18(2)->19->20->21->22->24' n out='1'>
<Transition name='25' path info='->15->17->18(2)->19->20->21->22->23' n out='1'>
<Transition name='27' path info='' n out='1'>
<Transition name='28' path info='->27' n out='1'>
<Transition name='29' path info='->27->28' n out='1'>
<Transition name='30' path info='->27->28->29' n out='1'>
<Transition name='31' path info='->27->28->29' n out='1'>
<Transition name='33' path info='->27->28->29->31' n out='1'>
<Transition name='32' path info='->27->28->29->30' n out='1'>
<Transition name='34' path info='' n out='1'>
<Transition name='41' path info='' n out='0'>
<Transition name='35' path info='->34' n out='1'>
<Transition name='36' path info='->34->35' n out='1'>
<Transition name='37' path info='->34->35->36' n out='1'>
<Transition name='38' path info='->34->35->36' n out='1'>
<Transition name='40' path info='->34->35->36->38' n out='1'>
<Transition name='39' path info='->34->35->36->37' n out='1'>
<Transition name='42' path info='' n out='1'>
<Transition name='43' path info='->42' n out='1'>
<EndState name='LouverFeatureEndState'/>
<Transition name='19' path info='->15->17->18' n out='0'>
</FREngine::Start>
<Results n_end_states=1 duration=1012ms />
```

Fig. 8 All possible transitions, and found faces during the feature recognition process

It is worth mentioning that these tests have been done only for testing, and with the final solution's release version speed is expected to increase between 20 and 100 times.

IX. Conclusions

In this paper we introduced our current work with feature recognition for sheet metal parts. Here, we have developed a system for feature identification, done the required integration, designed recognition algorithms and gone through multiple phases of testing and optimization. At first, the presented system does interface with 3D CAD and CAM tools, and a high level of programming was done to ensure correct recognition without any flaws. For precision, we considered the different sub-features that could exist within the same feature, and we could handle that by performing a thorough classification to all features and grouping them into different patterns. Following that, we combined the different variations with their similar algorithms, so in some cases one algorithm might handle up to 512 sub-features of the main feature. Finally, we performed intensive testing with either ideal models, or real parts coming from customers. Currently the system can handle up to 11 different features which are the most common ones to exist, with recognition successfulness up to 100%. The recognition system at a later stage does interface with the tooling sub-system that takes all processed data about features, their locations and their properties, so that it can specify the tools required to perform a certain model or a feature within a model. In the current phase we are trying to include more features even if they are practically minor features that might not occur or by the best very rarely, and we are aiming at this phase to introduce a complete system that can handle all sheet metal features.

References

[1] Nnaji, Bartholomew O., et al. "Feature reasoning for sheet metal components." The International Journal of Production Research 29.9 (1991): 1867-1896.
[2] Farsi, Mohammad Ali, and Behrooz Arezoo. "Feature recognition and design advisory system for sheet metal components." Proceeding of the 5th IATS (2009).
[3] Huhtala, M., M. Lohtander, and J. Varis. "Product data management and sheet metal features—Sheet metal part recognition for an easier designing process producing manufacture-friendly products." 2013 IEEE International Conference on Industrial Engineering and Engineering Management. IEEE, 2013.
[4] Groover, Mikell, and E. W. J. R. Zimmers. CAD/CAM: computer-aided design and manufacturing. Pearson Education, 1983.
[5] Mercer, Tim. CAD/CAM selection for small manufacturing companies. Diss. University of Wisconsin, 2000.
[6] Justin, Mitchell. "Types of CAD Software." Techwalla. N.p., n.d. Web. 21 Dec. 2016. <https://www.techwalla.com/articles/types-of-cad-software>.
[7] Siemens, P. L. M. "Software Inc." NX Nastran User's Guide (2008).
[8] Cannon, Larissa, et al. "How can NX Advanced Simulation support multi-user design?" Comput.-Aided Des. Appl., PACE (2), Aug (2012): 21-32.
[9] FRE is a copyright and a registered property of Prima Power, Metallitie 4, FI-62200, Kauhava, Finland.
[10] Kannan, T. R., and M. S. Shunmugam. "Processing of 3D sheet metal components in STEP AP-203 format. Part I: feature recognition system." International Journal of Production Research 47.4 (2009): 941-964.
[11] Kannan, T. R., and M. S. Shunmugam. "Processing of 3D sheet metal components in STEP AP-203 format. Part II: feature reasoning system." International journal of production research 47.5 (2009): 1287-1308.
[12] Fournier, Sue. Sheet Metal Handbook. Vol. 575. Penguin, 1989.
[13] Nigam, Swami D., and Joshua U. Turner. "Review of statistical approaches to tolerance analysis." Computer-Aided Design 27.1 (1995): 6-15.